Package 'hydroMOPSO'

May 8, 2026

Type Package

Title Multi-Objective Optimisation with Focus on Environmental Models

Version 0.1-14

Author Rodrigo Marinao-Rivas [aut, cre, cph],
Mauricio Zambrano-Bigiarini [aut, ctb, cph] (ORCID:
<https://orcid.org/0000-0002-9536-643X>)

Maintainer Rodrigo Marinao-Rivas <ra.marinao.rivas@gmail.com>

Description State-of-the-art Multi-Objective Particle Swarm Optimiser (MOPSO), based on the algorithm developed by Lin et al. (2018) <doi:10.1109/TEVC.2016.2631279> with improvements described by Marinao-Rivas & Zambrano-Bigiarini (2020) <doi:10.1109/LA-CCI48322.2021.9769844>. This package is inspired by and closely follows the philosophy of the single objective 'hydroPSO' R package ((Zambrano-Bigiarini & Rojas, 2013) <doi:10.1016/j.envsoft.2013.01.004>), and can be used for global optimisation of non-smooth and non-linear R functions and R-base models (e.g., 'TUWmodel', 'GR4J', 'GR6J'). However, the main focus of 'hydroMOPSO' is optimising environmental and other real-world models that need to be run from the system console (e.g., 'SWAT+'). 'hydroMOPSO' communicates with the model to be optimised through its input and output files, without requiring modifying its source code. Thanks to its flexible design and the availability of several fine-tuning options, 'hydroMOPSO' can tackle a wide range of multi-objective optimisation problems (e.g., multi-objective functions, multiple model variables, multiple periods). Finally, 'hydroMOPSO' is designed to run on multi-core machines or network clusters, to alleviate the computational burden of complex models with long execution time.

License GPL (>= 2)

Depends R (>= 4.0.0)

Imports zoo, parallel, randtoolbox, lhs, hydroTSM, methods

Suggests knitr, rmarkdown, smoof, hydroGOF, airGR, TUWmodel

URL https://gitlab.com/rmarinao/hydroMOPSO

BugReports https://gitlab.com/rmarinao/hydroMOPSO/-/issues

LazyLoad yes

ByteCompile TRUE

NeedsCompilation no

Repository CRAN

Date/Publication 2025-06-18 08:10:02 UTC

Contents

`hydroMOPSO-package` *Multi-Objective Calibration of Hydrological Models using MOPSO*

Description

State-of-the-art Multi-Objective Particle Swarm Optimiser (MOPSO), based on the NMPSO algorithm developed by Lin et al. (2018), which in turn is based on the work by Coello and Lechuga (2002) and Kennedy and Eberhart (1995). To maintain diversity and accelerate convergence to the Pareto-optimal front (POF), NMPSO combines two search mechanism (Lin et al., 2015), these being a PSO search and the application of genetic operators. Lin et al. (2015) also included a balanceable fitness estimation (BFE) procedure to rank particles in a *external archive* (A), in order to provide an effective guidance to the true POF, while keeping diversity among particles. `hydroMOPSO` is developed with a focus on solving multi-objective problems with the least amount of function/model evaluations, taking the work done by Marinao-Rivas and Zambrano-Bigiarini (2021), who provided a default configuration of: i) the swarm size, ii) the maximum number of particles in the external archive, and iii) the maximum amount of genetic operations in the external archive.

This package and the attached tutorials have been developed with a special focus on the calibration of hydrological/environmental models and related real-world problems, but it can also be used for the calibration of any type of model that can be run from the command-line or the optimisation of certain functions to be defined by the user in the R environment.

For calibration problems, `hydroMOPSO` is model-independent, allowing the user to easily interface any computer simulation model with the calibration engine (NMPSO). Thus, the user only needs to indicate which model parameters will be modified and where, how and where to run the hydrological model, either in the R environment or from the system console; and finally how to read the model

results. With these aspects properly structured by the user, the algorithm it will take control over the model to be calibrated until a maximum number of iterations is reached.

Adittionally, this package provides intuitive plot summaries and detailed information about optimisation performance. These features make it easier to interpret and assess the multi-objective calibration results, particularly for non-experts. The package offers a comprehensive set of tools that streamline the calibration process, from model parameter estimation to result analysis. With these features, users can quickly identify the best-performing models and gain deeper insights into the underlying processes and mechanisms of hydrological systems.

The operating mechanism of `hydroMOPSO` is based on the `hydroPSO` R package developed by Zambrano-Bigiarini and Rojas (2013), inheriting the philosophy of flexibility that allows dealing with any calibration problem with different fine-tuning options, as well as taking advantage of multicore machines or network clusters to alleviate the computational burden of complex models with *long* execution time.

Details

Package:	hydroMOPSO
Type:	Package
Version:	0.1-3
Date:	2023-04-24
License:	GPL (>=2)
LazyLoad:	yes
Packaged:	2023-04-24; rmarinao
BuiltUnder:	R version 4.3.0 (2023-04-21) – "Already Tomorrow"; x86_64-pc-linux-gnu (64-bit)

Author(s)

Rodrigo Marinao-Rivas and Mauricio Zambrano-Bigiarini
Maintainer: Rodrigo Marinao-Rivas <ra.marinao.rivas@gmail.com>

References

Coello, C. A. C., & Lechuga, M. S. (2002). MOPSO: A proposal for multiple objective particle swarm optimization. Proceedings of the 2002 Congress on Evolutionary Computation, CEC 2002, 2, 1051-1056. doi:10.1109/CEC.2002.1004388

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. Proceedings of ICNN95 - International Conference on Neural Networks, 4, 1942-1948. doi:10.1109/ICNN.1995.488968

Lin, Q., Li, J., Du, Z., Chen, J., & Ming, Z. (2015). A novel multi-objective particle swarm optimization with multiple search strategies. European Journal of Operational Research, 247, 732-744. doi:10.1016/J.EJOR.2015.06.071

Lin, Q., Liu, S., Zhu, Q., Tang, C., Song, R., Chen, J., ... Zhang, J. (2016). Particle Swarm Optimization With a Balanceable Fitness Estimation for Many-Objective Optimization Problems. IEEE Transactions on Evolutionary Computation, 22, 32-46. doi:10.1109/TEVC.2016.2631279

Marinao-Rivas, R., & Zambrano-Bigiarini, M. (2021). Towards best default configuration settings for NMPSO in multi-objective optimization. 2021 IEEE Latin American Conference on Computational Intelligence, LA-CCI 2021. doi:10.1109/LA-CCI48322.2021.9769844

Zambrano-Bigiarini, M., & Rojas, R. (2013). A model-independent Particle Swarm Optimisation software for model calibration. Environmental Modelling & Software, 43, 5-25. doi:10.1016/j.envsoft.2013.01.004

See Also

https://CRAN.R-project.org/package=hydroPSO
https://CRAN.R-project.org/package=hydroGOF
https://CRAN.R-project.org/package=hydroTSM

`GR4JWrapperExamples` *Example wrapper functions to execute GR4J model*

Description

Example wrapper functions to execute the GR4J model and obtain the performance of two objective functions (KGE2012 and KGEGarcia), in a calibration (`GR4JExampleCal`) or a verification period (`GR4JExampleVer`). Keep in mind that, within `hydroMOPSO`, the calibration or verification wrapper functions essentially have to be **prepared by the user**, with the objective functions that are convenient and the output variables that are necessary.

Thus, the functions presented here are only intended to work with examples from the documentation and serve as a guide to: 1) The general scheme of the calibration/verification wrapper functions
2) The assimilation of mandatory inputs
3) The assimilation of mandatory outputs

Usage

```
GR4JExampleCal(param.values,
               Obs,
               Objs.names,
               var.names,
               var.units,
               full.period,
               warmup.period,
               cal.period,
               InputsModel,
               RunOptions,
               area)

GR4JExampleVer(param.values,
               Obs,
```

```
                Objs.names,
                var.names,
                var.units,
                full.period,
                warmup.period,
                cal.period,
                InputsModel,
                RunOptions,
                area)
```

Arguments

`param.values`	(`numeric`) Vector with parameter set of the model **This is a mandatory input for any wrapper function to work with hydroMOPSO, preserve name and class**
`Obs`	(`list`) List with time series of observations of the output variables **This is a mandatory input for any wrapper function to work with hydroMOPSO, preserve name and class**
`Objs.names`	(`character`) Vector with the names of the optimisation objectives **This is a mandatory input for any wrapper function to work with hydroMOPSO, preserve name and class**
`var.names`	(`character`) Vector with the names of the output variables **This is a mandatory input for any wrapper function to work with hydroMOPSO, preserve name and class**
`var.units`	(`character`) Vector with the units of measurement of the output variables **This is a mandatory input for any wrapper function to work with hydroMOPSO, preserve name and class**
`full.period`	(`Date`) Vector with the dates of the full period (warmup + calibration and/or verification) **This is a mandatory input for any wrapper function to work with hydroMOPSO, preserve name and class**
`warmup.period`	(`Date`) Vector with the dates of the warmup period **This is a mandatory input for any wrapper function to work with hydroMOPSO, preserve name and class**
`cal.period`	(`Date`) Vector with the dates of the calibration period **This is a mandatory input for any wrapper function to work with hydroMOPSO, preserve name and class**
`InputsModel`	(`list`) GR4J inputs structured with the function `airGR::CreateInputsModel`

This input has been included only for the execution of the GR4J model in particular

`RunOptions` (`list`)
GR4J run options specified with the function `airGR::CreateRunOptions`
This input has been included only for the execution of the GR4J model in particular

`area` (`numeric`)
Area of the basin (sq-m), necessary to pass the outlet streamflow to m3/s (cms)
This input has been included only for the execution of the GR4J model in particular

Value

(`list`)

The returned list contains two elements

Objs (`numeric`)
Vector with the numerical values of the objectives (GoF1 and GoF2).

sim (`list`)
List with as many elements as time series of the output variables of the model (in this case only one output variable: streamflows).

