Interpreting Regression Results using Average Marginal Effects with R's **margins**

Thomas J. Leeper

June 12, 2024

Abstract

Applied data analysts regularly need to make use of regression analysis to understand descriptive, predictive, and causal patterns in data. While many applications of ordinary least squares yield estimated regression coefficients that are readily interpretable as the predicted change in y due to a unit change in x, models that involve multiplicative interactions or other complex terms are subject to less clarity of interpretation. Generalized linear models that involve transformations of this linear predictor into binary, ordinal, count or other discrete outcomes lack such ready interpretation. As such, there has been much debate in the literature about how best to interpret these more complex models (e.g., what quantities of interest to extract? what types of graphical presentations to use?). This article proposes that marginal effects, specifically average marginal effects, provide a unified and intuitive way of describing relationships estimated with regression. To begin, I briefly discuss the challenges of interpreting complex models and review existing views on how to interpret such models, before describing average marginal effects and the somewhat challenging computational task of extracting this quantity of interest from regression results. I conclude with implications for statistical practice and for the design of statistical software.

Regression is a workhorse procedure in modern statistics. In disciplines like economics and political science, hardly any quantitative research manages to escape the use of regression modelling to describe patterns in multivariate data, to assess causal relationships, and to formulate predictions. Ordinary least squares (OLS) regression offers a particularly attractive procedure because of its limited and familiar assumptions and the ease with which it expresses a multivariate relationship as a linear additive relationship between many regressors (i.e., predictors, covariates, or righthand-side variables) and a single outcome variable. The coefficient estimates from an OLS procedure are typically easily interpretable as the expected increase in the outcome due to a unit change in the corresponding regressor.

This ease of interpretation of simple regression models, however, belies a potential for immense analytic and interpretative complexity. The generality of the regression framework means that it is easily generalized to examine more complex relationships, including the specification of non-linear relationships between regressor and outcome, multiplicative interactions between multiple regressors, and transformations via the generalized linear model (GLM) framework.[1] With this flexibility to specify potentially complex multivariate relationships comes the risk of misinterpretation [4, 3] and, indeed, frequent miscalculation of quantities of interest [1, 13]. Coefficient estimates in models that are non-linear or involve interactions lose their direct interpretation as unconditional marginal effects, meaning that interpretation of tabular or graphical presentations requires first understanding the details of the specified model to avoid interpretation errors. Coefficient estimates in GLMs are often not directly interpretable at all.

For these reasons, and in the interest of making intuitive tabular and visual displays of regression results, there is a growing interest in the display of substantively meaningful quantities of interest that can be drawn from regression estimates [10]. This article reviews the literature on substantive interpretation of regression estimates and argues that researchers are often interested in knowing the *marginal effect* of a regressor on an outcome. I propose *average marginal effects* as a particularly useful quantity of interest, discuss a computational approach to calculate marginal effects, and offer the **margins** package for R [11] as a general implementation.

The outline of this text is as follows: section 1 describes the statistical background of regression estimation and the distinctions between estimated coefficients and estimated marginal effects of righthand-side variables, Section 2 describes the computational implementation of **margins** used to obtain those quantities of interest, and Section 3 compares the results of the package to those produced by Stata's `margins` command [15, 19], and various R packages.

[1] Further complexities arise from other expansions of the regression approach, such as interdependent or hierarchically organized observations, instrumental variables methods, and so on.

1 Statistical Background

The quantity of interest typically reported by statistical software estimation commands for regression models is the regression coefficient (along with standard errors thereof, and various goodness-of-fit and summary statistics). Consider, for example, a trivial regression of country population size as a function of GDP per capita, life expectancy, and the interaction of the two. (As should be obvious, this model is not intended to carry any causal interpretation.)

Table 1: Example OLS Regression Output

	Dependent variable:	
	Population Size	
	(1)	(2)
loggdp	-26.440^{***}	-12.095
	(3.450)	(11.748)
lifeExp	2.586^{***}	4.412^{***}
	(0.332)	(1.468)
loggdp:lifeExp		-0.231
		(0.181)
Constant	91.543^{***}	-19.078
	(17.060)	(88.265)
Observations	1,704	1,704
R^2	0.037	0.038
Adjusted R^2	0.036	0.037
Residual Std. Error	104.212 (df = 1701)	104.193 (df = 1700)
F Statistic	33.092^{***} (df = 2; 1701)	22.613^{***} (df = 3; 1700)
Note:		*p<0.1; **p<0.05; ***p<0.01

This default output makes sense for additive linear models (i.e., ordinary least squares regression) because an estimated coefficient is readily and directly interpretable as the expected change in y given a unit change in x, holding all other terms constant (see, for example, the coefficient estimates for model 1). A unit change in life expectancy is associated with a population that is 2,586 larger. This *marginal effect* of life expectancy is constant across observations in the dataset, constant across observed values of other variables, and constant across levels of itself. The "effect" is thus *unconditional*.

When model specifications include other kinds of terms (e.g., multiple higher powers of a given variable, log transformations, or interactions between variables), the coefficients on those variables do not and cannot clearly communicate the association between

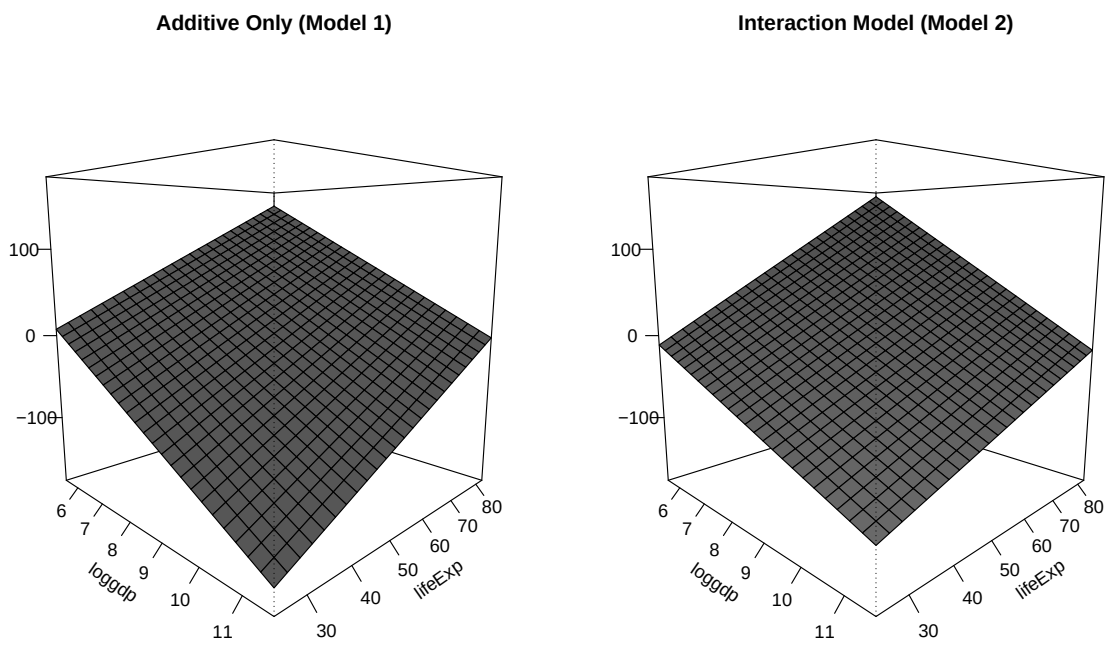


Figure 1: Expected Population (millions) by GDP/Capita and Life Expectancy

a given righthand-side variable and the outcome because that relationship is parameterized through multiple coefficients. In model 2, for example, the relationship between life expectancy and population is parameterized to account for heterogeneity due to GDP per capita (and vice versa, the relationship between GDP per capita and population is parameterized to be heterogeneous across levels of life expectancy). The estimated coefficient for life expectancy (4.412) only carries the meaning of an unconditional marginal effect when logged GDP per capita (and thus the value of `loggdp:lifeExp`) is zero, which is never. Looking at a single estimated coefficient and treating it as an unconditional marginal effect leads to numerous problematic interpretations [4].

Visualization of the fitted values from the model makes clear that the two specifications are actually producing fairly similar substantive estimates. Figure 1 shows the fitted response surface from model 1 (left) and model 2 (right). With the exception of the fitted values at the intersection of very low levels of life expectancy and very high levels of GDP per capita (where data are sparse), the two specifications yield nearly identical substantive interpretations. Higher GDP per capita is associated with lower national population, and higher levels of life expectancy are associated with higher national population. The ambiguous substantive meaning of the results in Table 1 is due to the fact that the coefficients are not intepretable as *marginal effects*.

