Package ‘BiProbitPartial’

January 10, 2019

Type Package
Title Bivariate Probit with Partial Observability
Version 1.0.3
Date 2019-01-10
Author Michael Guggisberg and Amrit Romana
Maintainer Michael Guggisberg <mguggisb@ida.org>
Description A suite of functions to estimate, summarize and perform predictions with the bivariate probit subject to partial observability. The frequentist and Bayesian probabilistic philosophies are both supported. The frequentist method is estimated with maximum likelihood and the Bayesian method is estimated with a Markov Chain Monte Carlo (MCMC) algorithm developed by Rajbanhdari, A (2014) <doi:10.1002/9781118771051.ch13>.
License GPL-3
Imports Rcpp(>= 0.12.19), Formula(>= 1.2-3), optimr(>= 2016-8.16), pbivnorm(>= 0.6.0), mvtnorm(>= 1.0-8), RcppTN(>= 0.2-2), coda(>= 0.19-2)
Depends numDeriv(>= 2016.8-1)
Suggests sampleSelection
LinkingTo Rcpp, RcppArmadillo, RcppTN
RoxygenNote 6.1.0
Encoding UTF-8
NeedsCompilation yes
Repository CRAN
Date/Publication 2019-01-10 22:12:04 UTC

R topics documented:

BiProbitPartial-package .................................................. 2
BiProbitPartial .............................................................. 2
grad1 ................................................................. 6
A suite of functions to estimate, summarize and perform predictions with the bivariate probit subject to partial observability. The frequentist and Bayesian probabilistic philosophies are both supported. The frequentist method is estimated with maximum likelihood and the Bayesian method is estimated with a Markov Chain Monte Carlo (MCMC) algorithm developed by Rajbanhdari, A (2014) <doi:10.1002/9781118771051.ch13>.

Description

BiProbitPartial estimates a bivariate probit with partial observability model.

The bivariate probit with partial observability model is defined as follows. Let \( i \) denote the \( i \)th observation which takes values from 1 to \( N \), \( X_1 \) be a covariate matrix of dimension \( N \times k_1 \), \( X_2 \) be a covariate matrix of dimension \( N \times k_2 \), \( X_{1i} \) be the \( i \)th row of \( X_1 \), \( X_{2i} \) be the \( i \)th row of \( X_2 \), \( \beta_1 \) be a coefficient vector of length \( k_1 \) and \( \beta_2 \) be a coefficient vector of length \( k_2 \). Define the latent response for stage one to be

\[
y_{1i}^* = X_{1i} \beta_1 + \epsilon_{1i}
\]

and stage two to be

\[
y_{2i}^* = X_{2i} \beta_2 + \epsilon_{2i}.
\]

Note the stages do not need to occur sequentially. Define the outcome of the first stage to be \( y_{1i} = 1 \) if \( y_{1i}^* > 0 \) and \( y_{1i} = 0 \) if \( y_{1i}^* \leq 0 \). Define the outcome of the second stage to be \( y_{2i} = 1 \) if \( y_{2i}^* > 0 \) and \( y_{2i} = 0 \) if \( y_{2i}^* \leq 0 \). The observed outcome is the product of the outcomes from the two stages

\[
z_i = y_{1i} y_{2i}.
\]

The pair \((\epsilon_{1i}, \epsilon_{2i})\) is distributed independently and identically multivariate normal with means \(E[\epsilon_{1i}] = E[\epsilon_{2i}] = 0\), variances \(Var[\epsilon_{1i}] = Var[\epsilon_{2i}] = 1\), and correlation (or equivalently covariance) \(Cov(\epsilon_{1i}, \epsilon_{2i}) = \rho\). A more general structural representation is presented in Poirier (1980).
The model can be estimated by Bayesian Markov Chain Monte Carlo (MCMC) or frequentist maximum likelihood methods. The correlation parameter $\rho$ can be estimated or fixed. The MCMC algorithm used is a block Gibbs sampler within Metropolis-Hastings scheme developed by Rajbhandari (2014). The default maximum likelihood method is based off the Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm. A modification of the algorithm is used to include box constraints for when $\rho$ is estimated. See optimr for details.