Author(s)

Rodrigo Marinao Rivas <ra.marinao.rivas@gmail.com>, Mauricio Zambrano-Bigiarini, <mzb.devel@gmail.com>

See Also

`hydroMOPSO`

`hydromod` *Definition and execution of the model to be optimised using an executable file that runs out of R (system console or external script)*

Description

It runs a user-defined model to be optimised and returns the output variables of the model requested by the user according to the number of functions indicated in the list `out.FUNs`. This specific function was designed to run an executable file from the system console

Usage

```
hydromod(param.values,
         param.files="ParamFiles.txt",
         param.ranges="ParamRanges.txt",
         model.drty=getwd(),
         exe.fname,
         exe.args = character(),
         stdout=FALSE,
         stderr="",
         verbose= FALSE,
         out.FUNs,
         out.FUNs.args
)
```

Arguments

param.values	(numeric) A numeric vector with the parameter set used to run the model specified in exe.fname.
param.files	character, file name (with full path) storing locations and names of the files that have to be modified for each parameter. By default param.files="ParamFiles.txt"
param.ranges	character, file name (with full path) storing the ranges (maximum and minum values) of the parameters to be used in the optimisation. By default param.files="ParamRanges.txt"
model.drty	(character) Path to the executable file of the model specified in exe.fname. ALL the files required to run the model have to be located within this directory (input files for the model may be located in a different directory, if properly referenced).
exe.fname	(character) Model command line arguments to be entered through a prompted string to execute the user-defined model.
exe.args	(character) Optional arguments to be passed in the command line to the user-defined model.
stdout	(logical or character) Where output to 'stdout' should be sent. Possible values are FALSE (discard output, the default), "", to the R console. See system2 By default stdout=FALSE and any message printed by the model code to the screen will be omitted. This setting is recommended when calibrating the model with hydroMOPSO. However, when trying to run the model code with hydromod by the first time, it is recommend to set stdout="", in order to detect if the model was properly executed or not.
stderr	(logical or character) Where output to 'stderr' should be sent. Possible values are FALSE (discard output, the default), "", to the R console. See system2 By default stderr="" and any error message of the model code will be printed to the screen

verbose	(logical) Indicate if progress messages are printed to the screen If verbose=TRUE, the following messages will appear: i) parameter values for each particle; (ii) model execution; iii) extraction of simulated values; and iv) computation of the goodness-of-fit measures
out.FUNs	(list) Name of valid R functions to read the model outputs and transform them into a (zoo) object (Should generally require at least basic use of read.table or read.csv. The list must have as many elements (names) as output variables of the model to read.
out.FUNs.args	(list) At a first level, each object inside this list corresponds to a set of arguments that, RESPECTIVELY, must be passed to out.FUNs. At a second level, the arguments to read each of the output variables are entered as lists (let's say sub-lists).

Value

A list with as many output variables (usually time series in zoo class) as functions listed in out.FUNs

Author(s)

Mauricio Zambrano-Bigiarini, <mzb.devel@gmail.com>, Rodrigo Marinao Rivas <ra.marinao.rivas@gmail.com>

See Also

hydroMOPSO

---

hydroMOPSO *Multi-Objective Particle Swarm Optimisation algorithm (NMPSO)*

Description

Multi-objective Particle Swarm Optimisation algorithm (NMPSO). The default configuration of hydroMOPSO has been adapted to obtain results with the fewest number of iterations possible.

Important: In Example 5 (calibration of GR4J hydrological model), maxit = 50 was set just for practical needs when testing the package. For acceptable results please change to maxit = 250. With any more robust model and with up to 12 parameters we recommend maxit = 1000.

Usage

```
hydroMOPSO(fn='hydromod',
           lower=-Inf,
           upper=Inf,
           control=list(),
           model.FUN=NULL,
```

```
model.FUN.args=list(),
BC.space.norm = FALSE,
obj.thr = NULL,
...)
```

Arguments

`fn` (`function` or `character`)
Object with the name of a valid R function to be optimised (minimised or maximised). When the goal is to optimise just simple functions (problems not associated with models with input data and output), it is possible to specify the name of any function correctly defined by the user. Special cases occur when the user is working with models, declared as internal or external functions of R. In these last cases, `fn='hydromod'` specifies that the optimisation is applied to a model that can be invoked from R(tipically, an executable file that must be run from the system console), but is executed entirely outside of this environment. On the other hand, `fn='hydromodInR'` specifies that the optimisation is applied to a model that can be executed within the R environment.
In detail:
-) When `fn!='hydromod'` & `fn!='hydromodInR'`, the first argument of `fn` has to be a vector of parameters over which optimisation is going to take place. It must return a vector with as many elements as objectives have been set in the function, and where each objective must be a scalar result. In this case, the algorithm uses the vector of values returned by `fn` as both model output and its corresponding set of optimised scalar results
-) When `fn=='hydromod'` the algorithm will optimise the R-external model defined by `model.FUN` and `model.FUN.args`, which are used to extract the values simulated by the model and to compute its corresponding goodness-of-fit measures.
-) When `fn=='hydromodInR'` the algorithm will optimise the R model defined by `model.FUN` and `model.args`, which are used to extract the values simulated by the model and to compute its corresponding goodness-of-fit measures.

When `fn=='hydromod' | fn=='hydromodInR'`, the function must return a list with two (2) specific elements, the first element of the list consists of the vector with as many elements as objectives have been established in the function, and where each objective must be a scalar result; the second element of the list corresponds to a matrix with the raw output data of the model that determines the scalar results of the objectives, for example time series of a hydrological model such as streamflow, evapotranspiration, soil moisture, among others. The matrix with the raw output data of the model must have as many columns as there are simulated variables being worked on in the optimisation, and this number of variables should not necessarily coincide with the number of objectives set. for example, flows could only be returned from a hydrological model to analyze three objectives.

`lower` (`numeric`)
Lower boundary for each parameter
In `hydroMOPSO` the length of `lower` and `upper` are used to defined the dimension

	of the solution space
upper	(numeric) Upper boundary for each parameter In hydroMOPSO the length of lower and upper are used to defined the dimension of the solution space
control	(list) A list of control parameters. See 'Details'
model.FUN	(character) (OPTIONAL) Used only when fn=='hydromod' | fn=='hydromodInR' A valid R function representing the model code to be optimised
model.FUN.args	(list) (OPTIONAL) Used only when fn=='hydromod' | fn=='hydromodInR' A list with the arguments to be passed to model.FUN
BC.space.norm	(logical) (OPTIONAL) Used only when fn=='hydromod' | fn=='hydromodInR' Indicates whether the space will be normalised when looking for a best compromise (BC) solution. For normalisation, strictly speaking it is necessary to specify the argument obj.thr with thresholds for each objective, but if these are not provided then the limits of the final Pareto optimal front (from the nadir and ideal point) will be used.
obj.thr	(list) (OPTIONAL) Used only when fn=='hydromod' | fn=='hydromodInR' Thresholds for each objective. These thresholds define a range for each objective, from the maximum/minimum acceptable value to the value denoting a perfect fit. The list has as many objects as there are targets in the calibration, and each one has two elements, the maximum and minimum threshold, e.g., list("NSE" = c(0,1), "KGE" = c(-0.41,1)). If these data are provided they will be used to normalise the objective space when looking for a best compromise (BC) solution.
...	further arguments to be passed to fn

Details

By default, hydroMOPSO performs minimisation on all objectives specified in fn (MinMax='min' in control list), but this can be changed to maximisation (MinMax='max' in control list). If in fn you have to maximise some objectives and minimise others, you must make them all point to the same direction (all maximising or all minimising), which can be handled simply with a sign (See Example 2 where this type of case is presented).
Although the NMPSO algorithm was formulated to deal with many objectives, the default definitions in hydroMOPSO, and therefore the applications made in research linked to this package, have been made with a focus on two and three objectives. Extending applications to optimisations with four or more objectives is possible with this package (so you are welcome to formulate such problems and solve them with hydroMOPSO!), but be very careful in analyzing your results.

The control argument is a list that can supply any of the following components:

drty.in (character)
(OPTIONAL) Used only when fn='hydromod'
Name of the directory storing the input files required for PSO, i.e. 'ParamRanges.txt' and 'ParamFiles.txt'.

drty.out (character)
Path to the directory storing the output files generated by hydroMOPSO.

param.ranges (character)
(OPTIONAL) Used only when fn=='hydromod' | fn=='hydromodInR'
Name of the file defining the minimum and maximum boundary values for each one of the parameters to be optimisated with NMPSO.

digits (numeric)
(OPTIONAL) Used only when write2disk=TRUE
Number of significant digits used for writing the output files with scientific notation.

digits.dom (numeric)
number of decimal places used in dominance check. Fewer decimal places (say, 16, 8, or 4, for example) may be necessary to prevent the algorithm from resulting in solutions that are nearly the same.
By default digits.dom=Inf, which basically means numbers are not rounded

MinMax (character)
Indicates whether a maximisation or minimisation multi-objetive problem needs to be solved. Valid values are in: c('min', 'max'). By default MinMax='min'. This control argument applies to all objective functions at the same time, so they must all go in the same direction (either all maximizing or all minimizing; keep in mind that for a particular function to go from maximizing to minimizing, or vice versa, it is only necessary add a minus sign (-)).

npart (numeric)
Number of particles in the swarm. By default npart=10, inherited from R-package hydroMOPSO.

maxrep (numeric)
Maximum number of particles to be stored in the updated Pareto Front in each iteration.
By default maxrep=100

maxcross (numeric)
Maximum number of Pareto Front particles that, for each iteration, perform the crossing and mutatiom in the application of genetic operators.
By default maxcross=50

maxit (numeric)
Maximum number of iterations.
By default maxit=1000

Xini.type (character)
Indicates how to initialise the particles' positions in the swarm within the ranges defined by lower and upper.
Valid values are:
-) Sobol: Sobol initialisation of positions, using npart number of samples contained in parameter space bounded by lower and upper. **It requires the randtoolbox package**
-) lhs: Latin Hypercube initialisation of positions, using npart number of strata to divide each parameter range. **It requires the lhs package**
-) random: random initialisation of positions within lower and upper
By default Xini.type='Sobol'

Vini.type (character)

Indicates how to initialise the particles' velocities in the swarm.
Valid values are:
-) zero: all the particles are initialised with zero velocity
-) random2011: random initialisation of velocities within lower-Xini and upper-Xini, as defined in SPSO 2011 ('Vini=U(lower-Xini, upper-Xini)') (see Clerc, 2012, 2010)
-) lhs2011: same as in random2011, but using a Latin Hypercube initialisation with npart number of strata instead of a random uniform distribution for each parameter. **It requires the lhs package**
-) random2007: random initialisation of velocities within lower and upper using the 'half-diff' method defined in SPSO 2007 ('Vini=[U(lower, upper)-Xini]/2') (see Clerc, 2012, 2010)
-) lhs2007: same as in random2007, but using a Latin Hypercube initialisation with npart number of strata instead of a random uniform distribution for each parameter. **It requires the lhs package**
By default Vini.type='zero'

boundary.wall (character)

Indicates the type of boundary condition to be applied during optimisation.
Valid values are: absorbing2011, absorbing2007, reflecting, damping, invisible
By default boundary.wall='absorbing2011'
Experience has shown that Clerc's constriction factor and the inertia weights do not always confine the particles within the solution space. To address this problem, Robinson and Rahmat-Samii (2004) and Huang and Mohan (2005) propose different boundary conditions, namely, reflecting, damping, absorbing and invisible to define how particles are treated when reaching the boundary of the searching space (see Robinson and Rahmat-Samii (2004) and Huang and Mohan (2005) for further details).

cal.hv (logical)

(OPTIONAL)
Indicates whether or not the hypervolume formed between the hyperplane of the Pareto Front and a nadir point designated as nadir.point will be calculated.
By default cal.hv=FALSE

nadir.point (numeric)

(OPTIONAL) Only required when cal.hv=TRUE
Nadir point from which the hypervolume will be calculated in each iteration step. It should correspond to a reference point considered as the worst acceptable optimal value.

n.samples (integer)

(OPTIONAL) Only required when cal.hv=TRUE
Number of points to estimate hypervolume, based on MonteCarlo sampling.
By default n.samples=10000

write2disk (logical)

Indicates if the output files will be written to the disk.
By default write2disk=TRUE

verbose (logical)

Indicates if progress messages are to be printed.
By default verbose=TRUE

plot (logical)

Indicates if a plot with the Pareto Front will be drawn after each iteration.

