

# Package ‘hydroGOF’

May 8, 2026

**Type** Package

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

**Version** 0.7-0

**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 (>= 0.8-6), xts (>= 0.8-2), methods, stats

**Suggests** knitr, rmarkdown, testthat

**VignetteBuilder** knitr

**URL** <https://hzambran.github.io/hydroGOF/>,  
<https://github.com/hzambran/hydroGOF>,  
<https://CRAN.R-project.org/package=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] (ORCID:  
<<https://orcid.org/0000-0002-9536-643X>>)

**Date/Publication** 2026-05-01 05:10:38 UTC

## Contents

hydroGOF-package	3
APFB	8
br2	13
cp	18
d	23
dr	27
EgaEnEstellaQts	32
ggof	33
gof	41
HFB	52
JDKGE	57
KGE	62
KGEkm	69
KGElf	77
KGENp	82
LCE	87
LME	92
mae	98
md	103
me	106
mNSE	110
mse	114
nrmse	118
NSE	123
pbias	127
pbiasfdc	132
pfactor	136
plot2	139
plotbands	142
plotbandsonly	145
PMR	146
R2	152
rd	156
rfactor	161
rmse	164
rNSE	168
rPearson	172
rSD	177
rSpearman	181
rsr	185
sKGE	189
ssq	196
ubRMSE	199
valindex	203
ve	204
wNSE	208

wsNSE . . . . .	213
-----------------	-----

<b>Index</b>	<b>220</b>
--------------	------------

---

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

---

## 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: 0.7-0  
 Date: 2026-04-30  
 License: GPL >= 2  
 LazyLoad: yes  
 Packaged: Thu Apr 30 14:31:35 -04 2026 ; MZB  
 BuiltUnder: R version 4.6.0 (2026-04-24) – "Because it was There" ; aarch64-apple-darwin23

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  
**KGE1f** 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  
**JDKGE** Joint Divergence Kling-Gupta Efficiency  
**APFB** Annual Peak Flow Bias  
**HFB** High Flow Bias  
**LME** Liu-Mean Efficiency  
**LCE** Lee and Choi Efficiency  
**PMR** Proxy for Model Robustness  
**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.
- Ficchi, A.; Bavera, D.; Grimaldi, S.; Moschini, F.; Pistocchi, A.; Russo, C.; Salamon, P.; Toreti, A. (2026). Improving low and high flow simulations at once: An enhanced metric for hydrological model calibrations. *EGUsphere [preprint]*, <https://doi.org/10.5194/egusphere-2026-43>.
- 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. Available 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
- Lee, J. S., & Choi, H. I. (2022). A rebalanced performance criterion for hydrological model calibration. *Journal of Hydrology*, 606, 127372. <https://doi.org/10.1016/j.jhydrol.2021.127372>
- 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.
- Liu, D.; Chen, X.; Lian, Y.; Lou, Z. (2020). A new performance measure for hydrologic models. *Journal of Hydrology*, 590, 125488. doi:10.1016/j.jhydrol.2020.125488.
- 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.
- Moriassi, 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.
- Royer-Gaspard, P., Andreassian, V., and Thirel, G. (2021). Technical note: PMR - a proxy metric to assess hydrological model robustness in a changing climate. *Hydrology and Earth System Sciences*, 25, 5703–5716. doi:10.5194/hess-25-5703-2021.
- 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.
- Zambrano-Bigiarini, Mauricio (2024). hydroGOF: Goodness-of-fit functions for comparison of simulated and observed hydrological time series. doi:10.5281/zenodo.839854, R package version 0.6-0.1 . doi:10.5281/zenodo.839854, <https://cran.r-project.org/package=hydroGOF>.

#### 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 proposed 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**

<code>sim</code>	numeric, zoo, matrix or data.frame with simulated values
<code>obs</code>	numeric, zoo, matrix or data.frame with observed values
<code>na.rm</code>	a logical value indicating whether 'NA' should be stripped before the computation proceeds. When an 'NA' value is found at the <i>i</i> -th position in <b>obs</b> <b>OR</b> <b>sim</b> , the <i>i</i> -th value of <b>obs</b> <b>AND</b> <b>sim</b> are removed before the computation.
<code>start.month</code>	[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 <code>start.month=1</code> .
<code>out.PerYear</code>	logical value indicating whether the output should include the annual peak flow bias computed for each individual year or not.  Valid values are: -) FALSE: the output is a numeric with the mean annual peak flow bias. -) TRUE: the output is a list including both the overall APFB value and the individual yearly values.

fun	function to be applied to <code>sim</code> and <code>obs</code> in order to obtain transformed values thereof before computing this goodness-of-fit index. The first argument <b>MUST BE</b> a numeric vector with any name (e.g., <code>x</code> ), and additional arguments are passed using <code>...</code>
...	arguments passed to <code>fun</code> , in addition to the mandatory first numeric vector.
epsilon.type	argument used to define a numeric value to be added to both <code>sim</code> and <code>obs</code> before applying <code>fun</code> . It is was designed to allow the use of logarithm and other similar functions that do not work with zero values. Valid values of <code>epsilon.type</code> are: 1) "none": <code>sim</code> and <code>obs</code> are used by <code>fun</code> 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 <code>sim</code> and <code>obs</code> before applying <code>fun</code> , as described in Pushpalatha et al. (2012). 3) "otherFactor": the numeric value defined in the <code>epsilon.value</code> 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 <code>sim</code> and <code>obs</code> , before applying <code>fun</code> . 4) "otherValue": the numeric value defined in the <code>epsilon.value</code> argument is directly added to both <code>sim</code> and <code>obs</code> , before applying <code>fun</code> .
epsilon.value	-) when <code>epsilon.type="otherValue"</code> it represents the numeric value to be added to both <code>sim</code> and <code>obs</code> before applying <code>fun</code> . -) when <code>epsilon.type="otherFactor"</code> 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 <code>sim</code> and <code>obs</code> before applying <code>fun</code> .

## 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.

In the computation of this index, the annual peak flow is first identified for each hydrological year in both `sim` and `obs`. The mean of the resulting annual peak flow series is then computed separately for simulated and observed data.

The annual peak flow bias is defined as the absolute relative difference between the mean simulated annual peak flow and the mean observed annual peak flow.

$$APFB = \sqrt{\left(\frac{\mu_{peak Q_s}}{\mu_{peak Q_o}} - 1\right)^2}$$

where:

- $\mu_{peak Q_s}$ : mean of the simulated annual peak flow series
- $\mu_{peak Q_o}$ : mean of the observed annual peak flow series

The APFB metric ranges from 0 to Inf. The optimal value is 0, indicating perfect agreement between simulated and observed annual peak flows.

Essentially, the closer to 0, the more similar the magnitude of simulated and observed annual peak flows.

Because APFB focuses exclusively on annual maxima, it is particularly suitable for calibration tasks targeting flood estimation, extreme-flow simulation, or infrastructure design applications. However, because APFB evaluates only annual maxima, it does not assess overall hydrograph dynamics and is typically used in combination with complementary metrics (e.g., KGE or NSE) when broader performance evaluation is required.

### 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](#), [JDKGE](#), [PMR](#), [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 efficiency
NSE(sim=sim, obs=obs)

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

# APFB (Mizukami et al., 2019):
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 ( $r^2$ ) 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 <b>OR</b> sim, the i-th value of obs <b>AND</b> 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 <b>MUST BE</b> 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

$$br2 = |b|R^2, b \leq 1; br2 = \frac{R^2}{|b|}, b > 1$$

A model that systematically over or under-predicts all the time will still result in "good"  $R^2$  (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  $R^2$ )

## 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 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)

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 <b>OR</b> sim, the i-th value of obs <b>AND</b> 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 <b>MUST BE</b> 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 (Moriassi 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

Kitadinis, 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.

Moriassi, 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 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)

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**

<code>sim</code>	numeric, zoo, matrix or data.frame with simulated values
<code>obs</code>	numeric, zoo, matrix or data.frame with observed values
<code>na.rm</code>	a logical value indicating whether 'NA' should be stripped before the computation proceeds. When an 'NA' value is found at the <i>i</i> -th position in <code>obs</code> <b>OR</b> <code>sim</code> , the <i>i</i> -th value of <code>obs</code> <b>AND</b> <code>sim</code> are removed before the computation.
<code>fun</code>	function to be applied to <code>sim</code> and <code>obs</code> in order to obtain transformed values thereof before computing the Nash-Sutcliffe efficiency. The first argument <b>MUST BE</b> a numeric vector with any name (e.g., <code>x</code> ), and additional arguments are passed using <code>...</code>
<code>...</code>	arguments passed to <code>fun</code> , in addition to the mandatory first numeric vector.

<code>epsilon.type</code>	<p>argument used to define a numeric value to be added to both <code>sim</code> and <code>obs</code> before applying FUN.</p> <p>It is was designed to allow the use of logarithm and other similar functions that do not work with zero values.</p> <p>Valid values of <code>epsilon.type</code> are:</p> <ol style="list-style-type: none"> <li>1) "none": <code>sim</code> and <code>obs</code> are used by FUN without the addition of any numeric value.</li> <li>2) "Pushpalatha2012": one hundredth (1/100) of the mean observed values is added to both <code>sim</code> and <code>obs</code> before applying FUN, as described in Pushpalatha et al. (2012).</li> <li>3) "otherFactor": the numeric value defined in the <code>epsilon.value</code> 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 <code>sim</code> and <code>obs</code>, before applying FUN.</li> <li>4) "otherValue": the numeric value defined in the <code>epsilon.value</code> argument is directly added to both <code>sim</code> and <code>obs</code>, before applying FUN.</li> </ol>
<code>epsilon.value</code>	<p>-) when <code>epsilon.type="otherValue"</code> it represents the numeric value to be added to both <code>sim</code> and <code>obs</code> before applying fun.</p> <p>-) when <code>epsilon.type="otherFactor"</code> 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 <code>sim</code> and <code>obs</code> before applying fun.</p>

## 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 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 <b>OR</b> sim, the i-th value of obs <b>AND</b> 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 <b>MUST BE</b> 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. 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

$$c = 2$$

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

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

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

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

The Refined Index of Agreement (`dr`, Willmott et al., 2012) is a reformulation of the original 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) described 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 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)

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.

## 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", "NSE", "d",
           "dr", "r", "R2", "KGE", "LCE", "JDKGE", "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

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

<code>f<sub>type</sub></code>	Character indicating how many plots are desired by the user. Valid values are: -) <code>o</code> : only the original <code>sim</code> and <code>obs</code> time series are plotted -) <code>dm</code> : it assumes that <code>sim</code> and <code>obs</code> are daily time series and Daily and Monthly values are plotted -) <code>ma</code> : it assumes that <code>sim</code> and <code>obs</code> are daily or monthly time series and Monthly and Annual values are plotted -) <code>dma</code> : it assumes that <code>sim</code> and <code>obs</code> are daily time series and Daily, Monthly and Annual values are plotted -) <code>seasonal</code> : seasonal values are plotted. See <code>stype</code> and <code>season.names</code>
<code>FUN</code>	OPTIONAL, ONLY required when <code>f<sub>type</sub></code> is in <code>c('dm', 'ma', 'dma', 'seasonal')</code> . Function that have to be applied for transforming the original <code>ts</code> into monthly, annual or seasonal time step (e.g., for precipitation <code>FUN</code> MUST be <code>sum</code> , for temperature and flow time series, <code>FUN</code> MUST be <code>mean</code> )
<code>stype</code>	OPTIONAL, only used when <code>f<sub>type</sub>=seasonal</code> . character, indicating what weather seasons will be used for computing the output. Possible values are: -) <code>default</code> => "winter"= DJF = Dec, Jan, Feb; "spring"= MAM = Mar, Apr, May; "summer"= JJA = Jun, Jul, Aug; "autumn"= SON = Sep, Oct, Nov -) <code>FrenchPolynesia</code> => "winter"= DJFM = Dec, Jan, Feb, Mar; "spring"= AM = Apr, May; "summer"= JJAS = Jun, Jul, Aug, Sep; "autumn"= ON = Oct, Nov
<code>season.names</code>	OPTIONAL, only used when <code>f<sub>type</sub>=seasonal</code> . character of length 4 indicating the names of each one of the weather seasons defined by <code>stype</code> . These names are only used for plotting purposes
<code>gof.leg</code>	logical, indicating if several numerical goodness of fit have to be computed between <code>sim</code> and <code>obs</code> , and plotted as a legend on the graph. If <code>leg.gof=TRUE</code> , then <code>x</code> is considered as observed and <code>y</code> as simulated values (for some <code>gof</code> functions this is important).
<code>digits</code>	OPTIONAL, only used when <code>leg.gof=TRUE</code> . Numeric, representing the decimal places used for rounding the goodness-of-fit indexes.
<code>gofs</code>	character, with one or more strings indicating the goodness-of-fit measures to be shown in the legend of the plot when <code>gof.leg=TRUE</code> . Possible values when <code>f<sub>type</sub>!='seasonal'</code> are in <code>c("ME", "MAE", "MSE", "RMSE", "NRMSE", "PBIAS", "RSR", "rSD", "NSE", "mNSE", "rNSE", "d", "md", "rd", "cp", "r", "R2", "bR2", "KGE", "VE")</code> Possible values when <code>f<sub>type</sub>='seasonal'</code> are in <code>c("ME", "RMSE", "PBIAS", "RSR", "NSE", "d", "R2", "KGE", "VE")</code>
<code>legend</code>	character of length 2 to appear in the legend.
<code>leg.cex</code>	OPTIONAL. ONLY used when <code>leg.gof=TRUE</code> . Character expansion factor for drawing the legend, *relative* to current <code>'par("cex")</code> '. Used for text, and provides the default for <code>'pt.cex'</code> and <code>'title.cex'</code> . Default value = 1
<code>tick.tstep</code>	character, indicating the time step that have to be used for putting the ticks on the time axis. Valid values are: <code>auto</code> , <code>years</code> , <code>months</code> , <code>weeks</code> , <code>days</code> , <code>hours</code> , <code>minutes</code> , <code>seconds</code> .
<code>lab.tstep</code>	character, indicating the time step that have to be used for putting the labels on the time axis. Valid values are: <code>auto</code> , <code>years</code> , <code>months</code> , <code>weeks</code> , <code>days</code> , <code>hours</code> , <code>minutes</code> , <code>seconds</code> .

