Package 'hydroPSO'

March 30, 2012

Type Package

Title Model-Independent Particle Swarm Optimisation for Environmental Models

Version 0.1-54-1

Date 2012-03-30

Author Mauricio Zambrano-Bigiarini [aut, cre] and Rodrigo Rojas [ctb]

Author@R c(person(``Mauricio", ``Zambrano-Bigiarini", email = ``mzb.devel@gmail.com", role=c(``aut", ``cre") ),person(``Rodrigo", ``Rojas", email = ``Rodrigo.RojasMujica@gmail.com", role=c(``ctb")))

Maintainer Mauricio Zambrano-Bigiarini <mauricio.zambrano@gmail.com>

Description This package implements a state-of-the-art version of the Particle Swarm Optimisation (PSO) algorithm, with a special focus on the calibration of environmental models. hydroPSO is model-independent, allowing the user to easily interface any model code with the calibration engine (PSO), and includes a series of controlling options and PSO variants to fine-tune the performance of the calibration engine. An advanced sensitivity analysis function together with user-friendly plotting summaries facilitate the interpretation and assessment of the calibration results. Bugs reports/comments/questions are very welcomed.

License GPL (>=2)

Depends R (>= 2.10.0)

Imports Hmisc, sp, lattice, grid

Suggests hydroGOF, hydroTSM, lhs, zoo, multicore, scatterplot3d,vioplot

URL http://www.rforge.net/hydroPSO, http://cran.r-project.org/web/packages/hydroPSO

#Classification
optimisation, optimization, calibration, environment,environmental sciences, hydrology, PSO

LazyLoad yes

ByteCompile TRUE

R topics documented:

`hydroPSO-package` *A flexible and model-independent Particle Swarm Optimisation (PSO) package for calibration/optimisation of environmental models*

Description

hydroPSO is a package implementing an enhanced version of the canonical Particle Swarm Optimisation (PSO) algorithm developed by Kennedy and Eberhart (1995) and Eberhart and Kennedy (1995). PSO is a population-based stochastic optimisation technique inspired by social behaviour of bird flocking and shares few similarities with other evolutionary optimisation techniques such as Genetic Algorithms (GA). In PSO, however, the multi-dimensional solution space is explored on the basis of individual and global best-known "particle positions" with no presence of evolution operators.

hydroPSO is capable of performing sensitivity analysis using the Latin Hypercube One-At-a-Time (LH-OAT) method (van Griensven et al., 2006), which together with advanced plotting summaries and detailed information about the evolution of hydroPSO's performance facilitate the interpretation and assessment of the model calibration. At the same time, hydroPSO features a suite of controlling options and PSO variants to fine-tuning and improve the performance of the calibration engine, thus, allowing the user to adapt it to different modelling problems. In principle, hydroPSO only needs to know "which" model parameters need to be calibrated and "where" they need to be written, but can also be interfaced with the model code through simple R wrapper functions. Then, it will take control over the model(s) to be calibrated until either a maximum number of iterations or an error tolerance are reached: both being problem-specific and user-defined.

The default control arguments in hydroPSO implements the Standard PSO 2007 - SPSO2007 (see Clerc 2005; Clerc et al., 2010). At the same time, hydroPSO can also implement 4 different topologies (gbest, lbest, von Neuman, random), (non-)linear / random / adaptive / best-ratio inertia weight

definitions (IW.type), time-variant acceleration coefficients (use.TVc1 and use.TVc2), time-varying maximum velocity (use.TVvmax), regrouping strategy when premature convergence is detected (use.RG), options for clamping the maximal velocity (lambda), random or LHS initialization of positions and velocities (Xini.type and Vini.type), synchronous or asynchronous update, 4 types of boundary conditions (reflecting, damping, absorbing, invisible) among others.

Details

Package:	hydroPSO
Type:	Package
Version:	0.1-54
Date:	2012-03-30
License:	GPL (>=2)
LazyLoad:	yes

Author(s)

Mauricio Zambrano-Bigiarini

Maintainer: Mauricio Zambrano-Bigiarini <mzb.devel@gmail.com>

See Also

http://www.rforge.net/hydroGOF/.
http://www.rforge.net/hydroTSM/.

http://cran.r-project.org/web/packages/hydroGOF/.
http://cran.r-project.org/web/packages/hydroTSM/.

http://rwiki.sciviews.org/doku.php?id=guides:tutorials:hydrological_data_analysis

---

`hydromod` *hydromod - Definition and execution of the model to be calibrated*

Description

Run a user-defined model to be calibrated, obtaining a goodness-of-fit value as measure of model performance by comparing observations against simulated equivalents

Usage

```
hydromod(param.values, param.files = "ParamFiles.txt", model.drty = getwd(),
         exe.fname = "./swat2005.out", stdout= FALSE, stderr="",
         verbose = FALSE, out.FUN, out.FUN.args, gof.FUN, gof.FUN.args=list(),
         gof.Ini, gof.Fin, date.fmt = "%Y-%m-%d", obs,
         do.png=FALSE, png.fname, width = 1200, height = 600, res=90,
         main, leg.cex=1.2, tick.tstep= "auto", lab.tstep= "auto", lab.fmt=NULL
         )
```

Arguments

param.values	numeric vector, parameter values that will be used in the model
param.files	character, file name (full path) storing location and names of the files that have to be modified for each parameter
model.drty	character, path storing the executable file of the model and ALL the input files required for the simulation
exe.fname	character, name and extension of the file executing the model
stdout, stderr	where output to 'stdout' or 'stderr' should be sent. Possible values are FALSE (discard output, the default), "", to the R console. See system2
verbose	logical; if TRUE, 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 measure
out.FUN	character, name of a valid R function to read the model outputs and transform them into a zoo object
out.FUN.args	list, arguments to be passed to out.FUN
gof.FUN	character, name of a valid (goodness-of-fit) R function to obtain model performance
gof.FUN.args	list, arguments to be passed to gof.FUN
gof.Ini	OPTIONAL. Character with the starting date used in the goodness-of-fit function It is used to subset obs (if necessary), AND to define the time period to compare simulated with observed values
gof.Fin	OPTIONAL. Character with the ending date used in the goodness-of-fit function It is used to subset obs (if necessary), AND to define the time period to compare simulated with observed values
date.fmt	character, format in which the dates are stored in Sim.Ini, Sim.Fin, gof.Ini, gof.Fin, e.g. *%Y-%m-%d*. See format in as.Date
obs	zoo object with the observed values
do.png	logical indicating if a png image with the results of the ggof.FUN function has to be produced
png.fname	OPTIONAL. Used only when do.png=TRUE Name of the PNG file to be produced. The default values is 'Obs_vs_Sim.png', within the model.drty directory
width	OPTIONAL. Used only when do.png=TRUE numeric, width of the output PNG image
height	OPTIONAL. Used only when do.png=TRUE numeric, height of the output PNG image
res	OPTIONAL. Used only when do.png=TRUE numeric, resolution of the output PNG image
main	OPTIONAL. Used only when do.png=TRUE character, representing the main title of the plot comparing observed and simulated values
leg.cex	See ggof
tick.tstep	See ggof
lab.tstep	See ggof
lab.fmt	See ggof

Value

A list of two elements:

sim	numeric, with the simulated values obtained by running the model
GoF	numeric, goodness-of-fit value representing how close each onbe of the simulated values in sim are to their observed counterparts, by using the USER-DEFINED gof.FUN function

Author(s)

Mauricio Zambrano-Bigiarini, <mzb.devel@gmail.com>

See Also

hydroPSO

---

hydroPSO *Enhanced Particle Swarm Optimisation algorithm*

Description

Particle Swarm Optimisation algorithm to calibrate environmental models. It includes a series of controlling options and PSO variants to improve the performance of the algorithm and customize it to different calibration problems

Usage

```
hydroPSO(par, fn= "hydromod", method=c("pso", "ipso", "fips", "wfips"),
         lower=-Inf, upper=Inf, control=list(),
         model.FUN=NULL, model.FUN.args=list() )
```

Arguments

par	OPTIONAL. numeric with a first guess for the parameters to be optimised (n, number of parameters and dimension of the solution space) All the particles are randomly initialised according to the value of Xini.type. If the user provides m parameter sets for par, they are used to overwrite the first m parameter sets randomly defined according to the value of Xini.type. If some elements in par are non finite (lower than lower or larger than upper) they are ignored
fn	character, name of a valid R function c('sphere','ackley','griewank','rastrigin', to be optimised or character value ''hydromod". When fn='hydromod' the algorithm uses model.FUN and model.FUN.args to extract the values simulated by the model and to compute its corresponding goodness-of-fit function. When fn!='hydromod' the algorithm uses the value(s) returned by fn as both model output and its corresponding goodness-of-fit When fn='hydromod' the algorithm will optimise the model defined by model.FUN and model.args

`method` character, variant of the PSO algorithm to be used. Valid values are in `c('pso', 'ipso', 'fips', 'wfips')`:

pso: at each iteration particles are attracted to its own best-known personal and to the best-known global position. Each particle is connected to a neighbourhood of particles depending on the `topology` value
ipso: at each iteration particles in the swarm are rearranged in descending order according to their goodness-of-fit and the best `ngbest` particles are used to modify particles' position and velocity (see Zhao, 2006). Each particle is connected to a neighbourhood of particles depending on the `topology` value
fips: at each iteration ALL particles contribute to modify the particles' position and velocity (see Mendes et al., 2004). Each particle is connected to a neighbourhood of particles depending on the `topology` value
wfips: same implementation as *fips* method, but the contribution of each particle is weighted according to their goodness-of-fit value (see Mendes et al., 2004)
By default `method=pso`

`lower` numeric, lower boundary for each parameter
Note for `optim` users: in *hydroPSO* the length of `lower` and `upper` are used to defined the dimension of the solution space

`upper` numeric, upper boundary for each parameter
Note for `optim` users: in *hydroPSO* the length of `lower` and `upper` are used to defined the dimension of the solution space

`control` a list of control parameters. See 'Details'

`model.FUN` OPTIONAL. Used only when `fn='hydromod'`
character, valid R function representing the model code to be calibrated/optimised

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

Details

By default the hydroPSO function performs minimization of `fn`, but it will maximize `fn` if `MinMax='max'`

The default control arguments in hydroPSO implements the Standard PSO 2007 - SPSO2007 (see Clerc 2005; Clerc et al., 2010). At the same time, hydroPSO function provides options for clamping the maximal velocity, regrouping strategy when premature convergence is detected, time-variant acceleration coefficients, time-varying maximum velocity, (non-)linear / random / adaptive / best-ratio inertia weight definitions, random or LHS initialization of positions and velocities, synchronous or asynchronous update, 4 alternative neighbourhood topologies among others

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

drty.in OPTIONAL. Used only when `fn='hydromod'`
character, name (without path) 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 hydroPSO

param.ranges OPTIONAL. Used only when `fn='hydromod'`
character, name of the file storing the desired range of variation of each parameter

digits OPTIONAL. Used only when `write2disk=TRUE`
numeric, number of significant digits used for writing the outputs in scientific notation

MinMax character, indicates whether a maximization or minimization problem needs to be solved. Valid values are in: `c('min', 'max')`. Default value is *min*

npart numeric, number of particles in the swarm

maxit numeric, maximum number of iterations

maxfn numeric, maximum number of function evaluations. Default value is `+Inf`

c1 numeric, cognitive acceleration coefficient. Encourages the exploration of the solution space and reflects how much the particle is influenced by its own best-known position

c2 numeric, social acceleration coefficient. Encourages the exploitation of the current global best and reflects how much the particle is influenced by the best-known optimum of the swarm

use.CF logical, indicates if the Clerc's Constriction Factor (see Clerc, 1999; Eberhart and Shi, 2000; Clerc and Kennedy, 2002) is used to avoid swarm explosion

lambda numeric in [0,1], represents a percentage to limit the maximum velocity (Vmax) for each dimension, which is computed as `vmax = lambda*(Xmax-Xmin)`

abstol numeric, absolute convergence tolerance. The algorithm stops if `gbest <= abstol` (minimisation problems) OR when `gbest >= abstol` (maximisation problems)
By default it is set to `-Inf` or `+Inf` for minimisation or maximisation problems, respectively

reltol numeric, relative convergence tolerance. The algorithm stops if it is unable to reduce the value by a factor of `reltol * (abs(val) + reltol)` at a given iteration. Defaults to `sqrt(.Machine$double.eps)`, typically, about 1e-8

Xini.type character, indicates how to initialise the particles' positions in the swarm within the ranges defined by `lower` and `upper`. Valid values are:
-) *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=lhs`

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
-) *random*: random initialisation of velocities within `lower` and `upper` using the 'half-diff' method ('`Vini=[U(lower, upper)-Xini]/2`') (see Clerc, 2010)
-) *lhs*: Latin Hypercube initialisation of velocities using `npart` number of strata to divide each parameter range and the *half-diff* method ('`Vini=[LHS(lower, upper)-Xini]/2`') (see Clerc, 2010). **It requires the lhs package**
By default `Vini.type=lhs`

best.update character, indicates how (when) to update the global and local best. Valid values are:
-)*sync*: the update is made synchronously, i.e. after computing the position and goodness-of-fit for ALL the particles in the swarm. This is the DEFAULT option
-)*async*: the update is made asynchronously, i.e. after computing the position and goodness-of-fit for EACH individual particle in the swarm

boundary.wall character, indicates the type of boundary condition to be applied during optimisation. Valid values are in `c('reflecting', 'damping', 'absorbing', 'invisible')`

