Package ‘mvProbit’

January 11, 2021

Version 0.1-10

Date 2021-01-07

Title Multivariate Probit Models

Author Arne Henningsen <arne.henningsen@gmail.com>

Maintainer Arne Henningsen <arne.henningsen@gmail.com>

Depends R (>= 2.4.0), mvtnorm (>= 0.9-9994), maxLik (>= 1.0-0), abind (>= 1.3-0)

Imports bayesm (>= 2.2-4), miscTools (>= 0.6-11)

Description Tools for estimating multivariate probit models, calculating conditional and unconditional expectations, and calculating marginal effects on conditional and unconditional expectations.

License GPL (>= 2)

URL http://www.sampleSelection.org

NeedsCompilation no

Repository CRAN

Date/Publication 2021-01-11 06:50:02 UTC

R topics documented:

mvProbit 2
mvProbitLogLik 5
mvProbitMargEff 7
summary.mvProbit 12
summary.mvProbitMargEff 13

Index **15**

mvProbit *Estimation of Multivariate Probit Models*

Description

Estimating multivariate probit models by the maximum likelihood method.

WARNING: this function is experimental and extremely (perhaps even unusably) slow!

Usage

```
mvProbit( formula, data, start = NULL, startSigma = NULL,
   method = "BHHH", finalHessian = "BHHH",
   algorithm = "GHK", nGHK = 1000,
   intGrad = TRUE, oneSidedGrad = FALSE, eps = 1e-6,
   random.seed = 123, ... )

## S3 method for class 'mvProbit'
print( x, digits = 4, ... )
```

Arguments

`formula`	a `"formula"`: a symbolic description of the model (currently, all binary outcome variables must have the same regressors).
`data`	a `data.frame` containing the data.
`start`	an optional numeric vector specifying the starting values for the model coefficients; if argument `startSigma` is not specified, this vector can also include the correlation coefficients; the order of elements is explained in the section "details"; if this argument is not specified, coefficients estimated by univariate probit models are used as starting values for the model coefficients.
`startSigma`	optional starting values for the covariance/correlation matrix of the residuals (must be symmetric and have ones on its diagonal); if this argument is not specified and the starting values for the correlation coefficients are not included in argument `start`, the correlation matrix of the 'response' residuals, i.e. y - pnorm(X' beta), is used as starting values for sigma.
`method`	maximisation method / algorithm (see `maxLik`).
`finalHessian`	Calculation of the final Hessian: either `FALSE` (no calculation of Hessian), `TRUE` (finite-distance calculation of Hessian), or `"BHHH"` (calculation based on information equality approach and finite-distance gradients, the default).
`algorithm`	algorithm for computing integrals of the multivariate normal distribution, either function `GenzBretz()`, `Miwa()`, or `TVPACK()` (see documentation of `pmvnorm`) or character string `"GHK"` (see documentation of `ghkvec`).
`nGHK`	numeric value specifying the number of simulation draws of the GHK algorithm for computing integrals of the multivariate normal distribution.

intGrad	logical. If TRUE, the computation of the gradients with respect to the estimated parameters is done internally in function mvProbitLogLik when it computes the log-likelihood values. If the optimization method requires gradients and this argument is FALSE, maxLik computes the gradients by numericGradient, which is usually slower than the calculation in function mvProbitLogLik. This argument should be set to FALSE if an optimisation algorithm is used that is not based on gradients.
oneSidedGrad	logical. If this argument and argument intGrad are both TRUE, the gradients of the log-likelihood function with respect to the estimated parameters are obtained by one-sided numeric finit-difference differentiation, which is faster but less precise than two-sided numeric finit-difference differentiation.
eps	numeric. The step size for the one-sided numeric finit-distance differentiation. Unfortunately, it is currently not possible to set the step size for the two-sided numeric finit-distance differentiation.
random.seed	an integer used to seed R's random number generator; this is to ensure replicability when computing (cumulative) probabilities of the multivariate normal distribution which is required to calculate the log likelihood values; set.seed(random.seed) is called each time before a (cumulative) probability of the multivariate normal distribution is computed; defaults to 123.
x	object of class mvProbit (returned by mvProbit).
digits	positive integer specifiying the minimum number of significant digits to be printed (see print.default).
...	additional arguments to mvProbit are passed to maxLik and pmvnorm; additional arguments to print.mvProbit are currently ignored.