lab.fmt	Character indicating the format to be used for the label of the axis. See lab.fmt in <a href="#">drawTimeAxis</a> .
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 is 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 <a href="#">par</a> .
cex.lab	numeric, representing the magnification to be used for x and y labels relative to the current setting of 'cex'. See <a href="#">par</a> .
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 ( $-\text{Inf} \leq \text{PBIAS} \leq \text{Inf} [\%]$ )
RSR	Ratio of RMSE to the Standard Deviation of the Observations, $\text{RSR} = \text{rms} / \text{sd}(\text{obs})$ . ( $0 \leq \text{RSR} \leq +\text{Inf}$ )
rSD	Ratio of Standard Deviations, $\text{rSD} = \text{sd}(\text{sim}) / \text{sd}(\text{obs})$
NSE	Nash-Sutcliffe Efficiency ( $-\text{Inf} \leq \text{NSE} \leq 1$ )
mNSE	Modified Nash-Sutcliffe Efficiency ( $-\text{Inf} \leq \text{mNSE} \leq 1$ )
rNSE	Relative Nash-Sutcliffe Efficiency ( $-\text{Inf} \leq \text{rNSE} \leq 1$ )
wNSE	Weighted Nash-Sutcliffe Efficiency ( $-\text{Inf} \leq \text{wNSE} \leq 1$ )
wsNSE	Weighted Seasonal Nash-Sutcliffe Efficiency ( $-\text{Inf} \leq \text{wsNSE} \leq 1$ )
d	Index of Agreement ( $0 \leq d \leq 1$ )
dr	Refined Index of Agreement ( $-1 \leq \text{dr} \leq 1$ )
md	Modified Index of Agreement ( $0 \leq \text{md} \leq 1$ )
rd	Relative Index of Agreement ( $0 \leq \text{rd} \leq 1$ )
cp	Persistence Index ( $0 \leq \text{cp} \leq 1$ )
r	Pearson Correlation coefficient ( $-1 \leq r \leq 1$ )
R2	Coefficient of Determination ( $0 \leq \text{R2} \leq 1$ )
bR2	R2 multiplied by the coefficient of the regression line between <code>sim</code> and <code>obs</code> ( $0 \leq \text{bR2} \leq 1$ )
VE	Volumetric efficiency between <code>sim</code> and <code>obs</code> ( $-\text{Inf} \leq \text{VE} \leq 1$ )
KGE	Kling-Gupta efficiency between <code>sim</code> and <code>obs</code> ( $-\text{Inf} \leq \text{KGE} \leq 1$ )
KGE1f	Kling-Gupta Efficiency for low values between <code>sim</code> and <code>obs</code> ( $-\text{Inf} \leq \text{KGE1f} \leq 1$ )
KGEmp	Non-parametric version of the Kling-Gupta Efficiency between <code>sim</code> and <code>obs</code> ( $-\text{Inf} \leq \text{KGEmp} \leq 1$ )
KGEkm	Knowable Moments Kling-Gupta Efficiency between <code>sim</code> and <code>obs</code> ( $-\text{Inf} \leq \text{KGEmp} \leq 1$ )

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

sKGE	Split Kling-Gupta Efficiency between <code>sim</code> and <code>obs</code> ( $-\text{Inf} \leq \text{sKGE} \leq 1$ ). Only computed when both <code>sim</code> and <code>obs</code> are zoo objects
APFB	Annual Peak Flow Bias ( $0 \leq \text{APFB} \leq \text{Inf}$ )
HFB	High Flow Bias ( $0 \leq \text{HFB} \leq \text{Inf}$ )
r.Spearman	Spearman Correlation coefficient ( $-1 \leq \text{r.Spearman} \leq 1$ ). Only computed when <code>do.spearman=TRUE</code>
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.
- Ficchi, A.; Bavera, D.; Grimaldi, S.; Moschini, F.; Pistocchi, A.; Russo, C.; Salamon, P.; Toreti, A. (2026). Improving low and high flow simulations at once: An enhanced metric for hydrological model calibrations. *EGUsphere* [preprint], <https://doi.org/10.5194/egusphere-2026-43>.
- 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. Available 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

Lee, J. S.; Choi, H. I. (2022). A rebalanced performance criterion for hydrological model calibration. *Journal of Hydrology*, 606, 127372. <https://doi.org/10.1016/j.jhydrol.2021.127372>

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.

Liu, D.; Chen, X.; Lian, Y.; Lou, Z. (2020). A new performance measure for hydrologic models. *Journal of Hydrology*, 590, 125488. doi:10.1016/j.jhydrol.2020.125488.

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.

- Royer-Gaspard, P., Andreassian, V., and Thirel, G. (2021). Technical note: PMR - a proxy metric to assess hydrological model robustness in a changing climate. *Hydrology and Earth System Sciences*, 25, 5703–5716. doi:10.5194/hess-25-5703-2021.
- 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](#), [KGE1f](#), [KGE1p](#), [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,
     do.pmr=FALSE, j=1, lambda=0.95, norm="sd", s=c(1,1,1,1),
     method=c("2009", "2012", "2021"), lQ.thr=0.6, hQ.thr=0.1, start.month=1,
     k=NULL, min.years=5, days.per.year=365,
     density.method=c("hist", "kde", "wasserstein"),
     nbins="paper", timestep=86400, kde.n.grid=512, wasserstein.n.quantiles=512,
     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,
     do.pmr=FALSE, j=1, lambda=0.95, norm="sd", s=c(1,1,1,1),
     method=c("2009", "2012", "2021"), lQ.thr=0.6, hQ.thr=0.1, start.month=1,
     k=NULL, min.years=5, days.per.year=365,
     density.method=c("hist", "kde", "wasserstein"),

```

```

nbins="paper", timestep=86400, kde.n.grid=512, wasserstein.n.quantiles=512,
  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,
  do.pmr=FALSE, j=1, lambda=0.95, norm="sd", s=c(1,1,1,1),
  method=c("2009", "2012", "2021"), lQ.thr=0.6, hQ.thr=0.1, start.month=1,
  k=NULL, min.years=5, days.per.year=365,
  density.method=c("hist", "kde", "wasserstein"),
nbins="paper", timestep=86400, kde.n.grid=512, wasserstein.n.quantiles=512,
  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,
  do.pmr=FALSE, j=1, lambda=0.95, norm="sd", s=c(1,1,1,1),
  method=c("2009", "2012", "2021"), lQ.thr=0.6, hQ.thr=0.1, start.month=1,
  k=NULL, min.years=5, days.per.year=365,
  density.method=c("hist", "kde", "wasserstein"),
nbins="paper", timestep=86400, kde.n.grid=512, wasserstein.n.quantiles=512,
  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>i</i> -th position in obs <b>OR</b> sim, the <i>i</i> -th value of obs <b>AND</b> 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 ( <a href="#">pbiasfdc</a> ) has to be computed. The default is FALSE.
do.pmr	logical. Indicates if the Proxy for Model Robustness ( <a href="#">PMR</a> ) has to be computed. The default is FALSE.
j	argument passed to the <a href="#">mNSE</a> and <a href="#">wsNSE</a> functions.
lambda	argument passed to the <a href="#">wsNSE</a> function.
norm	argument passed to the <a href="#">nrmse</a> function
s	argument passed to the <a href="#">KGE</a> , <a href="#">KGE1f</a> , <a href="#">sKGE</a> , <a href="#">KGEkm</a> and <a href="#">JDKGE</a> functions. The fourth element in <i>s</i> is only used in the <a href="#">JDKGE</a> function; while <a href="#">KGE</a> , <a href="#">KGE1f</a> , <a href="#">sKGE</a> , and <a href="#">KGEkm</a> only uses the first three elements in <i>s</i> .

method	argument passed to the <code>KGE</code> , <code>KGE1f</code> , <code>sKGE</code> and <code>KGEkm</code> functions.
lQ.thr	[OPTIONAL]. Only used for the computation of the <code>pbiasFDC</code> % (with the <code>pbiasfdc</code> function) and the weighted seasonal Nash-Sutcliffe Efficiency (with the <code>wsNSE</code> function).
hQ.thr	[OPTIONAL]. Only used for the computation of the <code>pbiasFDC</code> % (with the <code>pbiasfdc</code> function), the high flow bias (HFB, with the <code>HFB</code> function) and the weighted seasonal Nash-Sutcliffe Efficiency (with the <code>wsNSE</code> function).
start.month	[OPTIONAL]. Only used for the computation of the split KGE ( <code>sKGE</code> ), 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 <code>start.month=1</code> .
k	Only used for the computation of the Proxy for Model Robustness (PMR).  integer value representing the length of the moving window (number of time steps) used to compute the bias over sub-periods.  The <code>k</code> argument should reflect the temporal scale at which robustness is intended to be evaluated, and therefore depends primarily on the time resolution of the data. Royer-Gaspard et al. (2021) recommended to use multi-year windows, typically in the range of 3 to 5 years, to ensure that each sub-period captures meaningful hydroclimatic variability while still allowing enough windows for comparison.
min.years	Only used for the computation of the Proxy for Model Robustness (PMR).  Numeric, only used when the user does not explicitly define the value of <code>k</code> , i.e., when <code>k=NULL</code> .  Minimum numbers of years used to ensure that each sub-period used in the computation of PMR captures meaningful hydroclimatic variability while still allowing enough windows for comparison. By default, <code>min.years=5</code> .
days.per.year	Only used for the computation of the Proxy for Model Robustness (PMR).  Numeric, only used when the user does not explicitly define the value of <code>k</code> , i.e., when <code>k=NULL</code> .  Number of days in a year. A value of <code>Use 365.25</code> is recommended instead of the default value of <code>365</code> when <code>sim</code> and <code>obs</code> are long climatological series.
density.method	Only used for the computation of the Joint Divergence Kling-Gupta Efficiency (JDKGE).  Character, representing the method used to compute the divergence component. "hist" uses the paper-faithful histogram-based Jensen-Shannon divergence, "kde" uses a common-grid kernel density estimate followed by Jensen-Shannon divergence, and "wasserstein" uses a Wasserstein-distance similarity on log-flows.

nbins	Only used for the computation of the Joint Divergence Kling-Gupta Efficiency (JDKGE).
	Character, representing the binning rule used by the histogram divergence component. The default "paper" uses the procedure described by Ficchi et al. (2026). This argument is ignored for <code>density.method="kde"</code> and <code>density.method="wasserstein"</code> .
timestep	Only used for the computation of the Joint Divergence Kling-Gupta Efficiency (JDKGE).
	Numeric, representing the sampling time step in seconds used by the paper's bin-count adjustment. For zoo inputs this is inferred from the time index when omitted. The default for plain numeric vectors is one day (86400 seconds).
kde.n.grid	Only used for the computation of the Joint Divergence Kling-Gupta Efficiency (JDKGE).
	Integer, number of grid points used when <code>density.method="kde"</code> . Larger values provide a finer common support grid at higher computational cost.
wasserstein.n.quantiles	Only used for the computation of the Joint Divergence Kling-Gupta Efficiency (JDKGE).
	Integer, number of quantile levels used to approximate the first Wasserstein distance when <code>density.method="wasserstein"</code> . Larger values provide a finer approximation at higher computational cost.
digits	decimal places used for rounding the goodness-of-fit indexes.
fun	function to be applied to <code>sim</code> and <code>obs</code> 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., <code>x</code> ), and additional arguments are passed using <code>...</code>
...	arguments passed to <code>fun</code> , in addition to the mandatory first numeric vector.
epsilon.type	argument used to define a numeric value to be added to both <code>sim</code> and <code>obs</code> before applying <code>fun</code> . It is was designed to allow the use of logarithm and other similar functions that do not work with zero values. Valid values of <code>epsilon.type</code> are: 1) "none": <code>sim</code> and <code>obs</code> are used by FUN without the addition of any numeric value. 2) "Pushpalatha2012": one hundredth (1/100) of the mean observed values is added to both <code>sim</code> and <code>obs</code> before applying FUN, as described in Pushpalatha et al. (2012). 3) "otherFactor": the numeric value defined in the <code>epsilon.value</code> 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 <code>sim</code> and <code>obs</code> , before applying FUN. 4) "otherValue": the numeric value defined in the <code>epsilon.value</code> argument is directly added to both <code>sim</code> and <code>obs</code> , 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 )
KGE1f	Kling-Gupta Efficiency for low values between sim and obs ( -Inf <= KGE1f <= 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 with sub-annual temporal frequency:

sKGE	Split Kling-Gupta Efficiency between sim and obs ( -Inf <= sKGE <= 1 )
APFB	Annual Peak Flow Bias ( 0 <= APFB <= Inf )
HBF	High Flow Bias ( 0 <= HFB <= Inf )

The following outputs are only produced when default values of a specific argument is changed by the user:

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. Only computed when do.pbfdc=FALSE

### 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. Available 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.2013.12.001
- 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](#), [KGE1f](#), [KGE1p](#), [KGE1m](#), [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)

## Not run:
#####
# 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",]

# 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

*Median annual high-flow bias*


---

### Description

Median annual high-flow bias between `sim` and `obs`, with treatment of missing values and explicit focus on the reproduction of high-flow events.