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)

topology character, indicates the neighbourhood topology used in hydroPSO. Valid values are in `c('gbest', 'lbest', 'vonNeumann', 'random')`:
gbest: every particle is connected to each other and, hence the global best influences all particles in the swarm. This is also termed '`star`' topology, and it is generally assumed to have a fast convergence but is more vulnerable to the attraction to sub-optimal solutions (see Kennedy, 1999; Kennedy and Mendes, 2002, Schor et al., 2010)
lbest: each particle is connected to its `K` immediate neighbours only. This is also termed '`circles`' or '`ring`' topology, and generally the swarm will converge slower than the *gbest* topology but it is less vulnerable to sub-optimal solutions (see Kennedy, 1999; Kennedy and Mendes, 2002)
vonNeumann: each particle is connected to its `K=4` immediate neighbours only. This topology is more densely connected than '`lbest`' but less densely than '`gbest`', thus, showing some parallelism with '`lbest`' but benefiting from a bigger neighbourhood (see Kennedy and Mendes, 2003)
random: the random topology is a special case of '`lbest`' where connections among particles are randomly modified after an iteration showing no improvement in the global best (see Clerc, 2005; Clerc, 2010)
By default `topology=gbest`

K OPTIONAL. Only used when `topology` is in `c(lbest, vonNeumann, random)`
numeric, neighbourhood size, i.e. the number of informants for each particle (including the particle itself) to be considered in the computation of their personal best
When `topology=lbest` `K` MUST BE an even number in order to consider the same amount of neighbours to the left and the right of each particle.
As special case, `K` could be equal to `npart`
By default `K=3`

iter.ini OPTIONAL. Only used when `topology=='lbest'`
numeric, number of iterations for which the *gbest* topology will be used before using the *lbest* topology for the computation of the personal best of each particle
This option aims at making faster the identification of the global zone of attraction
By default `iter.ini=0`

ngbest OPTIONAL. Only used when `method=='ipso'`
numeric, number of particles considered in the computation of the global best
By default `ngbest=4` (see Zhao, 2006)

use.IW logical, indicates if an inertia weight (`w`) will be used to avoid swarm explosion, i.e. particles flying around their best position without converging into it (see Shi and Eberhart, 1998)

IW.type OPTIONAL. Used only when `use.IW= TRUE`
character, defines how the inertia weight `w` will vary along iterations. Valid values are:
-)*linear*: `w` varies linearly between the initial and final values specified in `IW.w` (see Shi and Eberhart, 1998; Zheng et al., 2003)
-)*non-linear*: `w` varies non-linearly between the initial and final values specified in `IW.w` with exponential factor *IW.exp* (see Chatterjee and Siarry, 2006)
-)*runif*: `w` is a uniform random variable in the range `[w.min, w.max]` specified in `IW.w`. It is a generalisation of the weight proposed in Eberhart and Shi (2001b)
-)*aiwf*: adaptive inertia weight factor, where the inertia weight is varied adaptively depending on the goodness-of-fit values of the particles (see Liu et al., 2005)
-)*GLratio*: `w` varies according to the ratio between the global best and the average of the particle's local best (see Arumugam and Rao, 2008)
By default `IW.type=linear`

IW.w OPTIONAL. Used only when `use.IW= TRUE & IW.type!='GLratio'`
numeric, value of the inertia weight(s) (`w` or `[w.ini, w.fin]`). It can be a single number

which is used for all iterations, or can be a vector of length 2 with the initial and final values (in that order) that `w` will take along the iterations

IW.exp OPTIONAL. Used only when `use.IW= TRUE` AND `IW.type= 'non-linear'`
numeric, non-linear modulation index (see Chatterjee and Siarry, 2006)
When `IW.type='linear'`, `IW.exp` is set to 1

use.TVc1 logical, indicates if the cognitive acceleration coefficient `c1` will have a time-varying value instead of a constant one provided by the user (see Ratnaweera et al. 2004)
By default `use.TVc1=TRUE`

TVc1.type character, required only when `use.TVc1 = TRUE`. Valid values are:
-)*`linear`*: `c1` varies linearly between the initial and final values specified in `TVc1.rng` (see Ratnaweera et al., 2004)
-)*`non-linear`*: `c1` varies non-linearly between the initial and final values specified in `TVc1.rng`. Proposed by the authors of hydroPSO taking into account the work of Chatterjee and Siarry (2006) for the inertia weight
-)*`GLratio`*: `c1` varies according to the ratio between the global best and the average of the particle's local best (see Arumugam and Rao, 2008)
By default `TVc1.type=linear`

TVc1.rng OPTIONAL. Used only when `use.TVc1= TRUE & TVc1.type!='GLratio'`
numeric, initial and final values for the cognitive acceleration coefficient `[c1.ini, c1.fin]` (in that order) along the iterations

TVc1.exp OPTIONAL. Used only when `use.TVc1= TRUE` AND `TVc1.type= 'non-linear'`
numeric, non-linear modulation index
When `TVc1.exp` is equal to 1, `TVc1` corresponds to the improvement proposed by Ratnaweera et al., (2004), whereas when `TVc1.exp` is different from one, no reference has been found in literature by the authors, but it was included as an option based on the work of Chatterjee and Siarry (2006) for the inertia weight
When `TVc1.type= linear` `TVc1.exp` is automatically set to 1

use.TVc2 logical, indicates whether the social acceleration coefficient `c2` will have a time-varying value or a constant one provided by the user (see Ratnaweera et al. 2004)
By default `use.TVc2=FALSE`

TVc2.type character, required only when `use.TVc2=TRUE`. Valid values are:
-)*`linear`*: `c2` varies linearly between the initial and final values specified in `TVc2.rng` (see Ratnaweera et al. 2004)
-)*`non-linear`*: `c2` varies non-linearly between the initial and final values specified in `TVc2.rng`. Proposed by the authors of hydroPSO taking into account the work of Chatterjee and Siarry (2006) for the inertia weight
By default `TVc2.type=linear`

TVc2.rng OPTIONAL. Used only when `use.TVc2=TRUE`
numeric, initial and final values for the social acceleration coefficient `[c2.ini, c2.fin]` (in that order) along the iterations

TVc2.exp OPTIONAL. Used only when `use.TVc2= TRUE` AND `TVc2.type='non-linear'`
numeric, non-linear modulation index
When `TVc2.exp` is equal to 1, `TVc2` corresponds to the improvement proposed by Ratnaweera et al., 2004, whereas when `TVc2.exp` is different from one, no reference has been found in literature by the authors, but it was included as an option based on the work of Chatterjee and Siarry (2006) for the inertia weight
When `TVc2.type= linear` `TVc2.exp` is automatically set to 1

use.TVlambda logical, indicates whether the percentage to limit the maximum velocity `lambda` will have a time-varying value or a constant value provided by the user. Proposed by the

authors of hydroPSO based on the work of Chatterjee and Siarry (2006) for the inertia weight
By default `use.TVlambda=TRUE`

TVlambda.type character, required only when `use.TVlambda=TRUE`. Valid values are:
-)*`linear`*: `TVvmax` varies linearly between the initial and final values specified in `TVlambda.rng`
-)*`non-linear`*: `TVvmax` varies non-linearly between the initial and final values specified in `TVlambda.rng`
By default `TVlambda.type=linear`

TVlambda.rng OPTIONAL. Used only when `use.TVlambda=TRUE`
numeric, initial and final values for the percentage to limit the maximum velocity `[TVlambda.ini, TVlambda.fin]` (in that order) along the iterations

TVlambda.exp OPTIONAL. only required when `use.TVlambda= TRUE` AND `TVlambda.type='non-linear'`
numeric, non-linear modulation index
When `TVlambda.type= linear` `TVlambda.exp` is automatically set to 1

use.RG logical, indicates if the swarm should be regrouped when premature convergence is detected. When `use.RG=TRUE` the swarm is regrouped in a search space centred around the current global best. This updated search space is hoped to be both small enough for efficient search and large enough to allow the swarm to escape from stagnation (see Evers and Ghalia, 2009)

RG.thr ONLY required when `use.RG=TRUE`
numeric, positive number representing the *`stagnation threshold`* used to decide whether the swarm has to be regrouped or not. See Evers and Galia (2009) for further details
Regrouping occurs when the *`normalised swarm radius`* is less than `RG.thr`

RG.r ONLY required when `use.RG=TRUE`.
numeric, positive number representing the *`regrouping factor`*, which is used to regroup the swarm in a search space centred around the current global best (see Evers and Galia, 2009 for further details)

RG.miniter ONLY required when `use.RG=TRUE`
numeric, minimum number of iterations needed before regrouping

plot logical, indicatesif a two-dimensional plot with the particles' position will be drawn after each iteration. For high dimensional functions, only the first two dimensions of all the particles are plotted

out.with.pbest logical, indicates if the best parameter values for each particle and their goodness-of-fit will be included in the output of the algorithm
By default `out.with.pbest=FALSE`

out.with.fit.iter logical, indicates if the goodness-of-fit of each particle for each iteration will be included in the output of the algorithm
By default `out.with.fit.iter=FALSE`

write2disk logical, indicates if the output files will be written to the disk

verbose logical, indicates if progress messages are to be printed

REPORT OPTIONAL. Used only when `verbose=TRUE`
The frequency of report messages printed to the screen. Default to every 10 iterations

Value

A list, compatible with the output from `optim`, with components:

`par`	optimum parameter set found
`value`	value of `fn` corresponding to `par`

counts	three-element vector containing the number of function evaluations, number of iterations, and number of regrouping
convergence	integer code where 0 indicates that the algorithm terminated by reaching the absolute tolerance, otherwise: 1: relative tolerance reached 2: maximum number of function evaluations reached 3: maximum number of iterations reached
message	character string giving human-friendly information about convergence

Note

Note for optim users:
1) In *hydroPSO*, par may be omitted. If not omitted, the m parameter sets provided by the user for par are used to overwrite the first m parameter sets randomly defined according to the value of Xini.type
2) In *hydroPSO* the length of lower and upper are used to define the dimension of the solution space (not the length of par)

Author(s)

Mauricio Zambrano-Bigiarini, <mzb.devel@gmail.com>

References

Kennedy, J. and R. Eberhart. Particle Swarm Optimization. in proceedings IEEE international conference on Neural networks. pages 1942-1948. 1995. doi: 10.1109/ICNN.1995.488968

Kennedy, J.; Mendes, R.. Population structure and particle swarm performance. Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on , vol.2, no., pp.1671-1676, 2002. doi: 10.1109/CEC.2002.1004493

Kennedy, J.; Mendes, R.; , Neighborhood topologies in fully-informed and best-of-neighborhood particle swarms. Soft Computing in Industrial Applications, 2003. SMCia/03. Proceedings of the 2003 IEEE International Workshop on , vol., no., pp. 45- 50, 23-25 June 2003. doi: 10.1109/SMCIA.2003.1231342

Kennedy, J.; Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance. Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on , vol.3, no., pp.3 vol. (xxxvii+2348), 1999. doi: 10.1109/CEC.1999.785509

Clerc, M and J Kennedy. The particle swarm - explosion, stability, and convergence in a multidimensional complex space. IEEE Transactions On Evolutionary Computation, 6:58-73, 2002. doi:10.1109/4235.985692

Clerc, M. Particle Swarm Optimization. ISTE, 2005

Clerc, M. From Theory to Practice in Particle Swarm Optimization, Handbook of Swarm Intelligence, Springer Berlin Heidelberg, 3-36, Eds: Panigrahi, Bijaya Ketan, Shi, Yuhui, Lim, Meng-Hiot, Hiot, Lim Meng, and Ong, Yew Soon, 2010, doi: 10.1007/978-3-642-17390-5_1

Clerc, M., Stagnation Analysis in Particle Swarm Optimisation or what happens when nothing happens. Technical Report. 2006. http://hal.archives-ouvertes.fr/hal-00122031

Clerc, M. Standard Particle Swarm. 2011. (SPSO-2007, SPSO-2011). clerc.maurice.free.fr/pso/SPSO_descriptions.pdf. Last visited [25-Jan-2012]

Chatterjee, A. and Siarry, P. Nonlinear inertia weight variation for dynamic adaptation in particle swarm optimization, Computers \& Operations Research, Volume 33, Issue 3, March 2006, Pages 859-871, ISSN 0305-0548, DOI: 10.1016/j.cor.2004.08.012

Eberhart, R.C.; Shi, Y.; Comparing inertia weights and constriction factors in particle swarm optimization. Evolutionary Computation, 2000. Proceedings of the 2000 Congress on , vol.1, no., pp.84-88 vol.1, 2000. doi: 10.1109/CEC.2000.870279

Evers, G.I.; Ben Ghalia, M. Regrouping particle swarm optimization: A new global optimization algorithm with improved performance consistency across benchmarks. Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on , vol., no., pp.3901-3908, 11-14 Oct. 2009. doi: 10.1109/ICSMC.2009.5346625

Huang, T.; Mohan, A.S.; , A hybrid boundary condition for robust particle swarm optimization. Antennas and Wireless Propagation Letters, IEEE , vol.4, no., pp. 112-117, 2005. doi: 10.1109/LAWP.2005.846166

Liu, B. and L. Wang, Y.-H. Jin, F. Tang, and D.-X. Huang. Improved particle swarm optimization combined with chaos. Chaos, Solitons \& Fractals, vol. 25, no. 5, pp.1261-1271, Sep. 2005. doi:10.1016/j.chaos.2004.11.095

Mendes, R.; Kennedy, J.; Neves, J. The fully informed particle swarm: simpler, maybe better. Evolutionary Computation, IEEE Transactions on , vol.8, no.3, pp. 204-210, June 2004. doi: 10.1109/TEVC.2004.826074

Ratnaweera, A.; Halgamuge, S.K.; Watson, H.C. Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. Evolutionary Computation, IEEE Transactions on , vol.8, no.3, pp. 240- 255, June 2004. doi: 10.1109/TEVC.2004.826071

Robinson, J.; Rahmat-Samii, Y.; Particle swarm optimization in electromagnetics. Antennas and Propagation, IEEE Transactions on , vol.52, no.2, pp. 397-407, Feb. 2004. doi: 10.1109/TAP.2004.823969

Shi, Y.; Eberhart, R. A modified particle swarm optimizer. Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence. The 1998 IEEE International Conference on , vol., no., pp.69-73, 4-9 May 1998. doi: 10.1109/ICEC.1998.699146

Schor, D.; Kinsner, W.; Anderson, J.; , A study of optimal topologies in swarm intelligence. Electrical and Computer Engineering (CCECE), 2010 23rd Canadian Conference on , vol., no., pp.1-8, 2-5 May 2010. doi: 10.1109/CCECE.2010.5575132

