# 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** Institute of Medical Sociology, University Medical Center Hamburg-Eppendorf, Germany **4** Independent Researcher, Ramat Gan, Israel **5** Independent Researcher, Tampa, FL, USA **6** Center for Humans and Machines, Max Planck Institute for Human Development, Berlin, Germany **7** Department of Psychology, Université du Québec à Montréal, Montréal, Québec, Canada

Statement of need

Applied statistics have historically been focused on statistical *tests* (e.g., *t*-tests, correlation tests, ANOVAs), seen as most apt to provide researchers with interpretable answers to the questions they seek. However, these tests are typically based on statistical *models*, which are the true underlying cornerstone of modern data science. The replication crisis ([**ref-camerer2018evaluating**]; [**ref-OSC2015estimating**]) and methodological revolution(s) ([**ref-makowski2023we**]) have underlined the issues with the traditional focus on statistical tests (the effacement of model assumptions, a focus on null-hypothesis testing, non-compatibility with more complex variance structures), and the need to shift 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 ([**ref-easystatspackage**]) has been developed to help researchers "tame, discipline, and harness" the power of statistical models. Existing packages are dedicated to model parameters (e.g., the `parameters` package, [**ref-ludecke2020extracting**]), predictive performance (e.g., the `performance` package, [**ref-ludecke2021performance**]) or effect importance (e.g., the `effectsize` package, [**ref-ben2020effectsize**]).

But there is even more power hidden inside models!

Their fundamental nature—being a statistical link between an outcome y and predictor variables X—enables the generation of predictions for any combination of predictors, observe or unobserved alike. These predictions, in turn, can be used to estimate expected values of the outcome and predictor effects of any levels, as well as contrasting them, making it possible to visualize the model's behaviour in a more meaningful and comprehensive way and answering a wide range of research questions.

The two probably most popular R packages for extracting these quantities of interest from statistical models are `emmeans` ([**ref-russell2024emmeans**]) and `marginaleffects` ([**ref-arel2024interpret**]). These packages are enormously feature rich and (almost) cover all imaginable needs for post-hoc analysis of statistical models. However, these packages are not always easy to use, especially for users who are not familiar with the underlying statistical concepts. The `modelbased` package, built on top of the two aforementioned, aims to unlock this currently underused 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 is designed to be user-friendly, with a focus on simplicity and flexibility.

Key concepts

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 individual observations might take (e.g., *around* the expectation $E[X]$). 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 observation in the original dataset, which is useful to assess the model's goodness-of-fit, or for new data (typically a "data grid"), which is useful for visualization or computing "marginal" estimates (see below).

For convenience, the `modelbased` package includes 4 related functions, that mostly differ in their default arguments for `data` and `predict`:

- `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.

Note: if the defaults are changed, then these functions can become redundant. For instance, `estimate_relation(..., predict = "link")` will be equivalent to `estimate_link(...)`.

Which function to use depends on the model and the research question at hand (visualizing effects, generating predictions, etc.).

Marginal means and effects

The concept of "marginal" in statistical modeling typically refers to the analysis of the effect of one or more "focus" variables while all other variables are held constant at specific values (e.g., their empirical or theoretical average or reference value). This in turn is convenient for understanding the effect of a variable in a complex model, where multiple variables interact with each other.

The `modelbased` package simplifies the extraction of these quantities, providing a clear interface to understand how different predictors interact with outcomes in various scenarios.

- `estimate_means()`: computes **Marginal Means**, i.e., the average predictions for each level of a categorical predictor, averaged across all levels of other predictors.
- `estimate_contrasts()`: computes **Marginal Contrasts**, i.e., the comparison the marginal means of different levels of a factor to assess differences or effects.
- `estimate_slopes()`: computes **Marginal Slopes**, i.e., the change in the response variable for an infinitesimal change in a predictor, averaged across all levels of other

predictors. They are essentially the partial derivatives of the response with respect to each predictor, useful for continuous predictors.

[**TODO: details about types of marginalization**]

Technical details

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

Of the two, `emmeans` ([**ref-russell2024emmeans**]) 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 counter-factual dataset containing all combinations of the categorical predictors in the model and typically 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:

$$L \cdot b$$

$$\sqrt{L \cdot V \cdot L}$$

These grid predictions are sometimes averaged over (averaging being a linear operation itself) to produce "marginal" (in the sense of marginalized-over) predictions—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` ([**ref-arel2024interpret**]) was more recently introduced, and used 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, [**ref-arel2024interpret**]). 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 allows for 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's implemented in `marginaleffects` uses iterative estimation, making it more computationally costly relative to the simple 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, but to obtain these requires some degree of competency in the relevant packages.

