Journal of Statistical Software

November 2024, Volume 111, Issue 9. *doi: 10.18637/jss.v111.i09*

How to Interpret Statistical Models Using marginaleffects for R and Python

Vincent Arel-Bundock
Université de Montréal

Noah Greifer
Harvard University

Andrew Heiss
Georgia State University

Abstract

The parameters of a statistical model can sometimes be difficult to interpret substantively, especially when that model includes nonlinear components, interactions, or transformations. Analysts who fit such complex models often seek to transform raw parameter estimates into quantities that are easier for domain experts and stakeholders to understand. This article presents a simple conceptual framework to describe a vast array of such quantities of interest, which are reported under imprecise and inconsistent terminology across disciplines: predictions, marginal predictions, marginal means, marginal effects, conditional effects, slopes, contrasts, risk ratios, etc. We introduce **marginaleffects**, a package for R and Python which offers a simple and powerful interface to compute all of those quantities, and to conduct (non-)linear hypothesis and equivalence tests on them. **marginaleffects** is lightweight; extensible; it works well in combination with other R and Python packages; and it supports over 100 classes of models, including linear, generalized linear, generalized additive, mixed effects, Bayesian, and several machine learning models.

Keywords: marginal effect, marginal mean, slope, prediction, fitted value, contrast, comparison, R, Python.

1. Introduction

The parameters of a statistical model can sometimes be difficult to interpret substantively, especially when that model includes nonlinear components, interactions, or transformations. Analysts who fit such complex models often seek to transform raw parameter estimates into quantities that are easier for domain experts and stakeholders to understand, such as predictions, contrasts, risk differences, ratios, odds ratios, lift, slopes, and so on.

Unfortunately, computing these quantities – along with associated standard errors – can be a tedious and error-prone task. This problem is compounded by the fact that modeling packages in R (R Core Team 2024) and Python (Van Rossum *et al.* 2011) produce objects with

varied structures, which hold different information. This means that end-users often have to write customized code to interpret the estimates obtained by fitting linear, generalized linear (GLM), generalized additive (GAM), Bayesian, mixed effects, and other model types. This can lead to wasted effort, confusion, and mistakes, and it can hinder the implementation of best practices.

In this article, we present a conceptual framework to describe many quantities of interest that researchers can compute to improve the interpretability of their models. We also introduce the **marginaleffects** package for R (Arel-Bundock 2024c) and Python (Arel-Bundock 2024b), which offers a single point of entry to easily interpret the results of over 100 classes of models with a simple, consistent, and powerful user interface.

The rest of this paper proceeds as follows. Section 2 frames our contribution by introducing the statistical setting, surveying alternative software solutions, and giving an overview of our package's functionality. Section 3 introduces a conceptual framework to guide analyses, focusing on five key decisions: quantity, grid, aggregation, uncertainty, and test. The case study in Section 4 illustrates how we can use the **marginaleffects** package to apply these ideas in practice. Section 5 discusses the Python implementation, and Section 6 describes the R package internal design.

Users who want to learn more are encouraged to read the free "Marginal Effects Zoo" online book, which includes over 25 chapters of detailed tutorials, technical material, and case studies (Arel-Bundock 2024a).

2. Motivation

Consider a typical statistical setting, where the analyst chooses a model to meet domain-specific requirements. Perhaps their goal is to capture a salient feature of the data generating process, reach sufficient goodness-of-fit, or satisfy a selection on observables condition for causal inference. The analyst uses an estimator to fit their model, and obtains parameter estimates and standard errors. Even if the model is relatively simple, those parameter estimates may not be straightforward to interpret directly. For instance, after fitting a probit model, analysts are often interested in the change in predicted probability associated with a change in treatment, but this statistical quantity is not given directly by the coefficients of the model. In many cases, the analyst will thus need to transform parameter estimates into quantities that stakeholders, colleagues, and domain experts will readily understand.

Post-estimation procedures are a key step in any data analysis, and software developers have already given us tools to help at this crucial stage. The **marginaleffects** package is newer than some alternatives, and it has the potential to make an important and original contribution to the ecosystem. The package website presents detailed side-by-side comparisons with most of the major post-estimation software packages (Arel-Bundock 2024a), but a few of them are worth mentioning here.

effects is a well-established package, first published on the Comprehensive R Archive Network (CRAN) by Fox (2003) over 20 years ago. **effects** can draw prediction and partial residual plots using **lattice** (Sarkar 2008), with many visual customization options. In contrast, **marginaleffects** draws and customizes its plots using **ggplot2** and related packages (Wickham 2016). **marginaleffects** also supports more than twice the number of models supported by **effects**, and it can compute more quantities of interest and conduct a much broader array of hypothesis tests.

Goal	Function
Predictions	`predictions()`
	`avg_predictions()`
	`plot_predictions()`
Comparisons	`comparisons()`
	`avg_comparisons()`
	`plot_comparisons()`
Slopes	`slopes()`
	`avg_slopes()`
	`plot_slopes()`
Grids	`datagrid()`
Hypothesis and Equivalence	`hypotheses()`
Bayes, Bootstrap, Simulation	`posterior_draws()`
	`inferences()`

Table 1: Main functions of the **marginaleffects** package.

The **margins** and **prediction** packages were designed to compute slopes and predictions, respectively. When creating those packages, Leeper (2024a,b) brought the R ecosystem much closer to what could be achieved at the time using the `margins` function in Stata (StataCorp 2023). The initial development of **marginaleffects** was largely inspired by **margins** and **prediction**, and the user interface still bears some resemblance. However, **marginaleffects** covers a superset of functionality, is actively maintained, supports more model types, and is more computationally efficient.

modelbased (Makowski, Ben-Shachar, Patil, and Lüdecke 2024) and **ggeffects** (Lüdecke 2018) are maintained by affiliates of the **easystats** development team (Lüdecke, Ben-Shachar, Patil, Wiernik, and Makowski 2024). They offer elegant convenience functions to compute and display predictions and contrasts in plots and tables. Many of the calculations that underlie these displays are delegated to **emmeans** or **marginaleffects**; the key contribution of these packages is thus in simplified syntaxes and reporting.

Finally, **emmeans** is arguably the most powerful alternative (Lenth 2024, 2016), and it is the package that we recommend to users who are not satisfied with **marginaleffects**. As the author notes, "much of what the **emmeans** package offers relate[s] more to experimental data than to observational data".[1] Although **emmeans** can accommodate designs with unbalanced treatment groups, the syntax may feel limiting in observational settings. In contrast, **marginaleffects** was designed from the ground up to assist researchers in the analysis of both observational and experimental data (with balanced or unbalanced groups).

Like all the packages mentioned above, the **marginaleffects** package intervenes at the post-estimation stage. If the model, estimator, or data chosen by the analyst are not fit for purpose, none of the quantities described below will be of much interest. However, when the analyst fits an adequate model to good data, **marginaleffects** can be an invaluable tool to make results more interpretable.

Table 1 shows a list of the main functions in the **marginaleffects** package. Using these func-

[1] Quote from the "Basics of estimated marginal means" vignette, version 1.9.0.

tions, analysts can compute a vast array of quantities of interest, which we can group in three categories, associated with three functions: `predictions()`, `comparisons()`, and `slopes()`.

First, the `predictions` family of functions can compute and plot predictions on different scales (aka "fitted values"). It can also aggregate or marginalize predicted values, over a whole dataset or by subgroups (i.e., "marginal means").

Second, the `comparisons` family of functions can compute and plot relationships between two or more predictions. This allows users to report many of the most common quantities of interest in statistical analyses: contrasts, differences, risk ratios, odds ratios, lift, or even arbitrary user-defined functions of two predictions. Many of these quantities are difficult or impossible to compute with the main alternative software packages cited above.

Third, the `slopes` family of functions can compute and plot partial derivatives of the outcome equation. In the econometrics tradition, researchers typically call these slopes "marginal effects," where the term "marginal" refers to a "small change." Analysts can easily evaluate these derivatives at different points in the predictor space, they can aggregate unit-level estimates to produce global summaries, and they conduct hypothesis tests to compare different slopes.[2]

Because computing average predictions, comparisons, and slopes is such common practice, the **marginaleffects** package exports three shortcut functions with prefixes: `avg_predictions()`, `avg_comparisons()`, and `avg_slopes()`. These are simple wrappers around the package's workhorse functions. Instead of returning unit-level estimates by default, they return averages taken over the whole dataset or by subgroup. The `avg_*()` functions do not expose new functionality; their purpose is solely to save keystrokes and improve code readability.[3]

marginaleffects includes a powerful toolkit to conduct linear and nonlinear hypothesis tests on parameter estimates or on functions of those parameters. This includes raw coefficient estimates, user-supplied functions of those estimates, and any of the quantities reported by the **marginaleffects** package: predictions, comparisons, or slopes. This functionality – accessed through the `hypothesis` argument or via the standalone `hypotheses()` function – can be thought of as a more flexible, convenient, and lower dependency alternative to the `deltaMethod` function from the **car** package (Fox and Weisberg 2019). As we show below, this is an extremely powerful feature, as it opens many opportunities for cross-group comparisons and for testing complex hypotheses.

Finally, **marginaleffects** includes a number of utility functions. `datagrid()` is a function to easily create grids of predictor values; `inferences()` can bootstrap estimates and apply other inference methods; and `posterior_draws()` allows a user to extract draws from the posterior distribution of quantities of interest in Bayesian analyses.

marginaleffects has several additional features that we cannot explore in this article, but which are documented on the package website and in help files. These include: equivalence tests, joint hypothesis tests; elasticities; multiple testing correction; alternative uncertainty quantification methods with robust standard errors, bootstrapping, simulation-based inference, and

[2]For example, the `slopes()` function can compute both "average marginal effects" and "marginal effects at the mean." The former is an average of unit-level partial derivatives evaluated at each point in the empirical distribution of the data. The latter is the partial derivative of the outcome equation evaluated with all predictors are held at their means.