Details

It is possible to specify starting values (a) both for the model coefficients and the correlation coefficients (using argument start alone or arguments start and startSigma together), (b) only for the model coefficients (using argument start alone), or (c) only for the correlation coefficients (using argument startSigma alone).

If the model has n dependent variables (equations) and k explanatory variables in each equation, the order of the starting values in argument start must be as follows: $b_{1,1}$, …, $b_{1,k}$, $b_{2,1}$, …, $b_{2,k}$, …, $b_{n,1}$, …, $b_{n,k}$, where $b_{i,j}$ is the coefficient of the jth explanatory variable in the ith equation. If argument startSigma is not specified, argument start can additionally include following elements: $R_{1,2}$, $R_{1,3}$, $R_{1,4}$, …, $R_{1,n}$, $R_{2,3}$, $R_{2,4}$, …, $R_{2,n}$, …, $R_{n-1,n}$, where $R_{i,j}$ is the correlation coefficient corresponding to the ith and jth equation.

The 'state' (or 'seed') of R's random number generator is saved at the beginning of the mvProbit function and restored at the end of this function so that this function does *not* affect the generation of random numbers outside this function although the random seed is set to argument random.seed and the calculation of the (cumulative) multivariate normal distribution uses random numbers.

Value

mvProbit returns an object of class "mvProbit" inheriting from class "maxLik". The returned object contains the same components as objects returned by maxLik and additionally the following components:

`call`	the matched call.
`start`	the vector of starting values.
`nDep`	the number of dependent variables.
`nReg`	the number of explanatory variables (regressors).
`nObs`	the number of observations.
`dummyVars`	vector of character strings indicating the names of explanatory variables that contain only zeros and ones or only `TRUE` and `FALSE`. It is `NULL`, if no explanatory variable is indentified as a dummy variable.

Author(s)

Arne Henningsen

References

Greene, W.H. (1996): *Marginal Effects in the Bivariate Probit Model*, NYU Working Paper No. EC-96-11. Available at `https://www.ssrn.com/abstract=1293106`.

See Also

`mvProbitLogLik`, `mvProbitMargEff`, `probit`, `glm`

Examples

```
## generate a simulated data set
set.seed( 123 )
# number of observations
nObs <- 50

# generate explanatory variables
xMat <- cbind(
   const = rep( 1, nObs ),
   x1 = as.numeric( rnorm( nObs ) > 0 ),
   x2 = rnorm( nObs ) )

# model coefficients
beta <- cbind( c(  0.8,  1.2, -0.8 ),
               c( -0.6,  1.0, -1.6 ),
               c(  0.5, -0.6,  1.2 ) )

# covariance matrix of error terms
library( miscTools )
sigma <- symMatrix( c( 1, 0.2, 0.4, 1, -0.1, 1 ) )

# generate dependent variables
yMatLin <- xMat %*% beta
yMat <- ( yMatLin + rmvnorm( nObs, sigma = sigma ) ) > 0
colnames( yMat ) <- paste( "y", 1:3, sep = "" )

# estimation (BHHH optimizer and GHK algorithm)
```

```
estResult <- mvProbit( cbind( y1, y2, y3 ) ~ x1 + x2,
   data = as.data.frame( cbind( xMat, yMat ) ), iterlim = 1, nGHK = 50 )
summary( estResult )

# same estimation with user-defined starting values
estResultStart <- mvProbit( cbind( y1, y2, y3 ) ~ x1 + x2,
   start = c( beta ), startSigma = sigma,
   data = as.data.frame( cbind( xMat, yMat ) ), iterlim = 1, nGHK = 50 )
summary( estResultStart )
```

`mvProbitLogLik` *Log Likelihood Values for Multivariate Probit Models*

Description

Function `mvProbitLogLik` calculates log likelihood values of multivariate probit models.

