Package: hydroGOF (via r-universe)

November 5, 2024

Type Package

Title Goodness-of-Fit Functions for Comparison of Simulated and Observed Hydrological Time Series

Version 0.6-0.1

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

Description S3 functions implementing both statistical and graphical goodness-of-fit measures between observed and simulated values, mainly oriented to be used during the calibration, validation, and application of hydrological models. Missing values in observed and/or simulated values can be removed before computations. Comments / questions / collaboration of any kind are very welcomed.

License GPL (>= 2)

Depends R (>= 2.10.0), zoo (>= 1.7-2)

Imports hydroTSM (>= 0.5-0), xts (>= 0.8-2), methods, stats

Suggests knitr, rmarkdown

VignetteBuilder knitr

URL https://github.com/hzambran/hydroGOF

MailingList https://stat.ethz.ch/mailman/listinfo/r-sig-ecology

BugReports https://github.com/hzambran/hydroGOF/issues

LazyLoad yes

NeedsCompilation no

Repository CRAN

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

Date/Publication 2024-11-04 16:55:25 UTC

Contents

`hydroGOF-package` *Goodness-of-fit (GoF) functions for numerical and graphical comparison of simulated and observed time series, mainly focused on hydrological modelling.*

Description

S3 functions implementing both statistical and graphical goodness-of-fit measures between observed and simulated values, to be used during the calibration, validation, and application of hydrological models.

Missing values in observed and/or simulated values can be removed before computations.

Details

Package:	hydroGOF
Type:	Package
Version:	0.6-0
Date:	2024-05-08
License:	GPL >= 2
LazyLoad:	yes
Packaged:	Wed 08 May 2024 05:13:53 PM -04 ; MZB
BuiltUnder:	R version 4.4.0 (2024-04-24) – "Puppy Cup" ;x86_64-pc-linux-gnu (64-bit)

Quantitative statistics included in this package are:

`me` Mean Error
`mae` Mean Absolute Error
`mse` Mean Squared Error
`rmse` Root Mean Square Error
`ubRMSE` Unbiased Root Mean Square Error
`nrmse` Normalized Root Mean Square Error
`pbias` Percent Bias
`rsr` Ratio of RMSE to the Standard Deviation of the Observations
`rSD` Ratio of Standard Deviations
`NSE` Nash-Sutcliffe Efficiency
`mNSE` Modified Nash-Sutcliffe Efficiency
`rNSE` Relative Nash-Sutcliffe Efficiency
`wNSE` Weighted Nash-Sutcliffe Efficiency
`wsNSE` Weighted Seasonal Nash-Sutcliffe Efficiency
`d` Index of Agreement
`dr` Refined Index of Agreement
`md` Modified Index of Agreement

rd Relative Index of Agreement
cp Persistence Index
rPearson Pearson correlation coefficient
R2 Coefficient of determination
br2 R2 multiplied by the coefficient of the regression line between sim and obs
VE Volumetric efficiency
KGE Kling-Gupta efficiency
KGElf Kling-Gupta Efficiency for low values
KGEnp Non-parametric version of the Kling-Gupta Efficiency
KGEkm Knowable Moments Kling-Gupta Efficiency
sKGE Split Kling-Gupta Efficiency
APFB Annual Peak Flow Bias
HFB High Flow Bias
rSpearman Spearman's rank correlation coefficient
ssq Sum of the Squared Residuals
pbiasfdc PBIAS in the slope of the midsegment of the flow duration curve
pfactor P-factor
rfactor R-factor

Author(s)

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

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

References

Abbaspour, K.C.; Faramarzi, M.; Ghasemi, S.S.; Yang, H. (2009), Assessing the impact of climate change on water resources in Iran, Water Resources Research, 45(10), W10,434, doi:10.1029/2008WR007615.

Abbaspour, K.C., Yang, J. ; Maximov, I.; Siber, R.; Bogner, K.; Mieleitner, J. ; Zobrist, J.; Srinivasan, R. (2007), Modelling hydrology and water quality in the pre-alpine/alpine Thur watershed using SWAT, Journal of Hydrology, 333(2-4), 413-430, doi:10.1016/j.jhydrol.2006.09.014.

Box, G.E. (1966). Use and abuse of regression. Technometrics, 8(4), 625-629. doi:10.1080/00401706.1966.10490407.

Barrett, J.P. (1974). The coefficient of determination-some limitations. The American Statistician, 28(1), 19-20. doi:10.1080/00031305.1974.10479056.

Chai, T.; Draxler, R.R. (2014). Root mean square error (RMSE) or mean absolute error (MAE)? - Arguments against avoiding RMSE in the literature, Geoscientific Model Development, 7, 1247-1250. doi:10.5194/gmd-7-1247-2014.

Cinkus, G.; Mazzilli, N.; Jourde, H.; Wunsch, A.; Liesch, T.; Ravbar, N.; Chen, Z.; and Goldscheider, N. (2023). When best is the enemy of good - critical evaluation of performance criteria in hydrological models. Hydrology and Earth System Sciences 27, 2397-2411, doi:10.5194/hess-27-2397-2023.

Criss, R. E.; Winston, W. E. (2008), Do Nash values have value? Discussion and alternate proposals. Hydrological Processes, 22: 2723-2725. doi:10.1002/hyp.7072.

Entekhabi, D.; Reichle, R.H.; Koster, R.D.; Crow, W.T. (2010). Performance metrics for soil moisture retrievals and application requirements. Journal of Hydrometeorology, 11(3), 832-840. doi: 10.1175/2010JHM1223.1.

Fowler, K.; Coxon, G.; Freer, J.; Peel, M.; Wagener, T.; Western, A.; Woods, R.; Zhang, L. (2018). Simulating runoff under changing climatic conditions: A framework for model improvement. Water Resources Research, 54(12), 812-9832. doi:10.1029/2018WR023989.

Garcia, F.; Folton, N.; Oudin, L. (2017). Which objective function to calibrate rainfall-runoff models for low-flow index simulations?. Hydrological sciences journal, 62(7), 1149-1166. doi:10.1080/02626667.2017.1308511.

Garrick, M.; Cunnane, C.; Nash, J.E. (1978). A criterion of efficiency for rainfall-runoff models. Journal of Hydrology 36, 375-381. doi:10.1016/0022-1694(78)90155-5.

Gupta, H.V.; Kling, H.; Yilmaz, K.K.; Martinez, G.F. (2009). Decomposition of the mean squared error and NSE performance criteria: Implications for improving hydrological modelling. Journal of hydrology, 377(1-2), 80-91. doi:10.1016/j.jhydrol.2009.08.003. ISSN 0022-1694.

Gupta, H.V.; Kling, H. (2011). On typical range, sensitivity, and normalization of Mean Squared Error and Nash-Sutcliffe Efficiency type metrics. Water Resources Research, 47(10). doi:10.1029/2011WR010962.

Hahn, G.J. (1973). The coefficient of determination exposed. Chemtech, 3(10), 609-612. Aailable online at: `https://www2.hawaii.edu/~cbaajwe/Ph.D.Seminar/Hahn1973.pdf`.

Hodson, T.O. (2022). Root-mean-square error (RMSE) or mean absolute error (MAE): when to use them or not, Geoscientific Model Development, 15, 5481-5487, doi:10.5194/gmd-15-5481-2022.

Hundecha, Y., Bardossy, A. (2004). Modeling of the effect of land use changes on the runoff generation of a river basin through parameter regionalization of a watershed model. Journal of hydrology, 292(1-4), 281-295. doi:10.1016/j.jhydrol.2004.01.002.

Kitanidis, P.K.; Bras, R.L. (1980). Real-time forecasting with a conceptual hydrologic model. 2. Applications and results. Water Resources Research, Vol. 16, No. 6, pp. 1034:1044. doi:10.1029/WR016i006p01034.

Kling, H.; Fuchs, M.; Paulin, M. (2012). Runoff conditions in the upper Danube basin under an ensemble of climate change scenarios. Journal of Hydrology, 424, 264-277, doi:10.1016/j.jhydrol.2012.01.011.

Knoben, W.J.; Freer, J.E.; Woods, R.A. (2019). Inherent benchmark or not? Comparing Nash-Sutcliffe and Kling-Gupta efficiency scores. Hydrology and Earth System Sciences, 23(10), 4323-4331. doi:10.5194/hess-23-4323-2019.

Krause, P.; Boyle, D.P.; Base, F. (2005). Comparison of different efficiency criteria for hydrological model assessment, Advances in Geosciences, 5, 89-97. doi:10.5194/adgeo-5-89-2005.

Krstic, G.; Krstic, N.S.; Zambrano-Bigiarini, M. (2016). The br2-weighting Method for Estimating the Effects of Air Pollution on Population Health. Journal of Modern Applied Statistical Methods, 15(2), 42. doi:10.22237/jmasm/1478004000

Legates, D.R.; McCabe, G. J. Jr. (1999), Evaluating the Use of "Goodness-of-Fit" Measures in Hydrologic and Hydroclimatic Model Validation, Water Resour. Res., 35(1), 233-241. doi:10.1029/1998WR900018.

Ling, X.; Huang, Y.; Guo, W.; Wang, Y.; Chen, C.; Qiu, B.; Ge, J.; Qin, K.; Xue, Y.; Peng, J. (2021). Comprehensive evaluation of satellite-based and reanalysis soil moisture products using in situ observations over China. Hydrology and Earth System Sciences, 25(7), 4209-4229. doi:10.5194/hess-25-4209-2021.

Mizukami, N.; Rakovec, O.; Newman, A.J.; Clark, M.P.; Wood, A.W.; Gupta, H.V.; Kumar, R.: (2019). On the choice of calibration metrics for "high-flow" estimation using hydrologic models, Hydrology Earth System Sciences 23, 2601-2614, doi:10.5194/hess-23-2601-2019.

Moriasi, D.N.; Arnold, J.G.; van Liew, M.W.; Bingner, R.L.; Harmel, R.D.; Veith, T.L. (2007). Model evaluation guidelines for systematic quantification of accuracy in watershed simulations. Transactions of the ASABE. 50(3):885-900

Nash, J.E. and Sutcliffe, J.V. (1970). River flow forecasting through conceptual models. Part 1: a discussion of principles, Journal of Hydrology 10, pp. 282-290. doi:10.1016/0022-1694(70)90255-6.

Pearson, K. (1920). Notes on the history of correlation. Biometrika, 13(1), 25-45. doi:10.2307/2331722.

Pfannerstill, M.; Guse, B.; Fohrer, N. (2014). Smart low flow signature metrics for an improved overall performance evaluation of hydrological models. Journal of Hydrology, 510, 447-458. doi:10.1016/j.jhydrol.2013.12.0

Pizarro, A.; Jorquera, J. (2024). Advancing objective functions in hydrological modelling: Integrating knowable moments for improved simulation accuracy. Journal of Hydrology, 634, 131071. doi:10.1016/j.jhydrol.2024.131071.

Pool, S.; Vis, M.; Seibert, J. (2018). Evaluating model performance: towards a non-parametric variant of the Kling-Gupta efficiency. Hydrological Sciences Journal, 63(13-14), pp.1941-1953. doi:/10.1080/02626667.2018.1552002.

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

Santos, L.; Thirel, G.; Perrin, C. (2018). Pitfalls in using log-transformed flows within the KGE criterion. doi:10.5194/hess-22-4583-2018.

Schaefli, B., Gupta, H. (2007). Do Nash values have value?. Hydrological Processes 21, 2075-2080. doi:10.1002/hyp.6825.

Schober, P.; Boer, C.; Schwarte, L.A. (2018). Correlation coefficients: appropriate use and interpretation. Anesthesia and Analgesia, 126(5), 1763-1768. doi:10.1213/ANE.0000000000002864.

Schuol, J.; Abbaspour, K.C.; Srinivasan, R.; Yang, H. (2008b), Estimation of freshwater availability in the West African sub-continent using the SWAT hydrologic model, Journal of Hydrology, 352(1-2), 30, doi:10.1016/j.jhydrol.2007.12.025

Sorooshian, S., Q. Duan, and V. K. Gupta. (1993). Calibration of rainfall-runoff models: Application of global optimization to the Sacramento Soil Moisture Accounting Model, Water Resources Research, 29 (4), 1185-1194, doi:10.1029/92WR02617.

Spearman, C. (1961). The Proof and Measurement of Association Between Two Things. In J. J. Jenkins and D. G. Paterson (Eds.), Studies in individual differences: The search for intelligence (pp. 45-58). Appleton-Century-Crofts. doi:10.1037/11491-005

Tang, G.; Clark, M.P.; Papalexiou, S.M. (2021). SC-earth: a station-based serially complete earth dataset from 1950 to 2019. Journal of Climate, 34(16), 6493-6511. doi:10.1175/JCLI-D-21-0067.1.

Yapo P.O.; Gupta H.V.; Sorooshian S. (1996). Automatic calibration of conceptual rainfall-runoff models: sensitivity to calibration data. Journal of Hydrology. v181 i1-4. 23-48. doi:10.1016/0022-1694(95)02918-4

Yilmaz, K.K., Gupta, H.V. ; Wagener, T. (2008), A process-based diagnostic approach to model evaluation: Application to the NWS distributed hydrologic model, Water Resources Research, 44, W09417, doi:10.1029/2007WR006716.

Willmott, C.J. (1981). On the validation of models. Physical Geography, 2, 184–194. doi:10.1080/02723646.1981.10642213.

Willmott, C.J. (1984). On the evaluation of model performance in physical geography. Spatial Statistics and Models, G. L. Gaile and C. J. Willmott, eds., 443-460. doi:10.1007/978-94-017-3048-8_23.

Willmott, C.J.; Ackleson, S.G. Davis, R.E.; Feddema, J.J.; Klink, K.M.; Legates, D.R.; O'Donnell, J.; Rowe, C.M. (1985), Statistics for the Evaluation and Comparison of Models, J. Geophys. Res., 90(C5), 8995-9005. doi:10.1029/JC090iC05p08995.

Willmott, C.J.; Matsuura, K. (2005). Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance, Climate Research, 30, 79-82, doi:10.3354/cr030079.

Willmott, C.J.; Matsuura, K.; Robeson, S.M. (2009). Ambiguities inherent in sums-of-squares-based error statistics, Atmospheric Environment, 43, 749-752, doi:10.1016/j.atmosenv.2008.10.005.

Willmott, C.J.; Robeson, S.M.; Matsuura, K. (2012). A refined index of model performance. International Journal of climatology, 32(13), pp.2088-2094. doi:10.1002/joc.2419.

Willmott, C.J.; Robeson, S.M.; Matsuura, K.; Ficklin, D.L. (2015). Assessment of three dimensionless measures of model performance. Environmental Modelling & Software, 73, pp.167-174. doi:10.1016/j.envsoft.2015.08.012

Zambrano-Bigiarini, M.; Bellin, A. (2012). Comparing goodness-of-fit measures for calibration of models focused on extreme events. EGU General Assembly 2012, Vienna, Austria, 22-27 Apr 2012, EGU2012-11549-1.

See Also

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

Examples

```
obs <- 1:100
sim <- obs

# Numerical goodness of fit
gof(sim,obs)

# Reverting the order of simulated values
sim <- 100:1
gof(sim,obs)

## Not run:
ggof(sim, obs)

## End(Not run)

##################
# Loading daily streamflows of the Ega River (Spain), from 1961 to 1970
require(zoo)
data(EgaEnEstellaQts)
obs <- EgaEnEstellaQts
```

```
# Generating a simulated daily time series, initially equal to observations
sim <- obs

# Getting the numeric goodness-of-fit measures for the "best" (unattainable) case
gof(sim=sim, obs=obs)

# Randomly changing the first 2000 elements of 'sim', by using a normal
# distribution  with mean 10 and standard deviation equal to 1 (default of 'rnorm').
sim[1:2000] <- obs[1:2000] + rnorm(2000, mean=10)

# Getting the new numeric goodness of fit
gof(sim=sim, obs=obs)

# Graphical representation of 'obs' vs 'sim', along with the numeric
# goodness-of-fit measures
## Not run:
ggof(sim=sim, obs=obs)

## End(Not run)
```

APFB *Annual Peak Flow Bias*

Description

Annual peak flow bias between `sim` and `obs`, with treatment of missing values.

This function was prposed by Mizukami et al. (2019) to identify differences in high (streamflow) values. See Details.

Usage

```
APFB(sim, obs, ...)

## Default S3 method:
APFB(sim, obs, na.rm=TRUE, start.month=1, out.PerYear=FALSE,
             fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
             epsilon.value=NA)

## S3 method for class 'data.frame'
APFB(sim, obs, na.rm=TRUE, start.month=1, out.PerYear=FALSE,
             fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
             epsilon.value=NA)

## S3 method for class 'matrix'
APFB(sim, obs, na.rm=TRUE, start.month=1, out.PerYear=FALSE,
```

```
            fun=NULL, ...,
        epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
            epsilon.value=NA)

## S3 method for class 'zoo'
APFB(sim, obs, na.rm=TRUE, start.month=1, out.PerYear=FALSE,
            fun=NULL, ...,
        epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
            epsilon.value=NA)
```

Arguments

`sim` numeric, zoo, matrix or data.frame with simulated values

`obs` numeric, zoo, matrix or data.frame with observed values

`na.rm` a logical value indicating whether 'NA' should be stripped before the computation proceeds.
When an 'NA' value is found at the i-th position in `obs` **OR** `sim`, the i-th value of `obs` **AND** `sim` are removed before the computation.

`start.month` [OPTIONAL]. Only used when the (hydrological) year of interest is different from the calendar year.
numeric in [1:12] indicating the starting month of the (hydrological) year. Numeric values in [1, 12] represent months in [January, December]. By default `start.month=1`.

`out.PerYear` logical, indicating whether the output of this function has to include the annual peak flow bias obtained for the individual years or not.

`fun` function to be applied to `sim` and `obs` in order to obtain transformed values thereof before computing this goodness-of-fit index.
The first argument MUST BE a numeric vector with any name (e.g., `x`), and additional arguments are passed using `...`.

`...` arguments passed to `fun`, in addition to the mandatory first numeric vector.

`epsilon.type` argument used to define a numeric value to be added to both `sim` and `obs` before applying `fun`.
It is was designed to allow the use of logarithm and other similar functions that do not work with zero values.
Valid values of `epsilon.type` are:
1) `"none"`: `sim` and `obs` are used by `fun` without the addition of any numeric value. This is the default option.
2) `"Pushpalatha2012"`: one hundredth (1/100) of the mean observed values is added to both `sim` and `obs` before applying `fun`, as described in Pushpalatha et al. (2012).
3) `"otherFactor"`: the numeric value defined in the `epsilon.value` argument is used to multiply the the mean observed values, instead of the one hundredth (1/100) described in Pushpalatha et al. (2012). The resulting value is then added to both `sim` and `obs`, before applying `fun`.
4) `"otherValue"`: the numeric value defined in the `epsilon.value` argument is directly added to both `sim` and `obs`, before applying `fun`.

epsilon.value
: -) when epsilon.type="otherValue" it represents the numeric value to be added to both sim and obs before applying fun.
-) when epsilon.type="otherFactor" it represents the numeric factor used to multiply the mean of the observed values, instead of the one hundredth (1/100) described in Pushpalatha et al. (2012). The resulting value is then added to both sim and obs before applying fun.

Details

The annual peak flow bias (APFB; Mizukami et al., 2019) is designed to drive the calibration of hydrological models focused in the reproduction of high-flow events.

The high flow bias (APFB) ranges from 0 to Inf, with an optimal value of 0. Higher values of APFB indicate stronger differences between the high values of sim and obs. Essentially, the closer to 0, the more similar the high values of sim and obs are.

Value

If out.PerYear=FALSE: numeric with the mean annual peak flow bias between sim and obs. If sim and obs are matrices, the output value is a vector, with the mean annual peak flow bias between each column of sim and obs.

If out.PerYear=TRUE: a list of two elements:

APFB.value
: numeric with the mean annual peak flow bias between sim and obs. If sim and obs are matrices, the output value is a vector, with the mean annual peak flow bias between each column of sim and obs.

APFB.PerYear
: -) If sim and obs are not data.frame/matrix, the output is numeric, with the mean annual peak flow bias obtained for the individual years between sim and obs.
-) If sim and obs are data.frame/matrix, this output is a data.frame, with the mean annual peak flow bias obtained for the individual years between sim and obs.

Note

obs and sim has to have the same length/dimension

The missing values in obs and sim are removed before the computation proceeds, and only those positions with non-missing values in obs and sim are considered in the computation

Author(s)

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

References

Mizukami, N.; Rakovec, O.; Newman, A.J.; Clark, M.P.; Wood, A.W.; Gupta, H.V.; Kumar, R.: (2019). On the choice of calibration metrics for "high-flow" estimation using hydrologic models, Hydrology Earth System Sciences 23, 2601-2614, doi:10.5194/hess-23-2601-2019.

See Also

NSE, wNSE, wsNSE, HFB, gof, ggof

Examples

```
##################
# Example 1: Looking at the difference between 'NSE', 'wNSE', and 'APFB'
# Loading daily streamflows of the Ega River (Spain), from 1961 to 1970
data(EgaEnEstellaQts)
obs <- EgaEnEstellaQts

# Simulated daily time series, created equal to the observed values and then
# random noise is added only to high flows, i.e., those equal or higher than
# the quantile 0.9 of the observed values.
sim      <- obs
hQ.thr   <- quantile(obs, probs=0.9, na.rm=TRUE)
hQ.index <- which(obs >= hQ.thr)
hQ.n     <- length(hQ.index)
sim[hQ.index] <- sim[hQ.index] + rnorm(hQ.n, mean=mean(sim[hQ.index], na.rm=TRUE))

# Traditional Nash-Sutcliffe eficiency
NSE(sim=sim, obs=obs)

# Weighted Nash-Sutcliffe efficiency (Hundecha and Bardossy, 2004)
wNSE(sim=sim, obs=obs)

# APFB (Garcia et al., 2017):
APFB(sim=sim, obs=obs)

##################
# Example 2:
# Loading daily streamflows of the Ega River (Spain), from 1961 to 1970
data(EgaEnEstellaQts)
obs <- EgaEnEstellaQts

# Generating a simulated daily time series, initially equal to the observed series
sim <- obs

# Computing the 'APFB' for the "best" (unattainable) case
APFB(sim=sim, obs=obs)

##################
# Example 3: APFB for simulated values created equal to the observed values and then
#            random noise is added only to high flows, i.e., those equal or higher than
#            the quantile 0.9 of the observed values.

sim           <- obs
hQ.thr        <- quantile(obs, probs=0.9, na.rm=TRUE)
hQ.index      <- which(obs >= hQ.thr)
hQ.n          <- length(hQ.index)
sim[hQ.index] <- sim[hQ.index] + rnorm(hQ.n, mean=mean(sim[hQ.index], na.rm=TRUE))
ggof(sim, obs)
```

```
APFB(sim=sim, obs=obs)

##################
# Example 4: APFB for simulated values created equal to the observed values and then
#            random noise is added only to high flows, i.e., those equal or higher than
#            the quantile 0.9 of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' during computations.

APFB(sim=sim, obs=obs, fun=log)

# Verifying the previous value:
lsim <- log(sim)
lobs <- log(obs)
APFB(sim=lsim, obs=lobs)


##################
# Example 5: APFB for simulated values created equal to the observed values and then
#            random noise is added only to high flows, i.e., those equal or higher than
#            the quantile 0.9 of the observed values and applying a
#            user-defined function to 'sim' and 'obs' during computations

fun1 <- function(x) {sqrt(x+1)}

APFB(sim=sim, obs=obs, fun=fun1)

# Verifying the previous value, with the epsilon value following Pushpalatha2012
sim1 <- sqrt(sim+1)
obs1 <- sqrt(obs+1)
APFB(sim=sim1, obs=obs1)
```

---

br2 *br2*

Description

Coefficient of determination (r2) multiplied by the slope of the regression line between `sim` and `obs`, with treatment of missing values.

Usage

```
br2(sim, obs, ...)

## Default S3 method:
br2(sim, obs, na.rm=TRUE, use.abs=FALSE, fun=NULL, ...,
          epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
               epsilon.value=NA)
```

```
## S3 method for class 'data.frame'
br2(sim, obs, na.rm=TRUE, use.abs=FALSE, fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
             epsilon.value=NA)

## S3 method for class 'matrix'
br2(sim, obs, na.rm=TRUE, use.abs=FALSE, fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
             epsilon.value=NA)

## S3 method for class 'zoo'
br2(sim, obs, na.rm=TRUE, use.abs=FALSE, fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
             epsilon.value=NA)
```

Arguments

`sim` numeric, zoo, matrix or data.frame with simulated values

`obs` numeric, zoo, matrix or data.frame with observed values

`na.rm` logical value indicating whether 'NA' should be stripped before the computation proceeds.
When an 'NA' value is found at the i-th position in `obs` **OR** `sim`, the i-th value of `obs` **AND** `sim` are removed before the computation.

`use.abs` logical value indicating whether the condition to select the formula used to compute `br2` should be 'b<=1' or 'abs(b) <=1'.
Krausse et al. (2005) uses 'b<=1' as condition, but strictly speaking this condition should be 'abs(b)<=1'. However, if your model simulations are somewhat "close" to the observations, this condition should not have much impact on the computation of 'br2'.
This argument was introduced in hydroGOF 0.4-0, following a comment by E. White. Its default value is `FALSE` to ensure compatibility with previous versions of hydroGOF.

`fun` function to be applied to `sim` and `obs` in order to obtain transformed values thereof before computing this goodness-of-fit index.
The first argument MUST BE a numeric vector with any name (e.g., `x`), and additional arguments are passed using `...`.

`...` arguments passed to `fun`, in addition to the mandatory first numeric vector.

`epsilon.type` argument used to define a numeric value to be added to both `sim` and `obs` before applying `fun`.
It is was designed to allow the use of logarithm and other similar functions that do not work with zero values.
Valid values of `epsilon.type` are:
1) `"none"`: `sim` and `obs` are used by `fun` without the addition of any numeric value. This is the default option.
2) `"Pushpalatha2012"`: one hundredth (1/100) of the mean observed values is added to both `sim` and `obs` before applying `fun`, as described in Pushpalatha et al. (2012).

3) "otherFactor": the numeric value defined in the epsilon.value argument is used to multiply the the mean observed values, instead of the one hundredth (1/100) described in Pushpalatha et al. (2012). The resulting value is then added to both sim and obs, before applying fun.

4) "otherValue": the numeric value defined in the epsilon.value argument is directly added to both sim and obs, before applying fun.

epsilon.value -) when epsilon.type="otherValue" it represents the numeric value to be added to both sim and obs before applying fun.
-) when epsilon.type="otherFactor" it represents the numeric factor used to multiply the mean of the observed values, instead of the one hundredth (1/100) described in Pushpalatha et al. (2012). The resulting value is then added to both sim and obs before applying fun.

Details

$$br2 = |b|R2, b <= 1; br2 = \frac{R2}{|b|}, b > 1$$

A model that systematically over or under-predicts all the time will still result in "good" R2 (close to 1), even if all predictions were wrong (Krause et al., 2005). The br2 coefficient allows accounting for the discrepancy in the magnitude of two signals (depicted by 'b') as well as their dynamics (depicted by R2)

Value

br2 between sim and obs.

If sim and obs are matrixes, the returned value is a vector, with the br2 between each column of sim and obs.

Note

obs and sim has to have the same length/dimension

The missing values in obs and sim are removed before the computation proceeds, and only those positions with non-missing values in obs and sim are considered in the computation

The slope b is computed as the coefficient of the linear regression between sim and obs, forcing the intercept be equal to zero.

Author(s)

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

References

Krause, P.; Boyle, D.P.; Base, F. (2005). Comparison of different efficiency criteria for hydrological model assessment, Advances in Geosciences, 5, 89-97. doi:10.5194/adgeo-5-89-2005.

Krstic, G.; Krstic, N.S.; Zambrano-Bigiarini, M. (2016). The br2-weighting Method for Estimating the Effects of Air Pollution on Population Health. Journal of Modern Applied Statistical Methods, 15(2), 42. doi:10.22237/jmasm/1478004000

See Also

R2, rPearson, rSpearman, cor, lm, gof, ggof

Examples

```
##################
# Example 1:
# Looking at the difference between r2 and br2 for a case with systematic
# over-prediction of observed values
obs <- 1:10
sim1 <- 2*obs + 5
sim2 <- 2*obs + 25

# The coefficient of determination is equal to 1 even if there is no one single
# simulated value equal to its corresponding observed counterpart
r2 <- (cor(sim1, obs, method="pearson"))^2 # r2=1

# 'br2' effectively penalises the systematic over-estimation
br2(sim1, obs) # br2 = 0.3684211
br2(sim2, obs) # br2 = 0.1794872

ggof(sim1, obs)
ggof(sim2, obs)

# Computing 'br2' without forcing the intercept be equal to zero
br2.2 <- r2/2 # br2 = 0.5

##################
# Example 2:
# Loading daily streamflows of the Ega River (Spain), from 1961 to 1970
data(EgaEnEstellaQts)
obs <- EgaEnEstellaQts

# Generating a simulated daily time series, initially equal to the observed series
sim <- obs

# Computing the 'br2' for the "best" (unattainable) case
br2(sim=sim, obs=obs)

##################
# Example 3: br2 for simulated values equal to observations plus random noise
#            on the first half of the observed values.
```

```
#            This random noise has more relative importance for ow flows than
#            for medium and high flows.

# Randomly changing the first 1826 elements of 'sim', by using a normal distribution
# with mean 10 and standard deviation equal to 1 (default of 'rnorm').
sim[1:1826] <- obs[1:1826] + rnorm(1826, mean=10)
ggof(sim, obs)

br2(sim=sim, obs=obs)

##################
# Example 4: br2 for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' during computations.

br2(sim=sim, obs=obs, fun=log)

# Verifying the previous value:
lsim <- log(sim)
lobs <- log(obs)
br2(sim=lsim, obs=lobs)

##################
# Example 5: br2 for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and adding the Pushpalatha2012 constant
#            during computations

br2(sim=sim, obs=obs, fun=log, epsilon.type="Pushpalatha2012")

# Verifying the previous value, with the epsilon value following Pushpalatha2012
eps  <- mean(obs, na.rm=TRUE)/100
lsim <- log(sim+eps)
lobs <- log(obs+eps)
br2(sim=lsim, obs=lobs)

##################
# Example 6: br2 for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and adding a user-defined constant
#            during computations

eps <- 0.01
br2(sim=sim, obs=obs, fun=log, epsilon.type="otherValue", epsilon.value=eps)

# Verifying the previous value:
lsim <- log(sim+eps)
lobs <- log(obs+eps)
br2(sim=lsim, obs=lobs)

##################
# Example 7: br2 for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
```

```
#            logarithm to 'sim' and 'obs' and using a user-defined factor
#            to multiply the mean of the observed values to obtain the constant
#            to be added to 'sim' and 'obs' during computations

fact <- 1/50
br2(sim=sim, obs=obs, fun=log, epsilon.type="otherFactor", epsilon.value=fact)

# Verifying the previous value:
eps  <- fact*mean(obs, na.rm=TRUE)
lsim <- log(sim+eps)
lobs <- log(obs+eps)
br2(sim=lsim, obs=lobs)

##################
# Example 8: br2 for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying a
#            user-defined function to 'sim' and 'obs' during computations

fun1 <- function(x) {sqrt(x+1)}

br2(sim=sim, obs=obs, fun=fun1)

# Verifying the previous value, with the epsilon value following Pushpalatha2012
sim1 <- sqrt(sim+1)
obs1 <- sqrt(obs+1)
br2(sim=sim1, obs=obs1)
```

---

cp *Coefficient of persistence*

Description

Coefficient of persistence between sim and obs, with treatment of missing values.

Usage

```
cp(sim, obs, ...)

## Default S3 method:
cp(sim, obs, na.rm=TRUE, fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
            epsilon.value=NA)

## S3 method for class 'data.frame'
cp(sim, obs, na.rm=TRUE, fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
            epsilon.value=NA)
```

```
## S3 method for class 'matrix'
cp(sim, obs, na.rm=TRUE, fun=NULL, ...,
        epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
           epsilon.value=NA)

## S3 method for class 'zoo'
cp(sim, obs, na.rm=TRUE, fun=NULL, ...,
        epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
           epsilon.value=NA)
```

Arguments

`sim` numeric, zoo, matrix or data.frame with simulated values

`obs` numeric, zoo, matrix or data.frame with observed values

`na.rm` a logical value indicating whether 'NA' should be stripped before the computation proceeds.
When an 'NA' value is found at the i-th position in `obs` **OR** `sim`, the i-th value of `obs` **AND** `sim` are removed before the computation.

`fun` function to be applied to `sim` and `obs` in order to obtain transformed values thereof before computing this goodness-of-fit index.

The first argument MUST BE a numeric vector with any name (e.g., `x`), and additional arguments are passed using `...`.

`...` arguments passed to `fun`, in addition to the mandatory first numeric vector.

`epsilon.type` argument used to define a numeric value to be added to both `sim` and `obs` before applying `fun`.

It is was designed to allow the use of logarithm and other similar functions that do not work with zero values.

Valid values of `epsilon.type` are:

1) `"none"`: `sim` and `obs` are used by `fun` without the addition of any numeric value. This is the default option.

2) `"Pushpalatha2012"`: one hundredth (1/100) of the mean observed values is added to both `sim` and `obs` before applying `fun`, as described in Pushpalatha et al. (2012).

3) `"otherFactor"`: the numeric value defined in the `epsilon.value` argument is used to multiply the the mean observed values, instead of the one hundredth (1/100) described in Pushpalatha et al. (2012). The resulting value is then added to both `sim` and `obs`, before applying `fun`.

4) `"otherValue"`: the numeric value defined in the `epsilon.value` argument is directly added to both `sim` and `obs`, before applying `fun`.

`epsilon.value` -) when `epsilon.type="otherValue"` it represents the numeric value to be added to both `sim` and `obs` before applying `fun`.
-) when `epsilon.type="otherFactor"` it represents the numeric factor used to multiply the mean of the observed values, instead of the one hundredth (1/100) described in Pushpalatha et al. (2012). The resulting value is then added to both `sim` and `obs` before applying `fun`.

Details

$$cp = 1 - \frac{\sum_{i=2}^{N} (S_i - O_i)^2}{\sum_{i=1}^{N-1} (O_{i+1} - O_i)^2}$$

Coefficient of persistence (Kitadinis and Bras, 1980; Corradini et al., 1986) is used to compare the model performance against a simple model using the observed value of the previous day as the prediction for the current day.

The coefficient of persistence compare the predictions of the model with the predictions obtained by assuming that the process is a Wiener process (variance increasing linearly with time), in which case, the best estimate for the future is given by the latest measurement (Kitadinis and Bras, 1980).

Persistence model efficiency is a normalized model evaluation statistic that quantifies the relative magnitude of the residual variance (noise) to the variance of the errors obtained by the use of a simple persistence model (Moriasi et al., 2007).

CP ranges from 0 to 1, with CP = 1 being the optimal value and it should be larger than 0.0 to indicate a minimally acceptable model performance.

Value

Coefficient of persistence between `sim` and `obs`.

If `sim` and `obs` are matrixes, the returned value is a vector, with the coefficient of persistence between each column of `sim` and `obs`.

Note

`obs` and `sim` has to have the same length/dimension

The missing values in `obs` and `sim` are removed before the computation proceeds, and only those positions with non-missing values in `obs` and `sim` are considered in the computation.

Author(s)

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

References

Kitanidis, P.K.; Bras, R.L. (1980). Real-time forecasting with a conceptual hydrologic model. 2. Applications and results. Water Resources Research, Vol. 16, No. 6, pp. 1034:1044. doi:10.1029/WR016i006p01034.

Moriasi, D.N.; Arnold, J.G.; van Liew, M.W.; Bingner, R.L.; Harmel, R.D.; Veith, T.L. (2007). Model evaluation guidelines for systematic quantification of accuracy in watershed simulations. Transactions of the ASABE. 50(3):885-900.

See Also

gof

Examples

```
##################
# Example 1: basic ideal case
obs <- 1:10
sim <- 1:10
cp(sim, obs)

obs <- 1:10
sim <- 2:11
cp(sim, obs)

##################
# Example 2:
# Loading daily streamflows of the Ega River (Spain), from 1961 to 1970
data(EgaEnEstellaQts)
obs <- EgaEnEstellaQts

# Generating a simulated daily time series, initially equal to the observed series
sim <- obs

# Computing the 'cp' for the "best" (unattainable) case
cp(sim=sim, obs=obs)

##################
# Example 3: cp for simulated values equal to observations plus random noise
#            on the first half of the observed values.
#            This random noise has more relative importance for ow flows than
#            for medium and high flows.

# Randomly changing the first 1826 elements of 'sim', by using a normal distribution
# with mean 10 and standard deviation equal to 1 (default of 'rnorm').
sim[1:1826] <- obs[1:1826] + rnorm(1826, mean=10)
ggof(sim, obs)

cp(sim=sim, obs=obs)

##################
# Example 4: cp for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' during computations.

cp(sim=sim, obs=obs, fun=log)

# Verifying the previous value:
lsim <- log(sim)
lobs <- log(obs)
cp(sim=lsim, obs=lobs)
```

```
##################
# Example 5: cp for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and adding the Pushpalatha2012 constant
#            during computations

cp(sim=sim, obs=obs, fun=log, epsilon.type="Pushpalatha2012")

# Verifying the previous value, with the epsilon value following Pushpalatha2012
eps  <- mean(obs, na.rm=TRUE)/100
lsim <- log(sim+eps)
lobs <- log(obs+eps)
cp(sim=lsim, obs=lobs)

##################
# Example 6: cp for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and adding a user-defined constant
#            during computations

eps <- 0.01
cp(sim=sim, obs=obs, fun=log, epsilon.type="otherValue", epsilon.value=eps)

# Verifying the previous value:
lsim <- log(sim+eps)
lobs <- log(obs+eps)
cp(sim=lsim, obs=lobs)

##################
# Example 7: cp for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and using a user-defined factor
#            to multiply the mean of the observed values to obtain the constant
#            to be added to 'sim' and 'obs' during computations

fact <- 1/50
cp(sim=sim, obs=obs, fun=log, epsilon.type="otherFactor", epsilon.value=fact)

# Verifying the previous value:
eps  <- fact*mean(obs, na.rm=TRUE)
lsim <- log(sim+eps)
lobs <- log(obs+eps)
cp(sim=lsim, obs=lobs)

##################
# Example 8: cp for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying a
#            user-defined function to 'sim' and 'obs' during computations

fun1 <- function(x) {sqrt(x+1)}

cp(sim=sim, obs=obs, fun=fun1)
```

```
# Verifying the previous value, with the epsilon value following Pushpalatha2012
sim1 <- sqrt(sim+1)
obs1 <- sqrt(obs+1)
cp(sim=sim1, obs=obs1)
```

d — *Index of Agreement*

Description

Index of Agreement between `sim` and obs, with treatment of missing values.

Usage

```
d(sim, obs, ...)

## Default S3 method:
d(sim, obs, na.rm=TRUE, fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
           epsilon.value=NA)

## S3 method for class 'data.frame'
d(sim, obs, na.rm=TRUE, fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
           epsilon.value=NA)

## S3 method for class 'matrix'
d(sim, obs, na.rm=TRUE, fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
           epsilon.value=NA)

## S3 method for class 'zoo'
d(sim, obs, na.rm=TRUE, fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
           epsilon.value=NA)
```

Arguments

`sim` numeric, zoo, matrix or data.frame with simulated values

`obs` numeric, zoo, matrix or data.frame with observed values

`na.rm` a logical value indicating whether 'NA' should be stripped before the computation proceeds.
When an 'NA' value is found at the i-th position in `obs` **OR** `sim`, the i-th value of `obs` **AND** `sim` are removed before the computation.

`fun` function to be applied to `sim` and `obs` in order to obtain transformed values thereof before computing the Nash-Sutcliffe efficiency.
The first argument MUST BE a numeric vector with any name (e.g., `x`), and additional arguments are passed using `...`.

... arguments passed to `fun`, in addition to the mandatory first numeric vector.

`epsilon.type` argument used to define a numeric value to be added to both `sim` and `obs` before applying `FUN`.

It is was designed to allow the use of logarithm and other similar functions that do not work with zero values.

Valid values of `epsilon.type` are:

1) `"none"`: `sim` and `obs` are used by `FUN` without the addition of any nummeric value.

2) `"Pushpalatha2012"`: one hundredth (1/100) of the mean observed values is added to both `sim` and `obs` before applying `FUN`, as described in Pushpalatha et al. (2012).

3) `"otherFactor"`: the numeric value defined in the `epsilon.value` argument is used to multiply the the mean observed values, instead of the one hundredth (1/100) described in Pushpalatha et al. (2012). The resulting value is then added to both `sim` and `obs`, before applying `FUN`.

4) `"otherValue"`: the numeric value defined in the `epsilon.value` argument is directly added to both `sim` and `obs`, before applying `FUN`.

`epsilon.value` -) when `epsilon.type="otherValue"` it represents the numeric value to be added to both `sim` and `obs` before applying `fun`.
-) when `epsilon.type="otherFactor"` it represents the numeric factor used to multiply the mean of the observed values, instead of the one hundredth (1/100) described in Pushpalatha et al. (2012). The resulting value is then added to both `sim` and `obs` before applying `fun`.

Details

$$d = 1 - \frac{\sum_{i=1}^{N} (O_i - S_i)^2}{\sum_{i=1}^{N} (|S_i - \bar{O}| + |O_i - \bar{O}|)^2}$$

The Index of Agreement (d) developed by Willmott (1981) as a standardized measure of the degree of model prediction error.

It is is dimensionless and varies between 0 and 1. A value of 1 indicates a perfect match, and 0 indicates no agreement at all (Willmott, 1981).

The index of agreement can detect additive and proportional differences in the observed and simulated means and variances; however, it is overly sensitive to extreme values due to the squared differences (Legates and McCabe, 1999).

Value

Index of agreement between `sim` and `obs`.

If `sim` and `obs` are matrixes or data.frames, the returned value is a vector, with the index of agreement between each column of `sim` and `obs`.

Note

obs and sim has to have the same length/dimension

The missing values in obs and sim are removed before the computation proceeds, and only those positions with non-missing values in obs and sim are considered in the computation

Author(s)

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

References

Willmott, C.J. (1981). On the validation of models. Physical Geography, 2, 184–194. doi:10.1080/02723646.1981.10642213.

Willmott, C.J. (1984). On the evaluation of model performance in physical geography. Spatial Statistics and Models, G. L. Gaile and C. J. Willmott, eds., 443-460. doi:10.1007/978-94-017-3048-8_23.

Willmott, C.J.; Ackleson, S.G. Davis, R.E.; Feddema, J.J.; Klink, K.M.; Legates, D.R.; O'Donnell, J.; Rowe, C.M. (1985), Statistics for the Evaluation and Comparison of Models, J. Geophys. Res., 90(C5), 8995-9005. doi:10.1029/JC090iC05p08995.

