

Package ‘mprobit’

April 17, 2009

Version 0.9-2

Date 2006-07-27

Title Multivariate probit model for binary/ordinal response

Author Harry Joe <harry@stat.ubc.ca>, Laing Wei Chou and Hongbin Zhang

Maintainer Hongbin Zhang <hongbinzhang711@yahoo.com>

Depends R (>= 1.8.0)

Description Multivariate normal rectangle probabilities (positive exchangeable, general, approximations); MLE of regression and correlation parameters in the multivariate binary/ordinal probit models: exchangeable, AR(1), and unstructured correlation matrix.

LazyLoad yes

License GPL (>= 2)

Repository CRAN

Date/Publication 2006-07-27 14:28:36

R topics documented:

binaryar	2
binaryex	2
exchmvn	2
mprobit	4
mvn.deriv	7
mvnapp	11
ordinalex	13
ordprobit	13
ordprobit.univar	16
pmnorm	17

Index	19
--------------	-----------

 binaryar

Multivariate Binary Test Data Set 2

Description

A simulated data set with a binary response y , a covariate x , and a cluster group variable id . Used in examples for `mprobit` function. The values used for the simulation are: $\beta_0 = -0.3$, $\beta_1 = 0.4$, latent AR(1) correlation = 0.7

Usage

```
data(binaryar)
```

binaryex

Multivariate Binary Test Data Set 1

Description

A simulated data set with a binary response y , a covariate x , and a cluster group variable id . Used in examples for `mprobit` function. The values used for the simulation are: $\beta_0 = 0.3$, $\beta_1 = 0.2$, latent exchangeable correlation = 0.6

Usage

```
data(binaryex)
```

exchmvn

Exchangeable (positive) multivariate normal

Description

Rectangle probability and derivatives of positive exchangeable multivariate normal

Usage

```
exchmvn(lb, ub, rh, mu=0, scale=1, eps = 1.e-06)
exchmvn.deriv.margin(lb, ub, rh, k, ksign, eps = 1.e-06)
exchmvn.deriv.rho(lb, ub, rh, eps = 1.e-06)
```

Arguments

lb	vector of lower limits of integral/probability
ub	vector of upper limits of integral/probability
rh	correlation, rho
mu	mean vector
scale	standard deviation
eps	tolerance for numerical integration
k	margin for which derivative is to be taken, that is, deriv of exchmvn(lb,ub,rh) with respect to lb[k] or ub[k]; use exchmvn.deriv.rh for deriv of exchmvn(lb,ub,rh) with respect to rho
ksign	=-1 for deriv of exchmvn(lb,ub,rh) with respect to lb[k] =+1 for deriv of exchmvn(lb,ub,rh) with respect to ub[k]

Value

rectangle probability or a derivative

Author(s)

H. Joe, Statistics Department, UBC

References

Kotz S and Johnson NL (1972). Continuous Multivariate Distributions. Wiley, New York, page 48.

Examples

```
# The tests here show clearly what the function parameters are.
# step size for numerical derivatives (accuracy of exchmvn etc about 1.e-6)
heps = 1.e-4

cat("case 1: m=3\n")
m=3
a=c(-1,-1,-1)
b=c(2,1.5,1)
rh=.6
pr=exchmvn(a,b,rh)
cat("pr=exchmvn(avec,bvec,rh)=",pr,"\n")
cat("derivative wrt rho\n")
rh2=rh+heps
pr2=exchmvn(a,b,rh2)
drh.numerical= (pr2-pr)/heps
drh.analytic= exchmvn.deriv.rho(a,b,rh)
cat(" numerical: ", drh.numerical, ", analytic: ", drh.analytic,"\n")

cat("derivative wrt a_k,b_k, k=1,...,m,\n")
for(k in 1:m)
{ cat(" k=", k, " lower\n")
```

```

a2=a
a2[k]=a[k]+heps
pr2=exchmvn(a2,b,rh)
da.numerical = (pr2-pr)/heps
da.analytic= exchmvn.deriv.margin(a,b,rh,k,-1)
cat(" numerical: ", da.numerical, ", analytic: ", da.analytic,"\n")
cat(" k=", k, " upper\n")
b2=b
b2[k]=b[k]+heps
pr2=exchmvn(a,b2,rh)
db.numerical = (pr2-pr)/heps
db.analytic= exchmvn.deriv.margin(a,b,rh,k,1)
cat(" numerical: ", db.numerical, ", analytic: ", db.analytic,"\n")
}

cat("\ncase 2: m=5\n")
m=5
a=rep(-1,m)
b=c(2,1.5,1,1.5,2)
rh=.6
pr=exchmvn(a,b,rh)
cat("pr=exchmvn(avec,bvec,rh)=", pr, "\n")
cat("derivative wrt rho\n")
rh2=rh+heps
pr2=exchmvn(a,b,rh2)
drh.numerical= (pr2-pr)/heps
drh.analytic= exchmvn.deriv.rho(a,b,rh)
cat(" numerical: ", drh.numerical, ", analytic: ", drh.analytic,"\n")

cat("derivative wrt a_k,b_k, k=1,..., ",m, "\n")
for(k in 1:m)
{ cat(" k=", k, " lower\n")
a2=a
a2[k]=a[k]+heps
pr2=exchmvn(a2,b,rh)
da.numerical = (pr2-pr)/heps
da.analytic= exchmvn.deriv.margin(a,b,rh,k,-1)
cat(" numerical: ", da.numerical, ", analytic: ", da.analytic,"\n")
cat(" k=", k, " upper\n")
b2=b
b2[k]=b[k]+heps
pr2=exchmvn(a,b2,rh)
db.numerical = (pr2-pr)/heps
db.analytic= exchmvn.deriv.margin(a,b,rh,k,1)
cat(" numerical: ", db.numerical, ", analytic: ", db.analytic,"\n")
}