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**

<code>sim</code>	numeric, zoo, matrix or data.frame with simulated values
<code>obs</code>	numeric, zoo, matrix or data.frame with observed values
<code>na.rm</code>	a logical value indicating whether 'NA' should be stripped before the computation proceeds. When an 'NA' value is found at the <i>i</i> -th position in <code>obs</code> <b>OR</b> <code>sim</code> , the <i>i</i> -th value of <code>obs</code> <b>AND</b> <code>sim</code> are removed before the computation.
<code>hQ.thr</code>	numeric, representing the exceedence probability used to identify high flows in <code>obs</code> . All values in <code>obs</code> that are equal or higher than <code>quantile(obs, probs=(1-hQ.thr))</code> are considered as high flows. By default <code>hQ.thr=0.1</code> . On the other hand, the high values in <code>sim</code> are those located at the same <i>i</i> -th position than the <i>i</i> -th value of the <code>obs</code> deemed as high flows.
<code>start.month</code>	[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 <code>start.month=1</code> .
<code>out.PerYear</code>	logical, indicating whether the output of this function has to include the median annual high-flows bias obtained for the individual years in <code>sim</code> and <code>obs</code> or not.
<code>fun</code>	function to be applied to <code>sim</code> and <code>obs</code> in order to obtain transformed values thereof before computing this goodness-of-fit index. The first argument <b>MUST BE</b> a numeric vector with any name (e.g., <code>x</code> ), and additional arguments are passed using <code>...</code>
<code>...</code>	arguments passed to <code>fun</code> , in addition to the mandatory first numeric vector.
<code>epsilon.type</code>	argument used to define a numeric value to be added to both <code>sim</code> and <code>obs</code> before applying <code>fun</code> . It is was designed to allow the use of logarithm and other similar functions that do not work with zero values. Valid values of <code>epsilon.type</code> are: 1) "none": <code>sim</code> and <code>obs</code> are used by <code>fun</code> 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 <code>sim</code> and <code>obs</code> before applying <code>fun</code> , as described in Pushpalatha et al. (2012). 3) "otherFactor": the numeric value defined in the <code>epsilon.value</code> 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 <code>sim</code> and <code>obs</code> , before applying <code>fun</code> . 4) "otherValue": the numeric value defined in the <code>epsilon.value</code> argument is directly added to both <code>sim</code> and <code>obs</code> , before applying <code>fun</code> .
<code>epsilon.value</code>	-) when <code>epsilon.type="otherValue"</code> it represents the numeric value to be added to both <code>sim</code> and <code>obs</code> before applying <code>fun</code> .

-) 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 a goodness-of-fit metric designed to support the calibration and evaluation of hydrological models with specific emphasis on the reproduction of high-flow conditions.

The HFB ranges from 0 to  $\text{Inf}$ , with an optimal value of 0. Values close to 0 indicate that the simulated high flows closely match the observed high flows, whereas larger values indicate increasing discrepancies between the simulated and observed high-flow magnitudes.

The current formulation of the HFB function was proposed by Zambrano-Bigiarini (2026), inspired by the annual peak-flow bias (APFB) objective function proposed by Mizukami et al. (2019). However, HFB differs from APFB in four important aspects:

- 1) Instead of considering only the single observed annual peak flow in each year, it considers all high flows in each year, where "high flows" are defined as all values equal to or greater than a user-defined exceedance probability threshold of the observed values. By default, high flows correspond to the upper 10% of observed flows (i.e., `hQ.thr=0.1`).
- 2) Instead of selecting simulated high flows independently of the observed events, it evaluates simulated flows occurring at the same time steps as the observed high flows, thereby preserving temporal correspondence between observed and simulated events.
- 3) For each year, the metric uses the median of the individual high-flow ratios rather than a single annual peak-flow value, providing a more robust summary of high-flow bias within that year.
- 4) When computing the final performance value, the metric uses the median of the annual values instead of the mean, reducing the influence of extreme years and improving robustness when the distribution of annual biases is asymmetric.

Mathematically, the annual high-flow bias for year  $y$  is defined as:

$$HFB_y = \sqrt{\left(\frac{\text{median}(Q_{y,i}^{sim})}{\text{median}(Q_{y,i}^{obs})} - 1\right)^2}$$

where  $Q_{y,i}^{sim}$  and  $Q_{y,i}^{obs}$  are the simulated and observed flows corresponding to the set of high-flow events  $i$  occurring in year  $y$ .

The overall HFB value is then computed as:

$$HFB = \text{median}(HFB_y)$$

This formulation yields a non-negative bias metric, with the minimum value of 0 representing perfect agreement between simulated and observed high flows.

**Value**

If `out.PerYear=FALSE`: 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 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**

Zambrano-Bigiarini, Mauricio (2026). *hydroGOF: Goodness-of-fit functions for comparison of simulated and observed hydrological time series*. R package version 0.7-0. URL:<https://cran.r-project.org/package=hydroGOF>. doi:10.32614/CRAN.package.hydroGOF.

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](#), [JDKGE](#), [PMR](#), [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 efficiency
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
sim1 <- sqrt(sim+1)
obs1 <- sqrt(obs+1)
HFB(sim=sim1, obs=obs1)
```

---

JDKGE

*Joint Divergence Kling-Gupta Efficiency*


---

## Description

Joint Divergence Kling-Gupta Efficiency between `sim` and `obs`, with treatment of missing values.

This implementation follows the technical formulation described by Ficchi et al. (2026), using by default the KGE-2012 variability term, and computing the distributional component from a histogram-based Jensen-Shannon divergence applied to log-transformed flows after paper-specific zero handling. However, this function also allows JDKGE-style variants based on the 2009 and 2021 KGE formulations, different methods for the distributional component, and different methods for the handling of low streamflow values.

Results by Ficchi et al. (2026) show that calibrations using JDKGE significantly improve **low-flow simulations** compared to KGE, NSE and other competitors, while maintaining comparable or improved performance in other regimes, including **high flows**.

## Usage

```
JDKGE(sim, obs, ...)

## Default S3 method:
JDKGE(sim, obs, na.rm=TRUE, s=c(1,1,1,1),
      method=c("2012", "2009", "2021"), out.type=c("single", "full"),
      density.method=c("hist", "kde", "wasserstein"), nbins="paper",
      timestep=86400, kde.n.grid=512, wasserstein.n.quantiles=512, fun=NULL, ...,
      epsilon.type=c("otherValue", "none", "Pushpalatha2012", "otherFactor"),
      epsilon.value=NA)

## S3 method for class 'data.frame'
JDKGE(sim, obs, na.rm=TRUE, s=c(1,1,1,1),
```

```

    method=c("2012", "2009", "2021"), out.type=c("single", "full"),
    density.method=c("hist", "kde", "wasserstein"), nbins="paper",
    timestep=86400, kde.n.grid=512, wasserstein.n.quantiles=512, fun=NULL, ...,
    epsilon.type=c("otherValue", "none", "Pushpalatha2012", "otherFactor"),
    epsilon.value=NA)

## S3 method for class 'matrix'
JDKGE(sim, obs, na.rm=TRUE, s=c(1,1,1,1),
    method=c("2012", "2009", "2021"), out.type=c("single", "full"),
    density.method=c("hist", "kde", "wasserstein"), nbins="paper",
    timestep=86400, kde.n.grid=512, wasserstein.n.quantiles=512, fun=NULL, ...,
    epsilon.type=c("otherValue", "none", "Pushpalatha2012", "otherFactor"),
    epsilon.value=NA)

## S3 method for class 'zoo'
JDKGE(sim, obs, na.rm=TRUE, s=c(1,1,1,1),
    method=c("2012", "2009", "2021"), out.type=c("single", "full"),
    density.method=c("hist", "kde", "wasserstein"), nbins="paper",
    timestep=86400, kde.n.grid=512, wasserstein.n.quantiles=512, fun=NULL, ...,
    epsilon.type=c("otherValue", "none", "Pushpalatha2012", "otherFactor"),
    epsilon.value=NA)

```

## Arguments

<code>sim</code>	numeric, zoo, matrix or data.frame with simulated values.
<code>obs</code>	numeric, zoo, matrix or data.frame with observed values.
<code>na.rm</code>	logical value indicating whether missing paired values should be removed before computing the metric.
<code>s</code>	numeric of length 4 with scaling factors for the Euclidean distance in criteria space. If <code>s</code> differs from <code>c(1, 1, 1, 1)</code> , then <code>sum(s)</code> must be equal to 1.
<code>method</code>	character string indicating the Kling-Gupta formulation used by JDKGE. Valid values are "2012" (default, the paper's KGE' formulation), "2009", and "2021".
<code>out.type</code>	character string indicating the output format. Use "single" to return the JDKGE value only, or "full" to also return the four diagnostic components.
<code>density.method</code>	character, representing the method used to compute the divergence component. "hist" uses the paper-faithful histogram-based Jensen-Shannon divergence, "kde" uses a common-grid kernel density estimate followed by Jensen-Shannon divergence, and "wasserstein" uses a Wasserstein-distance similarity on log-flows.
<code>nbins</code>	character, representing the binning rule used by the histogram divergence component. The default "paper" uses the procedure described by Ficchi et al. (2026). This argument is ignored for <code>density.method="kde"</code> and <code>density.method="wasserstein"</code> .
<code>timestep</code>	numeric, representing the sampling time step in seconds used by the paper's bin-count adjustment. For zoo inputs this is inferred from the time index when omitted. The default for plain numeric vectors is one day (86400 seconds).
<code>kde.n.grid</code>	integer, number of grid points used when <code>density.method="kde"</code> . Larger values provide a finer common support grid at higher computational cost.

wasserstein.n.quantiles	integer, number of quantile levels used to approximate the first Wasserstein distance when <code>density.method="wasserstein"</code> . Larger values provide a finer approximation at higher computational cost.
fun	optional function applied to <code>sim</code> and <code>obs</code> before computing JDKGE. The first argument of <code>fun</code> must be a numeric vector.
...	additional arguments passed to <code>fun</code> .
epsilon.type	rule used for zero-flow handling in the internal log-based divergence component, and also passed to <code>preproc</code> when <code>fun</code> needs an offset before transformation. The default is <code>"otherValue"</code> .
epsilon.value	numeric value used when <code>epsilon.type="otherValue"</code> or <code>epsilon.type="otherFactor"</code> . When <code>epsilon.type="otherValue"</code> and <code>epsilon.value=NA</code> (the default), the value described by Ficchi et al. (2026), $\epsilon = \min(10^{-6}, 10^{-1} \min(c))$ , is computed internally.

## Details

JDKGE combines four components:

1. the Pearson correlation coefficient  $r$ ,
2. the variability term, defined as  $\gamma = (\sigma_s/\mu_s)/(\sigma_o/\mu_o)$  for `method="2012"` and as  $\alpha = \sigma_s/\sigma_o$  for `method="2009"` and `method="2021"`,
3. the bias term, defined as  $\beta = \mu_s/\mu_o$  for `method="2009"` and `method="2012"`, and as  $\beta_{2021} = (\mu_s - \mu_o)/\sigma_o$  for `method="2021"`, and
4. the distributional similarity component  $\Delta$ .

For the divergence component, this implementation follows the paper's workflow:

1. exact zeros are replaced according to `epsilon.type`. With the default `"otherValue"` and `epsilon.value=NA`,  $\epsilon = \min(10^{-6}, 10^{-1} \min(c))$ , where  $c$  is the set of strictly positive simulated and observed values,
2. the transformed values  $\log(x)$  are binned using a histogram,
3. the Freedman-Diaconis width is lower-bounded by  $h_{min} = \min(10^2\epsilon, 10^{-1})$ ,
4. the number of bins is adjusted by the time-scale factor and clipped to the interval  $[25, 100]$ ,
5. additive smoothing with  $\alpha = \epsilon$  is applied to the empirical densities, and
6. Jensen-Shannon divergence is computed with base-2 logarithms.

For `density.method="kde"`, simulated and observed log-flows are smoothed with Gaussian kernel density estimates evaluated on a common grid over the pooled support using a shared bandwidth estimated from the pooled sample. Jensen-Shannon divergence is then computed from the two resulting probability vectors.

For `density.method="wasserstein"`, the log-flow distributions are compared with the first Wasserstein distance computed from empirical quantiles. The resulting distance is converted into a similarity component through

$$\Delta = \exp(-W_1/s_w)$$

where  $s_w$  is a robust scale estimated from the pooled log-flows using the interquartile range, with fallback to the standard deviation when needed.

The metric is then computed as:

$$JDKGE = 1 - \sqrt{(s[1](r - 1))^2 + (s[2](vr - 1))^2 + (s[3](br - 1))^2 + (s[4](\Delta - 1))^2}$$

Joint Divergence Kling-Gupta efficiencies range from  $-\infty$  to 1. Values closer to 1 indicate stronger agreement between simulated and observed values across correlation, variability, bias, and distributional similarity.

### Value

If `out.type="single"`: numeric with the Joint Divergence Kling-Gupta Efficiency between `sim` and `obs`. If `sim` and `obs` are matrices, the output is a vector with one efficiency value per column pair.

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

`JDKGE.value` numeric with the Joint Divergence Kling-Gupta Efficiency.

`JDKGE.elements` numeric with 4 elements: 'r', the selected bias term, the selected variability term, and 'Delta'.

### References

Ficchi, A.; Bavera, D.; Grimaldi, S.; Moschini, F.; Pistocchi, A.; Russo, C.; Salamon, P.; Toreti, A. (2026). Improving low and high flow simulations at once: An enhanced metric for hydrological model calibrations. *EGUsphere [preprint]*, <https://doi.org/10.5194/egusphere-2026-43>.

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.

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.

### See Also

[NSE](#), [wNSE](#), [wsNSE](#), [HFB](#), [JDKGE](#), [KGE1f](#), [rNSE](#), [gof](#), [ggof](#)

### Examples

```
# Example 0.1: ideal case
obs <- 1:10
sim <- 1:10

JDKGE(sim, obs)

#####
# Example 1: simulated values equal to twice the observations
```

```

sim <- 2*obs
JDKGE(sim=sim, obs=obs, out.type="full")

#####
# Example 2: using kernel density estimation, instead of histograms (the default)
JDKGE(sim=sim, obs=obs, density.method="kde")
JDKGE(sim=sim, obs=obs, density.method="kde", kde.n.grid=1024)
JDKGE(sim=sim, obs=obs, density.method="wasserstein")
JDKGE(sim=sim, obs=obs, density.method="wasserstein", wasserstein.n.quantiles=1024)

#####
# Example 3: Looking at the difference between JDKGE and KGE, both with '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")