Usage

```r
BiProbitPartial(formula, data, subset, na.action, 
    philosophy = "bayesian", control = list())
```

Arguments

- `formula`: an object of class Formula: a symbolic description of the model to be fitted. The details of model specification are given under 'Details'.
- `data`: an optional data frame, list or environment (or object coercible by `as.data.frame` to a data frame) containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the environment from which BiProbitPartial is called.
- `subset`: an optional vector specifying a subset of observations to be used in the fitting process.
- `na.action`: a function which indicates what should happen when the data contain NA observations. The default is set by the na.action setting of options, and is na.fail if that is unset. The 'factory-fresh' default is na.omit. Another possible value is na.exclude can be useful.
- `philosophy`: a character string indicating the philosophy to be used for estimation. For Bayesian MCMC estimation philosophy = "bayesian" should be used. For frequentist maximum likelihood estimation philosophy = "frequentist" should be used. The default is Bayesian MCMC estimation.
- `control`: a list of control parameters. See 'Details'.

Details

Models for BiProbitPartial are specified symbolically. A typical model has the form response ~ terms1 | terms2 where response is the name of the (numeric binary) response vector and terms1 and terms2 are each a series of terms which specifies a linear predictor for latent response equations 1 and 2. A terms1 specification of the form first + second indicates all the terms in first together with all the terms in second with duplicates removed. A specification of the form first:second indicates the set of terms obtained by taking the interactions of all terms in first with all terms in second. The specification first*second indicates the cross of first and second. This is the same as first + second + first:second. Likewise for terms2.

A Formula has an implied intercept term for both equations. To remove the intercept from equation 1 use either response ~ terms1 - 1 | terms2 or response ~ 0 + terms1 | terms2. It is analogous to remove the intercept from the equation 2.

If philosophy = "bayesian" is specified then the model is estimated by MCMC methods based on Rajbhandari (2014). The prior for the parameters in equations 1 and 2 is multivariate normal...
BiProbitPartial

with mean $\beta_0$ and covariance $B_0$. The prior for $\rho$ is truncated normal on the interval $[-1, 1]$ with mean parameter $\rho_0$ and variance parameter $v_0$ and is assumed to be apriori independent of the parameters in equations 1 and 2.

If philosophy = "frequentist" then the model is estimated by frequentist maximum likelihood using optimr from the package optimr.

The control argument is a list that can supply the tuning parameters of the Bayesian MCMC estimation and frequentist maximum likelihood estimation algorithms. For frequentist maximum likelihood the control argument is passed directly to control in the function optimr from the package optimr. If one wants to specify the method for the function optimr then method must be passed as an element of control. See optimr for further details.

The control argument can supply any of the following components for Bayesian MCMC estimation.

- **beta** Numeric vector or list of nchains elements each a numeric vector supplying starting values for the coefficients in equations 1 and 2. For each vector, the first $k_1$ values are for the coefficients in the first equation. The second $k_2$ values are for the coefficients in the second equation. Default is $\beta = \text{numeric}(k_1 + k_2)$, a vector of zeros.

- **rho** Numeric or list of nchains elements each a numeric starting value for $\rho$. Default is $\rho = 0$.

- **fixrho** Logical value to determine if $\rho$ is estimated. If fixrho = TRUE then $\rho$ is fixed at value rho. Default is fixrho = FALSE.

- **S** Number of MCMC iterations. Default is $S = 1000$. For philosophy = "bayesian" only.

- **burn** Number of initial pre-thinning MCMC iterations to remove after estimation. Default is $\text{burn} = \text{floor}(S/2)$, the floor of the number of MCMC iterations divided by 2. For philosophy = "bayesian" only.

- **thin** Positive integer to keep every thin post-burn in MCMC draw and drop all others. Default is thin = 1, keep all post burn-in draws. For philosophy = "bayesian" only.

- **seed** Positive integer for nchains = 1 or list of nchains elements each a positive integer fixing the seed of the random number generator. Typically used for replication. Default is seed = NULL, no seed. For philosophy = "bayesian" only.

- **nchains** Positive integer specifying the number of MCMC chains. Default is nchains = 1. For philosophy = "bayesian" only.

- **beta0** Numeric vector supplying the prior mean for the coefficients of equations 1 and 2. The first $k_1$ components are for the coefficients of equation 1. The second $k_2$ components are for the coefficients of equation 2. Default is beta0 = numeric($k_1 + k_2$), a vector of zeros. For philosophy = "bayesian" only.

