Assessment, Testing and Comparison of Statistical Models using R

21 March 2021

Summary

A crucial aspect in statistical analysis, particularly with regression models, is to evaluate the quality of modelfit. During data analysis, researchers should investigate how well models fit to the data to find out whether the best model has been chosen. In the context of reporting results, fit indices should be mentioned, so that readers can judge the quality of regression models. Functions to create diagnostic plots or to compute fit measures do exist, however, these are located in many different packages, and there is no unique and consistent approach to assess the model quality for different kind of models. This makes it hard for researchers to discover the package they need or to find out whether any packages for specific regression models exist at all.

Aims of the Package

performance is an R-package (R Core Team, 2021) that provides utilities for computing measures to assess model quality, which are not directly provided by R's *base* or *stats* packages. These include measures like R^2, intraclass correlation coefficient, root mean squared error, etc., or functions to check models for overdispersion, singularity or zero-inflation, and more. These functions support a large variety of regression models, including generalized linear models, mixed-effects models, their Bayesian cousins, and more.

performance is part of the *easystats* ecosystem, a collaborative project created to facilitate the usage of R for statistical analyses (Ben-Shachar, Makowski, & Lüdecke, 2020; Lüdecke, Ben-Shachar, Patil, & Makowski, 2020; Lüdecke, Ben-Shachar, Patil, Waggoner, & Makowski, 2020; Makowski, Ben-Shachar, & Lüdecke, 2019; Makowski, Ben-Shachar, Patil, & Lüdecke, 2020).

Comparison to other Packages

Compared to other packages (e.g., *lmtest* (Zeileis & Hothorn, 2002), *MuMIn::r.squaredGLMM()* (Barton, 2020), *car* (Fox & Weisberg, 2019), *broom::glance()* (Robinson, Hayes, & Couch, 2020)), *performance* package offers functions for *both* checking validity and quality of the model, and it does so systematically and comprehensively for linear, mixed-effects, Baysian, etc. regression model objects.

Features

performance functions also include plotting capabilities via the *see* package (Lüdecke, Ben-Shachar, Patil, Waggoner, & Makowski, 2020). A complete overview of plotting functions is available at the *see* website (https://easystats.github.io/see/articles/performance.html).

Checking if a Model is Valid

When a model is specified to describe the empirical data, its validity needs to be checked by assessing if any of the underlying assumptions are violated. These assumptions vary based on the model and *performance* offers a collection of functions to check them. We will look at a couple of them before we mention the key function that runs a comprehensive suite of checks in one go.

Linear models assume constant error variance (homoskedasticity), and `check_heteroscedasticity()` functions in *performance* checks if this assumption has been violated:

```
data(cars)
model <- lm(dist ~ speed, data = cars)

check_heteroscedasticity(model)
#> Warning: Heteroscedasticity (non-constant error variance) detected (p = 0.031).
```