But what is a marginal effect exactly? And why is it a useful quantity of interest? Figure 2 shows the calculation of the marginal effect of GDP per capita on population

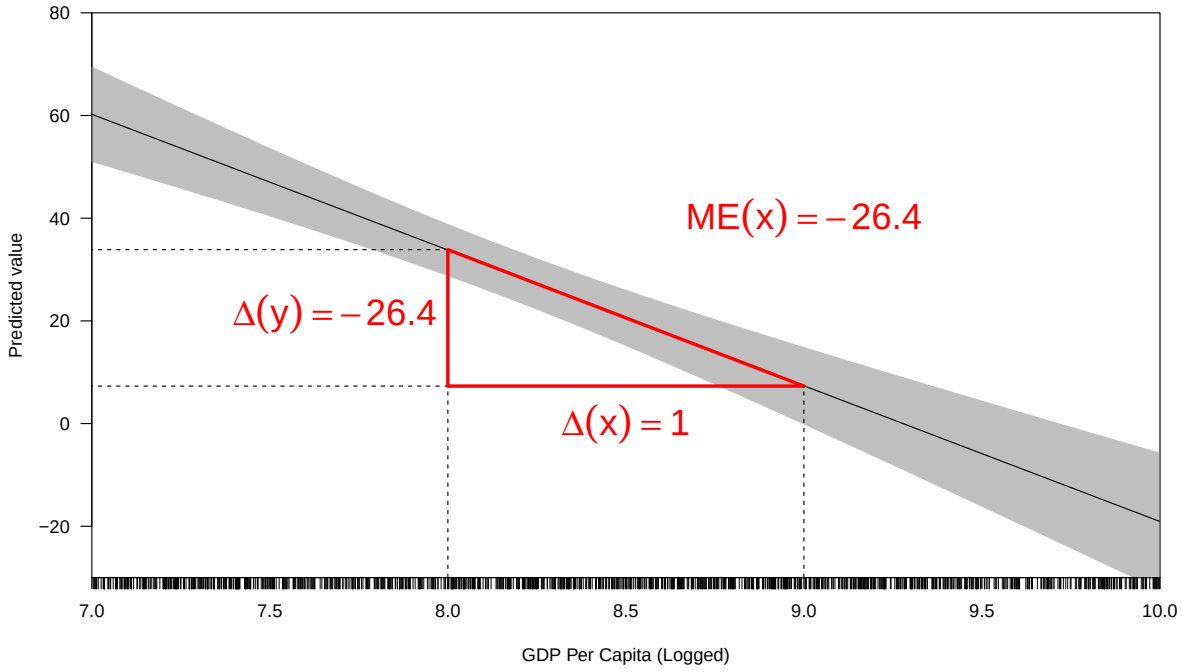


Figure 2: Marginal Effect of GDP per Capita on Population Size, as shown by Expected Value Plot

size, based upon model 1 in Table 1. Because the model is specified without interactions, the slope of the predicted relationship between x and y is straight linear, so the marginal effect of x is estimated as the slope of this fitted value line, using the usual $\frac{\Delta Y}{\Delta X}$ calculation and can be calculated irrespective of the value of life expectancy. The visibly apparent marginal effect of -26.4 is clearly consistent with the coefficient estimate reported in Table 1, model 1.

1.1 Generalized Linear Models

Furthermore, when models involve a non-linear transformation (e.g., generalized linear models such as logit or probit), the coefficients are typically not directly interpretable at all (even when no power terms, interactions, or other complex terms are included). This is because the coefficients express the influence of each separate variable onto the latent, linear scale of the outcome, not the discrete (or probability) scale of the observed outcome [12]. For example, in a logistic regression, the coefficients express the marginal effect of each included variable in terms of the change in log-odds that the outcome equals 1 given a unit change in the independent variable. In order to express the more intuitive change in the predicted probability that the outcome equals 1 requires conditioning on all other included variables (i.e., selecting a set of values for all righthand-side variables) and running that set of values through the link function to convert log-odds to proba-

bilities, thus making the marginal effect (in probability terms) of one variable a function of all other variables included in the model. As such, for both OLS and GLMs, the coefficients estimated for a regression model can often provide unintuitive insight into the statistical relationships between variables and, worse, can frequently fail to communicate those relationships at all (as in GLMs, where the size of coefficients can be completely uninformative about the size of the "effect" of a given righthand-side variable).

1.2 Quantities of Interest

Several quantities of interest may be derived from any regression model result. The first and most obvious, are the coefficient estimates themselves (as have already been discussed). Others include fitted or predicted values of the outcome generated from the model estimates, first-differences or discrete changes, marginal effects or partial effects.

Among these, fitted (predicted) values communicate the shape and position of the fitted regression surface (or line in a simple bivariate regression) across the possibly multidimensional covariate space. In essence, if we express the regression estimates as a function of X, $\hat{y} = \hat{f}(X)$, we can evaluate this function at any levels of the covariates. Predicted values communicate what outcome value would be expected given the patterns observed between covariates and the outcome. We might be interested in at least three different quantities that can be calculated from the regression fit:

1. Fitted values at representative, or particular, values of X
2. Fitted values at the mean of X
3. "Average" fitted values

The first of these is simply the evaluation of the fitted value function $\hat{f}(X = x)$, for some particular value x or combination of covariate values. This might be used, for example, to describe what population we would expect to see for a country with a particular combination of life expectancy and GDP values. It would simply report to us the value of the population along the surface shown in Figure 1.

The second, fitted values at the mean of X, is a summary measure that chooses the values of covariates based upon properties of the distribution of X rather than based on some theoretical reason. Unfortunately, such a quantity is not particularly useful to know because the multidimensional mean of X may not be an observed (or even observable) value.

The third measure, average fitted values, calculates the value $\hat{y}$ for every case in the data and averages the resulting fitted values. The interpretation of this quantity is as the average (or typical) outcome we would expect to observe were our model an accurate representation of the data-generating process for the outcome.

From these predicted values, one could generate a second class of quantities of interest: discrete changes or "first differences." As their name implies, this quantity expresses the *change* in the predicted outcome that occurs with a given change in the value of covariate(s). Common shifts might include the change in a value of a categorical variable (e.g., from "male" to "female") or a theoretical meaningful change in a numeric variable (e.g., from its minimum to maximum values).

Finally, we may be interested in the *marginal effect* of a given variable: that is, the slope of the regression surface with respect to a given covariate. The marginal effect communicates the rate at which y changes at a given point in covariate space, with respect to one covariate dimension and holding all covariate values constant. This quantity is particularly useful because it is intuitive — it is simply a slope — and because it can be calculated from essentially any set of regression estimates.

To calculate marginal effects requires the application of partial derivatives. A marginal effect is, in essence, the slope of multi-dimensional surface with respect to one dimension of that surface.[2] Marginal effects are a particularly useful quantity of interest because, in the case of OLS, they translate the coefficients estimated from any model parameterization back into the quantity that is expressed by coefficients in any unconditional model: namely the marginal contribution of x to the outcome. As with fitted values, we may be interested in one of three different quantities of interest derived from marginal effects:

- Marginal effects at representative values (MERs)
- Marginal effects at means (MEMs)
- Average marginal effects (AMEs)

These quantities are analogous to the three fitted values quantities discussed earlier. MERs calculate the marginal effect of each variable at a particular combination of X values that is theoretically interesting. MEMs calculate the marginal effects of each variable at the means of the covariates. AMEs calculate marginal effects at every observed value of X and average across the resulting effect estimates.

AMEs are particularly useful because — unlike MEMs — produce a single quantity summary that reflects the full distribution of X rather than an arbitrary prediction. Together with MERs, AMEs have the potential to convey a considerable amount of information about the influence of each covariate on the outcome. Because AMEs average across the variability in the fitted outcomes, they can also capture variability better than MEMs. While MERs provide a means to understand and communicate model estimates at theoretically important combinations of covariate values, AMEs provide a natural summary measure that respects both the distribution of the original data and does not rely on summarizing a substantively unobserved or unobservable X value (as in MEMs).

[2]In practice, when categorical covariates are used in a model, the "marginal effect" label is applied to quantities calculated as discrete changes.

1.3 Response to Earlier Critiques

It is worth noting that **(author?)** [10] make a strong case against marginal effects calculations premised largely on two points: (1) the computational challenge of calculating marginal effects, and (2) the common (at that time) omission of statements of uncertainty in the presentation of derivative-based quantities of interest (see, for example, displays in **(author?)** 12). While the former is challenging, the next section discusses various computational approaches to derivation including an approximation that is fully general. **margins** provides a first-of-its-kind implementation of this approach in the R language. This technical innovation makes this concern less problematic than it once was. With respect to the latter critique, the subsequent section then discusses the delta method approach to approximating variances of marginal effects, something **(author?)** [10] discuss but dismiss. While **margins** adopts a delta method approach, the simulation method described by these authors is also implemented as an alternative, alongside a third alternative (bootstrapping).

2 Computational Details

This section describes the basic computational features of **margins**, which offers a fully general approach to estimating marginal effects from an arbitrary modelling result. Specifically, it describes the procedures for calculating marginal effects from the information stored in a model object (e.g., an R object of class `"lm"` or `"glm"`) and the procedures for estimating the variances of those marginal effects.

2.1 Symbolic Derivatives

If we were to calculate marginal effects by hand, we might rightly choose to rely on manual or "symbolical" differentiate (i.e., using a set of known derivation rules, derive the partial derivative of a regression equation with respect to its constituent variables). The advantage of this approach is that, with the exception of any human error, produces numerically perfect calculations of the marginal effects. Yet this approach is not particularly easy to implement computationally. It may seem straightforward, but even in relatively simple regression specifications, the formulae for marginal effects quickly become complicated.

Consider, for example, the three regression equations shown in the first column of Table 2. The first includes a simple two-way multiplicative interaction term, the second includes a power term, and the third includes a three-way interaction and its constituent two-way interactions. The formula for the marginal effect with respect to each variable in each equation is shown in the third column. While the equations vary in their complexity,

Table 2: Marginal Effect Formulae for a Few Simple Regression Equations

	Regression Equation	ME with respect to	ME Formula
(1)	$\hat{Y} = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1 X_2$	X_1	$\frac{\partial Y}{\partial X_1} = \beta_1 + \beta_3 X_2$
		X_2	$\frac{\partial Y}{\partial X_2} = \beta_2 + \beta_3 X_1$
(2)	$\hat{Y} = \beta_0 + \beta_1 X_1 + \beta_2 X^2 + \beta_3 X_2$	X_1	$\frac{\partial Y}{\partial X_1} = \beta_1 + 2\beta_2 X_1$
		X_2	$\frac{\partial Y}{\partial X_2} = \beta_3$
(3)	$\hat{Y} = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \beta_4 X_1 X_2 + \beta_5 X_2 X_3 + \beta_6 X_1 X_3 + \beta_7 X_1 X_2 X_3$	X_1	$\frac{\partial Y}{\partial X_1} = \beta_1 + \beta_4 X_2 + \beta_6 X_3 + \beta_7 X_2 X_3$
		X_2	$\frac{\partial Y}{\partial X_2} = \beta_2 + \beta_4 X_1 + \beta_5 X_3 + \beta_7 X_1 X_3$
		X_3	$\frac{\partial Y}{\partial X_3} = \beta_3 + \beta_5 X_2 + \beta_6 X_1 + \beta_7 X_1 X_2$

the key intuition to draw is that calculating a marginal effect from an equation of any complexity requires three steps:

