The Journal of Open Source Software

modelbased: An R package to make the most out of your statistical models through marginal means, marginal effects, and model predictions

Dominique Makowski[1, 2], **Mattan S. Ben-Shachar**[3], **Brenton M. Wiernik**[4], **Indrajeet Patil**[5], **Rémi Thériault**[6], **and Daniel Lüdecke**[7]

1 School of Psychology, University of Sussex, Brighton, UK **2** Sussex Centre for Consciousness Science, University of Sussex, Brighton, UK **3** Independent Researcher, Ramat Gan, Israel **4** Independent Researcher, Tampa, FL, USA **5** Center for Humans and Machines, Max Planck Institute for Human Development, Berlin, Germany **6** Department of Psychology, New York University, New York, NY, USA **7** Institute of Medical Sociology, University Medical Center Hamburg-Eppendorf, Germany

DOI:

Software

- Review
- Repository
- Archive

Submitted:
Published:

Statement of need

Applied statistics have historically focused on statistical *tests* (e.g., *t*-tests, correlation tests, and analyses of variances, ANOVAs), seen as most apt to provide researchers with interpretable answers to the questions they seek. These tests, however, typically rely on statistical *models* — the true underlying cornerstone of modern data science. The replication crisis (Camerer et al., 2018; Open Science Collaboration, 2015) and methodological (r)evolutions (Makowski & Waggoner, 2023) have underlined some of the issues with the traditional focus on statistical tests (e.g., the effacement of model assumptions, an emphasis on null-hypothesis testing, non-compatibility with more complex variance structures) and called for shifting the focus to the models themselves.

In line with these efforts, new tools have been created to facilitate the direct usage and reporting of statistical models. For instance, the `easystats` collection of R packages (Lüdecke et al., 2023) has been developed to help researchers "tame, discipline, and harness" the power of statistical models. Within this framework, specific packages are dedicated to model parameters (the `parameters` package, Lüdecke et al., 2020), predictive performance (the `performance` package, Lüdecke et al., 2021) or effect importance (the `effectsize` package, Ben-Shachar et al., 2020).

But the models themselves withhold even more usefulness!

The fundamental nature of these models—a statistical link between an outcome y and predictor variables X—enables the generation of predictions for any observed or unobserved combination of predictors. These predictions refer to expected values of the outcome for given levels of predictors of interest, making it possible to test and visualize the model's behaviour in a more meaningful and comprehensive way, and answering a broad range of research questions.

The two most popular R packages for extracting these quantities of interest from statistical models are `emmeans` (Lenth, 2024) and `marginaleffects` (Arel-Bundock et al., 2024). These packages pack an enormously rich set of features and cover (almost) all imaginable needs for post-hoc analysis of statistical models. However, their power and flexibility come at a cost: ease of use—especially for users not familiar with the underlying statistical concepts. The `modelbased` package, built on top of these two packages, aims to unleash this vast, untapped potential by providing a unified interface to extract marginal means, marginal effects, contrasts, comparisons, and model predictions from a wide range of

statistical models. In line with the `easystats`' *raison d'être*, the `modelbased` package focuses on simplicity, flexibility, and user-friendliness to help researchers harness the full power of their models.

Key concepts

Answering research questions based on statistical models means describing the relationship between predictors of interest (also called *focal* predictors) and the outcome, as well as differences between observed groups in the sample. There are four key concepts in `modelbased` to achieve this: Predictions, and Marginal Means, Effects, and Contrasts.

Predictions

At a fundamental level, `modelbased` and similar packages leverage model *predictions*. These predictions can be of different types, depending on the model and the question at hand. For instance, predictions can be associated with **confidence intervals** (`predict = "expectation"`) or **prediction intervals** (`predict = "prediction"`). The former corresponds to the uncertainty around the "relationship" (i.e., the conditional estimate, typically of the expectation ($E[X]$) according to a model's parameters), while the latter is typically larger and provides information about the range which individual observations might fall in. Moreover, for generalized linear models (GLMs), predictions can be made on the **response scale** (`predict = "response"`) or the **link scale** (`predict = "link"`). This corresponds for instance to predictions in terms of probability (response scale) or log odds (link scale) for logistic regression models.