[3]Identical results are obtained by calling `avg_predictions(fit)` and `predictions(fit, by = TRUE)`. The latter is more verbose and arguably less clear, as the name of the `by` argument communicates to users its ability to take averages by subset.

conformal prediction; and helpful tools to interpret some non-parametric or machine learning models (ex: the `plot_predictions()` function can be used to draw partial dependence plots).

These functions can greatly simplify the analysis of randomized experiments, but they also allow **marginaleffects** to play a central role in the analysis of observational data with matching, inverse probability weighting, G-computation, multi-level regression with post-stratification (MRP), conjoint experiments, multiple imputation for missing data, and more. These empirical strategies are illustrated by detailed case studies in a free book-length website, published in support of the **marginaleffects** package (Arel-Bundock 2024a).

All of these features are made available in a single software package, in two different languages, and for over 100 classes of models. This is more types of models than is currently supported by any comparable package, and it includes linear, generalized linear (GLM), generalized additive (GAM), mixed-effects, fixed-effects, Bayesian models, and more.

In the next section, we introduce a conceptual framework to describe a vast array of quantities of interest which can be computed at the post-estimation stage of data analysis.

3. Quantity, grid, aggregation, uncertainty, and test

Different statistical models present unique interpretation challenges. This article introduces simple tools which can help overcome these challenges, and allow us to interpret estimates for a wide variety of models in a consistent and transparent manner. At the heart of our conceptual framework are five critical questions that analysts must answer when interpreting statistical results:

1. Quantity: What is the quantity of interest? Do we want to report a prediction or a function of predictions (average, difference, ratio, derivative, etc.)?

2. Grid: What predictor values are we interested in? Do we want to report estimates for the units in our dataset, or for hypothetical or representative individuals?

3. Aggregation: Do we report estimates for every observation in the grid or a global summary?

4. Uncertainty: How do we quantify uncertainty about our estimates?

5. Test: Which (non-)linear hypothesis or equivalence tests do we conduct?

These five elements offer a structured way to make critical decisions for the computation of quantities of interest, thereby streamlining the process of interpreting complex statistical models.

3.1. Quantity

The **marginaleffects** package allows R and Python users to compute and plot three principal quantities of interest: predictions, comparisons, and slopes. Predictions are the foundation on which all other quantities are built: comparisons and slopes are functions of predictions.

Predictions

> *Predictions are the outcomes predicted by a fitted model on a specified scale for a given combination of values of the predictor variables, such as their observed values, their means, or factor levels.*

Depending on the field and context, predictions can also be referred to as "fitted values" or "adjusted predictions."

A model's predictions depend on the values of the predictors. The prediction for an individual with a particular combination of predictor values – or "profile" – is simply the fitted value for the corresponding row in the dataset. We can also make predictions for "synthetic" individuals, by creating and feeding different profiles to our model. For example, we could compute a prediction for a hypothetical person with a given age and education, or the predicted outcome for an individual whose personal characteristics are exactly average on all dimensions. Alternatively, we could compute a model's prediction for a partially synthetic unit where all predictors are held at their observed values, but the treatment is set to 1 instead of its true value. The resulting quantity would be a "counterfactual" prediction in the sense that the treatment indicator would be "counter-to-fact," or different from its actually observed value.

Predictions can be expressed on various scales. For example, in a logistic regression model with a binary outcome, predictions can be computed on the response scale (probabilities) or on the link scale (log-odds). In a zero-inflated count regression model, we can consider the predicted mean of the count component or the predicted probability of the zero component. For an ordinal model, predictions can be probabilities of a given category or expectations over all categories.

Comparisons

> *Comparisons are functions of two or more predictions. Examples of comparisons include contrasts, differences, risk ratios, odds, lift, etc.*

A comparison is often the key quantity of interest in scientific inquiry. When certain assumptions are met (Hernán and Robins 2020; Imbens and Rubin 2015), a comparison can help answer counterfactual queries like: What happens to Y when X increases by one unit?

A useful way to think about such a counterfactual query is that it involves a comparison between two predictions made on different grids. Imagine that we are interested in the "effect" of a change of 1 unit in predictor X on outcome Y. First, we compute a prediction for unit of observation i, when all predictors are held at their observed values. Then, we compute a different prediction for the same individual, when all predictors are held at their observed values, but X is incremented by one unit. In other words, we compute predictions for two profiles (or one-row grids) which differ only in terms of X. Finally, we choose a function to compare our two predictions: difference, ratio, odds, etc.

Many interesting quantities can be expressed as functions of counterfactual predictions. For example, in an education study we may want to compare the predicted probability of an outcome for university $\mathsf{P}(Y = 1 \mid U)$ and high school graduates $\mathsf{P}(Y = 1 \mid H)$. We could do

this by taking a simple difference $\mathsf{P}(Y = 1 \mid U) - \mathsf{P}(Y = 1 \mid H)$, a risk ratio $\frac{\mathsf{P}(Y=1|U)}{\mathsf{P}(Y=1|H)}$, a log odds ratio $\ln\left[\frac{\mathsf{P}(Y=1|U)/\mathsf{P}(Y=0|U)}{\mathsf{P}(Y=1|H)/\mathsf{P}(Y=0|H)}\right]$, or any other function of two predictions.

In the **marginaleffects** framework, comparisons necessarily involve partially (or fully) synthetic units, because the analyst must fix a predictor to take on different values. They may want to look at how the predictions for a specific individual change along with a change in a predictor, or could repeat the operation for all units in a dataset to observe the distribution of counterfactual comparisons. The analyst may also consider a comparison between hypothetical individuals with representative or interesting characteristics along other dimensions than the focal variable.

Slopes

> *A slope is the partial derivative of the regression equation with respect to a predictor of interest.*

If a predictor were to change from its observed or set value by a small amount, the slope is the rate at which the prediction would change. It can be considered as the comparison (described above) between two profiles that differ on a single predictor by a small amount divided by the difference in the predictor values as that difference shrinks to an infinitesimal size. In this way, slopes are linked to comparisons. Slopes are only defined for predictors for which it makes sense to consider a small change in the predictor of interest (i.e., not for categorical predictors). In some disciplines like economics and political science, slopes are known as "marginal effects," where "marginal" refers to the small change in the predictor value.

Slopes can be obtained analytically using the rules of calculus, but this process can be difficult and tedious. **marginaleffects** computes derivatives numerically, which allows it to support complex models with arbitrary transformations.

3.2. Grid

Predictions, comparisons, and slopes are conditional quantities. Except in the simplest linear models, these quantities will vary based on the values of all the predictors in a model. Thus, once an analyst chooses which quantity of interest they are targeting, they must decide where in the predictor space to evaluate that quantity. In other words, they must choose a "grid" of predictor values.

A grid is a collection of one or more profiles, each of which is a vector of predictor values. Those profiles can be observed, partially synthetic, or purely synthetic. For example, we might consider the predicted earnings for a single actual person in our dataset (observed). Alternatively, we could target the same quantity for two individuals in our dataset, with all characteristics held at their observed values, except for education which we artificially increment by 1 year (partially synthetic). Finally, we could make predictions for a hypothetical individual with all predictors held at the sample median or at other representative values (purely synthetic).

3.3. Aggregation

When the grid has many rows, **marginaleffects** will generate many point estimates. These can be unwieldy, making it more difficult to extract clear and meaningful insights from our models. To simplify things, analysts might want to aggregate (marginalize or average) these estimates, either by groups of observations or over the entire dataset.

Average estimates are typically easier to interpret, and they are estimated with greater precision than individual-level quantities. The downside of reporting aggregated estimates is that they can mask interesting variation across the sample. For example, if the effect of a treatment is heterogeneous, the average slope computed across a grid might be close to 0 even if all unit-level slopes are large, because the positive and negative estimates cancel out.

Predictions, comparisons, and slopes can all be aggregated to provide meaningful and intuitive summary quantities.

Average predictions

We can compute an "average prediction" by simply taking the mean of predictions made for various combinations of predictor values. A common use-case is to report the mean of fitted values in the observed sample, or the mean for some subset of the sample (e.g., the average predicted income for high school graduates).

Another approach is to compute average predictions over grids of partially synthetic units. For example, the analyst could compute average "counterfactual" predictions by taking the observed dataset, modifying the treatment variable T to 1 for each observation, computing predictions, and taking their average.

It can also be useful to average predictions over grids of purely synthetic units. For example, the analyst could build a grid of all potential combinations of categorical predictors, make predictions for each of the resulting profiles, and take averages of those predictions across some dimension.

Average comparisons

Recall that in the context of this paper, a "comparison" is a function of two (sets of) predictions. From that perspective, an "average comparison" could refer to two related ideas, depending on the order of operations. We could take two average predictions and compare them (e.g., compute their difference or ratio). Instead, the analyst could take two vectors of predictions, compare them by taking element-wise differences or ratios, and then average the results. Both approaches are supported by **marginaleffects**.

When the comparison between average predictions (the "marginal" comparison) and averages of ("conditional") comparisons are not equal to each other, the contrast used in the comparison is said to be "noncollapsible." Contrasts specified as differences between (average) predictions are collapsible; most other contrasts, including ratios and ratios of odds, are noncollapsible. Averages of noncollapsible comparisons rarely have useful interpretations.

