Package ‘endogeneity’

October 13, 2022

Type Package
Title Recursive Two-Stage Models to Address Endogeneity
Version 2.0.1
Date 2022-02-02
Author Jing Peng
Maintainer Jing Peng <jing.peng@uconn.edu>
Description Various recursive two-stage models to address the endogeneity issue of treatment variables in observational study or mediators in experiments. The details of the models are discussed in Peng (2022) <doi:10.2139/ssrn.3494856>.
License GPL (>= 3)
Encoding UTF-8
Imports pbivnorm, maxLik, statmod, MASS
RoxygenNote 7.1.2
NeedsCompilation no
Repository CRAN
Date/Publication 2022-02-03 09:10:05 UTC

R topics documented:

bilinear ................................................................. 2
biprobit ................................................................. 3
biprobit_latent ....................................................... 4
biprobit_partial ..................................................... 6
endogeneity ............................................................ 8
pln ................................................................. 9
pln_linear ............................................................ 10
pln_probit ............................................................ 12
probit_linear ........................................................ 13
probit_linear_latent ............................................... 15
probit_linear_partial ............................................... 16

Index 19
bilinear

Recursive Bivariate Linear Model

Description

Estimate two linear models with bivariate normally distributed error terms. This command still works if the first-stage dependent variable is not a regressor in the second stage. The identification of a recursive bilinear model requires an instrument for the first dependent variable.

Usage

```r
bilinear(
    form1,
    form2,
    data = NULL,
    par = NULL,
    method = "BFGS",
    verbose = 0,
    accu = 10000
)
```

Arguments

- `form1`: Formula for the first linear model
- `form2`: Formula for the second linear model
- `data`: Input data, a data frame
- `par`: Starting values for estimates
- `method`: Optimization algorithm. Default is BFGS
- `verbose`: Level of output during estimation. Lowest is 0.
- `accu`: 1e12 for low accuracy; 1e7 for moderate accuracy; 10.0 for extremely high accuracy. See optim

Value

A list containing the results of the estimated model

References


See Also

Other endogeneity: `biprobit_latent()`, `biprobit_partial()`, `biprobit()`, `pln_linear()`, `pln_probit()`, `probit_linear_latent()`, `probit_linear_partial()`, `probit_linear()`
Examples

```r
library(MASS)
N = 2000
rho = -0.5
set.seed(1)

x = rbinom(N, 1, 0.5)
z = rnorm(N)

e = mvrnorm(N, mu=c(0,0), Sigma=matrix(c(1,rho,rho,1), nrow=2))
e1 = e[,1]
e2 = e[,2]

y1 = -1 + x + z + e1
y2 = -1 + x + y1 + e2

est = bilinear(y1~x+z, y2~x+y1)
est$estimates
```

biprobit  

Recusrive Bivariate Probit Model

Description

Estimate two probit models with bivariate normally distributed error terms. This command still works if the first-stage dependent variable is not a regressor in the second stage.

Usage

```r
biprobit(
  form1,
  form2,
  data = NULL,
  par = NULL,
  method = "BFGS",
  verbose = 0,
  accu = 10000
)
```

Arguments

- `form1`: Formula for the first probit model
- `form2`: Formula for the second probit model
- `data`: Input data, a data frame
- `par`: Starting values for estimates
- `method`: Optimization algorithm. Default is BFGS
- `verbose`: Level of output during estimation. Lowest is 0.
biprobit_latent

Value

A list containing the results of the estimated model

References


See Also

Other endogeneity: bilinear(), biprobit_latent(), biprobit_partial(), pln_linear(), pln_probit(), probit_linear_latent(), probit_linear_partial(), probit_linear()

Examples

```r
library(MASS)
N = 2000
rho = -0.5
set.seed(1)
x = rbinom(N, 1, 0.5)
z = rnorm(N)
e = mvrnorm(N, mu=c(0,0), Sigma=matrix(c(1,rho,rho,1), nrow=2))
e1 = e[,1]
e2 = e[,2]
y1 = as.numeric(1 + x + z + e1 > 0)
y2 = as.numeric(1 + x + z + y1 + e2 > 0)
est = biprobit(y1~x+z, y2~x+z+y1)
est$estimates
```

biprobit_latent

**Recursive Bivariate Probit Model with Latent First Stage**

Description

Estimate two probit models with bivariate normally distributed error terms, in which the dependent variable of the first stage model is unobserved. The identification of this model is weak if the first-stage does not include regressors that are good predictors of the first-stage dependent variable.
Usage

biprobity latent(
  form1, 
  form2, 
  data = NULL, 
  EM = FALSE, 
  par = NULL, 
  method = "BFGS", 
  verbose = 0, 
  accu = 10000, 
  maxIter = 500, 
  tol = 1e-05, 
  tol_LL = 1e-06 
)