- **B0** Numeric matrix supplying the prior covariance of the parameters of equations 1 and 2. The first $k_1$ rows are for the parameters of equation 1. The second $k_2$ rows are for the parameters of equation 2. Likewise for columns. If unspecified the default is set such that the inverse of $B_0$ is a zero matrix of dimension $(k_1+k_2) \times (k_1+k_2)$, a 'flat' prior. For philosophy = "bayesian" only.

- **rho0** Numeric value supplying a prior parameter for $\rho$ which is the mean of a normal distribution that is truncated to the interval $[-1, 1]$. Default is $\rho_0 = 0$. For philosophy = "bayesian" only.
v0 Numeric value supplying a prior parameter for $\rho$ which is the variance of a normal distribution that is truncated to the interval $[-1, 1]$. Default is $v0 = 1$. For philosophy = "bayesian" only.

nu Numeric degrees of freedom parameter for setting the degrees of freedom for $\rho$'s proposal t-distribution. Default is $nu = 10$.

tauSq Numeric scaling parameter for scaling $\rho$'s proposal t-distribution. Default is $tauSq = 1$.

$P$ Determines how aggressive proposal draws for $\rho$ are. Set to $P = 0$ normal or $P = -1$ for aggressive. See Rajbhandari (2014) and for details. Default is $P = 0$. For philosophy = "bayesian" only.

trace Numeric value determining the value of intermediate reporting. A negative value is no reporting, larger positive values provide higher degrees of reporting.

Note: If the Bayesian MCMC chains appear to not be converging and/or frequentist maximum likelihood produces errors with summary, the model may be unidentified. One possible solution is to add regressors to the first equation that are excluded from the second equation or visa-versa. See Poirier (1980) for more details.

Value

BiProbitPartial returns an $S \times (k_1 + k_2 + 1) \times nchains$ array of MCMC draws of primary class mcmc.list and secondary class BiProbitPartialb if philosophy = "bayesian". Each element in the first dimension represents a MCMC draw. The first $k_1$ elements in the second dimension are draws for the coefficients in the first equation. The next $k_2$ elements of the second dimension are draws for the coefficients in the second equation. The last element of the second dimension are draws for the correlation parameter. The elements of the third dimension are the chains. If $\rho$ was fixed (fixrho = TRUE) then each draw for the last element in the second dimension is returned as the value it was fixed at (the starting value, rho).

If philosophy = "frequentist" a list equivalent to the output optimr with primary class optimrml and secondary class BiProbitPartialf.

Author(s)

BiProbitPartial was written by Michael Guggisberg. The majority of the MCMC estimation was written by Amrit Romana based on Rajbhandari (2014). The development of this package was partially funded by the Institute for Defense Analyses (IDA).

References


Examples

data('Mroz87', package = 'sampleSelection')
Mroz87$Z = Mroz87$lfp*(Mroz87$wage >= 5)

f1 = BiProbitPartial(Z ~ educ + age + kids5 + kids618 + nwifeinc | educ + exper + city,
                   data = Mroz87, philosophy = "frequentist")
summary(f1)

# Use the estimates from the frequentist philosophy as starting values
b1 = BiProbitPartial(Z ~ educ + age + kids5 + kids618 + nwifeinc | educ + exper + city,
                   data = Mroz87, philosophy = "bayesian",
                   control = list(beta = f1$par[1:(length(f1$par)-1)], rho = tail(f1$par,1)))
summary(b1)

## Not run: #The example used in the package sampleSelection is likely unidentified for
## this model
f2 = BiProbitPartial(Z ~ educ + age + kids5 + kids618 + nwifeinc | educ,
                   data = Mroz87, philosophy = "frequentist") #crashes
summary(f2) #crashes (f2 non-existent)

# Bayesian methods typically still work for unidentified models
b2 = BiProbitPartial(Z ~ educ + age + kids5 + kids618 + nwifeinc | educ,
                   data = Mroz87, philosophy = "bayesian",
                   control = list(beta = f1$par[1:(length(f1$par)-3)], rho = tail(f1$par,1)))
save(b2)