The `logLik` model returns or calculates log likelihood values of multivariate probit models estimated by `mvProbit`.

Usage

```
mvProbitLogLik( formula, coef, sigma = NULL, data,
   algorithm = "GHK", nGHK = 1000,
   returnGrad = oneSidedGrad, oneSidedGrad = FALSE, eps = 1e-6,
   random.seed = 123, ... )

## S3 method for class 'mvProbit'
logLik( object, coef = NULL, data = NULL,
   algorithm = NULL, nGHK = NULL, random.seed = NULL, ... )
```

Arguments

`formula`	a `"formula"`: a symbolic description of the model (currently, all binary outcome variables must have the same regressors).
`coef`	a numeric vector of the model coefficients; if argument `sigma` is not specified, this vector must also include the correlation coefficients; the order of elements is explained in the section “details”.
`sigma`	optional argument for specifying the covariance/correlation matrix of the residuals (must be symmetric and have ones on its diagonal); if this argument is not specified, the correlation coefficients must be specified by argument `coef`.
`data`	a `data.frame` containing the data.
`algorithm`	algorithm for computing integrals of the multivariate normal distribution, either function `GenzBretz()`, `Miwa()`, or `TVPACK()` (see documentation of `pmvnorm`) or character string `"GHK"` (see documentation of `ghkvec`).
`nGHK`	numeric value specifying the number of simulation draws of the GHK algorithm for computing integrals of the multivariate normal distribution.

returnGrad	logical. If TRUE, the returned object has an attribute "gradient", which is a matrix and provides the gradients of the log-likelihood function with respect to all parameters (coef and upper triangle of sigma) evaluated at each observation and obtained by (two-sided) numeric finite-difference differentiation.
oneSidedGrad	logical. If TRUE, attribute "gradient" of the returned object is obtained by one-sided numeric finite-difference differentiation.
eps	numeric. The step size for the numeric finite-distance differentiation.
random.seed	an integer used to seed R's random number generator; this is to ensure replicability when computing (cumulative) probabilities of the multivariate normal distribution which is required to calculate the log likelihood values; set.seed(random.seed) is called each time before a (cumulative) probability of the multivariate normal distribution is computed; defaults to 123.
object	an object of class "mvProbit" (returned by mvProbit.
...	additional arguments are passed to pmvnorm when calculating conditional expectations.

Details

If the logLik method is called with object as the only argument, it returns the log-likelihood value found in the maximum likelihood estimation. If any other argument is not NULL, the logLik method calculates the log-likelihood value by calling mvProbitLogLik. All arguments that are NULL, are taken from argument object.

If the model has n dependent variables (equations) and k explanatory variables in each equation, the order of the model coefficients in argument coef must be as follows: $b_{1,1}, \ldots, b_{1,k}, b_{2,1}, \ldots, b_{2,k}, \ldots, b_{n,1}, \ldots, b_{n,k}$, where $b_{i,j}$ is the coefficient of the jth explanatory variable in the ith equation. If argument sigma is not specified, argument coef must additionally include following elements: $R_{1,2}, R_{1,3}, R_{1,4}, \ldots, R_{1,n}, R_{2,3}, R_{2,4}, \ldots, R_{2,n}, \ldots, R_{n-1,n}$, where $R_{i,j}$ is the correlation coefficient corresponding to the ith and jth equation.

The 'state' (or 'seed') of R's random number generator is saved at the beginning of the mvProbitLogLik function and restored at the end of this function so that this function does *not* affect the generation of random numbers outside this function although the random seed is set to argument random.seed and the calculation of the (cumulative) multivariate normal distribution uses random numbers.

Value

mvProbitLogLik returns a vector containing the log likelihood values for each observation.

If argument returnGrad is TRUE, the vector returned by mvProbitLogLik has an attribute "gradient", which is a matrix and provides the gradients of the log-likelihood function with respect to all parameters (coef and upper triangle of sigma) evaluated at each observation and obtained by numeric finite-difference differentiation.