By default plot=FALSE

REPORT (integer)
(OPTIONAL) Used only when verbose=TRUE
The frequency of report messages printed to the screen.
By default REPORT=10

parallel (character)
Indicates how to parallelise 'hydroMOPSO' (to be precise, only the evaluation of the objective function fn is parallelised). Valid values are:
-)none: no parallelisation is made (this is the default value)
-)parallel: parallel computations for network clusters or machines with multiple cores or CPUs. A 'FORK' cluster is created with the makeForkCluster function. When fn.name='hydromod' the evaluation of the objective function fn is done with the clusterApply function of the **parallel** package. When fn.name != 'hydromod' the evaluation of the objective function fn is done with the parRapply function of the **parallel** package.
-)parallelWin: parallel computations for network clusters or machines with multiple cores or CPUs (this is the only parallel implementation that works on Windows machines). A 'PSOCK' cluster is created with the makeCluster function. When fn.name='hydromod' the evaluation of the objective function fn is done with the clusterApply function of the **parallel** package. When fn.name != 'hydromod' the evaluation of the objective function fn is done with the parRapply function of the **parallel** package.

par.nnodes (numeric)
(OPTIONAL) Used only when parallel!='none'
Indicates the number of cores/CPUs to be used in the local multi-core machine, or the number of nodes to be used in the network cluster.
By default par.nnodes is set to the amount of cores detected by the function detectCores() (**parallel** package)

par.pkgs (character)
(OPTIONAL) Used only when parallel='parallelWin'
List of package names (as characters) that need to be loaded on each node for allowing the objective function fn to be evaluated.

Value

(list)

The returned list contains elements that vary according to the input specifications

Rep (list)
Particle repository for the last iteration (just second last phase of NMPSO), detailing:

- *Position* (matrix)
Positions of each set of Pareto Front particles until the last iteration.
- *Objs* (matrix)
Objective values of each set of Pareto Front particles until the last iteration.
- *BFE* (numeric)
Balanceable Fitness Estimation (BFE) of each set of Pareto Front particles until the last iteration.

- *Ranking.BFE* (`matrix`)
Ranking of each set of Pareto Front particles until the last iteration, according to the BFE value.

MOPSOResults (`list`)
Particle repository history of all iterations (both phases of NMPSO), detailing:

- *ParetoFront* (`data.frame`)
History of objectives values of each Pareto Front particles in all iterations (both phases). In this `data.frame`, the first column indicates the iteration `Iter`; the second column the phase `Phase` (1 or 2); and the following columns are as many as objectives treated, being identified with the assigned name.
- *Particles_ParetoFront* (`data.frame`)
History of positions of each Pareto Front particles in all iterations (both phases). In this `data.frame`, the first column indicates the iteration `Iter`; the second column the phase `Phase` (1 or 2); then as many columns as objectives treated, being identified with the assigned name; and finally, as many columns as decision variables (parameters).
- *MaxMin* (`data.frame`)
Specification on whether the objectives are maximised or minimised.
- *ObjsNames* (`data.frame`)
Name of each of the objectives (`Obj1, Obj2, ...`).

hydroDetails (`list`)
(ONLY ADDED WHEN `fn=='hydromod' | fn=='hydromodInR'`)
Details about the modeling involved in optimisation:

- *Dimensions* (`data.frame`)
Number of objectives and number of output variables involved in the optimisation.
- *NamesAndUnitsVars* (`data.frame`)
Name and unit of measure of the output variables involved in the optimisation (`var1, var1_unit, var2, var2_unit, ...`).
- *Obs* (`list`)
Observed values of each of the variables involved in the optimisation, keeping in mind that the same format indicated as mandatory input data `Obs` within the `FUN` function is maintained.
- *WarmUp* (`data.frame`)
Time series indicating the warm-up period used in the optimisation.
- *DatesCal* (`data.frame`)
Time series indicating the calibration period used in the optimisation.

hydroResults (`list`)
(ONLY ADDED WHEN `fn=='hydromod' | fn=='hydromodInR'`)
Post-processed results about the modeling involved in optimisation:

- *ParticlesFull* (`data.frame`)
History of positions of each Pareto Front particles in all iterations. In this `data.frame`, the first column indicates the simulation number `Sim`, in ascending order from the first simulation (first iteration, phase 1) to the last simulation (last iteration, phase 2); then as many columns

as objectives treated, being identified with the assigned name; and finally, as many columns as decision variables (parameters).
- *FilledPOF* (data.frame)
Filled Pareto front, built from evaluating the dominance of the solutions of all the iterations performed in the optimisation. To prevent the filled Pareto Front from having too many solutions, the parameters and objective values are rounded according to input DigitsDom (number of decimal places). In this data.frame, the first column indicates the simulation number Sim; then as many columns as objectives treated, being identified with the assigned name.
- *ParticlesFilledPOF* (data.frame)
Perticles from filled Pareto Front. In this data.frame, the first column indicates the simulation number Sim; then as many columns as objectives treated, being identified with the assigned name; and finally, as many columns as decision variables (parameters).
- *ModelOut* (list)
Time series of the model output variables, for all solutions of the filled Pareto Front. This list has as many objects as output variables, and each one corresponds to an object of class zoo with as many columns as solutions of the filled Pareto Front.
- *ParticleBestCS* (data.frame)
Best compromise solution, i.e., the solution with the minimum Euclidean distance from the maximum values of each objective. data.frame with only one row and several columns: the first column indicates the simulation number Sim; then as many columns as objectives treated, being identified with the assigned name; and finally, as many columns as decision variables (parameters).
- *ModelOutBestCS* (list)
Time series of the model output variables, just for the best compromise solution. This list has as many objects as output variables, and each one corresponds to an object of class zoo with a single time serie.
- *ParticleBestObjs* (list)
Solutions that minimise/maximise each of the objectives. data.frame with only one row. In a first level, this list has as many objects as objectives involves in the optimisation, each one with a data.frame with only one row and several columns: the first column indicates the simulation number Sim; then as many columns as objectives treated, being identified with the assigned name; and finally, as many columns as decision variables (parameters).
- *ModelOutBestObjs* (list)
Time series of the model output variables, for the maximisation/minimisation of each objective. In a first level, this list has as many objects as objectives involves in the optimisation and, in a second level, each one corresponds to a list with as many objects as output variables, each one corresponding to an object of class zoo with a single time serie.
- *AnalysisPeriod* (character)
String indicating the analysis period, in this case "calibration".
- *DigitsDom* (numeric)
Number of decimal places used in dominance check. Fewer decimal places (say, 16, 8, or 4, for example) may be necessary to prevent the algorithm from resulting in solutions that are nearly the same.
- *ObjsNames* (data.frame)
Name of each of the objectives (Obj1, Obj2, ...).
- *MaxMin* (data.frame)
Specification on whether the objectives are maximised or minimised, must be in c("max", "min").

- *Obs* (`list`)
Observed values of each of the variables involved in the optimisation, keeping in mind that the same format indicated as mandatory input data `Obs` within the `FUN` function is maintained.
- *Dimensions* (`data.frame`)
Number of objectives and number of output variables involved in the optimisation.
- *NamesAndUnitsVars* (`data.frame`)
Name and unit of measure of the output variables involved in the optimisation (`var1, var1_unit, var2, var2_unit, ...`).
- *WarmUp* (`data.frame`)
Time series indicating the warm-up period used in the optimisation.
- *DatesCal* (`data.frame`)
Time series indicating the calibration period used in the optimisation.

Note

1) For a better understanding of the application cases in which `fn=='hydromod' | fn=='hydromodInR'`, it is strongly recommended to review the complementary tutorials to this package.

Author(s)

Rodrigo Marinao Rivas <ra.marinao.rivas@gmail.com>, Mauricio Zambrano-Bigiarini <mzb.devel@gmail.com>

References

Lin, Q., Liu, S., Zhu, Q., Tang, C., Song, R., Chen, J., Coello, C. A. C., Wong, K.-C., & Zhang, J. (2018). Particle Swarm Optimization With a Balanceable Fitness Estimation for Many-Objective Optimization Problems. IEEE Transactions on Evolutionary Computation, 22(1), 32-46. doi:10.1109/TEVC.2016.2631279

Marinao-Rivas, R., & Zambrano-Bigiarini, M. (2021). Towards best default configuration settings for NMPSO in Multiobjective Optimization. 2021 IEEE Latin American Conference on Computational Intelligence. (Accepted).

Zambrano-Bigiarini, M.; R. Rojas (2013), A model-independent Particle Swarm Optimization software for model calibration, Environmental Modelling & Software, 43, 5-25, doi:10.1016/j.envsoft.2013.01.004

Coello, C. A. C., & Lechuga, M. S. (2002). MOPSO: A proposal for multiple objective particle swarm optimization. Proceedings of the 2002 Congress on Evolutionary Computation, CEC 2002, 2, 1051-1056. doi:10.1109/CEC.2002.1004388

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. Proceedings of ICNN'95 - International Conference on Neural Networks, 4, 1942-1948. doi:10.1109/ICNN.1995.488968

Deb, K. (1999). Multi-objective genetic algorithms: problem difficulties and construction of test problems. Evolutionary computation, 7, 205-230. doi:10.1162/EVCO.1999.7.3.205

Kursawe, F. (1991). A variant of evolution strategies for vector optimization. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 496 LNCS, 193-197. doi:10.1007/BFB0029752

Deb, K., Thiele, L., Laumanns, M., & Zitzler, E. (2005). Scalable Test Problems for Evolutionary Multiobjective Optimization (bll 105-145; A. Abraham, L. Jain, & R. Goldberg, Reds). doi:10.1007/1-84628-137-7_6