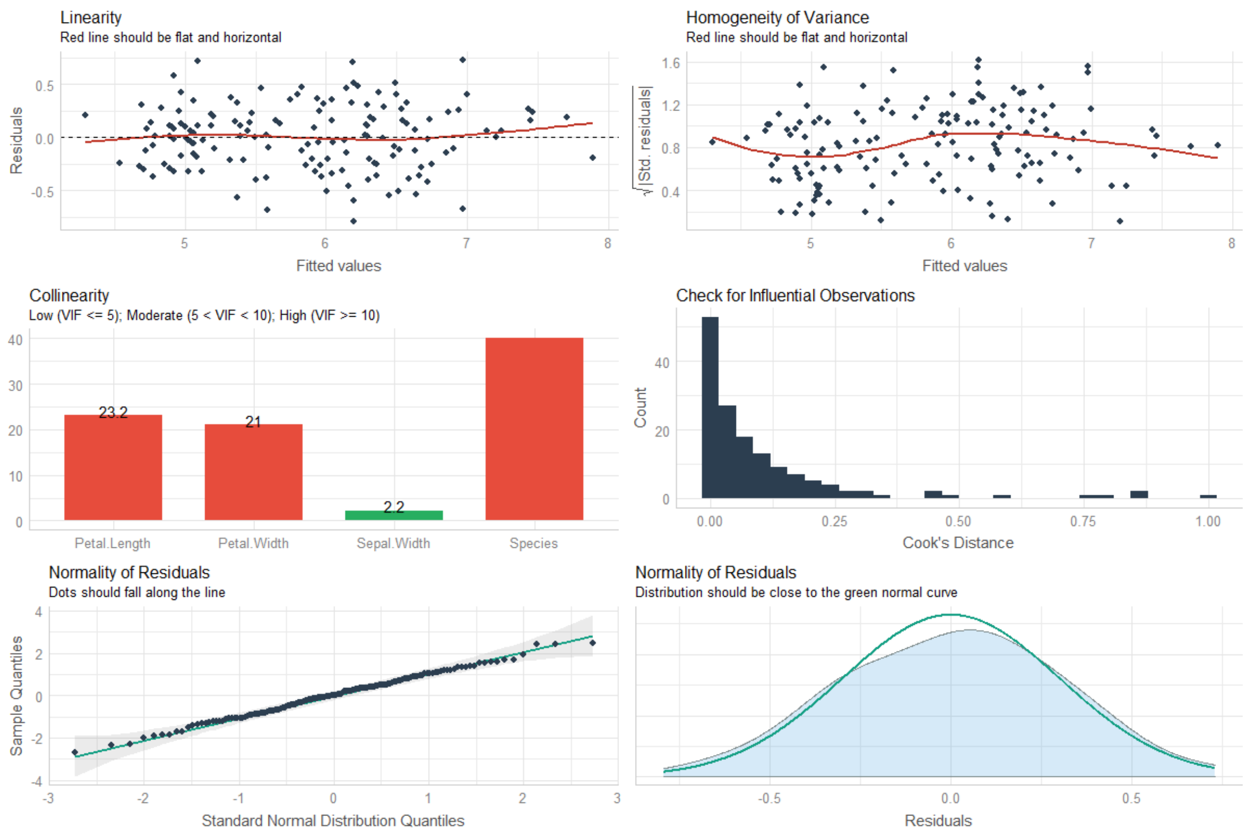
Another concern for regression models can be overdispersion, which occurs when the observed variance in the data is higher than the expected variance from the model assumption. The `check_overdispersion()` in *performance* checks this assumption.

```
library(glmmTMB)
data(Salamanders)
model <- glm(count ~ spp + mined, family = poisson, data = Salamanders)
check_overdispersion(model)
#> # Overdispersion test
#>
#>        dispersion ratio =    2.946
#>   Pearson's Chi-Squared = 1873.710
#>                 p-value =  < 0.001
```

In addition to providing such numerical indices of model fits, *performance* also provides convenience functions to *visually* assess statistical assumptions for regression models. Moreover, these visual checks adjust to the object entered and support various regression models, like linear models, linear mixed-effects models, their Bayesian equivalents, and more.

Here we show what the function output looks like for linear models:

```
library(see)
model <- lm(Sepal.Length ~ Species + Sepal.Width +
            Petal.Length + Petal.Width, data = iris)
check_model(model)
```



Computing Quality of Model

performance offers a number of indices to assess the goodness of fit of a model. We will discuss only a few before discussing a key function that returns all of these indices in one fell swoop.

For example, R^2, also known as the coefficient of determination, is a popular statistical measure to gauges how much of the variance in the dependent variable is accounted for by the specified model. The `r2()` function in *performance* can compute this index for a wide variety of regression models. Depending on the model, R^2, pseudo-R^2, or marginal / adjusted R^2, etc. values are returned.