Average slopes

An average slope – or average marginal effect in the econometrics tradition – is simply the average of individual-level estimates of the slope. These are most often computed across a

grid of observed units to measure the average strength of the relationship between a predictor and the outcome.

3.4. Uncertainty

Uncertainty estimates for the quantities described above can be obtained via a number of strategies. By default, **marginaleffects** uses the delta method to compute standard errors, t statistics, p values, s values, and confidence intervals. The delta method is a fast and flexible approach to uncertainty estimation. It often performs well in large samples.[4]

When computing standard errors via the delta method, **marginaleffects** can use classical variance estimates, or robust, clustered, and heteroskedasticity-consistent standard errors supplied by the **sandwich** package (Zeileis, Köll, and Graham 2020). To use these types of standard errors, users can simply specify the `vcov` argument, which accepts variance-covariance matrices or convenient string shortcuts for many common use-cases. This makes it easy for users to report robust standard errors for all the quantities computed by the package.

Users who need an alternative to the delta method can call the `inferences()` function to deploy various forms of bootstrapping, powered by the **boot** (Canty and Ripley 2024), **rsample** (Frick, Chow, Kuhn, Mahoney, Silge, and Wickham 2024), and **fwb** (Greifer 2023) packages. Function `inferences()` can also conduct simulation-based inference using the normal approximation method described in Krinsky and Robb (1986) and popularized by King, Tomz, and Wittenberg (2000).

3.5. Test

marginaleffects can use the delta method to conduct linear or nonlinear hypothesis tests on the coefficients estimated by over 100 classes of models. It can also conduct such tests on arbitrary functions of coefficients, and on any of the quantities estimated by the package: predictions, comparisons, slopes, marginal means, etc. Users can test joint hypotheses for multiple coefficients or quantities simultaneously.

The **marginaleffects** package also provides user-friendly support for conducting equivalence tests on the coefficients of a model, or on any of the quantities computed by the package. The `equivalence` argument implements the Two One-Sided Tests (TOST) procedure, a method designed to establish statistical equivalence or to show that a meaningful difference does not exist (Lakens, Scheel, and Isager 2018). TOST involves conducting two one-sided hypothesis tests: one that tests if the effect is significantly below a lower equivalence bound, and another that tests if the effect is significantly above an upper equivalence bound.

4. Case study

We now present a case study to illustrate a typical workflow with some of the core **marginaleffects** functions listed in Table 1. For simplicity of exposition, the code in this section uses the R version of **marginaleffects**. In Section 5, we show that the Python syntax is extremely

[4]The delta method is a statistical technique used for approximating the variance and standard error of a function of a random variable based on the random variable's own variance and standard error. It is particularly useful when dealing with complex functions of estimators for which the distribution is difficult to derive directly.

similar, so most of the insights from this case study apply to both versions of the package. Moreover, the replication script which accompanies this article holds the code to reproduce all the results and figures in this section using Python.

Our example is inspired by Rothstein and Teorell (2008), who argue that a critical component of good governance in a country is the impartiality of its government institutions, or the degree to which state power is exercised according to written laws rather than personal connections or biases. Many factors can influence the impartiality of governance, including the level of economic inequality in a country (Suzuki and Demircioglu 2021) and the presence of democratic political institutions (Rothstein and Teorell 2008).

To study these relationships, we consider four variables drawn from the Varieties of Democracy project (Coppedge *et al.* 2023; Maerz, Edgell, Hellemeier, and Illchenko 2022):[5]

- `impartial`: A binary indicator equal to 1 if a country's public officials are impartial in the performance of their duties.
- `equal`: A continuous scale from 0 to 100 where higher values indicate that resources like public goods and welfare policies are distributed more equally across society.
- `democracy`: A binary indicator equal to 1 when a country is a democracy.
- `continent`: The continent on which a country is located.

Our dataset includes information on 166 countries. We can read it and display the first few rows as follows:

```
R> library("marginaleffects")
R> library("modelsummary")
R> dat <- read.csv("https://marginaleffects.com/data/impartiality.csv")
R> head(dat)
```

```
  X      country continent impartial equal democracy
1 0       Mexico  Americas         1  31.2 Democracy
2 1     Suriname  Americas         1  68.1 Democracy
3 2       Sweden    Europe         1  93.7 Democracy
4 3  Switzerland    Europe         1  96.6 Democracy
5 4        Ghana    Africa         0  53.3 Democracy
6 5 South Africa    Africa         1  38.6 Democracy
```

To start, we estimate a logistic regression model to predict whether a country's public officials are impartial based on resource equality, regime type, and continent.

[5]We simplified many of these variables for the sake of illustration. For `impartial`, we collapse the five levels of V-Dem's `v2clrspct_ord` into a binary indicator, where levels 2–4 represent impartial governance and levels 0–1 represent not impartial governance; for `equal` we multiply V-Dem's `v2xeg_eqdr` variable by 100; for `democracy`, we collapse the four levels of V-Dem's "Regimes of the world" measure (`v2x_regime`) into a binary indicator, where "Autocracy" includes closed autocracies and electoral autocracies and "Democracy" includes electoral democracies and liberal democracies.

	(1)
Equal	0.046
	(0.013)
Democracy	1.570
	(1.541)
Equal × Democracy	0.038
	(0.039)
Americas	0.866
	(0.874)
Asia	−0.293
	(0.596)
Europe	−0.263
	(1.071)
(Intercept)	−2.692
	(0.664)
Num.Obs.	166
AIC	118.3
BIC	140.1
Log.Lik.	−52.158
F	5.468
RMSE	0.32

Table 2: Logistic regression model with impartiality as the outcome.

```
R> m <- glm(impartial ~ equal * democracy + continent, data = dat,
+    family = binomial)
R> modelsummary(m,
+    title = "Logistic regression model with impartiality as the outcome.")
```

The coefficient estimates from this logistic regression are shown in Table 2. Since this model is not purely linear, and given that it includes a multiplicative interaction, it is not straightforward to interpret the reported coefficients on a scale that will make intuitive sense to most readers. In the rest of this case study, we show how **marginaleffects** functions can extract insights from those estimates by computing predictions, comparisons, and slopes.

4.1. Predictions

Quantity of interest

The first quantity of interest that we consider is the predicted probability of having an impartial public sector, which we compute using the `predictions()` function. `predictions()` mimics and extends the behavior of `stats::predict()`, which is the workhorse function for generating predictions used by many modeling packages in R. Both functions include a `newdata` argument to define the grid of values over which we want to make predictions, and a `type` argument to set the scale of predictions (e.g., `"response"` or `"link"`).

Grid

Predictions are a "conditional" quantity: they depend on the values of the predictors in the

model, which means that each unit of observation will typically have its own predicted value. To generate predictions, we thus need to specify a "grid" of predictors, that is, we need to specify the combinations of predictor values for which we need estimates. For example, we could compute predictions for:

- Every individual in the original dataset.
- One specific individual.
- A hypothetical individual with each predictor held at the sample mean or mode.
- User-specified combinations of predictor values.

By default, `predictions()` returns predictions for each row in the dataset used to fit the model. Our case study data includes 166 observations, so calling `predictions()` with no other argument generates a dataset with 166 rows, with predictions on the probability (response) scale:

```
R> p <- predictions(m)
R> p
```

```
 Estimate Pr(>|z|)    S 2.5 % 97.5 %
    0.914  0.00444  7.8 0.676  0.982
    0.996  0.00108  9.9 0.899  1.000
    0.998  0.00677  7.2 0.857  1.000
    0.999  0.00699  7.2 0.862  1.000
    0.966  < 0.001 10.6 0.807  0.995
--- 156 rows omitted. See ?avg_predictions and ?print.marginaleffects ---
    0.794  0.02311 5.4 0.546  0.925
    0.996  0.00611 7.4 0.830  1.000
    0.999  0.00684 7.2 0.858  1.000
    0.746  0.04504 4.5 0.506  0.894
    0.991  0.00603 7.4 0.792  1.000
Columns: rowid, estimate, p.value, s.value, conf.low, conf.high, impartial,
equal, democracy, continent
```

The default printout for every **marginaleffects** object includes all the usual test statistics, such as p values and confidence intervals. Note that the default null hypothesis is always zero, which may not be appropriate in this case. Users can posit a different null using the `hypothesis` argument.

marginaleffects objects are "tidy" data frames (Wickham 2014). They can be manipulated with standard functions and accessors like `[,]`, `$`, `head()`, `subset()`, or add-on functions like `dplyr::filter()` (Wickham, François, Henry, Müller, and Vaughan 2023):

```
R> p$estimate[1:4]
```

```
[1] 0.9136850 0.9957328 0.9984526 0.9987861
```

```
R> p[2, "p.value"]

[1] 0.001078207

R> p |> subset(estimate == min(estimate))

 Estimate Pr(>|z|)    S CI low CI high
   0.0677   <0.001 14.2 0.0199   0.206

Columns: rowid, estimate, p.value, s.value, conf.low, conf.high, impartial,
equal, democracy, continent
```

Although **marginaleffects** objects are standard data frames, they are printed by a custom method which displays the most important columns and rows, and hides the rest. Printing options are controlled by arguments of the print method which are documented in `?print.marginaleffects`. As can be seen above, the default output includes a printout of the available column names below the estimates. This can be disabled with:

```
R> options(marginaleffects_print_column_names = FALSE)
```

We can use the `newdata` argument to make predictions for a subset of observations, and the `type` argument to change the scale of predictions. Here, we make predictions on the logit (link) scale for the first two observations in the dataset:

```
R> predictions(m, newdata = head(dat, 2), type = "link")

 Estimate Std. Error    z Pr(>|z|)   S 2.5 % 97.5 %
     2.36      0.829 2.84  0.00444 7.8 0.734   3.99
     5.45      1.668 3.27  0.00108 9.9 2.184   8.72
```

The `newdata` argument also supports various shortcuts. For instance, we can calculate a prediction at the mean – or a prediction for an "average country" – by holding every predictor at its mean (numeric) or mode (categorical or binary), thus creating a one-row grid:

```
R> predictions(m, newdata = "mean") |> data.frame()

  rowid  estimate      p.value  s.value  conf.low conf.high   equal democracy
1     1 0.9792756 0.0009489451 10.04139 0.8276753 0.9978535 59.3741 Democracy
  continent impartial
1    Africa         1
```

In a hypothetical African democracy with an equal distribution of resources index of 59.4, the predicted probability of having impartial public officials is 0.979. This combination of predictors does not necessarily reflect the characteristics of any actual country – it is simply a combination of the means and modes of all covariates.