Arguments

  form1  Formula for the first probit model, in which the dependent variable is unob-
          served. Use a formula like ~x to avoid specifying the dependent variable.
  form2  Formula for the second probit model, the latent dependent variable of the first
          stage is automatically added as a regressor in this model
  data   Input data, a data frame
  EM     Whether to maximize likelihood use the Expectation-Maximization (EM) algo-
          rithm.
  par    Starting values for estimates
  method Optimization algorithm. Default is BFGS
  verbose Level of output during estimation. Lowest is 0.
  accu   1e12 for low accuracy; 1e7 for moderate accuracy; 10.0 for extremely high ac-
          curacy. See optim
  maxIter max iterations for EM algorithm
  tol    tolerance for convergence of EM algorithm
  tol_LL tolerance for convergence of likelihood

Value

A list containing the results of the estimated model

References

Peng, Jing. (2022) Identification of Causal Mechanisms from Randomized Experiments: A Frame-
work for Endogenous Mediation Analysis. Information Systems Research (Forthcoming), Available
at SSRN: https://ssrn.com/abstract=3494856

See Also

Other endogeneity: bilinear(), biprobity_partial(), biprobity(), pln_linear(), pln_probit(),
probit_linear_latent(), probit_linear_partial(), probit_linear()
Examples

```r
library(MASS)
N = 2000
rho = -0.5
set.seed(1)

x = rbinom(N, 1, 0.5)
z = rnorm(N)

e = mvrnorm(N, mu=c(0,0), Sigma=matrix(c(1,rho,rho,1), nrow=2))
e1 = e[,1]
e2 = e[,2]

y1 = as.numeric(1 + x + z + e1 > 0)
y2 = as.numeric(1 + x + z + y1 + e2 > 0)

est = biprobit(y1~x+z, y2~x+z+y1)
est$estimates

est_latent = biprobit_latent(~x+z, y2~x+z)
est_latent$estimates
```

biprobit_partial

Recursive Bivariate Probit Model with Partially Observed First Stage

Description

Estimate two probit models with bivariate normally distributed error terms, in which the dependent variable of the first stage model is partially observed (or unobserved)

Usage

```r
biprobit_partial(
  form1,
  form2,
  data = NULL,
  EM = FALSE,
  par = NULL,
  method = "BFGS",
  verbose = 0,
  accu = 10000,
  maxIter = 500,
  tol = 1e-05,
  tol_LL = 1e-06
)
```
biprobite\_partial

Arguments

form1 Formula for the first probit model, in which the dependent variable is partially observed.

form2 Formula for the second probit model, the partially observed dependent variable of the first stage is automatically added as a regressor in this model (do not add manually)

data Input data, a data frame

EM Whether to maximize likelihood use the Expectation-Maximization (EM) algorithm.

par Starting values for estimates

method Optimization algorithm. Default is BFGS

verbose Level of output during estimation. Lowest is 0.

accu 1e12 for low accuracy; 1e7 for moderate accuracy; 10.0 for extremely high accuracy. See optim

maxIter max iterations for EM algorithm

tol tolerance for convergence of EM algorithm

tol\_LL tolerance for convergence of likelihood

Value

A list containing the results of the estimated model

References


See Also

Other endogeneity: bilinear(), biprobite\_latent(), biprobite(), pln\_linear(), pln\_probit(), probit\_linear\_latent(), probit\_linear\_partial(), probit\_linear()

Examples

library(MASS)
N = 5000
rho = -0.5
set.seed(1)
x = rbinom(N, 1, 0.5)
z = rnorm(N)
e = mvrnorm(N, mu=c(0,0), Sigma=matrix(c(1,rho,rho,1), nrow=2))
e1 = e[,1]
e2 = e[,2]
endogeneity

Recursive two-stage models to address endogeneity

Description

This package supports various recursive two-stage models to address the endogeneity issue. The details of the implemented models are discussed in Peng (2022). In a recursive two-stage model, the dependent variable of the first stage is an endogenous regressor in the second stage. The dependent variable of the second stage is the outcome of interest. The endogeneity is captured by the correlation in the error terms of the two stages.

