performance: An R Package for Assessment, Comparison and Testing of Statistical Models

Daniel Lüdecke[1], Mattan S. Ben-Shachar[2], Indrajeet Patil[3], Philip Waggoner[4], and Dominique Makowski[5]

1 University Medical Center Hamburg-Eppendorf, Germany **2** Ben-Gurion University of the Negev, Israel **3** Center for Humans and Machines, Max Planck Institute for Human Development, Berlin, Germany **4** University of Chicago, USA **5** Nanyang Technological University, Singapore

Summary

A crucial part of statistical analysis is evaluating a model's quality and fit, or *performance*. During analysis, especially with regression models, investigating the fit of models to data also often involves selecting the best fitting model amongst many competing models. Upon investigation, fit indices should also be reported both visually and numerically to bring readers in on the investigative effort.

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

Statement of Need

While functions to build and produce diagnostic plots or to compute fit statistics exist, these are located across many packages, which results in a lack of a unique and consistent approach to assess the performance of many types of models. The result is a difficult-to-navigate, unorganized ecosystem of individual packages with different syntax, making it onerous for researchers to locate and use fit indices relevant for their unique purposes. The *performance* package in R fills this gap by offering researchers a suite of intuitive functions with consistent syntax for computing, building, and presenting regression model fit statistics and visualizations.

performance is part of the *easystats* ecosystem, which is a collaborative project focused on facilitating simple and intuitive usage of R for statistical analysis (Ben-Shachar et al., 2020; Lüdecke, Ben-Shachar, Patil, Waggoner, et al., 2020; Lüdecke, Ben-Shachar, Patil, & Makowski, 2020; Makowski et al., 2019, 2020).

Comparison to other Packages

Compared to other packages (e.g., *lmtest* (Zeileis & Hothorn, 2002), *MuMIn* (Barton, 2020), *car* (Fox & Weisberg, 2019), *broom* (Robinson et al., 2020)), the *performance* package offers functions for checking validity *and* model quality systematically and comprehensively for many regression model objects such as (generalized) linear models, mixed-effects models, and Bayesian models. *performance* also offers functions to compare and test multiple models simultaneously to evaluate the best fitting model to the data.

Features

Beyond validity and quality checks, *performance* also includes plotting functions via the *see* package (Lüdecke, Ben-Shachar, Patil, Waggoner, et al., 2020).[1]

Checking Model Assumptions

Inferences made from regression models such as significance tests or interpretation of coefficients require meeting several assumptions, which vary based on the type of model. *performance* offers a collection of functions to check if assumptions are met. To demonstrate the efficiency of the package, we provide examples for a few functions, followed by a broader function that runs a comprehensive suite of checks in a single call.

For example, linear (Gaussian) models assume constant error variance (homoscedasticity). We can use `check_heteroscedasticity()` from *performance* to check if this assumption has been violated.

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

check_heteroscedasticity(model)

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

For another example, Poisson regression models assume equidispersion. Violating this assumption leads to *overdispersion*, which occurs when the observed variance in the data is higher than the expected variance from the model. We can call `check_overdispersion()` to check if overdispersion is an issue.

```r
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
#>
#> Overdispersion detected.
```