Example with linear regression model:

```r
model <- lm(mpg ~ wt + cyl, data = mtcars)

r2(model)
#> # R2 for Linear Regression
#>
#>        R2: 0.830
#>   adj. R2: 0.819
```

Example with generalized Bayesian mixed-effects model:

```r
library(rstanarm)
model <- stan_glmer(
  Petal.Length ~ Petal.Width + (1 | Species),
  data = iris,
  cores = 4
)
```

```
r2(model)
#> # Bayesian R2 with Standard Error
#>
#>   Conditional R2: 0.953 (0.89% CI [0.944, 0.962])
#>      Marginal R2: 0.824 (0.89% CI [0.748, 0.890])
```

Similar to R^2, the Intraclass Correlation Coefficient (ICC) provides information on the explained variance and can be interpreted as the proportion of the variance explained by the grouping structure in the population (Hox, Moerbeek, & Van de Schoot, 2017). The `icc()` function in *performance* calculates the ICC for various mixed-effects regression models.

```
library(brms)
set.seed(123)
model <- brm(mpg ~ wt + (1 | cyl) + (1 + wt | gear), data = mtcars)

icc(model)
#> # Intraclass Correlation Coefficient
#>
#>      Adjusted ICC: 0.930
#>   Conditional ICC: 0.771
```

The `model_performance()` function is the workhorse of this package when it comes to extracting a comprehensive set of model fit indices from various models in a consistent manner. Depending on the regression model object, the list of computed indices might include R^2, AIC, BIC, RMSE, ICC, LOOIC, etc.

Example with linear model:

```
m1 <- lm(mpg ~ wt + cyl, data = mtcars)
model_performance(m1)
#> # Indices of model performance
#>
#> AIC     |     BIC |    R2 | R2 (adj.) |  RMSE | Sigma
#> -----------------------------------------------------
#> 156.010 | 161.873 | 0.830 |     0.819 | 2.444 | 2.568
```

Example with linear mixed-effects model:

```
library(lme4)
m3 <- lmer(Reaction ~ Days + (1 + Days | Subject), data = sleepstudy)
model_performance(m3)
#> # Indices of model performance
#>
#> AIC      |      BIC | R2 (cond.) | R2 (marg.) |   ICC |   RMSE |  Sigma
#> -----------------------------------------------------------------------
#> 1755.628 | 1774.786 |      0.799 |      0.279 | 0.722 | 23.438 | 25.592
```

Comparing Multiple Models

For multiple models, one can obtain a useful table to compare these indices at a glance using the `compare_performance()` function.