1. Identify which coefficients are attached to terms that include the variable of interest, x
2. Using a set of derivative rules, specify the partial derivative of the equation with respect to x
3. Implement the partial derivative equation computationally

As should be clear from the formulae in the third column, the first step is fairly easy (though potentially error-prone), the second requires only knowledge of fairly basic derivation, and the third requires simply expressing an additive formula as code. For example, the three marginal effects for Equation (3) in Table 2 are simply:

```
library("stats")
m <- lm(y ~ x1 * x2 * x3)
cf <- coef(m)[-1] # drop beta_0
me_x1 <- cf[1] + cf[4]*x2 + cf[6]*x3 + cf[7]*x2*x3
me_x2 <- cf[2] + cf[4]*x1 + cf[5]*x3 + cf[7]*x1*x3
me_x3 <- cf[3] + cf[5]*x2 + cf[6]*x1 + cf[7]*x1*x2
```

The code necessary to perform these calculations is simple, but it is sensitive to human error. And if the model is modified, the positions of coefficients may change, breaking code. Furthermore, the second analytic step (defining the partial derivatives according to a set of derivation rules) is actually extremely complicated, at least in order to handle any general class of model formulae. The challenges are quickly apparent in the complexity of the code for `deltaMethod()` from the **car** package [8], which attempts to implement symbolic derivation for model formulae. In computational terms, this is especially true when considering the fairly limited information contained in the formula expression used to specify the model (`y ~x1 * x2 * x3`).

If we were able to fully expand that notation and introduce placeholders for coefficients, then R's symbolic derivative function (`deriv()`) could return accurate specifications of the three marginal effects as *expressions* shown in the `.grad[, "x1"]`, `.grad[, "x1"]`, etc.:

```
deriv(y ~ b1*x1 + b2*x2 + b3*x3 + b4*x1*x2 +
          b5*x2*x3 + b6*x1*x3 + b7*x1*x2*x3,
      c("x1", "x2", "x3"))

## expression({
##     .expr6 <- b4 * x1
```

```
##     .expr9 <- b5 * x2
##     .expr12 <- b6 * x1
##     .expr15 <- b7 * x1
##     .expr16 <- .expr15 * x2
##     .value <- b1 * x1 + b2 * x2 + b3 * x3 + .expr6 * x2 + .expr9 *
##         x3 + .expr12 * x3 + .expr16 * x3
##     .grad <- array(0, c(length(.value), 3L), list(NULL, c("x1",
##         "x2", "x3")))
##     .grad[, "x1"] <- b1 + b4 * x2 + b6 * x3 + b7 * x2 * x3
##     .grad[, "x2"] <- b2 + .expr6 + b5 * x3 + .expr15 * x3
##     .grad[, "x3"] <- b3 + .expr9 + .expr12 + .expr16
##     attr(.value, "gradient") <- .grad
##     .value
## })
```

But, this is prone to the same limitations as the manual calculation of marginal effects, shown above, wherein human error and iterative changes to the model easily produce incorrectly results.

A further challenge arises due to the terse syntax needed to express a complex model in R's modelling language — the "formula" class, derived from GENSTAT [18]. It is quite easy to express a model for which the mapping of righthand-side variables to coefficients is completely intransparent, such as `y  0 - (x1 + x2) * (x3 + x4)`. Additional sources of useful generality in the language, such as the expression of "factors" (categorical variables via `factor()`) and on-the-fly variable transformations through `I()` statements, impede symbolic derivation. These useful features mean it is possible to express a model in formula markup for which no symbolic derivative rule is available, such as Equation (2) in Table 2:

```
deriv(y ~ b1*x1 + b2*I(x1^2) + b3*x2, c("x1", "x2"))

Error in deriv.formula(y ~ b1 * x1 + b2 * I(x1^2) + b3 * x2, c("x1", "x2")) :
  Function 'I' is not in the derivatives table
```

and for which any reasonable workaround would produce an inaccurate set of symbolically arrived at partial derivatives.[3] For example, we could define a new variable `x1squared` equal to x_1^2, but the symbolic differentiation rules are blind to the underlying relationship between x_1 and the new variable:

[3]The result of `deriv(y ~b1*x1 + b2*x1^2 + b3*x2, c("x1", "x2"))` is correct, but would require recognizing when `I()` is and is not meaningful in a formula and modifying it accordingly.

```
deriv(y ~ b1*x1 + b2*x1squared + b3*x2, c("x1", "x2"))

## expression({
##     .value <- b1 * x1 + b2 * x1squared + b3 * x2
##     .grad <- array(0, c(length(.value), 2L), list(NULL, c("x1",
##         "x2")))
##     .grad[, "x1"] <- b1
##     .grad[, "x2"] <- b3
##     attr(.value, "gradient") <- .grad
##     .value
## })
```

The marginal effect arrived at symbolically ignores that the effect of x_1 depends on the value of itself because of the additional squared term. By breaking the relationship between x_1 and its squared term, x_1^2, the derivative rules lead us to a plainly inaccurate result (that the marginal effect of x_1 is simply β_1).

2.2 Numerical Derivatives

What then can be done? A standard answer — and the answer chosen by Stata's `margins` command [15] — is to rely on numerical derivatives (i.e., numeric approximations of the partial derivatives). The R **margins** package follows this approach.

What is a numerical approximation? Rather than defining a partial derivative as an exact formula using symbolic derivation rules, a numerical derivative approximates the slope of the response function from a model by taking small steps, h in x, calculating the $\hat{y}$ at each point, and then applying a simple difference method to define the slope at point x:

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \tag{1}$$

While seemingly crude, as $h \lim 0$, the result converges on the true partial derivative and every value of x and requires no knowledge of formula for the partial derivative(s). To provide some intuition, Figure 3 displays a few numerical derivatives of $f(x) = x^2$ at the point $x = 1$ for various large values of h (2.0, 1.0, 0.5, 0.25). As should be clear, as h decreases, the approximations approach the true slope of the derivative, $f'(x) = 2 * x$, which is 2. Inferring the partial derivative across the full domain of x requires repeating this approximation process at every substantively meaningful value of x.

At large values of h and even fairly small ones, this "one-sided" derivative can be quite inaccurate. A "two-sided" or "symmetric difference" approach uses points above and below x:

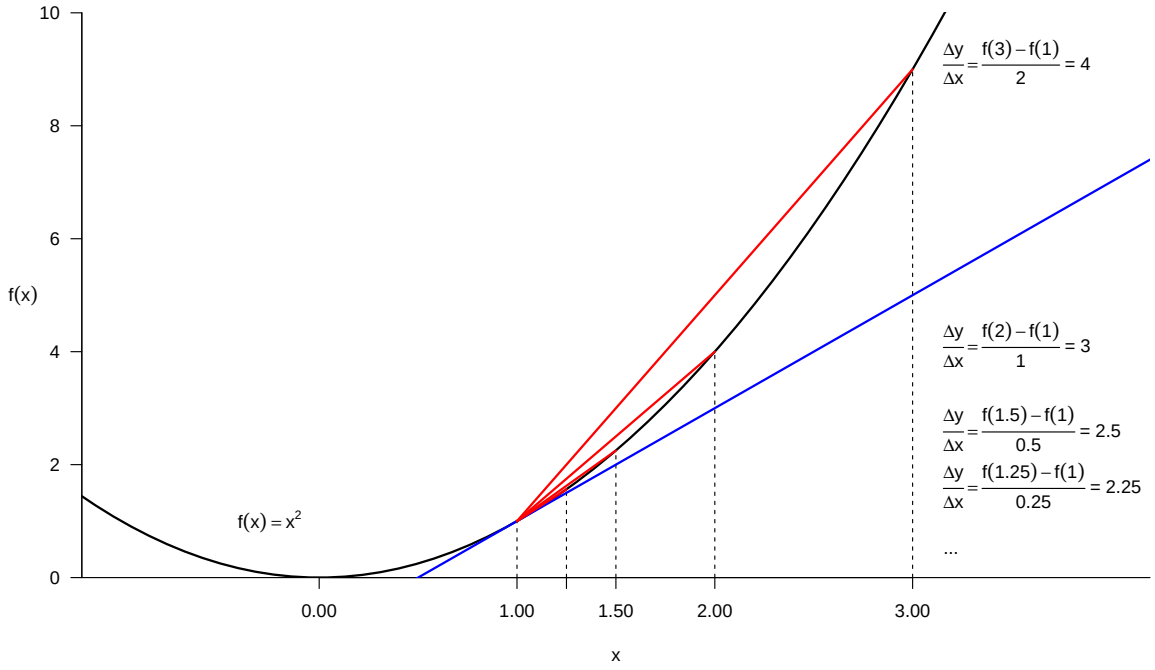


Figure 3: Approximation of Derivative via One-Sided Numerical Approach

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x-h)}{2h} \tag{2}$$

This approach, which is shown in Figure 4, will tend to be more accurate. As it so happens, for the function $f(x) = x^2$, this approach calculates $f'(x)$ accurately even for very large values of h.

Computationally, this two-sided numerical approximation is achieved via R's `predict()` method, which provides (for a given model result) the fitted value $\hat{Y}$ for every observation in a data frame.[4] In essence, `predict()` represents the equation $\hat{Y} = f(X)$ as the function of a model object (containing coefficient estimates) and a data frame (defaulting the original data used in estimation, such that `fitted(model)` and `predict(model)` are equivalent). This means that **margins** can produce marginal effects estimates for any model object class that offers a `predict()` method.

At a low level, **margins** provides a function, `dydx()`, which implements the numerical derivative procedure. Taking a model object, `model`, and data frame, `data`, as input, the function calculates a value of h that accounts for floating point errors (the internal function `setstep()`,[5] shown below), generates two data frames `d0` (representing $f(x-h)$) and `d1` (representing $f(x+h)$), calls `predict()` on `d0` and `d1`, and calculates the

[4] **margins** provides a type-consistent wrapper for this, called `prediction()` that always returns a data frame (rather than a vector or list of fitted values).

