Package 'marginaleffects'

January 9, 2022

Title Marginal Effects, Marginal Means, Predictions, and Contrasts

Version 0.3.1

Description Compute, summarize, and plot marginal effects, adjusted predictions, contrasts, and marginal means for a wide variety of models.

License GPL (>= 3)

Copyright inst/COPYRIGHTS

Encoding UTF-8

URL https://vincentarelbundock.github.io/marginaleffects/
https://github.com/vincentarelbundock/marginaleffects

BugReports https://github.com/vincentarelbundock/marginaleffects/issues

RoxygenNote 7.1.2

VignetteBuilder knitr

Config/testthat/edition 3

Config/testthat/parallel true

Depends R (>= 3.5.0)

Imports checkmate, generics, insight (>= 0.14.5), methods, numDeriv

Suggests bench, brms, broom, data.table, dplyr, emmeans, ggdist, ggplot2, ggbeeswarm, gt, haven, kableExtra, knitr, magrittr, margins, modelsummary, patchwork, rlang, rmarkdown, testthat (>= 3.0.0), tidyverse, vdiffr, AER, aod, betareg, bife, brglm2, crch, estimatr, fixest (>= 0.10.1), gam, geepack, glmx, ivreg, lme4, MASS, mgcv, nlme, nnet, ordinal, plm, pscl, quantreg, rms, robustbase, rstantools, rstanarm, sandwich, speedglm, survey, survival, truncreg, covr, spelling

Collate 'complete_levels.R' 'datagrid.R' 'get_coef.R' 'get_contrasts.R' 'get_dydx.R' 'get_group_names.R' 'get_hdi.R' 'get_predict.R' 'get_vcov.R' 'marginaleffects.R' 'marginalmeans.R' 'mean_or_mode.R' 'set_coef.R' 'methods_MASS.R' 'methods_aod.R' 'methods_betareg.R' 'methods_bife.R' 'sanity_model.R' 'methods_nnet.R'

'methods_brglm2.R' 'methods_brms.R' 'methods_crch.R'
'methods_glmx.R' 'methods_lme4.R' 'methods_mgcv.R'
'methods_ordinal.R' 'methods_plm.R' 'methods_pscl.R'
'methods_quantreg.R' 'methods_rstanarm.R' 'methods_stats.R'
'methods_survival.R' 'package.R' 'plot.R' 'plot_cap.R'
'plot_cme.R' 'poorman.R' 'posteriordraws.R' 'predictions.R'
'sanity.R' 'sanity_type.R' 'standard_errors_delta.R'
'summary.R' 'tidy.R' 'type_dictionary.R' 'utils.R'

Language en-US

NeedsCompilation no

Author Vincent Arel-Bundock [aut, cre, cph]
(<https://orcid.org/0000-0003-2042-7063>)

Maintainer Vincent Arel-Bundock <vincent.arel-bundock@umontreal.ca>

Repository CRAN

Date/Publication 2022-01-09 22:52:41 UTC

R topics documented:

`counterfactual` *Superseded by datagrid(..., grid.type = "counterfactual")*

Description

Superseded by datagrid(..., grid.type = "counterfactual")

Usage

```
counterfactual(..., model = NULL, newdata = NULL)
```

Arguments

...	named arguments with vectors of values for user-specified variables. The output will include all combinations of these variables (see Examples below.)
model	Model object
newdata	data.frame (one and only one of the model and newdata arguments

---

datagrid *Generate a data grid of "typical," "counterfactual," or user-specified values for use in the* newdata *argument of the* marginaleffects *or* predictions *functions.*

Description

Generate a data grid of "typical," "counterfactual," or user-specified values for use in the newdata argument of the marginaleffects or predictions functions.

Usage

```
datagrid(
  ...,
  model = NULL,
  newdata = NULL,
  grid.type = "typical",
  FUN.character = Mode,
  FUN.factor = Mode,
  FUN.logical = Mode,
  FUN.numeric = function(x) mean(x, na.rm = TRUE),
  FUN.other = function(x) mean(x, na.rm = TRUE)
)
```

Arguments

...	named arguments with vectors of values for user-specified variables. The output will include all combinations of these variables (see Examples below.)
model	Model object
newdata	data.frame (one and only one of the model and newdata arguments
grid.type	character • "typical": variables whose values are not explicitly specified by the user in ... are set to the output of the functions supplied to the FUN.type arguments.

