Package 'mfx'

February 20, 2015

Type Package

Title Marginal Effects, Odds Ratios and Incidence Rate Ratios for GLMs

Version 1.1

Date 2013-12-12

Author Alan Fernihough

Maintainer Alan Fernihough <alan.fernihough@gmail.com>

Description Estimates probit, logit, Poisson, negative binomial, and beta regression models, returning their marginal effects, odds ratios, or incidence rate ratios as an output.

License GPL-2 | GPL-3

Depends stats, sandwich, lmtest, MASS, betareg

NeedsCompilation no

Repository CRAN

Date/Publication 2014-01-19 20:45:05

R topics documented:

betamfx *Marginal effects for a beta regression.*

Description

This function estimates a beta regression model and calculates the corresponding marginal effects.

Usage

```
betamfx(formula, data, atmean = TRUE, robust = FALSE,
        clustervar1 = NULL, clustervar2 = NULL,
        control = betareg.control(), link.phi = NULL, type = "ML")
```

Arguments

formula	an object of class "formula" (or one that can be coerced to that class).
data	the data frame containing these data. This argument must be used.
atmean	default marginal effects represent the partial effects for the average observation. If `atmean = FALSE` the function calculates average partial effects.
robust	if TRUE the function reports White/robust standard errors.
clustervar1	a character value naming the first cluster on which to adjust the standard errors.
clustervar2	a character value naming the second cluster on which to adjust the standard errors for two-way clustering.
control	a list of control arguments specified via betareg.control.
link.phi	as in the betareg function.
type	as in the betareg function.

Details

The underlying link function in the mean model (mu) is "logit". If both `robust=TRUE` and `!is.null(clustervar1)` the function overrides the `robust` command and computes clustered standard errors.

Value

mfxest	a coefficient matrix with columns containing the estimates, associated standard errors, test statistics and p-values.
fit	the fitted betareg object.
dcvar	a character vector containing the variable names where the marginal effect refers to the impact of a discrete change on the outcome. For example, a factor variable.
call	the matched call.

References

Francisco Cribari-Neto, Achim Zeileis (2010). Beta Regression in R. Journal of Statistical Software 34(2), 1-24.

Bettina Gruen, Ioannis Kosmidis, Achim Zeileis (2012). Extended Beta Regression in R: Shaken, Stirred, Mixed, and Partitioned. Journal of Statistical Software, 48(11), 1-25.

See Also

betaor, betareg

Examples

```
# simulate some data
set.seed(12345)
n = 1000
x = rnorm(n)

# beta outcome
y = rbeta(n, shape1 = plogis(1 + 0.5 * x), shape2 = (abs(0.2*x)))
# use Smithson and Verkuilen correction
y = (y*(n-1)+0.5)/n

data = data.frame(y,x)
betamfx(y~x|x, data=data)
```

`betaor` *Odds ratios for a beta regression.*

Description

This function estimates a beta regression model and calculates the corresponding odds ratios.

Usage

```
betaor(formula, data, robust = FALSE, clustervar1 = NULL, clustervar2 = NULL,
       control = betareg.control(), link.phi = NULL, type = "ML")
```

Arguments

`formula`	an object of class "formula" (or one that can be coerced to that class).
`data`	the data frame containing these data. This argument must be used.
`robust`	if `TRUE` the function reports White/robust standard errors.
`clustervar1`	a character value naming the first cluster on which to adjust the standard errors.
`clustervar2`	a character value naming the second cluster on which to adjust the standard errors for two-way clustering.
`control`	a list of control arguments specified via `betareg.control`.
`link.phi`	as in the `betareg` function.
`type`	as in the `betareg` function.

Details

The underlying link function in the mean model (mu) is "logit". If both `robust=TRUE` and `!is.null(clustervar1)` the function overrides the `robust` command and computes clustered standard errors.