These different types of estimates can be obtained for observations in the original dataset - which is useful to assess the model's goodness-of-fit - for new data (typically a "data grid"), which is useful for visualization.

For convenience, the `modelbased` package includes four related functions, which mostly differ in their default arguments for `data` and `predict`[1]:

- `estimate_prediction()`: original data, prediction intervals.
- `estimate_expectation()`: original data, confidence intervals.
- `estimate_relation()`: data grid, predictions on the response scale.
- `estimate_link()`: data grid, predictions on the link scale.

Marginal means, contrasts and effects

Means

The concept of "marginal" in this context refers to how non-focal predictors (i.e., those not of direct interest, for instance "adjustment" variables added to "control" for it) are treated. While predictions, as described above, fix by default non-focal variables at their reference level, marginal means compute the empirical or theoretical averages over them. These kind of predictions are a good representation of the sample, because they are not based on very specific characteristics. For example, predictions can be made for specific combinations of predictors, such as people with *high* income, while marginal means might calculate the expected outcome for an *average* observation (averaged over income).

The `modelbased` package provides a simple and clear interface to extract marginal means via the `estimate_means()` function (with focal predictors specified using the `by` argument), which can be considered as "marginal" pendant to `estimate_relation()`.

[1] These functions can become redundant if the defaults are changed. For instance, `estimate_relation(..., predict = "link")` is equivalent to `estimate_link(...)`.

Contrasts

The computation of these model-based quantities allow for the direct statistical comparison of the predicted outcomes across different groups or levels of a focal predictor, in the form of marginal contrasts. Rather than simply observing differences in marginal means, contrast analysis quantifies these differences and assesses their statistical significance, as well as its associated uncertainty (e.g., confidence intervals).

In `modelbased`, this can be achieved using the `estimate_contrasts()` function. Focal predictors are specified in the `contrast` argument. As with the other functions, further stratification or grouping can be made with the `by` argument (for instance, to analyze the difference between two levels of a factor alongside the values of another interacting predictor).

Effects

Finally, the same approach can be used for the effects, i.e., the parameters of the model. While predictions and marginal means can be used to better understand the *relationship* of predictors with the outcome (e.g., to estimate "the average health score for a person at the age of sixty is 80 points"), marginal *effects* evaluate the (average) *effect* of a parameter (often called "slope"), telling you that "the average effect (i.e., relationship) of age on the health score is a decrease of 5 points per year".

For the simple case of linear regression without interaction terms, the regression coefficient (slope) equals the marginal effect. However, in even slightly more complex situations (e.g., with interactions or non-linear effects), the slope is not constant across the predictor's values, and estimating the *average* slope, or marginal effect, can become useful.

Again, the `modelbased` package has a simple function to do so, `estimate_slopes()`. This function calculates the *trend* or average effect, usually for numeric predictors. The `trend` argument specifies the focal predictors, while `by` allows for further grouping.

Marginalization Types

Until this point we have discussed marginal means and effects as being "averaged" over non-focal predictors, but there are actually various ways of doing so. The `estimate_means()`, `estimate_contrasts()`, and `estimate_slopes()` have an `estimate` argument that determines how predictions are averaged ("marginalized"). The options are:

- **"typical"** (default): Calculates predictions for a balanced data grid representing all combinations of focal predictor levels (specified in `by`). For non-focal numeric predictors, it uses the mean; for non-focal categorical predictors, it averages over all the levels. This represents a "typical" observation based on the data grid and is useful for comparing groups. It answers: *"What would the average outcome be for a 'typical' observation?"*. This is the default approach when estimating marginal means using the `emmeans` package.
- **"average"**: Calculates predictions for each observation in the sample and then averages these predictions within each group defined by the focal predictors. This reflects the sample's actual distribution of non-focal predictors, not a balanced grid. It answers: *"What is the predicted value for an average observation in my data?"*.
- **"population"**: "Clones" each observation, creating copies with all possible combinations of focal predictor levels. It then averages the predictions across these "counterfactual" observations (non-observed permutations) within each group. This extrapolates to a hypothetical broader population, considering "what if" scenarios. It answers: *"What is the predicted response for the 'average' observation in a broader possible target population?"*. This approach entails more assumptions about the likelihood of different combinations, but can be more apt to generalize.