Legates, D.R.; McCabe, G. J. Jr. (1999), Evaluating the Use of "Goodness-of-Fit" Measures in Hydrologic and Hydroclimatic Model Validation, Water Resour. Res., 35(1), 233-241. doi:10.1029/1998WR900018.

See Also

md, rd, dr, gof, ggof

Examples

```
##################
# Example 1: basic ideal case
obs <- 1:10
sim <- 1:10
d(sim, obs)

obs <- 1:10
sim <- 2:11
d(sim, obs)

##################
# Example 2:
# Loading daily streamflows of the Ega River (Spain), from 1961 to 1970
data(EgaEnEstellaQts)
obs <- EgaEnEstellaQts

# Generating a simulated daily time series, initially equal to the observed series
sim <- obs

# Computing the 'd' for the "best" (unattainable) case
d(sim=sim, obs=obs)
```

```
##################
# Example 3: d for simulated values equal to observations plus random noise
#            on the first half of the observed values.
#            This random noise has more relative importance for ow flows than
#            for medium and high flows.

# Randomly changing the first 1826 elements of 'sim', by using a normal distribution
# with mean 10 and standard deviation equal to 1 (default of 'rnorm').
sim[1:1826] <- obs[1:1826] + rnorm(1826, mean=10)
ggof(sim, obs)

d(sim=sim, obs=obs)

##################
# Example 4: d for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' during computations.

d(sim=sim, obs=obs, fun=log)

# Verifying the previous value:
lsim <- log(sim)
lobs <- log(obs)
d(sim=lsim, obs=lobs)

##################
# Example 5: d for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and adding the Pushpalatha2012 constant
#            during computations

d(sim=sim, obs=obs, fun=log, epsilon.type="Pushpalatha2012")

# Verifying the previous value, with the epsilon value following Pushpalatha2012
eps  <- mean(obs, na.rm=TRUE)/100
lsim <- log(sim+eps)
lobs <- log(obs+eps)
d(sim=lsim, obs=lobs)

##################
# Example 6: d for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and adding a user-defined constant
#            during computations

eps <- 0.01
d(sim=sim, obs=obs, fun=log, epsilon.type="otherValue", epsilon.value=eps)

# Verifying the previous value:
lsim <- log(sim+eps)
lobs <- log(obs+eps)
d(sim=lsim, obs=lobs)
```

```
##################
# Example 7: d for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and using a user-defined factor
#            to multiply the mean of the observed values to obtain the constant
#            to be added to 'sim' and 'obs' during computations

fact <- 1/50
d(sim=sim, obs=obs, fun=log, epsilon.type="otherFactor", epsilon.value=fact)

# Verifying the previous value:
eps  <- fact*mean(obs, na.rm=TRUE)
lsim <- log(sim+eps)
lobs <- log(obs+eps)
d(sim=lsim, obs=lobs)

##################
# Example 8: d for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying a
#            user-defined function to 'sim' and 'obs' during computations

fun1 <- function(x) {sqrt(x+1)}

d(sim=sim, obs=obs, fun=fun1)

# Verifying the previous value, with the epsilon value following Pushpalatha2012
sim1 <- sqrt(sim+1)
obs1 <- sqrt(obs+1)
d(sim=sim1, obs=obs1)
```

---

dr *Refined Index of Agreement*

Description

Refined Index of Agreement (dr) between `sim` and `obs`, with treatment of missing values.

Usage

```
dr(sim, obs, ...)

## Default S3 method:
dr(sim, obs, na.rm=TRUE, fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
            epsilon.value=NA)

## S3 method for class 'data.frame'
dr(sim, obs, na.rm=TRUE, fun=NULL, ...,
```

```
        epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
           epsilon.value=NA)

## S3 method for class 'matrix'
dr(sim, obs, na.rm=TRUE, fun=NULL, ...,
        epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
           epsilon.value=NA)

## S3 method for class 'zoo'
dr(sim, obs, na.rm=TRUE, fun=NULL, ...,
        epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
           epsilon.value=NA)
```

Arguments

`sim` numeric, zoo, matrix or data.frame with simulated values

`obs` numeric, zoo, matrix or data.frame with observed values

`na.rm` a logical value indicating whether 'NA' should be stripped before the computation proceeds.
When an 'NA' value is found at the i-th position in `obs` **OR** `sim`, the i-th value of `obs` **AND** `sim` are removed before the computation.

`fun` function to be applied to `sim` and `obs` in order to obtain transformed values thereof before computing the Nash-Sutcliffe efficiency.

The first argument MUST BE a numeric vector with any name (e.g., `x`), and additional arguments are passed using `...`.

`...` arguments passed to `fun`, in addition to the mandatory first numeric vector.

`epsilon.type` argument used to define a numeric value to be added to both `sim` and `obs` before applying `FUN`.

It is was designed to allow the use of logarithm and other similar functions that do not work with zero values.

Valid values of `epsilon.type` are:

1) `"none"`: `sim` and `obs` are used by `FUN` without the addition of any nummeric value.

2) `"Pushpalatha2012"`: one hundredth (1/100) of the mean observed values is added to both `sim` and `obs` before applying `FUN`, as described in Pushpalatha et al. (2012).

3) `"otherFactor"`: the numeric value defined in the `epsilon.value` argument is used to multiply the the mean observed values, instead of the one hundredth (1/100) described in Pushpalatha et al. (2012). The resulting value is then added to both `sim` and `obs`, before applying `FUN`.

4) `"otherValue"`: the numeric value defined in the `epsilon.value` argument is directly added to both `sim` and `obs`, before applying `FUN`.

`epsilon.value` -) when `epsilon.type="otherValue"` it represents the numeric value to be added to both `sim` and `obs` before applying `fun`.
-) when `epsilon.type="otherFactor"` it represents the numeric factor used to multiply the mean of the observed values, instead of the one hundredth (1/100)

described in Pushpalatha et al. (2012). The resulting value is then added to both `sim` and `obs` before applying `fun`.

Details

$$c = 2$$

$$A = \sum_{i=1}^{N} |S_i - O_i|$$

$$B = c \sum_{i=1}^{N} \left|O_i - \bar{O}\right|$$

$$dr = 1 - \frac{A}{B}; A \leq B$$

$$dr = 1 - \frac{B}{A}; A > B$$

The Refined Index of Agreement (dr, Willmott et al., 2012) is a reformulation of the orginal Willmott's index of agreement developed in the 1980s (Willmott, 1981; Willmott, 1984; Willmott et al., 1985)

The Refined Index of Agreement (dr) is dimensionless, and it varies between -1 to 1 (in contrast to the original d, which varies in [0, 1]).

The Refined Index of Agreement (dr) is monotonically related with the modified Nash-Sutcliffe (E1) desribed in Legates and McCabe (1999).

In general, dr is more rationally related to model accuracy than are other existing indices (Willmott et al., 2012; Willmott et al., 2015). It also is quite flexible, making it applicable to a wide range of model-performance problems (Willmott et al., 2012)

Value

Refined Index of Agreement (dr) between `sim` and `obs`.

If `sim` and `obs` are matrixes or data.frames, the returned value is a vector, with the Refined Index of Agreement (dr) between each column of `sim` and `obs`.

Note

`obs` and `sim` has to have the same length/dimension

The missing values in `obs` and `sim` are removed before the computation proceeds, and only those positions with non-missing values in `obs` and `sim` are considered in the computation

Author(s)

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

References

Willmott, C.J.; Robeson, S.M.; Matsuura, K. (2012). A refined index of model performance. International Journal of climatology, 32(13), pp.2088-2094. doi:10.1002/joc.2419.

Willmott, C.J.; Robeson, S.M.; Matsuura, K.; Ficklin, D.L. (2015). Assessment of three dimensionless measures of model performance. Environmental Modelling & Software, 73, pp.167-174. doi:10.1016/j.envsoft.2015.08.012

Willmott, C.J. (1981). On the validation of models. Physical Geography, 2, 184–194. doi:10.1080/02723646.1981.10642213.

Willmott, C.J. (1984). On the evaluation of model performance in physical geography. Spatial Statistics and Models, G. L. Gaile and C. J. Willmott, eds., 443-460. doi:10.1007/978-94-017-3048-8_23.

Willmott, C.J.; Ackleson, S.G. Davis, R.E.; Feddema, J.J.; Klink, K.M.; Legates, D.R.; O'Donnell, J.; Rowe, C.M. (1985), Statistics for the Evaluation and Comparison of Models, J. Geophys. Res., 90(C5), 8995-9005. doi:10.1029/JC090iC05p08995.

See Also

d, md, rd, gof, ggof

Examples

```
##################
# Example 1: basic ideal case
obs <- 1:10
sim <- 1:10
dr(sim, obs)

obs <- 1:10
sim <- 2:11
dr(sim, obs)

##################
# Example 2:
# Loading daily streamflows of the Ega River (Spain), from 1961 to 1970
data(EgaEnEstellaQts)
obs <- EgaEnEstellaQts

# Generating a simulated daily time series, initially equal to the observed series
sim <- obs

# Computing the 'dr' for the "best" (unattainable) case
dr(sim=sim, obs=obs)

##################
# Example 3: dr for simulated values equal to observations plus random noise
#            on the first half of the observed values.
#            This random noise has more relative importance for ow flows than
#            for medium and high flows.

# Randomly changing the first 1826 elements of 'sim', by using a normal distribution
# with mean 10 and standard deviation equal to 1 (default of 'rnorm').
```

```
sim[1:1826] <- obs[1:1826] + rnorm(1826, mean=10)
ggof(sim, obs)

dr(sim=sim, obs=obs)

##################
# Example 4: dr for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' during computations.

dr(sim=sim, obs=obs, fun=log)

# Verifying the previous value:
lsim <- log(sim)
lobs <- log(obs)
dr(sim=lsim, obs=lobs)

##################
# Example 5: dr for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and adding the Pushpalatha2012 constant
#            during computations

dr(sim=sim, obs=obs, fun=log, epsilon.type="Pushpalatha2012")

# Verifying the previous value, with the epsilon value following Pushpalatha2012
eps  <- mean(obs, na.rm=TRUE)/100
lsim <- log(sim+eps)
lobs <- log(obs+eps)
dr(sim=lsim, obs=lobs)

##################
# Example 6: dr for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and adding a user-defined constant
#            during computations

eps <- 0.01
dr(sim=sim, obs=obs, fun=log, epsilon.type="otherValue", epsilon.value=eps)

# Verifying the previous value:
lsim <- log(sim+eps)
lobs <- log(obs+eps)
dr(sim=lsim, obs=lobs)

##################
# Example 7: dr for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and using a user-defined factor
#            to multiply the mean of the observed values to obtain the constant
#            to be added to 'sim' and 'obs' during computations

fact <- 1/50
```

```
dr(sim=sim, obs=obs, fun=log, epsilon.type="otherFactor", epsilon.value=fact)

# Verifying the previous value:
eps  <- fact*mean(obs, na.rm=TRUE)
lsim <- log(sim+eps)
lobs <- log(obs+eps)
dr(sim=lsim, obs=lobs)

##################
# Example 8: dr for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying a
#            user-defined function to 'sim' and 'obs' during computations

fun1 <- function(x) {sqrt(x+1)}

dr(sim=sim, obs=obs, fun=fun1)

# Verifying the previous value, with the epsilon value following Pushpalatha2012
sim1 <- sqrt(sim+1)
obs1 <- sqrt(obs+1)
dr(sim=sim1, obs=obs1)
```

EgaEnEstellaQts *Ega in "Estella" (Q071), ts with daily streamflows.*

Description

Time series with daily streamflows of the Ega River (subcatchment of the Ebro River basin, Spain) measured at the gauging station "Estella" (Q071), for the period 01/Jan/1961 to 31/Dec/1970

Usage

```
data(EgaEnEstellaQts)
```

Format

zoo object.

Source

Downloaded from: https://www.chebro.es. Last accessed [March 2010].
These data are intended to be used for research purposes only, being distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY.

ggof *Graphical Goodness of Fit*

Description

Graphical comparison between two vectors (numeric, ts or zoo), with several numerical goodness of fit printed as a legend.
Missing values in observed and/or simulated values can removed before the computations.

Usage

```
ggof(sim, obs, na.rm = TRUE, dates, date.fmt = "%Y-%m-%d",
     pt.style = "ts", ftype = "o",  FUN,
     stype="default", season.names=c("Winter", "Spring", "Summer", "Autumn"),
     gof.leg = TRUE,  digits=2,
     gofs=c("ME", "MAE", "RMSE", "NRMSE", "PBIAS", "RSR", "rSD", "NSE", "mNSE",
             "rNSE", "d", "md", "rd", "r", "R2", "bR2", "KGE", "VE"),
     legend, leg.cex=1,
     tick.tstep = "auto", lab.tstep = "auto", lab.fmt=NULL,
     cal.ini=NA, val.ini=NA,
     main, xlab = "Time", ylab=c("Q, [m3/s]"),
     col = c("blue", "black"),
     cex = c(0.5, 0.5), cex.axis=1.2, cex.lab=1.2,
     lwd = c(1, 1), lty = c(1, 3), pch = c(1, 9), ...)
```

Arguments

`sim`	numeric or zoo object with with simulated values
`obs`	numeric or zoo object with observed values
`na.rm`	a logical value indicating whether 'NA' should be stripped before the computation proceeds. When an 'NA' value is found at the i-th position in obs **OR** `sim`, the i-th value of obs **AND** `sim` are removed before the computation.
`dates`	character, factor, Date or POSIXct object indicating how to obtain the dates for the corresponding values in the `sim` and obs time series If `dates` is a character or factor, it is converted into Date/POSIXct class, using the date format specified by `date.fmt`
`date.fmt`	OPTIONAL. character indicating the format in which the dates are stored in `dates`, `cal.ini` and `val.ini`. See `format` in `as.Date`. Default value is `%Y-%m-%d` ONLY required when `class(dates)=="character"` or `class(dates)=="factor"` or when `cal.ini` and/or `val.ini` is provided.
`pt.style`	Character indicating if the 2 ts have to be plotted as lines or bars. When `ftype` is NOT o, it only applies to the annual values. Valid values are: -) `ts` : (default) each ts is plotted as a lines along the 'x' axis -) `bar`: both series are plotted as barplots.

ftype
: Character indicating how many plots are desired by the user. Valid values are:
-) `o` : only the original `sim` and `obs` time series are plotted
-) `dm` : it assumes that `sim` and `obs` are daily time series and Daily and Monthly values are plotted
-) `ma` : it assumes that `sim` and `obs` are daily or monthly time series and Monthly and Annual values are plotted
-) `dma` : it assumes that `sim` and `obs` are daily time series and Daily, Monthly and Annual values are plotted
-) `seasonal`: seasonal values are plotted. See `stype` and `season.names`

FUN
: OPTIONAL, ONLY required when `ftype` is in `c('dm', 'ma', 'dma', 'seasonal')`. Function that have to be applied for transforming teh original ts into monthly, annual or seasonal time step (e.g., for precipitation FUN MUST be `sum`, for temperature and flow time series, FUN MUST be `mean`)

stype
: OPTIONAL, only used when `ftype=seasonal`.
character, indicating whath weather seasons will be used for computing the output. Possible values are:
-) `default` => "winter"= DJF = Dec, Jan, Feb; "spring"= MAM = Mar, Apr, May; "summer"= JJA = Jun, Jul, Aug; "autumn"= SON = Sep, Oct, Nov
-) `FrenchPolynesia` => "winter"= DJFM = Dec, Jan, Feb, Mar; "spring"= AM = Apr, May; "summer"= JJAS = Jun, Jul, Aug, Sep; "autumn"= ON = Oct, Nov

season.names
: OPTIONAL, only used when `ftype=seasonal`.
character of length 4 indicating the names of each one of the weather seasons defined by `stype`.These names are only used for plotting purposes

gof.leg
: logical, indicating if several numerical goodness of fit have to be computed between `sim` and `obs`, and plotted as a legend on the graph. If `leg.gof=TRUE`, then `x` is considered as observed and `y` as simulated values (for some gof functions this is important).

digits
: OPTIONAL, only used when `leg.gof=TRUE`. Numeric, representing the decimal places used for rounding the goodness-of-fit indexes.

gofs
: character, with one or more strings indicating the goodness-of-fit measures to be shown in the legend of the plot when `gof.leg=TRUE`.
Possible values when `ftype!='seasonal'` are in `c("ME", "MAE", "MSE", "RMSE", "NRMSE", "PBIAS", "RSR", "rSD", "NSE", "mNSE", "rNSE", "d", "md", "rd", "cp", "r", "R2", "bR2", "KGE", "VE")`
Possible values when `ftype='seasonal'` are in c("ME", "RMSE", "PBIAS", "RSR", "NSE", "d", "R2", "KGE", "VE")

legend
: character of length 2 to appear in the legend.

leg.cex
: OPTIONAL. ONLY used when `leg.gof=TRUE`. Character expansion factor for drawing the legend, *relative* to current 'par("cex")'. Used for text, and provides the default for 'pt.cex' and 'title.cex'. Default value = 1

tick.tstep
: character, indicating the time step that have to be used for putting the ticks on the time axis. Valid values are: `auto`, `years`, `months`,`weeks`, `days`, `hours`, `minutes`, `seconds`.

lab.tstep
: character, indicating the time step that have to be used for putting the labels on the time axis. Valid values are: `auto`, `years`, `months`,`weeks`, `days`, `hours`, `minutes`, `seconds`.

`lab.fmt`	Character indicating the format to be used for the label of the axis. See `lab.fmt` in `drawTimeAxis`.
`cal.ini`	OPTIONAL. Character, indicating the date in which the calibration period started. When `cal.ini` is provided, all the values in `obs` and `sim` with dates previous to `cal.ini` are SKIPPED from the computation of the goodness-of-fit measures (when `gof.leg=TRUE`), but their values are still plotted, in order to examine if the warming up period was too short, acceptable or too long for the chosen calibration period. In addition, a vertical red line in drawn at this date.
`val.ini`	OPTIONAL. Character, the date in which the validation period started. ONLY used for drawing a vertical red line at this date.
`main`	character representing the main title of the plot.
`xlab`	label for the 'x' axis.
`ylab`	label for the 'y' axis.
`col`	character, representing the colors of `sim` and `obs`
`cex`	numeric, representing the values controlling the size of text and symbols of 'x' and 'y' with respect to the default
`cex.axis`	numeric, representing the magnification to be used for the axis annotation relative to 'cex'. See `par`.
`cex.lab`	numeric, representing the magnification to be used for x and y labels relative to the current setting of 'cex'. See `par`.
`lwd`	vector with the line width of `sim` and `obs`
`lty`	numeric with the line type of `sim` and `obs`
`pch`	numeric with the type of symbol for `x` and `y`. (e.g., 1: white circle; 9: white rhombus with a cross inside)
`...`	further arguments passed to or from other methods.

Details

Plots observed and simulated values in the same graph.

If `gof.leg=TRUE`, it computes the numerical values of:
'me', 'mae', 'rmse', 'nrmse', 'PBIAS', 'RSR, 'rSD', 'NSE', 'mNSE', 'rNSE', 'd', 'md, 'rd', 'cp', 'r', 'r.Spearman', 'R2', 'bR2', 'KGE', 'VE'

Value

The output of the `gof` function is a matrix with one column only, and the following rows:

ME	Mean Error
MAE	Mean Absolute Error
MSE	Mean Squared Error
RMSE	Root Mean Square Error
ubRMSE	Unbiased Root Mean Square Error
NRMSE	Normalized Root Mean Square Error (-100% <= NRMSE <= 100%)

PBIAS	Percent Bias (-Inf <= PBIAS <= Inf [%])
RSR	Ratio of RMSE to the Standard Deviation of the Observations, RSR = rms / sd(obs). (0 <= RSR <= +Inf)
rSD	Ratio of Standard Deviations, rSD = sd(sim) / sd(obs)
NSE	Nash-Sutcliffe Efficiency (-Inf <= NSE <= 1)
mNSE	Modified Nash-Sutcliffe Efficiency (-Inf <= mNSE <= 1)
rNSE	Relative Nash-Sutcliffe Efficiency (-Inf <= rNSE <= 1)
wNSE	Weighted Nash-Sutcliffe Efficiency (-Inf <= wNSE <= 1)
wsNSE	Weighted Seasonal Nash-Sutcliffe Efficiency (-Inf <= wsNSE <= 1)
d	Index of Agreement (0 <= d <= 1)
dr	Refined Index of Agreement (-1 <= dr <= 1)
md	Modified Index of Agreement (0 <= md <= 1)
rd	Relative Index of Agreement (0 <= rd <= 1)
cp	Persistence Index (0 <= cp <= 1)
r	Pearson Correlation coefficient (-1 <= r <= 1)
R2	Coefficient of Determination (0 <= R2 <= 1)
bR2	R2 multiplied by the coefficient of the regression line between `sim` and `obs` (0 <= bR2 <= 1)
VE	Volumetric efficiency between `sim` and `obs` (-Inf <= VE <= 1)
KGE	Kling-Gupta efficiency between `sim` and `obs` (-Inf <= KGE <= 1)
KGElf	Kling-Gupta Efficiency for low values between `sim` and `obs` (-Inf <= KGElf <= 1)
KGEnp	Non-parametric version of the Kling-Gupta Efficiency between `sim` and `obs` (-Inf <= KGEnp <= 1)
KGEkm	Knowable Moments Kling-Gupta Efficiency between `sim` and `obs` (-Inf <= KGEnp <= 1)

The following outputs are only produced when both `sim` and `obs` are zoo objects:

sKGE	Split Kling-Gupta Efficiency between `sim` and `obs` (-Inf <= sKGE <= 1). Only computed when both `sim` and `obs` are zoo objects
APFB	Annual Peak Flow Bias (0 <= APFB <= Inf)
HBF	High Flow Bias (0 <= HFB <= Inf)
r.Spearman	Spearman Correlation coefficient (-1 <= r.Spearman <= 1). Only computed when `do.spearman=TRUE`
pbiasfdc	PBIAS in the slope of the midsegment of the Flow Duration Curve

Author(s)

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

References

Abbaspour, K.C.; Faramarzi, M.; Ghasemi, S.S.; Yang, H. (2009), Assessing the impact of climate change on water resources in Iran, Water Resources Research, 45(10), W10,434, doi:10.1029/2008WR007615.

Abbaspour, K.C., Yang, J. ; Maximov, I.; Siber, R.; Bogner, K.; Mieleitner, J. ; Zobrist, J.; Srinivasan, R. (2007), Modelling hydrology and water quality in the pre-alpine/alpine Thur watershed using SWAT, Journal of Hydrology, 333(2-4), 413-430, doi:10.1016/j.jhydrol.2006.09.014.

Box, G.E. (1966). Use and abuse of regression. Technometrics, 8(4), 625-629. doi:10.1080/00401706.1966.10490407.

Barrett, J.P. (1974). The coefficient of determination-some limitations. The American Statistician, 28(1), 19-20. doi:10.1080/00031305.1974.10479056.

Chai, T.; Draxler, R.R. (2014). Root mean square error (RMSE) or mean absolute error (MAE)? - Arguments against avoiding RMSE in the literature, Geoscientific Model Development, 7, 1247-1250. doi:10.5194/gmd-7-1247-2014.

Cinkus, G.; Mazzilli, N.; Jourde, H.; Wunsch, A.; Liesch, T.; Ravbar, N.; Chen, Z.; and Goldscheider, N. (2023). When best is the enemy of good - critical evaluation of performance criteria in hydrological models. Hydrology and Earth System Sciences 27, 2397-2411, doi:10.5194/hess-27-2397-2023.

Criss, R. E.; Winston, W. E. (2008), Do Nash values have value? Discussion and alternate proposals. Hydrological Processes, 22: 2723-2725. doi:10.1002/hyp.7072.

Entekhabi, D.; Reichle, R.H.; Koster, R.D.; Crow, W.T. (2010). Performance metrics for soil moisture retrievals and application requirements. Journal of Hydrometeorology, 11(3), 832-840. doi: 10.1175/2010JHM1223.1.

Fowler, K.; Coxon, G.; Freer, J.; Peel, M.; Wagener, T.; Western, A.; Woods, R.; Zhang, L. (2018). Simulating runoff under changing climatic conditions: A framework for model improvement. Water Resources Research, 54(12), 812-9832. doi:10.1029/2018WR023989.

Garcia, F.; Folton, N.; Oudin, L. (2017). Which objective function to calibrate rainfall-runoff models for low-flow index simulations?. Hydrological sciences journal, 62(7), 1149-1166. doi:10.1080/02626667.2017.1308511.

Garrick, M.; Cunnane, C.; Nash, J.E. (1978). A criterion of efficiency for rainfall-runoff models. Journal of Hydrology 36, 375-381. doi:10.1016/0022-1694(78)90155-5.

Gupta, H.V.; Kling, H.; Yilmaz, K.K.; Martinez, G.F. (2009). Decomposition of the mean squared error and NSE performance criteria: Implications for improving hydrological modelling. Journal of hydrology, 377(1-2), 80-91. doi:10.1016/j.jhydrol.2009.08.003. ISSN 0022-1694.

Gupta, H.V.; Kling, H. (2011). On typical range, sensitivity, and normalization of Mean Squared Error and Nash-Sutcliffe Efficiency type metrics. Water Resources Research, 47(10). doi:10.1029/2011WR010962.

Hahn, G.J. (1973). The coefficient of determination exposed. Chemtech, 3(10), 609-612. Aailable online at: `https://www2.hawaii.edu/~cbaajwe/Ph.D.Seminar/Hahn1973.pdf`*.*

Hodson, T.O. (2022). Root-mean-square error (RMSE) or mean absolute error (MAE): when to use them or not, Geoscientific Model Development, 15, 5481-5487, doi:10.5194/gmd-15-5481-2022.

Hundecha, Y., Bardossy, A. (2004). Modeling of the effect of land use changes on the runoff generation of a river basin through parameter regionalization of a watershed model. Journal of hydrology, 292(1-4), 281-295. doi:10.1016/j.jhydrol.2004.01.002.

Kitanidis, P.K.; Bras, R.L. (1980). Real-time forecasting with a conceptual hydrologic model. 2. Applications and results. Water Resources Research, Vol. 16, No. 6, pp. 1034:1044. doi:10.1029/WR016i006p01034.

Kling, H.; Fuchs, M.; Paulin, M. (2012). Runoff conditions in the upper Danube basin under an ensemble of climate change scenarios. Journal of Hydrology, 424, 264-277, doi:10.1016/j.jhydrol.2012.01.011.

Knoben, W.J.; Freer, J.E.; Woods, R.A. (2019). Inherent benchmark or not? Comparing Nash-Sutcliffe and Kling-Gupta efficiency scores. Hydrology and Earth System Sciences, 23(10), 4323-4331. doi:10.5194/hess-23-4323-2019.

Krause, P.; Boyle, D.P.; Base, F. (2005). Comparison of different efficiency criteria for hydrological model assessment, Advances in Geosciences, 5, 89-97. doi:10.5194/adgeo-5-89-2005.

Krstic, G.; Krstic, N.S.; Zambrano-Bigiarini, M. (2016). The br2-weighting Method for Estimating the Effects of Air Pollution on Population Health. Journal of Modern Applied Statistical Methods, 15(2), 42. doi:10.22237/jmasm/1478004000

Legates, D.R.; McCabe, G. J. Jr. (1999), Evaluating the Use of "Goodness-of-Fit" Measures in Hydrologic and Hydroclimatic Model Validation, Water Resour. Res., 35(1), 233-241. doi:10.1029/1998WR900018.

Ling, X.; Huang, Y.; Guo, W.; Wang, Y.; Chen, C.; Qiu, B.; Ge, J.; Qin, K.; Xue, Y.; Peng, J. (2021). Comprehensive evaluation of satellite-based and reanalysis soil moisture products using in situ observations over China. Hydrology and Earth System Sciences, 25(7), 4209-4229. doi:10.5194/hess-25-4209-2021.

Mizukami, N.; Rakovec, O.; Newman, A.J.; Clark, M.P.; Wood, A.W.; Gupta, H.V.; Kumar, R.: (2019). On the choice of calibration metrics for "high-flow" estimation using hydrologic models, Hydrology Earth System Sciences 23, 2601-2614, doi:10.5194/hess-23-2601-2019.

Moriasi, D.N.; Arnold, J.G.; van Liew, M.W.; Bingner, R.L.; Harmel, R.D.; Veith, T.L. (2007). Model evaluation guidelines for systematic quantification of accuracy in watershed simulations. Transactions of the ASABE. 50(3):885-900

Nash, J.E. and Sutcliffe, J.V. (1970). River flow forecasting through conceptual models. Part 1: a discussion of principles, Journal of Hydrology 10, pp. 282-290. doi:10.1016/0022-1694(70)90255-6.

Pearson, K. (1920). Notes on the history of correlation. Biometrika, 13(1), 25-45. doi:10.2307/2331722.

Pfannerstill, M.; Guse, B.; Fohrer, N. (2014). Smart low flow signature metrics for an improved overall performance evaluation of hydrological models. Journal of Hydrology, 510, 447-458. doi:10.1016/j.jhydrol.2013.12.0

Pizarro, A.; Jorquera, J. (2024). Advancing objective functions in hydrological modelling: Integrating knowable moments for improved simulation accuracy. Journal of Hydrology, 634, 131071. doi:10.1016/j.jhydrol.2024.131071.

Pool, S.; Vis, M.; Seibert, J. (2018). Evaluating model performance: towards a non-parametric variant of the Kling-Gupta efficiency. Hydrological Sciences Journal, 63(13-14), pp.1941-1953. doi:/10.1080/02626667.2018.1552002.

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

Santos, L.; Thirel, G.; Perrin, C. (2018). Pitfalls in using log-transformed flows within the KGE criterion. doi:10.5194/hess-22-4583-2018.

Schaefli, B., Gupta, H. (2007). Do Nash values have value?. Hydrological Processes 21, 2075-2080. doi:10.1002/hyp.6825.

Schober, P.; Boer, C.; Schwarte, L.A. (2018). Correlation coefficients: appropriate use and interpretation. Anesthesia and Analgesia, 126(5), 1763-1768. doi:10.1213/ANE.0000000000002864.

Schuol, J.; Abbaspour, K.C.; Srinivasan, R.; Yang, H. (2008b), Estimation of freshwater availability in the West African sub-continent using the SWAT hydrologic model, Journal of Hydrology, 352(1-2), 30, doi:10.1016/j.jhydrol.2007.12.025

Sorooshian, S., Q. Duan, and V. K. Gupta. (1993). Calibration of rainfall-runoff models: Application of global optimization to the Sacramento Soil Moisture Accounting Model, Water Resources Research, 29 (4), 1185-1194, doi:10.1029/92WR02617.

Spearman, C. (1961). The Proof and Measurement of Association Between Two Things. In J. J. Jenkins and D. G. Paterson (Eds.), Studies in individual differences: The search for intelligence (pp. 45-58). Appleton-Century-Crofts. doi:10.1037/11491-005

Tang, G.; Clark, M.P.; Papalexiou, S.M. (2021). SC-earth: a station-based serially complete earth dataset from 1950 to 2019. Journal of Climate, 34(16), 6493-6511. doi:10.1175/JCLI-D-21-0067.1.

Yapo P.O.; Gupta H.V.; Sorooshian S. (1996). Automatic calibration of conceptual rainfall-runoff models: sensitivity to calibration data. Journal of Hydrology. v181 i1-4. 23-48. doi:10.1016/0022-1694(95)02918-4

Yilmaz, K.K., Gupta, H.V. ; Wagener, T. (2008), A process-based diagnostic approach to model evaluation: Application to the NWS distributed hydrologic model, Water Resources Research, 44, W09417, doi:10.1029/2007WR006716.

Willmott, C.J. (1981). On the validation of models. Physical Geography, 2, 184–194. doi:10.1080/02723646.1981.10642213.

Willmott, C.J. (1984). On the evaluation of model performance in physical geography. Spatial Statistics and Models, G. L. Gaile and C. J. Willmott, eds., 443-460. doi:10.1007/978-94-017-3048-8_23.

Willmott, C.J.; Ackleson, S.G. Davis, R.E.; Feddema, J.J.; Klink, K.M.; Legates, D.R.; O'Donnell, J.; Rowe, C.M. (1985), Statistics for the Evaluation and Comparison of Models, J. Geophys. Res., 90(C5), 8995-9005. doi:10.1029/JC090iC05p08995.

Willmott, C.J.; Matsuura, K. (2005). Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance, Climate Research, 30, 79-82, doi:10.3354/cr030079.

Willmott, C.J.; Matsuura, K.; Robeson, S.M. (2009). Ambiguities inherent in sums-of-squares-based error statistics, Atmospheric Environment, 43, 749-752, doi:10.1016/j.atmosenv.2008.10.005.

Willmott, C.J.; Robeson, S.M.; Matsuura, K. (2012). A refined index of model performance. International Journal of climatology, 32(13), pp.2088-2094. doi:10.1002/joc.2419.

Willmott, C.J.; Robeson, S.M.; Matsuura, K.; Ficklin, D.L. (2015). Assessment of three dimensionless measures of model performance. Environmental Modelling & Software, 73, pp.167-174. doi:10.1016/j.envsoft.2015.08.012

Zambrano-Bigiarini, M.; Bellin, A. (2012). Comparing goodness-of-fit measures for calibration of models focused on extreme events. EGU General Assembly 2012, Vienna, Austria, 22-27 Apr 2012, EGU2012-11549-1.

See Also

`gof`, `plot2`, `ggof`, `me`, `mae`, `mse`, `rmse`, `ubRMSE`, `nrmse`, `pbias`, `rsr`, `rSD`, `NSE`, `mNSE`, `rNSE`, `wNSE`, `d`, `dr`, `md`, `rd`, `cp`, `rPearson`, `R2`, `br2`, `KGE`, `KGElf`, `KGEnp`, `sKGE`, `VE`, `rSpearman`, `pbiasfdc`

Examples

```
obs <- 1:10
sim <- 2:11

## Not run:
ggof(sim, obs)

## End(Not run)

##################
# Loading daily streamflows of the Ega River (Spain), from 1961 to 1970
data(EgaEnEstellaQts)
obs <- EgaEnEstellaQts

# Generating a simulated daily time series, initially equal to the observed series
sim <- obs

# Getting the numeric goodness of fit for the "best" (unattainable) case
gof(sim=sim, obs=obs)

# Randomly changing the first 2000 elements of 'sim', by using a normal distribution
# with mean 10 and standard deviation equal to 1 (default of 'rnorm').
sim[1:2000] <- obs[1:2000] + rnorm(2000, mean=10)

# Getting the new numeric goodness-of-fit measures
gof(sim=sim, obs=obs)

# Getting the graphical representation of 'obs' and 'sim' along with the numeric
# goodness-of-fit measures for the daily and monthly time series
## Not run:
ggof(sim=sim, obs=obs, ftype="dm", FUN=mean)

## End(Not run)

# Getting the graphical representation of 'obs' and 'sim' along with some numeric
# goodness-of-fit measures for the seasonal time series
## Not run:
ggof(sim=sim, obs=obs, ftype="seasonal", FUN=mean)

## End(Not run)

# Computing the daily residuals
# even if this is a dummy example, it is enough for illustrating the capability
r <- sim-obs

# Summarizing and plotting the residuals
## Not run:
library(hydroTSM)

# summary
smry(r)
```

```
# daily, monthly and annual plots, boxplots and histograms
hydroplot(r, FUN=mean)

# seasonal plots and boxplots
hydroplot(r, FUN=mean, pfreq="seasonal")

## End(Not run)
```

---

gof *Numerical Goodness-of-fit measures*

Description

Numerical goodness-of-fit measures between `sim` and obs, with treatment of missing values. Several performance indices for comparing two vectors, matrices or data.frames

Usage

```
gof(sim, obs, ...)

## Default S3 method:
gof(sim, obs, na.rm=TRUE, do.spearman=FALSE, do.pbfdc=FALSE,
      j=1, lambda=0.95, norm="sd", s=c(1,1,1), method=c("2009", "2012", "2021"),
        lQ.thr=0.6, hQ.thr=0.1, start.month=1, digits=2, fun=NULL, ...,
        epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
        epsilon.value=NA)

## S3 method for class 'matrix'
gof(sim, obs, na.rm=TRUE, do.spearman=FALSE, do.pbfdc=FALSE,
      j=1, lambda=0.95, norm="sd", s=c(1,1,1), method=c("2009", "2012", "2021"),
        lQ.thr=0.6, hQ.thr=0.1, start.month=1, digits=2, fun=NULL, ...,
        epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
        epsilon.value=NA)

## S3 method for class 'data.frame'
gof(sim, obs, na.rm=TRUE, do.spearman=FALSE, do.pbfdc=FALSE,
      j=1, lambda=0.95, norm="sd", s=c(1,1,1), method=c("2009", "2012", "2021"),
        lQ.thr=0.6, hQ.thr=0.1, start.month=1, digits=2, fun=NULL, ...,
        epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
        epsilon.value=NA)

## S3 method for class 'zoo'
gof(sim, obs, na.rm=TRUE, do.spearman=FALSE, do.pbfdc=FALSE,
      j=1, lambda=0.95, norm="sd", s=c(1,1,1), method=c("2009", "2012", "2021"),
        lQ.thr=0.6, hQ.thr=0.1, start.month=1, digits=2, fun=NULL, ...,
        epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
        epsilon.value=NA)
```

Arguments

sim	numeric, zoo, matrix or data.frame with simulated values
obs	numeric, zoo, matrix or data.frame with observed values
na.rm	a logical value indicating whether 'NA' should be stripped before the computation proceeds. When an 'NA' value is found at the i-th position in obs **OR** sim, the i-th value of obs **AND** sim are removed before the computation.
do.spearman	logical. Indicates if the Spearman correlation has to be computed. The default is FALSE.
do.pbfdc	logical. Indicates if the Percent Bias in the Slope of the midsegment of the Flow Duration Curve (pbiasfdc) has to be computed. The default is FALSE.
j	argument passed to the mNSE and wsNSE functions.
lambda	argument passed to the wsNSE function.
norm	argument passed to the nrmse function
s	argument passed to the KGE, KGElf, sKGE and KGEkm functions.
method	argument passed to the KGE, KGElf, sKGE and KGEkm functions.
lQ.thr	[OPTIONAL]. Only used for the computation of the pbiasFDC % (with the pbiasfdc function) and the weighted seasonal Nash-Sutcliffe Efficiency (with the wsNSE function.
hQ.thr	[OPTIONAL]. Only used for the computation of the pbiasFDC % (with the pbiasfdc function), the high flow bias (HFB, with the HFB function) and the weighted seasonal Nash-Sutcliffe Efficiency (with the wsNSE function.
start.month	[OPTIONAL]. Only used for the computation of the split KGE (sKGE), annual peak flow bias (APFB) and high flow bias (HFB) when the (hydrological) year of interest is different from the calendar year. numeric in [1:12] indicating the starting month of the (hydrological) year. Numeric values in [1, 12] represent months in [January, December]. By default start.month=1.
digits	decimal places used for rounding the goodness-of-fit indexes.
fun	function to be applied to sim and obs in order to obtain transformed values thereof before computing the all the goodness-of-fit functions. The first argument MUST BE a numeric vector with any name (e.g., x), and additional arguments are passed using ....
...	arguments passed to fun, in addition to the mandatory first numeric vector.
epsilon.type	argument used to define a numeric value to be added to both sim and obs before applying fun. It is was designed to allow the use of logarithm and other similar functions that do not work with zero values. Valid values of epsilon.type are: 1) "none": sim and obs are used by FUN without the addition of any nummeric value.

2) "Pushpalatha2012": one hundredth (1/100) of the mean observed values is added to both sim and obs before applying FUN, as described in Pushpalatha et al. (2012).

3) "otherFactor": the numeric value defined in the epsilon.value argument is used to multiply the the mean observed values, instead of the one hundredth (1/100) described in Pushpalatha et al. (2012). The resulting value is then added to both sim and obs, before applying FUN.

4) "otherValue": the numeric value defined in the epsilon.value argument is directly added to both sim and obs, before applying FUN.

epsilon.value -) when epsilon.type="otherValue" it represents the numeric value to be added to both sim and obs before applying fun.
-) when epsilon.type="otherFactor" it represents the numeric factor used to multiply the mean of the observed values, instead of the one hundredth (1/100) described in Pushpalatha et al. (2012). The resulting value is then added to both sim and obs before applying fun.

Value

The output of the gof function is a matrix with one column only, and the following rows:

ME Mean Error

MAE Mean Absolute Error

MSE Mean Squared Error

RMSE Root Mean Square Error

ubRMSE Unbiased Root Mean Square Error

NRMSE Normalized Root Mean Square Error (-100% <= NRMSE <= 100%)

PBIAS Percent Bias (-Inf <= PBIAS <= Inf [%])

RSR Ratio of RMSE to the Standard Deviation of the Observations, RSR = rms / sd(obs). (0 <= RSR <= +Inf)

rSD Ratio of Standard Deviations, rSD = sd(sim) / sd(obs)

NSE Nash-Sutcliffe Efficiency (-Inf <= NSE <= 1)

mNSE Modified Nash-Sutcliffe Efficiency (-Inf <= mNSE <= 1)

rNSE Relative Nash-Sutcliffe Efficiency (-Inf <= rNSE <= 1)

wNSE Weighted Nash-Sutcliffe Efficiency (-Inf <= wNSE <= 1)

wsNSE Weighted Seasonal Nash-Sutcliffe Efficiency (-Inf <= wsNSE <= 1)

d Index of Agreement (0 <= d <= 1)

dr Refined Index of Agreement (-1 <= dr <= 1)

md Modified Index of Agreement (0 <= md <= 1)

rd Relative Index of Agreement (0 <= rd <= 1)

cp Persistence Index (0 <= cp <= 1)

r Pearson Correlation coefficient (-1 <= r <= 1)

R2 Coefficient of Determination (0 <= R2 <= 1)

`bR2`	R2 multiplied by the coefficient of the regression line between `sim` and `obs` (0 <= bR2 <= 1)
`VE`	Volumetric efficiency between `sim` and `obs` (-Inf <= VE <= 1)
`KGE`	Kling-Gupta efficiency between `sim` and `obs` (-Inf <= KGE <= 1)
`KGElf`	Kling-Gupta Efficiency for low values between `sim` and `obs` (-Inf <= KGElf <= 1)
`KGEnp`	Non-parametric version of the Kling-Gupta Efficiency between `sim` and `obs` (-Inf <= KGEnp <= 1)
`KGEkm`	Knowable Moments Kling-Gupta Efficiency between `sim` and `obs` (-Inf <= KGEnp <= 1)

The following outputs are only produced when both `sim` and `obs` are zoo objects:

`sKGE`	Split Kling-Gupta Efficiency between `sim` and `obs` (-Inf <= sKGE <= 1). Only computed when both `sim` and `obs` are zoo objects
`APFB`	Annual Peak Flow Bias (0 <= APFB <= Inf)
`HBF`	High Flow Bias (0 <= HFB <= Inf)
`r.Spearman`	Spearman Correlation coefficient (-1 <= r.Spearman <= 1). Only computed when `do.spearman=TRUE`
`pbiasfdc`	PBIAS in the slope of the midsegment of the Flow Duration Curve

Note

`obs` and `sim` has to have the same length/dimension.