## End(Not run)

---

`grad1`

*Gradient of bivariate probit with partial observability*

**Description**

Gradient of bivariate probit with partial observability

**Usage**

```r
grad1(theta, X1, X2, Z, rho = 0, p = NULL, summed = T, fixrho = F)
```

**Arguments**

- `theta` numeric vector of dimension equal to that of the free parameter space
- `X1` numeric matrix of covariates for the first equation
- `X2` numeric matrix of covariates for the second equation
- `Z` numeric matrix or column vector of response observations
- `rho` numeric value for rho if fixed
llhood1

\[ p \]
numeric precomputed probabilities of \( Pr(Y1=1,Y2=1) \)

\[ \text{summed} \]
logical if the gradient observations should be summed

\[ \text{fixrho} \]
logical if rho should be fixed

**Value**

if summed is TRUE then the function returns the numeric column sum of the gradient matrix, else it returns a numeric vector with each entry a value of the gradient vector

---

**llhood1**

*log likelihood of bivariate probit with partial observability*

**Description**

log likelihood of bivariate probit with partial observability

**Usage**

\[ \text{llhood1}(\text{theta}, X1, X2, Z, \rho = 0, p = \text{NULL}, \text{summed} = T, \text{fixrho} = F) \]

**Arguments**

- **theta**
  numeric vector of dimension equal to that of the free parameter space
- **X1**
  numeric matrix of covariates for the first equation
- **X2**
  numeric matrix of covariates for the second equation
- **Z**
  numeric matrix or column vector of response observations
- **rho**
  numeric value for rho if fixed
- **p**
  numeric precomputed probabilities of \( Pr(Y1=1,Y2=1) \)
- **summed**
  logical if the log likelihood observations should be summed
- **fixrho**
  logical if rho should be fixed

**Value**

if summed is TRUE then the function returns the numeric sum of the likelihood vector, else it returns a numeric vector with each entry a value of the likelihood vector
MCMC1

MCMC algorithm to sample from bivariate probit with partial observability

Description

MCMC1() produces MCMC draws from the posterior of the bivariate probit with partial observability. It does not perform input validation. It is recommended to use BiProbitPartial instead of this function. BiProbitPartial performs input validation and then calls this function if philosophy == "bayesian".

Usage

MCMC1(X1, X2, Z, beta1, beta2, rho, fixrho, S, beta0, B0inv, rho0, v0, nu, P, tauSq, seed)

Arguments

X1 a matrix of covariates for the first equation
X2 a matrix of covariates for the second equation
Z a matrix of response values
beta1 a matrix of starting values for beta1
beta2 a matrix of starting values for beta2
rho a numeric starting value for rho
fixrho a logical determining if rho is fixed
S a numeric for the number of MCMC iterations
beta0 a matrix of the beta prior mean parameter
B0inv a matrix of the inverse of beta prior covariance parameter
rho0 a numeric for the mu prior parameter for rho
v0 a numeric for the Sigma prior parameter for rho
nu a numeric for MCMC tuning parameter 1
P a numeric for MCMC tuning parameter 2
tauSq a numeric for MCMC tuning parameter 3
seed a numeric seed for determining the random draw sequence

Value

a matrix of MCMC draws
predict.BiProbitPartialb

predict method for class 'BiProbitPartialb'

Description

Note this produces a Bayesian posterior predictive distribution. This accounts for estimation uncer-
tainty. If you desire a simple prediction that does not account for estimation uncertainty then the
frequentist philosophy should be used. If nchains is greater than 1 then the chains are combined.

Usage

## S3 method for class 'BiProbitPartialb'
predict(object, newdata, k1, k2,
      mRule = c(0.5, 0.5), jRule = NULL, ...)

Arguments

object a object of class BiProbitPartialb
newdata a matrix of column dimension k1 + k2 where the first k1 columns correspond
to the predictors of the first equations and the second k2 columns correspond
to predictors of the second equation. If intercepts were used they need to be
explicitly input.
k1 a numeric declaring the number of covariates (including intercept) in the first
equation
k2 a numeric declaring the number of covariates (including intercept) in the second
equation
mRule a vector of length 1 or 2. This is the marginal decision rule for classifying the
outcomes for stages 1 and 2. Stage 1 is classified as 1 if the probability of stage
1 being 1 is greater than or equal to mRule[1]. Likewise for stage 2. If length
of mRule is 1 then that value is recycled. The values of mRule must be between
0 and 1. The default value is mRule = c(0.5,0.5).
jRule an optional numerical value between 0 and 1. If specified then the observable
outcome (both stages being 1) is 1 if the joint probability of both stages being
1 is greater than jRule. If jRule is unspecified or set to NULL then the observ-
able outcome is the product of the marginal outcomes. The default value is
jRule = NULL. Note, if jRule is specified then the observable outcome might
not equal the product of stages 1 and 2.
... unused