Setting `estimate = "average"` can be useful to calculate the average expected outcome from those observations *from the sample* at hand. For analyses emphasizing outcome differences between groups (e.g., when computing contrasts) and particularly when causal effects are being considered, it may be beneficial to model a hypothetical population not directly represented in the sample. This approach, known as *G-computation* (Chatton & Rohrer, 2024), is implemented by setting `estimate = "population"`.

Group-level estimates for Mixed Models

The `modelbased` package also provides the `estimate_grouplevel()` function to conveniently extract parameters related to random factors, which typically correspond to group-level parameters (e.g., the intercept's or slope's value for each participant). Estimating these indices using mixed models can have important benefits (such as shrinkage and better convergence) over an empirical approach consisting of fitting individual models to all individuals separately. These group-level estimates can be estimated in two manners:

- **"random"** (default): Corresponds to the deviation of each individual group from their fixed effect. As such, a coefficient close to 0 means that the participants' effect is the same as the population-level effect
- **"total"**: Returns the sum of the random effect and its corresponding fixed effects. These are known as BLUPs (Best Linear Unbiased Predictions) and are "absolute" values of the individual-level effects on the same scale as their corresponding fixed effects.

Technical details

The algorithmic heavy lifting is done by `modelbased`'s two back-end packages, `emmeans` and `marginaleffects` (the default), which can be set as a global option (e.g., with `options(modelbased_backend = "emmeans")`).

Of the two, `emmeans` (Lenth, 2024) is the more senior package and was originally known as `lsmeans` (for "Least-Squares Means"). This term has been historically used to describe what are now more commonly referred to as "Estimated Marginal Means" or EMMs: predictions made over a regular grid—a grid typically constructed from all possible combinations of the categorical predictors in the model and the mean of numerical predictors. The package was renamed in 2016 to `emmeans` to clarify its extension beyond least-squares estimation and its support of a wider range of models (e.g., Bayesian models).

Within `emmeans`, estimates are generated as a linear function of the model's coefficients, with standard errors (SEs) produced in a similar manner by taking a linear combination of the coefficients' variance-covariance matrix. For example if b is a vector of 4 coefficients, and V is a 4-by-4 matrix of the coefficients' variance-covariance, we can get an estimate and SE for a linear combination (or set of linear combinations) L like so:

$$\hat{b} = L \cdot b$$

$$SE_{\hat{b}} = \sqrt{\text{diag}(L \cdot V \cdot L^T)}$$

These grid predictions are sometimes averaged over (averaging being a linear operation itself) to produce "marginal" predictions (in the sense of marginalized-over): means. These predictions can then be contrasted using various built-in or custom contrasts weights to obtain meaningful estimates of various effects. Using linear combinations with regular grids often means that results from `emmeans` directly correspond to a models coefficients

(which is a benefit for those who are used to understanding models by examining coefficient tables).

`marginaleffects` (Arel-Bundock et al., 2024) was more recently introduced, and uses a different approach: various effects are estimated by generating two counter-factual predictions of unit-level observations, and then taking the difference between them (with SEs computed using the delta method). By default, such effects are estimated for every observation in the original model frame. These unit-level effects are typically averaged to obtain average effects (e.g., an Average Treatment Effect, ATE).

Using the delta method affords more flexibility in the specification of the marginal effects to be estimated. For example, while `emmeans` by default compares predictions from GLMs on the link scale and then transforms the comparison back to something closer to the response scales (e.g., the difference between two log-odds is taken, and then exponentiation to produce odds ratios), `marginaleffects` by default compares predictions on the response scale directly (e.g, taking the difference between two probabilities). The delta method implemented in `marginaleffects` uses iterative estimation, making it more computationally costly relative to the simple matrix multiplication used for estimating linear combinations (though `marginaleffects` is very efficient).