Note that in the code above, we used a pipe to wrap the output in `data.frame()`. This disabled pretty-print and printed all columns.

The `datagrid()` function offers a more powerful way to create prediction grids based on user-specified values. `datagrid()` accepts both vectors and functions that return vectors. For instance, we can create a grid with all unique values of `democracy` and two specific levels of `inequality`, while holding all other variables at their means or modes:

```
R> datagrid(model = m, democracy = unique, equal = c(30, 90))
```

```
  continent democracy equal rowid
1    Africa Democracy    30     1
2    Africa Democracy    90     2
3    Africa Autocracy    30     3
4    Africa Autocracy    90     4
```

We can generate adjusted predictions at these representative values using `predictions()`:

```
R> predictions(m,
+    newdata = datagrid(democracy = unique, equal = c(30, 90)))
```

```
 democracy equal Estimate Pr(>|z|)   S 2.5 % 97.5 %
 Democracy    30    0.801  0.03400 4.9 0.526  0.936
 Democracy    90    0.998  0.00438 7.8 0.881  1.000
 Autocracy    30    0.213  0.00240 8.7 0.104  0.386
 Autocracy    90    0.812  0.05428 4.2 0.493  0.950
```

Aggregation

When using `predictions()` without an explicit grid in the `newdata` argument, we get estimates for each of the 166 rows in the dataset. To make these results more intelligible, it can be useful to collapse unit-level estimates into aggregates. For example, we can calculate the overall average prediction with `avg_predictions()`, which takes the average of all the predicted values in the full dataset.

```
R> avg_predictions(m, type = "response")
```

```
 Estimate Std. Error  z Pr(>|z|)     S 2.5 % 97.5 %
    0.693     0.0247 28   <0.001 572.2 0.644  0.741
```

This is equivalent to computing predictions for every row of the original dataset, and then taking the mean of these predictions:[6]

```
R> mean(predict(m, newdata = dat, type = "response"))
```

```
[1] 0.6927711
```

[6]When the model is a GLM and the `type` argument is left unspecified, the `predictions()` function makes predictions on the link scale, and then backtransforms the results using the inverse link function. Calling `avg_predictions(m)` is thus equivalent to `plogis(mean(predict(m, type = "link")))`.

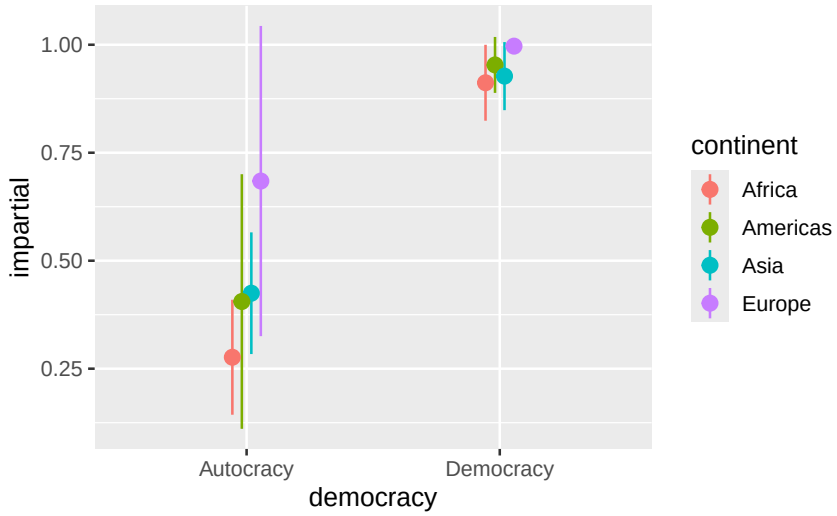

Figure 1: Average predicted probability of impartiality by regime type and continent.

This result tells us that the average predicted probability of having an impartial public sector is 0.693.

We can also calculate the average predictions across subgroups in the data with the `by` argument. This command tells us the average predicted probability of having an impartial public sector among actually-observed democracies and autocracies:

```
R> avg_predictions(m, by = "democracy", type = "response")
```

```
 democracy Estimate Std. Error     z Pr(>|z|)    S 2.5 % 97.5 %
 Democracy    0.956     0.0200 47.78   <0.001  Inf 0.916  0.995
 Autocracy    0.382     0.0485  7.87   <0.001 48.0 0.287  0.477
```

Note that the `by` argument allows much more complicated aggregation schemes and functions. See the **marginaleffects** documentation for details.

Visualization

It is very easy to plot average predictions using the parallel syntax of the `plot_predictions()` function:

```
R> plot_predictions(m, by = c("democracy", "continent"), type = "response")
```

The output of `plot_predictions()` in Figure 1 is a standard **ggplot2** object, which means that we can customize its appearance using **ggplot2** functions and related theming packages. We can also use the `condition` argument to draw estimates over a grid of values which cover the range of a numeric variable. `condition` creates a grid with equally spaced values of the first named variable, and unique values of subsequent elements (see `?plot_predictions` for details). The result of this code is shown in Figure 2.

```
R> library("ggplot2")
R> theme_set(theme_bw(base_family = "serif"))
```

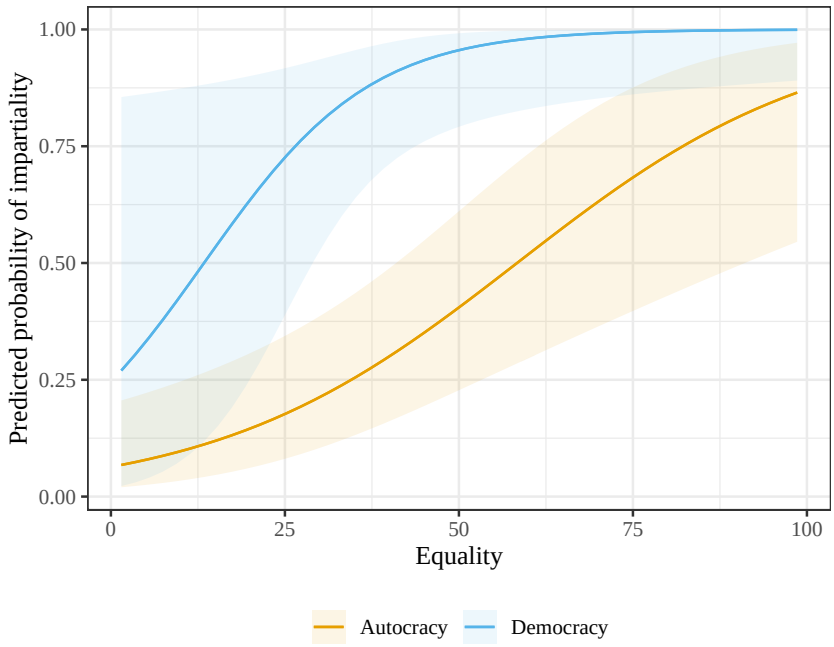

Figure 2: Predicted probability of impartiality by levels of equality and democracy.

```
R> okabeito <- c("#E69F00", "#56B4E9")
R> plot_predictions(m, condition = c("equal", "democracy")) +
+    labs(color = NULL, fill = NULL, x = "Equality",
+      y = "Predicted probability of impartiality") +
+    scale_colour_manual(values = okabeito) +
+    scale_fill_manual(values = okabeito) +
+    theme(legend.position = "bottom")
```

Uncertainty

The procedure used to quantify uncertainty about our estimates can often be crucial. By default, all uncertainty estimates are computed using the delta method, and derivatives are computed numerically using simple finite differences. The package also supports the Richardson method and allows users to set the step size and other parameters used to obtain the numerical derivative.

All **marginaleffects** functions have a `vcov` argument which can be used to report alternative standard errors. `vcov` accepts variance-covariance matrices, functions that return variance-covariance matrices, and shortcuts to select commonly used standard errors (e.g., Hubert-White or cluster-robust). Behind the scenes, **marginaleffects** delegates parts of the computation to the **sandwich** package (Zeileis *et al.* 2020).