Value

`oddsratio`	a coefficient matrix with columns containing the estimates, associated standard errors, test statistics and p-values.
`fit`	the fitted `betareg` object.
`call`	the matched call.

References

Francisco Cribari-Neto, Achim Zeileis (2010). Beta Regression in R. Journal of Statistical Software 34(2), 1-24.

Bettina Gruen, Ioannis Kosmidis, Achim Zeileis (2012). Extended Beta Regression in R: Shaken, Stirred, Mixed, and Partitioned. Journal of Statistical Software, 48(11), 1-25.

See Also

`betamfx`, `betareg`

Examples

```
# simulate some data
set.seed(12345)
n = 1000
x = rnorm(n)

# beta outcome
y = rbeta(n, shape1 = plogis(1 + 0.5 * x), shape2 = (abs(0.2*x)))
# use Smithson and Verkuilen correction
y = (y*(n-1)+0.5)/n

data = data.frame(y,x)
betaor(y~x|x, data=data)
```

---

logitmfx *Marginal effects for a logit regression.*

Description

This function estimates a binary logistic regression model and calculates the corresponding marginal effects.

Usage

```
logitmfx(formula, data, atmean = TRUE, robust = FALSE, clustervar1 = NULL,
         clustervar2 = NULL, start = NULL, control = list())
```

Arguments

formula	an object of class "formula" (or one that can be coerced to that class).
data	the data frame containing these data. This argument must be used.
atmean	default marginal effects represent the partial effects for the average observation. If `atmean = FALSE` the function calculates average partial effects.
robust	if `TRUE` the function reports White/robust standard errors.
clustervar1	a character value naming the first cluster on which to adjust the standard errors.
clustervar2	a character value naming the second cluster on which to adjust the standard errors for two-way clustering.
start	starting values for the parameters in the `glm` model.
control	see `glm.control`.

Details

If both `robust=TRUE` and `!is.null(clustervar1)` the function overrides the `robust` command and computes clustered standard errors.

Value

mfxest	a coefficient matrix with columns containing the estimates, associated standard errors, test statistics and p-values.
fit	the fitted `glm` object.
dcvar	a character vector containing the variable names where the marginal effect refers to the impact of a discrete change on the outcome. For example, a factor variable.
call	the matched call.

References

William H. Greene (2008). Econometric Analysis (6th ed.). Prentice Hall, N.Y. pp 770-787.

See Also

`logitor`, `glm`

Examples

```
# simulate some data
set.seed(12345)
n = 1000
x = rnorm(n)

# binary outcome
y = ifelse(pnorm(1 + 0.5*x + rnorm(n))>0.5, 1, 0)

data = data.frame(y,x)
logitmfx(formula=y~x, data=data)
```

`logitor` *Odds ratios for a logit regression.*

Description

This function estimates a binary logistic regression model and calculates the corresponding odds ratios.

Usage

```
logitor(formula, data, robust = FALSE, clustervar1 = NULL, clustervar2 = NULL,
        start = NULL, control = list())
```

Arguments

`formula`	an object of class "formula" (or one that can be coerced to that class).
`data`	the data frame containing these data. This argument must be used.
`robust`	if `TRUE` the function reports White/robust standard errors.
`clustervar1`	a character value naming the first cluster on which to adjust the standard errors.
`clustervar2`	a character value naming the second cluster on which to adjust the standard errors for two-way clustering.
`start`	starting values for the parameters in the `glm` model.
`control`	see `glm.control`.

Details

If both `robust=TRUE` and `!is.null(clustervar1)` the function overrides the `robust` command and computes clustered standard errors.

Value

`oddsratio`	a coefficient matrix with columns containing the estimates, associated standard errors, test statistics and p-values.
`fit`	the fitted `glm` object.
`call`	the matched call.