Examples

```
#################################################################################################
# Example 1. Basic Benchmark function in minimisation
#################################################################################################

# This basic Benchmark function has two objectives (M = 2) in minimisation, its Pareto optimal
# front is discontinuous with four disconnected curves.This function works with 2 decision
# variables (D = 2).

# Main reference for function: Deb (1999)

library(hydroMOPSO)

lower <- c(0, 0)
upper <- c(1, 1)

fnBasic <- function(param){

   x1 <- param[1]
   x2 <- param[2]

   obj1 <- x1
   obj2 <- (1 + 10*x2)*(1-(x1/(1+10*x2))^2 - x1/(1+10*x2)*sin(2*pi*4*x1))

   out <- list(c(obj1, obj2)) # For consistency with further examples, this must be a list
   names(out) <- "Objs"       # The name "Objs" is a mandatory requirement

   return(out)
}

set.seed(100) # Setting the seed (for reproducible results)
out <- hydroMOPSO(fn  = fnBasic,
                  lower = lower,
                  upper = upper,
                  control=list(npart = 10, maxrep = 100, maxcross = 50,
                               MinMax = "min", maxit = 50, plot = TRUE)
                  )



#################################################################################################
# Example 2. Basic Benchmark function in maximisation
#################################################################################################
#
# This example is identical to Example 1, but the functions are in maximisation
#
# IMPORTANT:
# In the literature related to multi-objective optimisation, test functions are usually
# presented with minimisation objectives (such as the function used in Example 1). However,
# this does not necessarily always have to be formulated that way, especially when it comes to
# real-world applications.
#
```

```
# In this second example we just want to remind you that the disjunctive between maximising or
# minimising objectives is "a matter of signs".
#
# With this in account, as explained in the documentation, for hydroMOPSO operation there is
# one requirement which you MUST TAKE CARE OF:
#
# "The problems must be formulated in such a way that ALL objectives are either maximising or
# minimising. If your problem mixes both types of objectives, just add minus signs (-) in the
# results that require it..."
#
# Main reference for function: Kursawe (1991)

library(hydroMOPSO)

lower <- c(0, 0)
upper <- c(1, 1)


fnBasic <- function(param){

   x1 <- param[1]
   x2 <- param[2]

   obj1 <- -( x1 )
   obj2 <- -( (1 + 10*x2)*(1-(x1/(1+10*x2))^2 - x1/(1+10*x2)*sin(2*pi*4*x1)) )
   # note tha minus sign was added in obj1 and obj2

   out <- list(c(obj1, obj2)) # For consistency with further examples, this must be a list
   names(out) <- "Objs"       # The name "Objs" is a mandatory requirement

   return(out)
}

set.seed(100) # Setting the seed (for reproducible results)
out <- hydroMOPSO(fn  = fnBasic,
                  lower = lower,
                  upper = upper,
                  control=list(npart = 10, maxrep = 100, maxcross = 50,
                       MinMax = "max", maxit = 50, plot = TRUE) # note that now MinMax="max"
                  )




##############################################################################################
# Example 3. Using 'smoof' package: Kursawe function
##############################################################################################
#
# This Benchmark function has two objectives (M = 2), its Pareto optimal front is discontinuous
# and non-convex. For this example it will be implemented with 3 decision variables (D = 3)
#
# Main reference for function: Kursawe (1991)
```

```
library(hydroMOPSO)
library(smoof)

D <- 3
lower <- rep(-5,D)
upper <- rep(5,D)

Kursawe <- smoof::makeKursaweFunction(D) # using 'smoof' package

fnKursawe <- function(param){

   objs <- Kursawe(x = param)
   obj1 <- objs[1]
   obj2 <- objs[2]

   out <- list(c(obj1, obj2)) # For consistency with further examples, this must be a list
   names(out) <- "Objs"       # The name "Objs" is a mandatory requirement

   return(out)
}

set.seed(100) # Setting the seed (for reproducible results)
out <- hydroMOPSO(fn  = fnKursawe,
                  lower = lower,
                  upper = upper,
                  control=list(npart = 10, maxrep = 100, maxcross = 50,
                               MinMax = "min", maxit = 50, plot = TRUE)
                  )


#############################################################################################
# Example 4. Using 'smoof' package: DTLZ2 function with three objectives
#############################################################################################
#
# In this example, this Benchmark is formulated with two objectives (M = 3) and 12 decision
# variables (D = 12) its Pareto optimal front is concave.
#
# Main reference for function: Deb (2005)

library(hydroMOPSO)
library(smoof)

M <- 3
D <- 12
lower <- rep(0,D)
upper <- rep(1,D)

DTLZ2 <- smoof::makeDTLZ2Function(D, M) # using 'smoof' package

fnDTLZ2 <- function(param){

   objs <- DTLZ2(x = param)
   obj1 <- objs[1]
```

```
   obj2 <- objs[2]
   obj3 <- objs[3]

  out <- list(c(obj1, obj2, obj3)) # For consistency with further examples, this must be a list
   names(out) <- "Objs"       # The name "Objs" is a mandatory requirement

   return(out)
}

set.seed(100) # Setting the seed (for reproducible results)
out <- hydroMOPSO(fn  = fnDTLZ2,
                  lower = lower,
                  upper = upper,
                  control=list(npart = 10, maxrep = 100, maxcross = 50,
                               MinMax = "min", maxit = 50, plot = TRUE)
                  )

#############################################################################################
# Example 5. Calibration of GR4J hydrological model
#############################################################################################
#
# For this example, a "real-world" problem has been formulated: the calibration of a
# hydrological model
#
# In detail...
# Hydrological model: GR4J (Perrin et al., 2004)
# Number of parameters: four (X1, X2, X3, X4; see Perrin et al. (2004))
# Study area: Trancura River Basin (RTL)
# Input variables: Precipitation (pcp) and Potetntial EvapoTranspiration (pet)
# Calibration output variable: Streamflow (qobs)

library(hydroMOPSO)
library(airGR)
library(hydroTSM)
library(hydroGOF)

# RTL basin ------------------------------------------------
basin.area <- 1415025887 # basin area in square meters

# Load time series -----------------------------------------
data(Trancura9414001plus) # Load RTL data set

# Dates ----------------------------------------------------
dates.raw <- Trancura9414001plus[,"Date"]
dates <- as.Date(dates.raw) # dates

# INPUTS time series ---------------------------------------

# Precipitation (input variable)
ts.pcp.raw <- Trancura9414001plus[,"P_mm"]
ts.pcp <- zoo(ts.pcp.raw, dates)

# Potential EvapoTranspiration (input variable)
```

```
ts.pet.raw <- Trancura9414001plus[,"PET_mm"]
ts.pet <- zoo(ts.pet.raw, dates)

# OUTPUTS time series ---------------------------------------
# Observed streamflow (calibration output variable)
ts.qobs.raw <- Trancura9414001plus[,"Qobs_m3s"]
ts.qobs <- zoo(ts.qobs.raw, dates)

# Parameter ranges and noCal parameters --------------------

lower <- c("X1" = 0.01, "X2" = -100, "X3" = 0.01, "X4" = 0.5) # parameter range lower threshold
upper <- c("X1" = 1200, "X3" =  100, "X3" = 5000, "X4" = 5  ) # parameter range upper threshold

noCal.param <- (lower + upper)/2 # uncalibrated parameters

# Names and units of observed output variables -------------

char.obs.names <- "Streamflow"
char.obs.units <- "m3/s"

# Objectives names -----------------------------------------

char.objs.names <- c("KGE2012_Q", "KGEGarcia_Q")

# Calibration dates and subsetting -------------------------

WarmUpCal.dates <- dip("1979-01-01", "1979-12-31") # WarmUp for Calibration
Cal.dates <- dip("1980-01-01", "1999-12-31") # Calibration
FullCal.dates <- dip("1979-01-01", "1999-12-31") # WarmUp + Calibration

start.FullCal <- FullCal.dates[1]
end.FullCal   <- FullCal.dates[length(FullCal.dates)]

# INPUTS time series ---------------------------------------

# Precipitation (input variable)
ts.pcp.FullCal <- window(ts.pcp, start = start.FullCal, end = end.FullCal) # subsetting pcp

# Potential EvapoTranspiration (input variable)
ts.pet.FullCal <- window(ts.pet, start = start.FullCal, end = end.FullCal) # subsetting pet

# OUTPUTS time series ---------------------------------------

# Observed streamflow (calibration output variable)
ts.qobs.FullCal <- window(ts.qobs, start = start.FullCal, end = end.FullCal) # subsetting qobs

list.obs.Cal <- list(Q = ts.qobs.FullCal)

# Structuring Inputs and Options of GR4J model -------------

InputsModel.Cal <- CreateInputsModel(FUN_MOD= RunModel_GR4J,
                                     DatesR= as.POSIXlt(FullCal.dates),
                                     Precip= coredata(ts.pcp.FullCal),
```

```
                                    PotEvap= coredata(ts.pet.FullCal))

RunOptions.Cal <- CreateRunOptions(FUN_MOD= RunModel_GR4J, InputsModel= InputsModel.Cal,
                                   IndPeriod_Run = 1:length(FullCal.dates), warnings = FALSE)

# hydroMOPSO calibration ----------------------------------

set.seed(100) # Setting the seed (for reproducible results)
Cal.results <- hydroMOPSO(fn="hydromodInR",
                          lower=lower,
                          upper=upper,
                          control=list(MinMax="max", Xini.type = "lhs", npart=10,
                               maxit=50,  # for better results set maxit=250 in this case
                                      maxrep = 100, maxcross = 50,
                                      maxeval = 15000, write2disk = FALSE,  REPORT=1,
                                      digits = 8, plot = TRUE, parallel = "none"),
                          model.FUN="GR4JExampleCal",
                          model.FUN.args = list(Obs=list.obs.Cal, # mandatory
                                                Objs.names = char.objs.names, # mandatory
                                                 var.names = char.obs.names, # mandatory
                                                 var.units = char.obs.units, # mandatory
                                                 full.period = FullCal.dates, # mandatory
                                                 warmup.period  = WarmUpCal.dates,
                                                 cal.period = Cal.dates,
                                                 # Model specific inputs
                                             InputsModel = InputsModel.Cal, # model specific
                                               RunOptions = RunOptions.Cal, # model specific
                                                 area = basin.area # model specific
                                                 )
                          )
```

---

hydroVerification *Verification of a optimised model*

Description

It takes the optimisation results of a model and reruns the simulations in a verification period. Only applicable when the results of the previous optimisation were done with `fn=='hydromod' | fn=='hydromodInR'`

Usage

```
hydroVerification(Results,
                  fn = NULL,
                  control = list(),
                  model.FUN = NULL,
                  model.FUN.args = list())
```