# JDKGE (Ficchi et al., 2026):
JDKGE(sim=sim, obs=obs, method="2009", out.type="full")

# KGE 2012
KGE(sim=sim, obs=obs, method="2012", out.type="full")

# JDKGE (Ficchi et al., 2026):
JDKGE(sim=sim, obs=obs, method="2012", out.type="full")

# KGE 2021
KGE(sim=sim, obs=obs, method="2021", out.type="full")

# JDKGE (Ficchi et al., 2026):
JDKGE(sim=sim, obs=obs, method="2021", out.type="full")

#####
# Example 4:
# 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 'JDKGE' for the "best" (unattainable) case
JDKGE(sim=sim, obs=obs)

#####
# Example 5: JDKGE 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)

JDKGE(sim=sim, obs=obs)

#####
# Example 6: JDKGE 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)}

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

# Verifying the previous value
sim1 <- sqrt(sim+1)
obs1 <- sqrt(obs+1)
JDKGE(sim=sim1, obs=obs1)

#####
# Example 7: JDKGE 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)

JDKGE(sim=SIM, obs=OBS)

```

---

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, please 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 <b>OR</b> sim, the i-th value of obs <b>AND</b> 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 <code>sim</code> values to the coefficient of variation of <code>obs</code> . See Kling et al. (2012).
	-) 2021: the bias is defined as 'Beta', the ratio of <code>mean(sim)</code> minus <code>mean(obs)</code> to the standard deviation of <code>obs</code> . The variability is defined as 'Alpha', the ratio of the standard deviation of <code>sim</code> values to the standard deviation of <code>obs</code> . See Tang et al. (2021).
<code>out.type</code>	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: -) <code>single</code> : the output is a numeric with the Kling-Gupta efficiency only. -) <code>full</code> : 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 ( <code>'r'</code> ), the ratio between the mean of the simulated values to the mean of observations ( <code>'Beta'</code> ), and the variability measure ( <code>'Gamma'</code> or <code>'Alpha'</code> , depending on the value of <code>method</code> ).
<code>fun</code>	function to be applied to <code>sim</code> and <code>obs</code> 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., <code>x</code> ), and additional arguments are passed using <code>...</code>
<code>...</code>	arguments passed to <code>fun</code> , in addition to the mandatory first numeric vector.
<code>epsilon.type</code>	argument used to define a numeric value to be added to both <code>sim</code> and <code>obs</code> before applying <code>fun</code> . It is was designed to allow the use of logarithm and other similar functions that do not work with zero values. Valid values of <code>epsilon.type</code> are: 1) <code>"none"</code> : <code>sim</code> and <code>obs</code> are used by <code>fun</code> without the addition of any numeric value. 2) <code>"Pushpalatha2012"</code> : one hundredth (1/100) of the mean observed values is added to both <code>sim</code> and <code>obs</code> before applying <code>fun</code> , as described in Pushpalatha et al. (2012). 3) <code>"otherFactor"</code> : the numeric value defined in the <code>epsilon.value</code> 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 <code>sim</code> and <code>obs</code> , before applying <code>fun</code> . 4) <code>"otherValue"</code> : the numeric value defined in the <code>epsilon.value</code> argument is directly added to both <code>sim</code> and <code>obs</code> , before applying <code>fun</code> .
<code>epsilon.value</code>	-) when <code>epsilon.type="otherValue"</code> it represents the numeric value to be added to both <code>sim</code> and <code>obs</code> before applying <code>fun</code> . -) when <code>epsilon.type="otherFactor"</code> 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 <code>sim</code> and <code>obs</code> before applying <code>fun</code> .

## 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.

$$KGE = 1 - ED$$

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

$r = \text{Pearsonproduct} - \text{momentcorrelationcoefficient}$

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

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

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

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

## 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**

[KGE1f](#), [sKGE](#), [KGE<sub>np</sub>](#), [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 efficiency (Gupta and Kling, 2009)
KGE(sim=sim, obs=obs, method="2009", out.type="full")

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

# KGE'': Kling-Gupta efficiency 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 efficiency (Gupta and Kling, 2009)
KGE(sim=sim, obs=obs, method="2009", out.type="full")

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

# KGE'': Kling-Gupta efficiency 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**

<code>sim</code>	numeric, zoo, matrix or data.frame with simulated values
<code>obs</code>	numeric, zoo, matrix or data.frame with observed values
<code>s</code>	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., <code>s</code> elements are used for adjusting the emphasis on different components. The first element 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
<code>na.rm</code>	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 <code>obs</code> <b>OR</b> <code>sim</code> , the $i$ -th value of <code>obs</code> <b>AND</b> <code>sim</code> are removed before the computation.
<code>method</code>	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 <code>sim</code> values to the coefficient of variation of <code>obs</code> . See Pizarro and Jorquera (2024) and Kling et al. (2012). -) 2009: the variability is defined as 'Alpha', the ratio of the standard deviation of <code>sim</code> values to the standard deviation of <code>obs</code> . This is the default option. See Gupta et al. (2009). -) 2021: the bias is defined as 'Beta', the ratio of $\text{mean}(\text{sim}) - \text{mean}(\text{obs})$ to the standard deviation of <code>obs</code> . The variability is defined as 'Alpha', the ratio of the standard deviation of <code>sim</code> values to the standard deviation of <code>obs</code> . See Tang et al. (2021).
<code>out.type</code>	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: -) <code>single</code> : the output is a numeric with the Kling-Gupta efficiency only. -) <code>full</code> : 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 <code>method</code> ).
<code>fun</code>	function to be applied to <code>sim</code> and <code>obs</code> in order to obtain transformed values thereof before computing the Kling-Gupta efficiency. The first argument <b>MUST BE</b> a numeric vector with any name (e.g., <code>x</code> ), and additional arguments are passed using <code>...</code>
<code>...</code>	arguments passed to <code>fun</code> , in addition to the mandatory first numeric vector.
<code>epsilon.type</code>	argument used to define a numeric value to be added to both <code>sim</code> and <code>obs</code> before applying <code>fun</code> . It is was designed to allow the use of logarithm and other similar functions that do not work with zero values. Valid values of <code>epsilon.type</code> are:

- 1) "none": sim and obs are used by fun without the addition of any numeric 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 (KGE<sub>km</sub>) 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  $KGE_{km}$ , the dispersion is quantified using knowable moments (computed over ordered values of the samples in ascending order) instead of the standard deviation, while retaining the decomposition into correlation, variability, and bias components.

The general formulation of Kling–Gupta Efficiency with knowable moments ( $KGE_{km}$ ) is:

$$KGE_{km} = 1 - \sqrt{[s_1(r - 1)]^2 + [s_2(vr - 1)]^2 + [s_3(br - 1)]^2}$$

where  $r$  is the Pearson product–moment correlation coefficient between simulated ( $Q_t^{sim}$ ) and observed ( $Q_t^{obs}$ ) values,  $vr$  is the variability ratio,  $br$  is the bias ratio, and  $s = (s_1, s_2, s_3)$  is a vector of non-negative scaling factors that control the relative importance of each component.

Dispersion is computed from the second knowable moment. For a sample  $x_1, x_2, \dots, x_n$ , the second knowable moment is defined as:

$$K_2 = \frac{1}{n(n-1)} \sum_{i=1}^n 2(i-1)x_{(i)}$$

where  $x_{(i)}$  denotes the **ordered values of the sample in ascending order**. The corresponding dispersion measure is:

$$\sigma_{km} = \sqrt{2K_2}$$

The variability ratio depends on the selected method:

- For method = "2009", variability is defined as the ratio of knowable-moment dispersions:

$$vr = \alpha = \frac{\sigma_{km}^{sim}}{\sigma_{km}^{obs}}$$

- For method = "2012", variability is defined as the ratio of coefficients of variation:

$$vr = \gamma = \frac{\sigma_{km}^{sim} / \mu^{sim}}{\sigma_{km}^{obs} / \mu^{obs}}$$

- For method = "2021", variability is defined as in the 2009 formulation:

$$vr = \alpha = \frac{\sigma_{km}^{sim}}{\sigma_{km}^{obs}}$$

The bias component also depends on the selected method:

- For method = "2009" and method = "2012":

$$br = \beta = \frac{\mu^{sim}}{\mu^{obs}}$$

- For method = "2021":

$$br = \beta_{2021} = \frac{\mu^{sim} - \mu^{obs}}{\sigma_{km}^{obs}}$$

In the same line that the traditional Kling-Gupta efficiency, the ( $KGE_{km}$ ) ranges from -Inf to 1. Essentially, the closer to 1, the more similar sim and obs are.

As with other KGE-type metrics, the statistic integrates information about correlation, variability, and bias into a single performance measure while allowing explicit control over the relative contribution of each component through the scaling factors  $s$ .

## 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<sub>km</sub>.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<sub>km</sub>.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](#), [KGE1f](#), [sKGE](#), [KGE<sub>np</sub>](#), [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 efficiency 2012 (Kling et al.,2012)
KGE(sim=sim, obs=obs, method="2012")

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

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

# Traditional Nash-Sutcliffe efficiency (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 efficiency 2012 (Kling et al.,2012)
KGE(sim=sim, obs=obs, method="2012")

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

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

# Traditional Nash-Sutcliffe efficiency (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)

```

---

KGEIf

*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

```

KGEIf(sim, obs, ...)

## Default S3 method:
KGEIf(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'
KGEIf(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'
KGEIf(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'
KGEIf(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 element 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 <b>OR</b> sim, the $i$ -th value of obs <b>AND</b> 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 $\text{mean}(\text{sim}) - \text{mean}(\text{obs})$ 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 acceptable 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  $-\infty$  to 1 and, therefore,  $KGE_{lf}$  should also range from  $-\infty$  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](#), [KGE<sub>np</sub>](#), [sKGE](#), [gof](#), [ggof](#)

## Examples

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

obs <- 1:10
sim <- 2:11
KGEIf(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")
```

```

# KGEIf (Garcia et al., 2017):
KGEIf(sim=sim, obs=obs, method="2012")

#####
# Example3: KGEIf 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 'KGEIf'
KGEIf(sim=sim, obs=obs)

#####
# Example 4: KGEIf 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.

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

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

#####
# Example 5: KGEIf 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

KGEIf(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)
KGEIf(sim=lsim, obs=lobs)

#####
# Example 6: KGEIf 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
KGEIf(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)

```

---

KGE<sub>np</sub>


---

*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

<code>sim</code>	numeric, zoo, matrix or data.frame with simulated values
<code>obs</code>	numeric, zoo, matrix or data.frame with observed values
<code>na.rm</code>	a logical value indicating whether 'NA' should be stripped before the computation proceeds. When an 'NA' value is found at the <i>i</i> -th position in <code>obs</code> <b>OR</b> <code>sim</code> , the <i>i</i> -th value of <code>obs</code> <b>AND</b> <code>sim</code> are removed before the computation.
<code>out.type</code>	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: -) <code>single</code> : the output is a numeric with the Kling-Gupta efficiency only. -) <code>full</code> : 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 ( <code>rSpearman</code> ), the ratio between the mean of the simulated values to the mean of observations ( <code>Beta</code> ), and the variability measure ( <code>Alpha</code> ).
<code>fun</code>	function to be applied to <code>sim</code> and <code>obs</code> in order to obtain transformed values thereof before computing this goodness-of-fit index. The first argument <b>MUST BE</b> a numeric vector with any name (e.g., <code>x</code> ), and additional arguments are passed using <code>...</code>
<code>...</code>	arguments passed to <code>fun</code> , in addition to the mandatory first numeric vector.
<code>epsilon.type</code>	argument used to define a numeric value to be added to both <code>sim</code> and <code>obs</code> before applying <code>fun</code> . 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 version of the Kling-Gupta efficiency keeps the bias term  $\text{Alpha} (\text{mean}(\text{sim}) / \text{mean}(\text{obs}))$ , 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).

$$KGE_{np} = 1 - ED$$

$$ED = \sqrt{((\rho - 1)^2 + (\alpha - 1)^2 + (\beta - 1)^2)}$$

$$\rho = \text{Spearman rank correlation coefficient}$$

$$\alpha = 1 - 0.5 * \text{sum}(\text{sim}(I(k)) / (n * \mu_s) - \text{obs}(J(k)) / (n * \mu_o))$$

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

Traditional Kling-Gupta efficiencies (Gupta et al., 2009; Kling et al., 2012) range from -Inf to 1, and therefore KGE<sub>np</sub> 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](#), [KGE<sub>lf</sub>](#), [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)

```

## Description

Lee and Choi Efficiency between `sim` and `obs`, with treatment of missing values.

This goodness-of-fit measure was proposed by Lee and Choi (2022) as an alternative to the Liu-Mean Efficiency (LME), designed to provide a multi-dimensional and diagnostically balanced evaluation of model performance; by jointly considering correlation, variability and bias; whereas LME is fundamentally a single-error-based metric.

Unlike some single-error-based criteria, LCE explicitly combines the correlation coefficient and the variability ratio through two complementary terms,  $r \cdot \text{Alpha}$  and  $r / \text{Alpha}$ , in order to penalize inconsistent representations of timing and variability.

For a short description of its components and the numeric range of values, please see [Details](#).

## Usage