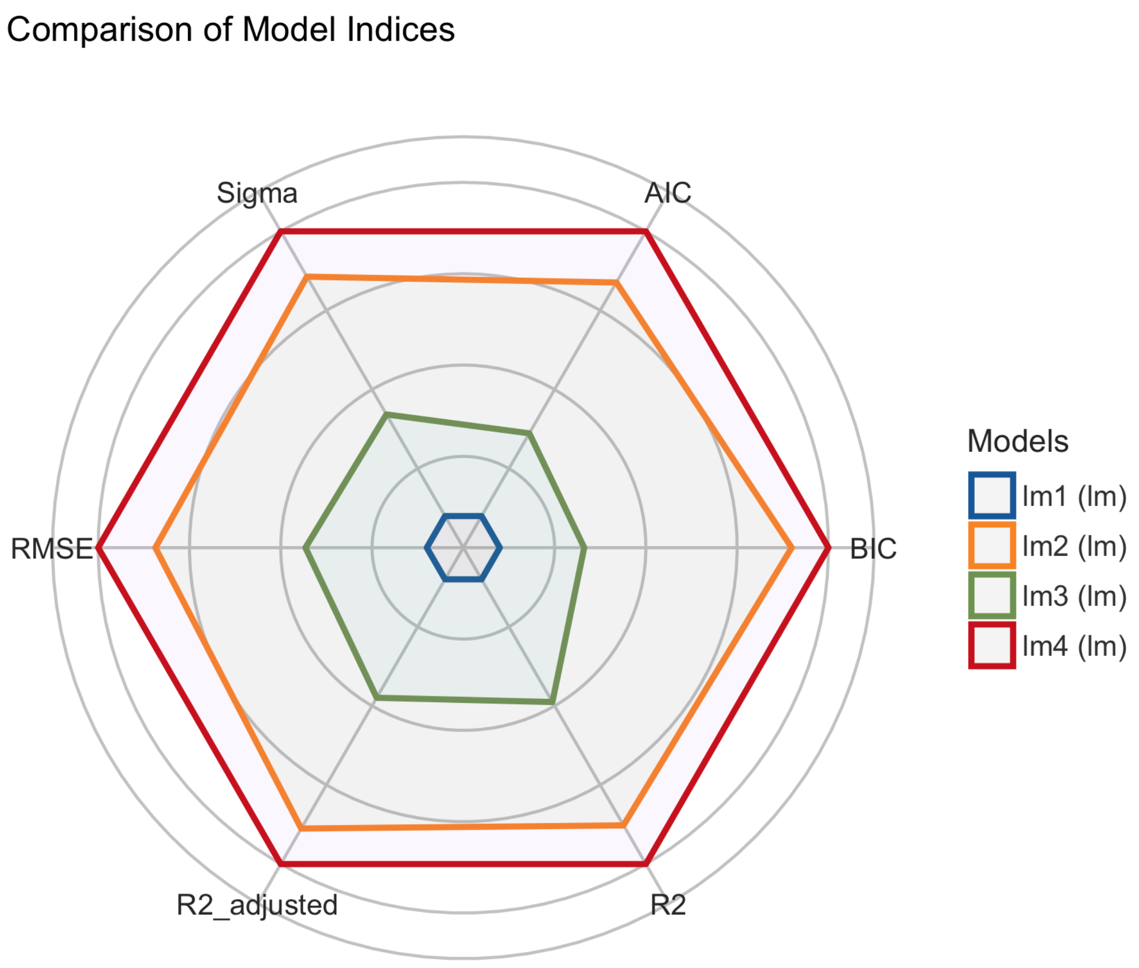
```
data(iris)

lm1 <- lm(Sepal.Length ~ Species, data = iris)
lm2 <- lm(Sepal.Length ~ Species + Petal.Length, data = iris)
lm3 <- lm(Sepal.Length ~ Species * Sepal.Width, data = iris)
lm4 <- lm(Sepal.Length ~ Species * Sepal.Width +
          Petal.Length + Petal.Width, data = iris)

compare_performance(lm1, lm2, lm3, lm4)
#> # Comparison of Model Performance Indices
#>
#> Name | Model |     AIC |     BIC |    R2 | R2 (adj.) |  RMSE | Sigma
#> ---------------------------------------------------------------------
#> lm1  |    lm | 231.452 | 243.494 | 0.619 |     0.614 | 0.510 | 0.515
#> lm2  |    lm | 106.233 | 121.286 | 0.837 |     0.833 | 0.333 | 0.338
#> lm3  |    lm | 187.092 | 208.167 | 0.727 |     0.718 | 0.431 | 0.440
#> lm4  |    lm |  78.797 | 105.892 | 0.871 |     0.865 | 0.296 | 0.305
```

Similarly, in addition to the formal tests to compare several models, *performance* also provides visual ways to compare model fit indices.

```
library(see)
plot(compare_performance(lm1, lm2, lm3, lm4))
```



Testing Models

While **comparing** these indices is often useful, making a decision (for instance, which model to keep or drop) can often be hard, as the indices can give conflicting suggestions. Additionally, it is sometimes unclear which index to favour in the given context.

This is one of the reason why **tests** are useful, as they facilitate decisions via "significance" indices, like p-values (in Frequentist framework) or Bayes Factors (in Bayesian framework).