Arguments

Results	(list) List with hydroMOPSO optimisation results. The details of this input are explained in the value returned by hydroMOPSO function
fn	(function or character) Object with the name of a valid R function to be optimised (minimised or maximised). When the goal is to optimise just simple functions (problems not associated with models with input and output data), it is possible to specify the name of any function correctly defined by the user. Special cases occur when the user is working with models, declared as internal or external functions of R. In these last cases, fn='hydromod' specifies that the optimisation is applied to a model that can be invoked from R(tipically, an executable file that must be run from the system console), but is executed entirely outside of this environment. On the other hand, fn='hydromodInR' specifies that the optimisation is applied to a model that can be executed within the R environment. In detail: -) When fn!='hydromod' & fn!='hydromodInR', the first argument of fn has to be a vector of parameters over which optimisation is going to take place. It must return a vector with as many elements as objectives have been set in the function, and where each objective must be a scalar result. In this case, the algorithm uses the vector of values returned by fn as both model output and its corresponding set of optimised scalar results -) When fn=='hydromod' the algorithm will optimise the R-external model defined by model.FUN and model.args, which are used to extract the values simulated by the model and to compute its corresponding goodness-of-fit measures. -) When fn=='hydromodInR' the algorithm will optimise the R model defined by model.FUN and model.args, which are used to extract the values simulated by the model and to compute its corresponding goodness-of-fit measures. When fn=='hydromod' | fn=='hydromodInR', the function must return a list with two (2) specific elements, the first element of the list consists of the vector with as many elements as objectives have been established in the function, and where each objective must be a scalar result; the second element of the list corresponds to a matrix with the raw output data of the model that determines the scalar results of the objectives, for example time series of a hydrological model such as streamflow, evapotranspiration, soil moisture, among others. The matrix with the raw output data of the model must have as many columns as there are simulated variables being worked on in the optimisation, and this number of variables should not necessarily coincide with the number of objectives set. for example, flows could only be returned from a hydrological model to analyze three objectives.
control	(list) A list of control parameters. See 'Details'
model.FUN	(character) (OPTIONAL) Used only when fn=='hydromod' | fn=='hydromodInR' A valid R function representing the model code to be optimised
model.FUN.args	(list)

(OPTIONAL) Used only when `fn=='hydromod' | fn=='hydromodInR'`
A list with the arguments to be passed to `model.FUN`

Value

(`list`)

ParticlesFull (`data.frame`)
History of positions of each Pareto Front particles in all iterations. In this `data.frame`, the first column indicates the simulation number `Sim`, in ascending order from the first simulation (first iteration, phase 1) to the last simulation (last iteration, phase 2); then as many columns as objectives treated, being identified with the assigned name; and finally, as many columns as decision variables (parameters).

FilledPOF (`data.frame`)
Filled Pareto front degraded in verification period. Keep in mind that strictly speaking this is not a Pareto Front since it is reached only by extending the solutions of the original front obtained by calibration to a verification period.

ParticlesFilledPOF (`data.frame`)
Perticles from filled Pareto Front. In this `data.frame`, the first column indicates the simulation number `Sim`; then as many columns as objectives treated, being identified with the assigned name; and finally, as many columns as decision variables (parameters). Note that in the objective columns the original calibration values have been replaced by those of the filled Pareto front degraded in verification period.

ModelOut (`list`)
Time series of the model output variables in verification period, for all solutions of the filled Pareto Front obtained in calibration period. This list has as many objects as output variables, and each one corresponds to an object of class zoo with as many columns as solutions of the filled Pareto Front.

ParticleBestCS (`data.frame`)
Best compromise solution, i.e., the solution with the minimum Euclidean distance from the maximum values of each objective, in calibration period. data.frame with only one row and several columns: the first column indicates the simulation number `Sim`; then as many columns as objectives treated, being identified with the assigned name; and finally, as many columns as decision variables (parameters). Note that in the objective columns the original calibration values have been replaced by those obtained in the verification period.

ModelOutBestCS (`list`)
Time series of the model output variables in verification period, just for the best compromise solution obtained in calibration period. This list has as many objects as output variables, and each one corresponds to an object of class zoo with a single time serie.

ParticleBestObjs (list)
Solutions that minimise/maximise each of the objectives, obtained in calibration period. data.frame with only one row. In a first level, this list has as many objects as objectives involves in the optimisation, each one with a data.frame with only one row and several columns: the first column indicates the simulation number Sim; then as many columns as objectives treated, being identified with the assigned name; and finally, as many columns as decision variables (parameters). Note that in the objective columns the original calibration values have been replaced by those obtained in the verification period.

ModelOutBestObjs (list)
Time series of the model output variables in verification period, for the maximisation/minimisation of each objective in calibration. In a first level, this list has as many objects as objectives involves in the optimisation and, in a second level, each one corresponds to a list with as many objects as output variables, each one corresponding to an object of class zoo with a single time serie.

AnalysisPeriod (character)
String indicating the analysis period, in this case "verification".

DigitsDom (numeric)
Number of decimal places used in dominance check. Fewer decimal places (say, 16, 8, or 4, for example) may be necessary to prevent the algorithm from resulting in solutions that are nearly the same.

ObjsNames (data.frame)
Name of each of the objectives (Obj1, Obj2, ...).

MaxMin (data.frame)
Specification on whether the objectives are maximised or minimised, must be in c("max", "min").

Obs (list)
Observed values of each of the variables involved in the optimisation, but now of the verification period. Keep in mind that the same format indicated as mandatory input data Obs within the FUN function is maintained.

Dimensions (data.frame)
Number of objectives and number of output variables involved in the optimisation.

NamesAndUnitsVars (data.frame)
Name and unit of measure of the output variables involved in the optimisation (var1, var1_unit, var2, var2_unit, ...).

WarmUp (data.frame)
Time series indicating the warm-up period used in the optimisation.

DatesCal (data.frame)
Time series indicating the calibration period used in the optimisation.

Note

1) The intended workflow is that first you must have the results of the optimisation done with the hydroMOPSO function, which are then entered into this function (hydroVerification)

Author(s)

Rodrigo Marinao Rivas <ra.marinao.rivas@gmail.com>, Mauricio Zambrano-Bigiarini, <mzb.devel@gmail.com>

See Also

hydroMOPSO

---

plot_out *Plotting observed vs. simulated data and uncertainty bands*

Description

The function plot_out takes the optimization/calibration or verification results of a hydrological model and generates the following plots:

(*): When do.png == TRUE, graphics are written to disk.
(**): Multiple graphs are generated corresponding to the targets specified in the optimization process.

1. **ModelOut_BCS_from_Pareto_Optimal_Front_vs_Obs**: A graphical comparison of time series for observed and simulated variables, using the simulation of the best compromise solution.

2. **ModelOut_from_Pareto_Optimal_Front**: An uncertainty band that encompasses the simulations given by all the solutions on the Pareto Optimal Front, displayed in a time series. The time series of the best compromise solution is also distinguished.

3. **ModelOut_from_Pareto_Optimal_Front_vs_Obs**: A graphical comparison of time series for observed and simulated variables, using an uncertainty band that encompasses the simulations given by all solutions on the Pareto Optimal Front.

Usage

```
plot_out(Results,
         model.out = NULL,
         analysis.period = NULL,
         model.out.bcs = NULL,
         bcs = NULL,
         obs.var = NULL,
         dimensions = NULL,
         obj.names = NULL,
         dates.cal = NULL,
         dates.warmup = NULL,
         var.names = NULL,
         var.units = NULL,
         xlim = NULL,
         ylim = NULL,
         digits = 4,
         col.band = "skyblue",
         col.bcs = "mediumblue",
         col.obs = "black",
         lwd = 0.75,
         pch.bcs = 15,
         pch.obs = 15,
         main = "study case #1",
         drty.out = "MOPSO.out",
         cex.pt = 0.25,
         cex.main = 1,
         cex.lab = 1,
         cex.axis = 1,
         do.png = FALSE,
         legend.obs = "Observation",
         legend.bcs = "Best compromise solution",
         legend.band = "Pareto front bands")
```

Arguments

`Results`	(`list`) Object containing preprocessed hydrological results.
`model.out`	(`list` or `NULL`) Output from the hydrological model used for evaluation.
`analysis.period`	(`character` or `NULL`) Time period for analysis (e.g., "calibration" or "verification").
`model.out.bcs`	(`list` or `NULL`) Model output representing the "Best Compromise Solution" (BCS).
`bcs`	(`matrix` or `NULL`) Parameters or results corresponding to the best compromise solution (BCS).

obs.var (list or NULL)
Observed variables to be compared against the model outputs.

dimensions (matrix or NULL)
Dimensions of the modeled problem, typically indicating the number of objectives and variables.

obj.names (character or NULL)
Names of the objectives.

dates.cal (Date or NULL)
Dates of the calibration period.

dates.warmup (Date or NULL)
Dates of the model's warm-up period.

var.names (character or NULL)
Names of the modeled and observed variables.

var.units (character or NULL)
Units of the modeled and observed variables.

xlim (numeric or NULL)
Limits for the x-axis in the plots.

ylim (numeric or NULL)
Limits for the y-axis in the plots.

digits (integer)
Number of digits to use for rounding values in plots and legends.

col.band (character)
Color used for the model uncertainty bands in the plots.

col.bcs (character)
Color used for the line representing the best compromise solution (BCS).

col.obs (character)
Color used for the observed variable lines in the plots.

lwd (numeric)
Line width used in the plots for model and observation lines.

pch.bcs (integer)
Symbol type for points in the plot representing the best compromise solution (BCS).

pch.obs (integer)
Symbol type for points in the plot representing the observations.

main (character)
Main title for the plot.

drty.out (character)
Output directory where plots will be saved if specified to save as PNG files.

cex.pt (numeric)
Size of points in the plots.

cex.main (numeric)
Size of the main title text in the plot.

cex.lab	(numeric) Size of the axis label text in the plots.
cex.axis	(numeric) Size of the axis values text in the plots.
do.png	(logical) Boolean value indicating whether the plots should be saved as PNG files.
legend.obs	(character) Legend text for observations.
legend.bcs	(character) Legend text for the best compromise solution (BCS).
legend.band	(character) Legend text for Pareto front bands.

Value

No return value; generates plots.

Author(s)

Rodrigo Marinao Rivas <ra.marinao.rivas@gmail.com>, Mauricio Zambrano-Bigiarini, <mzb.devel@gmail.com>

See Also

hydroMOPSO

---

plot_param *Plotting parameter with boxplots and dotty plots*

Description

The function plot_param generates plots to visualize parameter sensitivity and distribution in the context of multi-objective optimization using hydrological models. It helps in understanding how different parameters affect the model's performance and the trade-offs involved.

(*): When do.png == TRUE, graphics are written to disk.

Usage

```
plot_param(Results,
           legend.param = NULL,
           col = NULL,
           col.param = NULL,
           col.lines = NULL,
           name.param = NULL,
           lwd = 2,
```

```
            main = "study case #1",
            drty.out = "MOPSO.out",
            cex.pt = 1,
            cex.main = 1,
            cex.lab = 1,
            cex.axis = 1,
            cex.leg = 1,
            do.png = FALSE)
```

Arguments

`Results`	(`list`) Object containing preprocessed hydrological results.
`legend.param`	(`character` or `NULL`) Legend text for the parameters in the plots.
`col`	(`character` or `NULL`) Colors used for points in the parameter dotty plots.
`col.param`	(`character` or `NULL`) Specific colors for lines or points representing parameters in the parameter boxplots.
`col.lines`	(`character` or `NULL`) Specific colors for lines in the parameter boxplots.
`name.param`	(`character` or `NULL`) Custom names for the parameters to be plotted.
`lwd`	(`numeric`) Line width for plotting parameters in boxplots.
`main`	(`character`) Main title for the plot.
`drty.out`	(`character`) Output directory where plots will be saved if specified to save as PNG files.
`cex.pt`	(`numeric`) Size of points in the plots.
`cex.main`	(`numeric`) Size of the main title text in the plot.
`cex.lab`	(`numeric`) Size of the axis label text in the plots.
`cex.axis`	(`numeric`) Size of the axis values text in the plots.
`cex.leg`	(`numeric`) Size of the legend text in the plots.
`do.png`	(`logical`) Boolean value indicating whether the plots should be saved as PNG files.