```
LCE(sim, obs, ...)
```

```
## Default S3 method:
LCE(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'
LCE(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'
LCE(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'
LCE(sim, obs, na.rm=TRUE, out.type=c("single", "full"),
     fun=NULL, ...,
     epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
     epsilon.value=NA)
```

## Arguments

<code>sim</code>	numeric, zoo, matrix or data.frame with simulated values
<code>obs</code>	numeric, zoo, matrix or data.frame with observed values
<code>na.rm</code>	a logical value indicating whether 'NA' should be stripped before the computation proceeds. When an 'NA' value is found at the <i>i</i> -th position in <code>obs</code> <b>OR</b> <code>sim</code> , the <i>i</i> -th value of <code>obs</code> <b>AND</b> <code>sim</code> are removed before the computation.

out.type	<p>character, indicating whether the output of the function has to include each one of the terms used in the computation of the Lee and Choi Efficiency or not. Valid values are:</p> <ul style="list-style-type: none"> <li>-) single: the output is a numeric with the Lee and Choi Efficiency only.</li> <li>-) full: the output is a list of two elements: the first one with the Lee and Choi Efficiency, and the second is a numeric with 5 elements: the Pearson product-moment correlation coefficient ('r'), the variability ratio ('Alpha'), the bias ratio ('Beta'), the product between correlation and variability ('rAlpha'), and the ratio between correlation and variability ('rOverAlpha').</li> </ul>
fun	<p>function to be applied to sim and obs in order to obtain transformed values thereof before computing the Lee and Choi Efficiency.</p> <p>The first argument MUST BE a numeric vector with any name (e.g., x), and additional arguments are passed using ...</p>
...	arguments passed to fun, in addition to the mandatory first numeric vector.
epsilon.type	<p>argument used to define a numeric value to be added to both sim and obs before applying fun.</p> <p>It was designed to allow the use of logarithm and other similar functions that do not work with zero values.</p> <p>Valid values of epsilon.type are:</p> <ol style="list-style-type: none"> <li>1) "none": sim and obs are used by fun without the addition of any numeric value.</li> <li>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).</li> <li>3) "otherFactor": the numeric value defined in the epsilon.value argument is used to multiply 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.</li> <li>4) "otherValue": the numeric value defined in the epsilon.value argument is directly added to both sim and obs, before applying fun.</li> </ol>
epsilon.value	<ul style="list-style-type: none"> <li>-) when epsilon.type="otherValue" it represents the numeric value to be added to both sim and obs before applying fun.</li> <li>-) 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.</li> </ul>

## Details

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

- 1) r : the Pearson product-moment correlation coefficient. Ideal value is  $r=1$ .
- 2) Alpha : the variability ratio, defined as the ratio between the standard deviation of the simulated values and the standard deviation of the observed ones. Ideal value is  $\text{Alpha}=1$ .
- 3) Beta : the bias ratio, defined as the ratio between the mean of the simulated values and the mean of the observed ones. Ideal value is  $\text{Beta}=1$ .

The Lee and Choi Efficiency combines these terms through two additional correlation-variability components:

$$r\alpha$$

and

$$r/\alpha$$

Both terms have an ideal value of 1. The first term,  $r\alpha$ , penalizes cases where the combined effect of correlation and variability is inconsistent with the observed dynamics. The second term,  $r/\alpha$ , penalizes the opposite imbalance between correlation and variability. Together, these two terms make LCE sensitive to compensating errors between timing and dispersion.

Lee and Choi efficiencies range from -Inf to 1. Essentially, the closer to 1, the more similar sim and obs are in terms of timing, variability and bias.

$$LCE = 1 - ED$$

$$ED = \sqrt{(r\alpha - 1)^2 + (r/\alpha - 1)^2 + (\beta - 1)^2}$$

where:

$$r = \text{Pearson product - moment correlation coefficient}$$

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

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

$$r\alpha = r \times \alpha$$

$$r/\alpha = r / \alpha$$

where  $\mu_s$  and  $\mu_o$  are the mean simulated and observed values, respectively, and  $\sigma_s$  and  $\sigma_o$  are their corresponding standard deviations.

A value of LCE=1 indicates perfect agreement between simulated and observed values. Values close to 1 indicate strong agreement across correlation, variability and bias, whereas negative values indicate progressively poorer model performance.

**Value**

If `out.type=single`: numeric with the Lee and Choi Efficiency between `sim` and `obs`. If `sim` and `obs` are matrices, the output value is a vector, with the Lee and Choi Efficiency between each column of `sim` and `obs`.

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

`LCE.value`            numeric with the Lee and Choi Efficiency. If `sim` and `obs` are matrices, the output value is a vector, with the Lee and Choi Efficiency between each column of `sim` and `obs`

`LCE.elements`        numeric with 5 elements: the Pearson product-moment correlation coefficient ('`r`'), the variability ratio ('`Alpha`'), the bias ratio ('`Beta`'), the product between correlation and variability ('`rAlpha`'), and the ratio between correlation and variability ('`rOverAlpha`'). If `sim` and `obs` are matrices, the output value is a matrix, with the previous five elements computed for 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**

Lee, J. S., & Choi, H. I. (2022). A rebalanced performance criterion for hydrological model calibration. *Journal of Hydrology*, 606, 127372. <https://doi.org/10.1016/j.jhydrol.2021.127372>

Liu, D. (2020). A rational performance criterion for hydrological model. *Journal of Hydrology*, 590, 125488. <https://doi.org/10.1016/j.jhydrol.2020.125488>

Choi, H. I. (2021). Comment on Liu (2020): A rational performance criterion for hydrological model. *Journal of Hydrology*, 606, 126927. <https://doi.org/10.1016/j.jhydrol.2021.126927>

Liu, D. (2021). Reply to "Comment on Liu (2020): A rational performance criterion for a hydrological model" by HyunIl Choi. *Journal of Hydrology*, 603, 126935. <https://doi.org/10.1016/j.jhydrol.2021.126927>

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.

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

**See Also**

[LME](#), [KGE](#), [NSE](#), [gof](#), [ggof](#)

### Examples

```

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

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

#####
# Example2: simulated values equal to observations plus random noise

set.seed(123)
obs <- 1:100
sim <- obs + rnorm(100, mean=0, sd=5)

LCE(sim=sim, obs=obs)

LCE(sim=sim, obs=obs, out.type="full")

#####
# Example3: applying logarithmic transformation

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

#####
# Example4: applying logarithmic transformation with Pushpalatha constant

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

```

---

LME

*Liu-Mean Efficiency*


---

### Description

Liu-Mean Efficiency between `sim` and `obs`, with treatment of missing values.

This goodness-of-fit measure was proposed by Liu et al. (2020) as an alternative to the Nash-Sutcliffe efficiency (NSE), designed to provide a more balanced assessment of model performance by normalising the mean squared error using the mean of the observed values instead of their variance.

The Liu-Mean Efficiency evaluates how large the error is compared to the average level of the observations, making it particularly useful in hydrological applications where the mean value is a meaningful scale for evaluating prediction accuracy.

The normalisation makes that this performance measure behave like a dimensionless relative error, scaled by the characteristic magnitude of the variable. As a result, the same absolute error will be judged differently depending on whether the mean flow is small or large.

For a short description of the metric and the numeric range of values, please see Details.

**Usage**

```

LME(sim, obs, ...)

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

```

**Arguments**

<code>sim</code>	numeric, zoo, matrix or data.frame with simulated values.
<code>obs</code>	numeric, zoo, matrix or data.frame with observed values.
<code>na.rm</code>	a logical value indicating whether 'NA' should be stripped before the computation proceeds. When an 'NA' value is found at the <i>i</i> -th position in <code>obs</code> <b>OR</b> <code>sim</code> , the <i>i</i> -th value of <code>obs</code> <b>AND</b> <code>sim</code> are removed before the computation.
<code>out.type</code>	character, indicating whether the output of the function has to include only the Liu-Mean Efficiency or also the intermediate quantities used in its computation. Valid values are: -) <code>single</code> : the output is a numeric with the Liu-Mean Efficiency only. -) <code>full</code> : the output is a list of two elements: the first one with the Liu-Mean Efficiency, and the second is a numeric with 2 elements: the mean squared error ('MSE') between <code>sim</code> and <code>obs</code> , and the mean of the observed values ('MeanObs') used as the normalization term in the computation of the Liu-Mean Efficiency.
<code>fun</code>	function to be applied to <code>sim</code> and <code>obs</code> in order to obtain transformed values thereof before computing the Liu-Mean Efficiency. The first argument <b>MUST BE</b> a numeric vector with any name (e.g., <code>x</code> ), and additional arguments are passed using <code>...</code>
<code>...</code>	arguments passed to <code>fun</code> , in addition to the mandatory first numeric vector.
<code>epsilon.type</code>	argument used to define a numeric value to be added to both <code>sim</code> and <code>obs</code> before applying <code>fun</code> .

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) "none": `sim` and `obs` are used by `fun` without the addition of any numeric 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 mean observed values. 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. The resulting value is then added to both `sim` and `obs` before applying `fun`.

## Details

The Liu-Mean Efficiency (LME) is based on the mean squared error (MSE) normalized by the squared mean of the observed values.

Its formulation is conceptually similar to the Nash-Sutcliffe efficiency, but uses the mean of observations as the reference scaling factor instead of the variance. This modification reduces the sensitivity of the metric to high variability and makes the performance evaluation more directly interpretable in terms of proportional error relative to the mean magnitude of the observed variable.

The Liu-Mean Efficiency ranges from -Inf to 1.

A value of:

- LME = 1 indicates perfect agreement between `sim` and `obs`. - LME = 0 indicates that the mean squared error equals the squared mean of the observed values. - LME < 0 indicates that the model predictions are worse than the squared mean of the observed values.

Essentially, the closer the LME value is to 1, the more similar `sim` and `obs` are.

$$LME = 1 - \frac{MSE}{\mu_o^2}$$

$$MSE = \frac{1}{n} \sum_{i=1}^n (sim_i - obs_i)^2$$

$$\mu_o = \frac{1}{n} \sum_{i=1}^n obs_i$$

where:

- `n` is the number of paired observations

- `sim_i` is the simulated value at time step `i`
- `obs_i` is the observed value at time step `i`
- `\mu_o` is the mean of the observed values

### Value

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

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

<code>LME.value</code>	numeric with the Liu-Mean Efficiency. If <code>sim</code> and <code>obs</code> are matrices, the output value is a vector, with the Liu-Mean Efficiency between each column of <code>sim</code> and <code>obs</code>
<code>LME.elements</code>	numeric with 2 elements: the mean squared error ('MSE') between <code>sim</code> and <code>obs</code> , and the mean of the observed values ('MeanObs'). If <code>sim</code> and <code>obs</code> are matrices, the output value is a matrix, with the previous two elements computed for each column of <code>sim</code> and <code>obs</code>

### 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

- Liu, D.; Chen, X.; Lian, Y.; Lou, Z. (2020). A new performance measure for hydrologic models. Journal of Hydrology, 590, 125488. doi:10.1016/j.jhydrol.2020.125488.*
- Nash, J.E.; Sutcliffe, J.V. (1970). River flow forecasting through conceptual models part I - A discussion of principles. Journal of Hydrology, 10(3), 282-290.*
- Pushpalatha, R.; Perrin, C.; Le Moine, N.; Andreassian, V. (2012). A review of efficiency criteria suitable for evaluating low-flow simulations. Journal of Hydrology, 420-421, 171-182.*

### See Also

[LCE](#), [me](#), [pbias](#), [NSE](#), [KGE](#), [gof](#), [ggof](#)

**Examples**

```

# Example 0: basic ideal case
obs <- 1:10
sim <- 1:10
LME(sim, obs)

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

#####
# Example 1: Looking at the difference between LME and KGE, both with '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")

# LME (Liu et al., 2020):
LME(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 'LME' for the "best" (unattainable) case
LME(sim=sim, obs=obs)

#####
# Example 3: LME 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)

```

```

LME(sim=sim, obs=obs)

#####
# Example 4: LME 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.

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

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

#####
# Example 5: LME 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

LME(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)
LME(sim=lsim, obs=lobs)

#####
# Example 6: LME 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
LME(sim=sim, obs=obs, fun=log, epsilon.type="otherValue", epsilon.value=eps)

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

#####
# Example 7: LME 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
LME(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)
LME(sim=lsim, obs=lobs)

#####
# Example 8: LME 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)}

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

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

#####
# Example 9: LME 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)

LME(sim=SIM, obs=OBS)