This means that while `emmeans` typically produces *effects at the mean*, `marginaleffects` typically produces *mean effects*. Depending on the quantity of interest, model, use of a link function, design balance and weights, these can be nearly identical, or very different.

Note that `emmeans` can also use the delta method and can use non-regular grids, and `marginaleffects` can also generate linear predictions at the mean. However, obtaining these requires some deeper knowledge of the relevant packages. This is easier to achieve and more accessible in `modelbased`, by simply modulating the `estimate` argument (see above).

Finally, `modelbased` leverages the `get_datagrid()` function from the `insight` package (Lüdecke et al., 2019) to intuitively generate an appropriate grid of data points for which predictions or effects or slopes will be estimated. Since these packages support a wider range of models - including generalized linear models, mixed models, and Bayesian models - `modelbased` also inherits the support for such models.

Examples

The `iris` dataset contains measures in centimeters of three different species of iris flowers (setosa, versicolor, and virginica, Becker et al., 1988). Imagine the following linear model in which we predict those flowers' petal width (`Petal.Width`) from the interaction between their petal length (`Petal.Length`) and their `Species`.

```
library(easystats)

model <- lm(Petal.Width ~ Petal.Length * Species, data = iris)

parameters::parameters(model) |>
  print(select = "minimal")
```

```
#> Parameter                          | Coefficient |        95% CI |      p
#> ------------------------------------------------------------------------
#> (Intercept)                        |       -0.05 | [-0.47, 0.38] | 0.823
#> Petal Length                       |        0.20 | [-0.09, 0.49] | 0.170
#> Species [versicolor]               |       -0.04 | [-0.66, 0.59] | 0.909
#> Species [virginica]                |        1.18 | [ 0.52, 1.84] | < .001
#> Petal Length × Species [versicolor] |        0.13 | [-0.18, 0.44] | 0.405
```

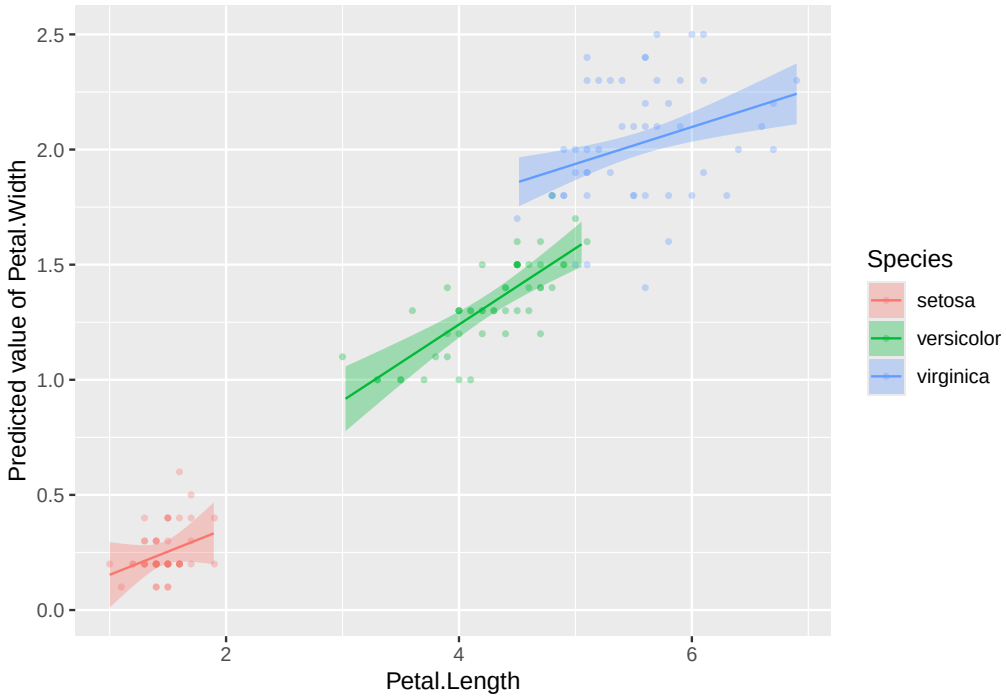

Figure 1: Scatter plot of petal length by pelal width, grouped by species

```
#> Petal Length × Species [virginica]  |       -0.04 | [-0.34, 0.26] | 0.789
#>
#> Uncertainty intervals (equal-tailed) and p-values (two-tailed) computed
#>   using a Wald t-distribution approximation.
```

The model's **parameters** can be challenging to interpret and do not offer us all the insights that this model actually contains.