Yong-Ling Zheng; Long-Hua Ma; Li-Yan Zhang; Ji-Xin Qian. On the convergence analysis and parameter selection in particle swarm optimization. Machine Learning and Cybernetics, 2003 International Conference on , vol.3, no., pp. 1802-1807 Vol.3, 2-5 Nov. 2003. doi: 10.1109/ICMLC.2003.1259789

Zhao, B. An Improved Particle Swarm Optimization Algorithm for Global Numerical Optimization. In Proceedings of International Conference on Computational Science (1). 2006, 657-664

Neighborhood Topologies, http://tracer.uc3m.es/tws/pso/neighborhood.html. Last visited [15-Feb-2012]

See Also

optim

Examples

```
# Number of dimensions to be optimised
nparam <- 5

## Not run:
# Setting the home directory of the user as working directory
setwd("~")

# Setting the seed
set.seed(100)
```

```
hydroPSO(
        fn="rastrigrin",
        lower=rep(-5.12, nparam), upper=rep(5.12, nparam),
        control=list(
                    MinMax="min",
                    npart=2*nparam,

                    use.TVlambda= TRUE, TVlambda.type= "linear",
                    TVlambda.rng= c(1, 0.5), TVlambda.exp= 1,

                    topology="gbest",
                    write2disk=TRUE
                    ) # control
        ) # hydroPSO

# Plotting the results
plot_results(MinMax="min")

## End(Not run) # dontrun END
```

`lhoat` *Latin-Hypercube One-factor-At-a-Time*

Description

This function implements the Latin-Hypercube One-factor-At-a-Time procedure developed by van Griensven et al., (2006) for sensitivity analysis of model parameters

Usage

```
lhoat(fn="hydromod", lower=-Inf, upper=Inf, control=list(),
      model.FUN=NULL, model.FUN.args=list() )
```

Arguments

`fn`	character, name of a valid R function to be optimised or character value '*hydromod*'. When `fn='hydromod'` the algorithm uses `model.FUN` and `model.FUN.args` to extract the values simulated by the model and to compute its corresponding goodness-of-fit function. When `fn!='hydromod'` the algorithm uses the value(s) returned by `fn` as both model output and its corresponding goodness-of-fit When `fn='hydromod'` the algorithm will optimise the model defined by `model.FUN` and `model.args`
`lower`	numeric, lower boundary for each parameter Note for `optim` users: in *hydroPSO* the length of `lower` and `upper` are used to defined the dimension of the solution space
`upper`	numeric, upper boundary for each parameter Note for `optim` users: in *hydroPSO* the length of `lower` and `upper` are used to defined the dimension of the solution space
`control`	a list of control parameters. See 'Details'

`model.FUN` OPTIONAL. Used only when `fn='hydromod'`
character, valid R function representing the model code to be calibrated/optimised

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

Details

By default the hydroPSO function performs minimization of `fn`, but it will maximize `fn` if `MinMax='max'`

The default control arguments in hydroPSO implements the Standard PSO 2007 - SPSO2007 (see Clerc 2005; Clerc et al., 2010). At the same time, hydroPSO function provides options for clamping the maximal velocity, regrouping strategy when premature convergence is detected, time-variant acceleration coefficients, time-varying maximum velocity, (non-)linear / random / adaptive / best-ratio inertia weight definitions, random or LHS initialization of positions and velocities, synchronous or asynchronous update, 4 alternative neighbourhood topologies among others

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

N numeric, number of strata to be used for sampling the range, as provided in `params.ranges`, for each parameter

f numeric, fraction of the parameter's range by which each single parameter of the initial LHS is changed within the Morris OAT design

drty.in character, path to 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 *hydroPSO*

param.ranges OPTIONAL. Used only when `fn='hydromod'`
character, name of the file storing the desired range of variation of each parameter

digits OPTIONAL. Used only when `write2disk=TRUE`
numeric, number of significant digits used for writing the outputs in scientific notation

gof.name character, ONLY used for identifying the goodness-of-fit of each model run and writing it to the *LH_OAT-gof.txt* output file

do.plots logical, if TRUE a PNG plot with the comparison between observed and simulated values is produced for each parameter set used in the LH-OAT

write2disk logical, indicates if the output files will be written to the disk

verbose logical, if TRUE progress messages are printed

Value

A list of two elements:

`ParameterSets`
a matrix with all the parameter sets used in the LH-OAT

`Ranking` a dataframe with a ranking, parameter id, and relative importance indicator for each parameter, sorted in decreasing order of importance

Author(s)

Mauricio Zambrano-Bigiarini, <mzb.devel@gmail.com>

References

A. van Griensven, T. Meixner, S. Grunwald, T. Bishop, M. Diluzio, R. Srinivasan, A global sensitivity analysis tool for the parameters of multi-variable catchment models, Journal of Hydrology, Volume 324, Issues 1-4, 15 June 2006, Pages 10-23, DOI: 10.1016/j.jhydrol.2005.09.008.

See Also

hydroPSO, hydromod

Examples

```
# Number of dimensions to be optimised
nparam <- 5

# Running the Latin-Hypercube One-factor-At-a-Time sensitivity analysis.
# Dummy case, because for this test function all the parameters have the
# same sensitivity
## Not run:
lhoat(
      fn="sphere",
      lower=rep(-100,nparam),
      upper=rep(100,nparam),
      control=list(N=10, f=0.1, write2disk=TRUE)

      )

# Plotting dotty plots
read_params(file="LH_OAT-gof.txt", header=TRUE, skip=0, param.cols=2:(nparam+1), of.col=1

## End(Not run)
```

---

params2ecdf *Parameter Values -> Empirical CDFs*

Description

This function computes (weighted) empirical CDFs (ECDFs) for each calibrated parameter, by using the parameter values obtained during the optimisation with PSO with optional plot

Usage

```
params2ecdf(params, ...)

## Default S3 method:
params2ecdf(params, param.names=NULL, weights=NULL,
        byrow=FALSE, plot=TRUE, obs=NULL, main=NULL, nrows="auto",
        ylab="Probability", col="blue", leg.cex=1.2, leg.pos="topleft",
        cex.axis=1.2, cex.main=1.2, cex.lab=1.2, verbose=TRUE, ...,
        do.png=FALSE, png.width=1500, png.height=900, png.res=90,
        png.fname="Params_ECDFs.png")
```

```
## S3 method for class 'matrix'
params2ecdf(params, param.names=colnames(params), weights,
        byrow=FALSE, plot=TRUE, obs=NULL, main=NULL, nrows="auto",
        ylab="Probability", col="blue", leg.cex=1.2, leg.pos="topleft",
        cex.axis=1.2, cex.main=1.2, cex.lab=1.2, verbose=TRUE, ...,
        do.png=FALSE, png.width=1500, png.height=900, png.res=90,
        png.fname="Params_ECDFs.png")

## S3 method for class 'data.frame'
params2ecdf(params, param.names=colnames(params), weights,
        byrow=FALSE, plot=TRUE, obs=NULL, main=NULL, nrows="auto",
        ylab="Probability", col="blue", leg.cex=1.2, leg.pos="topleft",
        cex.axis=1.2, cex.main=1.2, cex.lab=1.2, verbose=TRUE, ...,
        do.png=FALSE, png.width=1500, png.height=900, png.res=90,
        png.fname="Params_ECDFs.png")
```

Arguments

`params`	matrix or data.frame with the parameter values, where each row represent a different parameter set and each column represent the value of a different model parameter
`param.names`	character vector, names to be used for each parameter in `params` (by default its column names)
`weights`	numeric vector, values of the weights to be used for computing the empirical CDFs Omitting the `weights` argument or specifying `NULL` or a zero-length vector will result in the usual un-weighted estimates
`byrow`	logical, indicates whether the computations have to be made for each column or for each row of `params` When the parameter sets are stored in rows, i.e. values for different model's parameter are stored in columns, `byrow` must be *`FALSE`* When the parameter sets are stored in columns, i.e. values for different model's parameter are stored in rows, `byrow` must be *`TRUE`*
`plot`	logical, indicates whether a plot with the Empirical CDF for each model's parameter has to be produced or not
`obs`	OPTIONAL. Only used when `plot=TRUE` Numeric or zoo object with observed values (one for each `params`), which are used in the output plot
`main`	an overall title for the plot
`nrows`	OPTIONAL. Only used when `plot=TRUE` numeric, number of rows to be used in the plotting window. If `nrows` is set to *`auto`*, the number of rows is automatically computed depending on the number of columns of `params`
`ylab`	OPTIONAL. Only used when `plot=TRUE` a title for the y axis. See `plot`
`col`	OPTIONAL. Only used when `plot=TRUE` a specification for the default plotting colour. See `par`
`leg.cex`	OPTIONAL. Only used when `plot=TRUE` character expansion factor *relative* to current 'par("cex")'. Used for text, and provides the default for 'pt.cex' and 'title.cex'. Default value = 1.2

leg.pos	OPTIONAL. Only used when plot=TRUE keyword to be used to position the legend. See legend
cex.axis	OPTIONAL. Only used when plot=TRUE numeric, magnification to be used for axis annotation relative to the current setting of cex
cex.main	OPTIONAL. Only used when plot=TRUE numeric, magnification to be used for main titles relative to the current setting of cex
cex.lab	OPTIONAL. Only used when plot=TRUE numeric, magnification to be used for x and y labels relative to the current setting of cex
verbose	logical, if TRUE, progress messages are printed
...	further arguments passed to the plot function or from other methods
do.png	logical, indicates if all the figures have to be saved into PNG files instead of the screen device
png.width	OPTIONAL. Only used when do.png=TRUE numeric with the width of the device. See png
png.height	OPTIONAL. Only used when do.png=TRUE numeric with the height of the device. See png
png.res	OPTIONAL. Only used when do.png=TRUE numeric with the nominal resolution in ppi which will be recorded in the PNG file, if a positive integer of the device. See png
png.fname	OPTIONAL. Only used when do.png=TRUE character, with the filename used to store the PNG file

Author(s)

Mauricio Zambrano-Bigiarini, <mzb.devel@gmail.com>

See Also

wtd.Ecdf, quant2ecdf

Examples

```
## Not run:
# Setting the user's home directory as working directory
setwd("~")

# matrix with 100 random uniform parameter sets (in rows) for 10 different model's
# parameters (in columns)
params           <- matrix(rnorm(1000), ncol=10, nrow=100)
colnames(params) <- paste("Param", 1:10, sep="")

# empirical CDFs for each one of the 10 parameters in 'params', with equal weight
# for each one of the 100 parameter sets
params2ecdf(params, weights=rep(1,100))

## End(Not run)
```

plot_2parOF *plot_2parOF*

Description

This function plots the values of the objective function in a two dimensional box, where the boundaries of each parameter are used as axis limit

Usage

```
plot_2parOF(params, gofs, p1.name, p2.name, type="sp", MinMax=c("min", "max"),
            gof.name="GoF", main=paste(gof.name, "Surface"), GOFcuts,
            colorRamp= colorRampPalette(c("darkred", "red", "orange", "yellow",
            "green", "darkgreen", "cyan")), points.cex=0.7, alpha=1,
            axis.rot=c(0, 0), auto.key=TRUE, key.space= "right")
```

Arguments

params	matrix or data.frame with the parameter values
gofs	numeric with the values of goodness-of-fit values for each one of the parameters in params (in the same order!)
p1.name	character, name of the 1st parameter to be plotted
p2.name	character, name of the 2nd parameter to be plotted
type	character, type of plot. Valid values are: -) *sp*: spatial plot -) *scatter3d*: 3d scatterogram
MinMax	character, indicates whether the optimum value in gofs corresponds to the minimum or maximum of the objective function. Valid values are in: c('min', 'max')
gof.name	character, name of the objective function to be plotted. It has to correspond to the name of one column of params
main	character with the title for the plot
GOFcuts	numeric, specifies at which values of the objective function gof.name the colours of the plot have to change If GOFcuts is missing, the interval for colours change are defined by the five quantiles of the objective function computed by fivenum
colorRamp	R function defining the colour ramp to be used for colouring the pseudo-3D dotty plots of Parameter Values, OR character representing those colours
points.cex	size of the points to be plotted
alpha	numeric between 0 and 1 representing the transparency level to apply to colorRamp, '0' means fully transparent and '1' means opaque
axis.rot	numeric vector of length 2 representing the angle (in degrees) by which the axis labels are to be rotated, left/bottom and right/top, respectively.
auto.key	logical, indicates whether the legend has to be drawn or not
key.space	character, position of the legend with respect to the plot

Author(s)

Mauricio Zambrano-Bigiarini, <mzb.devel@gmail.com>

See Also

read_results, plot_results, plot_GofPerParticle, plot_ParamsPerIter

---

plot_NparOF *N 2-dimensional plots of Parameter Values against the Objective Function*

Description

For *n* user-defined parameters, the function creates *sum(1:(npar-1))* plot_2parOF plots, with the values of the objective function in a 2D box, where the boundaries of each parameter are used as axis
The *sum(1:(npar-1))* plots corresponds to all the possible combinations of 2 parameters among all the *n* parameters provided

Usage

```
plot_NparOF(params, gofs, param.names=colnames(params), MinMax=c("min", "max"),
            nrows="auto",  gof.name="GoF", main=paste(gof.name, "Surface"),
            GOFcuts="auto", colorRamp= colorRampPalette(c("darkred", "red",
            "orange", "yellow", "green", "darkgreen", "cyan")), points.cex=0.7,
            alpha=1, axis.rot=c(0, 0), verbose=TRUE)
```