[5] A relatively large value of h is used by default: $sqrt(1x10^{-}7)$. Alternative values can be specified manually.

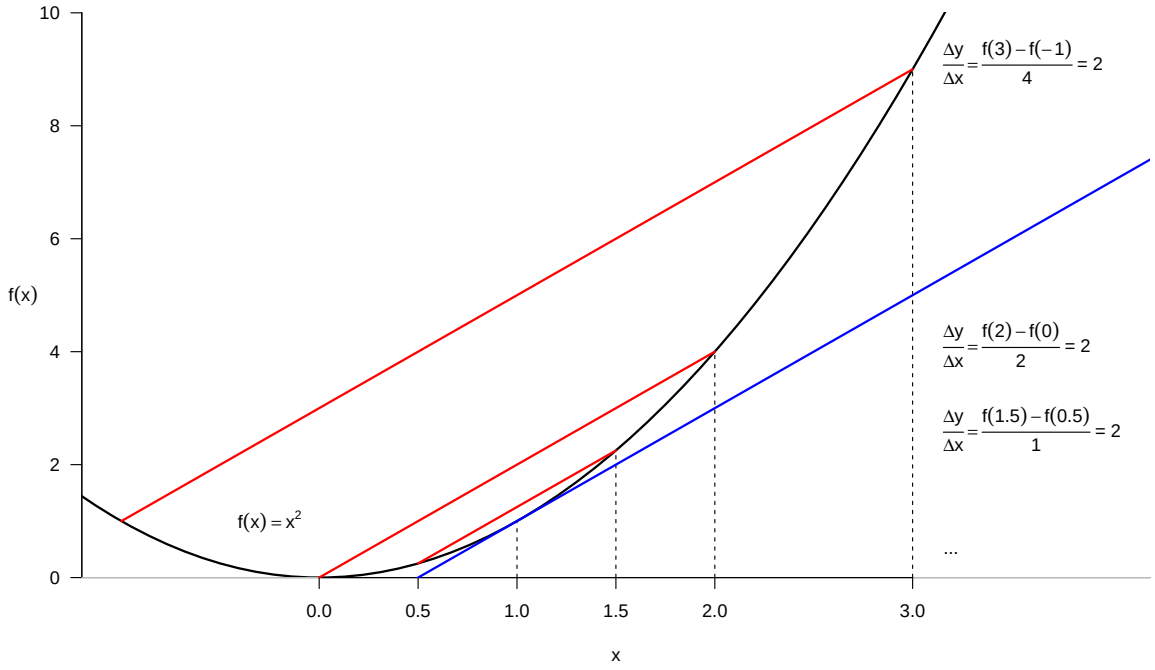


Figure 4: Approximation of Derivative via Two-Sided Numerical Approach

numerical derivative from the resulting fitted $\hat{Y}$ values according to a two-sided numerical approximation:

```
d0 <- d1 <- data
...
d0[[variable]] <- d0[[variable]] - setstep(d0[[variable]])
d1[[variable]] <- d1[[variable]] + setstep(d1[[variable]])
...
P0 <- prediction(model = model, data = d0, type = type)[["fitted"]]
P1 <- prediction(model = model, data = d1, type = type)[["fitted"]]
...
out <- (P1 - P0) / (d1[[variable]] - d0[[variable]])
```

The result is a vector of marginal effects estimates — the marginal effect at every observed combination of X in the data. This low-level function is wrapped within a function `margins()`, which is an S3 generic with methods for various common modelling procedures. `margins()` calculations marginal effects of *all* variables used in a model, along with their variances, and accommodates the estimation of those effects across arbitrary values of X variables (emulating Stata's `margins, at()` notation). Calling `dydx()` or `marginal_effects()` provides a means to calculate marginal effects without specifying new data or calculating variances, which can be time-consuming. The output of

`margins()` is a "data.frame" with an additional "margins" class, that contains the original input data, the fitted values from the model, and the marginal effects estimates for each observation:

```
str(mar2 <- margins(m2))

## Classes 'margins' and 'data.frame': 1704 obs. of  15 variables:
##  $ country         : Factor w/ 142 levels "Afghanistan",..: 1 1 1 1 1 1 1 1 1 1 ..
##  $ continent       : Factor w/ 5 levels "Africa","Americas",..: 3 3 3 3 3 3 3 3 3 3
##  $ year            : int  1952 1957 1962 1967 1972 1977 1982 1987 1992 1997 ...
##  $ lifeExp         : num  28.8 30.3 32 34 36.1 ...
##  $ pop             : num  8.43 9.24 10.27 11.54 13.08 ...
##  $ gdpPercap       : num  779 821 853 836 740 ...
##  $ loggdp          : num  6.66 6.71 6.75 6.73 6.61 ...
##  $ fitted          : num  -16.83 -13.41 -9.4 -3.23 5.18 ...
##  $ se.fitted       : num  8.93 8.27 7.63 6.99 6.58 ...
##  $ dydx_loggdp     : 'marginaleffect' num  -18.7 -19.1 -19.5 -20 -20.4 ...
##  $ dydx_lifeExp    : 'marginaleffect' num  2.87 2.86 2.85 2.86 2.89 ...
##  $ Var_dydx_loggdp : num  12.1 12.1 12.1 12.1 12.1 ...
##  $ Var_dydx_lifeExp: num  0.112 0.112 0.112 0.112 0.112 ...
##  $ _weights        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ _at_number      : int  1 1 1 1 1 1 1 1 1 1 ...
##  - attr(*, "vcov")= num [1:2, 1:2] 12.126 -0.945 -0.945 0.112
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:2] "dydx_loggdp.1" "dydx_lifeExp.1"
##   .. ..$ : chr [1:2] "dydx_loggdp.1" "dydx_lifeExp.1"
##  - attr(*, "jacobian")= num [1:2, 1:4] -3.20e-07 4.44e-08 1.00 6.22e-08 7.11e-08 .
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:2] "dydx_loggdp.1" "dydx_lifeExp.1"
##   .. ..$ : chr [1:4] "(Intercept)" "loggdp" "lifeExp" "loggdp:lifeExp"
##  - attr(*, "type")= chr "response"
##  - attr(*, "call")= language lm(formula = pop ~ loggdp * lifeExp, data = gapminder
##  - attr(*, "model_class")= chr "lm"
##  - attr(*, "vce")= chr "delta"
##  - attr(*, "weighted")= logi FALSE
```

Attributes contain some additional information about the procedure. By default the AMEs are printed and a `summary()` method provides a more complete output that is useful for interactive sessions:

```
mar2

## Average marginal effects
## lm(formula = pop ~ loggdp * lifeExp, data = gapminder)

##  loggdp lifeExp
##  -25.83   2.528

summary(mar2)

##   factor      AME     SE       z      p    lower    upper
##  lifeExp   2.5280 0.3345  7.5570 0.0000   1.8723   3.1836
##   loggdp -25.8320 3.4823 -7.4181 0.0000 -32.6572 -19.0068
```

For users of Stata's `margins` command, this output should look very familiar.

A few final points about the computational details of **margins** are worth noting. First, much numerical differentiation in R is conducted using the **numDeriv** package. This approach is not used here because **numDeriv** is not vectorized and is thus quite slow. The issue arises because **numDeriv** calculates $f(x-h)$ and $f(x+h)$ via for-loop, iterating over observations in a data set. **margins** provides a significant performance enhancement by using the vectorized procedures shown above.

Second, **margins** detects the *class* of variables entered into a regression, distinguishing numeric variables from factor, ordered, and logical. For the non-numeric classes, discrete differences rather than partial derivatives are reported as the partial derivative of a discrete variable is undefined. For factors (and "ordered") variables, changes are expressed moving from the base category to a particular category (e.g., from male to female, high school to university education, etc.). For logical variables, discrete changes are expressed moving from `FALSE` to `TRUE`. The treatment of ordered variables (in essence treating them as factors) differs from R's default behavior.

Third, the `type` argument accommodates different quantities of interest in non-linear models, such as generalized linear models. For a logistic regression model, for example, we may want to interpret marginal effects (sometimes "partial effects") on the scale of the observed outcome, so that we can understand the marginal effects as changes in the predicted probability of the outcome. By default, **margins** sets `type = "response"`. This can, however, be modified. For GLMs, this could be set to `type = "link"` in order to calculate true marginal effects on the scale of the linear predictor.

And, finally, instead of the default *instantaneous* marginal effects, discrete changes can be requested for numeric X variables, by specifying the `change` argument to the workhorse `dydx()` function, which allows for expression of changes from observed minimum to maximum, the interquartile range, mean +/- a standard deviation, or any

arbitrary step.

2.3 Variance Approximation with the Delta Method

Calculating the variances of marginal effects is — like the calculation of marginal effects themselves — possible if one can easily express and compute a marginal effect *symbolically*. But just as a general solution to the problem of marginal effect calculation quickly necessitated a numerical approximation, so too does the calculation of variances in that framework.