To illustrate, we use the `vcov`, `conf_level`, `hypothesis` arguments to test the null hypothesis that average predicted probabilities are equal to 0.4, using heteroskedasticity-consistent (type 3) standard errors and 99% confidence intervals:

```
R> avg_predictions(m, vcov = "HC3", conf_level = 0.99, hypothesis = 0.4,
+    by = "democracy", type = "response")
```

```
 democracy Estimate Std. Error      z Pr(>|z|)     S 0.5 % 99.5 %
 Democracy    0.956     0.0212 26.150   <0.001 498.3 0.901  1.010
 Autocracy    0.382     0.0523 -0.352    0.725   0.5 0.247  0.516
```

Or we can cluster standard errors by continent by using a one-sided formula:

```
R> avg_predictions(m, by = "continent", vcov = ~continent)
```

```
 continent Estimate Std. Error        z Pr(>|z|)   S 2.5 % 97.5 %
  Americas    0.852   6.66e-09 1.28e+08   <0.001 Inf 0.852  0.852
  Europe      0.949   4.42e-08 2.15e+07   <0.001 Inf 0.949  0.949
  Africa      0.528   2.25e-08 2.35e+07   <0.001 Inf 0.528  0.528
  Asia        0.574   1.07e-08 5.35e+07   <0.001 Inf 0.574  0.574
```

Instead of the delta method, analysts can also use the bootstrap and simulation-based inference using the **boot** (Canty and Ripley 2024), **rsample** (Frick *et al.* 2024), or **fwb** (Greifer 2023) packages. For this, we pipe our initial call into the `inferences()` function:

```
R> set.seed(1024)
R> avg_predictions(m, by = "democracy", type = "response") |>
+    inferences(method = "simulation")
```

```
 democracy Estimate 2.5 % 97.5 %
 Democracy    0.956 0.859  0.981
 Autocracy    0.382 0.300  0.482
```

Test

We can use the `hypotheses()` function to conduct linear or nonlinear hypothesis test on a model's coefficients or on arbitrary functions of coefficients. For instance, we can test the null hypothesis that the coefficient for Asia is equal to the coefficient for the Americas:

```
R> coef(m)[c("continentAsia", "continentAmericas")]
```

```
    continentAsia continentAmericas
       -0.2926119         0.8655884
```

```
R> hypotheses(m, hypothesis = "continentAsia = continentAmericas")
```

```
  Estimate Std. Error     z Pr(>|z|)   S 2.5 % 97.5 %
    -1.16       0.94 -1.23    0.218 2.2    -3  0.685

Term: continentAsia = continentAmericas
```

All the core **marginaleffects** functions also have a `hypothesis` argument which allows us to conduct linear and nonlinear hypothesis tests on any of the estimates produced by the package. To illustrate, start by estimating the predicted probabilities of having impartial institutions in autocracies and democracies:[7]

[7]We use the `type` argument to avoid the automatic back-transformation which is normally applied to GLM models. See the "Standard Errors" chapter of the online book for details (Arel-Bundock 2024a).

```
R> avg_predictions(m, by = "democracy", type = "response")

 democracy Estimate Std. Error     z Pr(>|z|)    S 2.5 % 97.5 %
 Democracy    0.956     0.0200 47.78   <0.001  Inf 0.916  0.995
 Autocracy    0.382     0.0485  7.87   <0.001 48.0 0.287  0.477
```

Now, we add the `hypothesis` argument to compute pairwise differences between the estimates in each row:

```
R> avg_predictions(m, by = "democracy", type = "response",
+    hypothesis = "revpairwise")

 Estimate Std. Error     z Pr(>|z|)    S  2.5 % 97.5 %
   -0.574     0.0525 -10.9   <0.001 90.1 -0.677 -0.471

Term: Autocracy - Democracy
```

We can also formulate null hypotheses in terms of specific rows of a function's output: `b1`, `b2`, `b3`, etc. Consider this statement:

> *On average, the predicted outcome is two times larger for democracies (row 1) than for autocracies (row 2).*

We can test it by modifying the `hypothesis` argument:

```
R> avg_predictions(m, by = "democracy", type = "response",
+    hypothesis = "b1 = b2 * 2")

 Estimate Std. Error     z Pr(>|z|)     S 2.5 % 97.5 %
    -1.53     0.0629 -24.3   <0.001 432.0 -1.65  -1.41

Term: b2=b1*2
```

This estimate is equivalent to $0.9556 - 2 \cdot 0.3816 = 0.1924$, with a test statistic that is very close to conventional thresholds of statistical significance. Using this string-equation syntax, we can test a wide variety of null hypotheses, using nonlinear functions and including more than two estimates.

Now, let's say that for scientific reasons, we consider that estimates between -0.2 and 0.2 are functionally equivalent to zero. We can use the TOST equivalence test to see if the quantity of interest is likely to fall outside that equivalence band:

```
R> avg_predictions(m, by = "democracy", type = "response",
+    hypothesis = "b1 = b2 * 2", equivalence = c(-0.2, 0.2))

 Estimate Std. Error     z Pr(>|z|)     S 2.5 % 97.5 % p (NonSup) p (NonInf)
    -1.53     0.0629 -24.3   <0.001 432.0 -1.65  -1.41     <0.001          1
 p (Equiv)
         1

Term: b2=b1*2
```

The output above shows the p value associated with the non-inferiority, non-superiority, and equivalence tests. We can reject the null hypothesis that the estimate is inferior to -0.2, but we cannot reject the null hypothesis that it lies outside the $[-0.2, 0.2]$ interval.

4.2. Comparisons

The `comparisons()` function allows us to compare two sets of predictions made with different predictor values. For example, we can ask the following research question:

> *How does the predicted probability of having impartial government institutions change if a country is an autocracy or a democracy?*

Our quantity of interest for this question is the difference between predictions when the `democracy` variable takes on different values. We estimate this quantity using function `comparisons()` and by specifying the `variables` argument. Again, it is important to emphasize that risk differences are conditional quantities, which can vary with the values of all predictors in the model. By default, we thus obtain one estimate of the comparison (risk difference) for each unit of observation:

```
R> comparisons(m, variables = "democracy")

 Estimate Std. Error    z Pr(>|z|)    S    2.5 % 97.5 %
    0.509     0.1635 3.12  0.00184  9.1  0.18883  0.830
    0.207     0.1519 1.37  0.17225  2.5 -0.09038  0.505
    0.201     0.1426 1.41  0.15759  2.7 -0.07796  0.481
    0.181     0.1327 1.36  0.17263  2.5 -0.07914  0.441
    0.524     0.1056 4.96  < 0.001 20.5  0.31703  0.731
--- 156 rows omitted. See ?avg_comparisons and ?print.marginaleffects ---
    0.205     0.0968 2.11  0.03445 4.9  0.01499  0.395
    0.293     0.1786 1.64  0.10149 3.3 -0.05757  0.643
    0.195     0.1395 1.40  0.16226 2.6 -0.07847  0.468
    0.251     0.1015 2.47  0.01342 6.2  0.05204  0.450
    0.398     0.2043 1.95  0.05152 4.3 -0.00262  0.798
Term: democracy
Comparison: Democracy - Autocracy
```

We can aggregate these results to compute average risk differences:

```
R> avg_comparisons(m)

       Term                             Contrast Estimate Std. Error      z
 continent mean(Americas) - mean(Africa)          0.08627    0.08608  1.002
 continent mean(Asia) - mean(Africa)             -0.02998    0.06012 -0.499
 continent mean(Europe) - mean(Africa)           -0.02695    0.10870 -0.248
 democracy mean(Democracy) - mean(Autocracy)      0.38299    0.07443  5.146
 equal     mean(+1)                               0.00538    0.00114  4.709
```

```
 Pr(>|z|)    S    2.5 %  97.5 %
    0.316  1.7 -0.08245 0.25500
    0.618  0.7 -0.14782 0.08786
    0.804  0.3 -0.24000 0.18609
   <0.001 21.8  0.23711 0.52887
   <0.001 18.6  0.00314 0.00762
```

On average, autocracies have a 38.3 percentage point lower predicted probability of having an impartial public sector than democracies. This result can be replicated using these base R commands:

```
R> dat_lo <- transform(dat, democracy = "Autocracy")
R> dat_hi <- transform(dat, democracy = "Democracy")
R> pred_lo <- predict(m, newdata = dat_lo, type = "response")
R> pred_hi <- predict(m, newdata = dat_hi, type = "response")
R> mean(pred_hi - pred_lo)

[1] 0.3829926
```

For numeric variables, `comparisons()` reports the risk difference associated with a one unit increase in the variable. We can use the `variables` argument to specify a different increment, a on e standard deviation change, movement across the interquartile range, or the risk difference between two specific values of the predictors:

```
R> avg_comparisons(m, variables = list("equal" = 4))
R> avg_comparisons(m, variables = list("equal" = "sd"))
R> avg_comparisons(m, variables = list("equal" = "iqr"))
R> avg_comparisons(m, variables = list("equal" = c(30, 90)))
```

We can move beyond differences in predictions by specifying the `comparison` or `transform` arguments. For instance, we can compute average risk ratios as follows:

```
R> avg_comparisons(m, variables = "democracy", comparison = "ratio")

 Estimate Std. Error    z Pr(>|z|)    S 2.5 % 97.5 %
     1.74      0.221 7.91   <0.001 48.5  1.31   2.18

Term: democracy
Comparison: mean(Democracy) / mean(Autocracy)
```

We can compute the log-odds ratio and then exponentiate the result:

```
R> avg_comparisons(m, comparison = "lnor", transform = exp)

      Term                              Contrast Estimate Pr(>|z|)    S 2.5 %
 continent ln(odds(Americas) / odds(Africa))        1.571    0.354  1.5 0.604
 continent ln(odds(Asia) / odds(Africa))            0.871    0.618  0.7 0.505
```

```
 continent ln(odds(Europe) / odds(Africa))          0.883    0.802  0.3 0.334
 democracy ln(odds(Democracy) / odds(Autocracy))    8.239   <0.001 14.1 2.945
 equal     +1                                       1.026   <0.001 18.4 1.015
 97.5 %
   4.08
   1.50
   2.33
  23.04
   1.04
```