The logLik method returns the total log likelihood value (sum over all observations) with attribute df equal to the number of estimated parameters (model coefficients and correlation coefficients).

Author(s)

Arne Henningsen

References

Greene, W.H. (1996): *Marginal Effects in the Bivariate Probit Model*, NYU Working Paper No. EC-96-11. Available at https://www.ssrn.com/abstract=1293106.

See Also

mvProbit, mvProbitMargEff, probit, glm

Examples

```
## generate a simulated data set
set.seed( 123 )
# number of observations
nObs <- 10

# generate explanatory variables
xMat <- cbind(
   const = rep( 1, nObs ),
   x1 = as.numeric( rnorm( nObs ) > 0 ),
   x2 = as.numeric( rnorm( nObs ) > 0 ),
   x3 = rnorm( nObs ),
   x4 = rnorm( nObs ) )

# model coefficients
beta <- cbind( c(  0.8,  1.2, -1.0,  1.4, -0.8 ),
               c( -0.6,  1.0,  0.6, -1.2, -1.6 ),
               c(  0.5, -0.6, -0.7,  1.1,  1.2 ) )

# covariance matrix of error terms
library( miscTools )
sigma <- symMatrix( c( 1, 0.2, 0.4, 1, -0.1, 1 ) )

# generate dependent variables
yMatLin <- xMat %*% beta
yMat <- ( yMatLin + rmvnorm( nObs, sigma = sigma ) ) > 0
colnames( yMat ) <- paste( "y", 1:3, sep = "" )

# log likelihood values
myData <- as.data.frame( cbind( xMat, yMat ) )
logLikVal <- mvProbitLogLik( cbind( y1, y2, y3 ) ~ x1 + x2 + x3 + x4,
   coef = c( beta ), sigma = sigma, data = myData )
print( logLikVal )
```

---

mvProbitMargEff *Expectations and Marginal Effects from Multivariate Probit Models*

Description

mvProbitExp calculates expected outcomes from multivariate probit models.

mvProbitMargEff calculates marginal effects of the explanatory variables on expected outcomes from multivariate probit models.

The margEff method for objects of class "mvProbit" is a wrapper function that (for the convenience of the user) extracts the relevant information from the estimation results and then calls mvProbitMargEff.

Usage

```
mvProbitExp( formula, coef, sigma = NULL, data,
   cond = FALSE, algorithm = "GHK", nGHK = 1000, random.seed = 123, ... )

mvProbitMargEff( formula, coef, sigma = NULL, vcov = NULL, data,
   cond = FALSE, algorithm = "GHK", nGHK = 1000,
   eps = 1e-06, dummyVars = NA, addMean = FALSE,
   returnJacobian = FALSE, random.seed = 123,
   ... )

## S3 method for class 'mvProbit'
margEff( object, data = eval( object$call$data ),
   cond = FALSE, othDepVar = NULL, dummyVars = object$dummyVars,
   atMean = FALSE, calcVCov = FALSE, ... )
```

Arguments

formula	a one-sided or two-sided "formula": a symbolic description of the model (currently, all binary outcome variables must have the same explanatory variables).
coef	a numeric vector of the model coefficients; if argument sigma is not specified, this vector must also include the correlation coefficients; the order of elements is explained in the section “details”.
sigma	optional argument for specifying the covariance/correlation matrix of the residuals (must be symmetric and have ones on its diagonal); if this argument is not specified, the correlation coefficients must be specified by argument coef.
vcov	an optional symmetric matrix specifying the variance-covariance matrix of all coefficients (model coefficients and correlation coefficients); if this argument is specified, the approximate variance covariance matrices of the marginal effects are calculated and returned as an attribute (see below).
data	a data.frame containing the data.
cond	logical value indicating whether (marginal effects on) conditional expectations (if TRUE) or (marginal effects on) unconditional expectations (if FALSE, default) should be returned.
algorithm	algorithm for computing integrals of the multivariate normal distribution, either function GenzBretz(), Miwa(), or TVPACK() (see documentation of pmvnorm) or character string "GHK" (see documentation of ghkvec).