The first step is to acknowledge that the marginal effects are nested functions of X. Consider, for example, Equation 1 in Table 2, which provides two marginal effects:[6]

$$ME(X_1) = \frac{\partial Y}{\partial X_1} = f_1'(X) = g_1(f(X)) \tag{3}$$

$$ME(X_2) = \frac{\partial Y}{\partial X_2} = f_2'(X) = g_2(f(X)) \tag{4}$$

To calculate the variances of these marginal effects, **margins** relies on the delta method to provide an approximation (following the lead of Stata). The delta method provides that the variance-covariance matrix of the marginal effect of each variable on Y is given by:

$$Var(ME) = J \times Var(\beta) \times J' \tag{5}$$

where $Var(\beta)$ is the variance-covariance matrix of the regression coefficients, estimated by $Var(\hat{\beta})$ and the Jacobian matrix, J, is an M-x-K matrix in which each row corresponds to a marginal effect and each column corresponds to a coefficient:

$$\mathrm{J} = \begin{bmatrix} \frac{\partial g_1}{\partial \beta_0} & \frac{\partial g_1}{\partial \beta_1} & \frac{\partial g_1}{\partial \beta_2} & \cdots & \frac{\partial g_1}{\partial \beta_K} \\ \frac{\partial g_2}{\partial \beta_0} & \frac{\partial g_2}{\partial \beta_1} & \frac{\partial g_2}{\partial \beta_2} & \cdots & \frac{\partial g_2}{\partial \beta_K} \\ \cdots & & & & \\ \frac{\partial g_M}{\partial \beta_0} & \frac{\partial g_M}{\partial \beta_1} & \frac{\partial g_M}{\partial \beta_2} & \cdots & \frac{\partial g_M}{\partial \beta_K} \end{bmatrix}$$

Intuition surrounding the Jacobian can be challenging because the entries are partial derivatives of the marginal effects with respect to the β's, not the X's. Thus it involves the somewhat unintuitive exercise of treating the coefficients (β's) as variables and the original data variables (X's) as constants. Continuing the running example, the Jacobian for the two marginal effects of Equation (1) in Table 2 would be a 2-x-4 matrix, where

[6] It would, of course, be possible to specify marginal effects with respect to other X variables but because they are not included in the regression equation, the marginal effects of all other variables are, by definition, zero.

the first column (expressing the partial derivative of each marginal effect with respect to the intercept) is always zero:

$$\mathrm{J} = \begin{bmatrix} 0 & 1 & 0 & X_2 \\ 0 & 0 & 1 & X_1 \end{bmatrix}$$

such that $Var(ME)$ is:

$$\begin{bmatrix} 0 & 1 & 0 & X_2 \\ 0 & 0 & 1 & X_1 \end{bmatrix} \times \begin{bmatrix} Var(\beta_0) & Cov(\beta_0, \beta_1) & Cov(\beta_0, \beta_2) & Cov(\beta_0, \beta_3) \\ Cov(\beta_0, \beta_1) & Var(\beta_1) & Cov(\beta_1, \beta_2) & Cov(\beta_1, \beta_3) \\ Cov(\beta_0, \beta_2) & Cov(\beta_1, \beta_2) & Var(\beta_2) & Cov(\beta_2, \beta_3) \\ Cov(\beta_0, \beta_3) & Cov(\beta_1, \beta_3) & Cov(\beta_2, \beta_3) & Var(\beta_1) \end{bmatrix} \times \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \\ X_2 & X_1 \end{bmatrix}$$

Multiplying this through, we arrive at a 2-x-2 variance-covariance matrix for the marginal effects:

$$\begin{bmatrix} Var(ME(X_1)) & Cov(ME(X_1), ME(X_2)) \\ Cov(ME(X_1), ME(X_2)) & Var(ME(X_2)) \end{bmatrix}$$

where

$$\begin{aligned} Var(ME(X_1)) &= Var(\beta_1) + 2X_2 Cov(\beta_1, \beta_3) + X_2^2 Var(\beta_1)) \\ Var(ME(X_2)) &= Var(\beta_2) + 2X_1 Cov(\beta_2, \beta_3) + X_1^2 Var(\beta_1) \\ Cov(ME(X_1), ME(X_2)) &= Cov(\beta_1, \beta_2) + X_2 Cov(\beta_2, \beta_3) + \\ &\quad X_1 Cov(\beta_1, \beta_3) + X_1 X_2 Var(\beta_1) \end{aligned}$$

To achieve this computationally, **margins** uses a numerical approximation of the Jacobian. The computational details necessary to express this for any regression model are similar to those for approximating the marginal effects themselves. This is achieved by creating a "function factory" that accepts data and a model object as input and returns a function that holds data constant at observed values, but modifies the estimated coefficients according some new input, applying `predict()` to the original data and modified coefficients.[7] The same numerical differentiation methods as above are then applied to this function, to approximate the Jacobian.[8]

[7] This re-expresses $g(x, \beta)$ as a function only of coefficients: $g(\beta)$, holding x constant.

[8] As a computational note, **margins** uses the standard variance-covariance matrix returned by any modelling function as the value of $Var(\hat{\beta})$ but also alternative values to be specified via `vcov` argument to `margins()`.

Type	Function	Behavior
Core	`dydx()`	Calculate marginal effect of one, named variable
	`marginal_effects()`	Calculate marginal effects of all variables in a model
	`margins()`	An S3 generic to calculate marginal effects for all variables and their variances
Visualization	`plot.margins()`	An analogue to Stata's `marginsplot` command that plots calculated marginal effects.
	`cplot()`	An S3 generic that plots *conditional* fitted values or marginal effects across a named covariate
	`persp.lm()`, etc.	S3 methods for the `persp()` generic that provide three-dimensional representations akin to `cplot()` but for two covariates
	`image.lm()`, etc.	S3 methods for the `image()` generic that produce flat representations of the `persp()` plots
Utilities	`build_margins()`	The workhorse function underlying `margins()` that assembles the response `"margins"` object for one data frame input.

Table 3: Functions in the **margins** Package

3 Package Functionality

At its core, **margins** offers one function: an S3 generic `margins()` that takes a model object as input and returns a list of data frames of class `"margins"`, which contain the original data, fitted values, standard errors of fitted values, marginal effects for all variables included in the model formula, and variances of those marginal effects. The internals of this function are mostly exported from the package to allow users to calculate just the marginal effects without the other data (using `marginal_effects()`, to calculate the marginal effect of just one variable (using `dydx()`), and to plot and summarize the model and marginal effects in various ways (using `cplot()`, and `plot()`, `persp()`, and `image()` methods). Table 3 provides a full list of exported functions and a brief summary of their behavior.

At present, `margins()` methods exist for objects of class `"lm"`, `"glm"`, and `"loess"`. The `margins.default()` method may work for other object classes, but is untested. The use of the package is meant to be extremely straight forward and to be consistent across model classes. To use it, one needs only specify estimate a model using, for example, `glm()`, and then pass the resulting object to `margins()` to obtain the marginal effects

estimates as "margins" object. For interactive use, the summary.margins() method will be useful:

```
library("datasets")
m <- lm(mpg ~ wt + am + factor(cyl), data = mtcars)
margins(m)

## Average marginal effects
## lm(formula = mpg ~ wt + am + factor(cyl), data = mtcars)

##     wt     am   cyl6   cyl8
##  -3.15 0.1501 -4.257 -6.079

summary(margins(m))

##  factor     AME     SE       z      p   lower   upper
##      am  0.1501 1.3002  0.1154 0.9081 -2.3983  2.6985
##    cyl6 -4.2573 1.4112 -3.0167 0.0026 -7.0233 -1.4913
##    cyl8 -6.0791 1.6837 -3.6105 0.0003 -9.3791 -2.7791
##      wt -3.1496 0.9080 -3.4685 0.0005 -4.9293 -1.3699
```

By default, print.margins() shows the AMEs for each variable. Factor variables are handled automatically by expressing their influence as discrete changes in the outcome moving from the base category to each other category. The summary.margins() output shows the AMEs, along with corresponding standard errors, z-statistics, p-values, and a 95% confidence interval.[9]

To obtain average MERs (an AME with the value of one or more covariates replaced by a theoretically interesting value), users can specify the at option to margins(). In effect, this uses build_datalist() to create a list of data frames to be used as input to marginal_effects() and calculates the marginal effects of every variable separately for each dataset. This can be useful to understand, for example, what the marginal effect of a vehicle's weight is on its fuel efficiency, separately for manual and automatic vehicles, when the relationship between weight and fuel economy is possibly non-linear:

```
m <- lm(mpg ~ wt * am + I(wt^2) * am, data = mtcars)
summary(margins(m, at = list(am = 0:1)))

##  factor     am     AME     SE       z      p    lower   upper
##      am 0.0000 -3.5045 2.8388 -1.2345 0.2170  -9.0684  2.0593
```

[9]The confidence level can be changed by setting level = beta, such as summary(margins(m), level = 0.67) to obtain a 67% CI.

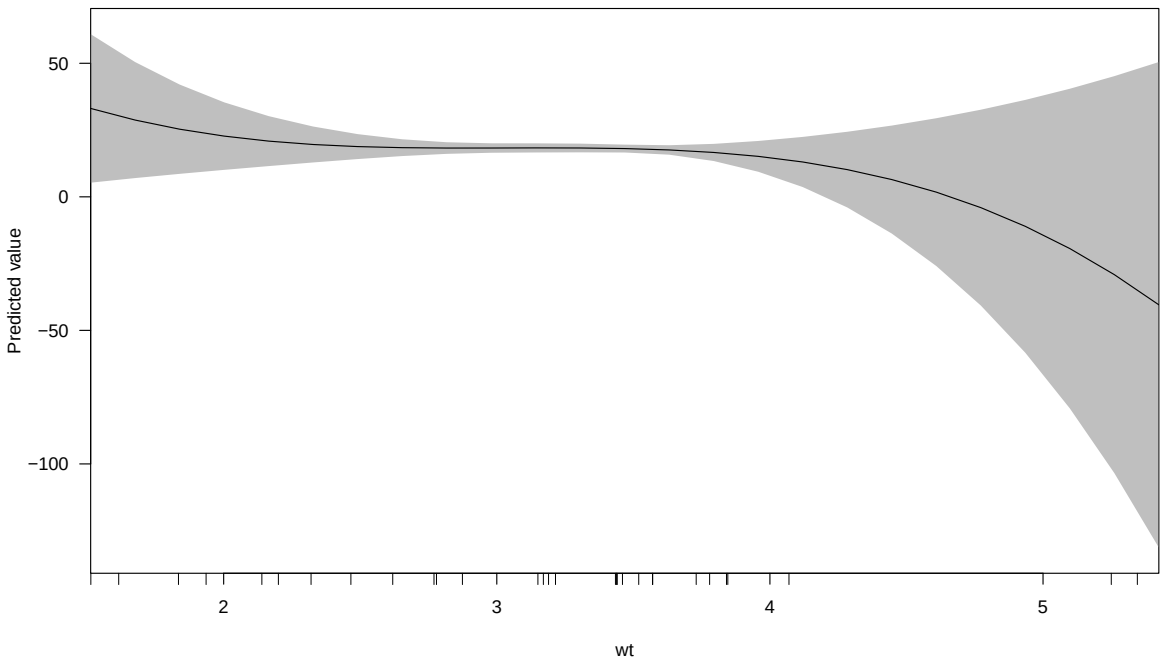
```
##       am 1.0000 -3.5045 2.8382 -1.2348 0.2169  -9.0673  2.0582
##       wt 0.0000 -5.3281 1.8968 -2.8091 0.0050  -9.0457 -1.6105
##       wt 1.0000 -9.6730 3.1983 -3.0244 0.0025 -15.9416 -3.4045
```