Missing values in `obs` and/or `sim` can be removed before the computations, depending on the value of `na.rm`.

Although `r` and `r2` have been widely used for model evaluation, these statistics are over-sensitive to outliers and insensitive to additive and proportional differences between model predictions and measured data (Legates and McCabe, 1999)

Author(s)

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

References

Abbaspour, K.C.; Faramarzi, M.; Ghasemi, S.S.; Yang, H. (2009), Assessing the impact of climate change on water resources in Iran, Water Resources Research, 45(10), W10,434, doi:10.1029/2008WR007615.

Abbaspour, K.C., Yang, J. ; Maximov, I.; Siber, R.; Bogner, K.; Mieleitner, J. ; Zobrist, J.; Srinivasan, R. (2007), Modelling hydrology and water quality in the pre-alpine/alpine Thur watershed using SWAT, Journal of Hydrology, 333(2-4), 413-430, doi:10.1016/j.jhydrol.2006.09.014.

Box, G.E. (1966). Use and abuse of regression. Technometrics, 8(4), 625-629. doi:10.1080/00401706.1966.10490407.

Barrett, J.P. (1974). The coefficient of determination-some limitations. The American Statistician, 28(1), 19-20. doi:10.1080/00031305.1974.10479056.

Chai, T.; Draxler, R.R. (2014). Root mean square error (RMSE) or mean absolute error (MAE)? - Arguments against avoiding RMSE in the literature, Geoscientific Model Development, 7, 1247-1250. doi:10.5194/gmd-7-1247-2014.

Cinkus, G.; Mazzilli, N.; Jourde, H.; Wunsch, A.; Liesch, T.; Ravbar, N.; Chen, Z.; and Goldscheider, N. (2023). When best is the enemy of good - critical evaluation of performance criteria in hydrological models. Hydrology and Earth System Sciences 27, 2397-2411, doi:10.5194/hess-27-2397-2023.

Criss, R. E.; Winston, W. E. (2008), Do Nash values have value? Discussion and alternate proposals. Hydrological Processes, 22: 2723-2725. doi:10.1002/hyp.7072.

Entekhabi, D.; Reichle, R.H.; Koster, R.D.; Crow, W.T. (2010). Performance metrics for soil moisture retrievals and application requirements. Journal of Hydrometeorology, 11(3), 832-840. doi: 10.1175/2010JHM1223.1.

Fowler, K.; Coxon, G.; Freer, J.; Peel, M.; Wagener, T.; Western, A.; Woods, R.; Zhang, L. (2018). Simulating runoff under changing climatic conditions: A framework for model improvement. Water Resources Research, 54(12), 812-9832. doi:10.1029/2018WR023989.

Garcia, F.; Folton, N.; Oudin, L. (2017). Which objective function to calibrate rainfall-runoff models for low-flow index simulations?. Hydrological sciences journal, 62(7), 1149-1166. doi:10.1080/02626667.2017.1308511.

Garrick, M.; Cunnane, C.; Nash, J.E. (1978). A criterion of efficiency for rainfall-runoff models. Journal of Hydrology 36, 375-381. doi:10.1016/0022-1694(78)90155-5.

Gupta, H.V.; Kling, H.; Yilmaz, K.K.; Martinez, G.F. (2009). Decomposition of the mean squared error and NSE performance criteria: Implications for improving hydrological modelling. Journal of hydrology, 377(1-2), 80-91. doi:10.1016/j.jhydrol.2009.08.003. ISSN 0022-1694.

Gupta, H.V.; Kling, H. (2011). On typical range, sensitivity, and normalization of Mean Squared Error and Nash-Sutcliffe Efficiency type metrics. Water Resources Research, 47(10). doi:10.1029/2011WR010962.

Hahn, G.J. (1973). The coefficient of determination exposed. Chemtech, 3(10), 609-612. Aailable online at: `https://www2.hawaii.edu/~cbaajwe/Ph.D.Seminar/Hahn1973.pdf`.

Hodson, T.O. (2022). Root-mean-square error (RMSE) or mean absolute error (MAE): when to use them or not, Geoscientific Model Development, 15, 5481-5487, doi:10.5194/gmd-15-5481-2022.

Hundecha, Y., Bardossy, A. (2004). Modeling of the effect of land use changes on the runoff generation of a river basin through parameter regionalization of a watershed model. Journal of hydrology, 292(1-4), 281-295. doi:10.1016/j.jhydrol.2004.01.002.

Kitanidis, P.K.; Bras, R.L. (1980). Real-time forecasting with a conceptual hydrologic model. 2. Applications and results. Water Resources Research, Vol. 16, No. 6, pp. 1034:1044. doi:10.1029/WR016i006p01034.

Kling, H.; Fuchs, M.; Paulin, M. (2012). Runoff conditions in the upper Danube basin under an ensemble of climate change scenarios. Journal of Hydrology, 424, 264-277, doi:10.1016/j.jhydrol.2012.01.011.

Knoben, W.J.; Freer, J.E.; Woods, R.A. (2019). Inherent benchmark or not? Comparing Nash-Sutcliffe and Kling-Gupta efficiency scores. Hydrology and Earth System Sciences, 23(10), 4323-4331. doi:10.5194/hess-23-4323-2019.

Krause, P.; Boyle, D.P.; Base, F. (2005). Comparison of different efficiency criteria for hydrological model assessment, Advances in Geosciences, 5, 89-97. doi:10.5194/adgeo-5-89-2005.

Krstic, G.; Krstic, N.S.; Zambrano-Bigiarini, M. (2016). The br2-weighting Method for Estimating the Effects of Air Pollution on Population Health. Journal of Modern Applied Statistical Methods, 15(2), 42. doi:10.22237/jmasm/1478004000

Legates, D.R.; McCabe, G. J. Jr. (1999), Evaluating the Use of "Goodness-of-Fit" Measures in Hydrologic and Hydroclimatic Model Validation, Water Resour. Res., 35(1), 233-241. doi:10.1029/1998WR900018.

Ling, X.; Huang, Y.; Guo, W.; Wang, Y.; Chen, C.; Qiu, B.; Ge, J.; Qin, K.; Xue, Y.; Peng, J. (2021). Comprehensive evaluation of satellite-based and reanalysis soil moisture products using in situ observations over China. Hydrology and Earth System Sciences, 25(7), 4209-4229. doi:10.5194/hess-25-4209-2021.

Mizukami, N.; Rakovec, O.; Newman, A.J.; Clark, M.P.; Wood, A.W.; Gupta, H.V.; Kumar, R.: (2019). On the choice of calibration metrics for "high-flow" estimation using hydrologic models, Hydrology Earth System Sciences 23, 2601-2614, doi:10.5194/hess-23-2601-2019.

Moriasi, D.N.; Arnold, J.G.; van Liew, M.W.; Bingner, R.L.; Harmel, R.D.; Veith, T.L. (2007). Model evaluation guidelines for systematic quantification of accuracy in watershed simulations. Transactions of the ASABE. 50(3):885-900

Nash, J.E. and Sutcliffe, J.V. (1970). River flow forecasting through conceptual models. Part 1: a discussion of principles, Journal of Hydrology 10, pp. 282-290. doi:10.1016/0022-1694(70)90255-6.

Pearson, K. (1920). Notes on the history of correlation. Biometrika, 13(1), 25-45. doi:10.2307/2331722.

Pfannerstill, M.; Guse, B.; Fohrer, N. (2014). Smart low flow signature metrics for an improved overall performance evaluation of hydrological models. Journal of Hydrology, 510, 447-458. doi:10.1016/j.jhydrol.2013.12.0

Pizarro, A.; Jorquera, J. (2024). Advancing objective functions in hydrological modelling: Integrating knowable moments for improved simulation accuracy. Journal of Hydrology, 634, 131071. doi:10.1016/j.jhydrol.2024.131071.

Pool, S.; Vis, M.; Seibert, J. (2018). Evaluating model performance: towards a non-parametric variant of the Kling-Gupta efficiency. Hydrological Sciences Journal, 63(13-14), pp.1941-1953. doi:/10.1080/02626667.2018.1552002.

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

Santos, L.; Thirel, G.; Perrin, C. (2018). Pitfalls in using log-transformed flows within the KGE criterion. doi:10.5194/hess-22-4583-2018.

Schaefli, B., Gupta, H. (2007). Do Nash values have value?. Hydrological Processes 21, 2075-2080. doi:10.1002/hyp.6825.

Schober, P.; Boer, C.; Schwarte, L.A. (2018). Correlation coefficients: appropriate use and interpretation. Anesthesia and Analgesia, 126(5), 1763-1768. doi:10.1213/ANE.0000000000002864.

Schuol, J.; Abbaspour, K.C.; Srinivasan, R.; Yang, H. (2008b), Estimation of freshwater availability in the West African sub-continent using the SWAT hydrologic model, Journal of Hydrology, 352(1-2), 30, doi:10.1016/j.jhydrol.2007.12.025

Sorooshian, S., Q. Duan, and V. K. Gupta. (1993). Calibration of rainfall-runoff models: Application of global optimization to the Sacramento Soil Moisture Accounting Model, Water Resources Research, 29 (4), 1185-1194, doi:10.1029/92WR02617.

Spearman, C. (1961). The Proof and Measurement of Association Between Two Things. In J. J. Jenkins and D. G. Paterson (Eds.), Studies in individual differences: The search for intelligence (pp. 45-58). Appleton-Century-Crofts. doi:10.1037/11491-005

Tang, G.; Clark, M.P.; Papalexiou, S.M. (2021). SC-earth: a station-based serially complete earth dataset from 1950 to 2019. Journal of Climate, 34(16), 6493-6511. doi:10.1175/JCLI-D-21-0067.1.

Yapo P.O.; Gupta H.V.; Sorooshian S. (1996). Automatic calibration of conceptual rainfall-runoff models: sensitivity to calibration data. Journal of Hydrology. v181 i1-4. 23-48. doi:10.1016/0022-1694(95)02918-4

Yilmaz, K.K., Gupta, H.V. ; Wagener, T. (2008), A process-based diagnostic approach to model evaluation: Application to the NWS distributed hydrologic model, Water Resources Research, 44, W09417, doi:10.1029/2007WR006716.

Willmott, C.J. (1981). On the validation of models. Physical Geography, 2, 184–194. doi:10.1080/02723646.1981.10642213.

Willmott, C.J. (1984). On the evaluation of model performance in physical geography. Spatial Statistics and Models, G. L. Gaile and C. J. Willmott, eds., 443-460. doi:10.1007/978-94-017-3048-8_23.

Willmott, C.J.; Ackleson, S.G. Davis, R.E.; Feddema, J.J.; Klink, K.M.; Legates, D.R.; O'Donnell, J.; Rowe, C.M. (1985), Statistics for the Evaluation and Comparison of Models, J. Geophys. Res., 90(C5), 8995-9005. doi:10.1029/JC090iC05p08995.

Willmott, C.J.; Matsuura, K. (2005). Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance, Climate Research, 30, 79-82, doi:10.3354/cr030079.

Willmott, C.J.; Matsuura, K.; Robeson, S.M. (2009). Ambiguities inherent in sums-of-squares-based error statistics, Atmospheric Environment, 43, 749-752, doi:10.1016/j.atmosenv.2008.10.005.

Willmott, C.J.; Robeson, S.M.; Matsuura, K. (2012). A refined index of model performance. International Journal of climatology, 32(13), pp.2088-2094. doi:10.1002/joc.2419.

Willmott, C.J.; Robeson, S.M.; Matsuura, K.; Ficklin, D.L. (2015). Assessment of three dimensionless measures of model performance. Environmental Modelling & Software, 73, pp.167-174. doi:10.1016/j.envsoft.2015.08.012

Zambrano-Bigiarini, M.; Bellin, A. (2012). Comparing goodness-of-fit measures for calibration of models focused on extreme events. EGU General Assembly 2012, Vienna, Austria, 22-27 Apr 2012, EGU2012-11549-1.

See Also

ggof, me, mae, mse, rmse, ubRMSE, nrmse, pbias, rsr, rSD, NSE, mNSE, rNSE, wNSE, wsNSE, d, dr, md, rd, cp, rPearson, R2, br2, VE, KGE, KGElf, KGEnp, , KGEkm, sKGE, APFB, HFB, rSpearman, pbiasfdc

Examples

```
##################
# Example 1: basic ideal case
obs <- 1:10
sim <- 1:10
gof(sim, obs)

obs <- 1:10
sim <- 2:11
gof(sim, obs)

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

```
# Example 2:
# Loading daily streamflows of the Ega River (Spain), from 1961 to 1970
data(EgaEnEstellaQts)
obs <- EgaEnEstellaQts

# Generating a simulated daily time series, initially equal to the observed series
sim <- obs

# Computing the 'gof' for the "best" (unattainable) case
gof(sim=sim, obs=obs)

##################
# Example 3: gof for simulated values equal to observations plus random noise
#            on the first half of the observed values.
#            This random noise has more relative importance for low flows than
#            for medium and high flows.

# Randomly changing the first 1826 elements of 'sim', by using a normal distribution
# with mean 10 and standard deviation equal to 1 (default of 'rnorm').
sim[1:1826] <- obs[1:1826] + rnorm(1826, mean=10)
ggof(sim, obs)

gof(sim=sim, obs=obs)

##################
# Example 4: gof for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' during computations.

gof(sim=sim, obs=obs, fun=log)

# Verifying the previous value:
lsim <- log(sim)
lobs <- log(obs)
gof(sim=lsim, obs=lobs)

##################
# Example 5: gof for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and adding the Pushpalatha2012 constant
#            during computations

gof(sim=sim, obs=obs, fun=log, epsilon.type="Pushpalatha2012")

# Verifying the previous value, with the epsilon value following Pushpalatha2012
eps  <- mean(obs, na.rm=TRUE)/100
lsim <- log(sim+eps)
lobs <- log(obs+eps)
gof(sim=lsim, obs=lobs)

##################
# Example 6: gof for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
```

```
#            logarithm to 'sim' and 'obs' and adding a user-defined constant
#            during computations

eps <- 0.01
gof(sim=sim, obs=obs, fun=log, epsilon.type="otherValue", epsilon.value=eps)

# Verifying the previous value:
lsim <- log(sim+eps)
lobs <- log(obs+eps)
gof(sim=lsim, obs=lobs)

##################
# Example 7: gof for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and using a user-defined factor
#            to multiply the mean of the observed values to obtain the constant
#            to be added to 'sim' and 'obs' during computations

fact <- 1/50
gof(sim=sim, obs=obs, fun=log, epsilon.type="otherFactor", epsilon.value=fact)

# Verifying the previous value:
eps  <- fact*mean(obs, na.rm=TRUE)
lsim <- log(sim+eps)
lobs <- log(obs+eps)
gof(sim=lsim, obs=lobs)

##################
# Example 8: gof for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying a
#            user-defined function to 'sim' and 'obs' during computations

fun1 <- function(x) {sqrt(x+1)}

gof(sim=sim, obs=obs, fun=fun1)

# Verifying the previous value, with the epsilon value following Pushpalatha2012
sim1 <- sqrt(sim+1)
obs1 <- sqrt(obs+1)
gof(sim=sim1, obs=obs1)

# Storing a matrix object with all the GoFs:
g <-  gof(sim, obs)

# Getting only the RMSE
g[4,1]
g["RMSE",]

## Not run:
# Writing all the GoFs into a TXT file
write.table(g, "GoFs.txt", col.names=FALSE, quote=FALSE)

# Getting the graphical representation of 'obs' and 'sim' along with the
```

```
# numeric goodness of fit
ggof(sim=sim, obs=obs)

## End(Not run)
```

HFB *High-flows bias*

Description

High flow bias between `sim` and `obs`, with treatment of missing values.

This function is designed to identify differences in high values. See Details.

Usage

```
HFB(sim, obs, ...)

## Default S3 method:
HFB(sim, obs, na.rm=TRUE,
             hQ.thr=0.1, start.month=1, out.PerYear=FALSE,
             fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
             epsilon.value=NA)

## S3 method for class 'data.frame'
HFB(sim, obs, na.rm=TRUE,
             hQ.thr=0.1, start.month=1, out.PerYear=FALSE,
             fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
             epsilon.value=NA)

## S3 method for class 'matrix'
HFB(sim, obs, na.rm=TRUE,
             hQ.thr=0.1, start.month=1, out.PerYear=FALSE,
             fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
             epsilon.value=NA)

## S3 method for class 'zoo'
HFB(sim, obs, na.rm=TRUE,
             hQ.thr=0.1, start.month=1, out.PerYear=FALSE,
             fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
             epsilon.value=NA)
```

Arguments

`sim`	numeric, zoo, matrix or data.frame with simulated values
`obs`	numeric, zoo, matrix or data.frame with observed values
`na.rm`	a logical value indicating whether 'NA' should be stripped before the computation proceeds. When an 'NA' value is found at the i-th position in `obs` **OR** `sim`, the i-th value of `obs` **AND** `sim` are removed before the computation.
`hQ.thr`	numeric, representing the exceedence probabiliy used to identify high flows in `obs`. All values in `obs` that are equal or higher than `quantile(obs, probs=(1-hQ.thr))` are considered as high flows. By default `hQ.thr=0.1`. On the other hand, the high values in `sim` are those located at the same i-th position than the i-th value of the `obs` deemed as high flows.
`start.month`	[OPTIONAL]. Only used when the (hydrological) year of interest is different from the calendar year. numeric in [1:12] indicating the starting month of the (hydrological) year. Numeric values in [1, 12] represent months in [January, December]. By default `start.month=1`.
`out.PerYear`	logical, indicating whether the output of this function has to include the median annual high-flows bias obtained for the individual years in `sim` and `obs` or not.
`fun`	function to be applied to `sim` and `obs` in order to obtain transformed values thereof before computing this goodness-of-fit index. The first argument MUST BE a numeric vector with any name (e.g., `x`), and additional arguments are passed using `...`.
`...`	arguments passed to `fun`, in addition to the mandatory first numeric vector.
`epsilon.type`	argument used to define a numeric value to be added to both `sim` and `obs` before applying `fun`. It is was designed to allow the use of logarithm and other similar functions that do not work with zero values. Valid values of `epsilon.type` are: 1) `"none"`: `sim` and `obs` are used by `fun` without the addition of any numeric value. This is the default option. 2) `"Pushpalatha2012"`: one hundredth (1/100) of the mean observed values is added to both `sim` and `obs` before applying `fun`, as described in Pushpalatha et al. (2012). 3) `"otherFactor"`: the numeric value defined in the `epsilon.value` argument is used to multiply the the mean observed values, instead of the one hundredth (1/100) described in Pushpalatha et al. (2012). The resulting value is then added to both `sim` and `obs`, before applying `fun`. 4) `"otherValue"`: the numeric value defined in the `epsilon.value` argument is directly added to both `sim` and `obs`, before applying `fun`.
`epsilon.value`	-) when `epsilon.type="otherValue"` it represents the numeric value to be added to both `sim` and `obs` before applying `fun`. -) when `epsilon.type="otherFactor"` it represents the numeric factor used to multiply the mean of the observed values, instead of the one hundredth (1/100) described in Pushpalatha et al. (2012). The resulting value is then added to both `sim` and `obs` before applying `fun`.

Details

The median annual high flow bias (HFB) is designed to drive the calibration of hydrological models focused in the reproduction of high-flow events.

The high flow bias (HFB) ranges from 0 to Inf, with an optimal value of 0. Higher values of HFB indicate stronger differences between the high values of `sim` and `obs`. Essentially, the closer to 0, the more similar the high values of `sim` and `obs` are.

The HFB function is inspired in the annual peak-flow bias (APFB) objective function proposed by Mizukami et al. (2019). However, it has four important diferences:

1) instead of considering only the observed annual peak flow in each year, it considers all the high flows in each year, where "high flows" are all the values above a user-defined quantile of the observed values, by default 0.9 (`hQ.thr=0.1`).

2) insted of considering only the simulated high flows for each year, which might occur in a date/time different from the date in which occurs the observed annual peak flow, it considers as many high simulated flows as the number of high observed flows for each year, each one in the exact same date/time in which the corresponding observed high flow occurred.

3) for each year, instead of using a single bias value (i.e., the bias in the single annual peak flow), it uses the median of all the bias in the user-defined high flows

4) when computing the final value of this metric, instead o using the mean of the annual values, it uses the median, in order to take a stronger representation of the bias when its distribution is not symetric.

Value

If `out.PerYear=FALSE`: numeric with the median high flow bias between `sim` and `obs`. If `sim` and `obs` are matrices, the output value is a vector, with the high flow bias between each column of `sim` and `obs`.

If `out.PerYear=TRUE`: a list of two elements:

`HFB.value` numeric with the median annual high flow bias between `sim` and `obs`. If `sim` and `obs` are matrices, the output value is a vector, with the median annual high flow bias between each column of `sim` and `obs`.

`HFB.PerYear` -) If `sim` and `obs` are not data.frame/matrix, the output is numeric, with the median high flow bias obtained for the individual years between `sim` and `obs`.

-) If `sim` and `obs` are data.frame/matrix, this output is a data.frame, with the median high flow bias obtained for the individual years between `sim` and `obs`.

Note

`obs` and `sim` has to have the same length/dimension

The missing values in `obs` and `sim` are removed before the computation proceeds, and only those positions with non-missing values in `obs` and `sim` are considered in the computation

Author(s)

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

References

Mizukami, N.; Rakovec, O.; Newman, A.J.; Clark, M.P.; Wood, A.W.; Gupta, H.V.; Kumar, R.: (2019). On the choice of calibration metrics for "high-flow" estimation using hydrologic models, Hydrology Earth System Sciences 23, 2601-2614, doi:10.5194/hess-23-2601-2019.

See Also

APFB, NSE, wNSE, , wsNSE, gof, ggof

Examples

```
##################
# Example 1: Looking at the difference between 'NSE', 'wNSE', and 'HFB'
# Loading daily streamflows of the Ega River (Spain), from 1961 to 1970
data(EgaEnEstellaQts)
obs <- EgaEnEstellaQts

# Simulated daily time series, created equal to the observed values and then
# random noise is added only to high flows, i.e., those equal or higher than
# the quantile 0.9 of the observed values.
sim      <- obs
hQ.thr   <- quantile(obs, probs=0.9, na.rm=TRUE)
hQ.index <- which(obs >= hQ.thr)
hQ.n     <- length(hQ.index)
sim[hQ.index] <- sim[hQ.index] + rnorm(hQ.n, mean=mean(sim[hQ.index], na.rm=TRUE))

# Traditional Nash-Sutcliffe eficiency
NSE(sim=sim, obs=obs)

# Weighted Nash-Sutcliffe efficiency (Hundecha and Bardossy, 2004)
wNSE(sim=sim, obs=obs)

# HFB (Garcia et al., 2017):
HFB(sim=sim, obs=obs)

##################
# Example 2:
# Loading daily streamflows of the Ega River (Spain), from 1961 to 1970
data(EgaEnEstellaQts)
obs <- EgaEnEstellaQts

# Generating a simulated daily time series, initially equal to the observed series
sim <- obs

# Computing the 'HFB' for the "best" (unattainable) case
HFB(sim=sim, obs=obs)

##################
# Example 3: HFB for simulated values created equal to the observed values and then
#            random noise is added only to high flows, i.e., those equal or higher than
#            the quantile 0.9 of the observed values.
```

```
sim           <- obs
hQ.thr        <- quantile(obs, hQ.thr=0.9, na.rm=TRUE)
hQ.index      <- which(obs >= hQ.thr)
hQ.n          <- length(hQ.index)
sim[hQ.index] <- sim[hQ.index] + rnorm(hQ.n, mean=mean(sim[hQ.index], na.rm=TRUE))
ggof(sim, obs)

HFB(sim=sim, obs=obs)

##################
# Example 4: HFB for simulated values created equal to the observed values and then
#            random noise is added only to high flows, i.e., those equal or higher than
#            the quantile 0.9 of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' during computations.

HFB(sim=sim, obs=obs, fun=log)

# Verifying the previous value:
lsim <- log(sim)
lobs <- log(obs)
HFB(sim=lsim, obs=lobs)


##################
# Example 5: HFB for simulated values created equal to the observed values and then
#            random noise is added only to high flows, i.e., those equal or higher than
#            the quantile 0.9 of the observed values and applying a
#            user-defined function to 'sim' and 'obs' during computations

fun1 <- function(x) {sqrt(x+1)}

HFB(sim=sim, obs=obs, fun=fun1)

# Verifying the previous value, with the epsilon value following Pushpalatha2012
sim1 <- sqrt(sim+1)
obs1 <- sqrt(obs+1)
HFB(sim=sim1, obs=obs1)
```

KGE *Kling-Gupta Efficiency*

Description

Kling-Gupta efficiency between sim and obs, with treatment of missing values.

This goodness-of-fit measure was developed by Gupta et al. (2009) to provide a diagnostically interesting decomposition of the Nash-Sutcliffe efficiency (and hence MSE), which facilitates the analysis of the relative importance of its different components (correlation, bias and variability) in the context of hydrological modelling.

Kling et al. (2012) proposed a revised version of this index (KGE') to ensure that the bias and variability ratios are not cross-correlated.

Tang et al. (2021) proposed a revised version of this index (KGE") to avoid the anomalously negative KGE' or KGE values when the mean value is close to zero.

For a short description of its three components and the numeric range of varios, pleae see Details.

Usage

```
KGE(sim, obs, ...)

## Default S3 method:
KGE(sim, obs, s=c(1,1,1), na.rm=TRUE, method=c("2009", "2012", "2021"),
             out.type=c("single", "full"), fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
             epsilon.value=NA)

## S3 method for class 'data.frame'
KGE(sim, obs, s=c(1,1,1), na.rm=TRUE, method=c("2009", "2012", "2021"),
             out.type=c("single", "full"), fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
             epsilon.value=NA)

## S3 method for class 'matrix'
KGE(sim, obs, s=c(1,1,1), na.rm=TRUE, method=c("2009", "2012", "2021"),
             out.type=c("single", "full"), fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
             epsilon.value=NA)

## S3 method for class 'zoo'
KGE(sim, obs, s=c(1,1,1), na.rm=TRUE, method=c("2009", "2012", "2021"),
             out.type=c("single", "full"), fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
             epsilon.value=NA)
```

Arguments

`sim`	numeric, zoo, matrix or data.frame with simulated values
`obs`	numeric, zoo, matrix or data.frame with observed values
`s`	numeric of length 3, representing the scaling factors to be used for re-scaling the criteria space before computing the Euclidean distance from the ideal point c(1,1,1), i.e., `s` elements are used for adjusting the emphasis on different components. The first elements is used for rescaling the Pearson product-moment correlation coefficient (`r`), the second element is used for rescaling `Alpha` and the third element is used for re-scaling `Beta`
`na.rm`	a logical value indicating whether 'NA' should be stripped before the computation proceeds. When an 'NA' value is found at the i-th position in `obs` **OR** `sim`, the i-th value of `obs` **AND** `sim` are removed before the computation.

`method` character, indicating the formula used to compute the variability ratio in the Kling-Gupta efficiency. Valid values are:

-) `2009`: the variability is defined as 'Alpha', the ratio of the standard deviation of `sim` values to the standard deviation of `obs`. This is the default option. See Gupta et al. (2009).

-) `2012`: the variability is defined as 'Gamma', the ratio of the coefficient of variation of `sim` values to the coefficient of variation of `obs`. See Kling et al. (2012).

-) `2021`: the bias is defined as 'Beta', the ratio of `mean(sim)` minus `mean(obs)` to the standard deviation of `obs`. The variability is defined as 'Alpha', the ratio of the standard deviation of `sim` values to the standard deviation of `obs`. See Tang et al. (2021).

`out.type` character, indicating the whether the output of the function has to include each one of the three terms used in the computation of the Kling-Gupta efficiency or not. Valid values are:

-) `single`: the output is a numeric with the Kling-Gupta efficiency only.

-) `full`: the output is a list of two elements: the first one with the Kling-Gupta efficiency, and the second is a numeric with 3 elements: the Pearson product-moment correlation coefficient ('r'), the ratio between the mean of the simulated values to the mean of observations ('Beta'), and the variability measure ('Gamma' or 'Alpha', depending on the value of `method`).

`fun` function to be applied to `sim` and `obs` in order to obtain transformed values thereof before computing the Kling-Gupta efficiency.

The first argument MUST BE a numeric vector with any name (e.g., `x`), and additional arguments are passed using `...`.

`...` arguments passed to `fun`, in addition to the mandatory first numeric vector.

`epsilon.type` argument used to define a numeric value to be added to both `sim` and `obs` before applying `fun`.

It is was designed to allow the use of logarithm and other similar functions that do not work with zero values.

Valid values of `epsilon.type` are:

1) `"none"`: `sim` and `obs` are used by `fun` without the addition of any nummeric value.

2) `"Pushpalatha2012"`: one hundredth (1/100) of the mean observed values is added to both `sim` and `obs` before applying `fun`, as described in Pushpalatha et al. (2012).

3) `"otherFactor"`: the numeric value defined in the `epsilon.value` argument is used to multiply the the mean observed values, instead of the one hundredth (1/100) described in Pushpalatha et al. (2012). The resulting value is then added to both `sim` and `obs`, before applying `fun`.

4) `"otherValue"`: the numeric value defined in the `epsilon.value` argument is directly added to both `sim` and `obs`, before applying `fun`.

`epsilon.value` -) when `epsilon.type="otherValue"` it represents the numeric value to be added to both `sim` and `obs` before applying `fun`.
-) when `epsilon.type="otherFactor"` it represents the numeric factor used to

multiply the mean of the observed values, instead of the one hundredth (1/100) described in Pushpalatha et al. (2012). The resulting value is then added to both sim and obs before applying fun.

Details

In the computation of this index, there are three main components involved:

1) r : the Pearson product-moment correlation coefficient. Ideal value is r=1.

2) Beta : the ratio between the mean of the simulated values and the mean of the observed ones. Ideal value is Beta=1.

3) vr : variability ratio, which could be computed using the standard deviation (Alpha) or the coefficient of variation (Gamma) of sim and obs, depending on the value of method:

3.1) Alpha: the ratio between the standard deviation of the simulated values and the standard deviation of the observed ones. Its ideal value is Alpha=1.

3.2) Gamma: the ratio between the coefficient of variation (CV) of the simulated values to the coefficient of variation of the observed ones. Its ideal value is Gamma=1.

For a full discussion of the Kling-Gupta index, and its advantages over the Nash-Sutcliffe efficiency (NSE) see Gupta et al. (2009).

Kling-Gupta efficiencies range from -Inf to 1. Essentially, the closer to 1, the more similar sim and obs are.

Knoben et al. (2019) showed that KGE values greater than -0.41 indicate that a model improves upon the mean flow benchmark, even if the model's KGE value is negative.

$$KGE = 1 - ED$$

$$ED = \sqrt{(s[1] * (r-1))^2 + (s[2] * (vr-1))^2 + (s[3] * (\beta-1))^2}$$

$$r = Pearson product - moment correlation coefficient$$

$$vr = \begin{cases} \alpha & , method = 2009 \\ \gamma & , method = 2012 \end{cases}$$

$$\beta = \mu_s / \mu_o$$

$$\alpha = \sigma_s / \sigma_o$$

$$\gamma = \frac{CV_s}{CV_o} = \frac{\sigma_s / \mu_s}{\sigma_o / \mu_o}$$

Value

If out.type=single: numeric with the Kling-Gupta efficiency between sim and obs. If sim and obs are matrices, the output value is a vector, with the Kling-Gupta efficiency between each column of sim and obs

If out.type=full: a list of two elements:

KGE.value numeric with the Kling-Gupta efficiency. If sim and obs are matrices, the output value is a vector, with the Kling-Gupta efficiency between each column of sim and obs

KGE.elements numeric with 3 elements: the Pearson product-moment correlation coefficient ('r'), the ratio between the mean of the simulated values to the mean of observations ('Beta'), and the variability measure ('Gamma' or 'Alpha', depending on the value of method). If sim and obs are matrices, the output value is a matrix, with the previous three elements computed for each column of sim and obs

Note

obs and sim has to have the same length/dimension

The missing values in obs and sim are removed before the computation proceeds, and only those positions with non-missing values in obs and sim are considered in the computation

Author(s)

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

References

Gupta, H.V.; Kling, H.; Yilmaz, K.K.; Martinez, G.F. (2009). Decomposition of the mean squared error and NSE performance criteria: Implications for improving hydrological modelling. Journal of hydrology, 377(1-2), 80-91. doi:10.1016/j.jhydrol.2009.08.003. ISSN 0022-1694.

Kling, H.; Fuchs, M.; Paulin, M. (2012). Runoff conditions in the upper Danube basin under an ensemble of climate change scenarios. Journal of Hydrology, 424, 264-277, doi:10.1016/j.jhydrol.2012.01.011.

Tang, G.; Clark, M.P.; Papalexiou, S.M. (2021). SC-earth: a station-based serially complete earth dataset from 1950 to 2019. Journal of Climate, 34(16), 6493-6511. doi:10.1175/JCLI-D-21-0067.1.

Santos, L.; Thirel, G.; Perrin, C. (2018). Pitfalls in using log-transformed flows within the KGE criterion. doi:10.5194/hess-22-4583-2018.

Knoben, W.J.; Freer, J.E.; Woods, R.A. (2019). Inherent benchmark or not? Comparing Nash-Sutcliffe and Kling-Gupta efficiency scores. Hydrology and Earth System Sciences, 23(10), 4323-4331. doi:10.5194/hess-23-4323-2019.

Mizukami, N.; Rakovec, O.; Newman, A.J.; Clark, M.P.; Wood, A.W.; Gupta, H.V.; Kumar, R. (2019). On the choice of calibration metrics for "high-flow" estimation using hydrologic models. doi:10.5194/hess-23-2601-2019.

Cinkus, G.; Mazzilli, N.; Jourde, H.; Wunsch, A.; Liesch, T.; Ravbar, N.; Chen, Z.; and Goldscheider, N. (2023). When best is the enemy of good - critical evaluation of performance criteria in hydrological models. Hydrology and Earth System Sciences 27, 2397-2411, doi:10.5194/hess-27-2397-2023.

See Also

KGElf, sKGE, KGEnp, gof, ggof

Examples

```
# Example1: basic ideal case
obs <- 1:10
sim <- 1:10
KGE(sim, obs)

obs <- 1:10
sim <- 2:11
KGE(sim, obs)

##################
# Example2: Looking at the difference between 'method=2009' and 'method=2012'
# Loading daily streamflows of the Ega River (Spain), from 1961 to 1970
data(EgaEnEstellaQts)
obs <- EgaEnEstellaQts

# Simulated daily time series, initially equal to twice the observed values
sim <- 2*obs

# Traditional Kling-Gupta eficiency (Gupta and Kling, 2009)
KGE(sim=sim, obs=obs, method="2009", out.type="full")

# KGE': Kling-Gupta eficiency 2012 (Kling et al.,2012)
KGE(sim=sim, obs=obs, method="2012", out.type="full")

# KGE'': Kling-Gupta eficiency 2021 (Tang et al.,2021)
KGE(sim=sim, obs=obs, method="2021", out.type="full")

##################
# Example3: KGE for simulated values equal to observations plus random noise
#           on the first half of the observed values
# Randomly changing the first 1826 elements of 'sim', by using a normal distribution
# with mean 10 and standard deviation equal to 1 (default of 'rnorm').
sim <- obs
sim[1:1826] <- obs[1:1826] + rnorm(1826, mean=10)

# Computing the new 'KGE'
KGE(sim=sim, obs=obs)

# Randomly changing the first 2000 elements of 'sim', by using a normal distribution
# with mean 10 and standard deviation equal to 1 (default of 'rnorm').
sim[1:2000] <- obs[1:2000] + rnorm(2000, mean=10)

# Traditional Kling-Gupta eficiency (Gupta and Kling, 2009)
KGE(sim=sim, obs=obs, method="2009", out.type="full")

# KGE': Kling-Gupta eficiency 2012 (Kling et al.,2012)
KGE(sim=sim, obs=obs, method="2012", out.type="full")

# KGE'': Kling-Gupta eficiency 2021 (Tang et al.,2021)
KGE(sim=sim, obs=obs, method="2021", out.type="full")
```

```
##################
# Example 4: KGE for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' during computations.

KGE(sim=sim, obs=obs, fun=log)

# Verifying the previous value:
lsim <- log(sim)
lobs <- log(obs)
KGE(sim=lsim, obs=lobs)

##################
# Example 5: KGE for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and adding the Pushpalatha2012 constant
#            during computations

KGE(sim=sim, obs=obs, fun=log, epsilon.type="Pushpalatha2012")

# Verifying the previous value, with the epsilon value following Pushpalatha2012
eps  <- mean(obs, na.rm=TRUE)/100
lsim <- log(sim+eps)
lobs <- log(obs+eps)
KGE(sim=lsim, obs=lobs)

##################
# Example 6: KGE for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and adding a user-defined constant
#            during computations

eps <- 0.01
KGE(sim=sim, obs=obs, fun=log, epsilon.type="otherValue", epsilon.value=eps)

# Verifying the previous value:
lsim <- log(sim+eps)
lobs <- log(obs+eps)
KGE(sim=lsim, obs=lobs)

##################
# Example 7: KGE for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and using a user-defined factor
#            to multiply the mean of the observed values to obtain the constant
#            to be added to 'sim' and 'obs' during computations

fact <- 1/50
KGE(sim=sim, obs=obs, fun=log, epsilon.type="otherFactor", epsilon.value=fact)

# Verifying the previous value:
eps  <- fact*mean(obs, na.rm=TRUE)
lsim <- log(sim+eps)
```

```
lobs <- log(obs+eps)
KGE(sim=lsim, obs=lobs)

##################
# Example 8: KGE for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying a
#            user-defined function to 'sim' and 'obs' during computations

fun1 <- function(x) {sqrt(x+1)}

KGE(sim=sim, obs=obs, fun=fun1)

# Verifying the previous value, with the epsilon value following Pushpalatha2012
sim1 <- sqrt(sim+1)
obs1 <- sqrt(obs+1)
KGE(sim=sim1, obs=obs1)
```

KGEkm *Kling-Gupta Efficiency with knowable-moments*

Description

Kling-Gupta efficiency between sim and obs, with use of knowable moments and treatment of missing values.

This goodness-of-fit measure was developed by Pizarro and Jorquera (2024), as a modification to the original Kling-Gupta efficiency (KGE) proposed by Gupta et al. (2009). See Details.

Usage

```
KGEkm(sim, obs, ...)