See Also

logitmfx, glm

Examples

```
# simulate some data
set.seed(12345)
n = 1000
x = rnorm(n)

# binary outcome
y = ifelse(pnorm(1 + 0.5*x + rnorm(n))>0.5, 1, 0)

data = data.frame(y,x)
logitor(formula=y~x, data=data)
```

---

negbinirr *Incidence rate ratios for a negative binomial regression.*

Description

This function estimates a negative binomial regression model and calculates the corresponding incidence rate ratios.

Usage

```
negbinirr(formula, data, robust = FALSE, clustervar1 = NULL,
          clustervar2 = NULL, start = NULL, control = glm.control())
```

Arguments

formula	an object of class "formula" (or one that can be coerced to that class).
data	the data frame containing these data. This argument must be used.
robust	if TRUE the function reports White/robust standard errors.
clustervar1	a character value naming the first cluster on which to adjust the standard errors.
clustervar2	a character value naming the second cluster on which to adjust the standard errors for two-way clustering.
start	starting values for the parameters in the glm.nb model.
control	see glm.control.

Details

If both robust=TRUE and !is.null(clustervar1) the function overrides the robust command and computes clustered standard errors.

Value

irr	a coefficient matrix with columns containing the estimates, associated standard errors, test statistics and p-values.
fit	the fitted glm.nb object.
call	the matched call.

See Also

negbinmfx, glm.nb

Examples

```
# simulate some data
set.seed(12345)
n = 1000
x = rnorm(n)
y = rnegbin(n, mu = exp(1 + 0.5 * x), theta = 0.5)

data = data.frame(y,x)

negbinirr(formula=y~x,data=data)
```

---

negbinmfx *Marginal effects for a negative binomial regression.*

Description

This function estimates a negative binomial regression model and calculates the corresponding marginal effects.

Usage

```
negbinmfx(formula, data, atmean = TRUE, robust = FALSE, clustervar1 = NULL,
          clustervar2 = NULL, start = NULL, control = glm.control())
```

Arguments

formula	an object of class "formula" (or one that can be coerced to that class).
data	the data frame containing these data. This argument must be used.
atmean	default marginal effects represent the partial effects for the average observation. If atmean = FALSE the function calculates average partial effects.
robust	if TRUE the function reports White/robust standard errors.
clustervar1	a character value naming the first cluster on which to adjust the standard errors.
clustervar2	a character value naming the second cluster on which to adjust the standard errors for two-way clustering.
start	starting values for the parameters in the glm.nb model.
control	see glm.control.

Details

If both `robust=TRUE` and `!is.null(clustervar1)` the function overrides the `robust` command and computes clustered standard errors.

Value

`mfxest`	a coefficient matrix with columns containing the estimates, associated standard errors, test statistics and p-values.
`fit`	the fitted `glm.nb` object.
`dcvar`	a character vector containing the variable names where the marginal effect refers to the impact of a discrete change on the outcome. For example, a factor variable.
`call`	the matched call.

See Also

`negbinirr`, `glm.nb`

Examples

```
# simulate some data
set.seed(12345)
n = 1000
x = rnorm(n)
y = rnegbin(n, mu = exp(1 + 0.5 * x), theta = 0.5)

data = data.frame(y,x)

negbinmfx(formula=y~x,data=data)
```

---

poissonirr *Incidence rate ratios for a Poisson regression.*

Description

This function estimates a negative binomial regression model and calculates the corresponding incidence rate ratios.

Usage

```
poissonirr(formula, data, robust = FALSE, clustervar1 = NULL,
           clustervar2 = NULL, start = NULL, control = list())
```

Arguments

formula	an object of class "formula" (or one that can be coerced to that class).
data	the data frame containing these data. This argument must be used.
robust	if TRUE the function reports White/robust standard errors.
clustervar1	a character value naming the first cluster on which to adjust the standard errors.
clustervar2	a character value naming the second cluster on which to adjust the standard errors for two-way clustering.
start	starting values for the parameters in the glm model.
control	see glm.control.