Recursive two-stage models can be used to address the endogeneity of treatment variables in observational study and the endogeneity of mediators in experiments.

The first-stage supports linear model, probit model, and Poisson lognormal model. The second-stage supports linear and probit models. These models can be used to address the endogeneity of continuous, binary, and count variables. When the endogenous variable is binary, it can be unobserved or partially unobserved, but the identification can be weak.

Functions

bilinear: recursive bivariate linear model

biprobit: recursive bivariate probit model

biprobit_latent: recursive bivariate probit model with latent first stage

biprobit_partial: recursive bivariate probit model with partially observed first stage

probit_linear: recursive probit-linear or linear-probit model

probit_linear_latent: recursive probit-linear model with latent first stage

```r
y1 = as.numeric(1 + x + 3*z + e1 > 0)
y2 = as.numeric(1 + x + z + y1 + e2 > 0)
est = biprobit(y1~x+z, y2~x+z+y1)
est$estimates
observed_pct = 0.2
yp = y1
yp[sample(N, N*(1-observed_pct))] = NA
est_partial = biprobit_partial(yp~x+z, y2~x+z)
est_partial$estimates
```
probit_linear_partial: recursive probit-linear model with partially observed first stage

pln: Poisson lognormal (PLN) model

pln_linear: recursive PLN-linear model

pln_probit: recursive PLN-probit model

References

---

pln Poisson Lognormal Model

Description
Estimate a Poisson model with a log-normally distributed heterogeneity term. Also referred to as Poisson-Normal model.

Usage
pln(form, data = NULL, par = NULL, method = "BFGS", init = c("zero", "unif", "norm", "default")[4], H = 20, verbose = 0, accu = 10000)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>form</td>
<td>Formula</td>
</tr>
<tr>
<td>data</td>
<td>Input data, a data frame</td>
</tr>
<tr>
<td>par</td>
<td>Starting values for estimates</td>
</tr>
<tr>
<td>method</td>
<td>Optimization algorithm.</td>
</tr>
<tr>
<td>init</td>
<td>Initialization method</td>
</tr>
<tr>
<td>H</td>
<td>Number of quadrature points</td>
</tr>
</tbody>
</table>
pln_linear

Recursive PLN-Linear Model

Description
Estimate a Poisson Lognormal model (first-stage) and a linear model (second-stage) with bivariate normally distributed error terms. This command still works if the first-stage dependent variable is not a regressor in the second stage.

Usage
pln_linear(
  form_pln,
  form_linear,
  data = NULL,
  par = NULL,
  method = "BFGS",
  init = c("zero", "unif", "norm", "default")[4],
  H = 20,
  verbose = 0,
  accu = 10000
)
Arguments

- **form_pln**: Formula for the first-stage Poisson lognormal model
- **form_linear**: Formula for the second-stage linear model
- **data**: Input data, a data frame
- **par**: Starting values for estimates
- **method**: Optimization algorithm.
- **init**: Initialization method
- **H**: Number of quadrature points
- **verbose**: Level of output during estimation. Lowest is 0.
- **accu**: 1e12 for low accuracy; 1e7 for moderate accuracy; 10.0 for extremely high accuracy. See optim

Value

A list containing the results of the estimated model

References


See Also

Other endogeneity: `bilinear()`, `biprobit_latent()`, `biprobit_partial()`, `biprobit()`, `pln_probit()`, `probit_linear_latent()`, `probit_linear_partial()`, `probit_linear()`

Examples

```r
library(MASS)
N = 1000
rho = -0.5
set.seed(1)

x = rbinom(N, 1, 0.5)
z = rnorm(N)

e = mvrnorm(N, mu=c(0,0), Sigma=matrix(c(1,rho,rho,1), nrow=2))
e1 = e[,1]
e2 = e[,2]

y1 = rpois(N, exp(1 + x + z + e1))
y2 = 1 + x + y1 + e2

est = pln_linear(y1~x+z, y2~x+y1)
est$estimates
```
pln_probit

Recursive PLN-Probit Model

Description
Estimate a Poisson Lognormal model (first-stage) and a probit model (second-stage) whose error
terms are bivariate normally distributed. This model still works if the first-stage dependent variable
is not a regressor in the second stage.