- "counterfactual": the entire dataset is duplicated for each combination of the variable values specified in ....

`FUN.character` the function to be applied to character variables.

`FUN.factor` the function to be applied to factor variables.

`FUN.logical` the function to be applied to factor variables.

`FUN.numeric` the function to be applied to numeric variables.

`FUN.other` the function to be applied to other variable types.

Details

If `datagrid` is used in a `marginaleffects` or `predictions` call as the `newdata` argument, users do not need to specify the `model` or `newdata` argument. The data is extracted automatically from the model.

If users supply a model, the data used to fit that model is retrieved using the `insight::get_data` function.

Value

A `data.frame` in which each row corresponds to one combination of the named predictors supplied by the user via the `...` dots. Variables which are not explicitly defined are held at their mean or mode.

Examples

```
# The output only has 2 rows, and all the variables except `hp` are at their
# mean or mode.
datagrid(newdata = mtcars, hp = c(100, 110))

# We get the same result by feeding a model instead of a data.frame
mod <- lm(mpg ~ hp, mtcars)
datagrid(model = mod, hp = c(100, 110))

# Use in `marginaleffects` to compute "Typical Marginal Effects". When used
# in `marginaleffects()` or `predictions()` we do not need to specify the
#`model` or `newdata` arguments.
marginaleffects(mod, newdata = datagrid(hp = c(100, 110)))

# The full dataset is duplicated with each observation given counterfactual
# values of 100 and 110 for the `hp` variable. The original `mtcars` includes
# 32 rows, so the resulting dataset includes 64 rows.
dg <- datagrid(newdata = mtcars, hp = c(100, 110), grid.type = "counterfactual")
nrow(dg)

# We get the same result by feeding a model instead of a data.frame
mod <- lm(mpg ~ hp, mtcars)
dg <- datagrid(model = mod, hp = c(100, 110), grid.type = "counterfactual")
nrow(dg)
```

get_marginalmeans *Workhorse function for* marginalmeans

Description

Needs to be separate because we also need it in delta_method

Usage

```
get_marginalmeans(model, newdata, type, variables, ...)
```

Arguments

model	Model object
newdata	A dataset over which to compute adjusted predictions. NULL uses the original data used to fit the model.
type	Type(s) of prediction (string or character vector). This can differ based on the model type, but will typically be a string such as: "response", "link", "probs", or "zero".
variables	Categorical predictors over which to compute marginal means (character vector). NULL calculates marginal means for all logical, character, or factor variables in the dataset used to fit model.
...	absorb useless arguments from other get_* workhorse functions

marginaleffects *Marginal effects using numerical derivatives*

Description

A "marginal effect" is the partial derivative of the regression equation with respect to a variable in the model. This function uses automatic differentiation to compute marginal effects for a vast array of models, including non-linear models with transformations (e.g., polynomials). The list of supported models and of models whose numerical results have been validated against external software (Stata, margins, and/or emmeans) is reported on the package website: https://vincentarelbundock.github.io/marginaleffects/

Usage

```
marginaleffects(
  model,
  newdata = NULL,
  variables = NULL,
  vcov = TRUE,
  type = "response",
  ...
)
```

Arguments

model	Model object
newdata	A dataset over which to compute marginal effects. NULL uses the original data used to fit the model.
variables	Variables to consider (character vector). NULL calculates marginal effects for all terms in the model object.
vcov	Matrix or boolean • FALSE: does not compute unit-level standard errors. • TRUE: computes unit-level standard errors using the default vcov(model) variance-covariance matrix. • Named square matrix: computes standard errors with a user-supplied variance-covariance matrix. This matrix must be square and have dimensions equal to the number of coefficients in get_coef(model).
type	Type(s) of prediction as string or vector This can differ based on the model type, but will typically be a string such as: "response", "link", "probs", or "zero".
...	Additional arguments are pushed forward to predict().

Value

A data.frame with one row per observation (per term/group) and several columns:

- rowid: row number of the newdata data frame
- type: prediction type, as defined by the type argument
- group: (optional) value of the grouped outcome (e.g., categorical outcome models)
- term: the variable whose marginal effect is computed
- dydx: marginal effect of the term on the outcome for a given combination of regressor values
- std.error: standard errors computed by via the delta method.