nGHK	numeric value specifying the number of simulation draws of the GHK algorithm for computing integrals of the multivariate normal distribution.
eps	numeric, the difference between the two values of each (numerical) explanatory variable that is used for the numerical differentiation.
dummyVars	optional vector containing the names (character strings) of explanatory variables that should be treated as dummy variables (see section 'Details'). If NA (the default), dummy variables are detected automatically, i.e. all explanatory variables which contain only zeros and ones or only TRUE and FALSE in the data set specified by argument data are treated as dummy variables. If NULL, no variable is treated as dummy variable.
addMean	logical. If TRUE, the mean of values of all marginal effects are added in an additional row at the bottom of the returned data.frame. If argument returnJacobian is TRUE, the Jacobian of the mean marginal effects with respect to the coefficients is included in the returned array of the Jacobians (in an additional slot at the end of the first dimension). If argument vcov of mvProbitMargEff is specified or argument calcVCov of the margEff method is TRUE, the variance covariance matrix of the mean marginal effects is included in the returned array of the variance covariance matrices (in an additional slot at the end of the first dimension).
returnJacobian	logical. If TRUE, the Jacobian of the marginal effects with respect to the coefficients is returned.
random.seed	an integer used to seed R's random number generator; this is to ensure replicability when computing (cumulative) probabilities of the multivariate normal distribution which is required to calculate the conditional expectations; set.seed(random.seed) is called each time before a (cumulative) probability of the multivariate normal distribution is computed; defaults to 123.
object	an object of class "mvProbit" (returned by mvProbit.
othDepVar	optional scalar or vector for specifying the values of the (other) dependent variables when calculating the marginal effects on the conditional expectations. If this argument is a scalar (zero or one), it is assumed that all (other) dependent variables have this value at all observations. If this argument is a vector (filled with zeros or ones) with length equal to the number of dependent variables, it is assumed that the vector of dependent variables has these values at all observations. If this argument is NULL (the default), the dependent variables are assumed to have the values that these variables have in the data set data.
atMean	logical. If TRUE, the marginal effects are calculated not at each observation but at the mean values across all observations of the variables in the data set specified by argument data.
calcVCov	logical. If TRUE, the approximate variance covariance matrices of the marginal effects are calculated and returned as an attribute (see below).
...	additional arguments to mvProbitExp are passed to pmvnorm when calculating conditional expectations; additional arguments of mvProbitMargEff are passed to mvProbitExp and possibly further to pmvnorm; additional arguments of the margEff method are passed to mvProbitMargEff and possibly further to mvProbitMargEff and pmvnorm.

Details

When calculating (marginal effects on) unconditional expectations, the left-hand side of argument formula is ignored. When calculating (marginal effects on) conditional expectations and argument formula is a one-sided formula (i.e. only the right-hand side is specified) or argument othDepOne is TRUE, (the marginal effects on) the conditional expectations are calculated based on the assumption that all other dependent variables are one.

The computation of the marginal effects of dummy variables (i.e. variables specified in argument dummyVars) ignores argument eps and evaluates the effect of increasing these variables from zero to one. The marginal effects of (continuous) variables (i.e. variables not specified in argument dummyVars) are calculated by evaluating the effect of increasing these variables from their actual values minus 0.5 * eps to their actual values plus 0.5 * eps (divided by eps).

If the model has n dependent variables (equations) and k explanatory variables in each equation, the order of the model coefficients in argument coef must be as follows: $b_{1,1}$, …, $b_{1,k}$, $b_{2,1}$, …, $b_{2,k}$, …, $b_{n,1}$, …, $b_{n,k}$, where $b_{i,j}$ is the coefficient of the jth explanatory variable in the ith equation. If argument sigma is not specified, argument coef must additionally include following elements: $R_{1,2}, R_{1,3}, R_{1,4}, \ldots, R_{1,n}, R_{2,3}, R_{2,4}, \ldots, R_{2,n}, \ldots, R_{n-1,n}$, where $R_{i,j}$ is the correlation coefficient corresponding to the ith and jth equation.