Arguments

params	matrix or data.frame with the parameter values
gofs	numeric with the values of goodness-of-fit values for each one of the parameters in params (in the same order!)
param.names	character, names for the parameters in params that have to be plotted (param.names can be a subset of params)
MinMax	character, indicates whether the optimum value in gofs corresponds to the minimum or maximum of the objective function. Valid values are in: c('min', 'max')
nrows	numeric, number of rows to be used in the plotting window If nrows='auto' the number of columns is automatically computed depending on the number of parameters in params
gof.name	character, name of the objective function to be plotted. It has to correspond to the name of one column of params
main	character, title for the plot
GOFcuts	numeric, specifies at which values of the objective function gof.name the colours of the plot have to change If GOFcuts="auto", the interval for colours change are defined by the five quantiles of the objective function computed by fivenum
colorRamp	R function defining the colour ramp to be used for colouring the pseudo-3D dotty plots of Parameter Values, OR character representing those colours

points.cex	size of the points to be plotted
alpha	numeric between 0 and 1 representing the transparency level to apply to colorRamp, '0' means fully transparent and '1' means opaque
axis.rot	numeric vector of length 2 representing the angle (in degrees) by which the axis labels are to be rotated, left/bottom and right/top, respectively.
verbose	logical; if TRUE, progress messages are printed

Author(s)

Mauricio Zambrano-Bigiarini, <mzb.devel@gmail.com>

See Also

plot_2parOF, read_results, plot_results, plot_GofPerParticle, plot_ParamsPerIter

Examples

```
# Number of dimensions to be optimised
nparam <- 5

## Not run:

# Setting the user's home directory as working directory
setwd("~")

# Setting the seed
set.seed(100)

# Running PSO with the 'rosenbrock' test function, writing the results to text files
hydroPSO(fn="rosenbrock",
        lower=rep(-30, nparam), upper=rep(30, nparam),
        control=list(MinMax="min", npart=2*nparam, write2disk=TRUE)
        ) # hydroPSO

# reading the 'Particles.txt' output file of hydroPSO
setwd("PSO.out")
particles <- read_particles(plot=FALSE)

# plotting the value of each parameter and the objective function against the
# values of the objective function
plot_NparOF(params=particles[["part.params"]], gofs=particles[["part.gofs"]],
            gof.name="Rosenbrock", alpha=0.5)

## End(Not run)
```

---

plot_ParamsPerIter *Plot Parameter Values against the Iteration Number*

Description

Function to plot the value of each parameter against the iteration number

Usage

```
plot_ParamsPerIter(params,...)

## Default S3 method:
plot_ParamsPerIter(params, param.names=colnames(params), main=NULL,
                xlab="Number of evaluations", nrows="auto", cex=0.5, cex.main=1.2
                cex.axis=1.7, cex.lab=1.5, col=rainbow(ncol(params)),
                lty=3, verbose=TRUE, ..., do.png=FALSE, png.width=1500,
                png.height=900, png.res=90, png.fname="Params_ValuePerRun.png" )

## S3 method for class 'matrix'
plot_ParamsPerIter(params, param.names=colnames(params), main=NULL,
                xlab="Number of evaluations", nrows="auto", cex=0.5, cex.main=1.2
                cex.axis=1.7, cex.lab=1.5, col=rainbow(ncol(params)),
                lty=3, verbose=TRUE, ..., do.png=FALSE, png.width=1500,
                png.height=900, png.res=90, png.fname="Params_ValuePerRun.png" )

## S3 method for class 'data.frame'
plot_ParamsPerIter(params, param.names=colnames(params), main=NULL,
                xlab="Number of evaluations", nrows="auto", cex=0.5, cex.main=1.2
                cex.axis=1.7, cex.lab=1.5, col=rainbow(ncol(params)),
                lty=3, verbose=TRUE, ..., do.png=FALSE, png.width=1500,
                png.height=900, png.res=90, png.fname="Params_ValuePerRun.png" )
```

Arguments

params	matrix or data.frame with the parameter values, where each row represent a different parameter set, and each column represent the value of a different model's parameter
param.names	character vector, names to be used for each model's parameter in params (by default its column names)
main	character, title for the plot
xlab	character, title for the x axis. See plot
nrows	numeric, number of rows to be used in the plotting window. If nrows is set to *auto*, the number of rows is automatically computed depending on the number of columns of params
cex	numeric, magnification for text and symbols relative to the default. See par
cex.main	numeric, magnification to be used for main titles relative to the current setting of cex. See par
cex.axis	numeric, magnification to be used for axis annotation relative to the current setting of cex. See par
cex.lab	numeric, magnification to be used for x and y labels relative to the current setting of cex. See par
col	specification for the default plotting colour. See par
lty	line type. See par
verbose	logical, if TRUE, progress messages are printed
...	further arguments passed to the plot function or from other methods.
do.png	logical, indicates if all the figures have to be saved into PNG files instead of the screen device

png.width	OPTIONAL. Only used when do.png=TRUE numeric with the width of the device. See png
png.height	OPTIONAL. Only used when do.png=TRUE numeric with the height of the device. See png
png.res	OPTIONAL. Only used when do.png=TRUE numeric with the nominal resolution in ppi which will be recorded in the PNG file, if a positive integer of the device. See png
png.fname	OPTIONAL. Only used when do.png=TRUE character, with the filename used to store the PNG file

Author(s)

Mauricio Zambrano-Bigiarini, <mzb.devel@gmail.com>

See Also

plot_results, plot_2parOF, plot_NparOF, plot_GofPerParticle

Examples

```
# Number of dimensions to be optimised
nparam <- 5

## Not run:

# Setting the user's home directory as working directory
setwd("~")

# Setting the seed
set.seed(100)

# Running PSO with the 'griewank' test function, writing the results to text files
hydroPSO(
        fn="griewank",
        lower=rep(-600, nparam),
        upper=rep(600, nparam),

        control=list(
                    MinMax="min",
                    npart=2*nparam,
                    use.IW = TRUE, IW.type= "linear",
                    IW.w= c(1.0, 0.4), IW.exp= 1,
                    topology="gbest",
                    write2disk=TRUE
                    ) # control
        ) # hydroPSO

# reading the 'Particles.txt' output file of PSO
setwd("PSO.out")
particles <- read_particles(plot=FALSE)

# plotting the value of each parameter and the objective function against the
# iteration number
plot_ParamsPerIter(particles[["part.params"]])

## End(Not run)
```

`quant2ecdf` *Simulated Values -> Empirical CDFs*

Description

This function computes ECDFs for user-defined quantiles of the simulated equivalents, with optional plot

Usage

```
quant2ecdf(sim, ...)

## Default S3 method:
quant2ecdf(sim, weights=NULL, byrow=TRUE,
           quantiles.desired= c(0.05, 0.5, 0.95), plot=TRUE, obs=NULL,
           quantiles.labels= c("Q5", "Q50", "Q95"), main=NULL,
           ylab="Probability", col="blue", leg.cex=1.2, leg.pos="bottomright",
           cex.axis=1.2, cex.main=1.2, cex.lab=1.2, verbose=TRUE, ...)

## S3 method for class 'matrix'
quant2ecdf(sim, weights=NULL, byrow=TRUE,
           quantiles.desired= c(0.05, 0.5, 0.95), plot=TRUE, obs=NULL,
           quantiles.labels= c("Q5", "Q50", "Q95"), main=NULL,
           ylab="Probability", col="blue", leg.cex=1.2, leg.pos="bottomright",
           cex.axis=1.2, cex.main=1.2, cex.lab=1.2, verbose=TRUE, ...)

## S3 method for class 'data.frame'
quant2ecdf(sim, weights=NULL, byrow=TRUE,
           quantiles.desired= c(0.05, 0.5, 0.95), plot=TRUE, obs=NULL,
           quantiles.labels= c("Q5", "Q50", "Q95"), main=NULL,
           ylab="Probability", col="blue", leg.cex=1.2, leg.pos="bottomright",
           cex.axis=1.2, cex.main=1.2, cex.lab=1.2, verbose=TRUE, ...)
```

Arguments

`sim`	matrix or data.frame with the simulated equivalents obtained with different parameter sets, which, by default, are stored in columns
`weights`	numeric vector, values of the weights to be used for computing the quantiles Omitting the `weights` argument or specifying `NULL` or a zero-length vector will result in the usual un-weighted estimates
`byrow`	logical, indicates whether the computations have to be made for each column or for each row of `x` When the simulated equivalents are stored in columns, `byrow` must be *`TRUE`* When the simulated equivalents are stored in rows, `byrow` must be *`FALSE`*
`quantiles.desired`	numeric vector, quantiles to be computed. Default values are *`c(.025, .5, .975)`* (=> 2.5%, 50%, 97.5%)
`plot`	logical, indicates if a plot with the ECDFs has to be produced
`obs`	OPTIONAL. Only used when `plot=TRUE` Numeric or zoo object with observed values, which are used in the output plot

`quantiles.labels`
: OPTIONAL. Only used when `plot=TRUE`
character vector, names to `quantiles.desired`. Default value is *`c("Q5", "Q50", "Q95")`*

`main`
: OPTIONAL. Only used when `plot=TRUE`
title for the plot

`ylab`
: OPTIONAL. Only used when `plot=TRUE`
title for the y axis. See `plot`

`col`
: OPTIONAL. Only used when `plot=TRUE`
specification for the default plotting colour. See `par`

`leg.cex`
: OPTIONAL. Only used when `plot=TRUE`
character expansion factor *relative* to current 'par("cex")'. Used for text, and provides the default for 'pt.cex' and 'title.cex' Default value = 1.2

`leg.pos`
: OPTIONAL. Only used when `plot=TRUE`
keyword to be used to position the legend. See `legend`

`cex.axis`
: OPTIONAL. Only used when `plot=TRUE`
numeric, magnification to be used for the axis annotation relative to 'cex'. See `par`

`cex.main`
: OPTIONAL. Only used when `plot=TRUE`
numeric, representing the magnification to be used for main titles relative to the current setting of `cex`

`cex.lab`
: OPTIONAL. Only used when `plot=TRUE`
numeric, representing the magnification to be used for x and y labels relative to the current setting of 'cex'. See `par`

`verbose`
: logical, if TRUE, progress messages are printed

`...`
: further arguments passed to the `plot` function or from other methods

Details

Steps used in this function are:
1) Computation of un-weighted quantiles (e.g., Q5, Q50, Q95) for the simulated equivalents
2) Computation of ECDFs for each desired quantile, by weighting the quantiles of each parameter set by its corresponding weights (or less-formal likelihood in GLUE terminology)

Value

A list whose elements `x` and `ecdf` correspond to unique sorted values of `sim`. If the first CDF estimate is greater than zero, a point (min(sim),0) is placed at the beginning of the estimates

A list, with as many elements as the number of elements of `quantiles.desired`
Each element of the list, has the following components:

`x`
: *`n+1`* elements (with *`n`* as the number of columns of `x`)

`ecdf`
: Description of 'comp2'

Note

It requires the `wtd.Ecdf` function from the **Hmisc** package.

Author(s)

Mauricio Zambrano-Bigiarini, <mzb.devel@gmail.com>

See Also

wtd.Ecdf, params2ecdf

Examples

```
# random matrix with 100 simulated values (in columns) corresponding to 10
# different behavioural parameter sets
x <- matrix(rnorm(1000), ncol=10, nrow=100)

# empirical CDFs for the quantiles 0.05, 0.5 and 0.95, with equal weight for
# each parameter set
quant2ecdf(sim=x, weights=1:10, byrow=FALSE)
```

`rch2zoo` *Function for reading SWAT-2005 *.rch files*

Description

Function for reading the 'output.rch' files of SWAT-2005 and producing a zoo object, with the columns of the read file

This 'output.rch' file has 9 rows representing the header, and 1 column with the text 'REACH', and 43 additional columns with results regarding water quantity, sediments, and water quality

Usage

```
rch2zoo(file = "output.rch", out.type = "Q", rchID = 1,
        col.name = "FLOW_OUTcms", Date.Ini = "1962-01-01",
        Date.Fin = "1990-12-31", date.fmt = "%Y-%m-%d",
        tstep = "daily", verbose = TRUE)
```

Arguments

`file`	character, name of the file where data are stored. If it does not contain an _absolute_ path, the file name is _relative_ to the current working directory, 'getwd()'. Tilde-expansion is performed where supported
`out.type`	character, type of results to be read: -) "Q": only results related to water quantity (first 8 columns): c("RCH", "GIS", "MON", "DrAREAkm2", "FLOW_INcms", "FLOW_OUTcms", "EVAPcms", "TLOSScms") -) "Q+Sed": only results related to water quantity AND sediments (first 11 columns): The previously mentioned 8 along with c("SED_INtons", "SED_OUTtons", "SEDCONCmg/kg") -) "Q+Sed+WQ": all the columns of the 'output.rch'
`rchID`	OPTIONAL. Integer, number of the reach for which the results will be provided If this argument is not provided, the results will be given for all the reaches in 'output.rch'
`col.name`	character, column name in 'file' storing the results to be converted into a zoo object
`Date.Ini`	character, starting date for the results in `file`

Date.Fin	character, ending date for the results in file
date.fmt	character, format used to define Date.Ini, Date.Fin. See format argument in as.Date
tstep	character, time step of the results in file. Valid values are in c('daily', 'monthly', 'annual')
verbose	logical; if TRUE, progress messages are printed

Author(s)

Mauricio Zambrano-Bigiarini, <mzb.devel@gmail.com>

See Also

SWAT2R package

ReadPlot_convergence

Reading/Plotting the values of different parameter sets

Description

This function reads a file containing different parameter sets and ther corresponfing goodness-of-fit values

Usage

```
read_convergence(file="ConvergenceMeasures.txt", MinMax=NULL, beh.thr=NA,
     verbose=TRUE, plot=TRUE, col=c("black", "darkolivegreen"), lty=c(1,3),
     lwd=c(2,2), main="Gbest & Normalized Swarm Radius vs Iteration Number",
     xlab="Iteration Number", ylab=c("Gbest", expression(delta[norm])),
     pch=c(15, 18), cex=1, cex.main=1.4, cex.axis=1.2, cex.lab=1.2,
     legend.pos="topright", ..., do.png=FALSE, png.width=1500, png.height=900,
     png.res=90,png.fname="ConvergenceMeasures.png")

plot_convergence(x, verbose=TRUE, col=c("black", "darkolivegreen"), lty=c(1,3),
     lwd=c(2,2), main="Gbest & Normalized Swarm Radius vs Iteration Number",
     xlab="Iteration Number", ylab=c("Gbest", expression(delta[norm])),
     pch=c(15, 18), cex=1, cex.main=1.4, cex.axis=1.2, cex.lab=1.2,
     legend.pos="topright", ...,  do.png=FALSE, png.width=1500, png.height=900,
     png.res=90, png.fname="ConvergenceMeasures.png")
```