The `comparison` argument also accepts user-defined functions, so we can create fully customized comparisons. For example, we could compute the ratio between average predictions as:

```
R> avg_comparisons(m, variables = "equal",
+    comparison = \(hi, lo) mean(hi) / mean(lo))

 Estimate Std. Error   z Pr(>|z|)   S 2.5 % 97.5 %
     1.01     0.0017 593   <0.001 Inf     1   1.01

Term: equal
Comparison: +1
```

Like the other functions of the package, `comparisons()` supports the `hypothesis` argument, which allows much flexibility in testing questions like:

> *Does moving from low to high resource equality have a larger effect on the probability of having an impartial public sector for democracies than for autocracies?*

First, we see that a change from 30 to 90 in `equal` is associated with an increase of 0.196 in democracies and 0.592 for autocracies:

```
R> cmp <- avg_comparisons(m, by = "democracy",
+    variables = list(equal = c(30, 90)))
R> cmp

 democracy Estimate Std. Error    z Pr(>|z|)    S   2.5 % 97.5 %
 Autocracy    0.592      0.119 4.96   <0.001 20.4  0.3577  0.825
 Democracy    0.196      0.114 1.73    0.084  3.6 -0.0264  0.419

Term: equal
Comparison: mean(90) - mean(30)
```

We can visualize those risk differences with the `plot_comparisons()` function, adding the `labs()` function from **ggplot2** to control the axis labels (Figure 3).

```
R> plot_comparisons(m, by = "democracy",
+    variables = list(equal = c(30, 90))) +
+    labs(x = NULL, y = "Risk Difference (Impartiality)")
```

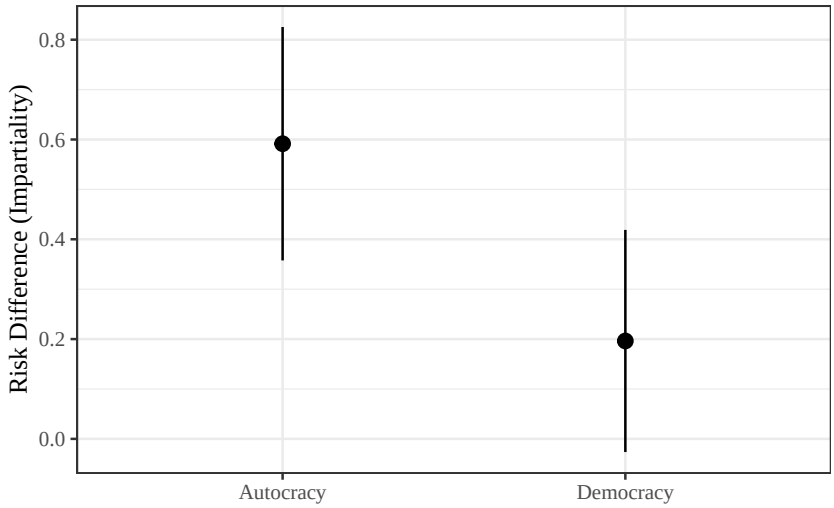

Figure 3: Effect of a change from 30 to 90 in resource equality on the predicted probability of having impartial public institutions.

Then, we use the `hypothesis` argument to compare the two estimates:

```
R> cmp <- avg_comparisons(m, by = "democracy",
+    variables = list(equal = c(30, 90)), hypothesis = "pairwise")
R> cmp

 Estimate Std. Error    z Pr(>|z|)   S 2.5 % 97.5 %
    0.395      0.142 2.79  0.00523 7.6 0.118  0.673

Term: Autocracy - Democracy
```

The p value is small, so we can reject the null hypothesis that a change in `equal` is associated with the same change in predicted outcome for democracies and autocracies.

4.3. Slopes and elasticities

Thus far, we have looked at the relationship between impartial public administration and resource equality by focusing on discrete changes in the predictors (e.g., from 30 to 90 on the `equal` variable). In some contexts, researchers are more interested in the slope of a relationship, that is, in the partial derivative of the outcome equation. Alternatively, analysts may want to estimate an elasticity or semi-elasticity. The `slopes()`, `avg_slopes()`, and `plot_slopes()` functions behave in the same way as their `predictions()` and `comparisons()` counterparts; they allow analysts to answer questions like:

> *How does the probability of impartial governance change as resource equality increases by a very small amount?*

More concretely, one may wish to estimate the slope of the prediction function at the two points where the tangents touch the curve in Figure 4. To achieve this, we call the `slopes()` function and specify the points we are interested in using the `newdata` argument:

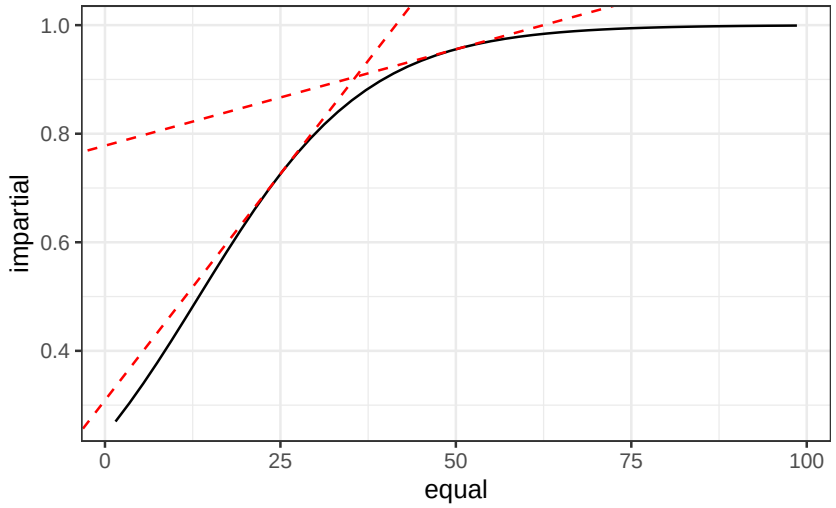

Figure 4: Tangents to the prediction function at 25 and 50.

```
R> slopes(m, newdata = datagrid(equal = c(25, 50)), variables = "equal")

  Term equal Estimate Std. Error    z Pr(>|z|)   S     2.5 %  97.5 %
 equal    25  0.01668    0.01142 1.46   0.1443 2.8 -0.005713 0.03907
 equal    50  0.00355    0.00209 1.70   0.0896 3.5 -0.000549 0.00766
```

As with the other functions, we can also compute other quantities such as the "mean slope" (aka "average marginal effect"), "slope at the mean" (aka "marginal effects at the mean"), "slope at the median by group", etc.

Average marginal effect:

```
R> avg_slopes(m, variables = "equal")

 Estimate Std. Error    z Pr(>|z|)    S   2.5 %  97.5 %
  0.00541    0.00116 4.66   <0.001 18.3 0.00314 0.00769

Term: equal
Comparison: mean(dY/dX)
```

Marginal effect at the mean:

```
R> slopes(m, variables = "equal", newdata = "mean")

 Estimate Std. Error    z Pr(>|z|)   S     2.5 %  97.5 %
   0.0017    0.00132 1.29    0.196 2.3 -0.000879 0.00428

Term: equal
```

Marginal effects at the median by group:

```
R> avg_slopes(m, variables = "equal", newdata = "median", by = "democracy")

  democracy Estimate Std. Error     z Pr(>|z|)    S     2.5 % 97.5 %
 Autocracy 0.011080   0.002516 4.404   <0.001 16.5  0.006149 0.0160
 Democracy 0.000912   0.000913 0.998    0.318  1.7 -0.000878 0.0027

Term: equal
Comparison: mean(dY/dX)
```

To compute elasticities or semi-elasticities, we use the `slope` argument:

```
R> avg_slopes(m, variables = "equal", slope = "eyex")

 Estimate Std. Error    z Pr(>|z|)    S 2.5 % 97.5 %
    0.546      0.134 4.08   <0.001 14.4 0.284  0.809

Term: equal
Comparison: mean(eY/eX)
```

Of course, most of the arguments introduced above are also available in `slopes()` (ex: `hypothesis`, `by`, `vcov`), and we can plot key quantities of interest using the dedicated function `plot_slopes()`, which uses a very similar API to the other plotting functions of this package.

5. Python

The **marginaleffects** package for Python is more recent than the R version and it is still maturing. Nevertheless, most of the functionality is already implemented, and the Python and R syntaxes mirror each other almost exactly. The package currently supports models estimated using the formula API of the **statsmodels** package (Seabold and Perktold 2010). This means that we can cover models like probit, logit, OLS, and quantile regression among others.