The generic `test_performance()` runs the most relevant and appropriate tests based on the input. For instance, in the example below, the results from *Vuong's Test* are displayed:

```
test_performance(lm1, lm2, lm3, lm4)
#> Name | Model | Omega2 | p (Omega2) |    LR | p (LR)
#> ---------------------------------------------------
#> lm1  |    lm |        |            |       |
#> lm2  |    lm |   0.69 |     < .001 | -6.25 | < .001
#> lm3  |    lm |   0.36 |     < .001 | -3.44 | < .001
#> lm4  |    lm |   0.73 |     < .001 | -7.77 | < .001
#> Each model is compared to lm1.
```

For Bayesian framework, *performance* also provides `test_bf()` function to compare models:

```
test_bf(lm1, lm2, lm3, lm4)
#> # Bayes Factors for Model Comparison
#>
#> Model                                                           BF
#> [lm2] Species + Petal.Length                             3.446e+26
#> [lm3] Species * Sepal.Width                              4.692e+07
#> [lm4] Species * Sepal.Width + Petal.Length + Petal.Width 7.584e+29
#>
#> * Against Denominator: [lm1] Species
#> *   Bayes Factor Type: BIC approximation
```

Licensing and Availability

performance is licensed under the GNU General Public License (v3.0), with all source code stored at GitHub (https://github.com/easystats/performance), and with a corresponding issue tracker for bug reporting and feature enhancements. In the spirit of honest and open science, we encourage requests/tips for fixes, feature updates, as well as general questions and concerns via direct interaction with contributors and developers.

Acknowledgments

performance is part of the collaborative *easystats* ecosystem. Thus, we would like to thank the members of easystats as well as the users.

References

Barton, K. (2020). *MuMIn: Multi-model inference.* Retrieved from https://CRAN.R-project.org/package=MuMIn

Ben-Shachar, M. S., Makowski, D., & Lüdecke, D. (2020). effectsize: Compute and interpret indices of effect size. *CRAN*. https://doi.org/10.5281/zenodo.3952214

Fox, J., & Weisberg, S. (2019). *An R companion to applied regression* (Third). Retrieved from https://socialsciences.mcmaster.ca/jfox/Books/Companion/

Hox, J. J., Moerbeek, M., & Van de Schoot, R. (2017). *Multilevel analysis: Techniques and applications.* Routledge.

Lüdecke, D., Ben-Shachar, M. S., Patil, I., & Makowski, D. (2020). Parameters: Extracting, computing and exploring the parameters of statistical models using R. *Journal of Open Source Software*, *5*(53), 2445. https://doi.org/10.21105/joss.02445

Lüdecke, D., Ben-Shachar, M. S., Patil, I., Waggoner, P., & Makowski, D. (2020). see: Visualisation toolbox for 'easystats' and extra geoms, themes and color palettes for 'ggplot2'. *CRAN*. https://doi.org/10.5281/zenodo.3952153

Makowski, D., Ben-Shachar, M., & Lüdecke, D. (2019). bayestestR: Describing effects and their uncertainty, existence and significance within the bayesian framework. *Journal of Open Source Software*, *4*(40), 1541. https://doi.org/10.21105/joss.01541

Makowski, D., Ben-Shachar, M. S., Patil, I., & Lüdecke, D. (2020). Methods and algorithms for correlation analysis in r. *Journal of Open Source Software*, *5*(51), 2306. https://doi.org/10.21105/joss.02306

R Core Team. (2021). *R: A language and environment for statistical computing*. Retrieved from https://www.R-project.org/

Robinson, D., Hayes, A., & Couch, S. (2020). *broom: Convert statistical objects into tidy tibbles*. Retrieved from https://CRAN.R-project.org/package=broom

Zeileis, A., & Hothorn, T. (2002). Diagnostic checking in regression relationships. *R News*, *2*(3), 7–10. Retrieved from https://CRAN.R-project.org/doc/Rnews/