Details

If both robust=TRUE and !is.null(clustervar1) the function overrides the robust command and computes clustered standard errors.

Value

irr	a coefficient matrix with columns containing the estimates, associated standard errors, test statistics and p-values.
fit	the fitted glm object.
call	the matched call.

See Also

poissonmfx, glm

Examples

```
# simulate some data
set.seed(12345)
n = 1000
x = rnorm(n)
y = rnegbin(n, mu = exp(1 + 0.5 * x), theta = 0.5)

data = data.frame(y,x)

poissonirr(formula=y~x,data=data)
```

poissonmfx *Marginal effects for a Poisson regression.*

Description

This function estimates a Poisson regression model and calculates the corresponding marginal effects.

Usage

```
poissonmfx(formula, data, atmean = TRUE, robust = FALSE, clustervar1 = NULL,
           clustervar2 = NULL, start = NULL, control = list())
```

Arguments

formula	an object of class "formula" (or one that can be coerced to that class).
data	the data frame containing these data. This argument must be used.
atmean	default marginal effects represent the partial effects for the average observation. If `atmean = FALSE` the function calculates average partial effects.
robust	if `TRUE` the function reports White/robust standard errors.
clustervar1	a character value naming the first cluster on which to adjust the standard errors.
clustervar2	a character value naming the second cluster on which to adjust the standard errors for two-way clustering.
start	starting values for the parameters in the `glm` model.
control	see `glm.control`.

Details

If both `robust=TRUE` and `!is.null(clustervar1)` the function overrides the `robust` command and computes clustered standard errors.

Value

mfxest	a coefficient matrix with columns containing the estimates, associated standard errors, test statistics and p-values.
fit	the fitted `glm` object.
dcvar	a character vector containing the variable names where the marginal effect refers to the impact of a discrete change on the outcome. For example, a factor variable.
call	the matched call.

See Also

`poissonirr`, `glm`

Examples

```
# simulate some data
set.seed(12345)
n = 1000
x = rnorm(n)
y = rnegbin(n, mu = exp(1 + 0.5 * x), theta = 0.5)

data = data.frame(y,x)

poissonmfx(formula=y~x,data=data)
```

---

probitmfx *Marginal effects for a probit regression.*

Description

This function estimates a probit regression model and calculates the corresponding marginal effects.

Usage

```
probitmfx(formula, data, atmean = TRUE, robust = FALSE, clustervar1 = NULL,
          clustervar2 = NULL, start = NULL, control = list())
```

Arguments

formula	an object of class "formula" (or one that can be coerced to that class).
data	the data frame containing these data. This argument must be used.
atmean	default marginal effects represent the partial effects for the average observation. If `atmean = FALSE` the function calculates average partial effects.
robust	if `TRUE` the function reports White/robust standard errors.
clustervar1	a character value naming the first cluster on which to adjust the standard errors.
clustervar2	a character value naming the second cluster on which to adjust the standard errors for two-way clustering.
start	starting values for the parameters in the `glm` model.
control	see `glm.control`.

Details

If both `robust=TRUE` and `!is.null(clustervar1)` the function overrides the `robust` command and computes clustered standard errors.

Value

`mfxest`	a coefficient matrix with columns containing the estimates, associated standard errors, test statistics and p-values.
`fit`	the fitted `glm` object.
`dcvar`	a character vector containing the variable names where the marginal effect refers to the impact of a discrete change on the outcome. For example, a factor variable.
`call`	the matched call.

References

William H. Greene (2008). Econometric Analysis (6th ed.). Prentice Hall, N.Y. pp 770-787.

See Also

`glm`

Examples

```
# simulate some data
set.seed(12345)
n = 1000
x = rnorm(n)

# binary outcome
y = ifelse(pnorm(1 + 0.5*x + rnorm(n))>0.5, 1, 0)

data = data.frame(y,x)
probitmfx(formula=y~x, data=data)
```

Index