To begin, we use **polars** (Vink 2024) to read a CSV file into a data frame (**pandas**, The **pandas** Development Team 2024, is also supported). Then we estimate a probit model using **statsmodels**:

```
>>> import pandas as pd
>>> import statsmodels.formula.api as smf
>>> from marginaleffects import avg_predictions, slopes
>>> dat = pd.read_csv("https://marginaleffects.com/data/impartiality.csv")
>>> mod = smf.logit("impartial ~ equal * democracy + continent",
...   data = dat).fit()
```

Finally, we use standard **marginaleffects** commands to compute average predictions by variable `continent`, and the slope at the mean with respect to `equal`:

```
>>> p = avg_predictions(mod, by = "continent")
>>> p
```

```
shape: (4, 8)
```

continent --- str	Estimate --- str	Std.Error --- str	z --- str	P(>|z|) --- str	S --- str	2.5% --- str	97.5% --- str
Africa	0.528	0.0492	10.7	0	inf	0.432	0.625
Americas	0.852	0.0493	17.3	0	inf	0.755	0.948
Asia	0.574	0.0559	10.3	0	inf	0.465	0.684
Europe	0.949	0.0301	31.5	0	inf	0.89	1.01

```
Columns: continent, estimate, std_error, statistic, p_value, s_value,
conf_low, conf_high
```

```
>>> s = slopes(mod, variables = "equal", newdata = "mean")
>>> s
```

```
shape: (1, 9)
```

Term --- str	Contrast --- str	Estimate --- str	Std.Error --- str	…	P(>|z|) --- str	S --- str	2.5% --- str	97.5% --- str
equal	dY/dX	0.0017	0.00128	…	0.183	2.45	-0.000803	0.00421

```
Columns: term, contrast, estimate, std_error, statistic, p_value, s_value,
conf_low, conf_high
```

6. Package internals and extensions

marginaleffects is a standards-compliant package in the sense that it produces results that conform to modern norms and best practices for R development. Its functions return data frames in "tidy" long format, which facilitates interoperability with packages like **ggplot2** to draw plots, or **modelsummary** to build tables (Arel-Bundock 2022). The data frames produced by **marginaleffects** use the simple naming convention spelled out by the **broom** package, with standard column names such as `estimate`, `std.error`, and `conf.low` (Robinson, Hayes, and Couch 2024).

marginaleffects depends on a few R packages for core operations. **data.table** is used for efficient data frame manipulation (Barrett, Dowle, Srinivasan, Gorecki, Chirico, Hocking, and Schwendinger 2024). **checkmate** is used to inspect arguments and return informative error

messages when user input does not conform to expectations (Lang 2017). **insight** is used to extract a variety of information from model objects (Lüdecke, Waggoner, and Makowski 2019). **generics** is imported to enable `tidy()` and `glance()` methods (Wickham, Kuhn, and Vaughan 2022), and **rlang** is used for more robust handling of some scoping issues (Henry and Wickham 2024).

All of these packages are themselves dependency-free, which helps keep the total number of recursive dependencies for **marginaleffects** relatively low. In turn, this makes it less costly for other developers to import **marginaleffects** to handle some computations, such as in the **etwfe** and **clarify** packages (McDermott 2024; Greifer, Worthington, Iacus, and King 2024).

In addition to its core dependencies, **marginaleffects**'s functionality can be enhanced by several optional packages that users can, but do not have to install: **ggplot2**, **collapse** (Krantz 2024), **numDeriv** (Gilbert and Varadhan 2019). The test suite – which includes thousands of expectations and numerical checks against alternative software like **emmeans** or Stata – can be run using the **tinytest** package by Van der Loo (2021). **sandwich** can be used to compute robust standard errors (Zeileis *et al.* 2020). **ggplot2** is used by the `plot_predictions()`, `plot_comparisons()`, and `plot_slopes()` functions to plot quantities of interest.

To compute predictions, comparisons, slopes, and marginal means, **marginaleffects** defines four S3 methods for each of the supported model classes. First, `get_coef()` accepts a model object and returns a named numeric vector of coefficients.

Second, `set_coef()` accepts a model object and a named vector of coefficients, and returns a new model object in which the original coefficients have been replaced by the user-supplied values. For example, for a `model` object produced by the `lm()` function, `set_coef()` will modify the values hosted internally in `model[["coefficients"]]`. This method is used in the process of computing standard errors with the delta method when we need to compute numerical derivatives with respect to the coefficients e.g., estimating the effect of a small change in one of the coefficients on the predicted outcome.

Third, `get_vcov()` accepts a model object and returns a square variance-covariance matrix. If the **sandwich** package is installed and supports the model class, then `get_vcov()` also accepts a `vcov` argument which controls the type of uncertainty estimates to compute (heteroskedasticity-consistent, cluster-robust, etc.).

Finally, `get_predict()` accepts a model object and a data frame, and returns a data frame with a column of unique sequential row identifiers (`rowid`) and a column of predicted outcomes (`estimate`). For models with a categorical or multivariate outcome, the output of `get_predict()` also includes a `group` column.

In many cases, the default versions of the methods described above will already be able to handle a new class of models. But in the worst of cases, a developer can define four simple S3 methods. Thus, it is normally very easy to add support for new model types, a process that often requires less than 10 lines of code. Thanks to this extensibility, the package developers have already added support for over 100 model classes. The **marginaleffects** website includes detailed examples on how to add support for new models, and even illustrates how one can post-process the results of statistical models estimated in Python using **reticulate** and **NumPyro** (Ushey, Allaire, and Tang 2024; Bingham *et al.* 2019), see Arel-Bundock (2024a).

The Python package is structured in roughly the same way as the R package, and those who have read the description above should be able to navigate the code base with relative ease. The package requires a version of Python greater or equal to 3.9 and depends on the following

packages: **statsmodels** (Seabold and Perktold 2010), **numpy** (Harris *et al.* 2020), **pyarrow** (Apache Arrow Developers 2024), **polars**, and **scipy** (Virtanen *et al.* 2020).

7. Conclusion

Imbuing parameter estimates with substantive meaning is one of the most important and difficult tasks of the applied statistician. Post-estimation processing is made even more arduous by the facts that the terminology used to describe different quantities of interest varies widely across fields and disciplines and that different software packages produce objects with different structures and information.

In this article, we presented a conceptual framework that can be used to think about and describe various quantities of interest in a consistent and transparent way. The five key characteristics of those quantities of interest are: (1) quantity, (2) grid, (3) aggregation, (4) uncertainty, and (5) test.

To operationalize this framework in practice, we introduced the **marginaleffects** package for R and Python. This new addition to these ecosystems has the potential to facilitate and improve statistical practice in many fields. Indeed, our case study showed that **marginaleffects** is a powerful tool: it can compute predictions, comparisons (contrasts, risk ratios, etc.), and slopes and conduct hypothesis tests for over 100 different classes of models. All the functions in the package share a simple, unified, and well-documented interface, which makes them easy to use. **marginaleffects** relies on relatively few dependencies, is easy to extend, and produces "tidy" results. These qualities facilitate interoperability with other R and Python packages and should ensure long-term maintainability. Finally, the package is accompanied by an extensive test suite, with many checks of numerical accuracy against alternative software.

Readers who want to learn more about **marginaleffects** are encouraged to read the free online book, which includes over 25 chapters of detailed tutorials, technical material, and case studies (Arel-Bundock 2024a).

Acknowledgments

We thank Arthur Albuquerque, Marcio Augusto, Etienne Bacher, Tyson Barrett, Mattan S. Ben-Shachar, Kyle F. Butts, Maël Coursonnais, Sam Crawley, Brett Gall, Nadjim Fréchet, Stefan Hansen, Karl Ove Hufthammer, Philippe Joly, Adrien Lamarche, Daniel Lüdecke, Grant McDermott, Marco Mendoza Aviña, A. Jordan Nafa, Resul Umit, Brenton Wiernik, Aaron Zipp.

References

Apache Arrow Developers (2024). ***pyarrow**: Python Library for Apache Arrow*. Python package version 17.0.0, URL https://pypi.org/project/pyarrow/.

Arel-Bundock V (2022). "**modelsummary**: Data and Model Summaries in R." *Journal of Statistical Software*, **103**(1), 1–23. doi:10.18637/jss.v103.i01.

Arel-Bundock V (2024a). "The Marginal Effects Zoo: How to Interpret Statistical Models in R and Python." URL https://marginaleffects.com/.

Arel-Bundock V (2024b). ***marginaleffects***. Python package version 0.0.14, URL https://pypi.org/project/marginaleffects/.

Arel-Bundock V (2024c). ***marginaleffects**: Predictions, Comparisons, Slopes, Marginal Means, and Hypothesis Tests*. doi:10.32614/CRAN.package.marginaleffects. R package version 0.24.0.

Barrett T, Dowle M, Srinivasan A, Gorecki J, Chirico M, Hocking T, Schwendinger B (2024). ***data.table**: Extension of* `data.frame`. doi:10.32614/CRAN.package.data.table. R package version 1.16.2.

Bingham E, Chen JP, Jankowiak M, Obermeyer F, Pradhan N, Karaletsos T, Singh R, Szerlip PA, Horsfall P, Goodman ND (2019). "**Pyro**: Deep Universal Probabilistic Programming." *Journal of Machine Learning Research*, **20**(28), 1–6.

Canty A, Ripley BD (2024). ***boot**: Bootstrap R (S-PLUS) Functions*. doi:10.32614/CRAN.package.boot. R package version 1.3-31.

Coppedge M, Gerring J, Knutsen CH, Lindberg SI, Teorell J, Altman D, Bernhard M, Cornell A, Fish MS, Gastaldi L, Gjerløw H, Glynn A, God AG, Grahn S, Hicken A, Kinzelbach K, Krusell J, Marquardt KL, McMann K, Mechkova V, Medzihorsky J, Natsika N, Neundorf A, Paxton P, Pemstein D, Pernes J, Rydén O, von Römer J, Seim B, Sigman R, Skaaning SE, Staton J, Sundström A, Tzelgov E, ting Wang Y, Wig T, Wilson S, Ziblatt D (2023). "V-Dem [Country-Year/Country-Date] Dataset V13." doi:10.23696/vdemds23. Varieties of Democracy (V-Dem) Project.

Fox J (2003). "Effect Displays in R for Generalised Linear Models." *Journal of Statistical Software*, **8**(15), 1–27. doi:10.18637/jss.v008.i15.

Fox J, Weisberg S (2019). *An R Companion to Applied Regression*. 3rd edition. Sage Publications, Thousand Oaks. URL https://socialsciences.mcmaster.ca/jfox/Books/Companion/.