## Default S3 method:
KGEkm(sim, obs, s=c(1,1,1), na.rm=TRUE, method=c("2012", "2009", "2021"),
             out.type=c("single", "full"), fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
             epsilon.value=NA)

## S3 method for class 'data.frame'
KGEkm(sim, obs, s=c(1,1,1), na.rm=TRUE, method=c("2012", "2009", "2021"),
             out.type=c("single", "full"), fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
             epsilon.value=NA)

## S3 method for class 'matrix'
KGEkm(sim, obs, s=c(1,1,1), na.rm=TRUE, method=c("2012", "2009", "2021"),
             out.type=c("single", "full"), fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
             epsilon.value=NA)
```

```
## S3 method for class 'zoo'
KGEkm(sim, obs, s=c(1,1,1), na.rm=TRUE, method=c("2012", "2009", "2021"),
             out.type=c("single", "full"), fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
             epsilon.value=NA)
```

Arguments

`sim` numeric, zoo, matrix or data.frame with simulated values

`obs` numeric, zoo, matrix or data.frame with observed values

`s` numeric of length 3, representing the scaling factors to be used for re-scaling the criteria space before computing the Euclidean distance from the ideal point c(1,1,1), i.e., `s` elements are used for adjusting the emphasis on different components. The first elements is used for rescaling the Pearson product-moment correlation coefficient (`r`), the second element is used for rescaling `Alpha` and the third element is used for re-scaling `Beta`

`na.rm` a logical value indicating whether 'NA' should be stripped before the computation proceeds.
When an 'NA' value is found at the i-th position in `obs` **OR** `sim`, the i-th value of `obs` **AND** `sim` are removed before the computation.

`method` character, indicating the formula used to compute the variability ratio in the Kling-Gupta efficiency. Valid values are:

-) `2012`: the variability is defined as 'Gamma', the ratio of the coefficient of variation of `sim` values to the coefficient of variation of `obs`. See Pizarro and Jorquera (2024) and Kling et al. (2012).

-) `2009`: the variability is defined as 'Alpha', the ratio of the standard deviation of `sim` values to the standard deviation of `obs`. This is the default option. See Gupta et al. (2009).

-) `2021`: the bias is defined as 'Beta', the ratio of `mean(sim)` minus `mean(obs)` to the standard deviation of `obs`. The variability is defined as 'Alpha', the ratio of the standard deviation of `sim` values to the standard deviation of `obs`. See Tang et al. (2021).

`out.type` character, indicating the whether the output of the function has to include each one of the three terms used in the computation of the Kling-Gupta efficiency or not. Valid values are:

-) `single`: the output is a numeric with the Kling-Gupta efficiency only.

-) `full`: the output is a list of two elements: the first one with the Kling-Gupta efficiency, and the second is a numeric with 3 elements: the Pearson product-moment correlation coefficient ('r'), the ratio between the mean of the simulated values to the mean of observations ('Beta'), and the variability measure ('Gamma' or 'Alpha', depending on the value of `method`).

`fun` function to be applied to `sim` and `obs` in order to obtain transformed values thereof before computing the Kling-Gupta efficiency.

The first argument MUST BE a numeric vector with any name (e.g., `x`), and additional arguments are passed using `...`.

`...` arguments passed to `fun`, in addition to the mandatory first numeric vector.

`epsilon.type` argument used to define a numeric value to be added to both `sim` and `obs` before applying `fun`.

It is was designed to allow the use of logarithm and other similar functions that do not work with zero values.

Valid values of `epsilon.type` are:

1) `"none"`: `sim` and `obs` are used by `fun` without the addition of any nummeric value.

2) `"Pushpalatha2012"`: one hundredth (1/100) of the mean observed values is added to both `sim` and `obs` before applying `fun`, as described in Pushpalatha et al. (2012).

3) `"otherFactor"`: the numeric value defined in the `epsilon.value` argument is used to multiply the the mean observed values, instead of the one hundredth (1/100) described in Pushpalatha et al. (2012). The resulting value is then added to both `sim` and `obs`, before applying `fun`.

4) `"otherValue"`: the numeric value defined in the `epsilon.value` argument is directly added to both `sim` and `obs`, before applying `fun`.

`epsilon.value` -) when `epsilon.type="otherValue"` it represents the numeric value to be added to both `sim` and `obs` before applying `fun`.
-) when `epsilon.type="otherFactor"` it represents the numeric factor used to multiply the mean of the observed values, instead of the one hundredth (1/100) described in Pushpalatha et al. (2012). The resulting value is then added to both `sim` and `obs` before applying `fun`.

Details

Traditional objective functions, such as Nash-Sutcliffe Efficiency (NSE) and Kling-Gupta Efficiency (KGE), often make assumptions about data distribution and are sensitive to outliers. The Kling-Gupta Efficiency with knowable-moments (KGEkm) goodness-of-fit measure was developed by Pizarro and Jorquera (2024) to provide a reliable estimation and effective description of high-order statistics from typical hydrological samples and, therefore, reducing uncertainty in their estimation and computation of the KGE.

In the same line that the traditional Kling-Gupta efficiency, the KGEkm ranges from -Inf to 1. Essentially, the closer to 1, the more similar `sim` and `obs` are.

In the computation of this index, there are three main components involved:

1) `r` : the Pearson product-moment correlation coefficient. Ideal value is r=1.

2) `Beta` : the ratio between the mean of the simulated values and the mean of the observed ones. Ideal value is Beta=1.

3) `vr` : variability ratio, which could be computed using the standard deviation (`Alpha`) or the coefficient of variation (`Gamma`) of `sim` and `obs`, depending on the value of `method`:

3.1) `Alpha`: the ratio between the standard deviation of the simulated values and the standard deviation of the observed ones. Its ideal value is `Alpha=1`.

3.2) `Gamma`: the ratio between the coefficient of variation (`CV`) of the simulated values to the coefficient of variation of the observed ones. Its ideal value is `Gamma=1`.

$$KGEkm = 1 - ED$$

$$ED = \sqrt{(s[1] * (r-1))^2 + (s[2] * (vr-1))^2 + (s[3] * (\beta - 1))^2}$$

$$r = Pearson product - moment correlation coefficient$$

$$vr = \begin{cases} \alpha & , method = 2009 \\ \gamma & , method = 2012 \end{cases}$$

$$\beta = \mu_s / \mu_o$$

$$\alpha = \sigma_s / \sigma_o$$

$$\gamma = \frac{CV_s}{CV_o} = \frac{\sigma_s / \mu_s}{\sigma_o / \mu_o}$$

Value

If `out.type=single`: numeric with the Kling-Gupta efficiency between `sim` and `obs`. If `sim` and `obs` are matrices, the output value is a vector, with the Kling-Gupta efficiency between each column of `sim` and `obs`

If `out.type=full`: a list of two elements:

`KGEkm.value` numeric with the Kling-Gupta efficiency. If `sim` and `obs` are matrices, the output value is a vector, with the Kling-Gupta efficiency between each column of `sim` and `obs`

`KGEkm.elements` numeric with 3 elements: the Pearson product-moment correlation coefficient (‘r’), the ratio between the mean of the simulated values to the mean of observations (‘Beta’), and the variability measure (‘Gamma’ or ‘Alpha’, depending on the value of `method`). If `sim` and `obs` are matrices, the output value is a matrix, with the previous three elements computed for each column of `sim` and `obs`

Note

`obs` and `sim` has to have the same length/dimension

The missing values in `obs` and `sim` are removed before the computation proceeds, and only those positions with non-missing values in `obs` and `sim` are considered in the computation

Author(s)

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

References

Pizarro, A.; Jorquera, J. (2024). Advancing objective functions in hydrological modelling: Integrating knowable moments for improved simulation accuracy. Journal of Hydrology, 634, 131071. doi:10.1016/j.jhydrol.2024.131071.

Kling, H.; Fuchs, M.; Paulin, M. (2012). Runoff conditions in the upper Danube basin under an ensemble of climate change scenarios. Journal of Hydrology, 424, 264-277, doi:10.1016/j.jhydrol.2012.01.011.

Gupta, H. V.; Kling, H.; Yilmaz, K. K.; Martinez, G. F. (2009). Decomposition of the mean squared error and NSE performance criteria: Implications for improving hydrological modelling. Journal of hydrology, 377(1-2), 80-91. doi:10.1016/j.jhydrol.2009.08.003. ISSN 0022-1694.

Tang, G.; Clark, M. P.; Papalexiou, S. M. (2021). SC-earth: a station-based serially complete earth dataset from 1950 to 2019. Journal of Climate, 34(16), 6493-6511. doi:10.1175/JCLI-D-21-0067.1.

Santos, L.; Thirel, G.; Perrin, C. (2018). Pitfalls in using log-transformed flows within the KGEkm criterion. doi:10.5194/hess-22-4583-2018.

Knoben, W.J.; Freer, J.E.; Woods, R.A. (2019). Inherent benchmark or not? Comparing Nash-Sutcliffe and Kling-Gupta efficiency scores. Hydrology and Earth System Sciences, 23(10), 4323-4331. doi:10.5194/hess-23-4323-2019.

Cinkus, G., Mazzilli, N., Jourde, H., Wunsch, A., Liesch, T., Ravbar, N., Chen, Z., and Goldscheider, N. (2023). When best is the enemy of good - critical evaluation of performance criteria in hydrological models. Hydrology and Earth System Sciences 27, 2397-2411, doi:10.5194/hess-27-2397-2023

See Also

`KGE`, `KGElf`, `sKGE`, `KGEnp`, `gof`, `ggof`

Examples

```
# Example1: basic ideal case
obs <- 1:10
sim <- 1:10
KGEkm(sim, obs)

obs <- 1:10
sim <- 2:11
KGEkm(sim, obs)

##################
# Example2: Looking at the difference between 'method=2009' and 'method=2012'

# Loading daily streamflows of the Ega River (Spain), from 1961 to 1970
data(EgaEnEstellaQts)
obs <- EgaEnEstellaQts

# Simulated daily time series, initially equal to twice the observed values
sim <- 2*obs

# KGEkm 2012 (method="2012" is the default option for KGEkm)
KGEkm(sim=sim, obs=obs, method="2012", out.type="full")
```

```
# KGEkm 2009
KGEkm(sim=sim, obs=obs, method="2009", out.type="full")


##################
# Example 2: Looking at the difference between 'KGEkm', KGE', 'NSE', 'wNSE',
#            'wsNSE' and 'APFB' for detecting differences in high flows

# Loading daily streamflows of the Ega River (Spain), from 1961 to 1970
data(EgaEnEstellaQts)
obs <- EgaEnEstellaQts

# Simulated daily time series, created equal to the observed values and then
# random noise is added only to high flows, i.e., those equal or higher than
# the quantile 0.9 of the observed values.
sim      <- obs
hQ.thr   <- quantile(obs, probs=0.9, na.rm=TRUE)
hQ.index <- which(obs >= hQ.thr)
hQ.n     <- length(hQ.index)
sim[hQ.index] <- sim[hQ.index] + rnorm(hQ.n, mean=mean(sim[hQ.index], na.rm=TRUE))

# KGEkm (Pizarro and Jorquera, 2024; method='2012')
KGEkm(sim=sim, obs=obs)

# KGE': Kling-Gupta eficiency 2012 (Kling et al.,2012)
KGE(sim=sim, obs=obs, method="2012")

# Traditional Kling-Gupta eficiency (Gupta and Kling, 2009)
KGE(sim=sim, obs=obs)

# KGE'': Kling-Gupta eficiency 2021 (Tang et al.,2021)
KGE(sim=sim, obs=obs, method="2021")

# Traditional Nash-Sutcliffe eficiency (Nash and Sutcliffe, 1970)
NSE(sim=sim, obs=obs)

# Weighted Nash-Sutcliffe efficiency (Hundecha and Bardossy, 2004)
wNSE(sim=sim, obs=obs)

# wsNSE (Zambrano-Bigiarini and Bellin, 2012):
wsNSE(sim=sim, obs=obs)

# APFB (Mizukami et al., 2019):
APFB(sim=sim, obs=obs)


##################
# Example 4: Looking at the difference between 'KGE', 'NSE', 'wsNSE',
#            'dr', 'rd', 'md', and 'KGElf' for detecting
#            differences in low flows

# Loading daily streamflows of the Ega River (Spain), from 1961 to 1970
```

```
data(EgaEnEstellaQts)
obs <- EgaEnEstellaQts

# Simulated daily time series, created equal to the observed values and then
# random noise is added only to low flows, i.e., those equal or lower than
# the quantile 0.4 of the observed values.
sim      <- obs
lQ.thr   <- quantile(obs, probs=0.4, na.rm=TRUE)
lQ.index <- which(obs <= lQ.thr)
lQ.n     <- length(lQ.index)
sim[lQ.index] <- sim[lQ.index] + rnorm(lQ.n, mean=mean(sim[lQ.index], na.rm=TRUE))

# KGEkm (Pizarro and Jorquera, 2024; method='2012')
KGEkm(sim=sim, obs=obs)

# KGE': Kling-Gupta eficiency 2012 (Kling et al.,2012)
KGE(sim=sim, obs=obs, method="2012")

# Traditional Kling-Gupta eficiency (Gupta and Kling, 2009)
KGE(sim=sim, obs=obs)

# KGE'': Kling-Gupta eficiency 2021 (Tang et al.,2021)
KGE(sim=sim, obs=obs, method="2021")

# Traditional Nash-Sutcliffe eficiency (Nash and Sutcliffe, 1970)
NSE(sim=sim, obs=obs)

# Weighted seasonal Nash-Sutcliffe efficiency (Zambrano-Bigiarini and Bellin, 2012):
wsNSE(sim=sim, obs=obs, lambda=0.05, j=1/2)

# Refined Index of Agreement (Willmott et al., 2012):
dr(sim=sim, obs=obs)

# Relative Index of Agreement (Krause et al., 2005):
rd(sim=sim, obs=obs)

# Modified Index of Agreement (Krause et al., 2005):
md(sim=sim, obs=obs)

# KGElf (Garcia et al., 2017):
KGElf(sim=sim, obs=obs)


##################
# Example 5: KGEkm for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' during computations.

KGEkm(sim=sim, obs=obs, fun=log)

# Verifying the previous value:
lsim <- log(sim)
lobs <- log(obs)
```

```
KGEkm(sim=lsim, obs=lobs)

##################
# Example 6: KGEkm for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and adding the Pushpalatha2012 constant
#            during computations

KGEkm(sim=sim, obs=obs, fun=log, epsilon.type="Pushpalatha2012")

# Verifying the previous value, with the epsilon value following Pushpalatha2012
eps  <- mean(obs, na.rm=TRUE)/100
lsim <- log(sim+eps)
lobs <- log(obs+eps)
KGEkm(sim=lsim, obs=lobs)

##################
# Example 7: KGEkm for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and adding a user-defined constant
#            during computations

eps <- 0.01
KGEkm(sim=sim, obs=obs, fun=log, epsilon.type="otherValue", epsilon.value=eps)

# Verifying the previous value:
lsim <- log(sim+eps)
lobs <- log(obs+eps)
KGEkm(sim=lsim, obs=lobs)

##################
# Example 8: KGEkm for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and using a user-defined factor
#            to multiply the mean of the observed values to obtain the constant
#            to be added to 'sim' and 'obs' during computations

fact <- 1/50
KGEkm(sim=sim, obs=obs, fun=log, epsilon.type="otherFactor", epsilon.value=fact)

# Verifying the previous value:
eps  <- fact*mean(obs, na.rm=TRUE)
lsim <- log(sim+eps)
lobs <- log(obs+eps)
KGEkm(sim=lsim, obs=lobs)

##################
# Example 9: KGEkm for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying a
#            user-defined function to 'sim' and 'obs' during computations

fun1 <- function(x) {sqrt(x+1)}
```

```
KGEkm(sim=sim, obs=obs, fun=fun1)

# Verifying the previous value, with the epsilon value following Pushpalatha2012
sim1 <- sqrt(sim+1)
obs1 <- sqrt(obs+1)
KGEkm(sim=sim1, obs=obs1)
```

KGElf *Kling-Gupta Efficiency for low values*

Description

Kling-Gupta efficiency between sim and obs, with focus on low (streamflow) values and treatment of missing values.

This goodness-of-fit measure was developed by Garcia et al. (2017), as a modification to the original Kling-Gupta efficiency (KGE) proposed by Gupta et al. (2009). See Details.

Usage

```
KGElf(sim, obs, ...)

## Default S3 method:
KGElf(sim, obs, s=c(1,1,1), na.rm=TRUE, method=c("2009", "2012", "2021"),
          epsilon.type=c("Pushpalatha2012", "otherFactor", "otherValue", "none"),
               epsilon.value=NA, ...)

## S3 method for class 'data.frame'
KGElf(sim, obs, s=c(1,1,1), na.rm=TRUE, method=c("2009", "2012", "2021"),
          epsilon.type=c("Pushpalatha2012", "otherFactor", "otherValue", "none"),
               epsilon.value=NA, ...)

## S3 method for class 'matrix'
KGElf(sim, obs, s=c(1,1,1), na.rm=TRUE, method=c("2009", "2012", "2021"),
          epsilon.type=c("Pushpalatha2012", "otherFactor", "otherValue", "none"),
               epsilon.value=NA, ...)

## S3 method for class 'zoo'
KGElf(sim, obs, s=c(1,1,1), na.rm=TRUE, method=c("2009", "2012", "2021"),
          epsilon.type=c("Pushpalatha2012", "otherFactor", "otherValue", "none"),
               epsilon.value=NA, ...)
```

Arguments

sim	numeric, zoo, matrix or data.frame with simulated values
obs	numeric, zoo, matrix or data.frame with observed values

s
: numeric of length 3, representing the scaling factors to be used for re-scaling the criteria space before computing the Euclidean distance from the ideal point c(1,1,1), i.e., s elements are used for adjusting the emphasis on different components. The first elements is used for rescaling the Pearson product-moment correlation coefficient (r), the second element is used for rescaling Alpha and the third element is used for re-scaling Beta

na.rm
: a logical value indicating whether 'NA' should be stripped before the computation proceeds.
When an 'NA' value is found at the i-th position in obs **OR** sim, the i-th value of obs **AND** sim are removed before the computation.

method
: character, indicating the formula used to compute the variability ratio in the Kling-Gupta efficiency. Valid values are:

-) 2009: the variability is defined as 'Alpha', the ratio of the standard deviation of sim values to the standard deviation of obs. This is the default option. See Gupta et al. (2009).

-) 2012: the variability is defined as 'Gamma', the ratio of the coefficient of variation of sim values to the coefficient of variation of obs. See Kling et al. (2012).

-) 2021: the bias is defined as 'Beta', the ratio of mean(sim) minus mean(obs) to the standard deviation of obs. The variability is defined as 'Alpha', the ratio of the standard deviation of sim values to the standard deviation of obs. See Tang et al. (2021).

epsilon.type
: argument used to define a numeric value to be added to both sim and obs before applying fun.

It is designed to allow the use of logarithm and other similar functions that do not work with zero values.

Valid values of epsilon.type are:

1) "Pushpalatha2012": one hundredth (1/100) of the mean observed values is added to both sim and obs before applying fun, as described in Pushpalatha et al. (2012). This is the default option.

2) "otherFactor": the numeric value defined in the epsilon.value argument is used to multiply the the mean observed values, instead of the one hundredth (1/100) described in Pushpalatha et al. (2012). The resulting value is then added to both sim and obs, before applying FUN.

3) "otherValue": the numeric value defined in the epsilon.value argument is directly added to both sim and obs, before applying fun.

4) "none": sim and obs are used by fun without the addition of any numeric value.

epsilon.value
: -) when epsilon.type="otherValue" it represents the numeric value to be added to both sim and obs before applying fun.
-) when epsilon.type="otherFactor" it represents the numeric factor used to multiply the mean of the observed values, instead of the one hundredth (1/100) described in Pushpalatha et al. (2012). The resulting value is then added to both sim and obs before applying fun.

...
: further arguments passed to or from other methods.

Details

Garcia et al. (2017) tested different objective functions and found that the mean value of the KGE applied to the streamflows (i.e., KGE(Q)) and the KGE applied to the inverse of the streamflows (i.e., KGE(1/Q) is able to provide a an aceptable representation of low-flow indices important for water management. They also found that KGE applied to a transformation of streamflow values (e.g., log) is inadequate to capture low-flow indices important for water management.

The robustness of their findings depends more on the climate variability rather than the objective function, and they are insensitive to the hydrological model used in the evaluation.

$$KGE_{lf} = \frac{KGE(Q) + KGE(1/Q)}{2}$$

Traditional Kling-Gupta efficiencies (Gupta et al., 2009; Kling et al., 2012) range from -Inf to 1 and, therefore, KGElf should also range from -Inf to 1. Essentially, the closer to 1, the more similar `sim` and `obs` are.

Knoben et al. (2019) showed that traditional Kling-Gupta (Gupta et al., 2009; Kling et al., 2012) values greater than -0.41 indicate that a model improves upon the mean flow benchmark, even if the model's KGE value is negative.

Value

numeric with the Kling-Gupta efficiency for low flows between `sim` and `obs`.

If `sim` and `obs` are matrices, the output value is a vector, with the Kling-Gupta efficiency between each column of `sim` and `obs`

Note

`obs` and `sim` has to have the same length/dimension

The missing values in `obs` and `sim` are removed before the computation proceeds, and only those positions with non-missing values in `obs` and `sim` are considered in the computation

Author(s)

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

References

Garcia, F.; Folton, N.; Oudin, L. (2017). Which objective function to calibrate rainfall-runoff models for low-flow index simulations?. Hydrological sciences journal, 62(7), 1149-1166. doi:10.1080/02626667.2017.1308511.

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

Pfannerstill, M.; Guse, B.; Fohrer, N. (2014). Smart low flow signature metrics for an improved overall performance evaluation of hydrological models. Journal of Hydrology, 510, 447-458. doi:10.1016/j.jhydrol.2013.12.0

Gupta, H. V.; Kling, H.; Yilmaz, K. K.; Martinez, G. F. (2009). Decomposition of the mean squared error and NSE performance criteria: Implications for improving hydrological modelling. Journal of hydrology, 377(1-2), 80-91. doi:10.1016/j.jhydrol.2009.08.003. ISSN 0022-1694.

Kling, H.; Fuchs, M.; Paulin, M. (2012). Runoff conditions in the upper Danube basin under an ensemble of climate change scenarios. Journal of Hydrology, 424, 264-277, doi:10.1016/j.jhydrol.2012.01.011.

Santos, L.; Thirel, G.; Perrin, C. (2018). Pitfalls in using log-transformed flows within the KGE criterion. doi:10.5194/hess-22-4583-2018.

Knoben, W. J.; Freer, J. E.; Woods, R. A. (2019). Inherent benchmark or not? Comparing Nash-Sutcliffe and Kling-Gupta efficiency scores. Hydrology and Earth System Sciences, 23(10), 4323-4331. doi:10.5194/hess-23-4323-2019.

See Also

KGE, KGEnp, sKGE, gof, ggof

Examples

```
##################
# Example1: basic ideal case
obs <- 1:10
sim <- 1:10
KGElf(sim, obs)

obs <- 1:10
sim <- 2:11
KGElf(sim, obs)

##################
# Example2: Looking at the difference between 'method=2009' and 'method=2012'
# Loading daily streamflows of the Ega River (Spain), from 1961 to 1970
data(EgaEnEstellaQts)
obs <- EgaEnEstellaQts

# Simulated daily time series, initially equal to twice the observed values
sim <- 2*obs

# KGE 2009
KGE(sim=sim, obs=obs, method="2009", out.type="full")

# KGE 2012
KGE(sim=sim, obs=obs, method="2012", out.type="full")

# KGElf (Garcia et al., 2017):
KGElf(sim=sim, obs=obs, method="2012")

##################
# Example3: KGElf for simulated values equal to observations plus random noise
#           on the first half of the observed values.
#           This random noise has more relative importance for low flows than
#           for medium and high flows.
```

```
# Randomly changing the first 1826 elements of 'sim', by using a normal distribution
# with mean 10 and standard deviation equal to 1 (default of 'rnorm').
sim <- obs
sim[1:1826] <- obs[1:1826] + rnorm(1826, mean=10)
ggof(sim, obs)

# Computing 'KGElf'
KGElf(sim=sim, obs=obs)

##################
# Example 4: KGElf for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' during computations.

KGElf(sim=sim, obs=obs, fun=log)

# Verifying the previous value:
lsim <- log(sim)
lobs <- log(obs)
KGElf(sim=lsim, obs=lobs)

##################
# Example 5: KGElf for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and adding the Pushpalatha2012 constant
#            during computations

KGElf(sim=sim, obs=obs, fun=log, epsilon.type="Pushpalatha2012")

# Verifying the previous value, with the epsilon value following Pushpalatha2012
eps  <- mean(obs, na.rm=TRUE)/100
lsim <- log(sim+eps)
lobs <- log(obs+eps)
KGElf(sim=lsim, obs=lobs)

##################
# Example 6: KGElf for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and adding a user-defined constant
#            during computations

eps <- 0.01
KGElf(sim=sim, obs=obs, fun=log, epsilon.type="otherValue", epsilon.value=eps)

# Verifying the previous value:
lsim <- log(sim+eps)
lobs <- log(obs+eps)
KGElf(sim=lsim, obs=lobs)

##################
# Example 7: KGElf for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and using a user-defined factor
```

```
#            to multiply the mean of the observed values to obtain the constant
#            to be added to 'sim' and 'obs' during computations

fact <- 1/50
KGElf(sim=sim, obs=obs, fun=log, epsilon.type="otherFactor", epsilon.value=fact)

# Verifying the previous value:
eps  <- fact*mean(obs, na.rm=TRUE)
lsim <- log(sim+eps)
lobs <- log(obs+eps)
KGElf(sim=lsim, obs=lobs)

##################
# Example 8: KGElf for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying a
#            user-defined function to 'sim' and 'obs' during computations

fun1 <- function(x) {sqrt(x+1)}

KGElf(sim=sim, obs=obs, fun=fun1)

# Verifying the previous value, with the epsilon value following Pushpalatha2012
sim1 <- sqrt(sim+1)
obs1 <- sqrt(obs+1)
KGElf(sim=sim1, obs=obs1)
```

KGEnp *Non-parametric version of the Kling-Gupta Efficiency*

Description

Non-parametric Kling-Gupta efficiency between `sim` and `obs`, with treatment of missing values.

This goodness-of-fit measure was developed by Pool et al. (2018), as a non-parametric alternative to the original Kling-Gupta efficiency (KGE) proposed by Gupta et al. (2009). See Details.

Usage

```
KGEnp(sim, obs, ...)

## Default S3 method:
KGEnp(sim, obs, na.rm=TRUE, out.type=c("single", "full"), fun=NULL, ...,
          epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
               epsilon.value=NA)

## S3 method for class 'data.frame'
KGEnp(sim, obs, na.rm=TRUE, out.type=c("single", "full"), fun=NULL, ...,
          epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
               epsilon.value=NA)
```

```
## S3 method for class 'matrix'
KGEnp(sim, obs, na.rm=TRUE, out.type=c("single", "full"), fun=NULL, ...,
          epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
               epsilon.value=NA)

## S3 method for class 'zoo'
KGEnp(sim, obs, na.rm=TRUE, out.type=c("single", "full"), fun=NULL, ...,
          epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
               epsilon.value=NA)
```

Arguments

`sim` numeric, zoo, matrix or data.frame with simulated values

`obs` numeric, zoo, matrix or data.frame with observed values

`na.rm` a logical value indicating whether 'NA' should be stripped before the computation proceeds.
When an 'NA' value is found at the i-th position in `obs` **OR** `sim`, the i-th value of `obs` **AND** `sim` are removed before the computation.

`out.type` character, indicating the whether the output of the function has to include each one of the three terms used in the computation of the Kling-Gupta efficiency or not. Valid values are:

-) `single`: the output is a numeric with the Kling-Gupta efficiency only.

-) `full`: the output is a list of two elements: the first one with the Kling-Gupta efficiency, and the second is a numeric with 3 elements: the Spearman rank correlation coefficient ('rSpearman'), the ratio between the mean of the simulated values to the mean of observations ('Beta'), and the variability measure ('Alpha').

`fun` function to be applied to `sim` and `obs` in order to obtain transformed values thereof before computing this goodness-of-fit index.

The first argument MUST BE a numeric vector with any name (e.g., `x`), and additional arguments are passed using `...`.

`...` arguments passed to `fun`, in addition to the mandatory first numeric vector.

`epsilon.type` argument used to define a numeric value to be added to both `sim` and `obs` before applying `fun`.

It is was designed to allow the use of logarithm and other similar functions that do not work with zero values.

Valid values of `epsilon.type` are:

1) `"none"`: `sim` and `obs` are used by `fun` without the addition of any numeric value. This is the default option.

2) `"Pushpalatha2012"`: one hundredth (1/100) of the mean observed values is added to both `sim` and `obs` before applying `fun`, as described in Pushpalatha et al. (2012).

3) `"otherFactor"`: the numeric value defined in the `epsilon.value` argument is used to multiply the the mean observed values, instead of the one hundredth

(1/100) described in Pushpalatha et al. (2012). The resulting value is then added to both sim and obs, before applying fun.

4) "otherValue": the numeric value defined in the epsilon.value argument is directly added to both sim and obs, before applying fun.

epsilon.value -) when epsilon.type="otherValue" it represents the numeric value to be added to both sim and obs before applying fun.
-) when epsilon.type="otherFactor" it represents the numeric factor used to multiply the mean of the observed values, instead of the one hundredth (1/100) described in Pushpalatha et al. (2012). The resulting value is then added to both sim and obs before applying fun.

Details

This non-parametric verison of the Kling-Gupta efficiency keeps the bias term Alpha (mean(sim) / mean(obs)), but for correlation uses the Spearman rank coefficient instead of the Pearson product-moment coefficient; and for variability it uses the normalized flow-duration curve instead of the standard deviation (or coefficient of variation).

The proposed non-parametric based multi-objective function can be seen as a useful alternative to existing performance measures when aiming at acceptable simulations of multiple hydrograph aspects (Pool et al., 2018).

$$KGE_{np} = 1 - ED$$
$$ED = \sqrt{((\rho - 1)^2 + (\alpha - 1)^2 + (\beta - 1)^2}$$
$$\rho = \text{Spearman rank correlation coefficient}$$
$$\alpha = 1 - 0.5 * sum(sim(I(k))/(n * \mu_s) - obs(J(k))/(n * \mu_o))$$
$$\beta = \mu_s/\mu_o$$

Traditional Kling-Gupta efficiencies (Gupta et al., 2009; Kling et al., 2012) range from -Inf to 1, and therefore KGEnp should do so. Essentially, the closer to 1, the more similar sim and obs are.

Knoben et al. (2019) showed that traditional Kling-Gupta (Gupta et al., 2009; Kling et al., 2012) values greater than -0.41 indicate that a model improves upon the mean flow benchmark, even if the model's KGE value is negative.

Value

numeric with the non-parametric Kling-Gupta efficiency between sim and obs.
If sim and obs are matrices, the output value is a vector, with the non-parametric Kling-Gupta efficiency between each column of sim and obs

Note

obs and sim has to have the same length/dimension

The missing values in obs and sim are removed before the computation proceeds, and only those positions with non-missing values in obs and sim are considered in the computation

Author(s)

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

References

Pool, S.; Vis, M.; Seibert, J. (2018). Evaluating model performance: towards a non-parametric variant of the Kling-Gupta efficiency. Hydrological Sciences Journal, 63(13-14), pp.1941-1953. doi:/10.1080/02626667.2018.1552002.

Garcia, F.; Folton, N.; Oudin, L. (2017). Which objective function to calibrate rainfall-runoff models for low-flow index simulations?. Hydrological sciences journal, 62(7), 1149-1166. doi:10.1080/02626667.2017.1308511.

Gupta, H. V.; Kling, H.; Yilmaz, K. K.; Martinez, G. F. (2009). Decomposition of the mean squared error and NSE performance criteria: Implications for improving hydrological modelling. Journal of hydrology, 377(1-2), 80-91. doi:10.1016/j.jhydrol.2009.08.003. ISSN 0022-1694.

Kling, H.; Fuchs, M.; Paulin, M. (2012). Runoff conditions in the upper Danube basin under an ensemble of climate change scenarios. Journal of Hydrology, 424, 264-277, doi:10.1016/j.jhydrol.2012.01.011.

Santos, L.; Thirel, G.; Perrin, C. (2018). Pitfalls in using log-transformed flows within the KGE criterion. doi:10.5194/hess-22-4583-2018.

Knoben, W.J.; Freer, J.E.; Woods, R.A. (2019). Inherent benchmark or not? Comparing Nash-Sutcliffe and Kling-Gupta efficiency scores. Hydrology and Earth System Sciences, 23(10), 4323-4331. doi:10.5194/hess-23-4323-2019.

See Also

KGE, KGElf, sKGE, gof, ggof

Examples

```
# Example1: basic ideal case
obs <- 1:10
sim <- 1:10
KGEnp(sim, obs)

obs <- 1:10
sim <- 2:11
KGEnp(sim, obs)

##################
# Example2: Looking at the difference between 'method=2009' and 'method=2012'
# Loading daily streamflows of the Ega River (Spain), from 1961 to 1970
data(EgaEnEstellaQts)
obs <- EgaEnEstellaQts

# Simulated daily time series, initially equal to twice the observed values
sim <- 2*obs

# KGE 2009
KGE(sim=sim, obs=obs, method="2009", out.type="full")

# KGE 2012
```

```
KGE(sim=sim, obs=obs, method="2012", out.type="full")

# KGEnp (Pool et al., 2018):
KGEnp(sim=sim, obs=obs)

##################
# Example3: KGEnp for simulated values equal to observations plus random noise
#           on the first half of the observed values
# Randomly changing the first 1826 elements of 'sim', by using a normal distribution
# with mean 10 and standard deviation equal to 1 (default of 'rnorm').
sim <- obs
sim[1:1826] <- obs[1:1826] + rnorm(1826, mean=10)

# Computing the new 'KGEnp'
KGEnp(sim=sim, obs=obs)

# Randomly changing the first 2000 elements of 'sim', by using a normal distribution
# with mean 10 and standard deviation equal to 1 (default of 'rnorm').
sim[1:2000] <- obs[1:2000] + rnorm(2000, mean=10)

# Computing the new 'KGEnp'
KGEnp(sim=sim, obs=obs)

##################
# Example 4: KGEnp for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' during computations.

KGEnp(sim=sim, obs=obs, fun=log)

# Verifying the previous value:
lsim <- log(sim)
lobs <- log(obs)
KGEnp(sim=lsim, obs=lobs)

##################
# Example 5: KGEnp for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and adding the Pushpalatha2012 constant
#            during computations

KGEnp(sim=sim, obs=obs, fun=log, epsilon.type="Pushpalatha2012")

# Verifying the previous value, with the epsilon value following Pushpalatha2012
eps  <- mean(obs, na.rm=TRUE)/100
lsim <- log(sim+eps)
lobs <- log(obs+eps)
KGEnp(sim=lsim, obs=lobs)

##################
# Example 6: KGEnp for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and adding a user-defined constant
```

```
#            during computations

eps <- 0.01
KGEnp(sim=sim, obs=obs, fun=log, epsilon.type="otherValue", epsilon.value=eps)

# Verifying the previous value:
lsim <- log(sim+eps)
lobs <- log(obs+eps)
KGEnp(sim=lsim, obs=lobs)

##################
# Example 7: KGEnp for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and using a user-defined factor
#            to multiply the mean of the observed values to obtain the constant
#            to be added to 'sim' and 'obs' during computations

fact <- 1/50
KGEnp(sim=sim, obs=obs, fun=log, epsilon.type="otherFactor", epsilon.value=fact)

# Verifying the previous value:
eps  <- fact*mean(obs, na.rm=TRUE)
lsim <- log(sim+eps)
lobs <- log(obs+eps)
KGEnp(sim=lsim, obs=lobs)

##################
# Example 8: KGEnp for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying a
#            user-defined function to 'sim' and 'obs' during computations

fun1 <- function(x) {sqrt(x+1)}

KGEnp(sim=sim, obs=obs, fun=fun1)

# Verifying the previous value, with the epsilon value following Pushpalatha2012
sim1 <- sqrt(sim+1)
obs1 <- sqrt(obs+1)
KGEnp(sim=sim1, obs=obs1)
```

mae *Mean Absolute Error*

Description

Mean absolute error between sim and obs, in the same units of them, with treatment of missing values.

Usage

```
mae(sim, obs, ...)

## Default S3 method:
mae(sim, obs, na.rm=TRUE, fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
             epsilon.value=NA)

## S3 method for class 'data.frame'
mae(sim, obs, na.rm=TRUE, fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
             epsilon.value=NA)

## S3 method for class 'matrix'
mae(sim, obs, na.rm=TRUE, fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
             epsilon.value=NA)

## S3 method for class 'zoo'
mae(sim, obs, na.rm=TRUE, fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
             epsilon.value=NA)
```

Arguments

`sim` numeric, zoo, matrix or data.frame with simulated values

`obs` numeric, zoo, matrix or data.frame with observed values

`na.rm` a logical value indicating whether 'NA' should be stripped before the computation proceeds.
When an 'NA' value is found at the i-th position in `obs` **OR** `sim`, the i-th value of `obs` **AND** `sim` are removed before the computation.

`fun` function to be applied to `sim` and `obs` in order to obtain transformed values thereof before computing this goodness-of-fit index.
The first argument MUST BE a numeric vector with any name (e.g., `x`), and additional arguments are passed using `...`.

`...` arguments passed to `fun`, in addition to the mandatory first numeric vector.

`epsilon.type` argument used to define a numeric value to be added to both `sim` and `obs` before applying `fun`.
It is was designed to allow the use of logarithm and other similar functions that do not work with zero values.
Valid values of `epsilon.type` are:
1) `"none"`: `sim` and `obs` are used by `fun` without the addition of any numeric value. This is the default option.
2) `"Pushpalatha2012"`: one hundredth (1/100) of the mean observed values is added to both `sim` and `obs` before applying `fun`, as described in Pushpalatha et al. (2012).

3) "otherFactor": the numeric value defined in the epsilon.value argument is used to multiply the the mean observed values, instead of the one hundredth (1/100) described in Pushpalatha et al. (2012). The resulting value is then added to both sim and obs, before applying fun.

4) "otherValue": the numeric value defined in the epsilon.value argument is directly added to both sim and obs, before applying fun.

epsilon.value
-) when epsilon.type="otherValue" it represents the numeric value to be added to both sim and obs before applying fun.
-) when epsilon.type="otherFactor" it represents the numeric factor used to multiply the mean of the observed values, instead of the one hundredth (1/100) described in Pushpalatha et al. (2012). The resulting value is then added to both sim and obs before applying fun.

Details

$$mae = \frac{1}{N} \sum_{i=1}^{N} |S_i - O_i)|$$

Value

Mean absolute error between sim and obs.

If sim and obs are matrixes, the returned value is a vector, with the mean absolute error between each column of sim and obs.

Note

obs and sim have to have the same length/dimension

The missing values in obs and sim are removed before the computation proceeds, and only those positions with non-missing values in obs and sim are considered in the computation

Author(s)

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

References

https://en.wikipedia.org/wiki/Mean_absolute_error

Willmott, C.J.; Matsuura, K. (2005). Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance, Climate Research, 30, 79-82, doi:10.3354/cr030079.

Chai, T.; Draxler, R.R. (2014). Root mean square error (RMSE) or mean absolute error (MAE)? - Arguments against avoiding RMSE in the literature, Geoscientific Model Development, 7, 1247-1250. doi:10.5194/gmd-7-1247-2014.

Hodson, T.O. (2022). Root-mean-square error (RMSE) or mean absolute error (MAE): when to use them or not, Geoscientific Model Development, 15, 5481-5487, doi:10.5194/gmd-15-5481-2022.

See Also

pbias, pbiasfdc, mse, rmse, ubRMSE, nrmse, ssq, gof, ggof

Examples

```
##################
# Example 1: basic ideal case
obs <- 1:10
sim <- 1:10
mae(sim, obs)

obs <- 1:10
sim <- 2:11
mae(sim, obs)

##################
# Example 2:
# Loading daily streamflows of the Ega River (Spain), from 1961 to 1970
data(EgaEnEstellaQts)
obs <- EgaEnEstellaQts

# Generating a simulated daily time series, initially equal to the observed series
sim <- obs

# Computing the 'mae' for the "best" (unattainable) case
mae(sim=sim, obs=obs)

##################
# Example 3: mae for simulated values equal to observations plus random noise
#            on the first half of the observed values.
#            This random noise has more relative importance for ow flows than
#            for medium and high flows.

# Randomly changing the first 1826 elements of 'sim', by using a normal distribution
# with mean 10 and standard deviation equal to 1 (default of 'rnorm').
sim[1:1826] <- obs[1:1826] + rnorm(1826, mean=10)
ggof(sim, obs)

mae(sim=sim, obs=obs)

##################
# Example 4: mae for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' during computations.

mae(sim=sim, obs=obs, fun=log)

# Verifying the previous value:
lsim <- log(sim)
lobs <- log(obs)
mae(sim=lsim, obs=lobs)
```

```
##################
# Example 5: mae for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and adding the Pushpalatha2012 constant
#            during computations

mae(sim=sim, obs=obs, fun=log, epsilon.type="Pushpalatha2012")

# Verifying the previous value, with the epsilon value following Pushpalatha2012
eps  <- mean(obs, na.rm=TRUE)/100
lsim <- log(sim+eps)
lobs <- log(obs+eps)
mae(sim=lsim, obs=lobs)

##################
# Example 6: mae for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and adding a user-defined constant
#            during computations

eps <- 0.01
mae(sim=sim, obs=obs, fun=log, epsilon.type="otherValue", epsilon.value=eps)

# Verifying the previous value:
lsim <- log(sim+eps)
lobs <- log(obs+eps)
mae(sim=lsim, obs=lobs)

##################
# Example 7: mae for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and using a user-defined factor
#            to multiply the mean of the observed values to obtain the constant
#            to be added to 'sim' and 'obs' during computations

fact <- 1/50
mae(sim=sim, obs=obs, fun=log, epsilon.type="otherFactor", epsilon.value=fact)

# Verifying the previous value:
eps  <- fact*mean(obs, na.rm=TRUE)
lsim <- log(sim+eps)
lobs <- log(obs+eps)
mae(sim=lsim, obs=lobs)

##################
# Example 8: mae for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying a
#            user-defined function to 'sim' and 'obs' during computations

fun1 <- function(x) {sqrt(x+1)}

mae(sim=sim, obs=obs, fun=fun1)
```

```
# Verifying the previous value, with the epsilon value following Pushpalatha2012
sim1 <- sqrt(sim+1)
obs1 <- sqrt(obs+1)
mae(sim=sim1, obs=obs1)
```

md *Modified Index of Agreement*

Description

This function computes the modified Index of Agreement between `sim` and `obs`, with treatment of missing values.

Usage

```
md(sim, obs, ...)

## Default S3 method:
md(sim, obs, j=1, na.rm=TRUE, fun=NULL, ...,
        epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
           epsilon.value=NA)

## S3 method for class 'data.frame'
md(sim, obs, j=1, na.rm=TRUE, fun=NULL, ...,
        epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
           epsilon.value=NA)

## S3 method for class 'matrix'
md(sim, obs, j=1, na.rm=TRUE, fun=NULL, ...,
        epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
           epsilon.value=NA)

## S3 method for class 'zoo'
md(sim, obs, j=1, na.rm=TRUE, fun=NULL, ...,
        epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
           epsilon.value=NA)
```

Arguments

`sim`	numeric, zoo, matrix or data.frame with simulated values
`obs`	numeric, zoo, matrix or data.frame with observed values
`j`	numeric, with the exponent to be used in the computation of the modified index of agreement. The default value is j=1.
`na.rm`	a logical value indicating whether 'NA' should be stripped before the computation proceeds. When an 'NA' value is found at the i-th position in `obs` **OR** `sim`, the i-th value of `obs` **AND** `sim` are removed before the computation.

fun	function to be applied to sim and obs in order to obtain transformed values thereof before computing the modified index of agreement. The first argument MUST BE a numeric vector with any name (e.g., x), and additional arguments are passed using ....
...	arguments passed to fun, in addition to the mandatory first numeric vector.
epsilon.type	argument used to define a numeric value to be added to both sim and obs before applying fun. It is was designed to allow the use of logarithm and other similar functions that do not work with zero values. Valid values of epsilon.type are: 1) "none": sim and obs are used by fun without the addition of any nummeric value. 2) "Pushpalatha2012": one hundredth (1/100) of the mean observed values is added to both sim and obs before applying fun, as described in Pushpalatha et al. (2012). 3) "otherFactor": the numeric value defined in the epsilon.value argument is used to multiply the the mean observed values, instead of the one hundredth (1/100) described in Pushpalatha et al. (2012). The resulting value is then added to both sim and obs, before applying fun. 4) "otherValue": the numeric value defined in the epsilon.value argument is directly added to both sim and obs, before applying fun.
epsilon.value	numeric value to be added to both sim and obs when epsilon.type="otherValue".

Details

$$md = 1 - \frac{\sum_{i=1}^{N} |O_i - S_i|^j}{\sum_{i=1}^{N} \left|S_i - \bar{O}\right| + \left|O_i - \bar{O}\right|^j}$$

The Index of Agreement (d) developed by Willmott (1981) as a standardized measure of the degree of model prediction error and varies between 0 and 1.
A value of 1 indicates a perfect match, and 0 indicates no agreement at all (Willmott, 1981).

The index of agreement can detect additive and proportional differences in the observed and simulated means and variances; however, it is overly sensitive to extreme values due to the squared differences (Legates and McCabe, 1999).

Value

Modified index of agreement between sim and obs.

If sim and obs are matrixes, the returned value is a vector, with the modified index of agreement between each column of sim and obs.

Note

obs and sim has to have the same length/dimension

The missing values in obs and sim are removed before the computation proceeds, and only those positions with non-missing values in obs and sim are considered in the computation

Author(s)

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

References

Krause, P.; Boyle, D.P.; Base, F. (2005). Comparison of different efficiency criteria for hydrological model assessment, Advances in Geosciences, 5, 89-97. doi:10.5194/adgeo-5-89-2005.

Willmott, C.J. (1981). On the validation of models. Physical Geography, 2, 184–194. doi:10.1080/02723646.1981.10642213.

Willmott, C.J. (1984). On the evaluation of model performance in physical geography. Spatial Statistics and Models, G. L. Gaile and C. J. Willmott, eds., 443-460. doi:10.1007/978-94-017-3048-8_23.

Willmott, C.J.; Ackleson, S.G. Davis, R.E.; Feddema, J.J.; Klink, K.M.; Legates, D.R.; O'Donnell, J.; Rowe, C.M. (1985), Statistics for the Evaluation and Comparison of Models, J. Geophys. Res., 90(C5), 8995-9005. doi:10.1029/JC090iC05p08995.

Legates, D.R.; McCabe, G. J. Jr. (1999), Evaluating the Use of "Goodness-of-Fit" Measures in Hydrologic and Hydroclimatic Model Validation, Water Resour. Res., 35(1), 233-241. doi:10.1029/1998WR900018.

See Also

d, dr, rd, gof, ggof

Examples

```
obs <- 1:10
sim <- 1:10
md(sim, obs)

obs <- 1:10
sim <- 2:11
md(sim, obs)

##################
# Loading daily streamflows of the Ega River (Spain), from 1961 to 1970
data(EgaEnEstellaQts)
obs <- EgaEnEstellaQts

# Generating a simulated daily time series, initially equal to the observed series
sim <- obs

# Computing the modified index of agreement for the "best" (unattainable) case
md(sim=sim, obs=obs)

# Randomly changing the first 2000 elements of 'sim', by using a normal distribution
# with mean 10 and standard deviation equal to 1 (default of 'rnorm').
sim[1:2000] <- obs[1:2000] + rnorm(2000, mean=10)

# Computing the new 'd1'
md(sim=sim, obs=obs)
```

me *Mean Error*

Description

Mean error between `sim` and `obs`, in the same units of them, with treatment of missing values.

Usage

```
me(sim, obs, ...)