Visualize relationship

We can start by easily visualizing the relationship between our response variable and our predictors (Figure 1).

```
estimate_relation(model, by = c("Petal.Length", "Species"), length = 100) |>
  plot(show_data = TRUE)
```

But what is the **average value** of `Petal.Width` for each species?

Marginal Means

The marginal means can be computed, which are the mean predictions for each level of a categorical predictor, *averaged across* all levels of other predictors (`Petal.Length` in this case).

```
estimate_means(model, by = "Species")
```

```
#> Estimated Marginal Means
#>
#> Species    | Mean |   SE |       95% CI | t(144)
#> ------------------------------------------------
#> setosa     | 0.71 | 0.34 | [0.04, 1.37] |   2.11
```

```
#> versicolor | 1.16 | 0.04 | [1.09, 1.23] |  31.44
#> virginica  | 1.74 | 0.09 | [1.57, 1.91] |  20.20
#>
#> Variable predicted: Petal.Width
#> Predictors modulated: Species
#> Predictors averaged: Petal.Length (3.8)
```

However, are these different species **significantly different** from each other?

Marginal Contrasts

We can estimate all the pairwise contrasts between the levels of the `Species` factor.

```
estimate_contrasts(model, contrast = "Species")
```

```
#> Marginal Contrasts Analysis
#>
#> Level1     | Level2     | Difference |   SE |        95% CI | t(144) |      p
#> ------------------------------------------------------------------------------
#> versicolor | setosa     |       0.45 | 0.34 | [-0.22, 1.12] |   1.34 |  0.183
#> virginica  | setosa     |       1.03 | 0.35 | [ 0.35, 1.72] |   2.97 |  0.003
#> virginica  | versicolor |       0.58 | 0.09 | [ 0.39, 0.76] |   6.18 | < .001
#>
#> Variable predicted: Petal.Width
#> Predictors contrasted: Species
#> Predictors averaged: Petal.Length (3.8)
#> p-values are uncorrected.
```

As we can see, the average difference between *versicolor* and *setosa* is not significant.

Marginal Slopes

Similarly, we can compute the marginal effect of `Petal.Length` (i.e., the "slope") for each species.

```
estimate_slopes(model, trend = "Petal.Length", by = "Species")
```

```
#> Estimated Marginal Effects
#>
#> Species    | Slope |   SE |        95% CI |    t |      p
#> ------------------------------------------------------------
#> setosa     |  0.20 | 0.15 | [-0.08, 0.49] | 1.38 |  0.168
#> versicolor |  0.33 | 0.05 | [ 0.23, 0.44] | 6.14 | < .001
#> virginica  |  0.16 | 0.05 | [ 0.07, 0.25] | 3.49 | < .001
#>
#> Marginal effects estimated for Petal.Length
#> Type of slope was dY/dX
```

This shows that there is a significant positive relationship between `Petal.Length` and `Petal.Width` for all species but *setosa*.

Marginal Contrasts of Slopes

Finally, we can even compute the contrasts between the slopes of `Petal.Length` for each species.

```
estimate_contrasts(model, contrast = "Petal.Length", by = "Species")
```

```
#> Marginal Contrasts Analysis
```

```
#>
#> Level1     | Level2     | Difference |   SE |         95% CI |     t |     p
#> ------------------------------------------------------------------------------
#> versicolor | setosa     |       0.13 | 0.16 | [-0.17,  0.43] |  0.83 | 0.404
#> virginica  | setosa     |      -0.04 | 0.15 | [-0.34,  0.26] | -0.27 | 0.789
#> virginica  | versicolor |      -0.17 | 0.07 | [-0.31, -0.03] | -2.41 | 0.016
#>
#> Variable predicted: Petal.Width
#> Predictors contrasted: Petal.Length
#> Predictors averaged: Petal.Length (3.8)
#> p-values are uncorrected.
```

The effect of `Petal.Length` on `Petal.Width` is significantly stronger in *virginica* compared to *versicolor*.