Arguments

file	character, name (including path) of the file to be read
verbose	logical; if TRUE, progress messages are printed
x	data.frame with the convergence outputs obtained with read_convergence.
MinMax	OPTIONAL character, indicates if the optimum value in params corresponds to the minimum or maximum of the the objective function. Valid values are in: c('min', 'max')

beh.thr	numeric, used for selecting only the behavioural parameter sets, i.e., those with a goodness-of-fit value larger/lowervalue than beh.th, depending on the value of MinMax. It is only used for drawing a horizontal line used for separating behavioural from non behavioural parameter sets.
plot	logical, indicates if a plot with the convergence measures has to be produced
col	OPTIONAL. Only used when plot=TRUE character, colour to be used for drawing the lines
lty	OPTIONAL. Only used when plot=TRUE numeric, line type to be used
lwd	OPTIONAL. Only used when plot=TRUE numeric, line width
xlab	OPTIONAL. Only used when plot=TRUE character, label for the 'x' axis
ylab	OPTIONAL. Only used when plot=TRUE character, label for the 'y' axis
main	OPTIONAL. Only used when plot=TRUE character, title for the plot
pch	OPTIONAL. Only used when plot=TRUE numeric, type of symbol for drawing the points of the dotty plots (e.g., 1: white circle)
cex	OPTIONAL. Only used when plot=TRUE numeric, values controlling the size of text and points with respect to the default
cex.main	OPTIONAL. Only used when plot=TRUE numeric, magnification to be used for main titles relative to the current setting of cex
cex.axis	OPTIONAL. Only used when plot=TRUE numeric, magnification to be used for axis annotation relative to the current setting of cex
cex.lab	OPTIONAL. Only used when plot=TRUE numeric, magnification to be used for x and y labels relative to the current setting of cex
legend.pos	OPTIONAL. Only used when plot=TRUE character, position of the legend. Valid values are in c("bottomright", "bottom", "bottomleft", "left", "topleft", "top", "topright", "right", "center"). See legend
...	OPTIONAL. Only used when plot=TRUE further arguments passed to the plot command or from other methods
do.png	logical, indicates if the plot with the convergence measures has to be saved into a PNG file instead of the screen device
png.width	OPTIONAL. Only used when do.png=TRUE numeric, width of the device. See png
png.height	OPTIONAL. Only used when do.png=TRUE numeric, height of the device. See png
png.res	OPTIONAL. Only used when do.png=TRUE numeric, nominal resolution in ppi which will be recorded in the PNG file, if a positive integer of the device. See png
png.fname	OPTIONAL. Only used when do.png=TRUE character, name of the output PNG file. See png

Value

A list with the following elements:

`Iter`	Description of 'comp1'
`Gbest`	Description of 'comp2'
`GbestRate`	Description of 'comp2'
`IterBestFit`	Description of 'comp2'
`normSwarmRadius`	Description of 'comp2'
`[gbest-mean(pbest)]/mean(pbest)`	Description of 'comp2'

Author(s)

Mauricio Zambrano-Bigiarini, <mzb.devel@gmail.com>

See Also

read_results, plot_results

Examples

```
# Setting the user home directory as working directory
setwd("~")

# Number of dimensions to be optimised
nparam <- 4

## Not run:

# Setting the seed
set.seed(100)

# Runing PSO with the 'sphere' test function, writting the results to text files
hydroPSO(
        fn= "sphere", lower=rep(-100, nparam), upper=rep(100, nparam),
        control=list(MinMax="min", npart=2*nparam, maxit=2000,
                     write2disk=TRUE, plot=TRUE)
        )

# Reading the convergence measures got by running hydroPSO
setwd("PSO.out")
read_convergence()


## End(Not run)
```

`ReadPlot_GofPerParticle` *plotParticlesGof*

Description

This function reads/plots the parameter values of each particle and the objective function against the iteration number

Usage

```
read_GofPerParticle(file="Particles_GofPerIter.txt", na.strings="NA",
     plot=TRUE, ptype="one", nrows="auto", main=NULL,
     xlab="Number of Iterations", cex=0.4, cex.main=1.5, cex.axis=1.7,
     cex.lab=1.5, col, lty=3, ylim, verbose=TRUE, do.png=FALSE,
     png.width=1500, png.height=900, png.res=90,
     png.fname="Particles_GofPerIter.png")

plot_GofPerParticle(x, ptype="one", nrows="auto", main=NULL,
     xlab="Number of Iterations", cex=0.4, cex.main=1.5, cex.axis=1.7,
     cex.lab=1.5, col=rainbow(ncol(x)), lty=3, ylim=NULL, verbose=TRUE, ...,
     do.png=FALSE, png.width=1500, png.height=900, png.res=90,
     png.fname="Particles_GofPerIter.png")
```

Arguments

`file`	character, name (including path) of the file to be read
`na.strings`	character vector, strings which are to be interpreted as `NA` values. See `read.table`
`plot`	logical, indicates if a plot with the convergence measures has to be produced
`x`	data.frame with the goodness-of-fit measure of each particle per iteration. The number of columns in `x` has to be equal to the number of particles, whereas the number of rows in `x` has to be equal to the number of iterations ((`ncol(x)` = number of particles ; `nrow(x)` = number of iterations)
`ptype`	character, representing the type of plot. Valid values are: in c("one", "many"), for plotting all the particles in the smae figure or in one windows per particle, respectively
`nrows`	OPTIONAL. Only used when `plot=TRUE` numeric, number of rows to be used in the plotting window If `nrows`is set to *`auto`*, the number of rows is automatically computed depending on the number of columns of `x`
`main`	OPTIONAL. Only used when `plot=TRUE` character, title for the plot
`xlab`	OPTIONAL. Only used when `plot=TRUE` character, label for the 'x' axis
`cex`	OPTIONAL. Only used when `plot=TRUE` numeric, values controlling the size of text and points with respect to the default
`cex.main`	OPTIONAL. Only used when `plot=TRUE` numeric, magnification for main titles relative to the current setting of `cex`

cex.axis	OPTIONAL. Only used when plot=TRUE numeric, magnification for axis annotation relative to the current setting of cex
cex.lab	OPTIONAL. Only used when plot=TRUE numeric, magnification for x and y labels relative to the current setting of cex
col	OPTIONAL. Only used when plot=TRUE character, colour to be used for drawing the lines
lty	OPTIONAL. Only used when plot=TRUE numeric, line type to be used
ylim	numeric with the the 'y' limits of the plot
verbose	logical, if TRUE, progress messages are printed
...	OPTIONAL. Only used when plot=TRUE further arguments passed to the plot command or from other methods
do.png	logical, indicates if all the figures have to be saved into PNG files instead of the screen device
png.width	OPTIONAL. Only used when do.png=TRUE numeric, width of the PNG device. See png
png.height	OPTIONAL. Only used when do.png=TRUE numeric, height of the PNG device. See png
png.res	OPTIONAL. Only used when do.png=TRUE numeric, nominal resolution in ppi which will be recorded in the PNG file, if a positive integer of the device. See png
png.fname	OPTIONAL. Only used when do.png=TRUE character, filename used to store the PNG file wih the dotty plots of the parameter values

Author(s)

Mauricio Zambrano-Bigiarini, <mzb.devel@gmail.com>

See Also

read_results, plot_results, plot_2parOF, plot_NparOF, plot_ParamsPerIter

Examples

```
# Setting the user home directory as working directory
setwd("~")

# Number of dimensions to be optimised
nparam <- 4

## Not run:

# Setting the seed
set.seed(100)

# Runing PSO with the 'sphere' test function, writting the results to text files
hydroPSO(
        fn= "sphere", lower=rep(-100, nparam),  upper=rep(100, nparam),
        control=list(MinMax="min", n.particles=10, maxit=100, topology="gbest",
                     boundary.wall="reflecting", write2disk=TRUE, plot=TRUE)
```

```
        )

# Reading the convergence measures got by running hydroPSO
setwd("PSO.out")
read_GofPerParticle() # all the particles in the same window
read_GofPerParticle(ptype="many") # each particle in a different pannel


## End(Not run)
```

`ReadPlot_out` *Reading/Plotting the 'Model_out.txt' output file of hydroPSO*

Description

This function reads the values of the objective function/model output for each particle and iteration with optional plot

Usage

```
read_out(file="Model_out.txt", modelout.cols=NULL, obs, MinMax=NULL, beh.thr=NA,
     verbose=TRUE, plot=TRUE, ptype=c("corr", "ts", "ecdf", "quant2ecdf"),
     ftype="dm", FUN=mean, weights=NULL, byrow=TRUE,
     quantiles.desired= c(0.05,0.5,0.95), quantiles.labels= c("Q5", "Q50", "Q95"
     main=NULL, ylab="Probability", col="blue", leg.cex=1.2, leg.pos="bottomrigh
     cex.axis=1.2, cex.main=1.2, cex.lab=1.2, do.png=FALSE, png.width=1500,
     png.height=900, png.res=90, png.fname="ModelOut_vs_Obs.png")
plot_out(sim, obs, dates=NULL, ptype=c("corr", "ts", "ecdf", "quant2ecdf"),
     MinMax=NULL, ftype="dm", FUN=mean, verbose=TRUE, weights=NULL, byrow=TRUE,
     quantiles.desired= c(0.05,0.5,0.95), quantiles.labels=c("Q5","Q50","Q95"),
     main=NULL, ylab="Probability", col="blue", leg.cex=1.2,
     leg.pos="bottomright", cex.axis=1.2, cex.main=1.2, cex.lab=1.2,
     do.png=FALSE, png.width=1500, png.height=900, png.res=90,
     png.fname="ModelOut_vs_Obs.png")
```

Arguments

`file`	character, name (including path) of the output file with the values of the model / objective function for each particle and iteration
`modelout.cols`	numeric, column number in `file` that store the outputs that have to be read/plotted, without counting the first three that correspond to iteration, particle and GoF. If `modelout.cols=NULL`, all the columns in `will be read`, but the first trhee that contains the iteration number, the particle number and the corresponding goodness-of-fit.
`sim`	numeric or zoo vector, simulated equivalent values of the model / objective funtion to be compared against observations
`obs`	OPTIONAL. numeric or zoo vector, observations to be compared against the best simulated value. If `obs` is not provided, its values are read from the output 'Observations.txt' file in the results directory (by default 'PSO.out')

dates — OPTIONAL. character or Date object used to assign time stamps to each element of sim and obs. If sim and/or obs already have a time stamp, it is over-written by dates
It must have the same length of sim and obs numeric or zoo vectors

MinMax — OPTIONAL. character, indicates whether the optimum value corresponds to the minimum or maximum of the the objective function. It is used to filter out model outputs with a non-acceptable performance
Valid values are in: c('min', 'max')

beh.thr — OPTIONAL. numeric, used for selecting only the behavioural parameter sets, i.e. those with a goodness-of-fit value larger/lower than beh.th, depending on the value of MinMax
It is used for drawing a horizontal line used for separating behavioural from non behavioural parameter sets

verbose — logical, if TRUE, progress messages are printed

plot — logical, indicates if a plot with the convergence measures has to be produced

ptype — character, type of plot. Valid values are:
-) *corr*: Scatterplot between the observed values and its best simulated counterpart
-) *ts*: Only possible for observed values of zoo type. A graphical comparison between observed values and its best simulated counterpart along time. It requires the **hydroGOF** package. See ggof
-) *ecdf*: Empirical CDFs computed and plotted for each column of sim
-) *quant2ecdf*: For each model output corresponding to a different parameter set (in rows or columns of sim, according to the value of byrow), different quantiles are computed (as many as indicated in quantiles.desired, and then Empirical CDFs are computed and plotted for each one of the previous quantiles)

ftype — OPTIONAL. Only used when plot=TRUE and ptype=="ts".See ggof

FUN — OPTIONAL. Only used when plot=TRUE and ptype=="ts".See ggof

weights — numeric vector, values of the weights to be used for computing the quantiles. See quant2ecdf
Omitting the weights argument or specifying NULL or a zero-length vector will result in the usual un-weighted estimates

byrow — logical, indicates whether the computations have to be made for each column or for each row of x. See quant2ecdf When the simulated equivalents are stored in columns, byrow must be *TRUE*
When the simulated equivalents are stored in rows, byrow must be *FALSE*

quantiles.desired — numeric vector, quantiles to be computed for model outputs. Default values are *c(.025, .5, .975)* (=> 2.5%, 50%, 97.5%). See quant2ecdf

quantiles.labels — OPTIONAL. Only used when plot=TRUE
character vector, names to quantiles.desired. Default value is *c("Q5", "Q50", "Q95")*. See quant2ecdf

main — OPTIONAL. Only used when plot=TRUE
title for the plot

ylab — OPTIONAL. Only used when plot=TRUE
title for the y axis. See plot

col	OPTIONAL. Only used when plot=TRUE specification for the default plotting colour. See par
leg.cex	OPTIONAL. Only used when plot=TRUE character expansion factor *relative* to current 'par("cex")'. Used for text, and provides the default for 'pt.cex' and 'title.cex' Default value = 1.2
leg.pos	OPTIONAL. Only used when plot=TRUE keyword to be used to position the legend. See legend
cex.axis	OPTIONAL. Only used when plot=TRUE numeric, magnification to be used for the axis annotation relative to 'cex'. See par
cex.main	OPTIONAL. Only used when plot=TRUE numeric, representing the magnification to be used for main titles relative to the current setting of cex
cex.lab	OPTIONAL. Only used when plot=TRUE numeric, representing the magnification to be used for x and y labels relative to the current setting of 'cex'. See par
do.png	logical, indicates if the plot with the comparison between model outputs and observations has to be saved into a PNG file instead of the screen device
png.width	OPTIONAL. Only used when do.png=TRUE numeric, width of the device. See png
png.height	OPTIONAL. Only used when do.png=TRUE numeric, height of the device. See png
png.res	OPTIONAL. Only used when do.png=TRUE numeric, nominal resolution in ppi which will be recorded in the PNG file, if a positive integer of the device. See png
png.fname	OPTIONAL. Only used when do.png=TRUE character, name of the output PNG file. See png