In addition to tests for checking assumptions, *performance* also provides convenience functions to *visually* assess these assumptions of regression models. *performance*'s visual checks detect the type of model passed to the function call, and return the appropriate visual checks for each model type. At present, there are many supported regression models, such as linear models, linear mixed-effects models or their Bayesian equivalents. Inspect the package documentation for a complete listing.
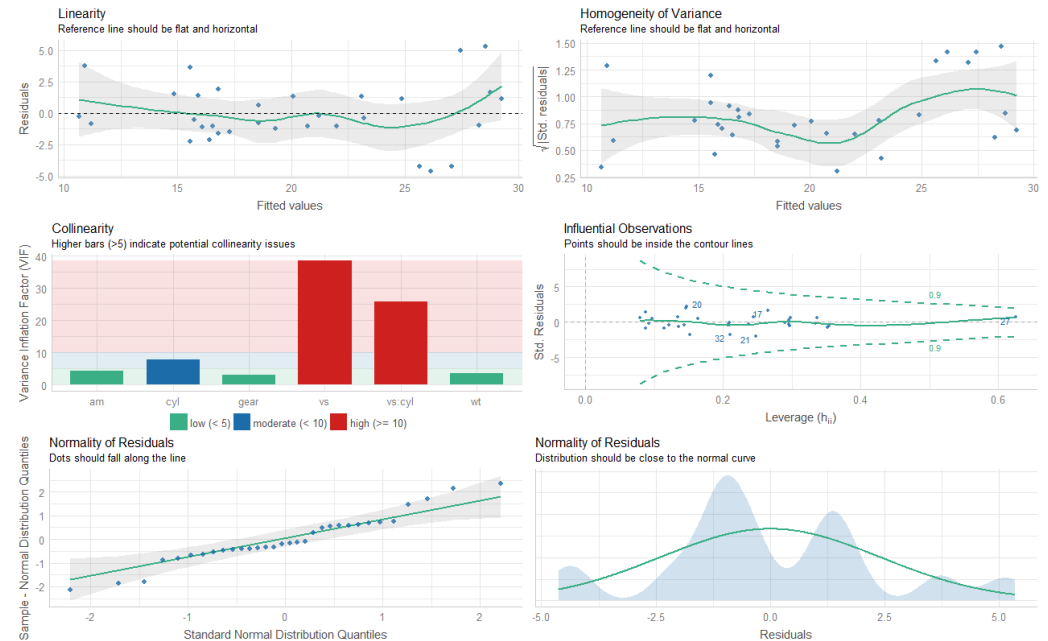
For example, consider the visual checks from a simple linear regression model.

---

[1]A complete overview of plotting functions is available at the *see* website (https://easystats.github.io/see/articles/performance.html).

```
library(see)

model <- lm(mpg ~ wt + am + gear + vs * cyl,
            data = mtcars)

check_model(model)
```


Computing Quality Indices of Models

performance offers a number of indices to assess the goodness of fit of a model. For example, R^2, also known as the coefficient of determination, is a popular statistical measure to gauge the amount of the variance in the dependent variable accounted for by the specified model. The `r2()` function from *performance* computes and returns this index for a variety of regression models. Depending on the model, the returned value may be R^2, pseudo-R^2, or marginal/adjusted R^2.

First, consider a simple linear model.

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

r2(model)

#> # R2 for Linear Regression
#>        R2: 0.830
#>   adj. R2: 0.819
```

Next, consider a mixed-effects model.

```
library(lme4)
model <- lmer(
  Petal.Length ~ Petal.Width + (1 | Species),
  data = iris
```

```
)

r2(model)

#> # R2 for Mixed Models
#>
#>    Conditional R2: 0.933
#>      Marginal R2: 0.303
```

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 et al., 2017). The `icc()` function from *performance* computes and returns 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: 0.874
#>   Conditional ICC: 0.663
```

Instead of computing and returning individual indices, users can obtain *all* indices from the model by simply passing the fitted model object to `model_performance()`. A list of computed indices is returned, which might include R^2, AIC, BIC, RMSE, ICC, LOOIC, etc.

For example, consider a simple 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
```

Next, consider a 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, users can inspect a table of these indices by calling, `compare_performance()`.