If argument vcov of function mvProbitMargEff is specified or argument calcVCov of the margEff method is TRUE, the approximate variance covariance matrices of the marginal effects are calculated at each observation by using the 'delta method', where the jacobian matrix of the marginal effects with respect to the coefficients is obtained by numerical differentiation.

The 'state' (or 'seed') of R's random number generator is saved at the beginning of the call to these functions and restored at the end so that these functions do *not* affect the generation of random numbers outside this function although the random seed is set to argument random.seed and the calculation of the (cumulative) multivariate normal distribution uses random numbers.

Value

mvProbitExp returns a data frame containing the expectations of the dependent variables.

mvProbitMargEff and the margEff method return a data frame containing the marginal effects of the explanatory variables on the expectations of the dependent variables.

If argument vcov of function mvProbitMargEff is specified or argument calcVCov of the margEff method is TRUE, the returned data frame has an attribute vcov, which is a three-dimensional array, where the first dimension corresponds to the observation and the latter two dimensions span the approximate variance covariance matrix of the marginal effects calculated for each observation.

If argument returnJacobian of function mvProbitMargEff or method margEff is set to TRUE, the returned data frame has an attribute jacobian, which is a three-dimensional array that contains the Jacobian matrices of the marginal effects with respect to the coefficients at each observation, where the first dimension corresponds to the observations, the second dimension corresponds to the marginal effects, and the third dimension corresponds to the coefficients.

Author(s)

Arne Henningsen

References

Greene, W.H. (1996): *Marginal Effects in the Bivariate Probit Model*, NYU Working Paper No. EC-96-11. Available at https://www.ssrn.com/abstract=1293106.

See Also

mvProbit, mvProbitLogLik, probit, glm

Examples

```
## generate a simulated data set
set.seed( 123 )
# number of observations
nObs <- 10

# generate explanatory variables
xData <- data.frame(
   const = rep( 1, nObs ),
   x1 = as.numeric( rnorm( nObs ) > 0 ),
   x2 = as.numeric( rnorm( nObs ) > 0 ),
   x3 = rnorm( nObs ),
   x4 = rnorm( nObs ) )

# model coefficients
beta <- c(  0.8,  1.2, -1.0,  1.4, -0.8,
           -0.6,  1.0,  0.6, -1.2, -1.6,
            0.5, -0.6, -0.7,  1.1,  1.2 )

# covariance matrix of error terms
library( miscTools )
sigma <- symMatrix( c( 1, 0.2, 0.4, 1, -0.1, 1 ) )

# unconditional expectations of dependent variables
yExp <- mvProbitExp( ~ x1 + x2 + x3 + x4, coef = c( beta ),
   sigma = sigma, data = xData )
print( yExp )

# marginal effects on unconditional expectations of dependent variables
margEffUnc <- mvProbitMargEff( ~ x1 + x2 + x3 + x4, coef = c( beta ),
   sigma = sigma, data = xData )
print( margEffUnc )

# conditional expectations of dependent variables
# (assuming that all other dependent variables are one)
yExpCond <- mvProbitExp( ~ x1 + x2 + x3 + x4, coef = beta,
   sigma = sigma, data = xData, cond = TRUE )
print( yExpCond )

# marginal effects on conditional expectations of dependent variables
# (assuming that all other dependent variables are one)
margEffCond <- mvProbitMargEff( ~ x1 + x2 + x3 + x4, coef = beta,
   sigma = sigma, data = xData, cond = TRUE )
```

```
print( margEffCond )

# conditional expectations of dependent variables
# (assuming that all other dependent variables are zero)
xData$y1 <- 0
xData$y2 <- 0
xData$y3 <- 0
yExpCond0 <- mvProbitExp( cbind( y1, y2, y3 ) ~ x1 + x2 + x3 + x4,
   coef = beta, sigma = sigma, data = xData, cond = TRUE )
print( yExpCond0 )

# marginal effects on conditional expectations of dependent variables
# (assuming that all other dependent variables are zero)
margEffCond0 <- mvProbitMargEff( cbind( y1, y2, y3 ) ~ x1 + x2 + x3 + x4,
   coef = beta, sigma = sigma, data = xData, cond = TRUE )
print( margEffCond0 )
```

---

summary.mvProbit *Summary Results of Multivariate Probit Models*

Description

These methods prepare and print summary results for multivariate probit models.