Value

method predict.BiProbitPartial returns a data.frame with columns

| linPredict1 | Predicted mean of the first stage latent outcome. This is typically not interesting for a Bayesian analysis. |
linPredict2 Predicted mean of the second stage latent outcome. This is typically not interesting for a Bayesian analysis.

p1 Probability the outcome of the first stage is 1

p.1 Probability the outcome of the second stage is 1

p00 Probability the outcome of both stages is 0

p01 Probability the outcome of the first stage is 0 and the second stage is 1

p10 Probability the outcome of stage 1 is 1 and stage 2 is 0

p11 Probability the outcome of both stages are 1

yHat1 Classification of the outcome for stage 1. This value is 1 if \( p_1 \geq mRule[1] \) and 0 else

yHat2 Classification of the outcome for stage 2. This value is 1 if \( p_2 \geq mRule[2] \) and 0 else

ZHat Classification of the observable outcome. If jRule is specified then this value is 1 if \( p_{12} \geq jRule \) and 0 else. If jRule is unspecified then this value is the element-wise product of yHat1 and yHat2.

Examples

```r
# Perform a prediction with the same covariates the model is estimated with

mrozX7Dz = mrozX7Dlfp*(mrozX7Dwage >= 5)

# Run the frequentist version first to get starting values
f1 = BiProbitPartial(Z ~ educ + age + kids5 + kids618 + nwifeinc | educ + exper + city,
                      data = Mroz87, philosophy = "frequentist")

b1 = BiProbitPartial(Z ~ educ + age + kids5 + kids618 + nwifeinc | educ + exper + city,
                      data = Mroz87, philosophy = "bayesian",
                      control = list(beta = f1$par[1:(length(f1$par)-1)], rho = tail(f1$par,1)))

library(Formula)
eqn = Formula::Formula(~ educ + age + kids5 + kids618 + nwifeinc | educ + exper + city)
matrix1 = model.matrix(eqn, lhs = 0, rhs=1, data= Mroz87)
matrix2 = model.matrix(eqn, lhs = 0, rhs=2, data= Mroz87)
newdat = cbind(matrix1,matrix2)
preds1 = predict(b1,newdat,k1 = dim(matrix1)[2],k2 = dim(matrix2)[2])
head(preds1)
preds2 = predict(b1,newdat,k1 = dim(matrix1)[2],k2 = dim(matrix2)[2], jRule = .25)

# Compare predicted outcome with realized outcome
head(cbind(Mroz87$Z,preds1$ZHat,preds2$ZHat),20)
```
predict.BiProbitPartialf

predict method for class 'BiProbitPartialf'

Description

Note, this is a simple frequentist prediction and does not account for estimation uncertainty. If one wants to account for estimation uncertainty it is recommended to use the Bayesian philosophy.

Usage

```r
## S3 method for class 'BiProbitPartialf'
predict(object, newdata, mRule = c(0.5, 0.5),
         jRule = NULL, ...)
```

Arguments

- `object`: a object of class `BiProbitPartialf`
- `newdata`: a matrix of column dimension `k1 + k2` where the first `k1` columns correspond to the predictors of the first equations and the second `k2` columns correspond to predictors of the second equation. If intercepts were used they need to be explicitly input.
- `mRule`: a vector of length 1 or 2. This is the marginal decision rule for classifying the outcomes for stages 1 and 2. Stage 1 is classified as 1 if the probability of stage 1 being 1 is greater than or equal to `mRule[1]`. Likewise for stage 2. If length of `mRule` is 1 then that value is recycled. The values of `mRule` must be between 0 and 1. The default value is `mRule = c(0.5, 0.5)`.
- `jRule`: an optional numerical value between 0 and 1. If specified then the observable outcome (both stages being 1) is 1 if the joint probability of both stages being 1 is greater than `jRule`. If `jRule` is unspecified or set to `NULL` then the observable outcome is the product of the marginal outcomes. The default value is `jRule = NULL`. Note, if `jRule` is specified then the observable outcome might not equal the product of stages 1 and 2.
- `...`: unused