```
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
```
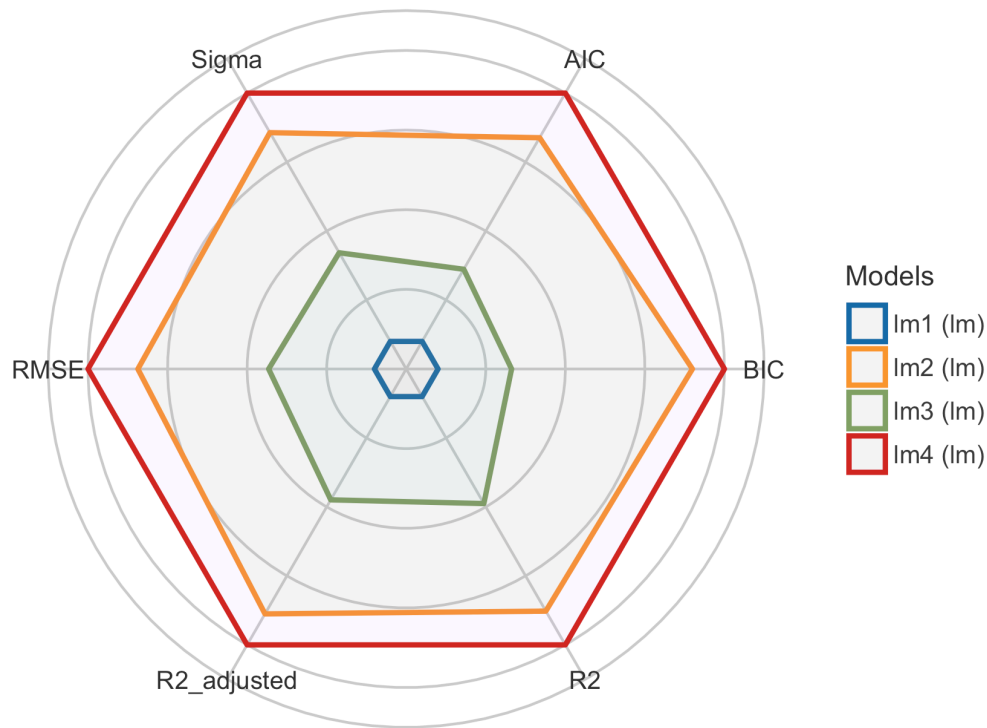
As noted previously, in addition to the returning numeric results, *performance* also offers visualizations of model fit indices.

```
library(see)

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

Comparison of Model Indices

Testing Models

While comparing these indices is often useful, making a decision such as whether to keep or drop a model, can often be difficult as some indices can give conflicting suggestions. Additionally, it may be unclear which index to favour in different contexts. This difficulty is one of the reasons why *tests* are often useful as they facilitate decisions via "significance" indices like *p*-values (in a frequentist framework) or Bayes Factors (in a Bayesian framework).

The generic `test_performance()` function computes the appropriate test(s) based on the supplied input. For instance, the following example shows results from *Vuong's Test* (Vuong, 1989).

```
test_performance(lm1, lm2, lm3, lm4)
```

```
#> Name | Model |      BF | Omega2 | p (Omega2) |    LR | p (LR)
#> ------------------------------------------------------------
#> lm1  |    lm |         |        |            |       |
#> lm2  |    lm | > 1000 |   0.69 |     < .001 | -6.25 | < .001
#> lm3  |    lm | > 1000 |   0.36 |     < .001 | -3.44 | < .001
#> lm4  |    lm | > 1000 |   0.73 |     < .001 | -7.77 | < .001
#> Each model is compared to lm1.
```

performance also provides `test_bf()` to compare models via Bayes factors in a Bayesian framework.

Lüdecke, (2021). performance: An R Package for Assessment, Comparison and Testing of Statistical Models. *Journal of Open Source Software*, 6(60), 3139. https://doi.org/10.21105/joss.03139

```
test_bf(lm1, lm2, lm3, lm4)

#> Bayes Factors for Model Comparison
#>
#>       Model                                               BF
#> [lm2] Species + Petal.Length                           > 1000
#> [lm3] Species * Sepal.Width                            > 1000
#> [lm4] Species * Sepal.Width + Petal.Length + Petal.Width > 1000
#>
#> * 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 thank the members of easystats as well as the users.

References

Barton, K. (2020). *MuMIn: Multi-model inference*. 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). Sage. ISBN: 9781544336473

Hox, J. J., Moerbeek, M., & Van de Schoot, R. (2017). *Multilevel analysis: Techniques and applications*. Routledge. ISBN: 9781138121362

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*. R Foundation for Statistical Computing. https://www.R-project.org/

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

Vuong, Q. H. (1989). Likelihood Ratio Tests for Model Selection and Non-Nested Hypotheses. *Econometrica*, *57*(2), 307. https://doi.org/10.2307/1912557

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