Value

No return value; generates plots as a side effect.

Author(s)

Rodrigo Marinao Rivas <ra.marinao.rivas@gmail.com>, Mauricio Zambrano-Bigiarini, <mzb.devel@gmail.com>

See Also

plot_out, plot_pof

---

plot_pof *Plotting Pareto-optimal Fronts and Best Compromise Solutions*

Description

The function plot_pof generates plots of the Pareto-optimal front (POF) and the best compromise solution (BCS) in multi-objective optimization for hydrological models. It visualizes trade-offs between different objectives and helps identify the most balanced solution, assisting in model calibration and evaluation processes.

(*): When do.png == TRUE, graphics are written to disk.

Usage

```
plot_pof(Results,
         pof = NULL,
         bcs = NULL,
         analysis.period = NULL,
         dimensions = NULL,
         maxmin = NULL,
         obj.thr = NULL,
         obj.names = NULL,
         main = "study case #1",
         drty.out = "MOPSO.out",
         pch.pof = 21,
         pch.bcs = 21,
         col.pof = "#f21b1b",
         col.bcs = "#004fcf",
       legend.pof = c("Pareto-optimal front solutions", "Best compromise solution"),
         cex.pt = 1.25,
         cex.main = 1,
         cex.lab = 1,
         cex.axis = 1,
         do.png = FALSE)
```

Arguments

`Results`	(`list`) Object containing preprocessed hydrological results.
`pof`	(`matrix` or `NULL`) Dataset representing the filled Pareto-optimal front (POF) solutions.
`bcs`	(`matrix` or `NULL`) Parameters or results corresponding to the best compromise solution (BCS).
`analysis.period`	(`character` or `NULL`) Time period for analysis (e.g., "calibration" or "verification").
`dimensions`	(`numeric` or `NULL`) Dimensions of the modeled problem, typically indicating the number of objectives and variables.
`maxmin`	(`character` or `NULL`) Indicator of whether objectives are to be maximized ("max") or minimized ("min").
`obj.thr`	(`numeric` or `NULL`) Objective thresholds.
`obj.names`	(`character` or `NULL`) Names of the objectives used in the plot.
`main`	(`character`) Main title for the plot.
`drty.out`	(`character`) Output directory where plots will be saved if specified to save as PNG files.
`pch.pof`	(`integer`) Symbol type for points in the plot representing the Pareto-optimal front solutions.
`pch.bcs`	(`integer`) Symbol type for points in the plot representing the best compromise solution (BCS).
`col.pof`	(`character`) Color used for the points representing the Pareto-optimal front solutions in the plots.
`col.bcs`	(`character`) Color used for the points representing the best compromise solution in the plots.
`legend.pof`	(`character`) Legend text for the Pareto-optimal front and best compromise solution.
`cex.pt`	(`numeric`) Size of points in the plots.
`cex.main`	(`numeric`) Size of the main title text in the plot.
`cex.lab`	(`numeric`) Size of the axis label text in the plots.

cex.axis	(numeric) Size of the axis values text in the plots.
do.png	(logical) Boolean value indicating whether the plots should be saved as PNG files.

Value

No return value; generates plots as a side effect.

Author(s)

Rodrigo Marinao Rivas <ra.marinao.rivas@gmail.com>, Mauricio Zambrano-Bigiarini, <mzb.devel@gmail.com>

See Also

plot_out, plot_param

---

plot_results *Master Function for Plotting Hydrological Model Results*

Description

The function plot_results consolidates the plotting capabilities for hydrological model outputs, Pareto-optimal front solutions, and parameter sensitivity analyses. It calls three subordinate functions (plot_out, plot_pof, plot_param) to generate the required plots, offering a comprehensive visualization tool for model evaluation and optimization results.

(*): When do.png == TRUE, graphics are written to disk.

Usage

```
plot_results(Results,
             model.out = NULL,
             analysis.period = NULL,
             model.out.bcs = NULL,
             bcs = NULL,
             obs.var = NULL,
             dimensions = NULL,
             obj.names = NULL,
             dates.cal = NULL,
             dates.warmup = NULL,
             var.names = NULL,
             var.units = NULL,
             xlim = NULL,
             ylim = NULL,
             digits = 4,
```

```
                col.band = "skyblue",
                col.bcs = "mediumblue",
                col.obs = "black",
                lwd = 0.75,
                pch.bcs = 15,
                pch.obs = 15,
                main = "study case #1",
                drty.out = "MOPSO.out",
                cex.pt.out = 0.25,
                cex.pt.pof = 1.25,
                cex.pt.param = 1,
                cex.main = 1,
                cex.lab = 1,
                cex.axis = 1,
                do.png = FALSE,
                legend.obs = "Observation",
                legend.bcs = "Best compromise solution",
                legend.band = "Pareto front bands",
                pof = NULL,
                maxmin = NULL,
                obj.thr = NULL,
                pch.pof = 21,
                col.pof = "#f21b1b",
                legend.pof = c("Pareto-optimal front solutions",
                               "Best compromise solution"),
                legend.param = NULL,
                col = NULL,
                col.param = NULL,
                col.lines = NULL,
                name.param = NULL,
                cex.leg = 1)
```

Arguments

`Results`	(`list`) Object containing preprocessed hydrological results.
`model.out`	(`list` or `NULL`) Output from the hydrological model used for evaluation.
`analysis.period`	(`character` or `NULL`) Time period for analysis (e.g., "calibration" or "verification").
`model.out.bcs`	(`list` or `NULL`) Model output representing the "Best Compromise Solution" (BCS).
`bcs`	(`matrix` or `NULL`) Parameters or results corresponding to the best compromise solution (BCS).
`obs.var`	(`list` or `NULL`) Observed variables to be compared against the model outputs.

`dimensions` (`matrix` or `NULL`)
Dimensions of the modeled problem, typically indicating the number of objectives and variables.

`obj.names` (`character` or `NULL`)
Names of the objectives.

`dates.cal` (`Date` or `NULL`)
Dates of the calibration period.

`dates.warmup` (`Date` or `NULL`)
Dates of the model's warm-up period.

`var.names` (`character` or `NULL`)
Names of the modeled and observed variables.

`var.units` (`character` or `NULL`)
Units of the modeled and observed variables.

`xlim` (`numeric` or `NULL`)
Limits for the x-axis in the plots.

`ylim` (`numeric` or `NULL`)
Limits for the y-axis in the plots.

`digits` (`integer`)
Number of digits to use for rounding values in plots and legends.

`col.band` (`character`)
Color used for the model uncertainty bands in the plots.

`col.bcs` (`character`)
Color used for the line representing the best compromise solution (BCS).

`col.obs` (`character`)
Color used for the observed variable lines in the plots.

`lwd` (`numeric`)
Line width used in the plots for model and observation lines.

`pch.bcs` (`integer`)
Symbol type for points in the plot representing the best compromise solution (BCS).

`pch.obs` (`integer`)
Symbol type for points in the plot representing the observations.

`main` (`character`)
Main title for the plot.

`drty.out` (`character`)
Output directory where plots will be saved if specified to save as PNG files.

`cex.pt.out` (`numeric`)
Size of points in the "out" plots.

`cex.pt.pof` (`numeric`)
Size of points in the "pof" plots.

`cex.pt.param` (`numeric`)
Size of points in the "param" plots.

`cex.main` (numeric)
Size of the main title text in the plot.

`cex.lab` (numeric)
Size of the axis label text in the plots.

`cex.axis` (numeric)
Size of the axis values text in the plots.

`do.png` (logical)
Boolean value indicating whether the plots should be saved as PNG files.

`legend.obs` (character)
Legend text for observations.

`legend.bcs` (character)
Legend text for the best compromise solution (BCS).

`legend.band` (character)
Legend text for Pareto front bands.

`pof` (matrix or NULL)
Dataset representing the filled Pareto-optimal front (POF) solutions.

`maxmin` (character or NULL)
Indicator of whether objectives are to be maximized ("max") or minimized ("min").

`obj.thr` (numeric or NULL)
Objective thresholds.

`pch.pof` (integer)
Symbol type for points in the plot representing the Pareto-optimal front solutions.

`col.pof` (character)
Color used for the points representing the Pareto-optimal front solutions in the plots.

`legend.pof` (character)
Legend text for the Pareto-optimal front and best compromise solution.

`legend.param` (character or NULL)
Legend text for the parameters in the plots.

`col` (character or NULL)
Colors used for points in the parameter dotty plots.

`col.param` (character or NULL)
Specific colors for lines or points representing parameters in the parameter boxplots.

`col.lines` (character or NULL)
Specific colors for lines in the parameter boxplots.

`name.param` (character or NULL)
Custom names for the parameters to be plotted.

`cex.leg` (numeric)
Size of the legend text in the plots.

Value

No return value; generates plots as a side effect.

Author(s)

Rodrigo Marinao Rivas <ra.marinao.rivas@gmail.com>, Mauricio Zambrano-Bigiarini, <mzb.devel@gmail.com>

See Also

plot_out, plot_pof, plot_param

---

read_results *Reading the output files of a optimised model*

Description

Read results saved on disk from an optimization with hydroMOPSO. This feature only applies when fn is in c("hydromod", "hydromodInR")

Usage

```
read_results(fn = NULL,
             control = list(),
             model.FUN = NULL,
             model.FUN.args = list()
             )
```

Arguments

fn	(function or character) Object with the name of a valid R function to be optimised (minimised or maximised). When the goal is to optimise just simple functions (problems not associated with models with input and output data), it is possible to specify the name of any function correctly defined by the user. Special cases occur when the user is working with models, declared as internal or external functions of R. In these last cases, fn='hydromod' specifies that the optimisation is applied to a model that can be invoked from R(tipically, an executable file that must be run from the system console), but is executed entirely outside of this environment. On the other hand, fn='hydromodInR' specifies that the optimisation is applied to a model that can be executed within the R environment. In detail: -) When fn!='hydromod' & fn!='hydromodInR', the first argument of fn has to be a vector of parameters over which optimisation is going to take place. It must return a vector with as many elements as objectives have been set in the function, and where each objective must be a scalar result. In this case, the algorithm uses the vector of values returned by fn as both model output and its corresponding set of optimised scalar results

-) When `fn=='hydromod'` the algorithm will optimise the R-external model defined by `model.FUN` and `model.args`, which are used to extract the values simulated by the model and to compute its corresponding goodness-of-fit measures.
-) When `fn=='hydromodInR'` the algorithm will optimise the R model defined by `model.FUN` and `model.args`, which are used to extract the values simulated by the model and to compute its corresponding goodness-of-fit measures.