Value

Method `predict.BiProbitPartialf` returns a data.frame with columns

- `linPredict1`: Predicted mean of the first stage latent outcome
- `linPredict2`: Predicted mean of the second stage latent outcome
- `p1`: Probability the outcome of the first stage is 1
- `p1_1`: Probability the outcome of the second stage is 1
- `p00`: Probability the outcome of both stages is 0
- `p01`: Probability the outcome of the first stage is 0 and the second stage is 1
**p10** Probability the outcome of stage 1 is 1 and stage 2 is 0

**p11** Probability the outcome of both stages are 1

**yHat1** Classification of the outcome for stage 1. This value is 1 if \( p_1 > m_{\text{Rule}[1]} \) and 0 else

**yHat2** Classification of the outcome for stage 2. This value is 1 if \( p_2 > m_{\text{Rule}[2]} \) and 0 else

**ZHat** Classification of the observable outcome. If \( j_{\text{Rule}} \) is specified then this value is 1 if \( p_{1R} > j_{\text{Rule}} \) and 0 else. If \( j_{\text{Rule}} \) is unspecified then this value is the element-wise product of \( y_{\text{Hat}1} \) and \( y_{\text{Hat}2} \).

**Examples**

```r
##
# Perform a prediction with the same covariates the model is estimated with
##

data('Mroz87', package = 'sampleSelection')
Mroz87$Z = Mroz87$flp*(Mroz87$wage > 5)

f1 = BiProbitPartial(Z ~ educ + age + kids5 + kids618 + nwifeinc | educ + exper + city,
data = Mroz87, philosophy = "frequentist")

library(Formula)
eqn = Formula::Formula(~ educ + age + kids5 + kids618 + nwifeinc | educ + exper + city)
matrix1 = model.matrix(eqn, lhs = 0, rhs=1, data= Mroz87)
matrix2 = model.matrix(eqn, lhs = 0, rhs=2, data= Mroz87)
newdat = cbind(matrix1,matrix2)
preds1 = predict(f1,newdat)
head(preds1)
preds2 = predict(f1,newdat, jRule = .25)

# Compare predicted outcome with realized outcome
head(cbind(Mroz87$Z,preds1$ZHat,preds2$ZHat),20)
```

---

**SimDat**

This is data to be included in my package

---

**Description**

Simulated data of 10,000 observations from a multivariate normal distribution. The true coefficients for equation 1 are 0 and 1. The true coefficients for equation 2 are 0 and -1. The true \( \rho \) is 0.

**Author(s)**

Michael Guggisberg <mguggisb@ida.org>
Summary method for class 'optimrml'

Usage

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

Arguments

- `object` Object of class `optimrml`
- `...` unused

Value

matrix summary of estimates. The columns are

- **Estimate** Maximum likelihood point estimate
- **Std. Error** Asymptotic standard error estimate of maximum likelihood point estimators using numerical hessian
- **z value** z value for zero value null hypothesis using asymptotic standard error estimate
- **Pr(>|z|)** P value for a two sided null hypothesis test using the z value

Examples

```r
data('Mroz87', package = 'sampleSelection')
Mroz87$Z = Mroz87$lfp*(Mroz87$wage >= 5)

f1 = BiProbitPartial(Z ~ educ + age + kids5 + kids618 + nwifeinc | educ + exper + city, 
  data = Mroz87, philosophy = "frequentist")
summary(f1)

b1 = BiProbitPartial(Z ~ educ + age + kids5 + kids618 + nwifeinc | educ + exper + city, 
  data = Mroz87, philosophy = "bayesian", 
  control = list(beta = f1$par[1:(length(f1$par)-1)], rho = tail(f1$par,1)))
summary(b1)
```
Index

*Topic data
  SimDat, 12
  _PACKAGE (BiProbitPartial-package), 2

as.data.frame, 3

BiProbitPartial, 2
BiProbitPartial-package, 2

Formula, 3

grad1, 6

llhood1, 7

MCMC1, 8

na.exclude, 3
na.fail, 3
na.omit, 3

optimr, 3–5
options, 3

predict.BiProbitPartialb, 9
predict.BiProbitPartialf, 11

SimDat, 12
summary.optimrml, 13