Examples

```
mod <- glm(am ~ hp * wt, data = mtcars, family = binomial)
mfx <- marginaleffects(mod)
summary(mfx)
tidy(mfx)
head(mfx)
plot(mfx)

# typical marginal effects
marginaleffects(mod, newdata = datagrid(hp = c(100, 110)))

# counterfactual average marginal effects
marginaleffects(mod, newdata = datagrid(hp = c(100, 110), grid.type = "counterfactual"))

# heteroskedasticity robust standard errors
marginaleffects(mod, vcov = sandwich::vcovHC(mod))
```

`marginalmeans` *Marginal Means*

Description

Compute estimated marginal means for specified factors.

Usage

```
marginalmeans(
  model,
  variables = NULL,
  variables_grid = NULL,
  vcov = NULL,
  type = "response"
)
```

Arguments

`model`	Model object
`variables`	Categorical predictors over which to compute marginal means (character vector). `NULL` calculates marginal means for all logical, character, or factor variables in the dataset used to fit `model`.
`variables_grid`	Categorical predictors used to construct the prediction grid over which adjusted predictions are averaged (character vector). `NULL` creates a grid with all combinations of all categorical predictors. This grid can be very large when there are many variables and many response levels, so it is advisable to select a limited number of variables in the `variables` and `variables_grid` arguments.
`vcov`	Matrix or boolean • FALSE: does not compute unit-level standard errors. • TRUE: computes unit-level standard errors using the default `vcov(model)` variance-covariance matrix. • Named square matrix: computes standard errors with a user-supplied variance-covariance matrix. This matrix must be square and have dimensions equal to the number of coefficients in `get_coef(model)`.
`type`	Type(s) of prediction (string or character vector). This can differ based on the model type, but will typically be a string such as: "response", "link", "probs", or "zero".

Details

This function begins by calling the `predictions` function to obtain a grid of predictors, and adjusted predictions for each cell. The grid includes all combinations of the categorical variables listed in the `variables` and `variables_grid` arguments, or all combinations of the categorical variables used to fit the model if `variables_grid` is `NULL`. In the prediction grid, numeric variables are held at their means.

After constructing the grid and filling the grid with adjusted predictions, `marginalmeans` computes marginal means for the variables listed in the `variables` argument, by average across all categories in the grid.

`marginalmeans` can only compute standard errors for linear models, or for predictions on the link scale, that is, with the `type` argument set to "link".

The `marginaleffects` website compares the output of this function to the popular `emmeans` package, which provides similar but more advanced functionality: https://vincentarelbundock.github.io/marginaleffects/

Value

Data frame of marginal means with one row per variable-value combination.

Examples

```
library(marginaleffects)

# Convert numeric variables to categorical before fitting the model
dat <- mtcars
dat$cyl <- as.factor(dat$cyl)
dat$am <- as.logical(dat$am)
mod <- lm(mpg ~ hp + cyl + am, data = dat)

# Compute and summarize marginal means
mm <- marginalmeans(mod)
summary(mm)
```

---

plot.marginaleffects *Point-range plot of average marginal effects*

Description

Uses the `ggplot2` package to draw a point-range plot of the average marginal effects computed by `tidy`.

Usage

```
## S3 method for class 'marginaleffects'
plot(x, conf.int = TRUE, conf.level = 0.95, ...)
```

Arguments

`x`	An object produced by the `marginaleffects` function.
`conf.int`	Logical indicating whether or not to include a confidence interval.
`conf.level`	The confidence level to use for the confidence interval if `conf.int=TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
`...`	Additional arguments are pushed forward to `predict()`.

Details

The `tidy` function calculates average marginal effects by taking the mean of all the unit-level marginal effects computed by the `marginaleffects` function.

Value

A `ggplot2` object

Examples

```
mod <- glm(am ~ hp + wt, data = mtcars)
mfx <- marginaleffects(mod)
plot(mfx)
```

---

plot_cap *Plot Conditional Adjusted Predictions*

Description

This function plots the adjusted predictions of the outcome (y-axis) against values of one or more predictors.