## Default S3 method:
me(sim, obs, na.rm=TRUE, fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
            epsilon.value=NA)

## S3 method for class 'data.frame'
me(sim, obs, na.rm=TRUE, fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
            epsilon.value=NA)

## S3 method for class 'matrix'
me(sim, obs, na.rm=TRUE, fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
            epsilon.value=NA)

## S3 method for class 'zoo'
me(sim, obs, na.rm=TRUE, fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
            epsilon.value=NA)
```

Arguments

`sim`	numeric, zoo, matrix or data.frame with simulated values
`obs`	numeric, zoo, matrix or data.frame with observed values
`na.rm`	a logical value indicating whether 'NA' should be stripped before the computation proceeds. When an 'NA' value is found at the i-th position in `obs` **OR** `sim`, the i-th value of `obs` **AND** `sim` are removed before the computation.
`fun`	function to be applied to `sim` and `obs` in order to obtain transformed values thereof before computing this goodness-of-fit index. The first argument MUST BE a numeric vector with any name (e.g., `x`), and additional arguments are passed using `...`.
`...`	arguments passed to `fun`, in addition to the mandatory first numeric vector.

epsilon.type — argument used to define a numeric value to be added to both sim and obs before applying fun.

It is was designed to allow the use of logarithm and other similar functions that do not work with zero values.

Valid values of epsilon.type are:

1) "none": sim and obs are used by fun without the addition of any numeric value. This is the default option.

2) "Pushpalatha2012": one hundredth (1/100) of the mean observed values is added to both sim and obs before applying fun, as described in Pushpalatha et al. (2012).

3) "otherFactor": the numeric value defined in the epsilon.value argument is used to multiply the the mean observed values, instead of the one hundredth (1/100) described in Pushpalatha et al. (2012). The resulting value is then added to both sim and obs, before applying fun.

4) "otherValue": the numeric value defined in the epsilon.value argument is directly added to both sim and obs, before applying fun.

epsilon.value — -) when epsilon.type="otherValue" it represents the numeric value to be added to both sim and obs before applying fun.
-) when epsilon.type="otherFactor" it represents the numeric factor used to multiply the mean of the observed values, instead of the one hundredth (1/100) described in Pushpalatha et al. (2012). The resulting value is then added to both sim and obs before applying fun.

Details

$$me = \frac{1}{N} \sum_{i=1}^{N} (S_i - O_i))$$

Value

Mean error between sim and obs.

If sim and obs are matrixes, the returned value is a vector, with the mean error between each column of sim and obs.

Note

obs and sim has to have the same length/dimension

The missing values in obs and sim are removed before the computation proceeds, and only those positions with non-missing values in obs and sim are considered in the computation

Author(s)

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

References

Hill, T.; Lewicki, P.; Lewicki, P. (2006). Statistics: methods and applications: a comprehensive reference for science, industry, and data mining. StatSoft, Inc.

See Also

mae, gof, ggof

Examples

```
##################
# Example 1: basic ideal case
obs <- 1:10
sim <- 1:10
me(sim, obs)

obs <- 1:10
sim <- 2:11
me(sim, obs)

##################
# Example 2:
# Loading daily streamflows of the Ega River (Spain), from 1961 to 1970
data(EgaEnEstellaQts)
obs <- EgaEnEstellaQts

# Generating a simulated daily time series, initially equal to the observed series
sim <- obs

# Computing the 'me' for the "best" (unattainable) case
me(sim=sim, obs=obs)

##################
# Example 3: me for simulated values equal to observations plus random noise
#            on the first half of the observed values.
#            This random noise has more relative importance for ow flows than
#            for medium and high flows.

# Randomly changing the first 1826 elements of 'sim', by using a normal distribution
# with mean 10 and standard deviation equal to 1 (default of 'rnorm').
sim[1:1826] <- obs[1:1826] + rnorm(1826, mean=10)
ggof(sim, obs)

me(sim=sim, obs=obs)

##################
# Example 4: me for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' during computations.

me(sim=sim, obs=obs, fun=log)
```

```
# Verifying the previous value:
lsim <- log(sim)
lobs <- log(obs)
me(sim=lsim, obs=lobs)

##################
# Example 5: me for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and adding the Pushpalatha2012 constant
#            during computations

me(sim=sim, obs=obs, fun=log, epsilon.type="Pushpalatha2012")

# Verifying the previous value, with the epsilon value following Pushpalatha2012
eps  <- mean(obs, na.rm=TRUE)/100
lsim <- log(sim+eps)
lobs <- log(obs+eps)
me(sim=lsim, obs=lobs)

##################
# Example 6: me for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and adding a user-defined constant
#            during computations

eps <- 0.01
me(sim=sim, obs=obs, fun=log, epsilon.type="otherValue", epsilon.value=eps)

# Verifying the previous value:
lsim <- log(sim+eps)
lobs <- log(obs+eps)
me(sim=lsim, obs=lobs)

##################
# Example 7: me for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and using a user-defined factor
#            to multiply the mean of the observed values to obtain the constant
#            to be added to 'sim' and 'obs' during computations

fact <- 1/50
me(sim=sim, obs=obs, fun=log, epsilon.type="otherFactor", epsilon.value=fact)

# Verifying the previous value:
eps  <- fact*mean(obs, na.rm=TRUE)
lsim <- log(sim+eps)
lobs <- log(obs+eps)
me(sim=lsim, obs=lobs)

##################
# Example 8: me for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying a
#            user-defined function to 'sim' and 'obs' during computations
```

```
fun1 <- function(x) {sqrt(x+1)}

me(sim=sim, obs=obs, fun=fun1)

# Verifying the previous value, with the epsilon value following Pushpalatha2012
sim1 <- sqrt(sim+1)
obs1 <- sqrt(obs+1)
me(sim=sim1, obs=obs1)
```

---

mNSE *Modified Nash-Sutcliffe efficiency*

Description

Modified Nash-Sutcliffe efficiency between sim and obs, with treatment of missing values.

Usage

```
mNSE(sim, obs, ...)

## Default S3 method:
mNSE(sim, obs, j=1, na.rm=TRUE, fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
              epsilon.value=NA)

## S3 method for class 'data.frame'
mNSE(sim, obs, j=1, na.rm=TRUE, fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
              epsilon.value=NA)

## S3 method for class 'matrix'
mNSE(sim, obs, j=1, na.rm=TRUE, fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
              epsilon.value=NA)

## S3 method for class 'zoo'
mNSE(sim, obs, j=1, na.rm=TRUE, fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
              epsilon.value=NA)
```

Arguments

sim	numeric, zoo, matrix or data.frame with simulated values
obs	numeric, zoo, matrix or data.frame with observed values
j	numeric, with the exponent to be used in the computation of the modified Nash-Sutcliffe efficiency. The default value is j=1.

`na.rm` a logical value indicating whether 'NA' should be stripped before the computation proceeds.
When an 'NA' value is found at the i-th position in `obs` **OR** `sim`, the i-th value of `obs` **AND** `sim` are removed before the computation.

`fun` function to be applied to `sim` and `obs` in order to obtain transformed values thereof before computing the Nash-Sutcliffe efficiency.
The first argument MUST BE a numeric vector with any name (e.g., `x`), and additional arguments are passed using `...`.

`...` arguments passed to `fun`, in addition to the mandatory first numeric vector.

`epsilon.type` argument used to define a numeric value to be added to both `sim` and `obs` before applying `FUN`.
It is was designed to allow the use of logarithm and other similar functions that do not work with zero values.
Valid values of `epsilon.type` are:
1) `"none"`: `sim` and `obs` are used by `FUN` without the addition of any nummeric value.
2) `"Pushpalatha2012"`: one hundredth (1/100) of the mean observed values is added to both `sim` and `obs` before applying `FUN`, as described in Pushpalatha et al. (2012).
3) `"otherFactor"`: the numeric value defined in the `epsilon.value` argument is used to multiply the the mean observed values, instead of the one hundredth (1/100) described in Pushpalatha et al. (2012). The resulting value is then added to both `sim` and `obs`, before applying `FUN`.
4) `"otherValue"`: the numeric value defined in the `epsilon.value` argument is directly added to both `sim` and `obs`, before applying `FUN`.

`epsilon.value` -) when `epsilon.type="otherValue"` it represents the numeric value to be added to both `sim` and `obs` before applying `fun`.
-) when `epsilon.type="otherFactor"` it represents the numeric factor used to multiply the mean of the observed values, instead of the one hundredth (1/100) described in Pushpalatha et al. (2012). The resulting value is then added to both `sim` and `obs` before applying `fun`.

Details

$$mNSE = 1 - \frac{\sum_{i=1}^{N} |S_i - O_i|^j}{\sum_{i=1}^{N} |O_i - \bar{O}|^j}$$

When `j=1`, the modified NSeff is not inflated by the squared values of the differences, because the squares are replaced by absolute values.

Value

Modified Nash-Sutcliffe efficiency between `sim` and `obs`.

If `sim` and `obs` are matrixes, the returned value is a vector, with the modified Nash-Sutcliffe efficiency between each column of `sim` and `obs`.

Note

obs and sim has to have the same length/dimension

The missing values in obs and sim are removed before the computation proceeds, and only those positions with non-missing values in obs and sim are considered in the computation

Author(s)

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

References

Krause, P.; Boyle, D.P.; Base, F. (2005). Comparison of different efficiency criteria for hydrological model assessment, Advances in Geosciences, 5, 89-97. doi:10.5194/adgeo-5-89-2005.

Legates, D.R.; McCabe, G. J. Jr. (1999), Evaluating the Use of "Goodness-of-Fit" Measures in Hydrologic and Hydroclimatic Model Validation, Water Resour. Res., 35(1), 233-241. doi:10.1029/1998WR900018.

See Also

NSE, rNSE, wNSE, KGE, gof, ggof

Examples

```
##################
# Example 1: basic ideal case
obs <- 1:10
sim <- 1:10
mNSE(sim, obs)

obs <- 1:10
sim <- 2:11
mNSE(sim, obs)

##################
# Example 2:
# Loading daily streamflows of the Ega River (Spain), from 1961 to 1970
data(EgaEnEstellaQts)
obs <- EgaEnEstellaQts

# Generating a simulated daily time series, initially equal to the observed series
sim <- obs

# Computing the 'mNSE' for the "best" (unattainable) case
mNSE(sim=sim, obs=obs)

##################
# Example 3: mNSE for simulated values equal to observations plus random noise
#            on the first half of the observed values.
```

```
#            This random noise has more relative importance for ow flows than
#            for medium and high flows.

# Randomly changing the first 1826 elements of 'sim', by using a normal distribution
# with mean 10 and standard deviation equal to 1 (default of 'rnorm').
sim[1:1826] <- obs[1:1826] + rnorm(1826, mean=10)
ggof(sim, obs)

mNSE(sim=sim, obs=obs)

##################
# Example 4: mNSE for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' during computations.

mNSE(sim=sim, obs=obs, fun=log)

# Verifying the previous value:
lsim <- log(sim)
lobs <- log(obs)
mNSE(sim=lsim, obs=lobs)

##################
# Example 5: mNSE for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and adding the Pushpalatha2012 constant
#            during computations

mNSE(sim=sim, obs=obs, fun=log, epsilon.type="Pushpalatha2012")

# Verifying the previous value, with the epsilon value following Pushpalatha2012
eps  <- mean(obs, na.rm=TRUE)/100
lsim <- log(sim+eps)
lobs <- log(obs+eps)
mNSE(sim=lsim, obs=lobs)

##################
# Example 6: mNSE for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and adding a user-defined constant
#            during computations

eps <- 0.01
mNSE(sim=sim, obs=obs, fun=log, epsilon.type="otherValue", epsilon.value=eps)

# Verifying the previous value:
lsim <- log(sim+eps)
lobs <- log(obs+eps)
mNSE(sim=lsim, obs=lobs)

##################
# Example 7: mNSE for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
```

```
#            logarithm to 'sim' and 'obs' and using a user-defined factor
#            to multiply the mean of the observed values to obtain the constant
#            to be added to 'sim' and 'obs' during computations

fact <- 1/50
mNSE(sim=sim, obs=obs, fun=log, epsilon.type="otherFactor", epsilon.value=fact)

# Verifying the previous value:
eps  <- fact*mean(obs, na.rm=TRUE)
lsim <- log(sim+eps)
lobs <- log(obs+eps)
mNSE(sim=lsim, obs=lobs)

##################
# Example 8: mNSE for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying a
#            user-defined function to 'sim' and 'obs' during computations

fun1 <- function(x) {sqrt(x+1)}

mNSE(sim=sim, obs=obs, fun=fun1)

# Verifying the previous value, with the epsilon value following Pushpalatha2012
sim1 <- sqrt(sim+1)
obs1 <- sqrt(obs+1)
mNSE(sim=sim1, obs=obs1)
```

---

mse *Mean Squared Error*

Description

Mean squared error between sim and obs, in the squared units of sim and obs, with treatment of missing values.

Usage

```
mse(sim, obs, ...)

## Default S3 method:
mse(sim, obs, na.rm=TRUE, fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
             epsilon.value=NA)

## S3 method for class 'data.frame'
mse(sim, obs, na.rm=TRUE, fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
             epsilon.value=NA)
```

```
## S3 method for class 'matrix'
mse(sim, obs, na.rm=TRUE, fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
             epsilon.value=NA)

## S3 method for class 'zoo'
mse(sim, obs, na.rm=TRUE, fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
             epsilon.value=NA)
```

Arguments

`sim` numeric, zoo, matrix or data.frame with simulated values

`obs` numeric, zoo, matrix or data.frame with observed values

`na.rm` a logical value indicating whether 'NA' should be stripped before the computation proceeds.
When an 'NA' value is found at the i-th position in `obs` **OR** `sim`, the i-th value of `obs` **AND** `sim` are removed before the computation.

`fun` function to be applied to `sim` and `obs` in order to obtain transformed values thereof before computing this goodness-of-fit index.

The first argument MUST BE a numeric vector with any name (e.g., `x`), and additional arguments are passed using `...`.

`...` arguments passed to `fun`, in addition to the mandatory first numeric vector.

`epsilon.type` argument used to define a numeric value to be added to both `sim` and `obs` before applying `fun`.

It is was designed to allow the use of logarithm and other similar functions that do not work with zero values.

Valid values of `epsilon.type` are:

1) `"none"`: `sim` and `obs` are used by `fun` without the addition of any numeric value. This is the default option.

2) `"Pushpalatha2012"`: one hundredth (1/100) of the mean observed values is added to both `sim` and `obs` before applying `fun`, as described in Pushpalatha et al. (2012).

3) `"otherFactor"`: the numeric value defined in the `epsilon.value` argument is used to multiply the the mean observed values, instead of the one hundredth (1/100) described in Pushpalatha et al. (2012). The resulting value is then added to both `sim` and `obs`, before applying `fun`.

4) `"otherValue"`: the numeric value defined in the `epsilon.value` argument is directly added to both `sim` and `obs`, before applying `fun`.

`epsilon.value` -) when `epsilon.type="otherValue"` it represents the numeric value to be added to both `sim` and `obs` before applying `fun`.
-) when `epsilon.type="otherFactor"` it represents the numeric factor used to multiply the mean of the observed values, instead of the one hundredth (1/100) described in Pushpalatha et al. (2012). The resulting value is then added to both `sim` and `obs` before applying `fun`.

Details

$$mse = \frac{1}{N} \sum_{i=1}^{N} (S_i - O_i)^2$$

Value

Mean squared error between `sim` and `obs`.

If `sim` and `obs` are matrixes, the returned value is a vector, with the mean squared error between each column of `sim` and `obs`.

Note

`obs` and `sim` has to have the same length/dimension

The missing values in `obs` and `sim` are removed before the computation proceeds, and only those positions with non-missing values in `obs` and `sim` are considered in the computation

Author(s)

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

References

Yapo P.O.; Gupta H.V.; Sorooshian S. (1996). Automatic calibration of conceptual rainfall-runoff models: sensitivity to calibration data. Journal of Hydrology. v181 i1-4. 23-48. doi:10.1016/0022-1694(95)02918-4

Gupta, H.V.; Kling, H. (2011). On typical range, sensitivity, and normalization of Mean Squared Error and Nash-Sutcliffe Efficiency type metrics. Water Resources Research, 47(10). doi:10.1029/2011WR010962.

Willmott, C.J.; Matsuura, K.; Robeson, S.M. (2009). Ambiguities inherent in sums-of-squares-based error statistics, Atmospheric Environment, 43, 749-752, doi:10.1016/j.atmosenv.2008.10.005.

See Also

`pbias`, `pbiasfdc`, `mae`, `rmse`, `ubRMSE`, `nrmse`, `ssq`, `gof`, `ggof`

Examples

```
##################
# Example 1: basic ideal case
obs <- 1:10
sim <- 1:10
mse(sim, obs)

obs <- 1:10
sim <- 2:11
```

```
mse(sim, obs)

##################
# Example 2:
# Loading daily streamflows of the Ega River (Spain), from 1961 to 1970
data(EgaEnEstellaQts)
obs <- EgaEnEstellaQts

# Generating a simulated daily time series, initially equal to the observed series
sim <- obs

# Computing the 'mse' for the "best" (unattainable) case
mse(sim=sim, obs=obs)

##################
# Example 3: mse for simulated values equal to observations plus random noise
#            on the first half of the observed values.
#            This random noise has more relative importance for ow flows than
#            for medium and high flows.

# Randomly changing the first 1826 elements of 'sim', by using a normal distribution
# with mean 10 and standard deviation equal to 1 (default of 'rnorm').
sim[1:1826] <- obs[1:1826] + rnorm(1826, mean=10)
ggof(sim, obs)

mse(sim=sim, obs=obs)

##################
# Example 4: mse for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' during computations.

mse(sim=sim, obs=obs, fun=log)

# Verifying the previous value:
lsim <- log(sim)
lobs <- log(obs)
mse(sim=lsim, obs=lobs)

##################
# Example 5: mse for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and adding the Pushpalatha2012 constant
#            during computations

mse(sim=sim, obs=obs, fun=log, epsilon.type="Pushpalatha2012")

# Verifying the previous value, with the epsilon value following Pushpalatha2012
eps  <- mean(obs, na.rm=TRUE)/100
lsim <- log(sim+eps)
lobs <- log(obs+eps)
mse(sim=lsim, obs=lobs)
```

```
##################
# Example 6: mse for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and adding a user-defined constant
#            during computations

eps <- 0.01
mse(sim=sim, obs=obs, fun=log, epsilon.type="otherValue", epsilon.value=eps)

# Verifying the previous value:
lsim <- log(sim+eps)
lobs <- log(obs+eps)
mse(sim=lsim, obs=lobs)

##################
# Example 7: mse for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and using a user-defined factor
#            to multiply the mean of the observed values to obtain the constant
#            to be added to 'sim' and 'obs' during computations

fact <- 1/50
mse(sim=sim, obs=obs, fun=log, epsilon.type="otherFactor", epsilon.value=fact)

# Verifying the previous value:
eps  <- fact*mean(obs, na.rm=TRUE)
lsim <- log(sim+eps)
lobs <- log(obs+eps)
mse(sim=lsim, obs=lobs)

##################
# Example 8: mse for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying a
#            user-defined function to 'sim' and 'obs' during computations

fun1 <- function(x) {sqrt(x+1)}

mse(sim=sim, obs=obs, fun=fun1)

# Verifying the previous value, with the epsilon value following Pushpalatha2012
sim1 <- sqrt(sim+1)
obs1 <- sqrt(obs+1)
mse(sim=sim1, obs=obs1)
```

`nrmse` *Normalized Root Mean Square Error*

Description

Normalized root mean square error (NRMSE) between `sim` and `obs`, with treatment of missing values.

Usage

```
nrmse(sim, obs, ...)

## Default S3 method:
nrmse(sim, obs, na.rm=TRUE, norm="sd", fun=NULL, ...,
          epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
               epsilon.value=NA)

## S3 method for class 'data.frame'
nrmse(sim, obs, na.rm=TRUE, norm="sd", fun=NULL, ...,
          epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
               epsilon.value=NA)

## S3 method for class 'matrix'
nrmse(sim, obs, na.rm=TRUE, norm="sd", fun=NULL, ...,
          epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
               epsilon.value=NA)

## S3 method for class 'zoo'
nrmse(sim, obs, na.rm=TRUE, norm="sd", fun=NULL, ...,
          epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
               epsilon.value=NA)
```

Arguments

`sim`	numeric, zoo, matrix or data.frame with simulated values
`obs`	numeric, zoo, matrix or data.frame with observed values
`na.rm`	a logical value indicating whether 'NA' should be stripped before the computation proceeds. When an 'NA' value is found at the i-th position in `obs` **OR** `sim`, the i-th value of `obs` **AND** `sim` are removed before the computation.
`norm`	character, indicating the value to be used for normalising the root mean square error (RMSE). Valid values are: -) `sd` : standard deviation of observations (default). -) `maxmin`: difference between the maximum and minimum observed values
`fun`	function to be applied to `sim` and `obs` in order to obtain transformed values thereof before computing this goodness-of-fit index. The first argument MUST BE a numeric vector with any name (e.g., `x`), and additional arguments are passed using `...`.
`...`	arguments passed to `fun`, in addition to the mandatory first numeric vector.
`epsilon.type`	argument used to define a numeric value to be added to both `sim` and `obs` before applying `fun`. It is was designed to allow the use of logarithm and other similar functions that do not work with zero values. Valid values of `epsilon.type` are:

1) "none": sim and obs are used by fun without the addition of any numeric value. This is the default option.

2) "Pushpalatha2012": one hundredth (1/100) of the mean observed values is added to both sim and obs before applying fun, as described in Pushpalatha et al. (2012).

3) "otherFactor": the numeric value defined in the epsilon.value argument is used to multiply the the mean observed values, instead of the one hundredth (1/100) described in Pushpalatha et al. (2012). The resulting value is then added to both sim and obs, before applying fun.

4) "otherValue": the numeric value defined in the epsilon.value argument is directly added to both sim and obs, before applying fun.

epsilon.value -) when epsilon.type="otherValue" it represents the numeric value to be added to both sim and obs before applying fun.
-) when epsilon.type="otherFactor" it represents the numeric factor used to multiply the mean of the observed values, instead of the one hundredth (1/100) described in Pushpalatha et al. (2012). The resulting value is then added to both sim and obs before applying fun.

Details

$$nrmse = 100 \frac{\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(S_i - O_i\right)^2}}{nval}$$

$$nval = \left\{ \begin{array}{cl} sd(O_i) & , \text{norm="sd"} \\ O_{max} - O_{min} & , \text{norm="maxmin"} \end{array} \right.$$

Value

Normalized root mean square error (nrmse) between sim and obs. The result is given in percentage (%)

If sim and obs are matrixes, the returned value is a vector, with the normalized root mean square error between each column of sim and obs.

Note

obs and sim have to have the same length/dimension

Missing values in obs and sim are removed before the computation proceeds, and only those positions with non-missing values in obs and sim are considered in the computation

Author(s)

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

See Also

pbias, pbiasfdc, mae, mse, rmse, ubRMSE, ssq, gof, ggof

Examples

```
##################
# Example 1: basic ideal case
obs <- 1:10
sim <- 1:10
nrmse(sim, obs)

obs <- 1:10
sim <- 2:11
nrmse(sim, obs)

##################
# Example 2:
# Loading daily streamflows of the Ega River (Spain), from 1961 to 1970
data(EgaEnEstellaQts)
obs <- EgaEnEstellaQts

# Generating a simulated daily time series, initially equal to the observed series
sim <- obs

# Computing the 'nrmse' for the "best" (unattainable) case
nrmse(sim=sim, obs=obs)

##################
# Example 3: nrmse for simulated values equal to observations plus random noise
#            on the first half of the observed values.
#            This random noise has more relative importance for ow flows than
#            for medium and high flows.

# Randomly changing the first 1826 elements of 'sim', by using a normal distribution
# with mean 10 and standard deviation equal to 1 (default of 'rnorm').
sim[1:1826] <- obs[1:1826] + rnorm(1826, mean=10)
ggof(sim, obs)

nrmse(sim=sim, obs=obs)

##################
# Example 4: nrmse for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' during computations.

nrmse(sim=sim, obs=obs, fun=log)

# Verifying the previous value:
lsim <- log(sim)
lobs <- log(obs)
nrmse(sim=lsim, obs=lobs)

##################
# Example 5: nrmse for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and adding the Pushpalatha2012 constant
```

```
#            during computations

nrmse(sim=sim, obs=obs, fun=log, epsilon.type="Pushpalatha2012")

# Verifying the previous value, with the epsilon value following Pushpalatha2012
eps  <- mean(obs, na.rm=TRUE)/100
lsim <- log(sim+eps)
lobs <- log(obs+eps)
nrmse(sim=lsim, obs=lobs)

##################
# Example 6: nrmse for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and adding a user-defined constant
#            during computations

eps <- 0.01
nrmse(sim=sim, obs=obs, fun=log, epsilon.type="otherValue", epsilon.value=eps)

# Verifying the previous value:
lsim <- log(sim+eps)
lobs <- log(obs+eps)
nrmse(sim=lsim, obs=lobs)

##################
# Example 7: nrmse for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and using a user-defined factor
#            to multiply the mean of the observed values to obtain the constant
#            to be added to 'sim' and 'obs' during computations

fact <- 1/50
nrmse(sim=sim, obs=obs, fun=log, epsilon.type="otherFactor", epsilon.value=fact)

# Verifying the previous value:
eps  <- fact*mean(obs, na.rm=TRUE)
lsim <- log(sim+eps)
lobs <- log(obs+eps)
nrmse(sim=lsim, obs=lobs)

##################
# Example 8: nrmse for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying a
#            user-defined function to 'sim' and 'obs' during computations

fun1 <- function(x) {sqrt(x+1)}

nrmse(sim=sim, obs=obs, fun=fun1)

# Verifying the previous value, with the epsilon value following Pushpalatha2012
sim1 <- sqrt(sim+1)
obs1 <- sqrt(obs+1)
nrmse(sim=sim1, obs=obs1)
```

NSE *Nash-Sutcliffe Efficiency*

Description

Nash-Sutcliffe efficiency between `sim` and `obs`, with treatment of missing values.

Usage

```
NSE(sim, obs, ...)

## Default S3 method:
NSE(sim, obs, na.rm=TRUE, fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
             epsilon.value=NA)

## S3 method for class 'data.frame'
NSE(sim, obs, na.rm=TRUE, fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
             epsilon.value=NA)

## S3 method for class 'matrix'
NSE(sim, obs, na.rm=TRUE, fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
             epsilon.value=NA)

## S3 method for class 'zoo'
NSE(sim, obs, na.rm=TRUE, fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
            epsilon.value=NA)
```

Arguments

`sim` numeric, zoo, matrix or data.frame with simulated values

`obs` numeric, zoo, matrix or data.frame with observed values

`na.rm` a logical value indicating whether 'NA' should be stripped before the computation proceeds.
When an 'NA' value is found at the i-th position in `obs` **OR** `sim`, the i-th value of `obs` **AND** `sim` are removed before the computation.

`fun` function to be applied to `sim` and `obs` in order to obtain transformed values thereof before computing the Nash-Sutcliffe efficiency.

The first argument MUST BE a numeric vector with any name (e.g., `x`), and additional arguments are passed using `...`.

`...` arguments passed to `fun`, in addition to the mandatory first numeric vector.

epsilon.type argument used to define a numeric value to be added to both sim and obs before applying FUN.

It is was designed to allow the use of logarithm and other similar functions that do not work with zero values.

Valid values of epsilon.type are:

1) "none": sim and obs are used by fun without the addition of any numeric value. This is the default option.

2) "Pushpalatha2012": one hundredth (1/100) of the mean observed values is added to both sim and obs before applying fun, as described in Pushpalatha et al. (2012).

3) "otherFactor": the numeric value defined in the epsilon.value argument is used to multiply the mean of the observed values, instead of the one hundredth (1/100) described in Pushpalatha et al. (2012). The resulting value is then added to both sim and obs, before applying fun.

4) "otherValue": the numeric value defined in the epsilon.value argument is directly added to both sim and obs, before applying fun.

epsilon.value -) when epsilon.type="otherValue" it represents the numeric value to be added to both sim and obs before applying fun.
-) when epsilon.type="otherFactor" it represents the numeric factor used to multiply the mean of the observed values, instead of the one hundredth (1/100) described in Pushpalatha et al. (2012). The resulting value is then added to both sim and obs before applying fun.

Details

$$NSE = 1 - \frac{\sum_{i=1}^{N} (S_i - O_i)^2}{\sum_{i=1}^{N} \left(O_i - \bar{O}\right)^2}$$

The Nash-Sutcliffe efficiency (NSE) is a normalized statistic that determines the relative magnitude of the residual variance ("noise") compared to the measured data variance ("information") (Nash and Sutcliffe, 1970).

NSE indicates how well the plot of observed versus simulated data fits the 1:1 line.

Nash-Sutcliffe efficiencies range from -Inf to 1. Essentially, the closer to 1, the more accurate the model is.
-) NSE = 1, corresponds to a perfect match of modelled to the observed data.
-) NSE = 0, indicates that the model predictions are as accurate as the mean of the observed data,
-) -Inf < NSE < 0, indicates that the observed mean is better predictor than the model.

Value

Nash-Sutcliffe efficiency between sim and obs.

If sim and obs are matrixes, the returned value is a vector, with the Nash-Sutcliffe efficiency between each column of sim and obs.

Note

obs and sim has to have the same length/dimension

The missing values in obs and sim are removed before the computation proceeds, and only those positions with non-missing values in obs and sim are considered in the computation

Author(s)

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

References

https://en.wikipedia.org/wiki/Nash%E2%80%93Sutcliffe_model_efficiency_coefficient

Nash, J.E. and Sutcliffe, J.V. (1970). River flow forecasting through conceptual models. Part 1: a discussion of principles, Journal of Hydrology 10, pp. 282-290. doi:10.1016/0022-1694(70)90255-6.

Garrick, M.; Cunnane, C.; Nash, J.E. (1978). A criterion of efficiency for rainfall-runoff models. Journal of Hydrology 36, 375-381. doi:10.1016/0022-1694(78)90155-5.

Schaefli, B., Gupta, H. (2007). Do Nash values have value?. Hydrological Processes 21, 2075-2080. doi:10.1002/hyp.6825.

Criss, R. E.; Winston, W. E. (2008), Do Nash values have value? Discussion and alternate proposals. Hydrological Processes, 22: 2723-2725. doi:10.1002/hyp.7072.

Gupta, H.V.; Kling, H. (2011). On typical range, sensitivity, and normalization of Mean Squared Error and Nash-Sutcliffe Efficiency type metrics. Water Resources Research, 47(10). doi:10.1029/2011WR010962.

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

Knoben, W. J.; Freer, J. E.; Woods, R. A. (2019). Inherent benchmark or not? Comparing Nash-Sutcliffe and Kling-Gupta efficiency scores. Hydrology and Earth System Sciences, 23(10), 4323-4331. doi:10.5194/hess-23-4323-2019.

See Also

mNSE, rNSE, wNSE, KGE, gof, ggof

Examples

```
##################
# Example 1: basic ideal case
obs <- 1:10
sim <- 1:10
NSE(sim, obs)

obs <- 1:10
sim <- 2:11
NSE(sim, obs)

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

```
# Example 2:
# Loading daily streamflows of the Ega River (Spain), from 1961 to 1970
data(EgaEnEstellaQts)
obs <- EgaEnEstellaQts

# Generating a simulated daily time series, initially equal to the observed series
sim <- obs

# Computing the 'NSE' for the "best" (unattainable) case
NSE(sim=sim, obs=obs)

##################
# Example 3: NSE for simulated values equal to observations plus random noise
#            on the first half of the observed values.
#            This random noise has more relative importance for ow flows than
#            for medium and high flows.

# Randomly changing the first 1826 elements of 'sim', by using a normal distribution
# with mean 10 and standard deviation equal to 1 (default of 'rnorm').
sim[1:1826] <- obs[1:1826] + rnorm(1826, mean=10)
ggof(sim, obs)

NSE(sim=sim, obs=obs)

##################
# Example 4: NSE for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' during computations.

NSE(sim=sim, obs=obs, fun=log)

# Verifying the previous value:
lsim <- log(sim)
lobs <- log(obs)
NSE(sim=lsim, obs=lobs)

##################
# Example 5: NSE for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and adding the Pushpalatha2012 constant
#            during computations

NSE(sim=sim, obs=obs, fun=log, epsilon.type="Pushpalatha2012")

# Verifying the previous value, with the epsilon value following Pushpalatha2012
eps  <- mean(obs, na.rm=TRUE)/100
lsim <- log(sim+eps)
lobs <- log(obs+eps)
NSE(sim=lsim, obs=lobs)

##################
# Example 6: NSE for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
```

```
#            logarithm to 'sim' and 'obs' and adding a user-defined constant
#            during computations

eps <- 0.01
NSE(sim=sim, obs=obs, fun=log, epsilon.type="otherValue", epsilon.value=eps)

# Verifying the previous value:
lsim <- log(sim+eps)
lobs <- log(obs+eps)
NSE(sim=lsim, obs=lobs)

##################
# Example 7: NSE for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and using a user-defined factor
#            to multiply the mean of the observed values to obtain the constant
#            to be added to 'sim' and 'obs' during computations

fact <- 1/50
NSE(sim=sim, obs=obs, fun=log, epsilon.type="otherFactor", epsilon.value=fact)

# Verifying the previous value:
eps  <- fact*mean(obs, na.rm=TRUE)
lsim <- log(sim+eps)
lobs <- log(obs+eps)
NSE(sim=lsim, obs=lobs)

##################
# Example 8: NSE for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying a
#            user-defined function to 'sim' and 'obs' during computations

fun1 <- function(x) {sqrt(x+1)}

NSE(sim=sim, obs=obs, fun=fun1)

# Verifying the previous value, with the epsilon value following Pushpalatha2012
sim1 <- sqrt(sim+1)
obs1 <- sqrt(obs+1)
NSE(sim=sim1, obs=obs1)
```

---

pbias *Percent Bias*

Description

Percent Bias between `sim` and `obs`, with treatment of missing values.

Usage

```
pbias(sim, obs, ...)

## Default S3 method:
pbias(sim, obs, na.rm=TRUE, dec=1, fun=NULL, ...,
          epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
               epsilon.value=NA)

## S3 method for class 'data.frame'
pbias(sim, obs, na.rm=TRUE, dec=1, fun=NULL, ...,
          epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
               epsilon.value=NA)

## S3 method for class 'matrix'
pbias(sim, obs, na.rm=TRUE, dec=1, fun=NULL, ...,
          epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
               epsilon.value=NA)

## S3 method for class 'zoo'
pbias(sim, obs, na.rm=TRUE, dec=1, fun=NULL, ...,
          epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
               epsilon.value=NA)
```

Arguments

`sim` numeric, zoo, matrix or data.frame with simulated values

`obs` numeric, zoo, matrix or data.frame with observed values

`na.rm` a logical value indicating whether 'NA' should be stripped before the computation proceeds.
When an 'NA' value is found at the i-th position in `obs` **OR** `sim`, the i-th value of `obs` **AND** `sim` are removed before the computation.

`dec` numeric, specifying the number of decimal places used to rounf the output object. Default value is 1.

`fun` function to be applied to `sim` and `obs` in order to obtain transformed values thereof before computing this goodness-of-fit index.

The first argument MUST BE a numeric vector with any name (e.g., `x`), and additional arguments are passed using `...`.

`...` arguments passed to `fun`, in addition to the mandatory first numeric vector.

`epsilon.type` argument used to define a numeric value to be added to both `sim` and `obs` before applying `fun`.

It is was designed to allow the use of logarithm and other similar functions that do not work with zero values.

Valid values of `epsilon.type` are:

1) `"none"`: `sim` and `obs` are used by `fun` without the addition of any numeric value. This is the default option.

2) "Pushpalatha2012": one hundredth (1/100) of the mean observed values is added to both sim and obs before applying fun, as described in Pushpalatha et al. (2012).

3) "otherFactor": the numeric value defined in the epsilon.value argument is used to multiply the the mean observed values, instead of the one hundredth (1/100) described in Pushpalatha et al. (2012). The resulting value is then added to both sim and obs, before applying fun.

4) "otherValue": the numeric value defined in the epsilon.value argument is directly added to both sim and obs, before applying fun.

epsilon.value -) when epsilon.type="otherValue" it represents the numeric value to be added to both sim and obs before applying fun.
-) when epsilon.type="otherFactor" it represents the numeric factor used to multiply the mean of the observed values, instead of the one hundredth (1/100) described in Pushpalatha et al. (2012). The resulting value is then added to both sim and obs before applying fun.

Details

$$PBIAS = 100 \frac{\sum_{i=1}^{N} (S_i - O_i)}{\sum_{i=1}^{N} O_i}$$

Percent bias (PBIAS) measures the average tendency of the simulated values to be larger or smaller than their observed ones.

The optimal value of PBIAS is 0.0, with low-magnitude values indicating accurate model simulation. Positive values indicate overestimation bias, whereas negative values indicate model underestimation bias

Value

Percent bias between sim and obs. The result is given in percentage (%)

If sim and obs are matrixes, the returned value is a vector, with the percent bias between each column of sim and obs.

Note

obs and sim has to have the same length/dimension

The missing values in obs and sim are removed before the computation proceeds, and only those positions with non-missing values in obs and sim are considered in the computation

Author(s)

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

References

Yapo, P.O.; Gupta, H.V.; Sorooshian S. (1996). Automatic calibration of conceptual rainfall-runoff models: sensitivity to calibration data. Journal of Hydrology. v181 i1-4. 23–48. doi:10.1016/0022-1694(95)02918-4

Sorooshian, S., Q. Duan, and V. K. Gupta. 1993. Calibration of rainfall-runoff models: Application of global optimization to the Sacramento Soil Moisture Accounting Model, Water Resources Research, 29 (4), 1185-1194, doi:10.1029/92WR02617.

See Also

pbias, pbiasfdc, mae, mse, rmse, ubRMSE, nrmse, ssq, gof, ggof

Examples

```
##################
# Example 1: basic ideal case
obs <- 1:10
sim <- 1:10
pbias(sim, obs)

obs <- 1:10
sim <- 2:11
pbias(sim, obs)

##################
# Example 2:
# Loading daily streamflows of the Ega River (Spain), from 1961 to 1970
data(EgaEnEstellaQts)
obs <- EgaEnEstellaQts

# Generating a simulated daily time series, initially equal to the observed series
sim <- obs

# Computing the 'pbias' for the "best" (unattainable) case
pbias(sim=sim, obs=obs)

##################
# Example 3: pbias for simulated values equal to observations plus random noise
#            on the first half of the observed values.
#            This random noise has more relative importance for ow flows than
#            for medium and high flows.

# Randomly changing the first 1826 elements of 'sim', by using a normal distribution
# with mean 10 and standard deviation equal to 1 (default of 'rnorm').
sim[1:1826] <- obs[1:1826] + rnorm(1826, mean=10)
ggof(sim, obs)

pbias(sim=sim, obs=obs)

##################
# Example 4: pbias for simulated values equal to observations plus random noise
```

```
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' during computations.

pbias(sim=sim, obs=obs, fun=log)

# Verifying the previous value:
lsim <- log(sim)
lobs <- log(obs)
pbias(sim=lsim, obs=lobs)

##################
# Example 5: pbias for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and adding the Pushpalatha2012 constant
#            during computations

pbias(sim=sim, obs=obs, fun=log, epsilon.type="Pushpalatha2012")

# Verifying the previous value, with the epsilon value following Pushpalatha2012
eps  <- mean(obs, na.rm=TRUE)/100
lsim <- log(sim+eps)
lobs <- log(obs+eps)
pbias(sim=lsim, obs=lobs)

##################
# Example 6: pbias for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and adding a user-defined constant
#            during computations

eps <- 0.01
pbias(sim=sim, obs=obs, fun=log, epsilon.type="otherValue", epsilon.value=eps)

# Verifying the previous value:
lsim <- log(sim+eps)
lobs <- log(obs+eps)
pbias(sim=lsim, obs=lobs)

##################
# Example 7: pbias for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and using a user-defined factor
#            to multiply the mean of the observed values to obtain the constant
#            to be added to 'sim' and 'obs' during computations

fact <- 1/50
pbias(sim=sim, obs=obs, fun=log, epsilon.type="otherFactor", epsilon.value=fact)

# Verifying the previous value:
eps  <- fact*mean(obs, na.rm=TRUE)
lsim <- log(sim+eps)
lobs <- log(obs+eps)
pbias(sim=lsim, obs=lobs)
```

```
##################
# Example 8: pbias for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying a
#            user-defined function to 'sim' and 'obs' during computations

fun1 <- function(x) {sqrt(x+1)}

pbias(sim=sim, obs=obs, fun=fun1)

# Verifying the previous value, with the epsilon value following Pushpalatha2012
sim1 <- sqrt(sim+1)
obs1 <- sqrt(obs+1)
pbias(sim=sim1, obs=obs1)
```

---

pbiasfdc *Percent Bias in the Slope of the Midsegment of the Flow Duration Curve*

Description

Percent Bias in the slope of the midsegment of the flow duration curve (FDC) [%]. It is related to the vertical soil moisture redistribution.

Usage

```
pbiasfdc(sim, obs, ...)

## Default S3 method:
pbiasfdc(sim, obs, lQ.thr=0.6, hQ.thr=0.1, na.rm=TRUE,
       plot=TRUE, verbose=FALSE, fun=NULL, ...,
       epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
       epsilon.value=NA)

## S3 method for class 'data.frame'
pbiasfdc(sim, obs, lQ.thr=0.6, hQ.thr=0.1, na.rm=TRUE,
        plot=TRUE, verbose=FALSE, fun=NULL, ...,
        epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
        epsilon.value=NA)

## S3 method for class 'matrix'
pbiasfdc(sim, obs, lQ.thr=0.6, hQ.thr=0.1, na.rm=TRUE,
        plot=TRUE, verbose=FALSE, fun=NULL, ...,
        epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
        epsilon.value=NA)

## S3 method for class 'zoo'
```

```
pbiasfdc(sim, obs, lQ.thr=0.6, hQ.thr=0.1, na.rm=TRUE,
        plot=TRUE, verbose=FALSE, fun=NULL, ...,
        epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
        epsilon.value=NA)
```

Arguments

`sim` numeric, zoo, matrix or data.frame with simulated values

`obs` numeric, zoo, matrix or data.frame with observed values

`lQ.thr` numeric, used to classify low flows. All the streamflows with a probability of exceedence larger or equal to `lQ.thr` are classified as low flows

`hQ.thr` numeric, used to classify high flows. All the streamflows with a probability of exceedence larger or equal to `hQ.thr` are classified as high flows

`na.rm` a logical value indicating whether 'NA' values should be stripped before the computation proceeds.