Value

list with three elements:

model.values	matrix/data.frame (or numeric) with the values of the model / objective function for each particle and iteration
model.gofs	numeric vector with the goodness-of-fit value for each row (or value) in 'model.values'
model.best	numeric with the best model / objective function value. In order to be computed, the user has to provide a valid value for MinMax
model.obs	numeric with the observed values used during the optimisation. See obs

Author(s)

Mauricio Zambrano-Bigiarini, <mzb.devel@gmail.com>

See Also

read_results, plot_results, quant2ecdf

Examples

```
# Setting the user home directory as working directory
setwd("~")

# Number of dimensions to be optimised
nparam <- 5

## Not run:

# Setting the seed
set.seed(100)

# Runing PSO with the 'sphere' test function, writting the results to text files
hydroPSO(
        fn= "sphere", lower=rep(-100, nparam), upper=rep(100, nparam),
        control=list(MinMax="min", npart=2*nparam, maxit=100, topology="gbest",
                     write2disk=TRUE, plot=TRUE)
        )

# Reading the convergence measures got by running hydroPSO
setwd("PSO.out")
read_out(MinMax="min") # each particle in a different pannel

## End(Not run)
```

`ReadPlot_params` *Reading/Plotting the values of different parameter sets*

Description

This function reads a file containing different parameter sets and their corresponding goodness-of-fit values

If `file` is one of the following:
-) *`modelpara.out`*, created by the GLUE algorithm of SWAT-CUP,
-) *`modelpara.beh`*, created by the GLUE algorithm of SWAT-CUP,
-) *`goal.sf2`*, created by the SUFI-2 algorithm of SWAT-CUP
-) *`goal.pso`*, created by the PSO algorithm of SWAT-CUP
-) *`ParameterValues.log`*, created by Nimbus calibration tool (Lisflood model)

`header` and `skip` are automatically set, in other case, they need to be provided

Usage

```
read_params(file, ...)

## Default S3 method:
read_params(file, header=TRUE, skip=0, param.cols, param.names,
     of.col=NULL, of.name="GoF", na.strings="-9999", plot=TRUE,
     ptype=c("histogram", "dottyplot", "boxplot", "vioplot", "pairs"),
     MinMax=NULL, beh.thr=NA, beh.col="red", beh.lty=1, beh.lwd=2,
     nrows="auto", col="#00000030", ylab=of.name, main=NULL, pch=19,
```

```
     cex=0.5, cex.main=1.5, cex.axis=1.5, cex.lab=1.5,
     breaks="Scott", freq=TRUE, verbose=TRUE, ..., do.png=FALSE, png.width=1500,
     png.height=900, png.res=90, png.fname="Parameters.png")

plot_params(params, ...)

## Default S3 method:
plot_params(params, gofs=NULL,
     ptype=c("histogram", "dottyplot", "boxplot", "vioplot", "pairs"),
     param.cols=1:ncol(params), param.names=colnames(params), of.name="GoF",
     MinMax=NULL, beh.thr=NA,  beh.col="red", beh.lty=1, beh.lwd=2,
     nrows="auto", col="#00000030", ylab=of.name, main=NULL, pch=19, cex=0.5,
     cex.main=1.5, cex.axis=1.5, cex.lab=1.5, breaks="Scott", freq=TRUE,
     verbose=TRUE, ..., do.png=FALSE, png.width=1500, png.height=900,
     png.res=90, png.fname="Parameters.png")

## S3 method for class 'data.frame'
plot_params(params, gofs=NULL,
     ptype=c("histogram", "dottyplot", "boxplot", "vioplot", "pairs"),
     param.cols=1:ncol(params), param.names=colnames(params), of.name="GoF",
     MinMax=NULL, beh.thr=NA,  beh.col="red", beh.lty=1, beh.lwd=2,
     nrows="auto", col="#00000030", ylab=of.name, main=NULL, pch=19, cex=0.5,
     cex.main=1.5, cex.axis=1.5, cex.lab=1.5, breaks="Scott", freq=TRUE,
     verbose=TRUE, ..., do.png=FALSE, png.width=1500, png.height=900,
     png.res=90, png.fname="Parameters.png")

## S3 method for class 'matrix'
plot_params(params, gofs=NULL,
     ptype=c("histogram", "dottyplot", "boxplot", "vioplot", "pairs"),
     param.cols=1:ncol(params), param.names=colnames(params), of.name="GoF",
     MinMax=NULL, beh.thr=NA,  beh.col="red", beh.lty=1, beh.lwd=2,
     nrows="auto", col="#00000030", ylab=of.name, main=NULL, pch=19, cex=0.5,
     cex.main=1.5, cex.axis=1.5, cex.lab=1.5, breaks="Scott", freq=TRUE,
     verbose=TRUE, ..., do.png=FALSE, png.width=1500, png.height=900,
     png.res=90, png.fname="Parameters.png")
```

Arguments

`file`	character, name (including path) of the file containing the results
`params`	data.frame whose rows represent the values of different parameter sets
`gofs`	OPTIONAL. numeric with the values of goodness-of-fit values for each one of the parameters in `params` (in the same order!)
`header`	logical, indicates whether the file contains the names of the variables as its first line If `file` is in *`c('modelpara.out', 'modelpara.beh', 'goal.sf2', 'goal.pso', 'ParameterValues.log')`* then `header` is automatically set
`skip`	numeric (integer), lines of the data file to skip before beginning to read data If `file` is in *`c('modelpara.out', 'modelpara.beh', 'goal.sf2', 'goal.pso', 'ParameterValues.log')`* then `skip` is automatically set

param.cols	numeric, number of the columns in file that store the values of each parameter
param.names	character, name of the parameters defined by param.cols
of.col	OPTIONAL. numeric, number of the column in file that store the values of objective function
of.name	OPTIONAL. Only used when of.col is provided. character, name that will be given to the column of.col
na.strings	character, string which is to be interpreted as NA values. read.table
plot	logical, indicates if a dotty-plot with the parameter values versus the objective function has to be produced
ptype	OPTIONAL. Only used when plot=TRUE character, indicating the type of plot to be done. It must be in: -) *dottyplot*: dotty plots for each parameter in params or file, with the value of the objective function against the parameter value -) *histogram*: histogram for each parameter in params or file, with an estimate of the probability distribution each parameter -) *boxplot*: box plots (or box-and-whisker diagram) for each parameter in params or file, with a graphical summary of the distribution of each parameter, through their five-number summary -) *vioplot*: beanplots for each parameter in params or file, similar to the boxplots, except that beanplots also show the probability density of the data at different values. See vioplot. It requires the **vioplot** package. -) *pairs*: Visualization of a correlation matrix among the parameters and goodness-of-fits measures in params (or file) and gofs. See hydropairs. It requires the **hydroTSM** package.
MinMax	OPTIONAL character, indicates whether the optimum value in params corresponds to the minimum or maximum of the the objective function given in of.col. It is used to filter out model outputs with a non-acceptable performance Valid values are in: c('min', 'max')
beh.thr	OPTIONAL numeric, value for drawing a horizontal line for separating behavioural from non behavioural parameter sets
beh.col	OPTIONAL. Only used when plot=TRUE character, colour for drawing a horizontal line for separating behavioural from non behavioural parameter sets
beh.lty	OPTIONAL. Only used when plot=TRUE numeric, line type for drawing a horizontal line for separating behavioural from non behavioural parameter sets
beh.lwd	OPTIONAL. Only used when plot=TRUE numeric, width for drawing a horizontal line for separating behavioural from non behavioural parameter sets
nrows	OPTIONAL. Only used when plot=TRUE numeric, number of rows to be used in the plotting window If nrowsis set to *auto*, the number of rows is automatically computed depending on the number of columns of params
col	OPTIONAL. Only used when plot=TRUE character, colour to be used for drawing the points of the dotty plots
ylab	OPTIONAL. Only used when plot=TRUE character, label for the 'y' axis

main	chracter, title for the plot
pch	OPTIONAL. Only used when plot=TRUE numeric, type of symbol to be used for drawing the points of the dotty plots (e.g., 1: white circle)
cex	OPTIONAL. Only used when plot=TRUE numeric, values controlling the size of text and points with respect to the default
cex.main	OPTIONAL. Only used when plot=TRUE numeric, magnification for the main title relative to the current setting of cex
cex.axis	OPTIONAL. Only used when plot=TRUE numeric, magnification for axis annotation relative to the current setting of cex
cex.lab	OPTIONAL. Only used when plot=TRUE numeric, magnification for x and y labels relative to the current setting of cex
breaks	breaks used for plotting the histograms of the parameter sets. See hist
freq	logical, if TRUE, the histogram graphic is a representation of frequencies, the counts component of the result; if FALSE, probability densities, component density, are plotted (so that the histogram has a total area of one). See hist
verbose	logical, if TRUE, progress messages are printed
...	OPTIONAL. Only used when plot=TRUE further arguments passed to the plot command or from other methods
do.png	logical, indicates if the plot with the convergence measures has to be saved into a PNG file instead of the screen device
png.width	OPTIONAL. Only used when do.png=TRUE numeric, width of the device. See png
png.height	OPTIONAL. Only used when do.png=TRUE numeric, height of the device. See png
png.res	OPTIONAL. Only used when do.png=TRUE numeric, nominal resolution in ppi which will be recorded in the PNG file, if a positive integer of the device. See png
png.fname	OPTIONAL. Only used when do.png=TRUE character, name of the output PNG file. See png

Value

A list with the following elements:

params	data.frame with the parameter sets tested during the optimisation
gofs	numeric with the fitness values computed during the optimisation (each element in 'gofs' corresponds to one row of 'params')

Author(s)

Mauricio Zambrano-Bigiarini, <mzb.devel@gmail.com>

See Also

vioplot

Examples

```
# Number of dimensions to be optimised
nparam <- 4

## Not run:
# Setting the user home directory as working directory
setwd("~")

# Setting the seed
set.seed(100)

# Runing PSO with the 'sphere' test function, writting the results to text files
hydroPSO(
        fn= "sphere", lower=rep(-100, nparam), upper=rep(100, nparam),
        control=list(npart=2*nparam, maxit=100, write2disk=TRUE, plot=TRUE)
        )

# 1) reading ALL the parameter sets used in PSO, and histograms (by default)
params <- read_params(file="~/PSO.out/Particles.txt",  param.cols=4:7, of.col=3)

# 2) summary of the parameter sets and their goodness-

# plotting the parameter sets as dotty plots
plot_params(params=params[["params"]], gofs=params[["gofs"]],
            ptype="dottyplot", main=fn, MinMax="min", freq=TRUE)

# plotting the parameter sets as boxplots
plot_params(params=params[["params"]], ptype="boxplot", MinMax="min")

# plotting the parameter sets as violing plots
library(vioplot)
plot_params(params=params[["params"]], ptype="vioplot", MinMax="min")


# 2) reading only the parameter sets with a goodness-of-fit measure <= 'beh.thr',
#    and dotty plots (by default)
params <- read_params(file="~/PSO.out/Particles.txt", param.cols=4:7, of.col=3,
                      beh.thr=1000, MinMax="min")

## End(Not run)
```

`ReadPlot_particles` *Reading/Plotting the 'Particles.txt' output file*

Description

The function `read_particles` reads the 'Particles.txt' output file, which stores all the parameter sets tested during the optimisation along with their corresponding goodness-of-fit values

The function `plot_particles` takes the parameter sets and their corresponding goodness-of-fit value, read by `read_particles`, and produces the following plots:
1) Dotty plots
2) Histograms

3) Boxplots
4) Correlation matrix
5) Empirical CDFs
6) Parameter values vs Number of Model Evaluations
7) (pseudo) 3D dotty plots

Usage

```
read_particles(file="Particles.txt", verbose=TRUE, plot=TRUE,
     gof.name="GoF", MinMax=NULL, beh.thr=NA, beh.col="red", beh.lty=1,
     beh.lwd=2, nrows="auto", col="black", ylab=gof.name, main=NULL, pch=19,
     cex=0.5, cex.main=1.5, cex.axis=1.5, cex.lab=1.5,

     breaks="Scott", freq=TRUE, dp3D.names="auto", GOFcuts="auto",
     colorRamp= colorRampPalette(c("darkred", "red", "orange", "yellow",
     "green", "darkgreen", "cyan")), alpha=1, points.cex=0.7,legend.pos="topleft
     do.png=FALSE, png.width=1500, png.height=900, png.res=90,
     dotty.png.fname="Params_DottyPlots.png",
     hist.png.fname="Params_Histograms.png",
     bxp.png.fname="Params_Boxplots.png",
     ecdf.png.fname="Params_ECDFs.png",
     runs.png.fname="Params_ValuesPerRun.png",
     dp3d.png.fname="Params_dp3d.png",
     pairs.png.fname="Params_Pairs.png")

plot_particles(params, gofs, gof.name="GoF", MinMax=NULL, beh.thr=NA,
     beh.col="red", beh.lty=1, beh.lwd=2, nrows="auto", col="black",
     ylab=gof.name, main=NULL, pch=19, cex=0.5, cex.main=1.5,
     cex.axis=1.5, cex.lab=1.5,

     breaks="Scott", freq=TRUE,
     weights=NULL, byrow=FALSE, leg.cex=1.5,
     dp3D.names="auto", GOFcuts="auto",
     colorRamp= colorRampPalette(c("darkred", "red", "orange", "yellow",
     "green", "darkgreen", "cyan")), alpha=1, points.cex=0.7,
     legend.pos="topleft", verbose=TRUE,
     do.png=FALSE, png.width=1500, png.height=900, png.res=90,
     dotty.png.fname="Params_DottyPlots.png",
     hist.png.fname="Params_Histograms.png",
     bxp.png.fname="Params_Boxplots.png",
     ecdf.png.fname="Params_ECDFs.png",
     runs.png.fname="Params_ValuesPerRun.png",
     dp3d.png.fname="Params_dp3d.png",
     pairs.png.fname="Params_Pairs.png")

read_velocities(file="Velocities.txt", ... )
```