Usage

```
plot_cap(
  model,
  condition,
  type = "response",
  conf.int = TRUE,
  conf.level = 0.95,
  draw = TRUE
)
```

Arguments

`model`	Model object
`condition`	String or vector of two strings. The first is a variable name to be displayed on the x-axis. The second is a variable whose values will be displayed in different colors.
`type`	Type(s) of prediction as string or vector This can differ based on the model type, but will typically be a string such as: "response", "link", "probs", or "zero".
`conf.int`	Logical indicating whether or not to include a confidence interval.
`conf.level`	The confidence level to use for the confidence interval if `conf.int=TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
`draw`	`TRUE` returns a `ggplot2` plot. `FALSE` returns a `data.frame` of the underlying data.

Value

A ggplot2 object

Examples

```
mod <- lm(mpg ~ hp + wt, data = mtcars)
plot_cap(mod, condition = "wt")

mod <- lm(mpg ~ hp * wt * am, data = mtcars)
plot_cap(mod, condition = c("hp", "wt"))
```

---

plot_cme *Plot Conditional Marginal Effects*

Description

In models where two continuous variables are interacted, the marginal effect of one variable is conditional on the value of the other variable. This function draws a plot of the marginal effect of the `effect` variable for different values of the `condition` variable.

Usage

```
plot_cme(
  model,
  effect,
  condition,
  type = "response",
  conf.int = TRUE,
  conf.level = 0.95,
  draw = TRUE
)
```

Arguments

`model`	Model object
`effect`	Name of the variable whose marginal effect we want to plot on the y-axis
`condition`	String or vector of two strings. The first is a variable name to be displayed on the x-axis. The second is a variable whose values will be displayed in different colors.
`type`	Type(s) of prediction as string or vector This can differ based on the model type, but will typically be a string such as: "response", "link", "probs", or "zero".
`conf.int`	Logical indicating whether or not to include a confidence interval.
`conf.level`	The confidence level to use for the confidence interval if `conf.int=TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
`draw`	`TRUE` returns a `ggplot2` plot. `FALSE` returns a `data.frame` of the underlying data.

Value

A ggplot2 object

Examples

```
mod <- lm(mpg ~ hp * wt, data = mtcars)
plot_cme(mod, effect = "hp", condition = "wt")

mod <- lm(mpg ~ hp * wt * am, data = mtcars)
plot_cme(mod, effect = "hp", condition = c("wt", "am"))
```

---

posteriordraws *Extract posterior draws from a* predictions *or* marginaleffects *object derived from Bayesian* brms *models*

Description

Extract posterior draws from a predictions or marginaleffects object derived from Bayesian brms models

Usage

```
posteriordraws(x)
```

Arguments

x	An object produced by the predictions or the marginaleffects functions

Value

A data.frame with drawid and draw columns.

---

predictions *Adjusted Predictions*

Description

Compute model-adjusted predictions (fitted values) for a "grid" of regressor values.

Usage

```
predictions(
  model,
  variables = NULL,
  newdata = NULL,
  conf.level = 0.95,
  type = "response",
  ...
)
```

Arguments

model	Model object
variables	Character vector. Compute Adjusted Predictions for combinations of each of these variables. Factor levels are considered at each of their levels. Numeric variables variables are considered at Tukey's Five-Number Summaries. NULL uses the original data used to fit the model.
newdata	A dataset over which to compute adjusted predictions. NULL uses the original data used to fit the model.
conf.level	The confidence level to use for the confidence interval. No interval is computed if conf.int=NULL. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
type	Type(s) of prediction as string or vector This can differ based on the model type, but will typically be a string such as: "response", "link", "probs", or "zero".
...	Additional arguments are pushed forward to predict().