When `fn=='hydromod' | fn=='hydromodInR'`, the function must return a list with two (2) specific elements, the first element of the list consists of the vector with as many elements as objectives have been established in the function, and where each objective must be a scalar result; the second element of the list corresponds to a matrix with the raw output data of the model that determines the scalar results of the objectives, for example time series of a hydrological model such as streamflow, evapotranspiration, soil moisture, among others. The matrix with the raw output data of the model must have as many columns as there are simulated variables being worked on in the optimisation, and this number of variables should not necessarily coincide with the number of objectives set. for example, flows could only be returned from a hydrological model to analyze three objectives.

`control` (`list`)
A list of control parameters. See 'Details'

`model.FUN` (`character`)
(OPTIONAL) Used only when `fn=='hydromod' | fn=='hydromodInR'`
A valid R function representing the model code to be optimised

`model.FUN.args` (`list`)
(OPTIONAL) Used only when `fn=='hydromod' | fn=='hydromodInR'`
A list with the arguments to be passed to `model.FUN`

Details

The `control` argument is a list that can supply any of the following components:

drty.in (`character`)
(OPTIONAL) Used only when `fn='hydromod'`
Name of the directory storing the input files required for PSO, i.e. 'ParamRanges.txt' and 'ParamFiles.txt'.

drty.out (`character`)
Path to the directory storing the output files generated by hydroMOPSO.

digits (`numeric`)
(OPTIONAL) Used only when `write2disk=TRUE`
Number of significant digits used for writing the output files with scientific notation.

digits.dom (`numeric`)
Number of decimal places used in dominance check. Fewer decimal places (say, 16, 8, or 4, for example) may be necessary to prevent the algorithm from resulting in solutions that are nearly the same.
By default `digits.dom=Inf`, which basically means numbers are not rounded

write2disk (logical)
Indicates if the output files will be written to the disk.
By default write2disk=TRUE

verbose (logical)
Indicates if progress messages are to be printed.
By default verbose=TRUE

REPORT (integer)
(OPTIONAL) Used only when verbose=TRUE
The frequency of report messages printed to the screen.
By default REPORT=10

parallel (character)
Indicates how to parallelise 'hydroMOPSO' (to be precise, only the evaluation of the objective function fn is parallelised). Valid values are:
-)none: no parallelisation is made (this is the default value)
-)parallel: parallel computations for network clusters or machines with multiple cores or CPUs. A 'FORK' cluster is created with the makeForkCluster function. When fn.name='hydromod' the evaluation of the objective function fn is done with the clusterApply function of the **parallel** package. When fn.name != 'hydromod' the evaluation of the objective function fn is done with the parRapply function of the **parallel** package.
-)parallelWin: parallel computations for network clusters or machines with multiple cores or CPUs (this is the only parallel implementation that works on Windows machines). A 'PSOCK' cluster is created with the makeCluster function. When fn.name='hydromod' the evaluation of the objective function fn is done with the clusterApply function of the **parallel** package. When fn.name != 'hydromod' the evaluation of the objective function fn is done with the parRapply function of the **parallel** package.

par.nnodes (numeric)
(OPTIONAL) Used only when parallel!='none'
Indicates the number of cores/CPUs to be used in the local multi-core machine, or the number of nodes to be used in the network cluster.
By default par.nnodes is set to the amount of cores detected by the function detectCores() (**parallel** package)

par.pkgs (character)
(OPTIONAL) Used only when parallel='parallelWin'
List of package names (as characters) that need to be loaded on each node for allowing the objective function fn to be evaluated.

Value

(list)

MOPSOResults (list)
Particle repository history of all iterations (both phases of NMPSO), detailing:

- *ParetoFront* (data.frame)
History of objectives values of each Pareto Front particles in all iterations (both phases). In this data.frame, the first column indicates the iteration Iter; the second column the phase

Phase (1 or 2); and the following columns are as many as objectives treated, being identified with the assigned name.
- *Particles_ParetoFront* (data.frame)
History of positions of each Pareto Front particles in all iterations (both phases). In this data.frame, the first column indicates the iteration Iter; the second column the phase Phase (1 or 2); then as many columns as objectives treated, being identified with the assigned name; and finally, as many columns as decision variables (parameters).
- *MaxMin* (data.frame)
Specification on whether the objectives are maximised or minimised.
- *ObjsNames* (data.frame)
Name of each of the objectives (Obj1, Obj2, ...).

hydroDetails (list)
(ONLY ADDED WHEN fn=='hydromod' | fn=='hydromodInR')
Details about the modeling involved in optimisation:

- *Dimensions* (data.frame)
Number of objectives and number of output variables involved in the optimisation.
- *NamesAndUnitsVars* (data.frame)
Name and unit of measure of the output variables involved in the optimisation (var1, var1_unit, var2, var2_unit, ...).
- *Obs* (list)
Observed values of each of the variables involved in the optimisation, keeping in mind that the same format indicated as mandatory input data Obs within the FUN function is maintained.
- *WarmUp* (data.frame)
Time series indicating the warm-up period used in the optimisation.
- *DatesCal* (data.frame)
Time series indicating the calibration period used in the optimisation.

hydroResults (list)
(ONLY ADDED WHEN fn=='hydromod' | fn=='hydromodInR')
Post-processed results about the modeling involved in optimisation:

- *ParticlesFull* (data.frame)
History of positions of each Pareto Front particles in all iterations. In this data.frame, the first column indicates the simulation number Sim, in ascending order from the first simulation (first iteration, phase 1) to the last simulation (last iteration, phase 2); then as many columns as objectives treated, being identified with the assigned name; and finally, as many columns as decision variables (parameters).
- *FilledPOF* (data.frame)
Filled Pareto front, built from evaluating the dominance of the solutions of all the iterations performed in the optimisation. To prevent the filled Pareto Front from having too many solutions, the parameters and objective values are rounded according to input DigitsDom (number of decimal places). In this data.frame, the first column indicates the simulation number Sim; then as many columns as objectives treated, being identified with the assigned name.
- *ParticlesFilledPOF* (data.frame)
Perticles from filled Pareto Front. In this data.frame, the first column indicates the simulation number Sim; then as many columns as objectives treated, being identified with the

assigned name; and finally, as many columns as decision variables (parameters).
- *ModelOut* (list)
Time series of the model output variables, for all solutions of the filled Pareto Front. This list has as many objects as output variables, and each one corresponds to an object of class zoo with as many columns as solutions of the filled Pareto Front.
- *ParticleBestCS* (data.frame)
Best compromise solution, i.e., the solution with the minimum Euclidean distance from the maximum values of each objective. data.frame with only one row and several columns: the first column indicates the simulation number Sim; then as many columns as objectives treated, being identified with the assigned name; and finally, as many columns as decision variables (parameters).
- *ModelOutBestCS* (list)
Time series of the model output variables, just for the best compromise solution. This list has as many objects as output variables, and each one corresponds to an object of class zoo with a single time serie.
- *ParticleBestObjs* (list)
Solutions that minimise/maximise each of the objectives. data.frame with only one row. In a first level, this list has as many objects as objectives involves in the optimisation, each one with a data.frame with only one row and several columns: the first column indicates the simulation number Sim; then as many columns as objectives treated, being identified with the assigned name; and finally, as many columns as decision variables (parameters).
- *ModelOutBestObjs* (list)
Time series of the model output variables, for the maximisation/minimisation of each objective. In a first level, this list has as many objects as objectives involves in the optimisation and, in a second level, each one corresponds to a list with as many objects as output variables, each one corresponding to an object of class zoo with a single time serie.
- *AnalysisPeriod* (character)
String indicating the analysis period, in this case "calibration".
- *DigitsDom* (numeric)
Number of decimal places used in dominance check. Fewer decimal places (say, 16, 8, or 4, for example) may be necessary to prevent the algorithm from resulting in solutions that are nearly the same.
- *ObjsNames* (data.frame)
Name of each of the objectives (Obj1, Obj2, ...).
- *MaxMin* (data.frame)
Specification on whether the objectives are maximised or minimised, must be in c("max", "min").
- *Obs* (list)
Observed values of each of the variables involved in the optimisation, keeping in mind that the same format indicated as mandatory input data Obs within the FUN function is maintained.
- *Dimensions* (data.frame)
Number of objectives and number of output variables involved in the optimisation.
- *NamesAndUnitsVars* (data.frame)
Name and unit of measure of the output variables involved in the optimisation (var1, var1_unit, var2, var2_unit, ...).
- *WarmUp* (data.frame)
Time series indicating the warm-up period used in the optimisation.
- *DatesCal* (data.frame)

Time series indicating the calibration period used in the optimisation.

Note

1) The intended workflow is that first you must have the results of the optimisation done with the hydroMOPSO function, having saved the results to disk (write2disk=TRUE in hydroMOPSO)
2) Based on the previous point, the user must ensure that the input arguments fn, control, model.FUN and model.FUN.args that are entered in the hydroMOPSO and read_results functions must be EXACTLY THE SAME

Author(s)

Rodrigo Marinao Rivas <ra.marinao.rivas@gmail.com>, Mauricio Zambrano-Bigiarini, <mzb.devel@gmail.com>

See Also

hydroMOPSO

SimVsObs *Comparison between observed and simulated variables*

Description

Simple comparison between time series of observed and simulated variables. This function is specially designed to check the correct operation of the wrapper functions prepared by the user. The premise is as follows: if the user can generate a 'nice' graph (graphically evidencing the simulated and observed values, and obtaining finite numerical values for the objectives) then they can proceed with greater confidence to the hydroMOPSO optimisation step.

Usage

```
SimVsObs(sim,
         obs,
         obj.values,
         obj.names,
         var.names,
         var.units,
         legend.sim = "Simulated",
         legend.obs = "Observed",
         cal.period,
         warmup.period,
         full.period,
         xlim = NULL,
         ylim = NULL,
         main = "study case #1",
         analysis.period,
```

```
          digits.round = 8,
          col.obs = "black",
          col.sim = "mediumblue",
          lwd = 0.75,
          pch.obs = 15,
          cex.pt = 0.25,
          cex.main = 1,
          cex.lab = 1,
          cex.axis = 1,
          cex.leg = 1)
```

Arguments

sim	(list) List with simulations of the output variables involved in the optimisation. The list must have as many elements as variables involved, each as a zoo class.
obs	(list) List with observations of the output variables involved in the optimisation. The list must have as many elements as variables involved, each as a zoo class.
obj.values	(numeric) Vector with values of the objectives considered in the optimisation.
obj.names	(character) Vector with the names of the optimisation objectives.
var.names	(character) Vector with the names of the output variables.
var.units	(character) Vector with the units of measurement of the output variables.
legend.sim	(character) Single character with an identifiable name for the simulated values in the output graph. By default legend.sim = "Simulated".
legend.obs	(character) Single character with an identifiable name for the observed values in the output graph. By default legend.sim = "Observed".
warmup.period	(Date) Vector with the dates of the warmup period.
cal.period	(Date) Vector with the dates of the calibration period.
full.period	(Date) Vector with the dates of the full period (warmup + calibration and/or verification).
xlim	(code) ToDo

ylim (code)
ToDo

main (character)
Title for the plot, usually an identifiable name of the case study.
By default main = "study case #1".

analysis.period
(character)
The graph to be plotted is in verification or calibration.

digits.round (numeric)
Number of decimal places to round Objs.values. By default digits.round = 8

col.obs (character)
Colour to identify the time series of observed values obs.
By default col.obs = "black".

col.sim (character)
Colour to identify the time series of simulated values obs.
By default col.obs = "mediumblue".

lwd (numeric)
Line width for time series sim and obs.
By default lwd = 1.

pch.obs (numeric)
Plotting symbol (pch) specification.
By default pch.obs = 15.

cex.pt (numeric)
Expansion factor for the plotting symbol.
By default cex.pt = 1.

cex.main (numeric)
Expansion factor for the main title.
By default cex.main = 1.

cex.lab (numeric)
Expansion factor for x and y labels.
By default cex.lab = 1.

cex.axis (numeric)
Expansion factor for the axis annotation.
By default cex.axis = 1.

cex.leg (numeric)
Expansion factor for the text legend.
By default cex.leg = 1.