Arguments

`file`	character, name (including path) of the output file with the position and fitness value of each particle and for each iteration
`params`	data.frame whose rows represent the values of different parameter sets

gofs	OPTIONAL. numeric with the values of goodness-of-fit values for each parameter in params (in the same order!)
verbose	logical, if TRUE, progress messages are printed
plot	logical, indicates if the following figures has to be produced: dotty plots, histograms, empirical CDFs, Parameter Values Against Number of Model Evaluations, and 3D dotty plots of Parameter Values
gof.name	character, name to be given to the goodness-of-fit values in all the plots
MinMax	OPTIONAL. character, indicates if the optimum value in params corresponds to the minimum or maximum of the the objective function. Only used to identify the optimum in the plot Valid values are in: c('min', 'max')
beh.thr	numeric, used for selecting only the behavioural parameter sets, i.e., those with a goodness-of-fit value larger/lowervalue than beh.th, depending on the value of MinMax. It is only used for drawing a horizontal line used for separating behavioural from non behavioural parameter sets.
beh.col	OPTIONAL. Only used when plot=TRUE character, colour for drawing a horizontal line for separating behavioural from non behavioural parameter sets
beh.lty	OPTIONAL. Only used when plot=TRUE numeric, line type for drawing a horizontal line for separating behavioural from non behavioural parameter sets
beh.lwd	OPTIONAL. Only used when plot=TRUE numeric, width for drawing a horizontal line for separating behavioural from non behavioural parameter sets
nrows	OPTIONAL. Only used when plot=TRUE numeric, number of rows to be used in the plotting window If nrowsis set to *auto*, the number of rows is automatically computed depending on the number of columns of params
col	OPTIONAL. Only used when plot=TRUE character, colour for drawing the points of the dotty plots
ylab	OPTIONAL. Only used when plot=TRUE character, label for the 'y' axis
main	OPTIONAL. Only used when plot=TRUE character, title for the plot
pch	OPTIONAL. Only used when plot=TRUE numeric, type of symbol to be used for drawing the points of the dotty plots (e.g., 1: white circle)
cex	OPTIONAL. Only used when plot=TRUE numeric, values controlling the size of text and points with respect to the default
cex.main	OPTIONAL. Only used when plot=TRUE numeric, magnification for main titles relative to the current setting of cex
cex.axis	OPTIONAL. Only used when plot=TRUE numeric, magnification for axis annotation relative to the current setting of cex
cex.lab	OPTIONAL. Only used when plot=TRUE numeric, magnification for x and y labels relative to the current setting of cex
...	OPTIONAL. Only used when plot=TRUE further arguments passed to the plot command or from other methods

`breaks` OPTIONAL. Only used when `plot=TRUE`
breaks for plotting the histograms of the parameter sets. See `hist`

`freq` OPTIONAL. Only used when `plot=TRUE`
logical, if TRUE, the histogram graphic is a representation of frequencies, the counts component of the result; if FALSE, probability densities, component density, are plotted (so that the histogram has a total area of one). Defaults to TRUE if and only if breaks are equidistant (and probability is not specified). See `hist`

`weights` OPTIONAL. Only used when `plot=TRUE`
numeric vector, values of the weights to be used for computing the empirical CDFs. See `params2ecdf`

`byrow` OPTIONAL. Only used when `plot=TRUE`
logical, indicates whether the computations have to be made for each column or for each row of `params`. See `params2ecdf`

`leg.cex` OPTIONAL. Only used when `plot=TRUE`
character expansion factor *relative* to current 'par("cex")'. Used for text, and provides the default for 'pt.cex' and 'title.cex'. Default value = 1.2

`dp3D.names` character, name of all the parameters (usually only the most sensitive ones) that will be used for plotting pseudo-3D plots
If `dp3D.names='auto'` half the number of parameters in `file` are chosen randomly for plotting. See `plot_NparOF`

`GOFcuts` numeric, specifies at which values of the objective function `gof.name` the colours of the plot have to change. See `plot_NparOF`

`colorRamp` R function defining the colour ramp to be used for colouring the pseudo-3D dotty plots of Parameter Values, OR character representing those colours. See `plot_NparOF`

`alpha` numeric between 0 and 1 representing the transparency level to apply to the colors of the pseudo-3D dotty plots. See `plot_NparOF`

`points.cex` size of the points to be plotted

`legend.pos` not used yet ...

`do.png` logical, indicates if the plot with the convergence measures has to be saved into a PNG file instead of the screen device

`png.width` OPTIONAL. Only used when `do.png=TRUE`
numeric, width of the device. See `png`

`png.height` OPTIONAL. Only used when `do.png=TRUE`
numeric, height of the device. See `png`

`png.res` OPTIONAL. Only used when `do.png=TRUE`
numeric, nominal resolution in ppi which will be recorded in the PNG file, if a positive integer of the device. See `png`

`dotty.png.fname`
OPTIONAL. Only used when `do.png=TRUE`
character, filename used to store the PNG file with the dotty plots of the parameter values

`hist.png.fname`
OPTIONAL. Only used when `do.png=TRUE`
character, filename used to store the PNG file with the histograms of the parameter values

bxp.png.fname
: OPTIONAL. Only used when do.png=TRUE
character, filename used to store the PNG file with the boxplots of the parameter values

ecdf.png.fname
: OPTIONAL. Only used when do.png=TRUE
character, filename used to store the PNG file with the empirical CDFs of the parameter values

runs.png.fname
: OPTIONAL. Only used when do.png=TRUE
character, filename used to store the PNG file with the parameter values vs the number of model evaluations

dp3d.png.fname
: OPTIONAL. Only used when do.png=TRUE
character, filename used to store the PNG file with the pseudo-3D plots of all the parameters defined in dp3D.names

pairs.png.fname
: OPTIONAL. Only used when do.png=TRUE
character, filename used to store the PNG file with the correlation matrix among the parameters and goodness-of-fits measures in params and gofs. See plot_params and hydropairs

Value

read_particles returns a list with four elements:

part.params
: numeric or matrix/data.frame with the parameter values for each particle and iteration

part.gofs
: numeric vector with the goodness-of-fit value for each particle and iteration

best.param
: numeric with the parameter values of the best particle. In order to be computed, the user has to provide a valid value for MinMax

best.gof
: numeric with the best godness-of-fit value among all the particles. In order to be computed, the user has to provide a valid value for MinMax

Author(s)

Mauricio Zambrano-Bigiarini, <mzb.devel@gmail.com>

See Also

read_results, plot_results, read_params, plot_params

Examples

```
# Setting the user home directory as working directory
setwd("~")

# Number of dimensions to be optimised
nparam <- 4

## Not run:

# Setting the seed
```

```
set.seed(100)

# Runing PSO with the 'sphere' test function, writting the results to text files
hydroPSO(
        fn= "sphere", lower=rep(-100, nparam), upper=rep(100, nparam),
        control=list(npart=2*nparam, maxit=100, write2disk=TRUE, plot=TRUE)
        )

# reading the 'Particles.txt' output file of PSO, and plotting dotty plots,
# histograms, eCDFs, ...
setwd("PSO.out")
particles <- read_particles()

# reading only the particles in 'Particles.txt' with a goodness-of-fit value
# lower than 'beh.thr'
particles <- read_particles(beh.thr=1000, MinMax="min")

## End(Not run)
```

`ReadPlot_results` *Reading/Plotting all the output files generated by 'hydroPSO'*

Description

The function `read_results` reads the following output files of hydroPSO:

1) 'BestParameterSet.txt': best parameter set and its corresponding goodness-of-fit found during the optimisation
2) 'Particles.txt': parameter values and their corresponding goodness-of-fit for all particles and iterations
3) 'Velocities.txt': velocity values and their corresponding goodness-of-fit for all particles and iterations
4) 'Model_out.txt': values of the objective function/model output for each particle and iteration
5) 'ConvergenceMeasures.txt': convergence measures summarizing performance of hydroPSO
6) 'Particles_GofPerIter.txt': goodness-of-fit only for all the particles during all the iterations

The function `plot_results` takes the outputs of the `read_results` function and then produces the following plots:

1) Dotty plots of parameter values
2) Histograms of parameter values
3) Boxplots of parameter values
4) Correlation matrix among parameter values
5) Empirical CDFs of parameter values
6) Parameter values vs Number of Model Evaluations
7) (pseudo) 3D dotty plots of (selected) parameter values
8) GoF for each particle against Number of Model Evaluations
9) Velocity values vs Number of Model Evaluations
10a) Scatterplot between Best Simulated values and Observations (OPTIONAL, only if `MinMax` is provided)
10b) Empirical CDFs for model's output (only produced if `obs` is NOT a zoo object)

10b) ggof (See ggof) between Best Simulated values and Observations (OPTIONAL, only if obs is a zoo object)
10d) Empirical CDFs for selected quantiles of model's output (OPTIONAL, only if obs is a zoo object)
11) Convergence Measures (Gbest and normSwarmRadius) vs Iteration Number

Usage

```
read_results(drty.out="PSO.out", MinMax=NULL, beh.thr=NA, modelout.cols=NULL, ve

plot_results(drty.out="PSO.out", param.names, gof.name="GoF", MinMax=NULL,
     beh.thr=NA, beh.col="red", beh.lty=1, beh.lwd=2, nrows="auto",
     col="black", ylab=gof.name, main=NULL, pch=19, cex=0.5, cex.main=1.7,
     cex.axis=1.3, cex.lab=1.5, breaks="Scott", freq=TRUE,
     weights=NULL, byrow=FALSE, leg.cex=1.2,


     dp3D.names="auto", GOFcuts="auto",
     colorRamp= colorRampPalette(c("darkred", "red", "orange", "yellow",
     "green", "darkgreen", "cyan")), alpha=1, points.cex=0.7,

     ptype="one",

     modelout.cols=NULL,
     ftype="dm", FUN=mean,
     quantiles.desired= c(0.05,0.5,0.95), quantiles.labels= c("Q5","Q50","Q95"),

     legend.pos="topright",

     do.png=FALSE, png.width=1500, png.height=900, png.res=90,
     dotty.png.fname="Params_DottyPlots.png",
     hist.png.fname ="Params_Histograms.png",
     bxp.png.fname="Params_Boxplots.png",
     ecdf.png.fname ="Params_ECDFs.png",
     pruns.png.fname="Params_ValuesPerRun.png",
     dp3d.png.fname ="Params_dp3d.png",
     pairs.png.fname="Params_Pairs.png",
     part.png.fname ="Particles_GofPerIter.png",
     vruns.png.fname="Velocities_ValuePerRun.png",
     modelout.best.png.fname="ModelOut_BestSim_vs_Obs.png",
     modelout.quant.png.fname="ModelOut_Quantiles.png",
     conv.png.fname ="ConvergenceMeasures.png", verbose=TRUE)
```

Arguments

drty.out	character, path to the directory storing the output files generated by *hydroPSO*
param.names	character, names for the parameters in params that have to be plotted (param.names can be a subset of params). Names for each parameter are taken from the first row of the 'Particles.txt' file
verbose	logical, if TRUE, progress messages are printed
gof.name	character, name of the goodness-of-fit variable in all plots

`MinMax` OPTIONAL. character, indicates whether the optimum value in `x` corresponds to the minimum or maximum of the objective function. It is only used to identify the optimum on the plots
Valid values are in: `c('min', 'max')`

`beh.thr` OPTIONAL. numeric, threshold to filter out parameter sets and model outputs with a non-acceptable performance (non behavioural parameter sets)

`modelout.cols` numeric, column number in `file` that store the outputs that have to be read/plotted, without counting the first three that correspond to iteration, particle and GoF. If `modelout.cols=NULL`, all the columns in `will be read`, but the first trhee that contains the iteration number, the particle number and the corresponding goodness-of-fit. See `read_out`

`beh.col` OPTIONAL. Only used when `plot=TRUE`
character, colour for drawing a horizontal line for separating behavioural from non behavioural parameter sets

`beh.lty` OPTIONAL. Only used when `plot=TRUE`
numeric, line type for drawing a horizontal line for separating behavioural from non behavioural parameter sets

`beh.lwd` OPTIONAL. Only used when `plot=TRUE`
numeric, width for drawing a horizontal line for separating behavioural from non behavioural parameter sets

`nrows` OPTIONAL. Only used when `plot=TRUE`
numeric, number of rows to be used in the plotting window
If `nrows`is set to *`auto`*, the number of rows is automatically computed depending on the number of columns of `x`

`col` OPTIONAL. Only used when `plot=TRUE`
character, colour to be used for drawing the points of the dotty plots

`ylab` OPTIONAL. Only used when `plot=TRUE`
character, label for the 'y' axis

`main` OPTIONAL. Only used when `plot=TRUE`
character, title for the plot

`pch` OPTIONAL. Only used when `plot=TRUE`
numeric, type of symbol to be used for drawing the points of the dotty plots. (e.g., 1: white circle)

`cex` OPTIONAL. Only used when `plot=TRUE`
numeric, values controlling the size of text and points with respect to the default

`cex.main` OPTIONAL. Only used when `plot=TRUE`
numeric, magnification for main titles relative to the current setting of `cex`

`cex.axis` OPTIONAL. Only used when `plot=TRUE`
numeric, magnification for axis annotation relative to the current setting of `cex`

`cex.lab` OPTIONAL. Only used when `plot=TRUE`
numeric, magnification for x and y labels relative to the current setting of `cex`

`breaks` OPTIONAL. Only used when `plot=TRUE`
breaks for plotting the histograms of the parameter sets. See `hist`

`freq` OPTIONAL. Only used when `plot=TRUE`
logical, if TRUE, the histogram graphic is a representation of frequencies, the counts component of the result; if FALSE, probability densities, component density, are plotted (so that the histogram has a total area of one). Defaults to TRUE if and only if breaks are equidistant (and probability is not specified). See `hist`

`weights` OPTIONAL. Only used when `plot=TRUE`
numeric vector, values of the weights to be used for computing the empirical CDFs. See `params2ecdf`

`byrow` OPTIONAL. Only used when `plot=TRUE`
logical, indicates whether the computations have to be made for each column or for each row of `x`. See `params2ecdf`

`leg.cex` OPTIONAL. Only used when `plot=TRUE`
character expansion factor *relative* to current 'par("cex")'. Used for text, and provides the default for 'pt.cex' and 'title.cex'. Default value = 1.2

`dp3D.names` character, name for all the parameters (usually only the most sensitive ones) that will be used for plotting pseudo-3D plots
If `dp3D.names='auto'` half the number of parameters in `file` are chosen randomly for plotting. See `plot_NparOF`

`GOFcuts` numeric, specifies at which values of the objective function `gof.name` the colours of the plot have to change. See `plot_NparOF`

`colorRamp` R function defining the colour ramp to be used for colouring the pseudo-3D dotty plots of Parameter Values, OR character representing those colours. See `plot_NparOF`

`alpha` numeric between 0 and 1 representing the transparency level to apply to the colors of the pseudo-3D dotty plots. See `plot_NparOF`

`points.cex` size of the points to be plotted

`ptype` character, represents the type of plot. Valid values are: in c("one", "many"), for plotting all the particles in the same figure or in one windows per particle, respectively
See `plot_GofPerParticle`

`ftype` OPTIONAL. Only used when `plot=TRUE` and the observed values provided by the user were zoo objects. See `plot_out` and `ggof`.