Finally, `modelbased` leverages `get_datagrid()` function from the `insight` package ([**ref-ludecke2019insight**]) to intuitively generate an appropriate grid of data points for which predictions or effects or slopes will be estimated. And since these these packages support a wider range of models—including generalized linear models, mixed models, and Bayesian models, this means that `modelbased` also inherits the support for such models.

Examples

Imagine the following linear model in which we predict flower's `Petal.Width` from the interaction between `Petal.Length` and `Species`.

```
library(easystats)

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

parameters::parameters(model)
```

```
#> Parameter                          | Coefficient |  SE  |        95% CI
#> --------------------------------------------------------------------------
#> (Intercept)                        |       -0.05 | 0.21 | [-0.47, 0.38]
#> Petal Length                       |        0.20 | 0.15 | [-0.09, 0.49]
#> Species [versicolor]               |       -0.04 | 0.32 | [-0.66, 0.59]
#> Species [virginica]                |        1.18 | 0.33 | [ 0.52, 1.84]
#> Petal Length × Species [versicolor]|        0.13 | 0.16 | [-0.18, 0.44]
#> Petal Length × Species [virginica] |       -0.04 | 0.15 | [-0.34, 0.26]
#>
#> Parameter                          | t(144) |    p
#> ----------------------------------------------------------
#> (Intercept)                        |  -0.22 | 0.823
#> Petal Length                       |   1.38 | 0.170
#> Species [versicolor]               |  -0.11 | 0.909
#> Species [virginica]                |   3.54 | < .001
#> Petal Length × Species [versicolor]|   0.83 | 0.405
#> Petal Length × Species [virginica] |  -0.27 | 0.789
#>
#> Uncertainty intervals (equal-tailed) and p-values (two-tailed) computed
#>   using a Wald t-distribution approximation.
```

The **parameters** of this model can be a bit hard to interpret and do not offer us all the insights that we could get from this model.

Visualize relationship

We can start by easily visualizing the relationship between our response variable and our predictors.

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

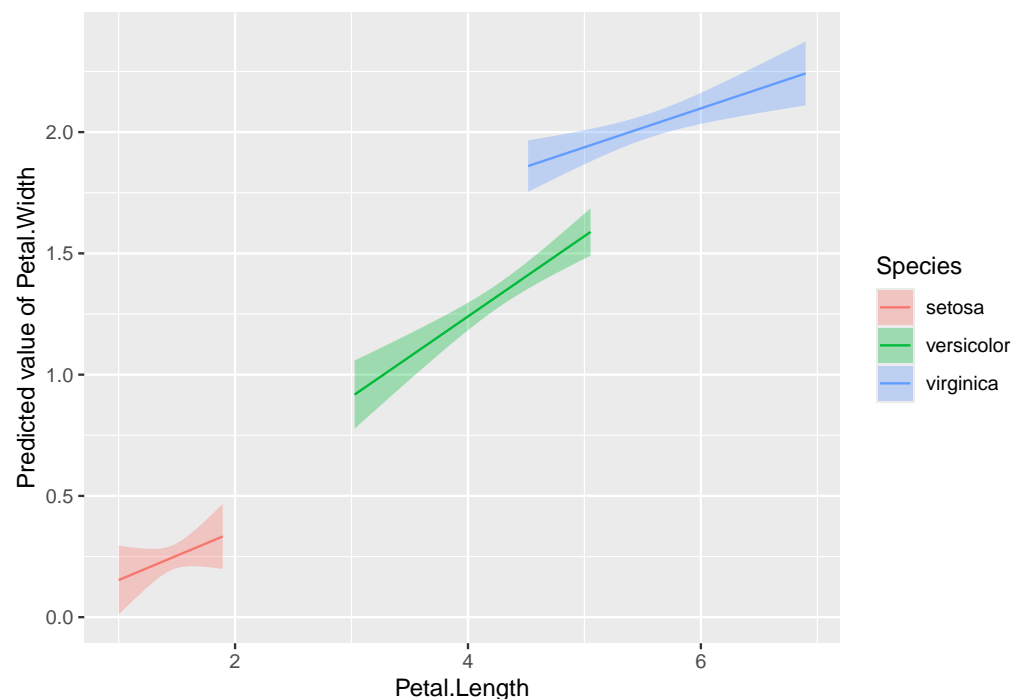But what is the **average value** of `Petal.Width` for each species?

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

But 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 betweeen *setosa* and *versicolor* 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

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 package 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")`.

Contributions

Acknowledgements

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

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

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