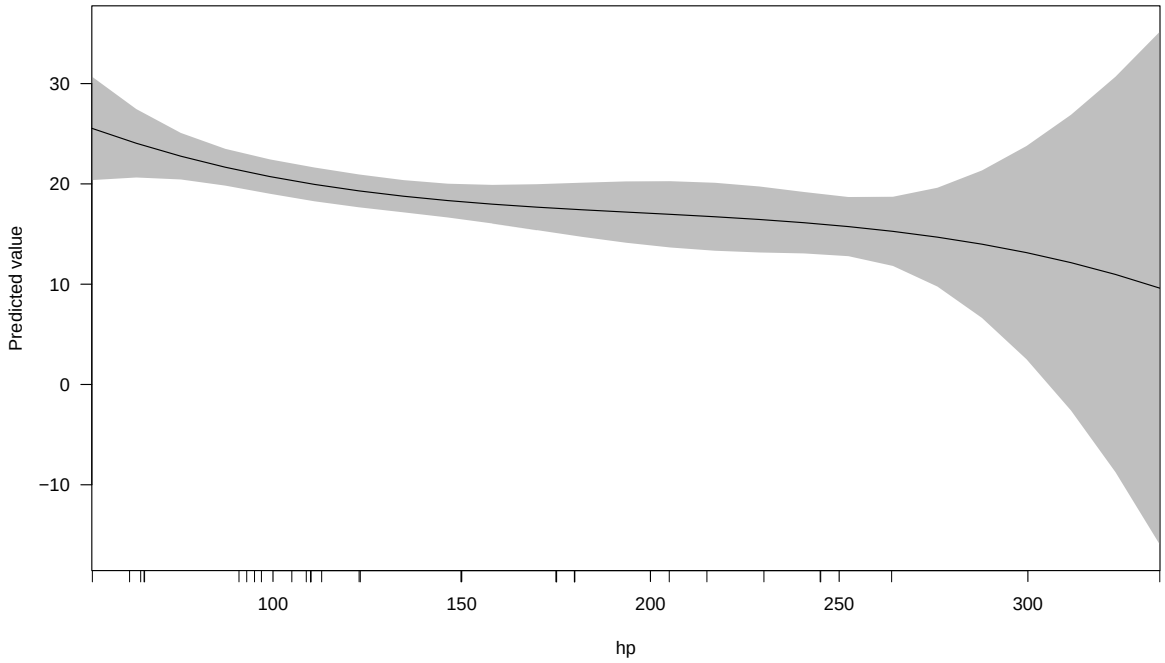
The plotting functionality provided by **margins** can be particularly useful for understanding models and the contribution of each covariate to the outcome. Consider for example, a model involving a simple interaction between two covariates. To understand the effect of each, we can calculate AMEs:

```
m <- lm(mpg ~ wt * I(wt^2) * hp * I(hp^2), data = mtcars)
margins(m)

## Average marginal effects
## lm(formula = mpg ~ wt * I(wt^2) * hp * I(hp^2), data = mtcars)

##       wt       hp
##  -3.936 -0.04343
```

But we can also display the results visually to better understand the results. For example, `cplot()` will show the predicted value of the outcome across levels of covariates:

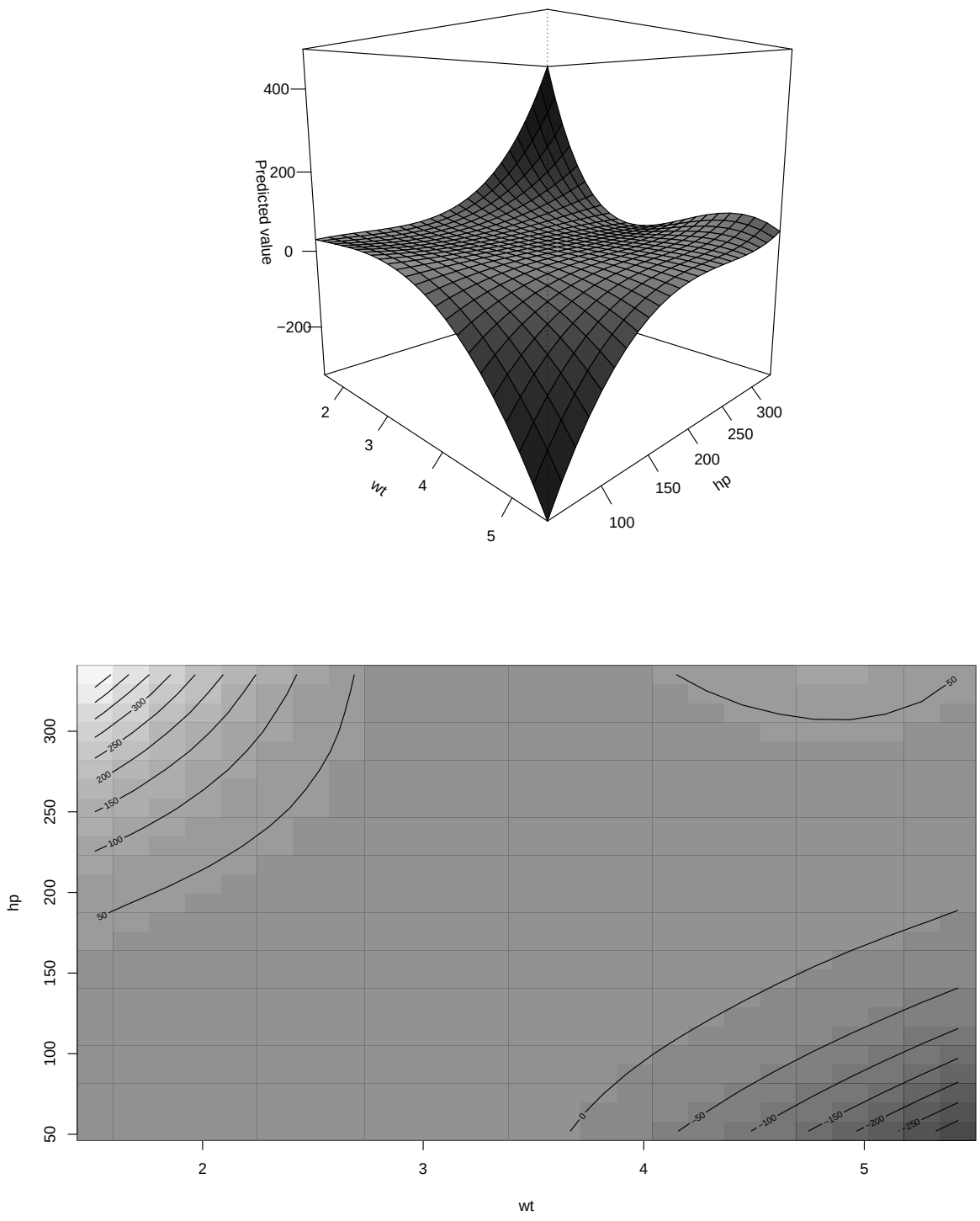
```
cplot(m, "wt")
cplot(m, "hp")
```





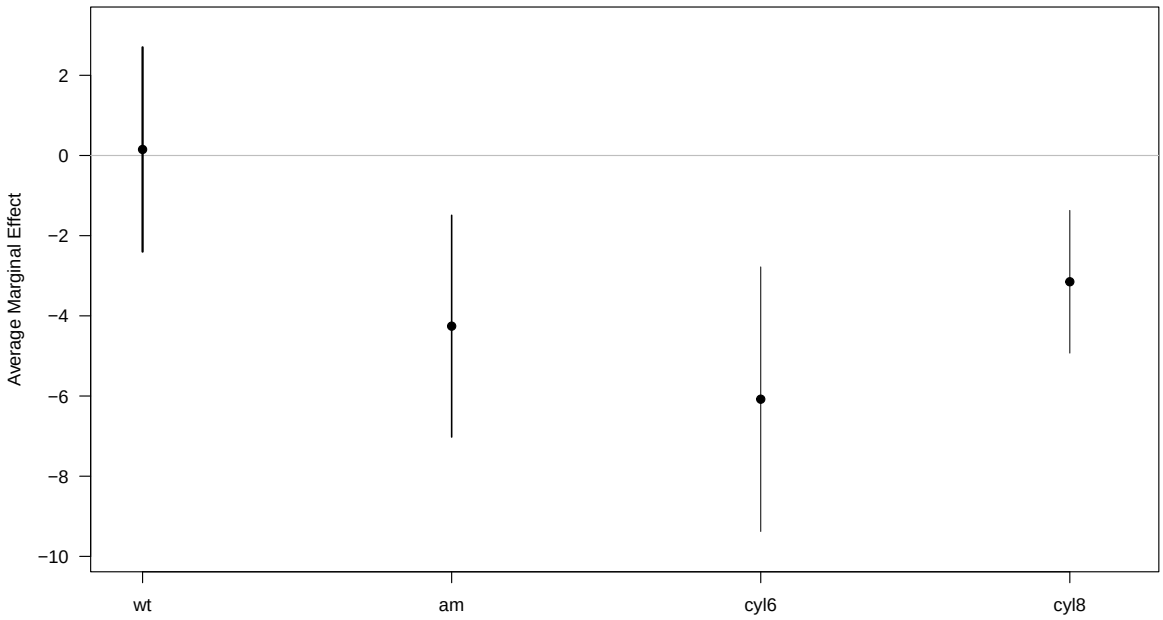
The `persp()` method will show the same but as a three-dimensional surface and the `image()` method will present that information as a two-dimensional "heatmap"-style format:

```
persp(m, "wt", "hp")
image(m, "wt", "hp")
```