Usage

```
## S3 method for class 'mvProbit'
summary( object, ... )

## S3 method for class 'summary.mvProbit'
print( x, digits = 4, ... )
```

Arguments

`object`	object of class "`mvProbit`" (returned by `mvProbit`).
`x`	object of class "`summary.mvProbit`" (returned by `summary.mvProbit`).
`digits`	positive integer specifiying the minimum number of significant digits to be printed (passed to `printCoefmat`).
`...`	currently not used.

Value

`summary.mvProbit` returns an object of class "`summary.mvProbit`" inheriting from class "`summary.maxLik`". The returned object contains the same components as objects returned by `summary.maxLik` and additionally the following components:

`call`	the matched call.
`start`	the vector of starting values.

nDep	the number of dependent variables.
nReg	the number of explanatory variables (regressors).
nObs	the number of observations.

Author(s)

Arne Henningsen

See Also

mvProbit

---

summary.mvProbitMargEff *Summarize Marginal Effects of Multivariate Probit Models*

Description

These methods prepare and print a statistical summary of marginal effects of multivariate probit models.

Usage

```
## S3 method for class 'mvProbitMargEff'
summary( object, ... )

## S3 method for class 'summary.mvProbitMargEff'
print( x, digits = 4, ... )
```

Arguments

object	object of class "mvProbitMargEff" (returned by mvProbitMargEff or margEff.mvProbit).
x	object of class "summary.mvProbitMargEff" (returned by summary.mvProbitMargEff).
digits	positive integer specifiying the minimum number of significant digits to be printed (passed to printCoefmat).
...	currently not used.

Value

summary.mvProbitMargEff returns an object of class "summary.mvProbitMargEff" inheriting from class "matrix". The returned object is a matrix with four columns, where the marginal effects are in the first column, their standard errors are in the second column, corresponding ‘z values’ are in the third column, and the resulting ‘P values’ are in the last column.

Author(s)

Arne Henningsen

See Also

`mvProbitMargEff`, `margEff.mvProbit`, `mvProbit`.

Index

∗ **methods**
    summary.mvProbit, 12
    summary.mvProbitMargEff, 13
∗ **models**
    mvProbit, 2
    mvProbitLogLik, 5
    mvProbitMargEff, 7
∗ **regression**
    mvProbit, 2
    mvProbitLogLik, 5
    mvProbitMargEff, 7

ghkvec, *2*, *5*, *8*
glm, *4*, *7*, *11*

logLik.mvProbit *(*mvProbitLogLik*)*, 5

margEff.mvProbit, *13, 14*
margEff.mvProbit *(*mvProbitMargEff*)*, 7
maxLik, *2, 3*
mvProbit, 2, *5–7*, *9*, *11–14*
mvProbitExp *(*mvProbitMargEff*)*, 7
mvProbitLogLik, *3, 4*, 5, *11*
mvProbitMargEff, *4*, *7*, 7, *13, 14*

numericGradient, *3*

pmvnorm, *2, 3*, *5, 6*, *8, 9*
print.default, *3*
print.mvProbit *(*mvProbit*)*, 2
print.summary.mvProbit
    *(*summary.mvProbit*)*, 12
print.summary.mvProbitMargEff
    *(*summary.mvProbitMargEff*)*, 13
printCoefmat, *12, 13*
probit, *4*, *7*, *11*

summary.maxLik, *12*
summary.mvProbit, *12*, 12
summary.mvProbitMargEff, *13*, 13