`plot` a logical value indicating if the flow duration curves corresponding to `obs` and `sim` have to be plotted or not.

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

`fun` function to be applied to `sim` and `obs` in order to obtain transformed values thereof before computing this goodness-of-fit index.

The first argument MUST BE a numeric vector with any name (e.g., `x`), and additional arguments are passed using `...`.

`...` arguments passed to `fun`, in addition to the mandatory first numeric vector.

`epsilon.type` argument used to define a numeric value to be added to both `sim` and `obs` before applying `fun`.

It is was designed to allow the use of logarithm and other similar functions that do not work with zero values.

Valid values of `epsilon.type` are:

1) `"none"`: `sim` and `obs` are used by `fun` without the addition of any numeric value. This is the default option.

2) `"Pushpalatha2012"`: one hundredth (1/100) of the mean observed values is added to both `sim` and `obs` before applying `fun`, as described in Pushpalatha et al. (2012).

3) `"otherFactor"`: the numeric value defined in the `epsilon.value` argument is used to multiply the the mean observed values, instead of the one hundredth (1/100) described in Pushpalatha et al. (2012). The resulting value is then added to both `sim` and `obs`, before applying `fun`.

4) `"otherValue"`: the numeric value defined in the `epsilon.value` argument is directly added to both `sim` and `obs`, before applying `fun`.

`epsilon.value` -) when `epsilon.type="otherValue"` it represents the numeric value to be added to both `sim` and `obs` before applying `fun`.
-) when `epsilon.type="otherFactor"` it represents the numeric factor used to multiply the mean of the observed values, instead of the one hundredth (1/100) described in Pushpalatha et al. (2012). The resulting value is then added to both `sim` and `obs` before applying `fun`.

Value

Percent Bias in the slope of the midsegment of the flow duration curve, between `sim` and `obs`.

If `sim` and `obs` are matrixes, the returned value is a vector, with the Percent Bias in the slope of the midsegment of the flow duration curve, between each column of `sim` and `obs`.

Note

The result is given in percentage (%).

It requires the **hydroTSM** package.

Author(s)

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

References

Yilmaz, K.K., Gupta, H.V. ; Wagener, T. (2008), A process-based diagnostic approach to model evaluation: Application to the NWS distributed hydrologic model, Water Resources Research, 44, W09417, doi:10.1029/2007WR006716.

See Also

`fdc`, `pbias`, `mae`, `mse`, `rmse`, `ubRMSE`, `nrmse`, `ssq`, `gof`, `ggof`

Examples

```
## Not run:
##################
# Example 1: basic ideal case
obs <- 1:10
sim <- 1:10
pbiasfdc(sim, obs)

obs <- 1:10
sim <- 2:11
pbiasfdc(sim, obs)

##################
# Example 2:
# Loading daily streamflows of the Ega River (Spain), from 1961 to 1970
data(EgaEnEstellaQts)
obs <- EgaEnEstellaQts

# Generating a simulated daily time series, initially equal to the observed series
sim <- obs

# Computing the 'pbiasfdc' for the "best" (unattainable) case
pbiasfdc(sim=sim, obs=obs)
```

```
##################
# Example 3: pbiasfdc for simulated values equal to observations plus random noise
#            on the first half of the observed values.
#            This random noise has more relative importance for ow flows than
#            for medium and high flows.

# Randomly changing the first 1826 elements of 'sim', by using a normal distribution
# with mean 10 and standard deviation equal to 1 (default of 'rnorm').
sim[1:1826] <- obs[1:1826] + rnorm(1826, mean=10)
ggof(sim, obs)

pbiasfdc(sim=sim, obs=obs)

##################
# Example 4: pbiasfdc for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' during computations.

pbiasfdc(sim=sim, obs=obs, fun=log)

# Verifying the previous value:
lsim <- log(sim)
lobs <- log(obs)
pbiasfdc(sim=lsim, obs=lobs)

##################
# Example 5: pbiasfdc for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and adding the Pushpalatha2012 constant
#            during computations

pbiasfdc(sim=sim, obs=obs, fun=log, epsilon.type="Pushpalatha2012")

# Verifying the previous value, with the epsilon value following Pushpalatha2012
eps  <- mean(obs, na.rm=TRUE)/100
lsim <- log(sim+eps)
lobs <- log(obs+eps)
pbiasfdc(sim=lsim, obs=lobs)

##################
# Example 6: pbiasfdc for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and adding a user-defined constant
#            during computations

eps <- 0.01
pbiasfdc(sim=sim, obs=obs, fun=log, epsilon.type="otherValue", epsilon.value=eps)

# Verifying the previous value:
lsim <- log(sim+eps)
lobs <- log(obs+eps)
pbiasfdc(sim=lsim, obs=lobs)
```

```
##################
# Example 7: pbiasfdc for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and using a user-defined factor
#            to multiply the mean of the observed values to obtain the constant
#            to be added to 'sim' and 'obs' during computations

fact <- 1/50
pbiasfdc(sim=sim, obs=obs, fun=log, epsilon.type="otherFactor", epsilon.value=fact)

# Verifying the previous value:
eps  <- fact*mean(obs, na.rm=TRUE)
lsim <- log(sim+eps)
lobs <- log(obs+eps)
pbiasfdc(sim=lsim, obs=lobs)

##################
# Example 8: pbiasfdc for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying a
#            user-defined function to 'sim' and 'obs' during computations

fun1 <- function(x) {sqrt(x+1)}

pbiasfdc(sim=sim, obs=obs, fun=fun1)

# Verifying the previous value, with the epsilon value following Pushpalatha2012
sim1 <- sqrt(sim+1)
obs1 <- sqrt(obs+1)
pbiasfdc(sim=sim1, obs=obs1)

## End(Not run)
```

---

pfactor *P-factor*

Description

P-factor is the percent of observations that are within the given uncertainty bounds.

Ideally, i.e., with a combination of model structure and parameter values that perfectly represents the catchment under study, and in absence of measurement errors and other additional sources of uncertainty, all the simulated values should be in a perfect match with the observations, leading to a *P-factor* equal to 1, and an *R-factor* equal to zero. However, in real-world applications we aim at encompassing as much observations as possible within the given uncertainty bounds (*P-factor* close to 1) while keeping the width of the uncertainty bounds as small as possible (*R-factor* close to 0), in order to avoid obtaining a good bracketing of observations at expense of uncertainty bounds too wide to be informative for the decision-making process.

Usage

```
pfactor(x, ...)

## Default S3 method:
pfactor(x, lband, uband, na.rm=TRUE, ...)

## S3 method for class 'data.frame'
pfactor(x, lband, uband, na.rm=TRUE, ...)

## S3 method for class 'matrix'
pfactor(x, lband, uband, na.rm=TRUE, ...)
```

Arguments

`x`	ts or zoo object with the observed values.
`lband`	numeric, ts or zoo object with the values of the lower uncertainty bound
`uband`	numeric, ts or zoo object with the values of the upper uncertainty bound
`na.rm`	a logical value indicating whether 'NA' values should be stripped before the computation proceeds.
`...`	further arguments passed to or from other methods.

Value

Percent of the `x` observations that are within the given uncertainty bounds given by `lband` and `uband`.

If `sim` and `obs` are matrixes, the returned value is a vector, with the *P-factor* between each column of `sim` and `obs`.

Note

So far, the argument `na.rm` is not being taken into account.

Author(s)

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

References

Abbaspour, K.C.; Faramarzi, M.; Ghasemi, S.S.; Yang, H. (2009), Assessing the impact of climate change on water resources in Iran, Water Resources Research, 45(10), W10,434, doi:10.1029/2008WR007615.

Abbaspour, K.C., Yang, J. ; Maximov, I.; Siber, R.; Bogner, K.; Mieleitner, J. ; Zobrist, J.; Srinivasan, R. (2007), Modelling hydrology and water quality in the pre-alpine/alpine Thur watershed using SWAT, Journal of Hydrology, 333(2-4), 413-430, doi:10.1016/j.jhydrol.2006.09.014.

Schuol, J.; Abbaspour, K.C.; Srinivasan, R.; Yang, H. (2008b), Estimation of freshwater availability in the West African sub-continent using the SWAT hydrologic model, Journal of Hydrology, 352(1-2), 30, doi:10.1016/j.jhydrol.2007.12.025

Abbaspour, K.C. (2007), User manual for SWAT-CUP, SWAT calibration and uncertainty analysis programs, 93pp, Eawag: Swiss Fed. Inst. of Aquat. Sci. and Technol. Dubendorf, Switzerland.

See Also

rfactor, plotbands

Examples

```
x <- 1:10
lband <- x - 0.1
uband <- x + 0.1
pfactor(x, lband, uband)

lband <- x - rnorm(10)
uband <- x + rnorm(10)
pfactor(x, lband, uband)

#############
# Loading daily streamflows of the Ega River (Spain), from 1961 to 1970
data(EgaEnEstellaQts)
obs <- EgaEnEstellaQts

# Selecting only the daily values belonging to the year 1961
obs <- window(obs, end=as.Date("1961-12-31"))

# Generating the lower and upper uncertainty bounds, centred at the observations
lband <- obs - 5
uband <- obs + 5

pfactor(obs, lband, uband)

# Randomly generating the lower and upper uncertainty bounds
uband <- obs + rnorm(length(obs))
lband <- obs - rnorm(length(obs))

pfactor(obs, lband, uband)
```

---

plot2 *Plotting 2 Time Series*

Description

Plotting of 2 time series, in two different vertical windows or overlapped in the same window.
It requires the **hydroTSM** package.

Usage

```
plot2(x, y, plot.type = "multiple",
      tick.tstep = "auto", lab.tstep = "auto", lab.fmt=NULL,
      main, xlab = "Time", ylab,
      cal.ini=NA, val.ini=NA, date.fmt="%Y-%m-%d",
      gof.leg = FALSE, gof.digits=2,
     gofs=c("ME", "MAE", "RMSE", "NRMSE", "PBIAS", "RSR", "rSD", "NSE", "mNSE",
            "rNSE", "d", "md", "rd", "r", "R2", "bR2", "KGE", "VE"),
      legend, leg.cex = 1,
      col = c("black", "blue"),
      cex = c(0.5, 0.5), cex.axis=1.2, cex.lab=1.2,
      lwd= c(1,1), lty=c(1,3), pch = c(1, 9),
      pt.style = "ts", add = FALSE,
      ...)
```

Arguments

x	time series that will be plotted. class(x) must be ts or zoo. If leg.gof=TRUE, then x is considered as **simulated** (for some goodness-of-fit functions this is important)
y	time series that will be plotted. class(x) must be ts or zoo. If leg.gof=TRUE, then y is considered as **observed** values (for some goodness-of-fit functions this is important)
plot.type	character, indicating if the 2 ts have to be plotted in the same window or in two different vertical ones. Valid values are: -) single : (default) superimposes the 2 ts on a single plot -) multiple: plots the 2 series on 2 multiple vertical plots
tick.tstep	character, indicating the time step that have to be used for putting the ticks on the time axis. Valid values are: auto, years, months,weeks, days, hours, minutes, seconds.
lab.tstep	character, indicating the time step that have to be used for putting the labels on the time axis. Valid values are: auto, years, months,weeks, days, hours, minutes, seconds.
lab.fmt	Character indicating the format to be used for the label of the axis. See lab.fmt in drawTimeAxis.
main	an overall title for the plot: see title
xlab	label for the 'x' axis
ylab	label for the 'y' axis
cal.ini	OPTIONAL. Character, indicating the date in which the calibration period started. When cal.ini is provided, all the values in obs and sim with dates previous to cal.ini are SKIPPED from the computation of the goodness-of-fit measures (when gof.leg=TRUE), but their values are still plotted, in order to examine if the warming up period was too short, acceptable or too long for the chosen calibration period. In addition, a vertical red line in drawn at this date.
val.ini	OPTIONAL. Character with the date in which the validation period started. ONLY used for drawing a vertical red line at this date.

`date.fmt` : OPTIONAL. Character indicating the format in which the dates entered are stored in `cal.ini` and `val.ini`. Default value is `%Y-%m-%d`. ONLY required when `cal.ini` or `val.ini` is provided.

`gof.leg` : logical, indicating if several numerical goodness-of-fit values have to be computed between `sim` and `obs`, and plotted as a legend on the graph. If `gof.leg=TRUE` (default value), then `x` is considered as observed and `y` as simulated values (for some gof functions this is important). This legend is ONLY plotted when `plot.type="single"`

`gof.digits` : OPTIONAL, only used when `gof.leg=TRUE`. Decimal places used for rounding the goodness-of-fit indexes.

`gofs` : character, with one or more strings indicating the goodness-of-fit measures to be shown in the legend of the plot when `gof.leg=TRUE`.
Possible values are in `c("ME", "MAE", "MSE", "RMSE", "NRMSE", "PBIAS", "RSR", "rSD", "NSE", "mNSE", "rNSE", "d", "md", "rd", "cp", "r", "R2", "bR2", "KGE", "VE")`.

`legend` : vector of length 2 to appear in the legend.

`leg.cex` : numeric, indicating the character expansion factor *relative* to current 'par("cex")'. Used for text, and provides the default for 'pt.cex' and 'title.cex'. Default value = 1
So far, it controls the expansion factor of the 'GoF' legend and the legend referring to `x` and `y`

`col` : character, with the colors of `x` and `y`

`cex` : numeric, with the values controlling the size of text and symbols of `x` and `y` with respect to the default

`cex.axis` : numeric, with the magnification of axis annotation relative to 'cex'. See `par`.

`cex.lab` : numeric, with the magnification to be used for `x` and `y` labels relative to the current setting of 'cex'. See `par`.

`lwd` : vector with the line width of `x` and `y`

`lty` : vector with the line type of `x` and `y`

`pch` : vector with the type of symbol for `x` and `y`. (e.g.: 1: white circle; 9: white rhombus with a cross inside)

`pt.style` : Character, indicating if the 2 ts have to be plotted as lines or bars. Valid values are:
-) `ts` : (default) each ts is plotted as a lines along the `x` axis
-) `bar`: the 2 series are plotted as a barplot.

`add` : logical indicating if other plots will be added in further calls to this function.
-) `FALSE` => the plot and the legend are plotted on the same graph
-) `TRUE` => the legend is plotted in a new graph, usually when called from another function (e.g.: `ggof`)

`...` : further arguments passed to `plot.zoo` function for plotting `x`, or from other methods

Note

It requires the package **hydroTSM**.

Author(s)

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

See Also

ggof, plot_pq

Examples

```
sim <- 2:11
obs <- 1:10
## Not run:
plot2(sim, obs)

## End(Not run)

##################
# Loading daily streamflows of the Ega River (Spain), from 1961 to 1970
data(EgaEnEstellaQts)
obs <- EgaEnEstellaQts

# Generating a simulated daily time series, initially equal to the observed series
sim <- obs

# Randomly changing the first 2000 elements of 'sim', by using a normal distribution
# with mean 10 and standard deviation equal to 1 (default of 'rnorm').
sim[1:2000] <- obs[1:2000] + rnorm(2000, mean=10)

# Plotting 'sim' and 'obs' in 2 separate panels
plot2(x=obs, y=sim)

# Plotting 'sim' and 'obs' in the same window
plot2(x=obs, y=sim, plot.type="single")
```

---

plotbands — *Plot a ts with observed values and two confidence bounds*

Description

It plots a ts with observed values and two confidence bounds. Optionally can also add a simulated time series, in order to be compared with 'x'.

Usage

```
plotbands(x, lband, uband, sim,
          dates, date.fmt="%Y-%m-%d",
          gof.leg= TRUE, gof.digits=2,
          legend=c("Obs", "Sim", "95PPU"), leg.cex=1,
          bands.col="lightblue", border= NA,
```

```
tick.tstep= "auto", lab.tstep= "auto", lab.fmt=NULL,
cal.ini=NA, val.ini=NA,
main="Confidence Bounds for 'x'",
xlab="Time", ylab="Q, [m3/s]", ylim,
col=c("black", "blue"), type= c("lines", "lines"),
cex= c(0.5, 0.5), cex.axis=1.2, cex.lab=1.2,
lwd=c(0.6, 1), lty=c(3, 4), pch=c(1,9), ...)
```

Arguments

x	zoo or xts object with the observed values.
lband	zoo or xts object with the values of the lower band.
uband	zoo or xts object with the values of the upper band.
sim	OPTIONAL. zoo or xts object with the simulated values.
dates	OPTIONAL. Date, factor, or character object indicating the dates that will be assigned to `x`, `lband`, `uband`, and `sim` (when provided). If `dates` is a factor or character vector, its values are converted to dates using the date format specified by `date.fmt`. When `x`, `lband`, `uband`, and `sim` are already of zoo class, **the values provided by `dates` over-write the original dates of the objects**.
date.fmt	OPTIONAL. Character indicating the format in which the dates entered are stored in `cal.ini` and `val.ini`. See `format` in `as.Date`. Default value is `%Y-%m-%d` ONLY required when `cal.ini`, `val.ini` or `dates` is provided.
gof.leg	logical indicating if the p-factor and r-factor have to be computed and plotted as legends on the graph.
gof.digits	OPTIONAL, numeric. Only used when `gof.leg=TRUE`. Decimal places used for rounding the goodness-of-fit indexes
legend	OPTIONAL. logical or character vector of length 3 with the strings that will be used for the legend of the plot. -) When `legend` is a character vector, the first element is used for labelling the observed series, the second for labelling the simulated series and the third one for the predictive uncertainty bounds. Default value is `c("obs", "sim", "95PPU")` -) When `legend=FALSE`, the legend is not drawn.
leg.cex	OPTIONAL. numeric. Used for the GoF legend. Character expansion factor *relative* to current 'par("cex")'. Used for text, and provides the default for 'pt.cex' and 'title.cex'. Default value is 1.
bands.col	See `polygon`. Color to be used for filling the area between the lower and upper uncertainty bound.
border	See `polygon`. The color to draw the border. The default, 'NULL', means to use 'par("fg")'. Use 'border = NA' to omit borders.
tick.tstep	character, indicating the time step that have to be used for putting the ticks on the time axis. Valid values are: `auto`, `years`, `months`,`weeks`, `days`, `hours`, `minutes`, `seconds`.

`lab.tstep`	character, indicating the time step that have to be used for putting the labels on the time axis. Valid values are: `auto`, `years`, `months`,`weeks`, `days`, `hours`, `minutes`, `seconds`.
`lab.fmt`	Character indicating the format to be used for the label of the axis. See `lab.fmt` in `drawTimeAxis`.
`cal.ini`	OPTIONAL. Character with the date in which the calibration period started. ONLY used for drawing a vertical red line at this date.
`val.ini`	OPTIONAL. Character with the date in which the validation period started. ONLY used for drawing a vertical red line at this date.
`main`	an overall title for the plot: see 'title'
`xlab`	a title for the x axis: see 'title'
`ylab`	a title for the y axis: see 'title'
`ylim`	the y limits of the plot. See `plot.default`.
`col`	colors to be used for plotting the `x` and `sim` ts.
`type`	character. Indicates if the observed and simulated series have to be plotted as lines or points. Possible values are: -) `lines` : the observed/simulated series are plotted as lines -) `points`: the observed/simulated series are plotted as points
`cex`	See code plot.default. A numerical vector giving the amount by which plotting characters and symbols should be scaled relative to the default. This works as a multiple of 'par("cex")'. 'NULL' and 'NA' are equivalent to '1.0'. Note that this does not affect annotation.
`cex.axis`	magnification of axis annotation relative to 'cex'.
`cex.lab`	Magnification to be used for x and y labels relative to the current setting of 'cex'. See '?par'.
`lwd`	See `plot.default`. The line width, see 'par'.
`lty`	See `plot.default`. The line type, see 'par'.
`pch`	numeric, with the type of symbol for `x` and `y`. (e.g.: 1: white circle; 9: white rhombus with a cross inside)
`...`	further arguments passed to the `points` function for plotting `x`, or from other methods

Note

It requires the **hydroTSM** package

Author(s)

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

See Also

`pfactor`, `rfactor`

Examples

```
# Loading daily streamflows of the Ega River (Spain), from 1961 to 1970
data(EgaEnEstellaQts)
obs <- EgaEnEstellaQts

# Selecting only the daily values belonging to the year 1961
obs <- window(obs, end=as.Date("1961-12-31"))

# Generating the lower and upper uncertainty bounds
lband <- obs - 5
uband <- obs + 5

## Not run:
plotbands(obs, lband, uband)

## End(Not run)

# Randomly generating a simulated time series
sim <- obs + rnorm(length(obs), mean=3)

## Not run:
plotbands(obs, lband, uband, sim)

## End(Not run)
```

`plotbandsonly` *Adds uncertainty bounds to an existing plot.*

Description

Adds a polygon representing uncertainty bounds to an existing plot.

Usage

```
plotbandsonly(lband, uband, dates, date.fmt="%Y-%m-%d",
          bands.col="lightblue", border= NA, ...)
```

Arguments

`lband`	zoo or xts object with the values of the lower band.
`uband`	zoo or xts object with the values of the upper band.
`dates`	OPTIONAL. Date, factor, or character object indicating the dates that will be assigned to `lband` and `uband`. If `dates` is a factor or character vector, its values are converted to dates using the date format specified by `date.fmt`. When `lband` and `uband` are already of zoo class, **the values provided by** `dates` **over-write the original dates of the objects**.

date.fmt	OPTIONAL. Character indicating the format of dates. See format in as.Date.
bands.col	See polygon. Color to be used for filling the area between the lower and upper uncertainty bound.
border	See polygon. The color to draw the border. The default, 'NULL', means to use 'par("fg")'. Use 'border = NA' to omit borders.
...	further arguments passed to the polygon function for plotting the bands, or from other methods

Note

It requires the **hydroTSM** package

Author(s)

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

See Also

pfactor, rfactor

Examples

```
# Loading daily streamflows of the Ega River (Spain), from 1961 to 1970
data(EgaEnEstellaQts)
obs <- EgaEnEstellaQts

# Selecting only the daily values belonging to the year 1961
obs <- window(obs, end=as.Date("1961-12-31"))

# Generating the lower and upper uncertainty bounds
lband <- obs - 5
uband <- obs + 5

## Not run:
plot(obs, type="n")
plotbandsonly(lband, uband)
points(obs, col="blue", cex=0.6, type="o")

## End(Not run)
```

R2 *Coefficient of determination*

Description

coefficient of determination between sim and obs, with treatment of missing values.

Usage

```
R2(sim, obs, ...)

## Default S3 method:
R2(sim, obs, fun=NULL, ...,
        epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
           epsilon.value=NA)

## S3 method for class 'matrix'
R2(sim, obs, na.rm=TRUE, fun=NULL, ...,
        epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
           epsilon.value=NA)

## S3 method for class 'data.frame'
R2(sim, obs, na.rm=TRUE, fun=NULL, ...,
        epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
           epsilon.value=NA)

## S3 method for class 'zoo'
R2(sim, obs, na.rm=TRUE, fun=NULL, ...,
        epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
          epsilon.value=NA)
```

Arguments

`sim` numeric, zoo, matrix or data.frame with simulated values

`obs` numeric, zoo, matrix or data.frame with observed values

`na.rm` a logical value indicating whether 'NA' should be stripped before the computation proceeds.
When an 'NA' value is found at the i-th position in `obs` **OR** `sim`, the i-th value of `obs` **AND** `sim` are removed before the computation.

`fun` function to be applied to `sim` and `obs` in order to obtain transformed values thereof before computing this goodness-of-fit index.

The first argument MUST BE a numeric vector with any name (e.g., `x`), and additional arguments are passed using `...`.

`...` arguments passed to `fun`, in addition to the mandatory first numeric vector.

`epsilon.type` argument used to define a numeric value to be added to both `sim` and `obs` before applying `fun`.

It is was designed to allow the use of logarithm and other similar functions that do not work with zero values.

Valid values of `epsilon.type` are:

1) `"none"`: `sim` and `obs` are used by `fun` without the addition of any numeric value. This is the default option.

2) `"Pushpalatha2012"`: one hundredth (1/100) of the mean observed values is added to both `sim` and `obs` before applying `fun`, as described in Pushpalatha et al. (2012).

3) "otherFactor": the numeric value defined in the epsilon.value argument is used to multiply the the mean observed values, instead of the one hundredth (1/100) described in Pushpalatha et al. (2012). The resulting value is then added to both sim and obs, before applying fun.

4) "otherValue": the numeric value defined in the epsilon.value argument is directly added to both sim and obs, before applying fun.

epsilon.value numeric value to be added to both sim and obs when epsilon.type="otherValue".

Details

The coefficient of determination (R2) is the proportion of the variation in the dependent variable that is predictable from the independent variable(s).

It is a statistic used in the context of statistical models whose main purpose is either the prediction of future outcomes or the testing of hypotheses, on the basis of other related information. It provides a measure of how well observed outcomes are replicated by the model, based on the proportion of total variation of outcomes explained by the model.

The coefficient of determination is a statistical measure of how well the regression predictions approximate the real data points. An R2 of 1 indicates that the regression predictions perfectly fit the data.

Values of R2 outside the range 0 to 1 occur when the model fits the data worse than the worst possible least-squares predictor (equivalent to a horizontal hyperplane at a height equal to the mean of the observed data). This occurs when a wrong model was chosen, or nonsensical constraints were applied by mistake.

Value

Coefficient of determination between sim and obs.

If sim and obs are matrixes, the returned value is a vector, with the coefficient of determination between each column of sim and obs.

Note

obs and sim has to have the same length/dimension

The missing values in obs and sim are removed before the computation proceeds, and only those positions with non-missing values in obs and sim are considered in the computation

Author(s)

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

References

https://en.wikipedia.org/wiki/Coefficient_of_determination

Box, G.E. (1966). Use and abuse of regression. Technometrics, 8(4), 625-629. doi:10.1080/00401706.1966.10490407.

Hahn, G.J. (1973). The coefficient of determination exposed. Chemtech, 3(10), 609-612. Aailable online at: https://www2.hawaii.edu/~cbaajwe/Ph.D.Seminar/Hahn1973.pdf.

Barrett, J.P. (1974). The coefficient of determination-some limitations. The American Statistician, 28(1), 19-20. doi:10.1080/00031305.1974.10479056.

See Also

cor

Examples

```
##################
# Example 1: basic ideal case
obs <- 1:10
sim <- 1:10
R2(sim, obs)

obs <- 1:10
sim <- 2:11
R2(sim, obs)

##################
# Example 2:
# Loading daily streamflows of the Ega River (Spain), from 1961 to 1970
data(EgaEnEstellaQts)
obs <- EgaEnEstellaQts

# Generating a simulated daily time series, initially equal to the observed series
sim <- obs

# Computing the 'R2' for the "best" (unattainable) case
R2(sim=sim, obs=obs)

##################
# Example 3: R2 for simulated values equal to observations plus random noise
#            on the first half of the observed values.
#            This random noise has more relative importance for ow flows than
#            for medium and high flows.

# Randomly changing the first 1826 elements of 'sim', by using a normal distribution
# with mean 10 and standard deviation equal to 1 (default of 'rnorm').
sim[1:1826] <- obs[1:1826] + rnorm(1826, mean=10)
ggof(sim, obs)

R2(sim=sim, obs=obs)

##################
# Example 4: R2 for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' during computations.

R2(sim=sim, obs=obs, fun=log)
```

```
# Verifying the previous value:
lsim <- log(sim)
lobs <- log(obs)
R2(sim=lsim, obs=lobs)

##################
# Example 5: R2 for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and adding the Pushpalatha2012 constant
#            during computations

R2(sim=sim, obs=obs, fun=log, epsilon.type="Pushpalatha2012")

# Verifying the previous value, with the epsilon value following Pushpalatha2012
eps  <- mean(obs, na.rm=TRUE)/100
lsim <- log(sim+eps)
lobs <- log(obs+eps)
R2(sim=lsim, obs=lobs)

##################
# Example 6: R2 for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and adding a user-defined constant
#            during computations

eps <- 0.01
R2(sim=sim, obs=obs, fun=log, epsilon.type="otherValue", epsilon.value=eps)

# Verifying the previous value:
lsim <- log(sim+eps)
lobs <- log(obs+eps)
R2(sim=lsim, obs=lobs)

##################
# Example 7: R2 for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and using a user-defined factor
#            to multiply the mean of the observed values to obtain the constant
#            to be added to 'sim' and 'obs' during computations

fact <- 1/50
R2(sim=sim, obs=obs, fun=log, epsilon.type="otherFactor", epsilon.value=fact)

# Verifying the previous value:
eps  <- fact*mean(obs, na.rm=TRUE)
lsim <- log(sim+eps)
lobs <- log(obs+eps)
R2(sim=lsim, obs=lobs)

##################
# Example 8: R2 for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying a
```

```
#           user-defined function to 'sim' and 'obs' during computations

fun1 <- function(x) {sqrt(x+1)}

R2(sim=sim, obs=obs, fun=fun1)

# Verifying the previous value, with the epsilon value following Pushpalatha2012
sim1 <- sqrt(sim+1)
obs1 <- sqrt(obs+1)
R2(sim=sim1, obs=obs1)
```

---

rd *Relative Index of Agreement*

Description

This function computes the Relative Index of Agreement (d) between sim and obs, with treatment of missing values.
If x is a matrix or a data frame, a vector of the relative index of agreement among the columns is returned.

Usage

```
rd(sim, obs, ...)

## Default S3 method:
rd(sim, obs, na.rm=TRUE, fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
            epsilon.value=NA)

## S3 method for class 'data.frame'
rd(sim, obs, na.rm=TRUE, fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
            epsilon.value=NA)

## S3 method for class 'matrix'
rd(sim, obs, na.rm=TRUE, fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
            epsilon.value=NA)

## S3 method for class 'zoo'
rd(sim, obs, na.rm=TRUE, fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
            epsilon.value=NA)
```

Arguments

- `sim` numeric, zoo, matrix or data.frame with simulated values
- `obs` numeric, zoo, matrix or data.frame with observed values
- `na.rm` a logical value indicating whether 'NA' should be stripped before the computation proceeds.
  When an 'NA' value is found at the i-th position in `obs` **OR** `sim`, the i-th value of `obs` **AND** `sim` are removed before the computation.
- `fun` function to be applied to `sim` and `obs` in order to obtain transformed values thereof before computing the Nash-Sutcliffe efficiency.
  The first argument MUST BE a numeric vector with any name (e.g., `x`), and additional arguments are passed using `...`.
- `...` arguments passed to `fun`, in addition to the mandatory first numeric vector.
- `epsilon.type` argument used to define a numeric value to be added to both `sim` and `obs` before applying `FUN`.
  It is was designed to allow the use of logarithm and other similar functions that do not work with zero values.
  Valid values of `epsilon.type` are:
  1) `"none"`: `sim` and `obs` are used by `FUN` without the addition of any nummeric value.
  2) `"Pushpalatha2012"`: one hundredth (1/100) of the mean observed values is added to both `sim` and `obs` before applying `FUN`, as described in Pushpalatha et al. (2012).
  3) `"otherFactor"`: the numeric value defined in the `epsilon.value` argument is used to multiply the the mean observed values, instead of the one hundredth (1/100) described in Pushpalatha et al. (2012). The resulting value is then added to both `sim` and `obs`, before applying `FUN`.
  4) `"otherValue"`: the numeric value defined in the `epsilon.value` argument is directly added to both `sim` and `obs`, before applying `FUN`.
- `epsilon.value` -) when `epsilon.type="otherValue"` it represents the numeric value to be added to both `sim` and `obs` before applying `fun`.
  -) when `epsilon.type="otherFactor"` it represents the numeric factor used to multiply the mean of the observed values, instead of the one hundredth (1/100) described in Pushpalatha et al. (2012). The resulting value is then added to both `sim` and `obs` before applying `fun`.

Details

$$rd = 1 - \frac{\sum_{i=1}^{N} \left(\frac{O_i - S_i}{O_i} \right)^2}{\sum_{i=1}^{N} \left(\frac{\left| S_i - \bar{O} \right| + \left| O_i - \bar{O} \right|}{\bar{O}} \right)^2}$$

It varies between 0 and 1. A value of 1 indicates a perfect match, and 0 indicates no agreement at all.

Value

Relative index of agreement between `sim` and `obs`.

If `sim` and `obs` are matrixes, the returned value is a vector, with the relative index of agreement between each column of `sim` and `obs`.

Note

`obs` and `sim` has to have the same length/dimension

The missing values in `obs` and `sim` are removed before the computation proceeds, and only those positions with non-missing values in `obs` and `sim` are considered in the computation.

If some of the observed values are equal to zero (at least one of them), this index can not be computed.

Author(s)

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

References

Krause, P.; Boyle, D.P.; Base, F. (2005). Comparison of different efficiency criteria for hydrological model assessment, Advances in Geosciences, 5, 89-97. doi:10.5194/adgeo-5-89-2005.

Willmott, C.J. (1981). On the validation of models. Physical Geography, 2, 184–194. doi:10.1080/02723646.1981.10642213.

Willmott, C.J. (1984). On the evaluation of model performance in physical geography. Spatial Statistics and Models, G. L. Gaile and C. J. Willmott, eds., 443-460. doi:10.1007/978-94-017-3048-8_23.

Willmott, C.J.; Ackleson, S.G. Davis, R.E.; Feddema, J.J.; Klink, K.M.; Legates, D.R.; O'Donnell, J.; Rowe, C.M. (1985), Statistics for the Evaluation and Comparison of Models, J. Geophys. Res., 90(C5), 8995-9005. doi:10.1029/JC090iC05p08995.

Legates, D.R.; McCabe, G. J. Jr. (1999), Evaluating the Use of "Goodness-of-Fit" Measures in Hydrologic and Hydroclimatic Model Validation, Water Resour. Res., 35(1), 233-241. doi:10.1029/1998WR900018.

See Also

`d`, `md`, `dr`, `gof`, `ggof`

Examples

```
##################
# Example 1: basic ideal case
obs <- 1:10
sim <- 1:10
rd(sim, obs)

obs <- 1:10
```

```
sim <- 2:11
rd(sim, obs)

##################
# Example 2:
# Loading daily streamflows of the Ega River (Spain), from 1961 to 1970
data(EgaEnEstellaQts)
obs <- EgaEnEstellaQts

# Generating a simulated daily time series, initially equal to the observed series
sim <- obs

# Computing the 'rd' for the "best" (unattainable) case
rd(sim=sim, obs=obs)

##################
# Example 3: rd for simulated values equal to observations plus random noise
#            on the first half of the observed values.
#            This random noise has more relative importance for ow flows than
#            for medium and high flows.

# Randomly changing the first 1826 elements of 'sim', by using a normal distribution
# with mean 10 and standard deviation equal to 1 (default of 'rnorm').
sim[1:1826] <- obs[1:1826] + rnorm(1826, mean=10)
ggof(sim, obs)

rd(sim=sim, obs=obs)

##################
# Example 4: rd for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' during computations.

rd(sim=sim, obs=obs, fun=log)

# Verifying the previous value:
lsim <- log(sim)
lobs <- log(obs)
rd(sim=lsim, obs=lobs)

##################
# Example 5: rd for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and adding the Pushpalatha2012 constant
#            during computations

rd(sim=sim, obs=obs, fun=log, epsilon.type="Pushpalatha2012")

# Verifying the previous value, with the epsilon value following Pushpalatha2012
eps  <- mean(obs, na.rm=TRUE)/100
lsim <- log(sim+eps)
lobs <- log(obs+eps)
rd(sim=lsim, obs=lobs)
```

```
##################
# Example 6: rd for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and adding a user-defined constant
#            during computations

eps <- 0.01
rd(sim=sim, obs=obs, fun=log, epsilon.type="otherValue", epsilon.value=eps)

# Verifying the previous value:
lsim <- log(sim+eps)
lobs <- log(obs+eps)
rd(sim=lsim, obs=lobs)

##################
# Example 7: rd for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and using a user-defined factor
#            to multiply the mean of the observed values to obtain the constant
#            to be added to 'sim' and 'obs' during computations

fact <- 1/50
rd(sim=sim, obs=obs, fun=log, epsilon.type="otherFactor", epsilon.value=fact)

# Verifying the previous value:
eps  <- fact*mean(obs, na.rm=TRUE)
lsim <- log(sim+eps)
lobs <- log(obs+eps)
rd(sim=lsim, obs=lobs)

##################
# Example 8: rd for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying a
#            user-defined function to 'sim' and 'obs' during computations

fun1 <- function(x) {sqrt(x+1)}

rd(sim=sim, obs=obs, fun=fun1)

# Verifying the previous value, with the epsilon value following Pushpalatha2012
sim1 <- sqrt(sim+1)
obs1 <- sqrt(obs+1)
rd(sim=sim1, obs=obs1)
```

rfactor *R-factor*

Description

R-factor represents the average width of the given uncertainty bounds divided by the standard deviation of the observations.

Ideally, i.e., with a combination of model structure and parameter values that perfectly represents the catchment under study, and in absence of measurement errors and other additional sources of uncertainty, all the simulated values should be in a perfect match with the observations, leading to a *P-factor* equal to 1, and an *R-factor* equal to zero. However, in real-world applications we aim at encompassing as much observations as possible within the given uncertainty bounds (*P-factor* close to 1) while keeping the width of the uncertainty bounds as small as possible (*R-factor* close to 0), in order to avoid obtaining a good bracketing of observations at expense of uncertainty bounds too wide to be informative for the decision-making process.

Usage

```
rfactor(x, ...)

## Default S3 method:
rfactor(x, lband, uband, na.rm=TRUE, ...)

## S3 method for class 'data.frame'
rfactor(x, lband, uband, na.rm=TRUE, ...)

## S3 method for class 'matrix'
rfactor(x, lband, uband, na.rm=TRUE, ...)
```

Arguments

`x`	ts or zoo object with the observed values.
`lband`	numeric, ts or zoo object with the values of the lower uncertainty bound
`uband`	numeric, ts or zoo object with the values of the upper uncertainty bound
`na.rm`	logical value indicating whether 'NA' values should be stripped before the computation proceeds.
`...`	further arguments passed to or from other methods.

Value

Average width of the given uncertainty bounds, given by `lband` and `uband`, divided by the standard deviation of the observations `x`

If `sim` and `obs` are matrixes, the returned value is a vector, with the R-factor between each column of `sim` and `obs`.

Note

So far, the argument `na.rm` is not being taken into account.

Author(s)

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

References

Abbaspour, K.C.; Faramarzi, M.; Ghasemi, S.S.; Yang, H. (2009), Assessing the impact of climate change on water resources in Iran, Water Resources Research, 45(10), W10,434, doi:10.1029/2008WR007615.

Abbaspour, K.C., Yang, J. ; Maximov, I.; Siber, R.; Bogner, K.; Mieleitner, J. ; Zobrist, J.; Srinivasan, R. (2007), Modelling hydrology and water quality in the pre-alpine/alpine Thur watershed using SWAT, Journal of Hydrology, 333(2-4), 413-430, doi:10.1016/j.jhydrol.2006.09.014.

Schuol, J.; Abbaspour, K.C.; Srinivasan, R.; Yang, H. (2008b), Estimation of freshwater availability in the West African sub-continent using the SWAT hydrologic model, Journal of Hydrology, 352(1-2), 30, doi:10.1016/j.jhydrol.2007.12.025

Abbaspour, K.C. (2007), User manual for SWAT-CUP, SWAT calibration and uncertainty analysis programs, 93pp, Eawag: Swiss Fed. Inst. of Aquat. Sci. and Technol. Dubendorf, Switzerland.

See Also

pfactor, plotbands

Examples

```
x <- 1:10
lband <- x - 0.1
uband <- x + 0.1
rfactor(x, lband, uband)

lband <- x - rnorm(10)
uband <- x + rnorm(10)
rfactor(x, lband, uband)

#############
# Loading daily streamflows of the Ega River (Spain), from 1961 to 1970
data(EgaEnEstellaQts)
obs <- EgaEnEstellaQts

# Selecting only the daily values belonging to the year 1961
obs <- window(obs, end=as.Date("1961-12-31"))

# Generating the lower and upper uncertainty bounds, centred at the observations
lband <- obs - 5
uband <- obs + 5

rfactor(obs, lband, uband)

# Randomly generating the lower and upper uncertainty bounds
uband <- obs + rnorm(length(obs))
lband <- obs - rnorm(length(obs))

rfactor(obs, lband, uband)
```

rmse *Root Mean Square Error*

Description

Root Mean Square Error (RMSE) between sim and obs, in the same units of sim and obs, with treatment of missing values.
RMSE gives the standard deviation of the model prediction error. A smaller value indicates better model performance.

Usage

```
rmse(sim, obs, ...)

## Default S3 method:
rmse(sim, obs, na.rm=TRUE, fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
              epsilon.value=NA)

## S3 method for class 'data.frame'
rmse(sim, obs, na.rm=TRUE, fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
              epsilon.value=NA)

## S3 method for class 'matrix'
rmse(sim, obs, na.rm=TRUE, fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
              epsilon.value=NA)

## S3 method for class 'zoo'
rmse(sim, obs, na.rm=TRUE, fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
              epsilon.value=NA)
```

Arguments

sim numeric, zoo, matrix or data.frame with simulated values

obs numeric, zoo, matrix or data.frame with observed values

na.rm a logical value indicating whether 'NA' should be stripped before the computation proceeds.
When an 'NA' value is found at the i-th position in obs **OR** sim, the i-th value of obs **AND** sim are removed before the computation.

fun
: function to be applied to sim and obs in order to obtain transformed values thereof before computing the Root Mean Square Error.

  The first argument MUST BE a numeric vector with any name (e.g., x), and additional arguments are passed using ....

...
: arguments passed to fun, in addition to the mandatory first numeric vector.

epsilon.type
: argument used to define a numeric value to be added to both sim and obs before applying FUN.

  It is was designed to allow the use of logarithm and other similar functions that do not work with zero values.

  Valid values of epsilon.type are:

  1) "none": sim and obs are used by fun without the addition of any numeric value. This is the default option.

  2) "Pushpalatha2012": one hundredth (1/100) of the mean observed values is added to both sim and obs before applying fun, as described in Pushpalatha et al. (2012).

  3) "otherFactor": the numeric value defined in the epsilon.value argument is used to multiply the the mean observed values, instead of the one hundredth (1/100) described in Pushpalatha et al. (2012). The resulting value is then added to both sim and obs, before applying fun.

  4) "otherValue": the numeric value defined in the epsilon.value argument is directly added to both sim and obs, before applying fun.

epsilon.value
: -) when epsilon.type="otherValue" it represents the numeric value to be added to both sim and obs before applying fun.
  -) when epsilon.type="otherFactor" it represents the numeric factor used to multiply the mean of the observed values, instead of the one hundredth (1/100) described in Pushpalatha et al. (2012). The resulting value is then added to both sim and obs before applying fun.