Usage
pln_probit(
  form_pln,
  form_probit,
  data = NULL,
  par = NULL,
  method = "BFGS",
  init = c("zero", "unif", "norm", "default")[4],
  H = 20,
  verbose = 0,
  accu = 10000
)

Arguments
  form_pln    Formula for the first-stage Poisson lognormal model
  form_probit Formula for the second-stage probit model
  data        Input data, a data frame
  par          Starting values for estimates
  method       Optimization algorithm. Without gradient, NM is much faster than BFGS
  init         Initialization method
  H            Number of quadrature points
  verbose      Level of output during estimation. Lowest is 0.
  accu         1e12 for low accuracy; 1e7 for moderate accuracy; 10.0 for extremely high ac-
                curacy. See optim

Value
A list containing the results of the estimated model

References
Peng, Jing. (2022) Identification of Causal Mechanisms from Randomized Experiments: A Frame-
work for Endogenous Mediation Analysis. Information Systems Research (Forthcoming), Available
at SSRN: https://ssrn.com/abstract=3494856
See Also

Other endogeneity: `bilinear()`, `biprobit_latent()`, `biprobit_partial()`, `biprobit()`, `pln_linear()`, `probit_linear_latent()`, `probit_linear_partial()`, `probit_linear()`

Examples

```r
library(MASS)
N = 1000
rho = -0.5
set.seed(1)
x = rbinom(N, 1, 0.5)
z = rnorm(N)
e = mvrnorm(N, mu=c(0,0), Sigma=matrix(c(1,rho,rho,1), nrow=2))
e1 = e[,1]
e2 = e[,2]
y1 = rpois(N, exp(-1 + x + z + e1))
y2 = as.numeric(1 + x + z + log(1+y1) + e2 > 0)
est = pln_probit(y1~x+z, y2~x+z+log(1+y1))
est$estimates
```

### probit_linear

**Recursive Probit-Linear Model**

**Description**

Estimate probit and linear models with bivariate normally distributed error terms. This command supports two models with opposite first and second stages.

1) Recursive Probit-Linear: the endogenous treatment effect model
2) Recursive Linear-Probit: the ivprobit model. The identification of this model requires an instrument.

This command still works if the first-stage dependent variable is not a regressor in the second stage.

**Usage**

```r
probit_linear(
    form_probit,  # Probit model formula
    form_linear,  # Linear model formula
    data = NULL,  # Data frame
    par = NULL,    # Initial parameter values
    method = "BFGS",  # Optimization method
    init = c("zero", "unif", "norm", "default")[4],  # Initial values
    verbose = 0,  # Verbosity level
    accu = 10000  # Accuracy level
)
```
**Arguments**

- `form_probit`: Formula for the probit model
- `form_linear`: Formula for the linear model
- `data`: Input data, a data frame
- `par`: Starting values for estimates
- `method`: Optimization algorithm. Default is BFGS
- `init`: Initialization method
- `verbose`: Level of output during estimation. Lowest is 0.
- `accu`: 1e12 for low accuracy; 1e7 for moderate accuracy; 10.0 for extremely high accuracy. See optim

**Value**

A list containing the results of the estimated model

**References**


**See Also**

Other endogeneity: `bilinear()`, `biprobit_latent()`, `biprobit_partial()`, `biprobit()`, `pln_linear()`, `pln_probit()`, `probit_linear_latent()`, `probit_linear_partial()`

**Examples**

```r
library(MASS)
N = 2000
rho = -0.5
set.seed(1)
x = rbinom(N, 1, 0.5)
z = rnorm(N)
e = mvrnorm(N, mu=c(0,0), Sigma=matrix(c(1,rho,rho,1), nrow=2))
e1 = e[,1]
e2 = e[,2]
y1 = as.numeric(1 + x + z + e1 > 0)
y2 = 1 + x + z + y1 + e2
est = probit_linear(y1~x+z, y2~x+z+y1)
est$estimates
```
probit_linear_latent  Recursive Probit-Linear Model with Latent First Stage

Description
The first stage is a probit model with unobserved dependent variable, the second stage is a linear model that includes the first-stage dependent variable as a regressor.