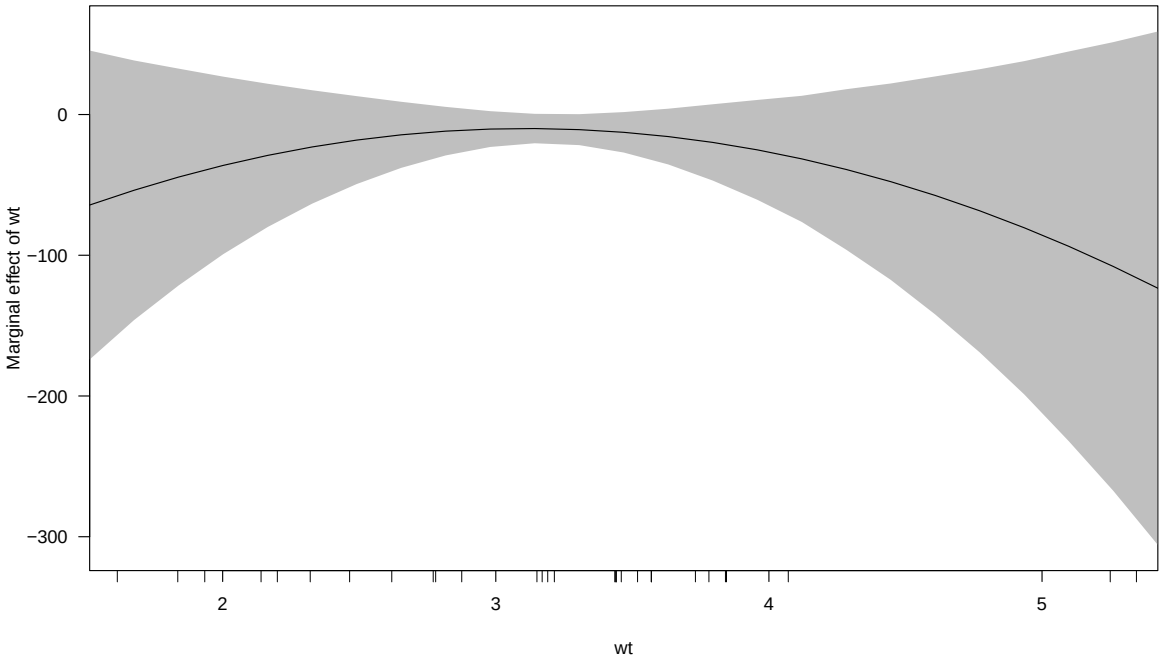
These plots all show fitted values, thereby indirectly communicating marginal effects. To show marginal effects themselves, one can use the `plot.margins()` method to simply show the average marginal effects visually:

```
m <- lm(mpg ~ wt + am + factor(cyl), data = mtcars)
plot(margins(m))
```

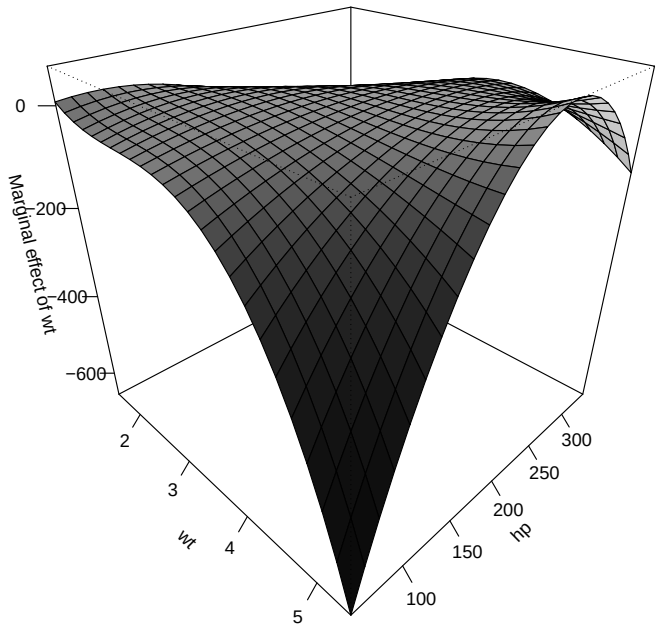


And the `cplot()`, `persp()`, and `image()` methods can be modified to request marginal effects rather than fitted values by setting the `what = "effect"` argument:

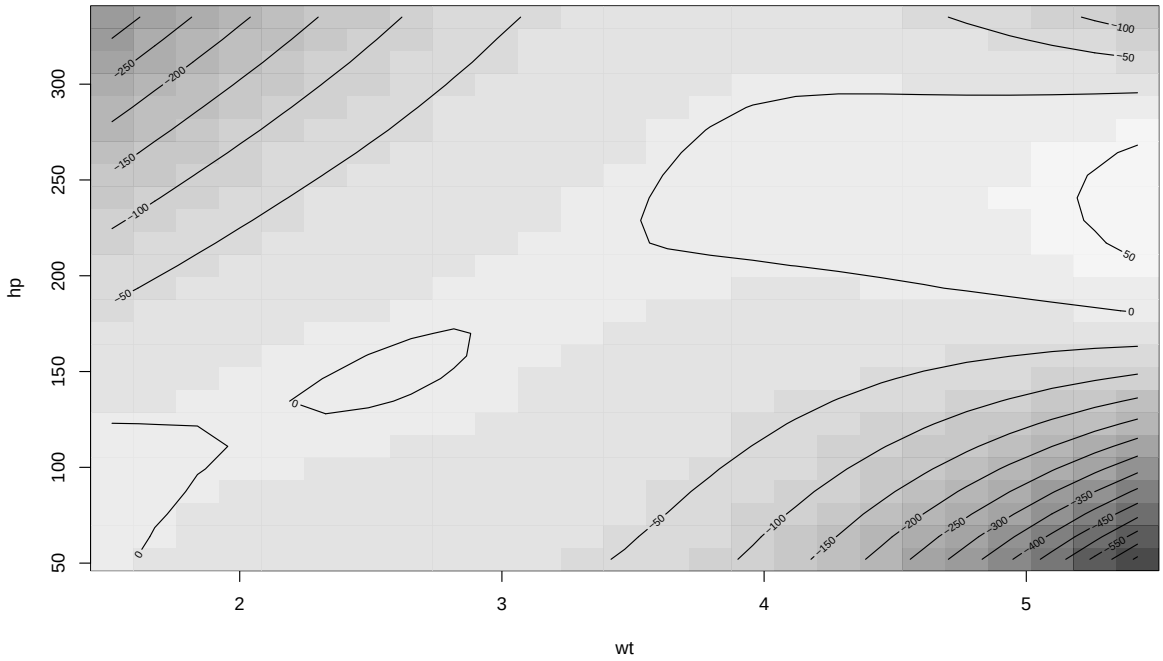
```
m <- lm(mpg ~ wt * I(wt^2) * hp * I(hp^2), data = mtcars)
cplot(m, "wt", what = "effect")
```




```
persp(m, "wt", "hp", what = "effect", phi = 30)
```




```
image(m, "wt", "hp", what = "effect")
```



All of the plotting functions in the package return data structures, which can also be passed to other plotting tools (such as **ggplot2**). The **LinRegInteractive** package provide an alternative set of visualization methods but calculates marginal effects in a quite different way.

3.1 Comparison with Other R Packages

Several existing R packages attempt to estimate quantities of interest, such as marginal effects, but all have limitations. For example, the **car** package noted above is constrained by its use of symbolic derivatives to the calculation of only particular combinations of effects that can be clearly expressed symbolically. This approach appears to also be used by the **effects** package, which provides various functions for describing fitted values and producing plots from those quantities. (Much of that plotting functionality can be achieved through **margin**'s `cplot()` function.) The package is somewhat intransparent in its approach and has an extensive list of explicit limitations about functionality. **lsmeans** can be used to calculate predictive means (or "predictive margins") from linear, generalized linear, and mixed models, as well as calculate "first-difference" style contrasts between tehse predictions (something margins achieves using `dydx()` using alternative `change` arguments).

Other packages show some promise but seem to fail in common situations. For example, the **mfx** package [7] provides several functions for calculating marginal effects from common GLMs (logit, probit, poisson, negative-binomial, beta) but does not properly

account for the interdependence between terms included in multiplicative interactions. For example:

```
> library("mfx")
> library("datasets")
> mfx1 <- glm(vs ~ disp * drat, data = mtcars, family = binomial)
> logitmfx(mfx1, mtcars)
Call:
logitmfx(formula = x, data = mtcars)


Marginal Effects:
                dF/dx   Std. Err.       z  P>|z|
disp      -0.00061817  0.00721983 -0.0856 0.9318
drat      -0.24921775  0.37514481 -0.6643 0.5065
disp:drat -0.00158312  0.00233612 -0.6777 0.4980
```

The **erer** package [16] suffers the same error:

```
> library("erer")
> maBina(x)
> x <- glm(vs ~ disp * drat, data = mtcars, family = binomial, x = TRUE)
> maBina(x)
            effect error t.value p.value
(Intercept)  2.060 1.571   1.311   0.200
disp        -0.001 0.007  -0.086   0.932
drat        -0.249 0.375  -0.664   0.512
disp:drat   -0.002 0.002  -0.678   0.504
```