Conclusion

The `modelbased` package provides a simple and intuitive interface to extract and visualize important information contained within statistical models.

Declarations

Funding information

This research received no external funding.

Competing Interests

The authors declare no conflict of interest

Availability of data and materials (data transparency)

All data used in this paper uses data included with base R.

Code availability

The `modelbased` package is available at the official website (https://easystats.github.io/modelbased), on CRAN (https://cran.r-project.org/package=modelbased), and on the R-Universe (https://easystats.r-universe.dev/modelbased). The source code is available on GitHub (https://github.com/easystats/modelbased), and the package can be installed from CRAN with `install.packages("modelbased")` or from R-Universe with `install.packages("modelbased", repos = "https://easystats.r-universe.dev")`.

Contributions

DM: Writing- Original draft preparation, Writing- Reviewing and Editing, Software. MSB-S, BMW, IP, RT, and DL: Writing- Reviewing and Editing, Software.

Acknowledgements

{modelbased} is part of the collaborative *easystats* ecosystem (Lüdecke et al., 2023). Thus, we thank all members of easystats, contributors, and users alike.

References

Arel-Bundock, V., Greifer, N., & Heiss, A. (2024). How to interpret statistical models using marginaleffects for R and Python. *Journal of Statistical Software*, *111*, 1–32. https://doi.org/10.18637/jss.v111.i09

Becker, R. A., Chambers, J. M., & Wilks, A. R. (1988). The new S language. *Cole.[Google Scholar]*.

Ben-Shachar, M. S., Lüdecke, D., & Makowski, D. (2020). effectsize: Estimation of effect size indices and standardized parameters. *Journal of Open Source Software*, *5*(56), 2815. https://doi.org/10.21105/joss.02815

Camerer, C. F., Dreber, A., Holzmeister, F., Ho, T.-H., Huber, J., Johannesson, M., Kirchler, M., Nave, G., Nosek, B. A., Pfeiffer, T., Altmejd, A., Buttrick, N., Chan, T., Chen, Y., Forsell, E., Gampa, A., Heikensten, E., Hummer, L., Imai, T., … Wu, H. (2018). Evaluating the replicability of social science experiments in Nature and Science between 2010 and 2015. *Nature Human Behaviour*, *2*, 637–644. https://doi.org/10.1038/s41562-018-0399-z

Chatton, A., & Rohrer, J. M. (2024). The causal cookbook: Recipes for propensity scores, g-computation, and doubly robust standardization. *Advances in Methods and Practices in Psychological Science*, *7*(1), 25152459241236149. https://doi.org/10.1177/25152459241236149

Lenth, R. V. (2024). *emmeans: Estimated marginal means, aka least-squares means.* https://doi.org/10.32614/CRAN.package.emmeans

Lüdecke, D., Ben-Shachar, M. S., Patil, I., & Makowski, D. (2020). 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. (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

Lüdecke, D., Makowski, D., Ben-Shachar, M. S., Patil, I., Wiernik, B. M., Bacher, E., & Thériault, R. (2023). *easystats: Streamline model interpretation, visualization, and reporting.* https://easystats.github.io/easystats/

Lüdecke, D., Waggoner, P. D., & Makowski, D. (2019). insight: A unified interface to access information from model objects in r. *Journal of Open Source Software*, *4*(38), 1412. https://doi.org/10.21105/joss.01412

Makowski, D., & Waggoner, P. D. (2023). Where are we going with statistical computing? From mathematical statistics to collaborative data science. In *Mathematics* (No. 8; Vol. 11, p. 1821). MDPI. https://doi.org/10.3390/math11081821

Open Science Collaboration. (2015). Estimating the reproducibility of psychological science. *Science*, *349*, aac4716. https://doi.org/10.1126/science.aac4716