#####
# Example 10: LME 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)
LME(sim=SIM, obs=OBS, out.PerYear=TRUE)

```

---

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

<code>sim</code>	numeric, zoo, matrix or data.frame with simulated values
<code>obs</code>	numeric, zoo, matrix or data.frame with observed values
<code>na.rm</code>	a logical value indicating whether 'NA' should be stripped before the computation proceeds. When an 'NA' value is found at the <i>i</i> -th position in <code>obs</code> <b>OR</b> <code>sim</code> , the <i>i</i> -th value of <code>obs</code> <b>AND</b> <code>sim</code> are removed before the computation.
<code>fun</code>	function to be applied to <code>sim</code> and <code>obs</code> in order to obtain transformed values thereof before computing this goodness-of-fit index. The first argument <b>MUST BE</b> a numeric vector with any name (e.g., <code>x</code> ), and additional arguments are passed using <code>...</code>
<code>...</code>	arguments passed to <code>fun</code> , in addition to the mandatory first numeric vector.
<code>epsilon.type</code>	argument used to define a numeric value to be added to both <code>sim</code> and <code>obs</code> before applying <code>fun</code> . It is was designed to allow the use of logarithm and other similar functions that do not work with zero values. Valid values of <code>epsilon.type</code> are: 1) "none": <code>sim</code> and <code>obs</code> are used by <code>fun</code> 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 <code>sim</code> and <code>obs</code> before applying <code>fun</code> , as described in Pushpalatha et al. (2012). 3) "otherFactor": the numeric value defined in the <code>epsilon.value</code> 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 <code>sim</code> and <code>obs</code> , before applying <code>fun</code> .

- 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](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 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)

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**

<code>sim</code>	numeric, zoo, matrix or data.frame with simulated values
<code>obs</code>	numeric, zoo, matrix or data.frame with observed values
<code>j</code>	numeric, with the exponent to be used in the computation of the modified index of agreement. The default value is <code>j=1</code> .
<code>na.rm</code>	a logical value indicating whether 'NA' should be stripped before the computation proceeds. When an 'NA' value is found at the <i>i</i> -th position in <code>obs</code> <b>OR</b> <code>sim</code> , the <i>i</i> -th value of <code>obs</code> <b>AND</b> <code>sim</code> are removed before the computation.
<code>fun</code>	function to be applied to <code>sim</code> and <code>obs</code> 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 numeric 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

$$md = 1 - \frac{\sum_{i=1}^N |O_i - S_i|^j}{\sum_{i=1}^N |S_i - \bar{O}| + |O_i - \bar{O}|^j}$$

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	<i>Mean Error</i>
----	-------------------

---

### 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

<code>sim</code>	numeric, zoo, matrix or data.frame with simulated values
<code>obs</code>	numeric, zoo, matrix or data.frame with observed values
<code>na.rm</code>	a logical value indicating whether 'NA' should be stripped before the computation proceeds. When an 'NA' value is found at the <i>i</i> -th position in <code>obs</code> <b>OR</b> <code>sim</code> , the <i>i</i> -th value of <code>obs</code> <b>AND</b> <code>sim</code> are removed before the computation.
<code>fun</code>	function to be applied to <code>sim</code> and <code>obs</code> in order to obtain transformed values thereof before computing this goodness-of-fit index. The first argument <b>MUST BE</b> a numeric vector with any name (e.g., <code>x</code> ), and additional arguments are passed using <code>...</code>
<code>...</code>	arguments passed to <code>fun</code> , in addition to the mandatory first numeric vector.

epsilon.type	<p>argument used to define a numeric value to be added to both <code>sim</code> and <code>obs</code> before applying fun.</p> <p>It is was designed to allow the use of logarithm and other similar functions that do not work with zero values.</p> <p>Valid values of <code>epsilon.type</code> are:</p> <ol style="list-style-type: none"> <li>1) "none": <code>sim</code> and <code>obs</code> are used by fun without the addition of any numeric value. This is the default option.</li> <li>2) "Pushpalatha2012": one hundredth (1/100) of the mean observed values is added to both <code>sim</code> and <code>obs</code> before applying fun, as described in Pushpalatha et al. (2012).</li> <li>3) "otherFactor": the numeric value defined in the <code>epsilon.value</code> 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 <code>sim</code> and <code>obs</code>, before applying fun.</li> <li>4) "otherValue": the numeric value defined in the <code>epsilon.value</code> argument is directly added to both <code>sim</code> and <code>obs</code>, before applying fun.</li> </ol>
epsilon.value	<p>-) when <code>epsilon.type="otherValue"</code> it represents the numeric value to be added to both <code>sim</code> and <code>obs</code> before applying fun.</p> <p>-) when <code>epsilon.type="otherFactor"</code> 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 <code>sim</code> and <code>obs</code> before applying fun.</p>

### 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 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)

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 <b>OR</b> sim, the i-th value of obs <b>AND</b> 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 <b>MUST BE</b> 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. 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 <b>OR</b> sim, the i-th value of obs <b>AND</b> 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 <b>MUST BE</b> 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 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)

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=c("sd", "maxmin", "mean", "IQR"),
      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=c("sd", "maxmin", "mean", "IQR"),
      fun=NULL, ...,
      epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
      epsilon.value=NA)

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

## S3 method for class 'zoo'
nrmse(sim, obs, na.rm=TRUE, norm=c("sd", "maxmin", "mean", "IQR"),
      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 <b>OR</b> sim, the i-th value of obs <b>AND</b> 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 -) mean : arithmetic mean of observed values -) IQR : interquantile range of observed values, computed using the stats::IQR function with type=7.
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 <b>MUST BE</b> 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

$$nrmse = 100 \frac{\sqrt{\frac{1}{N} \sum_{i=1}^N (S_i - O_i)^2}}{nval}$$

$$nval = \begin{cases} sd(O_i) & , \text{norm}="sd" \\ O_{max} - O_{min} & , \text{norm}="maxmin" \end{cases}$$

## 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 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)

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

<code>sim</code>	numeric, zoo, matrix or data.frame with simulated values
<code>obs</code>	numeric, zoo, matrix or data.frame with observed values
<code>na.rm</code>	a logical value indicating whether 'NA' should be stripped before the computation proceeds. When an 'NA' value is found at the <i>i</i> -th position in <code>obs</code> <b>OR</b> <code>sim</code> , the <i>i</i> -th value of <code>obs</code> <b>AND</b> <code>sim</code> are removed before the computation.

fun	function to be applied to <code>sim</code> and <code>obs</code> in order to obtain transformed values thereof before computing the Nash-Sutcliffe efficiency. The first argument <b>MUST BE</b> a numeric vector with any name (e.g., <code>x</code> ), and additional arguments are passed using <code>...</code>
...	arguments passed to <code>fun</code> , in addition to the mandatory first numeric vector.
epsilon.type	argument used to define a numeric value to be added to both <code>sim</code> and <code>obs</code> before applying <code>FUN</code> . It is was designed to allow the use of logarithm and other similar functions that do not work with zero values. Valid values of <code>epsilon.type</code> are: 1) "none": <code>sim</code> and <code>obs</code> are used by <code>fun</code> 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 <code>sim</code> and <code>obs</code> before applying <code>fun</code> , as described in Pushpalatha et al. (2012). 3) "otherFactor": the numeric value defined in the <code>epsilon.value</code> 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 <code>sim</code> and <code>obs</code> , before applying <code>fun</code> . 4) "otherValue": the numeric value defined in the <code>epsilon.value</code> argument is directly added to both <code>sim</code> and <code>obs</code> , before applying <code>fun</code> .
epsilon.value	-) when <code>epsilon.type="otherValue"</code> it represents the numeric value to be added to both <code>sim</code> and <code>obs</code> before applying <code>fun</code> . -) when <code>epsilon.type="otherFactor"</code> 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 <code>sim</code> and <code>obs</code> before applying <code>fun</code> .

## Details

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

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  $-\infty$  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,
- )  $-\infty < NSE < 0$ , 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](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 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)

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)

```

**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**

<code>sim</code>	numeric, zoo, matrix or data.frame with simulated values
<code>obs</code>	numeric, zoo, matrix or data.frame with observed values
<code>na.rm</code>	a logical value indicating whether 'NA' should be stripped before the computation proceeds. When an 'NA' value is found at the <i>i</i> -th position in <code>obs</code> <b>OR</b> <code>sim</code> , the <i>i</i> -th value of <code>obs</code> <b>AND</b> <code>sim</code> are removed before the computation.
<code>dec</code>	numeric, specifying the number of decimal places used to round the output object. Default value is 1.
<code>fun</code>	function to be applied to <code>sim</code> and <code>obs</code> in order to obtain transformed values thereof before computing this goodness-of-fit index. The first argument <b>MUST BE</b> a numeric vector with any name (e.g., <code>x</code> ), and additional arguments are passed using <code>...</code>
<code>...</code>	arguments passed to <code>fun</code> , in addition to the mandatory first numeric vector.
<code>epsilon.type</code>	argument used to define a numeric value to be added to both <code>sim</code> and <code>obs</code> before applying <code>fun</code> . It 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 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)
```

```
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	<i>Percent Bias in the Slope of the Midsegment of the Flow Duration Curve</i>
----------	---

---

### 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

<code>sim</code>	numeric, zoo, matrix or data.frame with simulated values
<code>obs</code>	numeric, zoo, matrix or data.frame with observed values
<code>lQ.thr</code>	numeric, used to classify low flows. All the streamflows with a probability of exceedence larger or equal to <code>lQ.thr</code> are classified as low flows
<code>hQ.thr</code>	numeric, used to classify high flows. All the streamflows with a probability of exceedence larger or equal to <code>hQ.thr</code> are classified as high flows
<code>na.rm</code>	a logical value indicating whether 'NA' values should be stripped before the computation proceeds.
<code>plot</code>	a logical value indicating if the flow duration curves corresponding to <code>obs</code> and <code>sim</code> have to be plotted or not.
<code>verbose</code>	logical; if TRUE, progress messages are printed
<code>fun</code>	function to be applied to <code>sim</code> and <code>obs</code> in order to obtain transformed values thereof before computing this goodness-of-fit index. The first argument <b>MUST BE</b> a numeric vector with any name (e.g., <code>x</code> ), and additional arguments are passed using <code>...</code>
<code>...</code>	arguments passed to <code>fun</code> , in addition to the mandatory first numeric vector.
<code>epsilon.type</code>	argument used to define a numeric value to be added to both <code>sim</code> and <code>obs</code> before applying <code>fun</code> . It is was designed to allow the use of logarithm and other similar functions that do not work with zero values. Valid values of <code>epsilon.type</code> are: 1) "none": <code>sim</code> and <code>obs</code> are used by <code>fun</code> 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 <code>sim</code> and <code>obs</code> before applying <code>fun</code> , as described in Pushpalatha et al. (2012). 3) "otherFactor": the numeric value defined in the <code>epsilon.value</code> 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 <code>sim</code> and <code>obs</code> , before applying <code>fun</code> . 4) "otherValue": the numeric value defined in the <code>epsilon.value</code> argument is directly added to both <code>sim</code> and <code>obs</code> , before applying <code>fun</code> .
<code>epsilon.value</code>	-) when <code>epsilon.type="otherValue"</code> it represents the numeric value to be added to both <code>sim</code> and <code>obs</code> before applying <code>fun</code> . -) when <code>epsilon.type="otherFactor"</code> 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 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)

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.

### Details

The *P-factor* quantifies the percentage of observed values that fall within the prediction uncertainty band defined by the lower and upper bounds. It is a measure of the coverage of the uncertainty interval.

Mathematically, the P-factor is defined as:

$$P\text{-factor} = \frac{1}{N} \sum_{i=1}^N I(lband_i \leq x_i \leq uband_i)$$

where  $N$  is the total number of observations,  $x_i$  is the observed value at time step  $i$ , and  $lband_i$  and  $uband_i$  are the lower and upper uncertainty bounds, respectively. The function  $I(\cdot)$  is an indicator function that takes the value 1 when the observed value lies within the uncertainty bounds and 0 otherwise.

The *P-factor* ranges from 0 to 1. A value of 1 indicates that all observations are bracketed by the uncertainty bounds, whereas a value of 0 indicates that none of the observations fall within the bounds.

**Value**

Percent of the  $x$  observations that are within the given uncertainty bounds given by  $lband$  and  $uband$ .

If  $x$ ,  $lband$ , and  $uband$  are matrices, the returned value is a vector with the  $P$ -factor computed for each column.

**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", "NSE", "d",
             "dr", "r", "R2", "KGE", "LCE", "JDKGE", "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. <code>class(x)</code> must be <code>ts</code> or <code>zoo</code> . If <code>leg.gof=TRUE</code> , then x is considered as <b>simulated</b> (for some goodness-of-fit functions this is important)       |
| y | time series that will be plotted. <code>class(x)</code> must be <code>ts</code> or <code>zoo</code> . If <code>leg.gof=TRUE</code> , then y is considered as <b>observed</b> 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 <a href="#">drawTimeAxis</a> .
main	an overall title for the plot: see <a href="#">title</a>
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 is 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 <a href="#">par</a> .
cex.lab	numeric, with the magnification to be used for x and y labels relative to the current setting of 'cex'. See <a href="#">par</a> .
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.: <a href="#">ggof</a> )
...	further arguments passed to <a href="#">plot.zoo</a> 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, <b>the values provided by dates over-write the original dates of the objects.</b>

date.fmt	OPTIONAL. Character indicating the format in which the dates entered are stored in cal.ini and val.ini. See format in <a href="#">as.Date</a> . 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 <a href="#">polygon</a> . Color to be used for filling the area between the lower and upper uncertainty bound.
border	See <a href="#">polygon</a> . 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 <a href="#">drawTimeAxis</a> .
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 <a href="#">plot.default</a> .
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 <a href="#">plot.default</a> . 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 <a href="#">plot.default</a> . The line width, see 'par'.
lty	See <a href="#">plot.default</a> . 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 <a href="#">points</a> 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	<i>Adds uncertainty bounds to an existing plot.</i>
---------------	---

---

### 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, <b>the values provided by dates over-write the original dates of the objects.</b>
date.fmt	OPTIONAL. Character indicating the format of dates. See format in <a href="#">as.Date</a> .
bands.col	See <a href="#">polygon</a> . Color to be used for filling the area between the lower and upper uncertainty bound.
border	See <a href="#">polygon</a> . 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 <a href="#">polygon</a> 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)

```

---

 PMR

*Proxy for Model Robustness*


---

**Description**

Proxy for model robustness (PMR) between `sim` and `obs`, with treatment of missing values.

**Usage**

```

PMR(sim, obs, ...)

## Default S3 method:
PMR(sim, obs, na.rm=TRUE, k=NULL, min.years=5,
     days.per.year=365, fun=NULL, ...,
     epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
     epsilon.value=NA)

## S3 method for class 'data.frame'
PMR(sim, obs, na.rm=TRUE, k=NULL, min.years=5,
     days.per.year=365, fun=NULL, ...,
     epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
     epsilon.value=NA)

## S3 method for class 'matrix'
PMR(sim, obs, na.rm=TRUE, k=NULL, min.years=5,
     days.per.year=365, fun=NULL, ...,
     epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
     epsilon.value=NA)

```