Details

$$rmse = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(S_i - O_i)^2}$$

Value

Root mean square error (rmse) between sim and obs.

If sim and obs are matrixes, the returned value is a vector, with the RMSE between each column of sim and obs.

Note

obs and sim has to have the same length/dimension

The missing values in obs and sim are removed before the computation proceeds, and only those positions with non-missing values in obs and sim are considered in the computation

Author(s)

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

References

https://en.wikipedia.org/wiki/Root_mean_square_deviation

Willmott, C.J.; Matsuura, K. (2005). Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance, Climate Research, 30, 79-82, doi:10.3354/cr030079.

Chai, T.; Draxler, R.R. (2014). Root mean square error (RMSE) or mean absolute error (MAE)? - Arguments against avoiding RMSE in the literature, Geoscientific Model Development, 7, 1247-1250. doi:10.5194/gmd-7-1247-2014.

Hodson, T.O. (2022). Root-mean-square error (RMSE) or mean absolute error (MAE): when to use them or not, Geoscientific Model Development, 15, 5481-5487, doi:10.5194/gmd-15-5481-2022.

See Also

pbias, pbiasfdc, mae, mse, ubRMSE, nrmse, ssq, gof, ggof

Examples

```
##################
# Example 1: basic ideal case
obs <- 1:10
sim <- 1:10
rmse(sim, obs)

obs <- 1:10
sim <- 2:11
rmse(sim, obs)

##################
# Example 2:
# Loading daily streamflows of the Ega River (Spain), from 1961 to 1970
data(EgaEnEstellaQts)
obs <- EgaEnEstellaQts

# Generating a simulated daily time series, initially equal to the observed series
sim <- obs

# Computing the 'rmse' for the "best" (unattainable) case
rmse(sim=sim, obs=obs)

##################
# Example 3: rmse for simulated values equal to observations plus random noise
#            on the first half of the observed values.
#            This random noise has more relative importance for ow flows than
#            for medium and high flows.

# Randomly changing the first 1826 elements of 'sim', by using a normal distribution
```

```
# with mean 10 and standard deviation equal to 1 (default of 'rnorm').
sim[1:1826] <- obs[1:1826] + rnorm(1826, mean=10)
ggof(sim, obs)

rmse(sim=sim, obs=obs)

##################
# Example 4: rmse for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' during computations.

rmse(sim=sim, obs=obs, fun=log)

# Verifying the previous value:
lsim <- log(sim)
lobs <- log(obs)
rmse(sim=lsim, obs=lobs)

##################
# Example 5: rmse for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and adding the Pushpalatha2012 constant
#            during computations

rmse(sim=sim, obs=obs, fun=log, epsilon.type="Pushpalatha2012")

# Verifying the previous value, with the epsilon value following Pushpalatha2012
eps  <- mean(obs, na.rm=TRUE)/100
lsim <- log(sim+eps)
lobs <- log(obs+eps)
rmse(sim=lsim, obs=lobs)

##################
# Example 6: rmse for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and adding a user-defined constant
#            during computations

eps <- 0.01
rmse(sim=sim, obs=obs, fun=log, epsilon.type="otherValue", epsilon.value=eps)

# Verifying the previous value:
lsim <- log(sim+eps)
lobs <- log(obs+eps)
rmse(sim=lsim, obs=lobs)

##################
# Example 7: rmse for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and using a user-defined factor
#            to multiply the mean of the observed values to obtain the constant
#            to be added to 'sim' and 'obs' during computations
```

```
fact <- 1/50
rmse(sim=sim, obs=obs, fun=log, epsilon.type="otherFactor", epsilon.value=fact)

# Verifying the previous value:
eps  <- fact*mean(obs, na.rm=TRUE)
lsim <- log(sim+eps)
lobs <- log(obs+eps)
rmse(sim=lsim, obs=lobs)

##################
# Example 8: rmse for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying a
#            user-defined function to 'sim' and 'obs' during computations

fun1 <- function(x) {sqrt(x+1)}

rmse(sim=sim, obs=obs, fun=fun1)

# Verifying the previous value, with the epsilon value following Pushpalatha2012
sim1 <- sqrt(sim+1)
obs1 <- sqrt(obs+1)
rmse(sim=sim1, obs=obs1)
```

---

rNSE *Relative Nash-Sutcliffe efficiency*

Description

Relative Nash-Sutcliffe efficiency between `sim` and `obs`, with treatment of missing values.

Usage

```
rNSE(sim, obs, ...)

## Default S3 method:
rNSE(sim, obs, na.rm=TRUE, fun=NULL, ...,
          epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
               epsilon.value=NA)

## S3 method for class 'data.frame'
rNSE(sim, obs, na.rm=TRUE, fun=NULL, ...,
          epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
               epsilon.value=NA)

## S3 method for class 'matrix'
rNSE(sim, obs, na.rm=TRUE, fun=NULL, ...,
          epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
               epsilon.value=NA)
```

```
## S3 method for class 'zoo'
rNSE(sim, obs, na.rm=TRUE, fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
              epsilon.value=NA)
```

Arguments

`sim` numeric, zoo, matrix or data.frame with simulated values

`obs` numeric, zoo, matrix or data.frame with observed values

`na.rm` a logical value indicating whether 'NA' should be stripped before the computation proceeds.
When an 'NA' value is found at the i-th position in `obs` **OR** `sim`, the i-th value of `obs` **AND** `sim` are removed before the computation.

`fun` function to be applied to `sim` and `obs` in order to obtain transformed values thereof before computing the relative Nash-Sutcliffe efficiency.
The first argument MUST BE a numeric vector with any name (e.g., `x`), and additional arguments are passed using `...`.

`...` arguments passed to `fun`, in addition to the mandatory first numeric vector.

`epsilon.type` argument used to define a numeric value to be added to both `sim` and `obs` before applying `fun`.
It is was designed to allow the use of logarithm and other similar functions that do not work with zero values.
Valid values of `epsilon.type` are:
1) `"none"`: `sim` and `obs` are used by `fun` without the addition of any nummeric value.
2) `"Pushpalatha2012"`: one hundredth (1/100) of the mean observed values is added to both `sim` and `obs` before applying `fun`, as described in Pushpalatha et al. (2012).
3) `"otherFactor"`: the numeric value defined in the `epsilon.value` argument is used to multiply the the mean observed values, instead of the one hundredth (1/100) described in Pushpalatha et al. (2012). The resulting value is then added to both `sim` and `obs`, before applying `fun`.
4) `"otherValue"`: the numeric value defined in the `epsilon.value` argument is directly added to both `sim` and `obs`, before applying `fun`.

`epsilon.value` -) when `epsilon.type="otherValue"` it represents the numeric value to be added to both `sim` and `obs` before applying `fun`.
-) when `epsilon.type="otherFactor"` it represents the numeric factor used to multiply the mean of the observed values, instead of the one hundredth (1/100) described in Pushpalatha et al. (2012). The resulting value is then added to both `sim` and `obs` before applying `fun`.

Details

$$rNSE = 1 - \frac{\sum_{i=1}^{N} \left(\frac{S_i - O_i}{O_i} \right)^2}{\sum_{i=1}^{N} \left(\frac{O_i - \bar{O}}{\bar{O}} \right)^2}$$

Value

Relative Nash-Sutcliffe efficiency between sim and obs.

If sim and obs are matrixes, the returned value is a vector, with the relative Nash-Sutcliffe efficiency between each column of sim and obs.

Note

obs and sim has to have the same length/dimension

The missing values in obs and sim are removed before the computation proceeds, and only those positions with non-missing values in obs and sim are considered in the computation

If some of the observed values are equal to zero (at least one of them), this index can not be computed.

Author(s)

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

References

Krause, P.; Boyle, D.P.; Base, F. (2005). Comparison of different efficiency criteria for hydrological model assessment, Adv. Geosci., 5, 89-97. doi:10.5194/adgeo-5-89-2005.

Legates, D.R.; McCabe, G. J. Jr. (1999), Evaluating the Use of "Goodness-of-Fit" Measures in Hydrologic and Hydroclimatic Model Validation, Water Resour. Res., 35(1), 233-241. doi:10.1029/1998WR900018.

See Also

NSE, mNSE, wNSE, KGE, gof, ggof

Examples

```
##################
# Example 1: basic ideal case
obs <- 1:10
sim <- 1:10
rNSE(sim, obs)

obs <- 1:10
sim <- 2:11
rNSE(sim, obs)

##################
# Example 2:
# Loading daily streamflows of the Ega River (Spain), from 1961 to 1970
data(EgaEnEstellaQts)
obs <- EgaEnEstellaQts
```

```
# Generating a simulated daily time series, initially equal to the observed series
sim <- obs

# Computing the 'rNSE' for the "best" (unattainable) case
rNSE(sim=sim, obs=obs)

##################
# Example 3: rNSE for simulated values equal to observations plus random noise
#            on the first half of the observed values.
#            This random noise has more relative importance for ow flows than
#            for medium and high flows.

# Randomly changing the first 1826 elements of 'sim', by using a normal distribution
# with mean 10 and standard deviation equal to 1 (default of 'rnorm').
sim[1:1826] <- obs[1:1826] + rnorm(1826, mean=10)
ggof(sim, obs)

rNSE(sim=sim, obs=obs)

##################
# Example 4: rNSE for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' during computations.

rNSE(sim=sim, obs=obs, fun=log)

# Verifying the previous value:
lsim <- log(sim)
lobs <- log(obs)
rNSE(sim=lsim, obs=lobs)

##################
# Example 5: rNSE for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and adding the Pushpalatha2012 constant
#            during computations

rNSE(sim=sim, obs=obs, fun=log, epsilon.type="Pushpalatha2012")

# Verifying the previous value, with the epsilon value following Pushpalatha2012
eps  <- mean(obs, na.rm=TRUE)/100
lsim <- log(sim+eps)
lobs <- log(obs+eps)
rNSE(sim=lsim, obs=lobs)

##################
# Example 6: rNSE for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and adding a user-defined constant
#            during computations

eps <- 0.01
```

```
rNSE(sim=sim, obs=obs, fun=log, epsilon.type="otherValue", epsilon.value=eps)

# Verifying the previous value:
lsim <- log(sim+eps)
lobs <- log(obs+eps)
rNSE(sim=lsim, obs=lobs)

##################
# Example 7: rNSE for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and using a user-defined factor
#            to multiply the mean of the observed values to obtain the constant
#            to be added to 'sim' and 'obs' during computations

fact <- 1/50
rNSE(sim=sim, obs=obs, fun=log, epsilon.type="otherFactor", epsilon.value=fact)

# Verifying the previous value:
eps  <- fact*mean(obs, na.rm=TRUE)
lsim <- log(sim+eps)
lobs <- log(obs+eps)
rNSE(sim=lsim, obs=lobs)

##################
# Example 8: rNSE for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying a
#            user-defined function to 'sim' and 'obs' during computations

fun1 <- function(x) {sqrt(x+1)}

rNSE(sim=sim, obs=obs, fun=fun1)

# Verifying the previous value, with the epsilon value following Pushpalatha2012
sim1 <- sqrt(sim+1)
obs1 <- sqrt(obs+1)
rNSE(sim=sim1, obs=obs1)
```

---

rPearson *Pearson correlation coefficient*

Description

Pearson correlation coefficient between `sim` and `obs`, with treatment of missing values.

Usage

```
rPearson(sim, obs, ...)

## Default S3 method:
rPearson(sim, obs, fun=NULL, ...,
```

```
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
             epsilon.value=NA)

## S3 method for class 'matrix'
rPearson(sim, obs, na.rm=TRUE, fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
             epsilon.value=NA)

## S3 method for class 'data.frame'
rPearson(sim, obs, na.rm=TRUE, fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
             epsilon.value=NA)

## S3 method for class 'zoo'
rPearson(sim, obs, na.rm=TRUE, fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
             epsilon.value=NA)
```

Arguments

- `sim` numeric, zoo, matrix or data.frame with simulated values
- `obs` numeric, zoo, matrix or data.frame with observed values
- `na.rm` a logical value indicating whether 'NA' should be stripped before the computation proceeds.

  When an 'NA' value is found at the i-th position in `obs` **OR** `sim`, the i-th value of `obs` **AND** `sim` are removed before the computation.
- `fun` function to be applied to `sim` and `obs` in order to obtain transformed values thereof before computing this goodness-of-fit index.

  The first argument MUST BE a numeric vector with any name (e.g., `x`), and additional arguments are passed using `...`.
- `...` arguments passed to `fun`, in addition to the mandatory first numeric vector.
- `epsilon.type` argument used to define a numeric value to be added to both `sim` and `obs` before applying `fun`.

  It is was designed to allow the use of logarithm and other similar functions that do not work with zero values.

  Valid values of `epsilon.type` are:

  1) `"none"`: `sim` and `obs` are used by `fun` without the addition of any numeric value. This is the default option.

  2) `"Pushpalatha2012"`: one hundredth (1/100) of the mean observed values is added to both `sim` and `obs` before applying `fun`, as described in Pushpalatha et al. (2012).

  3) `"otherFactor"`: the numeric value defined in the `epsilon.value` argument is used to multiply the the mean observed values, instead of the one hundredth (1/100) described in Pushpalatha et al. (2012). The resulting value is then added to both `sim` and `obs`, before applying `fun`.

  4) `"otherValue"`: the numeric value defined in the `epsilon.value` argument is directly added to both `sim` and `obs`, before applying `fun`.

epsilon.value numeric value to be added to both sim and obs when epsilon.type="otherValue".

Details

It is a wrapper to the cor function.

The Pearson correlation coefficient (PCC) is a correlation coefficient that measures linear correlation between two sets of data.

It is the ratio between the covariance of two variables and the product of their standard deviations; thus, it is essentially a normalized measurement of the covariance, such that the result always has a value between -1 and 1. As with covariance itself, the measure can only reflect a linear correlation of variables, and ignores many other types of relationships or correlations.

The correlation coefficient ranges from -1 to 1. An absolute value of exactly 1 implies that a linear equation describes the relationship between sim and obs perfectly, with all data points lying on a line. The correlation sign is determined by the regression slope: a value of +1 implies that all data points lie on a line for which sim increases as obs increases, and vice versa for -1. A value of 0 implies that there is no linear dependency between the variables.

Value

Pearson correlation coefficient between sim and obs.

If sim and obs are matrixes, the returned value is a vector, with the Pearson correlation coefficient between each column of sim and obs.

Note

obs and sim has to have the same length/dimension

The missing values in obs and sim are removed before the computation proceeds, and only those positions with non-missing values in obs and sim are considered in the computation

Author(s)

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

References

https://en.wikipedia.org/wiki/Pearson_correlation_coefficient

Pearson, K. (1920). Notes on the history of correlation. Biometrika, 13(1), 25-45. doi:10.2307/2331722.

Schober, P.; Boer, C.; Schwarte, L.A. (2018). Correlation coefficients: appropriate use and interpretation. Anesthesia and Analgesia, 126(5), 1763-1768. doi:10.1213/ANE.0000000000002864.

See Also

cor

Examples

```
##################
# Example 1: basic ideal case
obs <- 1:10
sim <- 1:10
rPearson(sim, obs)

obs <- 1:10
sim <- 2:11
rPearson(sim, obs)

##################
# Example 2:
# Loading daily streamflows of the Ega River (Spain), from 1961 to 1970
data(EgaEnEstellaQts)
obs <- EgaEnEstellaQts

# Generating a simulated daily time series, initially equal to the observed series
sim <- obs

# Computing the 'rPearson' for the "best" (unattainable) case
rPearson(sim=sim, obs=obs)

##################
# Example 3: rPearson for simulated values equal to observations plus random noise
#            on the first half of the observed values.
#            This random noise has more relative importance for ow flows than
#            for medium and high flows.

# Randomly changing the first 1826 elements of 'sim', by using a normal distribution
# with mean 10 and standard deviation equal to 1 (default of 'rnorm').
sim[1:1826] <- obs[1:1826] + rnorm(1826, mean=10)
ggof(sim, obs)

rPearson(sim=sim, obs=obs)

##################
# Example 4: rPearson for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' during computations.

rPearson(sim=sim, obs=obs, fun=log)

# Verifying the previous value:
lsim <- log(sim)
lobs <- log(obs)
rPearson(sim=lsim, obs=lobs)

##################
# Example 5: rPearson for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and adding the Pushpalatha2012 constant
```

```
#            during computations

rPearson(sim=sim, obs=obs, fun=log, epsilon.type="Pushpalatha2012")

# Verifying the previous value, with the epsilon value following Pushpalatha2012
eps  <- mean(obs, na.rm=TRUE)/100
lsim <- log(sim+eps)
lobs <- log(obs+eps)
rPearson(sim=lsim, obs=lobs)

##################
# Example 6: rPearson for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and adding a user-defined constant
#            during computations

eps <- 0.01
rPearson(sim=sim, obs=obs, fun=log, epsilon.type="otherValue", epsilon.value=eps)

# Verifying the previous value:
lsim <- log(sim+eps)
lobs <- log(obs+eps)
rPearson(sim=lsim, obs=lobs)

##################
# Example 7: rPearson for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and using a user-defined factor
#            to multiply the mean of the observed values to obtain the constant
#            to be added to 'sim' and 'obs' during computations

fact <- 1/50
rPearson(sim=sim, obs=obs, fun=log, epsilon.type="otherFactor", epsilon.value=fact)

# Verifying the previous value:
eps  <- fact*mean(obs, na.rm=TRUE)
lsim <- log(sim+eps)
lobs <- log(obs+eps)
rPearson(sim=lsim, obs=lobs)

##################
# Example 8: rPearson for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying a
#            user-defined function to 'sim' and 'obs' during computations

fun1 <- function(x) {sqrt(x+1)}

rPearson(sim=sim, obs=obs, fun=fun1)

# Verifying the previous value, with the epsilon value following Pushpalatha2012
sim1 <- sqrt(sim+1)
obs1 <- sqrt(obs+1)
rPearson(sim=sim1, obs=obs1)
```

rSD *Ratio of Standard Deviations*

Description

Ratio of standard deviations between sim and obs, with treatment of missing values.

Usage

```
rSD(sim, obs, ...)

## Default S3 method:
rSD(sim, obs, na.rm=TRUE, fun=NULL, ...,
        epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
            epsilon.value=NA)

## S3 method for class 'data.frame'
rSD(sim, obs, na.rm=TRUE, fun=NULL, ...,
        epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
            epsilon.value=NA)

## S3 method for class 'matrix'
rSD(sim, obs, na.rm=TRUE, fun=NULL, ...,
        epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
            epsilon.value=NA)

## S3 method for class 'zoo'
rSD(sim, obs, na.rm=TRUE, fun=NULL, ...,
        epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
            epsilon.value=NA)
```

Arguments

sim	numeric, zoo, matrix or data.frame with simulated values
obs	numeric, zoo, matrix or data.frame with observed values
na.rm	a logical value indicating whether 'NA' should be stripped before the computation proceeds. When an 'NA' value is found at the i-th position in obs **OR** sim, the i-th value of obs **AND** sim are removed before the computation.
fun	function to be applied to sim and obs in order to obtain transformed values thereof before computing this goodness-of-fit index. The first argument MUST BE a numeric vector with any name (e.g., x), and additional arguments are passed using ....
...	arguments passed to fun, in addition to the mandatory first numeric vector.

epsilon.type
: argument used to define a numeric value to be added to both sim and obs before applying fun.

  It is was designed to allow the use of logarithm and other similar functions that do not work with zero values.

  Valid values of epsilon.type are:

  1) "none": sim and obs are used by fun without the addition of any numeric value. This is the default option.

  2) "Pushpalatha2012": one hundredth (1/100) of the mean observed values is added to both sim and obs before applying fun, as described in Pushpalatha et al. (2012).

  3) "otherFactor": the numeric value defined in the epsilon.value argument is used to multiply the the mean observed values, instead of the one hundredth (1/100) described in Pushpalatha et al. (2012). The resulting value is then added to both sim and obs, before applying fun.

  4) "otherValue": the numeric value defined in the epsilon.value argument is directly added to both sim and obs, before applying fun.

epsilon.value
: -) when epsilon.type="otherValue" it represents the numeric value to be added to both sim and obs before applying fun.
  -) when epsilon.type="otherFactor" it represents the numeric factor used to multiply the mean of the observed values, instead of the one hundredth (1/100) described in Pushpalatha et al. (2012). The resulting value is then added to both sim and obs before applying fun.

Details

$$r_{SD} = \frac{sd_{sim}}{sd_{obs}}$$

Value

Ratio of standard deviations between sim and obs.

If sim and obs are matrixes, the returned value is a vector, with the ratio of standard deviations between each column of sim and obs.

Note

obs and sim has to have the same length/dimension

The missing values in obs and sim are removed before the computation proceeds, and only those positions with non-missing values in obs and sim are considered in the computation

Author(s)

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

See Also

sd, rsr, gof, ggof

Examples

```
##################
# Example 1: basic ideal case
obs <- 1:10
sim <- 1:10
rSD(sim, obs)

obs <- 1:10
sim <- 2:11
rSD(sim, obs)

##################
# Example 2:
# Loading daily streamflows of the Ega River (Spain), from 1961 to 1970
data(EgaEnEstellaQts)
obs <- EgaEnEstellaQts

# Generating a simulated daily time series, initially equal to the observed series
sim <- obs

# Computing the 'rSD' for the "best" (unattainable) case
rSD(sim=sim, obs=obs)

##################
# Example 3: rSD for simulated values equal to observations plus random noise
#            on the first half of the observed values.
#            This random noise has more relative importance for ow flows than
#            for medium and high flows.

# Randomly changing the first 1826 elements of 'sim', by using a normal distribution
# with mean 10 and standard deviation equal to 1 (default of 'rnorm').
sim[1:1826] <- obs[1:1826] + rnorm(1826, mean=10)
ggof(sim, obs)

rSD(sim=sim, obs=obs)

##################
# Example 4: rSD for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' during computations.

rSD(sim=sim, obs=obs, fun=log)

# Verifying the previous value:
lsim <- log(sim)
lobs <- log(obs)
rSD(sim=lsim, obs=lobs)
```

```
##################
# Example 5: rSD for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and adding the Pushpalatha2012 constant
#            during computations

rSD(sim=sim, obs=obs, fun=log, epsilon.type="Pushpalatha2012")

# Verifying the previous value, with the epsilon value following Pushpalatha2012
eps  <- mean(obs, na.rm=TRUE)/100
lsim <- log(sim+eps)
lobs <- log(obs+eps)
rSD(sim=lsim, obs=lobs)

##################
# Example 6: rSD for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and adding a user-defined constant
#            during computations

eps <- 0.01
rSD(sim=sim, obs=obs, fun=log, epsilon.type="otherValue", epsilon.value=eps)

# Verifying the previous value:
lsim <- log(sim+eps)
lobs <- log(obs+eps)
rSD(sim=lsim, obs=lobs)

##################
# Example 7: rSD for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and using a user-defined factor
#            to multiply the mean of the observed values to obtain the constant
#            to be added to 'sim' and 'obs' during computations

fact <- 1/50
rSD(sim=sim, obs=obs, fun=log, epsilon.type="otherFactor", epsilon.value=fact)

# Verifying the previous value:
eps  <- fact*mean(obs, na.rm=TRUE)
lsim <- log(sim+eps)
lobs <- log(obs+eps)
rSD(sim=lsim, obs=lobs)

##################
# Example 8: rSD for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying a
#            user-defined function to 'sim' and 'obs' during computations

fun1 <- function(x) {sqrt(x+1)}

rSD(sim=sim, obs=obs, fun=fun1)
```

```
# Verifying the previous value, with the epsilon value following Pushpalatha2012
sim1 <- sqrt(sim+1)
obs1 <- sqrt(obs+1)
rSD(sim=sim1, obs=obs1)
```

---

rSpearman *Spearman's rank correlation coefficient*

Description

Spearman's rank correlation coefficient between `sim` and `obs`, with treatment of missing values.

Usage

```
rSpearman(sim, obs, ...)

## Default S3 method:
rSpearman(sim, obs, fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
             epsilon.value=NA)

## S3 method for class 'matrix'
rSpearman(sim, obs, na.rm=TRUE, fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
             epsilon.value=NA)

## S3 method for class 'data.frame'
rSpearman(sim, obs, na.rm=TRUE, fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
             epsilon.value=NA)

## S3 method for class 'zoo'
rSpearman(sim, obs, na.rm=TRUE, fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
             epsilon.value=NA)
```

Arguments

`sim`	numeric, zoo, matrix or data.frame with simulated values
`obs`	numeric, zoo, matrix or data.frame with observed values
`na.rm`	a logical value indicating whether 'NA' should be stripped before the computation proceeds. When an 'NA' value is found at the i-th position in `obs` **OR** `sim`, the i-th value of `obs` **AND** `sim` are removed before the computation.
`fun`	function to be applied to `sim` and `obs` in order to obtain transformed values thereof before computing this goodness-of-fit index. The first argument MUST BE a numeric vector with any name (e.g., `x`), and additional arguments are passed using `...`.

...	arguments passed to fun, in addition to the mandatory first numeric vector.
epsilon.type	argument used to define a numeric value to be added to both sim and obs before applying fun. It is was designed to allow the use of logarithm and other similar functions that do not work with zero values. Valid values of epsilon.type are: 1) "none": sim and obs are used by fun without the addition of any numeric value. This is the default option. 2) "Pushpalatha2012": one hundredth (1/100) of the mean observed values is added to both sim and obs before applying fun, as described in Pushpalatha et al. (2012). 3) "otherFactor": the numeric value defined in the epsilon.value argument is used to multiply the the mean observed values, instead of the one hundredth (1/100) described in Pushpalatha et al. (2012). The resulting value is then added to both sim and obs, before applying fun. 4) "otherValue": the numeric value defined in the epsilon.value argument is directly added to both sim and obs, before applying fun.
epsilon.value	numeric value to be added to both sim and obs when epsilon.type="otherValue".

Details

It is a wrapper to the cor function.

The Spearman's rank correlation coefficient is a nonparametric measure of rank correlation (statistical dependence between the rankings of two variables).

It assesses how well the relationship between two variables can be described using a monotonic function.

The Spearman correlation between two variables is equal to the Pearson correlation between the rank values of those two variables. However, while Pearson's correlation assesses linear relationships, Spearman's correlation assesses monotonic relationships (whether linear or not).

If there are no repeated data values, a perfect Spearman correlation of +1 or -1 occurs when each of the variables is a perfect monotone function of the other.

Value

Spearman's rank correlation coefficient between sim and obs.

If sim and obs are matrixes, the returned value is a vector, with the Spearman's rank correlation coefficient between each column of sim and obs.

Note

obs and sim has to have the same length/dimension

The missing values in obs and sim are removed before the computation proceeds, and only those positions with non-missing values in obs and sim are considered in the computation

Author(s)

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

References

https://en.wikipedia.org/wiki/Spearman%27s_rank_correlation_coefficient

Spearman, C. (1961). The Proof and Measurement of Association Between Two Things. In J. J. Jenkins and D. G. Paterson (Eds.), Studies in individual differences: The search for intelligence (pp. 45-58). Appleton-Century-Crofts. doi:10.1037/11491-005

See Also

cor

Examples

```
##################
# Example 1: basic ideal case
obs <- 1:10
sim <- 1:10
rSpearman(sim, obs)

obs <- 1:10
sim <- 2:11
rSpearman(sim, obs)

##################
# Example 2:
# Loading daily streamflows of the Ega River (Spain), from 1961 to 1970
data(EgaEnEstellaQts)
obs <- EgaEnEstellaQts

# Generating a simulated daily time series, initially equal to the observed series
sim <- obs

# Computing the 'rSpearman' for the "best" (unattainable) case
rSpearman(sim=sim, obs=obs)

##################
# Example 3: rSpearman for simulated values equal to observations plus random noise
#            on the first half of the observed values.
#            This random noise has more relative importance for ow flows than
#            for medium and high flows.

# Randomly changing the first 1826 elements of 'sim', by using a normal distribution
# with mean 10 and standard deviation equal to 1 (default of 'rnorm').
sim[1:1826] <- obs[1:1826] + rnorm(1826, mean=10)
```

```
ggof(sim, obs)

rSpearman(sim=sim, obs=obs)

##################
# Example 4: rSpearman for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' during computations.

rSpearman(sim=sim, obs=obs, fun=log)

# Verifying the previous value:
lsim <- log(sim)
lobs <- log(obs)
rSpearman(sim=lsim, obs=lobs)

##################
# Example 5: rSpearman for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and adding the Pushpalatha2012 constant
#            during computations

rSpearman(sim=sim, obs=obs, fun=log, epsilon.type="Pushpalatha2012")

# Verifying the previous value, with the epsilon value following Pushpalatha2012
eps  <- mean(obs, na.rm=TRUE)/100
lsim <- log(sim+eps)
lobs <- log(obs+eps)
rSpearman(sim=lsim, obs=lobs)

##################
# Example 6: rSpearman for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and adding a user-defined constant
#            during computations

eps <- 0.01
rSpearman(sim=sim, obs=obs, fun=log, epsilon.type="otherValue", epsilon.value=eps)

# Verifying the previous value:
lsim <- log(sim+eps)
lobs <- log(obs+eps)
rSpearman(sim=lsim, obs=lobs)

##################
# Example 7: rSpearman for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and using a user-defined factor
#            to multiply the mean of the observed values to obtain the constant
#            to be added to 'sim' and 'obs' during computations

fact <- 1/50
rSpearman(sim=sim, obs=obs, fun=log, epsilon.type="otherFactor", epsilon.value=fact)
```

```
# Verifying the previous value:
eps  <- fact*mean(obs, na.rm=TRUE)
lsim <- log(sim+eps)
lobs <- log(obs+eps)
rSpearman(sim=lsim, obs=lobs)

##################
# Example 8: rSpearman for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying a
#            user-defined function to 'sim' and 'obs' during computations

fun1 <- function(x) {sqrt(x+1)}

rSpearman(sim=sim, obs=obs, fun=fun1)

# Verifying the previous value, with the epsilon value following Pushpalatha2012
sim1 <- sqrt(sim+1)
obs1 <- sqrt(obs+1)
rSpearman(sim=sim1, obs=obs1)
```

---

rsr *Ratio of RMSE to the standard deviation of the observations*

Description

Ratio of the RMSE between simulated and observed values to the standard deviation of the observations.

Usage

```
rsr(sim, obs, ...)

## Default S3 method:
rsr(sim, obs, na.rm=TRUE, fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
             epsilon.value=NA)

## S3 method for class 'data.frame'
rsr(sim, obs, na.rm=TRUE, fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
             epsilon.value=NA)

## S3 method for class 'matrix'
rsr(sim, obs, na.rm=TRUE, fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
             epsilon.value=NA)

## S3 method for class 'zoo'
```

```
rsr(sim, obs, na.rm=TRUE, fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
             epsilon.value=NA)
```

Arguments

`sim`	numeric, zoo, matrix or data.frame with simulated values
`obs`	numeric, zoo, matrix or data.frame with observed values
`na.rm`	a logical value indicating whether 'NA' should be stripped before the computation proceeds. When an 'NA' value is found at the i-th position in `obs` **OR** `sim`, the i-th value of `obs` **AND** `sim` are removed before the computation.
`fun`	function to be applied to `sim` and `obs` in order to obtain transformed values thereof before computing this goodness-of-fit index. The first argument MUST BE a numeric vector with any name (e.g., `x`), and additional arguments are passed using `...`.
`...`	arguments passed to `fun`, in addition to the mandatory first numeric vector.
`epsilon.type`	argument used to define a numeric value to be added to both `sim` and `obs` before applying `fun`. It is was designed to allow the use of logarithm and other similar functions that do not work with zero values. Valid values of `epsilon.type` are: 1) `"none"`: `sim` and `obs` are used by `fun` without the addition of any numeric value. This is the default option. 2) `"Pushpalatha2012"`: one hundredth (1/100) of the mean observed values is added to both `sim` and `obs` before applying `fun`, as described in Pushpalatha et al. (2012). 3) `"otherFactor"`: the numeric value defined in the `epsilon.value` argument is used to multiply the the mean observed values, instead of the one hundredth (1/100) described in Pushpalatha et al. (2012). The resulting value is then added to both `sim` and `obs`, before applying `fun`. 4) `"otherValue"`: the numeric value defined in the `epsilon.value` argument is directly added to both `sim` and `obs`, before applying `fun`.
`epsilon.value`	-) when `epsilon.type="otherValue"` it represents the numeric value to be added to both `sim` and `obs` before applying `fun`. -) when `epsilon.type="otherFactor"` it represents the numeric factor used to multiply the mean of the observed values, instead of the one hundredth (1/100) described in Pushpalatha et al. (2012). The resulting value is then added to both `sim` and `obs` before applying `fun`.

Value

Ratio of RMSE to the standard deviation of the observations.

If `sim` and `obs` are matrixes, the returned value is a vector, with the RSR between each column of `sim` and `obs`.

Note

obs and sim has to have the same length/dimension

The missing values in obs and sim are removed before the computation proceeds, and only those positions with non-missing values in obs and sim are considered in the computation

Author(s)

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

References

Moriasi, D.N.; Arnold, J.G.; van Liew, M.W.; Bingner, R.L.; Harmel, R.D.; Veith, T.L. (2007). Model evaluation guidelines for systematic quantification of accuracy in watershed simulations. Transactions of the ASABE. 50(3):885-900

See Also

sd, rSD, gof, ggof

Examples

```
##################
# Example 1: basic ideal case
obs <- 1:10
sim <- 1:10
rsr(sim, obs)

obs <- 1:10
sim <- 2:11
rsr(sim, obs)

##################
# Example 2:
# Loading daily streamflows of the Ega River (Spain), from 1961 to 1970
data(EgaEnEstellaQts)
obs <- EgaEnEstellaQts

# Generating a simulated daily time series, initially equal to the observed series
sim <- obs

# Computing the 'rsr' for the "best" (unattainable) case
rsr(sim=sim, obs=obs)

##################
# Example 3: rsr for simulated values equal to observations plus random noise
#            on the first half of the observed values.
#            This random noise has more relative importance for ow flows than
#            for medium and high flows.

# Randomly changing the first 1826 elements of 'sim', by using a normal distribution
```

```
# with mean 10 and standard deviation equal to 1 (default of 'rnorm').
sim[1:1826] <- obs[1:1826] + rnorm(1826, mean=10)
ggof(sim, obs)

rsr(sim=sim, obs=obs)

##################
# Example 4: rsr for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' during computations.

rsr(sim=sim, obs=obs, fun=log)

# Verifying the previous value:
lsim <- log(sim)
lobs <- log(obs)
rsr(sim=lsim, obs=lobs)

##################
# Example 5: rsr for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and adding the Pushpalatha2012 constant
#            during computations

rsr(sim=sim, obs=obs, fun=log, epsilon.type="Pushpalatha2012")

# Verifying the previous value, with the epsilon value following Pushpalatha2012
eps  <- mean(obs, na.rm=TRUE)/100
lsim <- log(sim+eps)
lobs <- log(obs+eps)
rsr(sim=lsim, obs=lobs)

##################
# Example 6: rsr for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and adding a user-defined constant
#            during computations

eps <- 0.01
rsr(sim=sim, obs=obs, fun=log, epsilon.type="otherValue", epsilon.value=eps)

# Verifying the previous value:
lsim <- log(sim+eps)
lobs <- log(obs+eps)
rsr(sim=lsim, obs=lobs)

##################
# Example 7: rsr for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and using a user-defined factor
#            to multiply the mean of the observed values to obtain the constant
#            to be added to 'sim' and 'obs' during computations
```

```
fact <- 1/50
rsr(sim=sim, obs=obs, fun=log, epsilon.type="otherFactor", epsilon.value=fact)

# Verifying the previous value:
eps  <- fact*mean(obs, na.rm=TRUE)
lsim <- log(sim+eps)
lobs <- log(obs+eps)
rsr(sim=lsim, obs=lobs)

##################
# Example 8: rsr for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying a
#            user-defined function to 'sim' and 'obs' during computations

fun1 <- function(x) {sqrt(x+1)}

rsr(sim=sim, obs=obs, fun=fun1)

# Verifying the previous value, with the epsilon value following Pushpalatha2012
sim1 <- sqrt(sim+1)
obs1 <- sqrt(obs+1)
rsr(sim=sim1, obs=obs1)
```

---

sKGE *Split Kling-Gupta Efficiency*

Description

Split Kling-Gupta efficiency between sim and obs.

This goodness-of-fit measure was developed by Fowler et al. (2018), as a modification to the original Kling-Gupta efficiency (KGE) proposed by Gupta et al. (2009). See Details.

Usage

```
sKGE(sim, obs, ...)

## Default S3 method:
sKGE(sim, obs, s=c(1,1,1), na.rm=TRUE, method=c("2009", "2012", "2021"),
              start.month=1, out.PerYear=FALSE, fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
              epsilon.value=NA)

## S3 method for class 'data.frame'
sKGE(sim, obs, s=c(1,1,1), na.rm=TRUE, method=c("2009", "2012", "2021"),
              start.month=1, out.PerYear=FALSE, fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
              epsilon.value=NA)
```

```
## S3 method for class 'matrix'
sKGE(sim, obs, s=c(1,1,1), na.rm=TRUE, method=c("2009", "2012", "2021"),
             start.month=1, out.PerYear=FALSE, fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
             epsilon.value=NA)

## S3 method for class 'zoo'
sKGE(sim, obs, s=c(1,1,1), na.rm=TRUE, method=c("2009", "2012", "2021"),
             start.month=1, out.PerYear=FALSE, fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
             epsilon.value=NA)
```

Arguments

`sim` numeric, zoo, matrix or data.frame with simulated values

`obs` numeric, zoo, matrix or data.frame with observed values

`s` numeric of length 3, representing the scaling factors to be used for re-scaling the criteria space before computing the Euclidean distance from the ideal point c(1,1,1), i.e., `s` elements are used for adjusting the emphasis on different components. The first elements is used for rescaling the Pearson product-moment correlation coefficient (`r`), the second element is used for rescaling `Alpha` and the third element is used for re-scaling `Beta`

`na.rm` a logical value indicating whether 'NA' should be stripped before the computation proceeds.
When an 'NA' value is found at the i-th position in `obs` **OR** `sim`, the i-th value of `obs` **AND** `sim` are removed before the computation.

`method` character, indicating the formula used to compute the variability ratio in the Kling-Gupta efficiency. Valid values are:

-) `2009`: the variability is defined as 'Alpha', the ratio of the standard deviation of `sim` values to the standard deviation of `obs`. This is the default option. See Gupta et al. (2009).

-) `2012`: the variability is defined as 'Gamma', the ratio of the coefficient of variation of `sim` values to the coefficient of variation of `obs`. See Kling et al. (2012).

-) `2021`: the bias is defined as 'Beta', the ratio of `mean(sim)` minus `mean(obs)` to the standard deviation of `obs`. The variability is defined as 'Alpha', the ratio of the standard deviation of `sim` values to the standard deviation of `obs`. See Tang et al. (2021).

`start.month` [OPTIONAL]. Only used when the (hydrological) year of interest is different from the calendar year.

numeric in [1:12] indicating the starting month of the (hydrological) year. Numeric values in [1, 12] represent months in [January, December]. By default `start.month=1`.

`out.PerYear` logical, indicating whether the output of this function has to include the Kling-Gupta efficiencies obtained for the individual years in `sim` and `obs` or not.

fun function to be applied to `sim` and `obs` in order to obtain transformed values thereof before computing this goodness-of-fit index.

The first argument MUST BE a numeric vector with any name (e.g., `x`), and additional arguments are passed using `...`.

... arguments passed to `fun`, in addition to the mandatory first numeric vector.

epsilon.type argument used to define a numeric value to be added to both `sim` and `obs` before applying `fun`.

It is was designed to allow the use of logarithm and other similar functions that do not work with zero values.

Valid values of `epsilon.type` are:

1) `"none"`: `sim` and `obs` are used by `fun` without the addition of any numeric value. This is the default option.

2) `"Pushpalatha2012"`: one hundredth (1/100) of the mean observed values is added to both `sim` and `obs` before applying `fun`, as described in Pushpalatha et al. (2012).

3) `"otherFactor"`: the numeric value defined in the `epsilon.value` argument is used to multiply the the mean observed values, instead of the one hundredth (1/100) described in Pushpalatha et al. (2012). The resulting value is then added to both `sim` and `obs`, before applying `fun`.

4) `"otherValue"`: the numeric value defined in the `epsilon.value` argument is directly added to both `sim` and `obs`, before applying `fun`.

epsilon.value -) when `epsilon.type="otherValue"` it represents the numeric value to be added to both `sim` and `obs` before applying `fun`.
-) when `epsilon.type="otherFactor"` it represents the numeric factor used to multiply the mean of the observed values, instead of the one hundredth (1/100) described in Pushpalatha et al. (2012). The resulting value is then added to both `sim` and `obs` before applying `fun`.

Details

Garcia et al. (2017) tested different objective functions and found that the mean value of the KGE applied to the streamflows (i.e., KGE(Q)) and the KGE applied to the inverse of the streamflows (i.e., KGE(1/Q) is able to provide a an aceptable representation of low-flow indices important for water management. They also found that KGE applied to a transformation of streamflow values (e.g., log) is inadequate to capture low-flow indices important for water management.

The robustness of their findings depends more on the climate variability rather than the objective function, and they are insensitive to the hydrological model used in the evaluation.

Traditional Kling-Gupta efficiencies (Gupta et al., 2009; Kling et al., 2012) range from -Inf to 1 and, therefore, sKGE should also range from -Inf to 1. Essentially, the closer to 1, the more similar `sim` and `obs` are.

Knoben et al. (2019) showed that traditional Kling-Gupta (Gupta et al., 2009; Kling et al., 2012) values greater than -0.41 indicate that a model improves upon the mean flow benchmark, even if the model's KGE value is negative.

Value

If `out.PerYear=FALSE`: numeric with the Split Kling-Gupta efficiency between `sim` and `obs`. If `sim` and `obs` are matrices, the output value is a vector, with the Split Kling-Gupta efficiency between each column of `sim` and `obs`

If `out.PerYear=TRUE`: a list of two elements:

`sKGE.value` numeric with the Split Kling-Gupta efficiency. If `sim` and `obs` are matrices, the output value is a vector, with the Split Kling-Gupta efficiency between each column of `sim` and `obs`

`KGE.PerYear` numeric with the Kling-Gupta efficincies obtained for the individual years in `sim` and `obs`.

Note

`obs` and `sim` has to have the same length/dimension

The missing values in `obs` and `sim` are removed before the computation proceeds, and only those positions with non-missing values in `obs` and `sim` are considered in the computation

Author(s)

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

References

Fowler, K.; Coxon, G.; Freer, J.; Peel, M.; Wagener, T.; Western, A.; Woods, R.; Zhang, L. (2018). Simulating runoff under changing climatic conditions: A framework for model improvement. Water Resources Research, 54(12), 812-9832. doi:10.1029/2018WR023989.

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

Kling, H.; Fuchs, M.; Paulin, M. (2012). Runoff conditions in the upper Danube basin under an ensemble of climate change scenarios. Journal of Hydrology, 424, 264-277, doi:10.1016/j.jhydrol.2012.01.011.