Usage
probit_linear_latent(
  form_probit,
  form_linear,
  data = NULL,
  EM = TRUE,
  par = NULL,
  method = "BFGS",
  verbose = 0,
  accu = 10000,
  maxIter = 500,
  tol = 1e-06,
  tol_LL = 1e-08
)

Arguments
form_probit  Formula for the first-stage probit model, in which the dependent variable is latent
form_linear  Formula for the second stage linear model. The latent dependent variable of the first stage is automatically added as a regressor in this model
data  Input data, a data frame
EM  Whether to maximize likelihood use the Expectation-Maximization algorithm. EM is slower but more robust
par  Starting values for estimates
method  Optimization algorithm. Default is BFGS
verbose  Level of output during estimation. Lowest is 0.
accu  1e12 for low accuracy; 1e7 for moderate accuracy; 10.0 for extremely high accuracy. See optim
maxIter  max iterations for EM algorithm
tol  tolerance for convergence of EM algorithm
tol_LL  tolerance for convergence of likelihood

Value
A list containing the results of the estimated model
References


See Also

Other endogeneity: `bilinear()`, `biprobit_latent()`, `biprobit_partial()`, `biprobit()`, `pln_linear()`, `pln_probit()`, `probit_linear()`, `probit_linear_partial()`, `probit_linear()`

Examples

```r
library(MASS)
N = 2000
rho = -0.5
set.seed(1)

x = rbinom(N, 1, 0.5)
z = rnorm(N)

e = mvrnorm(N, mu=c(0,0), Sigma=matrix(c(1,rho,rho,1), nrow=2))
e1 = e[,1]
e2 = e[,2]

y1 = as.numeric(1 + x + z + e1 > 0)
y2 = 1 + x + z + y1 + e2
est = probit_linear(y1~x+z, y2~x+z+y1)
est$estimates

est_latent = probit_linear_latent(~x+z, y2~x+z)
est_latent$estimates
```

---

**probit_linear_partial**  
Recursive Probit-Linear Model with Partially Observed First Stage

Description

The first stage is a probit model with partially observed (or unobserved) dependent variable, the second stage is a linear model that includes the first-stage dependent variable as a regressor.

Usage

```r
probit_linear_partial(
  form_probit,
  form_linear,
  data = NULL,
  EM = TRUE,
)```
Arguments

form_probit  Formula for the first-stage probit model, in which the dependent variable is partially observed
form_linear Formula for the second stage linear model. The partially observed dependent variable of the first stage is automatically added as a regressor in this model (do not add manually)
data Input data, a data frame
EM Whether to maximize likelihood use the Expectation-Maximization algorithm. EM is slower but more robust
par Starting values for estimates
method Optimization algorithm. Default is BFGS
verbose Level of output during estimation. Lowest is 0.
accu 1e12 for low accuracy; 1e7 for moderate accuracy; 10.0 for extremely high accuracy. See optim
maxIter max iterations for EM algorithm
tol tolerance for convergence of EM algorithm
tol_LL tolerance for convergence of likelihood

Value

A list containing the results of the estimated model

References


See Also

Other endogeneity: bilinear(), biprobit_latent(), biprobit_partial(), biprobit(), pln_linear(), pln_probit(), probit_linear_latent(), probit_linear()
Examples

```r
library(MASS)
N = 1000
rho = -0.5
set.seed(1)

x = rbinom(N, 1, 0.5)
z = rnorm(N)

e = mvrnorm(N, mu=c(0,0), Sigma=matrix(c(1,rho,rho,1), nrow=2))
e1 = e[,1]
e2 = e[,2]

y1 = as.numeric(1 + x + z + e1 > 0)
y2 = 1 + x + z + y1 + e2
est = probit_linear(y1~x+z, y2~x+z+y1)
est$estimates

observed_pct = 0.2
y1p = y1
y1p[sample(N, N*(1-observed_pct))] = NA
est_latent = probit_linear_partial(y1p~x+z, y2~x+z)
est_latent$estimates
```
Index

* endogeneity
  bilinear, 2
  biprobit, 3
  biprobit_latent, 4
  biprobit_partial, 6
  pln_linear, 10
  pln_probit, 12
  probit_linear, 13
  probit_linear_latent, 15
  probit_linear_partial, 16
  bilinear, 2, 4, 5, 7, 11, 13, 14, 16, 17
  biprobit, 2, 3, 5, 7, 11, 13, 14, 16, 17
  biprobit_latent, 2, 4, 4, 7, 11, 13, 14, 16, 17
  biprobit_partial, 2, 4, 5, 6, 11, 13, 14, 16, 17

endogeneity, 8

pln, 9
  pln_linear, 2, 4, 5, 7, 10, 13, 14, 16, 17
  pln_probit, 2, 4, 5, 7, 11, 12, 14, 16, 17
  probit_linear, 2, 4, 5, 7, 11, 13, 16, 17
  probit_linear_latent, 2, 4, 5, 7, 11, 13, 14, 15, 17
  probit_linear_partial, 2, 4, 5, 7, 11, 13, 14, 16, 16