The **DAMisc** package [2] specifically offers a function, `intEff()` for handling interactions. It, unfortunately, returns an ambiguous value of the "interaction effect" for each observation in a dataset that does not appear to correspond to the marginal effect for either term in the model:

```
> library("DAMisc")
> head(intEff(x, c("disp", "drat"), data = mtcars))
                        int_eff        linear        phat  se_int_eff      zstat
Mazda RX4         -0.0034080006 -3.151623e-03 0.502369027 0.018382454 -0.1853942
Mazda RX4 Wag     -0.0034080006 -3.151623e-03 0.502369027 0.018382454 -0.1853942
Datsun 710        -0.0079522078 -5.750136e-04 0.952093495 0.004907381 -1.6204588
Hornet 4 Drive     0.0183101233 -2.482300e-03 0.269570094 0.021788290  0.8403653
Hornet Sportabout  0.0007426565 -3.380259e-05 0.002688532 0.001359246  0.5463739
Valiant           -0.0167213771 -1.333991e-03 0.879716441 0.008581226 -1.9486000
```

Additionally, the **interflex**, **interplot** [14], and **plotMElm** [9] packages offer visualization functionality specifically for displaying results of models that include multiplicative interaction terms but do not provide general interfaces for calculating marginal effects from arbitrary models. The latter package only handles linear models. **modmarg** is a newer package that appears to provide comparable functionality to margins [17]. **interactionTest** [6] offers a useful `fdrInteraction()` for calculating a t-statistic that limits the false discovery rate for a marginal effect based on an interaction, but only works in the case of two-way interaction terms. Relatedly, **visreg** [5] provides a **lattice**-based visualization function, `visreg()` to visualize predicted values from models in sometimes complex ways. **margins** attempts to emulate some of this in the `cplot()` function. **condvis** and **LinRegInteractive** offer interactive visualization functionality for examining "conditional expectation" plots and surfaces, akin to both `cplot()` and the `persp()` methods offered in **margins**.

3.2 Comparison with Stata

As an aside, Stata relies on symbolic derivatives for calculating marginal effects for OLS models (and numerical derivatives in all other cases). This is made possibly by the fact that Stata offers a much less expressive modelling language that broadly allows only two variable times: continuous (denoted by a `c.` prefix) and factor (denoted by a `i.` prefix), enables simple interactions either explicitly (as `x1#x2`, akin to R's `x1:x2`) or implicitly (as `x1##x2`, akin to R's `x1*x2`), and disallows formula-based transformations (R's `I()` notation). Thus the calculation of marginal effects is simplified by significantly constraining the number of possible models that can be specified and thus the necessary sophistication of a symbolic derivation procedure.

The result is a syntax that can quickly and easily be used to calculate marginal effects for essentially any regression model:

```
. quietly reg pop gdpPercap
. margins, dydx(*)

Average marginal effects                        Number of obs     =      1,704
Model VCE    : OLS

Expression   : Linear prediction, predict()
dy/dx w.r.t. : gdpPercap

------------------------------------------------------------------------------
             |            Delta-method
             |      dy/dx   Std. Err.      t    P>|t|     [95% Conf. Interval]
```

```
-------------+----------------------------------------------------------------
   gdpPercap |  -275.6895   260.9549    -1.06   0.291    -787.5156    236.1366
-------------------------------------------------------------------------------
```

A downside of Stata's limited expressiveness becomes obvious when one considers a variable that is transformed for some specific modelling purpose but for which substantive interpretations are desired on the original scale. For example, one may log transform a covariate but desire to know the marginal effect of that variable (rather than the marginal effect of its logged form). Stata provides no easy means to achieve this; it only allows the pre-calculation of a new variable, breaking the program's ability to recognize a relationship between a variable and its transformed form. **margins**, by contrast, makes this simple. Consider, for example, the two regression models show in Table 4, which are identical except that model (1) is calculated from a pre-transformed value of GDP per capita and model (2) is calculated from a transformation expressed via `I()` notation in a regression formula.[10]

Table 4: Example of Log Transformation

	Dependent variable:	
	Population Size	
	(1)	(2)
loggdp	-4.708^{**}	
	(2.070)	
I(log(gdpPercap))		-4.708^{**}
		(2.070)
Constant	68.012^{***}	68.012^{***}
	(17.083)	(17.083)
Observations	1,704	1,704
R^2	0.003	0.003
Adjusted R^2	0.002	0.002
Residual Std. Error (df = 1702)	106.028	106.028
F Statistic (df = 1; 1702)	5.172^{**}	5.172^{**}
Note:	*p<0.1; **p<0.05; ***p<0.01	

What effect does GDP per capita have on the outcome? Calculating marginal effects reveals the answer. Because we have specified the transformation using `I()` notation in model (2), **margins** can quickly identify the contribution of the original variable:

[10]The models are: (1) `(pop ~loggdp` and (2) `pop ~I(log(gdpPercap))` .

```
summary(margins(m3b))

##     factor     AME     SE       z      p   lower   upper
##  gdpPercap -0.0026 0.0012 -2.2742 0.0230 -0.0049 -0.0004
```

And this result will differ from that for model (2) except in the resulting z-statistic and p-value which — as should be the case — are identical in the two marginal effect calculations:

```
summary(margins(m3a))

##  factor     AME     SE       z      p   lower   upper
##  loggdp -4.7078 2.0701 -2.2742 0.0230 -8.7651 -0.6506
```

Despite some of the underlying limitations, Stata's `margins` command is incredibly user friendly and easy-to-use. Its output is also clean and intuitive. As a result, the behavior of **margins** try (as closely as possible) to mimic the behavior. It does not attempt, however, to provide: (1) an easy way of calculating MEMs (as Stata does with the `, atmeans` option), (2) calculating of predicted values (since R already provides this via `predict()`), or (3) cover the full class of model types that Stata currently supports. One other key advantage of the R implementation is that because it relies on a fully functional programming paradigm, marginal effects can easily be calculated for multiple objects, whereas Stata's approach can only calculate effects for the *previous* modelling command using stored results.

4 Conclusion

Average marginal effects offer an intuitive technique for interpreting regression estimates from a wide class of linear and generalized linear models. While Stata has offered a simple and general computational interface for extracting these quantities of interest from regression models for many years, the approach has not been widely available in other statistical software. The **margins** port to R makes this functionality much more widely available. By describing the computational approach used by both packages, this article offers users of other languages guidelines for how to apply the approach elsewhere while offering applied data analysts a straightforward explanation of the marginal effect quantity and its derivation.

At present, **margins** estimates quantities of interest for a wide array of model formulae used in least squares regression and many common generalized linear models. Stata's `margins` and Zelig/Clarify produce quantities of interest for a wider array of model types. Extension of **margins** to other model types is planned for the future. The creation of

the core `margins` function as an S3 generic means that the package is easily extensible to other model types (e.g., those introduced in other user-created packages). Development of **margins** is handled on GitHub, allowing for easy contribution of bug fixes, enhancements, and additional model-specific methods. By publishing **margins** as free and open-source software (FOSS), it should be straightforward for users of other languages (Python, julia, etc.) to implement similar functionality. Indeed, the port of closed source statistical software to open source represents an underappreciated by critical step in making FOSS data analysis more accessible to those used to working with closed source products.

For applied data analysis, the most important feature of **margins** is its intuitive use and the near-direct translation of Stata code into R. For those used to Stata's `margins` command, R's **margins** package should be a painless transition. For R users not accustomed to calculating marginal effects, **margins** should also offer a straightforward and tidy way of calculating predicted values and marginal effects, and displaying the results thereof.

References

[1] Chunrong Ai and Edward C. Norton. Interaction terms in logit and probit models. 80:123–129, 2003.

[2] Dave Armstrong. Damisc: Dave armstrong's miscellaneous functions, 2016.

[3] William D. Berry, Matt Golder, and Daniel Milton. Improving tests of theories positing interaction. *The Journal of Politics*, 74(03):653–671, March 2012.

[4] Thomas Brambor, William Roberts Clark, and Matt Golder. Understanding interaction models: Improving empirical analyses. *Political Analysis*, 14(1):63–82, May 2005.

[5] Patrick Breheny and Woodrow Burchett. visreg: Visualization of regression models. Available at The Comprehensive R Archive Network (CRAN), 2016.

[6] Justin Esarey and Jane Lawrence Sumner. interactionTest: Calculates critical test statistics to control false discovery and familywise error rates in marginal effects plots. Available at The Comprehensive R Archive Network (CRAN), 2015.

[7] Alan Fernihough. mfx: Marginal effects, odds ratios and incidence rate ratios for glms. Available at The Comprehensive R Archive Network (CRAN), 2014.

[8] John Fox and Sanford Weisberg. *An R Companion to Applied Regression*. Sage, Thousand Oaks CA, second edition, 2011.

[9] Christopher Gandrud. plotMElm: Plot marginal effects from linear models. Available at The Comprehensive R Archive Network (CRAN), 2016.

[10] Gary King, Michael Tomz, and Jason Wittenberg. Making the most of statistical analyses: Improving interpretation and presentation. *American Journal of Political Science*, 44(2):347–361, April 2000.

[11] Thomas J. Leeper. margins: An r port of stata's 'margins' command. Available at The Comprehensive R Archive Network (CRAN), 2016.

[12] J. Scott Long. *Regression Models for Categorical and Limited Dependent Variables*. Sage Publications, Inc, 1997.

[13] Edward C. Norton, Hua Wang, and Chunrong Ai. Computing interaction effects and standard errors in logit and probit models. 4(2):154–167, 2004.

[14] Frederick Solt and Yue Hu. interplot: Plot the effects of variables in interaction terms. Available at The Comprehensive R Archive Network (CRAN), 2015.

[15] Inc. StataCorp. Stata statistical software: Release 11, 2009.

[16] Changyou Sun. erer: Empirical research in economics with r. Available at The Comprehensive R Archive Network (CRAN), 2014.

[17] Annie Wang. modmarg: Calculating marginal effects and levels with errors. Available at The Comprehensive R Archive Network (CRAN), 2017.

[18] G.N. Wilkinson and C.E. Rogers. Symbolic description of factorial models for analysis of variance. 22(3):392–99, 1973.

[19] Richard Williams. Using the margins command to estimate and interpret adjusted predictions and marginal effects. 12:308–331, 2012.