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

Pfannerstill, M.; Guse, B.; Fohrer, N. (2014). Smart low flow signature metrics for an improved overall performance evaluation of hydrological models. Journal of Hydrology, 510, 447-458. doi:10.1016/j.jhydrol.2013.12.0

Santos, L.; Thirel, G.; Perrin, C. (2018). Pitfalls in using log-transformed flows within the sKGE criterion. doi:10.5194/hess-22-4583-2018

Knoben, W.J.; Freer, J.E.; Woods, R.A. (2019). Inherent benchmark or not? Comparing Nash-Sutcliffe and Kling-Gupta efficiency scores. Hydrology and Earth System Sciences, 23(10), 4323-4331. doi:10.5194/hess-23-4323-2019.

See Also

`KGE`, `KGElf`, `KGEnp`, `gof`, `ggof`

Examples

```
##################
# Example 1: Looking at the difference between 'method=2009' and 'method=2012'
# Loading daily streamflows of the Ega River (Spain), from 1961 to 1970
data(EgaEnEstellaQts)
obs <- EgaEnEstellaQts

# Simulated daily time series, initially equal to twice the observed values
sim <- 2*obs

# KGE 2009
KGE(sim=sim, obs=obs, method="2009", out.type="full")

# KGE 2012
KGE(sim=sim, obs=obs, method="2012", out.type="full")

# sKGE (Fowler et al., 2018):
sKGE(sim=sim, obs=obs, method="2012")

##################
# Example 2:
# Loading daily streamflows of the Ega River (Spain), from 1961 to 1970
data(EgaEnEstellaQts)
obs <- EgaEnEstellaQts

# Generating a simulated daily time series, initially equal to the observed series
sim <- obs

# Computing the 'sKGE' for the "best" (unattainable) case
sKGE(sim=sim, obs=obs)

##################
# Example 3: sKGE for simulated values equal to observations plus random noise
#            on the first half of the observed values.
#            This random noise has more relative importance for ow flows than
#            for medium and high flows.

# Randomly changing the first 1826 elements of 'sim', by using a normal distribution
# with mean 10 and standard deviation equal to 1 (default of 'rnorm').
sim[1:1826] <- obs[1:1826] + rnorm(1826, mean=10)
ggof(sim, obs)

sKGE(sim=sim, obs=obs)

##################
# Example 4: sKGE for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' during computations.

sKGE(sim=sim, obs=obs, fun=log)

# Verifying the previous value:
```

```
lsim <- log(sim)
lobs <- log(obs)
sKGE(sim=lsim, obs=lobs)

##################
# Example 5: sKGE for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and adding the Pushpalatha2012 constant
#            during computations

sKGE(sim=sim, obs=obs, fun=log, epsilon.type="Pushpalatha2012")

# Verifying the previous value, with the epsilon value following Pushpalatha2012
eps  <- mean(obs, na.rm=TRUE)/100
lsim <- log(sim+eps)
lobs <- log(obs+eps)
sKGE(sim=lsim, obs=lobs)

##################
# Example 6: sKGE for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and adding a user-defined constant
#            during computations

eps <- 0.01
sKGE(sim=sim, obs=obs, fun=log, epsilon.type="otherValue", epsilon.value=eps)

# Verifying the previous value:
lsim <- log(sim+eps)
lobs <- log(obs+eps)
sKGE(sim=lsim, obs=lobs)

##################
# Example 7: sKGE for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and using a user-defined factor
#            to multiply the mean of the observed values to obtain the constant
#            to be added to 'sim' and 'obs' during computations

fact <- 1/50
sKGE(sim=sim, obs=obs, fun=log, epsilon.type="otherFactor", epsilon.value=fact)

# Verifying the previous value:
eps  <- fact*mean(obs, na.rm=TRUE)
lsim <- log(sim+eps)
lobs <- log(obs+eps)
sKGE(sim=lsim, obs=lobs)

##################
# Example 8: sKGE for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying a
#            user-defined function to 'sim' and 'obs' during computations
```

```
fun1 <- function(x) {sqrt(x+1)}

sKGE(sim=sim, obs=obs, fun=fun1)

# Verifying the previous value, with the epsilon value following Pushpalatha2012
sim1 <- sqrt(sim+1)
obs1 <- sqrt(obs+1)
sKGE(sim=sim1, obs=obs1)

##################
# Example 9: sKGE for a two-column data frame where simulated values are equal to
#            observations plus random noise on the first half of the observed values

SIM <- cbind(sim, sim)
OBS <- cbind(obs, obs)

sKGE(sim=SIM, obs=OBS)

##################
# Example 10: sKGE for each year, where simulated values are given in a two-column data
#             frame equal to the observations plus random noise on the first half of the
#             observed values
SIM <- cbind(sim, sim)
OBS <- cbind(obs, obs)
sKGE(sim=SIM, obs=OBS, out.PerYear=TRUE)
```

---

ssq *Sum of the Squared Residuals*

Description

Sum of the Squared Residuals between `sim` and `obs`, with treatment of missing values.

Its units are the squared measurement units of `sim` and `obs`.

Usage

```
ssq(sim, obs, ...)

## Default S3 method:
ssq(sim, obs, na.rm = TRUE, fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
            epsilon.value=NA)

## S3 method for class 'data.frame'
ssq(sim, obs, na.rm=TRUE, fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
            epsilon.value=NA)
```

```
## S3 method for class 'matrix'
ssq(sim, obs, na.rm=TRUE, fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
            epsilon.value=NA)

## S3 method for class 'zoo'
ssq(sim, obs, na.rm=TRUE, fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
            epsilon.value=NA)
```

Arguments

`sim` numeric, zoo, matrix or data.frame with simulated values

`obs` numeric, zoo, matrix or data.frame with observed values

`na.rm` a logical value indicating whether 'NA' should be stripped before the computation proceeds.
When an 'NA' value is found at the i-th position in `obs` **OR** `sim`, the i-th value of `obs` **AND** `sim` are removed before the computation.

`fun` function to be applied to `sim` and `obs` in order to obtain transformed values thereof before computing this goodness-of-fit index.

The first argument MUST BE a numeric vector with any name (e.g., `x`), and additional arguments are passed using `...`.

`...` arguments passed to `fun`, in addition to the mandatory first numeric vector.

`epsilon.type` argument used to define a numeric value to be added to both `sim` and `obs` before applying `fun`.

It is was designed to allow the use of logarithm and other similar functions that do not work with zero values.

Valid values of `epsilon.type` are:

1) `"none"`: `sim` and `obs` are used by `fun` without the addition of any numeric value. This is the default option.

2) `"Pushpalatha2012"`: one hundredth (1/100) of the mean observed values is added to both `sim` and `obs` before applying `fun`, as described in Pushpalatha et al. (2012).

3) `"otherFactor"`: the numeric value defined in the `epsilon.value` argument is used to multiply the the mean observed values, instead of the one hundredth (1/100) described in Pushpalatha et al. (2012). The resulting value is then added to both `sim` and `obs`, before applying `fun`.

4) `"otherValue"`: the numeric value defined in the `epsilon.value` argument is directly added to both `sim` and `obs`, before applying `fun`.

`epsilon.value` -) when `epsilon.type="otherValue"` it represents the numeric value to be added to both `sim` and `obs` before applying `fun`.
-) when `epsilon.type="otherFactor"` it represents the numeric factor used to multiply the mean of the observed values, instead of the one hundredth (1/100) described in Pushpalatha et al. (2012). The resulting value is then added to both `sim` and `obs` before applying `fun`.

Details

$$ssr = \sum_{i=1}^{N} (S_i - O_i)^2$$

Value

Sum of the squared residuals between `sim` and `obs`.

If `sim` and `obs` are matrixes, the returned value is a vector, with the SSR between each column of `sim` and `obs`.

Note

`obs` and `sim` has to have the same length/dimension

The missing values in `obs` and `sim` are removed before the computation proceeds, and only those positions with non-missing values in `obs` and `sim` are considered in the computation

Author(s)

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

References

Willmott, C.J.; Matsuura, K.; Robeson, S.M. (2009). Ambiguities inherent in sums-of-squares-based error statistics, Atmospheric Environment, 43, 749-752, doi:10.1016/j.atmosenv.2008.10.005.

See Also

`pbias`, `pbiasfdc`, `mae`, `mse`, `rmse`, `ubRMSE`, `nrmse`, `gof`, `ggof`

Examples

```
obs <- 1:10
sim <- 1:10
ssq(sim, obs)

obs <- 1:10
sim <- 2:11
ssq(sim, obs)

##################
# Loading daily streamflows of the Ega River (Spain), from 1961 to 1970
data(EgaEnEstellaQts)
obs <- EgaEnEstellaQts

# Generating a simulated daily time series, initially equal to the observed series
sim <- obs
```

```
# Computing the 'rNSeff' for the "best" (unattainable) case
ssq(sim=sim, obs=obs)

# Randomly changing the first 2000 elements of 'sim', by using a normal distribution
# with mean 10 and standard deviation equal to 1 (default of 'rnorm').
sim[1:2000] <- obs[1:2000] + rnorm(2000, mean=10)

# Computing the new 'rNSeff'
ssq(sim=sim, obs=obs)
```

---

ubRMSE *Unbiased Root Mean Square Error*

Description

unbiased Root Mean Square Error (ubRMSE) between `sim` and `obs`, in the same units of `sim` and `obs`, with treatment of missing values.

ubRMSE was introduced by Entekhabi et al. (2010) to improve the evaluation of the temporal dynamic of volumentric soil moisture, by removing from the traditional RMSE the mean bias error caused by the mistmatch between the spatial representativeness of in situ soil moisture and the corresponding gridded values.

A smaller value indicates better model performance.

Usage

```
ubRMSE(sim, obs, ...)

## Default S3 method:
ubRMSE(sim, obs, na.rm=TRUE, fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
              epsilon.value=NA)

## S3 method for class 'data.frame'
ubRMSE(sim, obs, na.rm=TRUE, fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
              epsilon.value=NA)

## S3 method for class 'matrix'
ubRMSE(sim, obs, na.rm=TRUE, fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
              epsilon.value=NA)

## S3 method for class 'zoo'
ubRMSE(sim, obs, na.rm=TRUE, fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
              epsilon.value=NA)
```

Arguments

sim
: numeric, zoo, matrix or data.frame with simulated values

obs
: numeric, zoo, matrix or data.frame with observed values

na.rm
: a logical value indicating whether 'NA' should be stripped before the computation proceeds.
When an 'NA' value is found at the i-th position in obs **OR** sim, the i-th value of obs **AND** sim are removed before the computation.

fun
: function to be applied to sim and obs in order to obtain transformed values thereof before computing the Root Mean Square Error.
The first argument MUST BE a numeric vector with any name (e.g., x), and additional arguments are passed using ....

...
: arguments passed to fun, in addition to the mandatory first numeric vector.

epsilon.type
: argument used to define a numeric value to be added to both sim and obs before applying FUN.
It is was designed to allow the use of logarithm and other similar functions that do not work with zero values.
Valid values of epsilon.type are:
1) "none": sim and obs are used by fun without the addition of any numeric value. This is the default option.
2) "Pushpalatha2012": one hundredth (1/100) of the mean observed values is added to both sim and obs before applying fun, as described in Pushpalatha et al. (2012).
3) "otherFactor": the numeric value defined in the epsilon.value argument is used to multiply the the mean observed values, instead of the one hundredth (1/100) described in Pushpalatha et al. (2012). The resulting value is then added to both sim and obs, before applying fun.
4) "otherValue": the numeric value defined in the epsilon.value argument is directly added to both sim and obs, before applying fun.

epsilon.value
: numeric value to be added to both sim and obs when epsilon.type="otherValue".

Details

The traditional root mean square error (RMSE) is severely compromised if there are biases in either the mean or the amplitude of fluctuations of the simulated values. If it can be estimated reliably, the mean-bias (BIAS) can easily be removed from RMSE, leading to the unbiased RMSE:

$$ubRMSE = \sqrt{RMSE^2 - BIAS^2}$$

Value

Unbiased Root mean square error (ubRMSE) between sim and obs.

If sim and obs are matrixes or data.frames, the returned value is a vector, with the ubRMSE between each column of sim and obs.

Note

obs and sim has to have the same length/dimension

The missing values in obs and sim are removed before the computation proceeds, and only those positions with non-missing values in obs and sim are considered in the computation

Author(s)

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

References

Entekhabi, D.; Reichle, R.H.; Koster, R.D.; Crow, W.T. (2010). Performance metrics for soil moisture retrievals and application requirements. Journal of Hydrometeorology, 11(3), 832-840. doi: 10.1175/2010JHM1223.1.

Ling, X.; Huang, Y.; Guo, W.; Wang, Y.; Chen, C.; Qiu, B.; Ge, J.; Qin, K.; Xue, Y.; Peng, J. (2021). Comprehensive evaluation of satellite-based and reanalysis soil moisture products using in situ observations over China. Hydrology and Earth System Sciences, 25(7), 4209-4229. doi:10.5194/hess-25-4209-2021.

See Also

pbias, pbiasfdc, mae, mse, rmse, nrmse, ssq, gof, ggof

Examples

```
##################
# Example 1: basic ideal case
obs <- 1:10
sim <- 1:10
ubRMSE(sim, obs)

obs <- 1:10
sim <- 2:11
ubRMSE(sim, obs)

##################
# Example 2:
# Loading daily streamflows of the Ega River (Spain), from 1961 to 1970
data(EgaEnEstellaQts)
obs <- EgaEnEstellaQts

# Generating a simulated daily time series, initially equal to the observed series
sim <- obs

# Computing the 'ubRMSE' for the "best" (unattainable) case
ubRMSE(sim=sim, obs=obs)

##################
# Example 3: ubRMSE for simulated values equal to observations plus random noise
```

```
#            on the first half of the observed values.
#            This random noise has more relative importance for ow flows than
#            for medium and high flows.

# Randomly changing the first 1826 elements of 'sim', by using a normal distribution
# with mean 10 and standard deviation equal to 1 (default of 'rnorm').
sim[1:1826] <- obs[1:1826] + rnorm(1826, mean=10)
ggof(sim, obs)

ubRMSE(sim=sim, obs=obs)

##################
# Example 4: ubRMSE for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' during computations.

ubRMSE(sim=sim, obs=obs, fun=log)

# Verifying the previous value:
lsim <- log(sim)
lobs <- log(obs)
ubRMSE(sim=lsim, obs=lobs)

##################
# Example 5: ubRMSE for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and adding the Pushpalatha2012 constant
#            during computations

ubRMSE(sim=sim, obs=obs, fun=log, epsilon.type="Pushpalatha2012")

# Verifying the previous value, with the epsilon value following Pushpalatha2012
eps  <- mean(obs, na.rm=TRUE)/100
lsim <- log(sim+eps)
lobs <- log(obs+eps)
ubRMSE(sim=lsim, obs=lobs)

##################
# Example 6: ubRMSE for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and adding a user-defined constant
#            during computations

eps <- 0.01
ubRMSE(sim=sim, obs=obs, fun=log, epsilon.type="otherValue", epsilon.value=eps)

# Verifying the previous value:
lsim <- log(sim+eps)
lobs <- log(obs+eps)
ubRMSE(sim=lsim, obs=lobs)

##################
# Example 7: ubRMSE for simulated values equal to observations plus random noise
```

```
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and using a user-defined factor
#            to multiply the mean of the observed values to obtain the constant
#            to be added to 'sim' and 'obs' during computations

fact <- 1/50
ubRMSE(sim=sim, obs=obs, fun=log, epsilon.type="otherFactor", epsilon.value=fact)

# Verifying the previous value:
eps  <- fact*mean(obs, na.rm=TRUE)
lsim <- log(sim+eps)
lobs <- log(obs+eps)
ubRMSE(sim=lsim, obs=lobs)

##################
# Example 8: ubRMSE for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying a
#            user-defined function to 'sim' and 'obs' during computations

fun1 <- function(x) {sqrt(x+1)}

ubRMSE(sim=sim, obs=obs, fun=fun1)

# Verifying the previous value, with the epsilon value following Pushpalatha2012
sim1 <- sqrt(sim+1)
obs1 <- sqrt(obs+1)
ubRMSE(sim=sim1, obs=obs1)
```

---

valindex *Valid Indexes*

Description

Identify the indexes that are simultaneously valid (not missing) in `sim` and `obs`.

Usage

```
valindex(sim, obs, ...)

## Default S3 method:
valindex(sim, obs, ...)

## S3 method for class 'matrix'
valindex(sim, obs, ...)
```

Arguments

`sim`	zoo, xts, numeric, matrix or data.frame with simulated values

obs	zoo, xts, numeric, matrix or data.frame with observed values
...	further arguments passed to or from other methods.

Value

A vector with the indexes that are simultaneously valid (not missing) in obs and sim.

Note

This function is used in the functions of this package for removing missing values from the observed and simulated time series.

Author(s)

Mauricio Zambrano Bigiarini <mauricio.zambrano@ing.unitn.it>

See Also

is.na, which

Examples

```
sim <- 1:5
obs <- c(1, NA, 3, NA, 5)
valindex(sim, obs)
```

---

ve *Volumetric Efficiency*

Description

Volumetric efficiency between sim and obs, with treatment of missing values.

Usage

```
VE(sim, obs, ...)

## Default S3 method:
VE(sim, obs, na.rm=TRUE, fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
            epsilon.value=NA)

## S3 method for class 'data.frame'
VE(sim, obs, na.rm=TRUE, fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
            epsilon.value=NA)
```

```
## S3 method for class 'matrix'
VE(sim, obs, na.rm=TRUE, fun=NULL, ...,
        epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
           epsilon.value=NA)

## S3 method for class 'zoo'
VE(sim, obs, na.rm=TRUE, fun=NULL, ...,
        epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
           epsilon.value=NA)
```

Arguments

`sim` numeric, zoo, matrix or data.frame with simulated values

`obs` numeric, zoo, matrix or data.frame with observed values

`na.rm` a logical value indicating whether 'NA' should be stripped before the computation proceeds.
When an 'NA' value is found at the i-th position in `obs` **OR** `sim`, the i-th value of `obs` **AND** `sim` are removed before the computation.

`fun` function to be applied to `sim` and `obs` in order to obtain transformed values thereof before computing this goodness-of-fit index.

The first argument MUST BE a numeric vector with any name (e.g., `x`), and additional arguments are passed using `...`.

`...` arguments passed to `fun`, in addition to the mandatory first numeric vector.

`epsilon.type` argument used to define a numeric value to be added to both `sim` and `obs` before applying `fun`.

It is was designed to allow the use of logarithm and other similar functions that do not work with zero values.

Valid values of `epsilon.type` are:

1) `"none"`: `sim` and `obs` are used by `fun` without the addition of any numeric value. This is the default option.

2) `"Pushpalatha2012"`: one hundredth (1/100) of the mean observed values is added to both `sim` and `obs` before applying `fun`, as described in Pushpalatha et al. (2012).

3) `"otherFactor"`: the numeric value defined in the `epsilon.value` argument is used to multiply the the mean observed values, instead of the one hundredth (1/100) described in Pushpalatha et al. (2012). The resulting value is then added to both `sim` and `obs`, before applying `fun`.

4) `"otherValue"`: the numeric value defined in the `epsilon.value` argument is directly added to both `sim` and `obs`, before applying `fun`.

`epsilon.value` -) when `epsilon.type="otherValue"` it represents the numeric value to be added to both `sim` and `obs` before applying `fun`.
-) when `epsilon.type="otherFactor"` it represents the numeric factor used to multiply the mean of the observed values, instead of the one hundredth (1/100) described in Pushpalatha et al. (2012). The resulting value is then added to both `sim` and `obs` before applying `fun`.

Details

$$VE = 1 - \frac{\sum_{i=1}^{N} |S_i - O_i|}{\sum_{i=1}^{N} (O_i)}$$

Volumetric efficiency was proposed in order to circumvent some problems associated to the Nash-Sutcliffe efficiency. It ranges from 0 to 1 and represents the fraction of water delivered at the proper time; its compliment represents the fractional volumetric mistmach (Criss and Winston, 2008).

Value

Volumetric efficiency between `sim` and `obs`.

If `sim` and `obs` are matrixes, the returned value is a vector, with the Volumetric efficiency between each column of `sim` and `obs`.

Note

`obs` and `sim` have to have the same length/dimension

The missing values in `obs` and `sim` are removed before the computation proceeds, and only those positions with non-missing values in `obs` and `sim` are considered in the computation

Author(s)

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

References

Criss, R.E.; Winston, W.E. (2008), Do Nash values have value? Discussion and alternate proposals. Hydrological Processes, 22: 2723-2725. doi:10.1002/hyp.7072.

See Also

`gof`, `ggof`, `NSE`

Examples

```
##################
# Example 1: basic ideal case
obs <- 1:10
sim <- 1:10
VE(sim, obs)

obs <- 1:10
sim <- 2:11
VE(sim, obs)

##################
# Example 2:
```

```
# Loading daily streamflows of the Ega River (Spain), from 1961 to 1970
data(EgaEnEstellaQts)
obs <- EgaEnEstellaQts

# Generating a simulated daily time series, initially equal to the observed series
sim <- obs

# Computing the 'VE' for the "best" (unattainable) case
VE(sim=sim, obs=obs)

##################
# Example 3: VE for simulated values equal to observations plus random noise
#            on the first half of the observed values.
#            This random noise has more relative importance for ow flows than
#            for medium and high flows.

# Randomly changing the first 1826 elements of 'sim', by using a normal distribution
# with mean 10 and standard deviation equal to 1 (default of 'rnorm').
sim[1:1826] <- obs[1:1826] + rnorm(1826, mean=10)
ggof(sim, obs)

VE(sim=sim, obs=obs)

##################
# Example 4: VE for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' during computations.

VE(sim=sim, obs=obs, fun=log)

# Verifying the previous value:
lsim <- log(sim)
lobs <- log(obs)
VE(sim=lsim, obs=lobs)

##################
# Example 5: VE for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and adding the Pushpalatha2012 constant
#            during computations

VE(sim=sim, obs=obs, fun=log, epsilon.type="Pushpalatha2012")

# Verifying the previous value, with the epsilon value following Pushpalatha2012
eps  <- mean(obs, na.rm=TRUE)/100
lsim <- log(sim+eps)
lobs <- log(obs+eps)
VE(sim=lsim, obs=lobs)

##################
# Example 6: VE for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and adding a user-defined constant
```

```
#            during computations

eps <- 0.01
VE(sim=sim, obs=obs, fun=log, epsilon.type="otherValue", epsilon.value=eps)

# Verifying the previous value:
lsim <- log(sim+eps)
lobs <- log(obs+eps)
VE(sim=lsim, obs=lobs)

##################
# Example 7: VE for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and using a user-defined factor
#            to multiply the mean of the observed values to obtain the constant
#            to be added to 'sim' and 'obs' during computations

fact <- 1/50
VE(sim=sim, obs=obs, fun=log, epsilon.type="otherFactor", epsilon.value=fact)

# Verifying the previous value:
eps  <- fact*mean(obs, na.rm=TRUE)
lsim <- log(sim+eps)
lobs <- log(obs+eps)
VE(sim=lsim, obs=lobs)

##################
# Example 8: VE for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying a
#            user-defined function to 'sim' and 'obs' during computations

fun1 <- function(x) {sqrt(x+1)}

VE(sim=sim, obs=obs, fun=fun1)

# Verifying the previous value, with the epsilon value following Pushpalatha2012
sim1 <- sqrt(sim+1)
obs1 <- sqrt(obs+1)
VE(sim=sim1, obs=obs1)
```

---

wNSE *Weighted Nash-Sutcliffe efficiency*

Description

Weighted Nash-Sutcliffe efficiency between `sim` and `obs`, with treatment of missing values.

This goodness-of-fit measure was proposed by Hundecha and Bardossy (2004) to put special focus on high values.

Usage

```
wNSE(sim, obs, ...)

## Default S3 method:
wNSE(sim, obs, na.rm=TRUE, fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
             epsilon.value=NA)

## S3 method for class 'data.frame'
wNSE(sim, obs, na.rm=TRUE, fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
             epsilon.value=NA)

## S3 method for class 'matrix'
wNSE(sim, obs, na.rm=TRUE, fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
             epsilon.value=NA)

## S3 method for class 'zoo'
wNSE(sim, obs, na.rm=TRUE, fun=NULL, ...,
         epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
             epsilon.value=NA)
```

Arguments

`sim` numeric, zoo, matrix or data.frame with simulated values

`obs` numeric, zoo, matrix or data.frame with observed values

`na.rm` a logical value indicating whether 'NA' should be stripped before the computation proceeds.
When an 'NA' value is found at the i-th position in `obs` **OR** `sim`, the i-th value of `obs` **AND** `sim` are removed before the computation.

`fun` function to be applied to `sim` and `obs` in order to obtain transformed values thereof before computing the weighted Nash-Sutcliffe efficiency.

The first argument MUST BE a numeric vector with any name (e.g., `x`), and additional arguments are passed using `...`.

`...` arguments passed to `fun`, in addition to the mandatory first numeric vector.

`epsilon.type` argument used to define a numeric value to be added to both `sim` and `obs` before applying `FUN`.

It is was designed to allow the use of logarithm and other similar functions that do not work with zero values.

Valid values of `epsilon.type` are:

1) `"none"`: `sim` and `obs` are used by `fun` without the addition of any numeric value. This is the default option.

2) `"Pushpalatha2012"`: one hundredth (1/100) of the mean observed values is added to both `sim` and `obs` before applying `fun`, as described in Pushpalatha et al. (2012).

3) "otherFactor": the numeric value defined in the epsilon.value argument is used to multiply the mean of the observed values, instead of the one hundredth (1/100) described in Pushpalatha et al. (2012). The resulting value is then added to both sim and obs, before applying fun.

4) "otherValue": the numeric value defined in the epsilon.value argument is directly added to both sim and obs, before applying fun.

epsilon.value -) when epsilon.type="otherValue" it represents the numeric value to be added to both sim and obs before applying fun.
-) when epsilon.type="otherFactor" it represents the numeric factor used to multiply the mean of the observed values, instead of the one hundredth (1/100) described in Pushpalatha et al. (2012). The resulting value is then added to both sim and obs before applying fun.

Details

$$wNSE = 1 - \frac{\sum_{i=1}^{N} O_i * (S_i - O_i)^2}{\sum_{i=1}^{N} O_i * (O_i - \bar{O})^2}$$

Value

Weighted Nash-Sutcliffe efficiency between sim and obs.

If sim and obs are matrixes, the returned value is a vector, with the relative Nash-Sutcliffe efficiency between each column of sim and obs.

Note

obs and sim has to have the same length/dimension

The missing values in obs and sim are removed before the computation proceeds, and only those positions with non-missing values in obs and sim are considered in the computation

If some of the observed values are equal to zero (at least one of them), this index can not be computed.

Author(s)

sluedtke (github user)

References

Nash, J.E. and J.V. Sutcliffe, River flow forecasting through conceptual models. Part 1: A discussion of principles, J. Hydrol. 10 (1970), pp. 282-290. doi:10.1016/0022-1694(70)90255-6.

Hundecha, Y., Bardossy, A. (2004). Modeling of the effect of land use changes on the runoff generation of a river basin through parameter regionalization of a watershed model. Journal of hydrology, 292(1-4), 281-295. doi:10.1016/j.jhydrol.2004.01.002.

Hundecha, Y., Ouarda, T. B., Bardossy, A. (2008). Regional estimation of parameters of a rainfall-runoff model at ungauged watersheds using the 'spatial' structures of the parameters within a canonical physiographic-climatic space. Water Resources Research, 44(1). doi:10.1029/2006WR005439.

Hundecha, Y. and Merz, B. (2012), Exploring the Relationship between Changes in Climate and Floods Using a Model-Based Analysis, Water Resour. Res., 48(4), 1-21, doi:10.1029/2011WR010527..

See Also

NSE, rNSE, mNSE, KGE, gof, ggof

Examples

```
##################
# Example 1: basic ideal case
obs <- 1:10
sim <- 1:10
wNSE(sim, obs)

obs <- 1:10
sim <- 2:11
wNSE(sim, obs)

##################
# Example 2:
# Loading daily streamflows of the Ega River (Spain), from 1961 to 1970
data(EgaEnEstellaQts)
obs <- EgaEnEstellaQts

# Generating a simulated daily time series, initially equal to the observed series
sim <- obs

# Computing the 'wNSE' for the "best" (unattainable) case
wNSE(sim=sim, obs=obs)

##################
# Example 3: wNSE for simulated values equal to observations plus random noise
#            on the first half of the observed values.
#            This random noise has more relative importance for ow flows than
#            for medium and high flows.

# Randomly changing the first 1826 elements of 'sim', by using a normal distribution
# with mean 10 and standard deviation equal to 1 (default of 'rnorm').
sim[1:1826] <- obs[1:1826] + rnorm(1826, mean=10)
ggof(sim, obs)

wNSE(sim=sim, obs=obs)

##################
# Example 4: wNSE for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' during computations.
```

```
wNSE(sim=sim, obs=obs, fun=log)

# Verifying the previous value:
lsim <- log(sim)
lobs <- log(obs)
wNSE(sim=lsim, obs=lobs)

##################
# Example 5: wNSE for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and adding the Pushpalatha2012 constant
#            during computations

wNSE(sim=sim, obs=obs, fun=log, epsilon.type="Pushpalatha2012")

# Verifying the previous value, with the epsilon value following Pushpalatha2012
eps  <- mean(obs, na.rm=TRUE)/100
lsim <- log(sim+eps)
lobs <- log(obs+eps)
wNSE(sim=lsim, obs=lobs)

##################
# Example 6: wNSE for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and adding a user-defined constant
#            during computations

eps <- 0.01
wNSE(sim=sim, obs=obs, fun=log, epsilon.type="otherValue", epsilon.value=eps)

# Verifying the previous value:
lsim <- log(sim+eps)
lobs <- log(obs+eps)
wNSE(sim=lsim, obs=lobs)

##################
# Example 7: wNSE for simulated values equal to observations plus random noise
#            on the first half of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' and using a user-defined factor
#            to multiply the mean of the observed values to obtain the constant
#            to be added to 'sim' and 'obs' during computations

fact <- 1/50
wNSE(sim=sim, obs=obs, fun=log, epsilon.type="otherFactor", epsilon.value=fact)

# Verifying the previous value:
eps  <- fact*mean(obs, na.rm=TRUE)
lsim <- log(sim+eps)
lobs <- log(obs+eps)
wNSE(sim=lsim, obs=lobs)

##################
# Example 8: wNSE for simulated values equal to observations plus random noise
```

```
#           on the first half of the observed values and applying a
#           user-defined function to 'sim' and 'obs' during computations

fun1 <- function(x) {sqrt(x+1)}

wNSE(sim=sim, obs=obs, fun=fun1)

# Verifying the previous value, with the epsilon value following Pushpalatha2012
sim1 <- sqrt(sim+1)
obs1 <- sqrt(obs+1)
wNSE(sim=sim1, obs=obs1)
```

---

wsNSE *Weighted seasonal Nash-Sutcliffe Efficiency*

Description

Weighted seasonal Nash-Sutcliffe Efficiency between `sim` and `obs`, with treatment of missing values.

This function is designed to identify differences in high or low values, depending on the user-defined value given to the `lambda` argument. See Usage and Details.

Usage

```
wsNSE(sim, obs, ...)

## Default S3 method:
wsNSE(sim, obs, na.rm=TRUE,
              j=2, lambda=0.95, lQ.thr=0.6, hQ.thr=0.1, fun=NULL, ...,
          epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
              epsilon.value=NA)

## S3 method for class 'data.frame'
wsNSE(sim, obs, na.rm=TRUE,
              j=2, lambda=0.95, lQ.thr=0.6, hQ.thr=0.1, fun=NULL, ...,
          epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
              epsilon.value=NA)

## S3 method for class 'matrix'
wsNSE(sim, obs, na.rm=TRUE,
              j=2, lambda=0.95, lQ.thr=0.6, hQ.thr=0.1, fun=NULL, ...,
          epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
              epsilon.value=NA)

## S3 method for class 'zoo'
wsNSE(sim, obs, na.rm=TRUE,
              j=2, lambda=0.95, lQ.thr=0.6, hQ.thr=0.1, fun=NULL, ...,
          epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
              epsilon.value=NA)
```

Arguments

- `sim` — numeric, zoo, matrix or data.frame with simulated values
- `obs` — numeric, zoo, matrix or data.frame with observed values
- `na.rm` — a logical value indicating whether 'NA' should be stripped before the computation proceeds.
  When an 'NA' value is found at the i-th position in `obs` **OR** `sim`, the i-th value of `obs` **AND** `sim` are removed before the computation.
- `j` — numeric, representing an arbitrary value used to power the differences between observations and simulations. By default `j=2`, which mimics the traditional Nash-Sutcliffe function, mainly focused on thr representation of high values. For low flows, suggested values for `j` are 1, 1/2 or 1/3. See Legates and McCabe, (1999) and Krausse et al. (2005) for a discussion of suggested values of `j`.
- `lambda` — numeric in [0, 1] representing the weight given to the high observed values. The closer the `lambda=1` value is to 1, the higher the weight given to high values. On the contrary, the closer the `lambda=1` value is to 0, the higher the weight given to low values.

  Low values get a weight equal to `1-lambda`. Between high and low values there is a linear transition from `lambda` to `1-lambda`, respectively.

  Suggested values for lambda are `lambda=0.95` when focusing in high (streamflow) values and `lambda=0.05` when focusing in low (streamflow) values.
- `lQ.thr` — numeric, representing the non-exceedence probabiliy used to identify low flows in `obs`. All values in `obs` that are equal or lower than `quantile(obs, probs=(1-lQ.thr))` are considered as low values. By default `lQ.thr=0.6`.

  On the other hand, the low values in `sim` are those located at the same i-th position than the i-th value of the `obs` deemed as low flows.
- `hQ.thr` — numeric, representing the non-exceedence probabiliy used to identify high flows in `obs`. All values in `obs` that are equal or higher than `quantile(obs, probs=(1-hQ.thr))` are considered as high flows. By default `hQ.thr=0.1`.

  On the other hand, the high values in `sim` are those located at the same i-th position than the i-th value of the `obs` deemed as high flows.
- `fun` — function to be applied to `sim` and `obs` in order to obtain transformed values thereof before computing this goodness-of-fit index.

  The first argument MUST BE a numeric vector with any name (e.g., `x`), and additional arguments are passed using `...`.
- `...` — arguments passed to `fun`, in addition to the mandatory first numeric vector.
- `epsilon.type` — argument used to define a numeric value to be added to both `sim` and `obs` before applying `fun`.

  It is was designed to allow the use of logarithm and other similar functions that do not work with zero values.

  Valid values of `epsilon.type` are:

  1) `"none"`: `sim` and `obs` are used by `fun` without the addition of any numeric value. This is the default option.

2) "Pushpalatha2012": one hundredth (1/100) of the mean observed values is added to both sim and obs before applying fun, as described in Pushpalatha et al. (2012).

3) "otherFactor": the numeric value defined in the epsilon.value argument is used to multiply the the mean observed values, instead of the one hundredth (1/100) described in Pushpalatha et al. (2012). The resulting value is then added to both sim and obs, before applying fun.

4) "otherValue": the numeric value defined in the epsilon.value argument is directly added to both sim and obs, before applying fun.

epsilon.value -) when epsilon.type="otherValue" it represents the numeric value to be added to both sim and obs before applying fun.
-) when epsilon.type="otherFactor" it represents the numeric factor used to multiply the mean of the observed values, instead of the one hundredth (1/100) described in Pushpalatha et al. (2012). The resulting value is then added to both sim and obs before applying fun.

Details

The weighted seasonal Nash-Sutcliffe Efficiency was proposed by Zambrano-Bigiarini and Bellin (2012), inspired by the well-known Nash-Sutcliffe efficiency (NSE, Nash and Sutcliffe, 1970), and the commentaries made by Schaefli and Gupta (2007) and Criss and Winston (2008).

This function gives different weights to the high/low values in the (obs_i - sim_i) terms used in the Nash-Sutcliffe formula, using high weights for high or low flows, depending on how close the user-defined 'lambda' value is to 1 or zero, respectively. Between high and low values there is a linear transition from lambda to 1-lambda, respectively.

Following the traditional Nash-Sutcliffe efficiency, the weighted seasonal Nash-Sutcliffe Efficiency (wsNSE) ranges from -Inf to 1, with an optimal value of 1. Higher values of wsNSE indicate lower differences between sim and obs. Essentially, the closer to 1, the more similarsim and obs are.

Value

numeric with the the weighted seasonal Nash-Sutcliffe Efficiency (wsNSE) between sim and obs. If sim and obs are matrices, the output value is a vector, with the the weighted seasonal Nash-Sutcliffe Efficiency (wsNSE) between each column of sim and obs.

Note

obs and sim has to have the same length/dimension

The missing values in obs and sim are removed before the computation proceeds, and only those positions with non-missing values in obs and sim are considered in the computation

Author(s)

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

References

Zambrano-Bigiarini, M.; Bellin, A. (2012). Comparing goodness-of-fit measures for calibration of models focused on extreme events. EGU General Assembly 2012, Vienna, Austria, 22-27 Apr 2012, EGU2012-11549-1.

Nash, J.E.; J.V. Sutcliffe. (1970). River flow forecasting through conceptual models. Part 1: a discussion of principles, Journal of Hydrology 10, pp. 282-290. doi:10.1016/0022-1694(70)90255-6.

Schaefli, B.; Gupta, H. (2007). Do Nash values have value?. Hydrological Processes 21, 2075-2080. doi:10.1002/hyp.6825.

Criss, R. E.; Winston, W. E. (2008), Do Nash values have value?. Discussion and alternate proposals. Hydrological Processes, 22: 2723-2725. doi:10.1002/hyp.7072.

Yilmaz, K. K.; Gupta, H. V.; Wagener, T. (2008), A process-based diagnostic approach to model evaluation: Application to the NWS distributed hydrologic model, Water Resources Research, 44, W09417, doi:10.1029/2007WR006716.

Krause, P.; Boyle, D.P.; Base, F. (2005). Comparison of different efficiency criteria for hydrological model assessment, Advances in Geosciences, 5, 89-97. doi:10.5194/adgeo-5-89-2005.

Legates, D.R.; McCabe, G. J. Jr. (1999), Evaluating the Use of "Goodness-of-Fit" Measures in Hydrologic and Hydroclimatic Model Validation, Water Resour. Res., 35(1), 233-241. doi:10.1029/1998WR900018.

See Also

NSE, wNSE, wsNSE, APFB, KGElf, gof, ggof

Examples

```
##################
# Example 1: Looking at the difference between 'KGE', 'NSE', 'wNSE', 'wsNSE',
# 'APFB' and 'KGElf' for detecting differences in high flows

# Loading daily streamflows of the Ega River (Spain), from 1961 to 1970
data(EgaEnEstellaQts)
obs <- EgaEnEstellaQts

# Simulated daily time series, created equal to the observed values and then
# random noise is added only to high flows, i.e., those equal or higher than
# the quantile 0.9 of the observed values.
sim      <- obs
hQ.thr   <- quantile(obs, probs=0.9, na.rm=TRUE)
hQ.index <- which(obs >= hQ.thr)
hQ.n     <- length(hQ.index)
sim[hQ.index] <- sim[hQ.index] + rnorm(hQ.n, mean=mean(sim[hQ.index], na.rm=TRUE))

# Traditional Kling-Gupta eficiency (Gupta and Kling, 2009)
KGE(sim=sim, obs=obs)

# Traditional Nash-Sutcliffe eficiency (Nash and Sutcliffe, 1970)
NSE(sim=sim, obs=obs)
```

```
# Weighted Nash-Sutcliffe efficiency (Hundecha and Bardossy, 2004)
wNSE(sim=sim, obs=obs)

# wsNSE (Zambrano-Bigiarini and Bellin, 2012):
wsNSE(sim=sim, obs=obs)

# APFB (Mizukami et al., 2019):
APFB(sim=sim, obs=obs)


##################
# Example 2: Looking at the difference between 'KGE', 'NSE', 'wsNSE',
# 'dr', 'rd', 'md', 'APFB' and 'KGElf' for detecting differences in low flows

# Loading daily streamflows of the Ega River (Spain), from 1961 to 1970
data(EgaEnEstellaQts)
obs <- EgaEnEstellaQts

# Simulated daily time series, created equal to the observed values and then
# random noise is added only to low flows, i.e., those equal or lower than
# the quantile 0.4 of the observed values.
sim      <- obs
lQ.thr   <- quantile(obs, probs=0.4, na.rm=TRUE)
lQ.index <- which(obs <= lQ.thr)
lQ.n     <- length(lQ.index)
sim[lQ.index] <- sim[lQ.index] + rnorm(lQ.n, mean=mean(sim[lQ.index], na.rm=TRUE))

# Traditional Kling-Gupta eficiency (Gupta and Kling, 2009)
KGE(sim=sim, obs=obs)

# Traditional Nash-Sutcliffe eficiency (Nash and Sutcliffe, 1970)
NSE(sim=sim, obs=obs)

# Weighted seasonal Nash-Sutcliffe efficiency (Zambrano-Bigiarini and Bellin, 2012):
wsNSE(sim=sim, obs=obs, lambda=0.05, j=1/2)

# Refined Index of Agreement (Willmott et al., 2012):
dr(sim=sim, obs=obs)

# Relative Index of Agreement (Krause et al., 2005):
rd(sim=sim, obs=obs)

# Modified Index of Agreement (Krause et al., 2005):
md(sim=sim, obs=obs)

# KGElf (Garcia et al., 2017):
KGElf(sim=sim, obs=obs)


##################
# Example 3:
# Loading daily streamflows of the Ega River (Spain), from 1961 to 1970
```

```
data(EgaEnEstellaQts)
obs <- EgaEnEstellaQts

# Generating a simulated daily time series, initially equal to the observed series
sim <- obs

# Computing the 'wsNSE' for the "best" (unattainable) case
wsNSE(sim=sim, obs=obs)


##################
# Example 4: wsNSE for simulated values created equal to the observed values and then
#            random noise is added only to high flows, i.e., those equal or higher than
#            the quantile 0.9 of the observed values and applying (natural)
#            logarithm to 'sim' and 'obs' during computations.

wsNSE(sim=sim, obs=obs, fun=log)

# Verifying the previous value:
lsim <- log(sim)
lobs <- log(obs)
wsNSE(sim=lsim, obs=lobs)


##################
# Example 5: wsNSE for simulated values created equal to the observed values and then
#            random noise is added only to high flows, i.e., those equal or higher than
#            the quantile 0.9 of the observed values and applying a
#            user-defined function to 'sim' and 'obs' during computations

fun1 <- function(x) {sqrt(x+1)}

wsNSE(sim=sim, obs=obs, fun=fun1)

# Verifying the previous value, with the epsilon value following Pushpalatha2012
sim1 <- sqrt(sim+1)
obs1 <- sqrt(obs+1)
wsNSE(sim=sim1, obs=obs1)
```

Index