Value

No return value

Author(s)

Rodrigo Marinao Rivas <ra.marinao.rivas@gmail.com>, Mauricio Zambrano-Bigiarini, <mzb.devel@gmail.com>

See Also

hydroMOPSO

Examples

```
#################################################################################################

# This example is derived from example 5 of the hydroMOPSO function

library(hydroMOPSO)
library(airGR)
library(hydroTSM)
library(hydroGOF)

# RTL basin ------------------------------------------------
basin.area <- 1415025887 # basin area in square meters

# Load time series -----------------------------------------
data(Trancura9414001plus) # Load RTL data set

# Dates ----------------------------------------------------
dates.raw <- Trancura9414001plus[,"Date"]
dates <- as.Date(dates.raw) # dates

# INPUTS time series ---------------------------------------

# Precipitation (input variable)
ts.pcp.raw <- Trancura9414001plus[,"P_mm"]
ts.pcp <- zoo(ts.pcp.raw, dates)

# Potential EvapoTranspiration (input variable)
ts.pet.raw <- Trancura9414001plus[,"PET_mm"]
ts.pet <- zoo(ts.pet.raw, dates)

# OUTPUTS time series --------------------------------------
# Observed streamflow (calibration output variable)
ts.qobs.raw <- Trancura9414001plus[,"Qobs_m3s"]
ts.qobs <- zoo(ts.qobs.raw, dates)

# Parameter ranges and noCal parameters --------------------

lower <- c("X1" = 0.01, "X2" = -100, "X3" = 0.01, "X4" = 0.5) # parameter range lower threshold
upper <- c("X1" = 1200, "X3" =  100, "X3" = 5000, "X4" = 5  ) # parameter range upper threshold

noCal.param <- (lower + upper)/2 # uncalibrated parameters

# Names and units of observed output variables -------------

char.obs.names <- "Streamflow"
char.obs.units <- "m3/s"

# Objectives names -----------------------------------------
```

```
char.objs.names <- c("KGE2012_Q", "KGEGarcia_Q")

# Calibration dates and subsetting -------------------------

WarmUpCal.dates <- dip("1979-01-01", "1979-12-31") # WarmUp for Calibration
Cal.dates <- dip("1980-01-01", "1999-12-31") # Calibration
FullCal.dates <- dip("1979-01-01", "1999-12-31") # WarmUp + Calibration

start.FullCal <- FullCal.dates[1]
end.FullCal   <- FullCal.dates[length(FullCal.dates)]

# INPUTS time series ---------------------------------------

# Precipitation (input variable)
ts.pcp.FullCal <- window(ts.pcp, start = start.FullCal, end = end.FullCal) # subsetting pcp

# Potential EvapoTranspiration (input variable)
ts.pet.FullCal <- window(ts.pet, start = start.FullCal, end = end.FullCal) # subsetting pet

# OUTPUTS time series --------------------------------------

# Observed streamflow (calibration output variable)
ts.qobs.FullCal <- window(ts.qobs, start = start.FullCal, end = end.FullCal) # subsetting qobs

list.obs.Cal <- list(Q = ts.qobs.FullCal)

# Structuring Inputs and Options of GR4J model -------------

InputsModel.Cal <- CreateInputsModel(FUN_MOD= RunModel_GR4J,
                                     DatesR= as.POSIXlt(FullCal.dates),
                                     Precip= coredata(ts.pcp.FullCal),
                                     PotEvap= coredata(ts.pet.FullCal))

RunOptions.Cal <- CreateRunOptions(FUN_MOD= RunModel_GR4J, InputsModel= InputsModel.Cal,
                                   IndPeriod_Run = 1:length(FullCal.dates), warnings = FALSE)

# Checking Wrapper function in calibration -----------------

noCal.results.Cal <- GR4JExampleCal(param.values = noCal.param,
                                    Obs = list.obs.Cal,
                                    Objs.names = char.objs.names,
                                    var.names = char.obs.names,
                                    var.units = char.obs.units,
                                    warmup.period  = WarmUpCal.dates,
                                    cal.period = Cal.dates,
                                    full.period = FullCal.dates,
                                    InputsModel = InputsModel.Cal,
                                    RunOptions = RunOptions.Cal,
                                    area = basin.area)

noCal.sim.Cal <- noCal.results.Cal[["sim"]]
noCal.objs.Cal <- noCal.results.Cal[["Objs"]]
```

```
dev.new()
SimVsObs(sim = noCal.sim.Cal, obs = list.obs.Cal,
         obj.values = noCal.objs.Cal, obj.names = char.objs.names,
         var.names = char.obs.names, var.units = char.obs.units,
         legend.sim = "Simulated", legend.obs = "Observed",
       warmup.period  = WarmUpCal.dates, cal.period = Cal.dates, full.period = FullCal.dates,
        main = "...just checking GR4JExampleCal function", analysis.period = "calibration",
         digits.round = 4)
```

`SpecificValueInFile` *Straightforward modification of a value in input text file*

Description

This function provides the capability to modify input text files directly within the R environment, requering only minimal instructions. It was conceived to facilitate users to execute the calibration procedure via `hydroMOPSO()`, while staying entirely within the R environment. Thus, it precludes the need for manually editing text files, eliminating undesirable coordination that could potentially lead to errors.

Usage

```
SpecificValueInFile(modlist)
```

Arguments

`modlist`	(`list`) A list with as many objects as modifications to be made in the text files. In turn, each object is a list with the specifications of the modification to be made. See 'Details' for the contents of each list.

Details

The `modlist` argument is a list with an indeterminate number of objects, which only depends on the number of modifications the user needs to make, for example, being something like `modlist=list(mod_1, mod_2, mod_3,mod_4)`. Each of the '`mod`' objects in this list **must** provide the following items:

ParamID (`character`)
: The ID of the parameter to be modified.

newvalue (`numeric`)
: Numeric value to be written into the text file.

filename (`character`)
: Name of the text file that will be modified.

row (`numeric`)
: Row number in `filename` where `newvalue` will be written.

col.ini (numeric)
Starting column number in filename where newvalue is going to be written.

col.fin (numeric)
Ending column number in filename where newvalue is going to be written.

decimals (numeric)
Number of decimal places used to write newvalue into filename.

Value

No return value

Author(s)

Rodrigo Marinao Rivas <ra.marinao.rivas@gmail.com>, Mauricio Zambrano-Bigiarini, <mzb.devel@gmail.com>

See Also

hydromod

Trancura9414001plus *Hydrometeorological time series for Trancura antes de Llafenco River Basin*

Description

Daily time series of precipitation, air temperature (max, min, mean), potential evapotranspiration and streamflows for the catchment draining into the 'Trancura antes de Llafenco' streamflow station (Cod.BNA: 9414001, drainage area= 1416 km2), Araucania Region, Chile (Lat:-39.3333, Lon:-71.6667), with data from 01/Jan/1979 to 31/Dec/2020 (including some gaps in streamflow data).

Usage

```
data(Trancura9414001plus)
```

Format

zoo with seven columns:
-) *Dates*: character with the date (YYYY-MM-DD) for each daily observation.
-) *Pp_mm*: Spatially-averaged mean daily values of precipitation computed based on the CR2met dataset, [mm/day].
-) *Tmax_degC*: Spatially-averaged mean daily values of maximum air temperature computed based on the CR2met dataset, [degree Celsius].
-) *Tmin_degC*: Spatially-averaged mean daily values of minimum air temperature computed based on the CR2met dataset, [degree Celsius].
-) *Tmean_degC*: Spatially-averaged mean daily values of mean air temperature computed based on

the CR2met dataset, [degree Celsius].
-) *PET_mm*: Spatially-averaged mean daily values of potential evapotranspiration (PET), computed with the Hargreaves-Samani equation based on daily maximum and minimum air temperatures obtained from the CR2met dataset, [mm/day].
-) *Qobs_m3s*: Daily streamflows measured at the Trancura antes de Llafenco (9414001) station.
-) *ETobs_mm*: Daily evapotranspiration (ET) estimated at the Trancura antes de Llafenco (9414001) with the 8-day evapotranspiration product PML v2 (Zhang, 2019). The 8-day time series are disaggregated on a daily scale just dividing by 8, so, it is recommended to use this information on a weekly, monthly or annual scale.

Source

CR2met v2 is a gridded product of observed daily precipitation an maximum/minimim temperature, covering the period 1979-01-01 to 2020-12-31. Its developed by Boisier et al. (2018) and provided by Center for Climate and Resilience Research, Universidad de Chile, Santiago, Chile (https://zenodo.org/records/7529682, last accessed [Dic 2023]).
PML v2 is a gridded product of estimated 8-day evapotranspiration, covering the period 2020-03-01 to 2020-04-30. The proper use of this information as "observed values" can be a subject of discussion, however, they are included in `hydroMOPSO` for mere didactic purposes, hoping that future research will provide more reliable information to use as ET observations.
These data are intended to be used for research purposes only, being distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY.

References

Boisier, J. P., Alvarez-Garreton, C., Cepeda, J., Osses, A., Vasquez, N., and Rondanelli, R. (2018). CR2MET: A high-resolution precipitation and temperature dataset for hydroclimatic research in Chile. EGUGA, 20, 19739. Opgehaal van https://ui.adsabs.harvard.edu/abs/2018EGUGA..2019739B/abstract

Zhang, Y., Kong, D., Gan, R., Chiew, F. H. S., McVicar, T. R., Zhang, Q., & Yang, Y. (2019). Coupled estimation of 500 m and 8-day resolution global evapotranspiration and gross primary production in 2002-2017. Remote Sensing of Environment, 222, 165-182. doi:10.1016/J.RSE.2018.12.031

Index