Package 'modmarg'

February 1, 2018

Title Calculating Marginal Effects and Levels with Errors

Version 0.9.2

Description Calculate predicted levels and marginal effects,
using the delta method to calculate standard errors. This is an R-based
version of the 'margins' command from Stata.

URL https://github.com/anniejw6/modmarg

BugReports https://github.com/anniejw6/modmarg/issues

Depends R (>= 2.10.0)

License GPL-3

Encoding UTF-8

LazyData true

Suggests knitr, rmarkdown, testthat, sandwich, AER

VignetteBuilder knitr

RoxygenNote 6.0.1

NeedsCompilation no

Author Alex Gold [aut],
Nat Olin [aut],
Annie Wang [aut, cre]

Maintainer Annie Wang <anniejw6@gmail.com>

Repository CRAN

Date/Publication 2018-02-01 16:56:21 UTC

R topics documented:

cvcov *Clustered variance-covariance matrices and T statistic d.o.f.*

Description

Variance-covariance matrices with robust clustered standard errors and degrees-of-freedom for T statistics, for tests and examples specifying `vcov` (d.o.f. defined as g - 1, where g is the number of clusters). Generated with `margex` data in this package.

Usage

```
cvcov
```

Format

A list of three lists, from an OLS model, logit model, and OLS with a polynomial interaction with missing data, each containing

clust 3-by-3 variance-covariance matrix

dof integer, degrees of freedom for the T statistic

Details

See data-raw/make_cluster_vcov.R for details.

Source

http://cameron.econ.ucdavis.edu/research/Cameron_Miller_JHR_2015_February.pdf

marg *Estimating predictive margins on a model*

Description

This function estimates the predictive effects and levels for variables within a model using the delta method.

Usage

```
marg(mod, var_interest, data = NULL, weights = NULL, vcov_mat = NULL,
  dof = NULL, type = "levels", base_rn = 1, at_var_interest = NULL,
  at = NULL, cofint = 0.95, ...)
```

Arguments

mod	model object, currently only support those of class glm or ivreg
var_interest	name of the variable of interest, must correspond to a covariate in the model
data	data.frame that margins should run over, defaults changes based on class-specific method
weights	numeric, vector of weights used to generate predicted levels, defaults changes based on class-specific method. Must be equal to the number of rows in data.
vcov_mat	the variance-covariance matrix, defaults changes based on class-specific method
dof	integer, the degrees of freedom used for the T statistic in an OLS model, defaults changes based on class-specific method
type	either 'levels' (predicted outcomes) or 'effects' $dydx$, defaults to 'levels'
base_rn	numeric, if type == 'effects', the base level (taken as the index of one of the ordered unique values in var_interest). if type == 'levels', this parameter is ignored. Defaults to 1.
at_var_interest	vector, if type == 'levels', the values for the variable of interest at which levels should be calculated. If NULL, indicates all levels for a factor variable, defaults to NULL
at	list, should be in the format of list('var_name' = c(values)), defaults to NULL. This calculates the margins of the variable at these particular variables. If all values are needed, suggested syntax is at = list('var' = unique(df$var)).
cofint	numeric, confidence interval (must be less than 1), defaults to 0.95
...	additional parameters passed to class-specific methods

Details

The variable for the predictive margin is specified by var_interest. If margins are only needed at particular values of var_interest, at_var_interest should be used. If margins of var_interest are needed at across the levels of a *different* variable in the model, at should be used.

If higher-order polynomial terms (e.g. $y\ x+x^2$) are added using the R function poly, the raw = TRUE argument should be used to include the basic polynomial terms instead of orthogonal polynomial terms. If orthogonal polynomials are used, marg will fail when the user specifies at for a small set of values for the variable in question (e.g. at = list(x = 10)), since poly needs more data to calculate orthogonal polynomials (e.g. poly(10, 2) fails, but poly(c(10, 8, 3), 2) will run).

P values are calculated with T tests for gaussian families, and Z tests otherwise. If a new variance-covariance matrix is provided (e.g. for clustering standard errors), the degrees of freedom for the T test / p-value calculation may need to be specified using dof.

This function currently only supports glm and ivreg objects. If you would like to use lm objects, consider running a glm with family gaussian.

When calculating predicted levels and effects for models built using weights, marg returns weighted averages for levels and effects by default. Users can remove this option by setting weights = NULL.

Value

list of dataframes with predicted margins/effects, standard errors, p-values, and confidence interval bounds

---

`marg.glm` *Predicted Margins for 'glm' objects*

Description

Obtains predicted margins and standard errors of those predictions from a fitted generalized linear model object.