```
## S3 method for class 'zoo'
PMR(sim, obs, na.rm=TRUE, k=NULL, min.years=5,
     days.per.year=365, fun=NULL, ...,
     epsilon.type=c("none", "Pushpalatha2012", "otherFactor", "otherValue"),
     epsilon.value=NA)
```

### Arguments

sim	zoo object with simulated values. Multicolumn zoo objects are allowed.
obs	zoo object with observed values. Multicolumn zoo objects are allowed.
na.rm	a logical value indicating whether 'NA' should be stripped before the computation proceeds. For the full-period bias, only positions with valid paired values in obs and sim are used. For the moving-window biases, each fixed-length window is selected first, and invalid pairs are then removed inside that window.
k	integer value representing the length of the moving window (number of time steps) used to compute the bias over sub-periods.  By default k=NULL, which means that its value is automatically computed based on the minimum numbers of years defined by min.years. The k argument should reflect the temporal scale at which robustness is intended to be evaluated, and therefore depends primarily on the time resolution of the data. Royer-Gaspard et al. (2021) recommended to use multi-year windows, typically in the range of 3 to 5 years, to ensure that each sub-period captures meaningful hydroclimatic variability while still allowing enough windows for comparison.
min.years	Numeric, only used when the user does not explicitly define the value of k, i.e., when k=NULL.  Minimum numbers of years used to ensure that each sub-period used in the computation of PMR captures meaningful hydroclimatic variability while still allowing enough windows for comparison. By default, min.years=5.
days.per.year	Numeric, only used when the user does not explicitly define the value of k, i.e., when k=NULL.  Number of days in a year. A value of 365.25 is recommended instead of the default value of 365 when sim and obs are long climatological series.
fun	function to be applied to sim and obs in order to obtain transformed values thereof before computing the proxy for model robustness. 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 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. 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 to obtain the constant to be added to both `sim` and `obs` before applying `fun`.

## Details

$$PMR = 2 \times \frac{1}{N} \sum_{i=1}^N |(\bar{S}_i - \bar{O}_i) - (\bar{S} - \bar{O})| \frac{1}{\bar{O}}$$

where:

- $\bar{S}_i$  and  $\bar{O}_i$  are the mean simulated and observed values computed over the  $i$ -th moving window of length  $k$ ,
- $\bar{S}$  and  $\bar{O}$  are the overall mean simulated and observed values,
- $N$  is the number of moving windows.

The proxy for model robustness (PMR) is a dimensionless statistic that quantifies the temporal stability of model bias by measuring the average deviation of sub-period bias from the overall bias.

PMR indicates how consistent the model performance is across different time periods or hydrological conditions.

The proxy for model robustness ranges from 0 to positive infinity. Essentially, the closer to 0, the more temporally robust the model is.

-)  $PMR = 0$  corresponds to a perfectly robust model, in which the model bias is identical across all sub-periods.

-)  $0 < PMR < 1$  indicates relatively stable model performance with moderate temporal variability in bias.

-) PMR > 1 indicates increasing variability in model bias across time periods, suggesting reduced robustness of model performance.

### Value

Proxy for model robustness between sim and obs.

If sim and obs are multicolumn zoo objects, the returned value is a vector with the proxy for model robustness between each column of sim and obs.

### Note

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

For the moving-window biases, the fixed-length window is selected before invalid pairs are removed inside that window.

The choice of window length k influences the temporal scale at which robustness is evaluated.

### Author(s)

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

### References

Royer-Gaspard, P., Andreassian, V., and Thirel, G. (2021). Technical note: PMR - a proxy metric to assess hydrological model robustness in a changing climate. *Hydrology and Earth System Sciences*, 25, 5703–5716. doi:10.5194/hess-25-5703-2021.

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.

### See Also

[pbias](#), [NSE](#), [VE](#), [JDKGE](#), [gof](#), [ggof](#)

### Examples

```
#####
# Example 1: Looking at the difference between PMR and KGE, both with '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")

# PMR (Royer-Gaspard et al., 2021):
PMR(sim=sim, obs=obs)

## Not run:
#####
# 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 'PMR' for the "best" (unattainable) case
PMR(sim=sim, obs=obs)

#####
# Example 3: PMR 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)

PMR(sim=sim, obs=obs)

#####
# Example 4: PMR 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.

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

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

#####
# Example 5: PMR 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

PMR(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)
PMR(sim=lsim, obs=lobs)

#####
# Example 6: PMR 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
PMR(sim=sim, obs=obs, fun=log, epsilon.type="otherValue", epsilon.value=eps)

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

#####
# Example 7: PMR 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
PMR(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)
PMR(sim=lsim, obs=lobs)

#####
# Example 8: PMR 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)}

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

# Verifying the previous value
sim1 <- sqrt(sim+1)
obs1 <- sqrt(obs+1)
PMR(sim=sim1, obs=obs1)
#####
# Example 9: PMR 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)

PMR(sim=SIM, obs=OBS)

## 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**

<code>sim</code>	numeric, zoo, matrix or data.frame with simulated values
<code>obs</code>	numeric, zoo, matrix or data.frame with observed values
<code>na.rm</code>	a logical value indicating whether 'NA' should be stripped before the computation proceeds. When an 'NA' value is found at the <i>i</i> -th position in <code>obs</code> <b>OR</b> <code>sim</code> , the <i>i</i> -th value of <code>obs</code> <b>AND</b> <code>sim</code> are removed before the computation.

fun	function to be applied to <code>sim</code> and <code>obs</code> in order to obtain transformed values thereof before computing this goodness-of-fit index. The first argument <b>MUST BE</b> a numeric vector with any name (e.g., <code>x</code> ), and additional arguments are passed using <code>...</code>
...	arguments passed to <code>fun</code> , in addition to the mandatory first numeric vector.
epsilon.type	argument used to define a numeric value to be added to both <code>sim</code> and <code>obs</code> before applying <code>fun</code> . It is was designed to allow the use of logarithm and other similar functions that do not work with zero values. Valid values of <code>epsilon.type</code> are: 1) "none": <code>sim</code> and <code>obs</code> are used by <code>fun</code> 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 <code>sim</code> and <code>obs</code> before applying <code>fun</code> , as described in Pushpalatha et al. (2012). 3) "otherFactor": the numeric value defined in the <code>epsilon.value</code> 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 <code>sim</code> and <code>obs</code> , before applying <code>fun</code> . 4) "otherValue": the numeric value defined in the <code>epsilon.value</code> argument is directly added to both <code>sim</code> and <code>obs</code> , before applying <code>fun</code> .
epsilon.value	numeric value to be added to both <code>sim</code> and <code>obs</code> when <code>epsilon.type="otherValue"</code> .

## 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](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. Available 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 <b>OR</b> sim, the i-th value of obs <b>AND</b> 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 <b>MUST BE</b> 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. 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

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

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 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)

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.

### Details

The *R-factor* quantifies the average width of the prediction uncertainty band relative to the variability of the observed data. It is a measure of the magnitude of predictive uncertainty associated with a model simulation.

Mathematically, the R-factor is defined as:

$$R\text{-factor} = \frac{\overline{d_x}}{\sigma_x}$$

where  $\sigma_x$  is the standard deviation of the observed variable  $x$ , and  $\overline{d_x}$  is the average thickness of the uncertainty band, computed as:

$$\overline{d_x} = \frac{1}{N} \sum_{i=1}^N (uband_i - lband_i)$$

where  $N$  is the total number of observations, and  $lband_i$  and  $uband_i$  are the lower and upper uncertainty bounds, respectively, at time step  $i$ .

The *R-factor* ranges from 0 to infinity, with an optimal value of 0 indicating perfect agreement between simulated and observed values (i.e., zero prediction uncertainty). In practical applications, the *R-factor* represents the width of the uncertainty interval and should be as small as possible. Values close to or smaller than 1 are commonly considered indicative of an acceptable level of predictive uncertainty, although acceptable thresholds depend on the quality of observations and the modelling context.

Because a larger fraction of observations can often be bracketed by widening the uncertainty bounds, the *R-factor* is typically interpreted jointly with the *P-factor*. A balance between a high *P-factor* (good coverage) and a low *R-factor* (narrow uncertainty bounds) is therefore sought during model calibration and uncertainty analysis.

### 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

<code>sim</code>	numeric, zoo, matrix or data.frame with simulated values
<code>obs</code>	numeric, zoo, matrix or data.frame with observed values
<code>na.rm</code>	a logical value indicating whether 'NA' should be stripped before the computation proceeds. When an 'NA' value is found at the <i>i</i> -th position in <code>obs</code> <b>OR</b> <code>sim</code> , the <i>i</i> -th value of <code>obs</code> <b>AND</b> <code>sim</code> are removed before the computation.
<code>fun</code>	function to be applied to <code>sim</code> and <code>obs</code> in order to obtain transformed values thereof before computing the Root Mean Square Error. The first argument <b>MUST BE</b> a numeric vector with any name (e.g., <code>x</code> ), and additional arguments are passed using <code>...</code>

... 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](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 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)
```

```

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 <b>OR</b> sim, the i-th value of obs <b>AND</b> 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 <b>MUST BE</b> 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. 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 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)

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 <b>OR</b> sim, the i-th value of obs <b>AND</b> 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 <b>MUST BE</b> 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](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 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)

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	<i>Ratio of Standard Deviations</i>
-----	-------------------------------------

---

**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**

<code>sim</code>	numeric, zoo, matrix or data.frame with simulated values
<code>obs</code>	numeric, zoo, matrix or data.frame with observed values
<code>na.rm</code>	a logical value indicating whether 'NA' should be stripped before the computation proceeds. When an 'NA' value is found at the <i>i</i> -th position in <code>obs</code> <b>OR</b> <code>sim</code> , the <i>i</i> -th value of <code>obs</code> <b>AND</b> <code>sim</code> are removed before the computation.
<code>fun</code>	function to be applied to <code>sim</code> and <code>obs</code> in order to obtain transformed values thereof before computing this goodness-of-fit index. The first argument <b>MUST BE</b> a numeric vector with any name (e.g., <code>x</code> ), and additional arguments are passed using <code>...</code>
<code>...</code>	arguments passed to <code>fun</code> , in addition to the mandatory first numeric vector.

epsilon.type	<p>argument used to define a numeric value to be added to both <code>sim</code> and <code>obs</code> before applying <code>fun</code>.</p> <p>It is was designed to allow the use of logarithm and other similar functions that do not work with zero values.</p> <p>Valid values of <code>epsilon.type</code> are:</p> <ol style="list-style-type: none"> <li>1) "none": <code>sim</code> and <code>obs</code> are used by <code>fun</code> without the addition of any numeric value. This is the default option.</li> <li>2) "Pushpalatha2012": one hundredth (1/100) of the mean observed values is added to both <code>sim</code> and <code>obs</code> before applying <code>fun</code>, as described in Pushpalatha et al. (2012).</li> <li>3) "otherFactor": the numeric value defined in the <code>epsilon.value</code> 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 <code>sim</code> and <code>obs</code>, before applying <code>fun</code>.</li> <li>4) "otherValue": the numeric value defined in the <code>epsilon.value</code> argument is directly added to both <code>sim</code> and <code>obs</code>, before applying <code>fun</code>.</li> </ol>
epsilon.value	<p>-) when <code>epsilon.type="otherValue"</code> it represents the numeric value to be added to both <code>sim</code> and <code>obs</code> before applying <code>fun</code>.</p> <p>-) when <code>epsilon.type="otherFactor"</code> 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 <code>sim</code> and <code>obs</code> before applying <code>fun</code>.</p>

### 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 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)

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	<i>Spearman's rank correlation coefficient</i>
-----------	--

---

## 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

<code>sim</code>	numeric, zoo, matrix or data.frame with simulated values
<code>obs</code>	numeric, zoo, matrix or data.frame with observed values
<code>na.rm</code>	a logical value indicating whether 'NA' should be stripped before the computation proceeds. When an 'NA' value is found at the <i>i</i> -th position in <code>obs</code> <b>OR</b> <code>sim</code> , the <i>i</i> -th value of <code>obs</code> <b>AND</b> <code>sim</code> are removed before the computation.
<code>fun</code>	function to be applied to <code>sim</code> and <code>obs</code> in order to obtain transformed values thereof before computing this goodness-of-fit index. The first argument <b>MUST BE</b> a numeric vector with any name (e.g., <code>x</code> ), and additional arguments are passed using <code>....</code>

... 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](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 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)

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 <b>OR</b> sim, the i-th value of obs <b>AND</b> 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 <b>MUST BE</b> 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 <b>OR</b> sim, the i-th value of obs <b>AND</b> 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 <code>sim</code> and <code>obs</code> in order to obtain transformed values thereof before computing this goodness-of-fit index. The first argument <b>MUST BE</b> a numeric vector with any name (e.g., <code>x</code> ), and additional arguments are passed using <code>...</code>
...	arguments passed to <code>fun</code> , in addition to the mandatory first numeric vector.
epsilon.type	argument used to define a numeric value to be added to both <code>sim</code> and <code>obs</code> before applying <code>fun</code> . It is was designed to allow the use of logarithm and other similar functions that do not work with zero values. Valid values of <code>epsilon.type</code> are: 1) "none": <code>sim</code> and <code>obs</code> are used by <code>fun</code> 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 <code>sim</code> and <code>obs</code> before applying <code>fun</code> , as described in Pushpalatha et al. (2012). 3) "otherFactor": the numeric value defined in the <code>epsilon.value</code> 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 <code>sim</code> and <code>obs</code> , before applying <code>fun</code> . 4) "otherValue": the numeric value defined in the <code>epsilon.value</code> argument is directly added to both <code>sim</code> and <code>obs</code> , before applying <code>fun</code> .
epsilon.value	-) when <code>epsilon.type="otherValue"</code> it represents the numeric value to be added to both <code>sim</code> and <code>obs</code> before applying <code>fun</code> . -) when <code>epsilon.type="otherFactor"</code> 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 <code>sim</code> and <code>obs</code> before applying <code>fun</code> .

## Details

The Split Kling-Gupta Efficiency (sKGE) is computed by dividing the time series into individual years (or hydrological years when `start.month != 1`) and calculating the Kling-Gupta Efficiency (KGE) for each year separately. The final sKGE value is defined as the arithmetic mean of the annual KGE values:

$$sKGE = \frac{1}{N} \sum_{y=1}^N KGE_y$$

where  $N$  is the number of years in the evaluation period and  $KGE_y$  is the Kling-Gupta Efficiency computed using the simulated and observed values belonging to year  $y$ .

The Kling-Gupta Efficiency is defined as:

$$KGE = 1 - \sqrt{(r - 1)^2 + (V - 1)^2 + (B - 1)^2}$$

where  $r$  is the linear correlation coefficient between simulated and observed values,  $V$  represents the variability component, and  $B$  represents the bias component.

The definitions of the variability and bias components depend on the selected method argument:

- **method = "2009"** (Gupta et al., 2009)

$$\alpha = \frac{\sigma_{sim}}{\sigma_{obs}}$$

$$\beta = \frac{\mu_{sim}}{\mu_{obs}}$$

where the variability component is  $V = \alpha$  and the bias component is  $B = \beta$ .

- **method = "2012"** (Kling et al., 2012)

$$\gamma = \frac{CV_{sim}}{CV_{obs}}$$

$$\beta = \frac{\mu_{sim}}{\mu_{obs}}$$

where  $CV = \sigma/\mu$ , the variability component is  $V = \gamma$ , and the bias component is  $B = \beta$ .

- **method = "2021"** (Tang et al., 2021)

$$\alpha = \frac{\sigma_{sim}}{\sigma_{obs}}$$

$$\beta = \frac{\mu_{sim} - \mu_{obs}}{\sigma_{obs}}$$

where the variability component is  $V = \alpha$  and the bias component is a standardized bias term.

The splitting procedure allows model performance to be evaluated at the annual scale, providing a measure that summarizes inter-annual performance while reducing the influence of temporal aggregation over long periods.