`FUN` OPTIONAL. Only used when `plot=TRUE` and the observed values provided by the user were zoo objects. See `plot_out` and `ggof`

`quantiles.desired`
numeric vector, quantiles to be computed. Default values are *`c(.025, .5, .975)`* (=> 2.5%, 50%, 97.5%). See `plot_out`

`quantiles.labels`
OPTIONAL. Only used when `plot=TRUE`
character vector, names to `quantiles.desired`. Default value is *`c("Q5", "Q50", "Q95")`*. See `plot_out`

`legend.pos` See `plot_convergence`

`do.png` logical, indicates if all the figures have to be saved into PNG files instead of the screen device

`png.width` OPTIONAL. Only used when `do.png=TRUE`
numeric, width of the PNG device. See `png`

`png.height` OPTIONAL. Only used when `do.png=TRUE`
numeric, height of the PNG device. See `png`

`png.res` OPTIONAL. Only used when `do.png=TRUE`
numeric, nominal resolution in ppi which will be recorded in the PNG file, if a positive integer of the device. See `png`

`dotty.png.fname`
: OPTIONAL. Only used when `do.png=TRUE`
character, filename used to store the PNG file with the dotty plots of the parameter values.

`hist.png.fname`
: OPTIONAL. Only used when `do.png=TRUE`
character, filename used to store the PNG file with the histograms of the parameter values.

`bxp.png.fname`
: OPTIONAL. Only used when `do.png=TRUE`
character, filename used to store the PNG file with the boxplots of the parameter values

`ecdf.png.fname`
: OPTIONAL. Only used when `do.png=TRUE`
character, filename used to store the PNG file with the empirical CDFs of the parameter values.

`pruns.png.fname`
: OPTIONAL. Only used when `do.png=TRUE`
character, filename used to store the PNG file with the parameter values vs the number of model evaluations

`dp3d.png.fname`
: OPTIONAL. Only used when `do.png=TRUE`
character, filename used to store the PNG file with the pseudo-3D plots of all the parameters defined in `dp3D.names`

`pairs.png.fname`
: OPTIONAL. Only used when `do.png=TRUE`
character, filename used to store the PNG file with the correlation matrix among the parameters and goodness-of-fits measures in `params` and `gofs`. See `plot_particles` and `hydropairs`

`part.png.fname`
: OPTIONAL. Only used when `do.png=TRUE`
character, filename used to store the PNG file with the goodness-of-fit for all the particles along the iterations

`vruns.png.fname`
: OPTIONAL. Only used when `do.png=TRUE`
character, filename used to store the PNG file with the velocity values vs the number of model evaluations

`modelout.best.png.fname`
: OPTIONAL. Only used when `do.png=TRUE`
character, filename used to store the PNG file with the observed values against its best simulated counterpart. See `plot_out`

`modelout.quant.png.fname`
: OPTIONAL. Only used when `do.png=TRUE`
character, filename used to store the PNG file with some quantiles of simulated values against its observed counterparts. See `plot_out`

`conv.png.fname`
: OPTIONAL. Only used when `do.png=TRUE`
character, filename used to store the PNG file with the convergence measures. See `plot_convergence`

Value

The function `read_results` returns a list with the following elements:

`best.param`	numeric with the best parameter set
`best.gof`	numeric with the best fitness value of the objective function
`params`	data.frame with all the parameter sets tested during the optimisation
`gofs`	numeric with all the fitness values computed during the optimisation (each element in `gofs` corresponds to one row of `params`)
`model.values`	numeric or matrix/data.frame with the values of the objective function / model for each particle and iteration. See `read_out`
`model.best`	numeric with the best model / objective function value. In order to be computed, the user has to provide a valid value for `MinMax`. See `read_out`
`model.obs`	numeric with the observed values used during the optimisation. See `obs`
`convergence.measures`	matrix/data.frame with the convergence measures. See `read_convergence` function
`part.GofPerIter`	matrix/data.frame with the goodness-of-fit only for all the particles during all the iterations

Author(s)

Mauricio Zambrano-Bigiarini, `<mzb.devel@gmail.com>`

See Also

`read_best`, `read_particles`, `read_velocities`, `read_out`, `read_convergence`, `read_GofPerParticle`, `plot_ParamsPerIter`

Examples

```
# Setting the user home directory as working directory
setwd("~")

# Number of dimensions to be optimised
nparam <- 5

## Not run:

# Setting the seed
set.seed(100)

# Runing PSO with the 'ackley' test function, writting the results to text files
hydroPSO(
        fn= "ackley", method="pso", lower=rep(-32, nparam), upper=rep(32, nparam),
        control=list(MinMax="min", npart=2*nparam, maxit=2000, topology="gbest",
                     write2disk=TRUE, REPORT=100,
                     use.RG=TRUE, RG.thr=1e-2, RG.r=1e-5)
        )

# Reading all the results and storing them in a variable
res <- read_results()
```

```
# Plotting all the results with a goodness-of-fit lower than 5
plot_results(MinMax="min", beh.thr=5)

## End(Not run)

## Not run:
##########################################################################
####################  SPSO-2007 example START ############################
##########################################################################
# Number of dimensions to be optimised
nparam <- 10

# boundaries for the test function
x <- c(-100, 100)      # "sphere"
#x <- c(-5.12, 5.12)   # "rastrigrin"
#x <- c(-32, 32)       # "ackley"

fn <- "sphere"
#fn <- "rastrigrin"
#fn <- "ackley"

########################################
#####   SPSO-2007 parameters   ########
npart  <- floor(10+2*sqrt(nparam))
c1     <- 0.5+log(2)
c2     <- 0.5+log(2)
abstol <- 1e-20
reltol <- 1e-20
maxit  <- 1000

use.IW        <- TRUE
IW.w          <- 1/(2*log(2))
REPORT        <- 100
lambda        <- 1
boundary.wall <- "absorbing"
########################################

# Setting the user home directory as working directory
setwd("~")

# Runing PSO  and writting the results to text files
set.seed(100)

hydroPSO(
        fn= fn, method="pso", lower=rep(x[1], nparam), upper=rep(x[2], nparam),
        control=list(MinMax="min", npart=npart,
                     c1=c1, c2=c2,
                     use.IW=use.IW, IW.w=IW.w,
                     maxit=maxit, topology="random", lambda=lambda, K=3,
                     Xini.type="random", Vini.type="random",
                     best.update="async",
                     boundary.wall=boundary.wall,
                     write2disk=TRUE, plot=FALSE, REPORT=REPORT,
                     abstol=abstol, reltol=reltol
                     )
        )
```

```
# Plotting all the results
plot_results(MinMax="min")

##########################################################################
####################  SPSO-2007 example END   ############################
##########################################################################

## End(Not run)
```

`read_best` *Reading the 'BestParameterSet.txt' output file*

Description

This function reads the contents of the the 'BestParameterSet.txt' output file, which stores the best parameter set and its corresponding goodness-of-fit value found during the optimisation

Usage

```
read_best(file="BestParameterSet.txt", verbose=TRUE)
```

Arguments

`file`	character, name (including path) of the output file with the best parameter set and its corresponding best fitness value found during the optimisation
`verbose`	logical, if TRUE, progress messages are printed

Author(s)

Mauricio Zambrano-Bigiarini, <mzb.devel@gmail.com>

See Also

`read_results`, `plot_results`

Examples

```
# Setting the user home directory as working directory
setwd("~")

# Number of dimensions to be optimised
nparam <- 4

## Not run:

# Setting the seed
set.seed(100)

# Runing PSO with the 'sphere' test function, writting the results to text files
hydroPSO(
        fn= "sphere", lower=rep(-100, nparam),  upper=rep(100, nparam),
        control=list(MinMax="min", npart=2*nparam, maxit=100, topology="gbest",
```

```
                        write2disk=TRUE, plot=TRUE)
        )

# Reading the best parameter set and its corresponsing gof found by hydroPSO
setwd("PSO.out")
read_best()

## End(Not run)
```

`test_functions` *Test Functions for Global Optimisation*

Description

Test functions commonly used as benchmark for global optimisation problems

Usage

```
ackley(x)
griewank(x)
sphere(x)
rastrigrin(x)
rosenbrock(x)
schafferF6(x)
```

Arguments

`x`	numeric vector to be evaluated

Author(s)

Mauricio Zambrano-Bigiarini, <mzb.devel@gmail.com>

References

http://www.zsd.ict.pwr.wroc.pl/files/docs/functions.pdf

http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/TestGO_files/Page364.htm

http://www.geatbx.com/docu/fcnindex-01.html

verification *verification*

Description

Run the model and get a goodness-of-fit value by comparing the simulated values against observations for the optimum parameter set found by optimisation

Usage

```
verification(fn="hydromod", par, control=list(),
             model.FUN=NULL, model.FUN.args=list() )
```

Arguments

fn	character, name of a valid R function to be optimised or character value *'hydromod'*. When fn='hydromod' the algorithm uses model.FUN and model.FUN.args to extract the values simulated by the model and to compute its corresponding goodness-of-fit function. When fn!='hydromod' the algorithm uses the value(s) returned by fn as both model output and its corresponding goodness-of-fit When fn='hydromod' the algorithm will optimise the model defined by model.FUN and model.args
par	numeric, or matrix/data.frame with the parameter sets that will be used for verification Parameter sets in par must be stored by row, i.e., each different row represents a different parameter set
control	a list of control parameters. See 'Details'
model.FUN	OPTIONAL. Only used when fn='hydromod' character, valid R function representing the model code to be calibrated/optimised
model.FUN.args	OPTIONAL. Only used when fn='hydromod' 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, path to 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 *hydroPSO*

digits OPTIONAL. Only used when write2disk=TRUE
numeric, number of significant digits used for writing the outputs in scientific notation

gof.name character, ONLY used for identifying the goodness-of-fit of each model run and writing it to the *LH_OAT-gof.txt* output file

MinMax character, indicates whether the optimum value for the analysed problem corresponds to the minimum or maximum of the the objective function. It is used to select the 'best' parameter set. Valid values are in: c('min', 'max')

do.plots logical, if TRUE a PNG plot with the comparison between observed and simulated values is produced for each parameter set used in the LH-OAT

write2disk logical, indicates if the output files will be written to the disk

verbose logical, if TRUE progress messages are printed

Value

A list of two elements:

`gofs`	goodness-of-fit values corresponding to each one of the parameter sets provided in `par`
`best.gof`	goodness-of-fit of the "best" parameter set found during the verification round
`best.par`	parameter values of the "best" paraemter set found during the verification round

Author(s)

Mauricio Zambrano-Bigiarini, <mzb.devel@gmail.com>

See Also

`hydromod`

---

wquantile *Weighted Quantiles*

Description

This function computes weighted quantiles of each column (by default, or for each row if specified by the user) of a matrix/data.frame

It is a wrapper to the `wtd.quantile` function of the **Hmisc** package, specially thought for a matrix containing streamflows simulated by different (behavioural) parameter sets

Usage

```
wquantile(x, weights=NULL, byrow=FALSE, probs=c(.025, .5, .975),
          normwt=TRUE, verbose=TRUE)
```

Arguments

`x`	numeric or matrix for the computation of the weighted quantiles
`weights`	numeric vector, values of the weights to be used for computing the quantiles. See `wtd.quantile`. Omitting the `weights` argument or specifying `NULL` or a zero-length vector will result in the usual unweighted estimates

byrow	logical, indicates if the computations have to be made for each column or for each row of x When the simulated values obtained with different behavioural parameter sets are stored in columns, byrow must be *TRUE* When the simulated values obtained with different behavioural parameter sets are stored in rows, byrow must be *FALSE*
probs	numeric vector, quantiles to be computed. wtd.quantile Default value is *c(.025, .5, .975)* (=> 2.5%, 50%, 97.5%)
normwt	See wtd.quantile. Specify normwt=TRUE to make weights sum to length(x) after deletion of NAs
verbose	logical; if TRUE, progress messages are printed

Author(s)

Mauricio Zambrano-Bigiarini, <mzb.devel@gmail.com>

See Also

wtd.quantile

Examples

```
# random matrix with 100 parameter sets (in rows) corresponding to 10
# different parameters
params <- matrix(rnorm(1000), ncol=10, nrow=100)
colnames(params) <- paste("Param", 1:10, sep="")

# empirical CDFs for each one of the 10 parameters of x, with equal weight for
# each one of the 100 parameter sets
wquantile(params, weights=rep(1,100), byrow=FALSE)
```

Index