Value

A data.frame with one row per observation and several columns:

- rowid: row number of the newdata data frame
- type: prediction type, as defined by the type argument
- group: (optional) value of the grouped outcome (e.g., categorical outcome models)
- predicted: predicted outcome
- std.error: standard errors computed by the insight::get_predicted function or, if unavailable, via marginaleffects delta method functionality.
- conf.low: lower bound of the confidence or highest density interval (for bayesian models)
- conf.high: upper bound of the confidence or highest density interval (for bayesian models)

Examples

```
# Predicted outcomes for every row of the original dataset
mod <- lm(mpg ~ hp + factor(cyl), data = mtcars)
pred <- predictions(mod)
head(pred)
```

```
# Predicted outcomes for user-specified values of the regressors
predictions(mod, newdata = datagrid(hp = c(100, 120), cyl = 4))

# Plot of predicted outcomes for different values of the regressor
plot_cap(mod, condition = "hp")
```

`sanity_type` *check type sanity*

Description

check type sanity

Usage

```
sanity_type(model, type)
```

Arguments

`model`	model object
`type`	character vector

Value

Named vector where value is the Base R `type` and name is the insight::get_predicted `predict`

`summary.marginaleffects` *Summarize a* `marginaleffects` *object*

Description

Summarize a `marginaleffects` object

Usage

```
## S3 method for class 'marginaleffects'
summary(object, conf.level = 0.95, ...)
```

Arguments

`object`	An object produced by the `marginaleffects` function
`conf.level`	The confidence level to use for the confidence interval if `conf.int=TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
`...`	Additional arguments are pushed forward to `predict()`.

Value

Data frame of summary statistics for an object produced by the marginaleffects function

---

summary.marginalmeans *Summarize a* marginalmeans *object*

Description

Summarize a marginalmeans object

Usage

```
## S3 method for class 'marginalmeans'
summary(object, conf.level = 0.95, ...)
```

Arguments

object	An object produced by the marginalmeans function
conf.level	The confidence level to use for the confidence interval if conf.int=TRUE. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
...	Additional arguments are pushed forward to predict().

Value

Data frame of summary statistics for an object produced by the marginalmeans function

---

tidy.marginaleffects *Tidy a* marginaleffects *object*

Description

Tidy a marginaleffects object

Usage

```
## S3 method for class 'marginaleffects'
tidy(x, conf.int = TRUE, conf.level = 0.95, ...)
```

Arguments

x	An object produced by the marginaleffects function.
conf.int	Logical indicating whether or not to include a confidence interval.
conf.level	The confidence level to use for the confidence interval if conf.int=TRUE. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
...	Additional arguments are pushed forward to predict().

Details

The tidy function calculates average marginal effects by taking the mean of all the unit-level marginal effects computed by the marginaleffects function.

Value

A "tidy" data.frame of summary statistics which conforms to the broom package specification.

Examples

```
mod <- lm(mpg ~ hp * wt + factor(gear), data = mtcars)
mfx <- marginaleffects(mod)
tidy(mfx)
```

---

tidy.marginalmeans *Tidy a* marginalmeans *object*

Description

Tidy a marginalmeans object

Usage

```
## S3 method for class 'marginalmeans'
tidy(x, conf.int = TRUE, conf.level = 0.95, ...)
```

Arguments

x	An object produced by the marginalmeans function.
conf.int	Logical indicating whether or not to include a confidence interval.
conf.level	The confidence level to use for the confidence interval if conf.int=TRUE. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
...	Additional arguments are pushed forward to predict().

Value

A "tidy" data.frame of summary statistics which conforms to the broom package specification.

type_dictionary *type dictionary*

Description

insight::get_predict accepts a `predict` argument stats::predict accepts a `type` argument this dictionary converts

Usage

```
type_dictionary
```

Format

An object of class `data.frame` with 88 rows and 5 columns.

typical *Superseded by datagrid(...)*

Description

Superseded by datagrid(...)

Usage

```
typical(
  ...,
  model = NULL,
  newdata = NULL,
  FUN.character = Mode,
  FUN.factor = Mode,
  FUN.logical = Mode,
  FUN.numeric = function(x) mean(x, na.rm = TRUE),
  FUN.other = function(x) mean(x, na.rm = TRUE)
)
```

Arguments

`...`	named arguments with vectors of values for user-specified variables. The output will include all combinations of these variables (see Examples below.)
`model`	Model object
`newdata`	data.frame (one and only one of the `model` and `newdata` arguments
`FUN.character`	the function to be applied to character variables.
`FUN.factor`	the function to be applied to factor variables.

FUN.logical	the function to be applied to factor variables.
FUN.numeric	the function to be applied to numeric variables.
FUN.other	the function to be applied to other variable types.

Index