Usage

```
## S3 method for class 'glm'
marg(mod, var_interest,
  data = mod$data[names(mod$prior.weights), ], weights = mod$prior.weights,
  ...)
```

Arguments

`mod`	model object, currently only support those of class `glm` or `ivreg`
`var_interest`	name of the variable of interest, must correspond to a covariate in the model
`data`	data.frame that margins should run over, defaults changes based on class-specific method
`weights`	numeric, vector of weights used to generate predicted levels, defaults changes based on class-specific method. Must be equal to the number of rows in `data`.
`...`	additional parameters passed to ?`marg`.

Examples

```
data(mtcars)
mod <- glm(vs ~ as.factor(gear) + mpg, data = mtcars, family = 'binomial')

# Get the level of the outcome variable at different values of `gear`
marg(mod, var_interest = 'gear', type = 'levels')
# Get the effect of `gear` on the outcome value, holding values of `mpg`
# constant
marg(mod, var_interest = 'gear', type = 'effects',
     at = list(mpg = c(15, 21)))

data(margex)
mod <- glm(outcome ~ as.factor(treatment) + distance,
       data = margex, family = 'binomial')
# Get the level of the outcome variable at different values of `treatment`
marg(mod, var_interest = 'treatment', type = 'levels', at = NULL)
# Get the effect of `treatment` on the outcome variable
```

```
marg(mod, var_interest = 'treatment', type = 'effects', at = NULL)
# Get the level of the outcome variable at different values of `distance`
marg(mod, var_interest = 'distance', type = 'levels',
          at = NULL, at_var_interest = c(10, 20, 30))

# Using a custom variance-covariance matrix for clustered standard errors
# (also requires custom degrees of freedom for T statistic with OLS model),
# clustering on the "arm" variable

data(margex)
data(cvcov)
# ?cvcov
v <- cvcov$ols$clust
d <- cvcov$ols$stata_dof
mod <- glm(outcome ~ treatment + distance,
           data = margex, family = 'binomial')
marg(mod, var_interest = 'treatment', type = 'levels',
          vcov_mat = v, dof = d)

# Using weights
data(margex)
mm <- glm(y ~ as.factor(treatment) + age, data = margex, family = 'gaussian',
          weights = distance)
z1 <- marg(mod = mm, var_interest = 'treatment', type = 'levels')[[1]]
z2 <- marg(mod = mm, var_interest = 'treatment', type = 'effects')[[1]]
```

`marg.ivreg` *Predicted Margins for 'ivreg' objects from the* AER *package*

Description

Obtains predicted margins and standard errors of those predictions from a fitted `ivreg` model object.

Usage

```
## S3 method for class 'ivreg'
marg(mod, var_interest, data, weights = NULL, ...)
```

Arguments

`mod`	model object, currently only support those of class `glm` or `ivreg`
`var_interest`	name of the variable of interest, must correspond to a covariate in the model
`data`	data.frame that margins should run over, defaults changes based on class-specific method
`weights`	numeric, vector of weights used to generate predicted levels, defaults changes based on class-specific method. Must be equal to the number of rows in `data`.
`...`	additional parameters passed to `?marg`.

Examples

```
# From ?AER::ivreg

# data
data("CigarettesSW", package = "AER")
CigarettesSW$rprice <- with(CigarettesSW, price/cpi)
CigarettesSW$rincome <- with(CigarettesSW, income/population/cpi)
CigarettesSW$tdiff <- with(CigarettesSW, (taxs - tax)/cpi)

# model
fm <- AER::ivreg(log(packs) ~ log(rprice) + log(rincome) |
                   log(rincome) + tdiff + I(tax/cpi),
                 data = CigarettesSW, subset = year == "1995")

# Get margins for different levels of price/cpi
rprice_levs <- round(quantile(CigarettesSW$rprice))

marg(fm, data = subset(CigarettesSW, year == "1995"),
     var_interest = 'rprice', at_var_interest = rprice_levs)
```

`margex` *Artificial data for margins*

Description

A fictitious dataset outcome, treatment, and demographic variables for 3000 observations.

Usage

```
margex
```

Format

A data frame with 3000 rows and 11 variables:

y numeric

outcome integer, 0 or 1

sex character: "female" or "male"

group integer

age integer

distance numeric

ycn numeric

yc numeric, 0 or 1

treatment integer

agegroup character: "20-29", "30-39", or "40+"

arm integer

Source

http://www.stata-press.com/data/r14/margex.dta

`pred_se` *Main wrapper function to calculate margins and standard errors*

Description

For one set of transformed covariates (not including the variable of interest), calculate the predicted level and standard error for the variable of interest.

Usage

```
pred_se(df_levels, model, type, base_rn, vcov_mat, weights, deriv_func,
  link_func)
```

Arguments

`df_levels`	data.frame, already transformed for variables not related to the variable of interest
`model`	model object
`type`	either effects or levels
`base_rn`	numeric, row number of the base level
`vcov_mat`	matrix, variance-covariance matrix
`weights`	vector of weights, or NULL
`deriv_func`	function for the derivative of the predicted outcomes
`link_func`	function to transform output of 'predict' method into response scale

Index