Package 'EMAR'

May 7, 2026

Type Package

Title Empirical Model Assessment

Version 1.0.0

Date 2023-11-10

Description A tool that allows users to generate various indices for evaluating statistical models. The fitstat() function computes indices based on the fitting data. The valstat() function computes indices based on the validation data set. Both fitstat() and valstat() will return 16 indices SSR: residual sum of squares, TRE: total relative error, Bias: mean bias, MRB: mean relative bias, MAB: mean absolute bias, MAPE: mean absolute percentage error, MSE: mean squared error, RMSE: root mean square error, Percent.RMSE: percentage root mean squared error, R2: coefficient of determination, R2adj: adjusted coefficient of determination, APC: Amemiya's prediction criterion, logL: Log-likelihood, AIC: Akaike information criterion, AICc: corrected Akaike information criterion, BIC: Bayesian information criterion, HQC: Hannan-Quin information criterion. The lower the better for the SSR, TRE, Bias, MRB, MAB, MAPE, MSE, RMSE, Percent.RMSE, APC, AIC, AICc, BIC and HQC indices. The higher the better for R2 and R2adj indices. Petre Stoica, P., Selén, Y. (2004) <doi:10.1109/MSP.2004.1311138>\n Zhou et al. (2023) <doi:10.3389/fpls.2023.1186250>\n Ogana canli, I. (2021) <doi:10.1007/s11676-021-01373-1>\n Musabbikhah et al. (2019) <doi:10.1088/1742-6596/1175/1/012270>.

Encoding UTF-8

RoxygenNote 7.2.3

License GPL (>= 3)

NeedsCompilation no

Author Friday Nwabueze Ogana [aut, cre] (ORCID: <https://orcid.org/0000-0002-8388-204X>), Sacramento Corral-Rivas [aut] (ORCID: <https://orcid.org/0000-0001-7624-0623>)

Maintainer Friday Nwabueze Ogana <ogana_fry@yahoo.com>

Repository CRAN

Date/Publication 2023-11-10 19:33:22 UTC

Contents

fitstat 2
valstat 3

Index **5**

`fitstat` *Fit statistics for model assessment*

Description

This function helps users to generate 16 fit indices for model assessment. Once a model(s) is fitted, users can apply the function `fitstat()` to get the indices.

Usage

```
fitstat(obj)
```

Arguments

`obj`	fitted model of the class lm, nls, gls, gnls, lme, or nlme.

Value

Model.name: fitted model name, p: number of parameters in the fitted model, n: number of observation, SSR: residual sum of squares, TRE: total relative error, Bias: mean bias, MRB: mean relative bias, MAB: mean absolute bias, MAPE: mean absolute percentage error, MSE: mean squared error, RMSE: root mean square error, Percent.RMSE: percentage root mean squared error, R2: coefficient of determination, R2adj: adjusted coefficient of determination, APC: Amemiya's prediction criterion, logL: Log-likelihood, AIC: Akaike information criterion, AICc: corrected Akaike information criterion, BIC: Bayesian information criterion, HQC: Hannan-Quin information criterion.

Note

The lower the better for the SSR, TRE, Bias, MRB, MAB, MAPE, MSE, RMSE, Percent.RMSE, APC, AIC, AICc, BIC and HQC indices. The higher the better for R2 and R2adj indices. Users can choose which indices to use to evaluate their models from the output.

Author(s)

Ogana F.N. and Corral-Rivas S.

See Also

`valstat()`, which gives the fit indices of the model based on the independent/validation data

Examples

```
library(EMAR)

# sample data
Age <- 1:50
Yield <- exp(6.5 - 39.5/Age)
fit_data <- data.frame(Age, Yield)

# fit your model(s)
Eq01 <- lm(Yield ~ Age, data=fit_data)
Eq02 <- nls(Yield ~ b0 * Age ^ b1, data=fit_data, start=list(b0 = 2, b1 = 1))

# Get the fit statistics for the model(s)
fitstat(Eq01)
fitstat(Eq02)

# with the 'rbind' function, Users can generate output for multiple models at once.
indices <- rbind(fitstat(Eq01), fitstat(Eq02))
print(indices)
```

`valstat` *Validation statistics for model assessment*

Description

This function helps users to generate 16 fit indices for model assessment based on independent/validation data set. In addition to empirical models, the function `valstat()` can generate fit indices for AI-based models such as artificial neural network, supervise vector machine, etc.

Usage

```
valstat(obs.y, pred.y, p)
```

Arguments

`obs.y`	observed values from the independent/validation data
`pred.y`	predicted values from the model
`p`	number of parameters in the model. This is needed to compute the 'criteria-based indices' and adjusted coefficient of determination. Users could enter any value for AI-based models with an unknown number of parameters (p) and assess their models using the indices that are invariant of p. See the section on note.

Value

n: number of observation in the validation data, SSR: residual sum of squares, TRE: total relative error, Bias: mean bias, MRB: mean relative bias, MAB: mean absolute bias, MAPE: mean absolute percentage error, MSE: mean squared error, RMSE: root mean squared error, Percent.RMSE: percentage root mean squared error, R2: coefficient of determination, R2adj: adjusted coefficient of determination, APC: Amemiya's prediction criterion, logL: Log-likelihood, AIC: Akaike information criterion, AICc: corrected Akaike information criterion, BIC: Bayesian information criterion, HQC: Hannan-Quin information criterion.

Note

The lower the better for the SSR, TRE, Bias, MRB, MAB, MAPE, MSE, RMSE, Percent.RMSE, APC, logL, AIC, AICc, BIC and HQC indices. The higher the better for R2 and R2adj indices. Users can choose which indices to use to evaluate their models from the output. Given the difficulty of determining the number of parameters (p) in AI-based models, users might consider using error-based indices, and coefficients of determination (R2).

Author(s)

Ogana F.N. and Corral-Rivas S.

See Also

`fitstat()`, which gives the fit indices of the model based on the fitting data

Examples

```
library(EMAR)

# fitting data
Age <- 1:50
Yield <- exp(6.5 - 39.5/Age)
dat <- data.frame(Age, Yield)

# fit the model to the fitting data
Eq01 <- lm(Yield ~ Age, data=dat)

# independent/validation data
test_data <- data.frame(Age=1:50, Yield=2.5*Age^1.4)

# predict with the model i.e. Eq01, using the independent/validation data
test_data$pred.Yield <- predict(Eq01, test_data)

# Evaluate the quality of the prediction using the 'valstat' function.
# You need the observed and predicted values. Specify the number of parameters in the model.

valstat(test_data$Yield, test_data$pred.Yield, 2)
```

Index

∗ **fitstat**
    fitstat, 2
∗ **valstat**
    valstat, 3

fitstat, 2
fitstat(), *2*, *4*

valstat, 3
valstat(), *2*, *3*