When `out.PerYear=TRUE`, the individual annual KGE values are also returned.

The optimal value of sKGE is 1. Similar to the traditional Kling-Gupta efficiency, sKGE values can range from  $-\infty$  to 1, where values closer to 1 indicate better agreement between simulated and observed values.

## 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:

<code>sKGE.value</code>	numeric with the Split Kling-Gupta efficiency. If <code>sim</code> and <code>obs</code> are matrices, the output value is a vector, with the Split Kling-Gupta efficiency between each column of <code>sim</code> and <code>obs</code>
<code>KGE.PerYear</code>	numeric with the Kling-Gupta efficiencies obtained for the individual years in <code>sim</code> and <code>obs</code> .

**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.2013.12.001.

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](#), [KGE1f](#), [KGE1p](#), [gof](#), [ggof](#)

**Examples**

```
#####
# Example 1: Looking at the difference between sKGE and KGE, both with '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")

# sKGE (Fowler et al., 2018):
sKGE(sim=sim, obs=obs, method="2009")

# 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 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)

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
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 <b>OR</b> sim, the i-th value of obs <b>AND</b> 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 <b>MUST BE</b> 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 volumetric soil moisture, by removing from the traditional RMSE the mean bias error caused by the mismatch 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

<code>sim</code>	numeric, zoo, matrix or data.frame with simulated values
<code>obs</code>	numeric, zoo, matrix or data.frame with observed values
<code>na.rm</code>	a logical value indicating whether 'NA' should be stripped before the computation proceeds.

	When an 'NA' value is found at the <i>i</i> -th position in <code>obs</code> <b>OR</b> <code>sim</code> , the <i>i</i> -th value of <code>obs</code> <b>AND</b> <code>sim</code> are removed before the computation.
<code>fun</code>	function to be applied to <code>sim</code> and <code>obs</code> in order to obtain transformed values thereof before computing the Root Mean Square Error. The first argument <b>MUST BE</b> a numeric vector with any name (e.g., <code>x</code> ), and additional arguments are passed using <code>...</code>
<code>...</code>	arguments passed to <code>fun</code> , in addition to the mandatory first numeric vector.
<code>epsilon.type</code>	argument used to define a numeric value to be added to both <code>sim</code> and <code>obs</code> before applying <code>FUN</code> . It is was designed to allow the use of logarithm and other similar functions that do not work with zero values. Valid values of <code>epsilon.type</code> are: 1) "none": <code>sim</code> and <code>obs</code> are used by <code>fun</code> 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 <code>sim</code> and <code>obs</code> before applying <code>fun</code> , as described in Pushpalatha et al. (2012). 3) "otherFactor": the numeric value defined in the <code>epsilon.value</code> 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 <code>sim</code> and <code>obs</code> , before applying <code>fun</code> . 4) "otherValue": the numeric value defined in the <code>epsilon.value</code> argument is directly added to both <code>sim</code> and <code>obs</code> , before applying <code>fun</code> .
<code>epsilon.value</code>	numeric value to be added to both <code>sim</code> and <code>obs</code> when <code>epsilon.type="otherValue"</code> .

### 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 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)
```

```

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

<code>sim</code>	zoo, xts, numeric, matrix or data.frame with simulated values
<code>obs</code>	zoo, xts, numeric, matrix or data.frame with observed values
<code>...</code>	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 <b>OR</b> sim, the i-th value of obs <b>AND</b> 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 <b>MUST BE</b> 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 <b>OR</b> sim, the i-th value of obs <b>AND</b> 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 <b>MUST BE</b> 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 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)

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
sim1 <- sqrt(sim+1)
obs1 <- sqrt(obs+1)
wsNSE(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**

<code>sim</code>	numeric, zoo, matrix or data.frame with simulated values
<code>obs</code>	numeric, zoo, matrix or data.frame with observed values

na.rm	<p>a logical value indicating whether 'NA' should be stripped before the computation proceeds.</p> <p>When an 'NA' value is found at the <math>i</math>-th position in obs <b>OR</b> sim, the <math>i</math>-th value of obs <b>AND</b> sim are removed before the computation.</p>
j	<p>numeric, representing an arbitrary value used to power the differences between observations and simulations. By default <math>j=2</math>, which mimics the traditional Nash-Sutcliffe function, mainly focused on thr representation of high values. For low flows, suggested values for <math>j</math> are 1, 1/2 or 1/3. See Legates and McCabe, (1999) and Krausse et al. (2005) for a discussion of suggested values of <math>j</math>.</p>
lambda	<p>numeric in <math>[0, 1]</math> 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.</p> <p>Low values get a weight equal to <math>1-\text{lambda}</math>. Between high and low values there is a linear transition from lambda to <math>1-\text{lambda}</math>, respectively.</p> <p>Suggested values for lambda are lambda=0.95 when focusing in high (streamflow) values and lambda=0.05 when focusing in low (streamflow) values.</p>
lQ.thr	<p>numeric, representing the non-exceedence probability used to identify low flows in obs. All values in obs that are equal or lower than <math>\text{quantile}(\text{obs}, \text{probs}=(1-\text{lQ.thr}))</math> are considered as low values. By default <math>\text{lQ.thr}=0.6</math>.</p> <p>On the other hand, the low values in sim are those located at the same <math>i</math>-th position than the <math>i</math>-th value of the obs deemed as low flows.</p>
hQ.thr	<p>numeric, representing the non-exceedence probability used to identify high flows in obs. All values in obs that are equal or higher than <math>\text{quantile}(\text{obs}, \text{probs}=(1-\text{hQ.thr}))</math> are considered as high flows. By default <math>\text{hQ.thr}=0.1</math>.</p> <p>On the other hand, the high values in sim are those located at the same <math>i</math>-th position than the <math>i</math>-th value of the obs deemed as high flows.</p>
fun	<p>function to be applied to sim and obs in order to obtain transformed values thereof before computing this goodness-of-fit index.</p> <p>The first argument <b>MUST BE</b> a numeric vector with any name (e.g., x), and additional arguments are passed using ...</p>
...	arguments passed to fun, in addition to the mandatory first numeric vector.
epsilon.type	<p>argument used to define a numeric value to be added to both sim and obs before applying fun.</p> <p>It is was designed to allow the use of logarithm and other similar functions that do not work with zero values.</p> <p>Valid values of epsilon.type are:</p> <ol style="list-style-type: none"> <li>1) "none": sim and obs are used by fun without the addition of any numeric value. This is the default option.</li> <li>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).</li> <li>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</li> </ol>

(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 classical Nash-Sutcliffe efficiency (NSE, Nash and Sutcliffe, 1970), but designed to give more emphasis to either high or low observed values.

In the implemented formulation, the low- and high-flow thresholds are obtained from the observed series as:

$$lQ = Q_{obs}(1 - lQ.thr)$$

$$hQ = Q_{obs}(1 - hQ.thr)$$

where  $Q_{obs}(p)$  is the empirical quantile of `obs` at probability  $p$ . A weight  $w_i$  is then assigned to each observed value  $obs_i$  according to the following piecewise-linear function:

$$w_i = \begin{cases} \lambda, & obs_i \geq hQ \\ 1 - \lambda, & obs_i \leq lQ \\ (1 - \lambda) + (2\lambda - 1) \frac{obs_i - lQ}{hQ - lQ}, & lQ < obs_i < hQ \end{cases}$$

Hence, `lambda` controls the emphasis of the metric:

- when `lambda > 0.5`, high observed values receive larger weights than low values;
- when `lambda < 0.5`, low observed values receive larger weights than high values;
- when `lambda = 0.5`, all values receive the same weight and the weighting becomes uniform.

Using these weights, `wsNSE` is computed as:

$$wsNSE = 1 - \frac{\sum_{i=1}^n |w_i(obs_i - sim_i)|^j}{\sum_{i=1}^n |w_i(obs_i - \overline{obs})|^j}$$

where  $\overline{obs}$  is the arithmetic mean of the observed series after removing missing pairs, and  $j$  is the user-defined exponent. Therefore, the numerator is a weighted error term and the denominator is the corresponding weighted dispersion of `obs` around its mean. This is the exact mathematical formulation implemented in `wsNSE.R`.

Following the traditional NSE, `wsNSE` ranges from  $-\infty$  to 1, with an optimal value of 1. Larger values indicate smaller weighted discrepancies between `sim` and `obs`.

**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](#), [KGE1f](#), [gof](#), [ggof](#)

**Examples**

```
#####
# Example 1: Looking at the difference between 'KGE', 'NSE', 'wnse', 'wsNSE',
# 'APFB' and 'KGE1f' 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 efficiency (Gupta and Kling, 2009)
KGE(sim=sim, obs=obs)

# Traditional Nash-Sutcliffe efficiency (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 efficiency (Gupta and Kling, 2009)
KGE(sim=sim, obs=obs)

# Traditional Nash-Sutcliffe efficiency (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

- \* **datasets**
  - EgaEnEstellaQts, 32
- \* **dplot**
  - ggof, 33
  - plot2, 139
  - plotbands, 142
  - plotbandsonly, 145
- \* **math**
  - APFB, 8
  - br2, 13
  - cp, 18
  - d, 23
  - dr, 27
  - ggof, 33
  - gof, 41
  - HFB, 52
  - KGE, 62
  - KGEkm, 69
  - KGElf, 77
  - KGEnp, 82
  - LCE, 87
  - LME, 92
  - mae, 98
  - md, 103
  - me, 106
  - mNSE, 110
  - mse, 114
  - nrmse, 118
  - NSE, 123
  - pbias, 127
  - pbiasfdc, 132
  - pfactor, 136
  - PMR, 146
  - R2, 152
  - rd, 156
  - rfactor, 161
  - rmse, 164
  - rNSE, 168
  - rPearson, 172
  - rSD, 177
  - rSpearman, 181
  - rsr, 185
  - sKGE, 189
  - ssq, 196
  - ubRMSE, 199
  - valindex, 203
  - ve, 204
  - wNSE, 208
  - wsNSE, 213
- \* **package**
  - hydroGOF-package, 3
- APFB, 4, 8, 49, 55, 216
- as.Date, 33, 143, 145
- br2, 4, 13, 40, 49
- cor, 16, 154, 174, 182, 183
- cp, 4, 18, 40, 49
- d, 4, 23, 30, 40, 49, 105, 159
- dr, 4, 25, 27, 40, 49, 105, 159
- drawTimeAxis, 35, 140, 143
- EgaEnEstellaQts, 32
- fdc, 134
- ggof, 11, 16, 25, 30, 33, 40, 49, 55, 60, 66, 73, 80, 85, 91, 95, 100, 105, 108, 112, 116, 121, 125, 130, 134, 141, 149, 159, 166, 170, 179, 187, 193, 198, 201, 206, 211, 216
- gof, 11, 16, 20, 25, 30, 40, 41, 55, 60, 66, 73, 80, 85, 91, 95, 100, 105, 108, 112, 116, 121, 125, 130, 134, 149, 159, 166, 170, 179, 187, 193, 198, 201, 206, 211, 216
- HFB, 4, 11, 49, 52, 60

- hydroGOF (hydroGOF-package), 3
- hydroGOF-package, 3
- is.na, 204
- JDKGE, 4, 11, 42, 55, 57, 60, 149
- KGE, 4, 40, 42, 43, 49, 62, 73, 80, 85, 91, 95, 112, 125, 170, 193, 211
- KGEkm, 4, 42, 43, 49, 69
- KGElf, 4, 40, 42, 43, 49, 60, 66, 73, 77, 85, 193, 216
- KGENp, 4, 40, 49, 66, 73, 80, 82, 193
- LCE, 4, 87, 95
- lm, 16
- LME, 4, 91, 92
- mae, 3, 40, 49, 98, 108, 116, 121, 130, 134, 166, 198, 201
- md, 4, 25, 30, 40, 49, 103, 159
- me, 3, 40, 49, 95, 106
- mNSE, 3, 40, 42, 49, 110, 125, 170, 211
- mNSEff (mNSE), 110
- mse, 3, 40, 49, 100, 114, 121, 130, 134, 166, 198, 201
- nrmse, 3, 40, 42, 49, 100, 116, 118, 130, 134, 166, 198, 201
- NSE, 3, 11, 40, 49, 55, 60, 65, 91, 95, 112, 123, 149, 170, 206, 211, 216
- NSEff (NSE), 123
- par, 35, 141
- pbias, 3, 40, 49, 95, 100, 116, 121, 127, 130, 134, 149, 166, 198, 201
- pbiasfdc, 4, 40, 42, 49, 100, 116, 121, 130, 132, 166, 198, 201
- pfactor, 4, 136, 144, 145, 163
- plot.default, 143, 144
- plot.zoo, 141
- plot2, 40, 139
- plot\_pq, 141
- plotbands, 138, 142, 163
- plotbandsonly, 145
- PMR, 4, 11, 42, 55, 146
- PMRrob (PMR), 146
- points, 144
- polygon, 143, 145
- R2, 4, 16, 40, 49, 152
- rd, 4, 25, 30, 40, 49, 105, 156
- rfactor, 4, 138, 144, 145, 161
- rmse, 3, 40, 49, 100, 116, 121, 130, 134, 164, 198, 201
- rNSE, 3, 40, 49, 60, 112, 125, 168, 211
- rNSEff (rNSE), 168
- rPearson, 4, 16, 40, 49, 172
- rSD, 3, 40, 49, 177, 187
- rSpearman, 4, 16, 40, 49, 181
- rsr, 3, 40, 49, 179, 185
- sd, 179, 187
- sKGE, 4, 40, 42, 43, 49, 66, 73, 80, 85, 189
- ssq, 4, 100, 116, 121, 130, 134, 166, 196, 201
- title, 140
- ubRMSE, 3, 40, 49, 100, 116, 121, 130, 134, 166, 198, 199
- valindex, 203
- VE, 4, 40, 49, 149
- VE (ve), 204
- ve, 204
- which, 204
- wNSE, 3, 11, 40, 49, 55, 60, 112, 125, 170, 208, 216
- wsNSE, 3, 11, 42, 43, 49, 55, 60, 213, 216