Frick H, Chow F, Kuhn M, Mahoney M, Silge J, Wickham H (2024). ***rsample**: General Resampling Infrastructure*. doi:10.32614/CRAN.package.rsample. R package version 1.2.1.

Gilbert P, Varadhan R (2019). ***numDeriv**: Accurate Numerical Derivatives*. doi:10.32614/CRAN.package.numderiv. R package version 2016.8-1.1.

Greifer N (2023). ***fwb**: Fractional Weighted Bootstrap*. doi:10.32614/CRAN.package.fwb. R package version 0.2.0.

Greifer N, Worthington S, Iacus S, King G (2024). ***clarify**: Simulation-Based Inference for Regression Models*. doi:10.32614/CRAN.package.clarify. R package version 0.2.1.

Harris CR, Millman KJ, Van der Walt SJ, Gommers R, Virtanen P, Cournapeau D, Wieser E, Taylor J, Berg S, Smith NJ, Kern R, Picus M, Hoyer S, Van Kerkwijk MH, Brett M, Haldane A, del Río JF, Wiebe M, Peterson P, Gérard-Marchant P, Sheppard K, Reddy

T, Weckesser W, Abbasi H, Gohlke C, Oliphant TE (2020). "Array Programming with **NumPy**." *Nature*, **585**(7825), 357–362. doi:10.1038/s41586-020-2649-2.

Henry L, Wickham H (2024). ***rlang**: Functions for Base Types and Core R and **tidyverse** Features*. doi:10.32614/CRAN.package.rlang. R package version 1.1.4.

Hernán MA, Robins JM (2020). *Causal Inference: What If*. Chapman & Hall/CRC, Boca Raton.

Imbens GW, Rubin DB (2015). *Causal Inference in Statistics, Social, and Biomedical Sciences*. Cambridge University Press.

King G, Tomz M, Wittenberg J (2000). "Making the Most of Statistical Analyses: Improving Interpretation and Presentation." *American Journal of Political Science*, **44**(2), 347–361. doi:10.2307/2669316.

Krantz S (2024). ***collapse**: Advanced and Fast Data Transformation*. doi:10.32614/CRAN.package.collapse. R package version 2.0.16.

Krinsky I, Robb AL (1986). "On Approximating the Statistical Properties of Elasticities." *Review of Economics and Statistics*, **68**(4), 715–719. doi:10.2307/1924536.

Lakens D, Scheel AM, Isager PM (2018). "Equivalence Testing for Psychological Research: A Tutorial." *Advances in Methods and Practices in Psychological Science*, **1**(2), 259–269. ISSN 2515-2459. doi:10.1177/2515245918770963.

Lang M (2017). "**checkmate**: Fast Argument Checks for Defensive R Programming." *The R Journal*, **9**(1), 437–445. doi:10.32614/rj-2017-028.

Leeper TJ (2024a). ***margins**: Marginal Effects for Model Objects*. doi:10.32614/CRAN.package.margins. R package version 0.3.28.

Leeper TJ (2024b). ***prediction**: Tidy, Type-Safe `prediction()` Methods*. doi:10.32614/CRAN.package.predictions. R package version 0.3.18.

Lenth RV (2016). "Least-Squares Means: The R Package **lsmeans**." *Journal of Statistical Software*, **69**(1), 1–33. doi:10.18637/jss.v069.i01.

Lenth RV (2024). ***emmeans**: Estimated Marginal Means, aka Least-Squares Means*. doi:10.32614/CRAN.package.emmeans. R package version ?1.10.4.

Lüdecke D (2018). "**ggeffects**: Tidy Data Frames of Marginal Effects from Regression Models." *Journal of Open Source Software*, **3**(26), 772. doi:10.21105/joss.00772.

Lüdecke D, Ben-Shachar MS, Patil I, Wiernik BM, Makowski D (2024). ***easystats**: Framework for Easy Statistical Modeling, Visualization, and Reporting*. doi:10.32614/CRAN.package.easystats. R package version 0.7.3.

Lüdecke D, Waggoner P, Makowski D (2019). "**insight**: A Unified Interface to Access Information from Model Objects in R." *Journal of Open Source Software*, **4**(38), 1412. doi:10.21105/joss.01412.

Maerz SF, Edgell AB, Hellemeier S, Illchenko N (2022). ***vdemdata**: An R Package to Load, Explore and Work with the Most Recent V-Dem (Varieties of Democracy) Dataset*. URL https://github.com/vdeminstitute/vdemdata.

Makowski D, Ben-Shachar MS, Patil I, Lüdecke D (2024). ***modelbased**: Estimation of Model-Based Predictions, Contrasts and Means*. doi:10.32614/CRAN.package.modelbased. R package version 0.8.8.

McDermott G (2024). ***etwfe**: Extended Two-Way Fixed Effects*. doi:10.32614/CRAN.package.etwfe. R package version 0.4.0.

R Core Team (2024). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Robinson D, Hayes A, Couch S (2024). ***broom**: Convert Statistical Objects into Tidy Tibbles*. doi:10.32614/CRAN.package.broom. R package version 1.0.7.

Rothstein B, Teorell J (2008). "What Is Quality of Government? A Theory of Impartial Government Institutions." *Governance*, **21**(2), 165–190. doi:10.1111/j.1468-0491.2008.00391.x.

Sarkar D (2008). ***lattice**: Multivariate Data Visualization with R*. Springer-Verlag, New York. doi:10.1007/978-0-387-75969-2.

Seabold S, Perktold J (2010). "**statsmodels**: Econometric and Statistical Modeling with Python." In *Proceedings of the 9th Python in Science Conference*, volume 57,61, pp. 92–96. Austin. doi:10.25080/majora-92bf1922-011.

StataCorp (2023). *Stata Statistical Software: Release 18*. StataCorp LLC, College Station. URL https://www.stata.com/.

Suzuki K, Demircioglu MA (2021). "Is Impartiality Enough? Government Impartiality and Citizens' Perceptions of Public Service Quality." *Governance*, **34**(3), 727–764. doi:10.1111/gove.12527.

The **pandas** Development Team (2024). "Pandas-Dev/Pandas: **Pandas**." doi:10.5281/zenodo.3509134. Python package version 2.2.3.

Ushey K, Allaire JJ, Tang Y (2024). ***reticulate**: Interface to Python*. doi:10.32614/CRAN.package.reticulate. R package version 1.39.0.

Van der Loo MPJ (2021). "A Method for Deriving Information from Running R Code." *The R Journal*, **13**, 42–52. ISSN 2073–4859. doi:10.32614/rj-2021-056.

Van Rossum G, *et al.* (2011). *Python Programming Language*. URL https://www.python.org.

Vink R (2024). ***polars**: Blazingly Fast DataFrames in Rust, Python, Node.js, R, and SQL*. doi:10.5281/zenodo.13835598. Python package version 1.8.2.

Virtanen P, Gommers R, Oliphant TE, Haberland M, Reddy T, Cournapeau D, Burovski E, Peterson P, Weckesser W, Bright J, Van der Walt SJ, Brett M, Wilson J, Millman KJ, Mayorov N, Nelson ARJ, Jones E, Kern R, Larson E, Carey CJ, Polat İ, Feng Y, Moore EW, VanderPlas J, Laxalde D, Perktold J, Cimrman R, Henriksen I, Quintero EA, Harris CR, Archibald AM, Ribeiro AH, Pedregosa F, Van Mulbregt P, SciPy 10 Contributors (2020). "**SciPy** 1.0: Fundamental Algorithms for Scientific Computing in Python." *Nature Methods*, **17**, 261–272. doi:10.1038/s41592-019-0686-2.

Wickham H (2014). "Tidy Data." *Journal of Statistical Software*, **59**(10), 1–23. doi:10.18637/jss.v059.i10.

Wickham H (2016). ***ggplot2**: Elegant Graphics for Data Analysis*. Springer-Verlag, New York. doi:10.1007/978-0-387-98141-3.

Wickham H, François R, Henry L, Müller K, Vaughan D (2023). ***dplyr**: A Grammar of Data Manipulation*. doi:10.32614/CRAN.package.dplyr. R package version 1.1.4.

Wickham H, Kuhn M, Vaughan D (2022). ***generics**: Common S3 Generics Not Provided by Base R Methods Related to Model Fitting*. doi:10.32614/CRAN.package.generics.

Zeileis A, Köll S, Graham N (2020). "Various Versatile Variances: An Object-Oriented Implementation of Clustered Covariances in R." *Journal of Statistical Software*, **95**(1), 1–36. doi:10.18637/jss.v095.i01.

Affiliation:

Vincent Arel-Bundock
Université de Montréal
Science Politique, Pavillon Lionel-Groulx
3150 Jean-Brillant, C-4020
Montréal, Québec, Canada, H3T 1N8
E-mail: vincent.arel-bundock@umontreal.ca
URL: https://arelbundock.com/

Noah Greifer
Harvard University
Institute for Quantitative Social Science
1737 Cambridge Street
CGIS Knafel Building, Room K350
Cambridge, MA 02138, United States of America
E-mail: ngreifer@iq.harvard.edu

Andrew Heiss
Georgia State University
Department of Public Management and Policy
Andrew Young School of Policy Studies
55 Park Place SE, Room 464
Atlanta, Georgia 30303, United States of America
E-mail: aheiss@gsu.edu
URL: https://www.andrewheiss.com/

Journal of Statistical Software https://www.jstatsoft.org/
published by the Foundation for Open Access Statistics https://www.foastat.org/
November 2024, Volume 111, Issue 9 *Submitted:* 2023-08-20
doi:10.18637/jss.v111.i09 *Accepted:* 2024-05-16