```

Description

Maximum Likelihood for Repeated Measures Multivariate Binary Probit: Exchangeable, AR(1) and Unstructured Correlation Matrices. Quasi-Newton minimization of negative log-likelihood is used with approximation of Joe (1995) for rectangle probabilities for AR(1) and unstructured correlation, and one-dimensional Romberg integration for exchangeable correlation.

Usage

```
mprobit(x, y, id, corstr="exch", iprint=0, startpar=0)
  or
mprobit.formula(formula, id, data, corstr="exch", iprint=0, startpar=0)
  or
mprobit.exch(x, y, id, iprint=0, startpar=0)
mprobit.ar(x, y, id, iprint=0, startpar=0)
mprobit.unstr(x, y, id, iprint=0, startpar=0)
```

Arguments

<code>x</code>	vector or matrix of explanatory variables. Each row corresponds to an observation and each column to a variable. The number of rows of <code>x</code> should equal the number of data values in <code>y</code> . Missing values are not allowed.
<code>y</code>	numeric vector containing the binary response (coded with values of 0,1). Missing values are not allowed.
<code>id</code>	group or cluster id, should be a vector of positive integers. If AR(1) correlation, records are assumed to be ordered within each cluster id. For unstructured correlation, the cluster size must be constant, and records are assumed to be ordered the same way with each cluster id (i.e., <i>j</i> th record within each cluster refers to a common time/condition for the repeated measurement). For AR(1) and exchangeable correlation, cluster size can vary. For the formula version, include the data structure.
<code>formula</code>	For the formula version of <code>mprobit</code> , a formula expression as for other regression models, of the form "response ~ predictors".
<code>data</code>	For the formula version of <code>mprobit</code> , the data frame which contains the variables in the formula, and the cluster id variable.
<code>corstr</code>	For <code>mprobit</code> as a front end to the other three functions, <code>corstr="exch"</code> means exchangeable correlation (the default); <code>corstr="ar"</code> means AR(1) correlation; <code>corstr="unstr"</code> means unstructured correlation (in which case cluster size must be a constant).
<code>iprint</code>	logical indicator, default is FALSE, for whether the iterations for numerical maximum likelihood should be printed.
<code>startpar</code>	initial parameter vector in the order: regression coefficients, correlation parameter(s). If not supplied, default = 0, <code>startpar</code> will be generated automatically.

Details

To get an initial version working, there are constraints: (a) For AR(1) and unstructured correlation, the maximum cluster size is 19 (although the joint probabilities get to be too small well before this

limit is reached); (b) The maximum total number of parameters (regression and latent correlation parameters combined is 23). So this means a smaller upper bound on the number of predictors for the unstructured correlation.

Also the performance of the quasi-Newton algorithm gets worse as the number of parameters increase, particular with the sample size (number of clusters) is too small.

The default starting point used by the code should usually be OK. If the returned cov=estimated covariance matrix is the identity matrix, then the quasi-Newton iterations did not finish cleanly with a gradient vector that is near zero. If this case, it would be useful to use `iprint=1` to print the iterations, and try different starting points in `startpar` to check on the sensitivity to the starting point. The SE estimates are better if the quasi-Newton iterations finish with gradient vector closer to zero.

Value

list of MLE of parameters and their associated standard errors, in the order of `b0,b1,b2...b`(number of covariates), `rho(s)`; order of rhos is `r12,r13,...,r23,...,r(d-1,d)` for unstructured

<code>negloglik</code>	value of negative log-likelihood, evaluated at MLE
<code>beta</code>	MLE of regression parameters
<code>rho</code>	MLE of latent correlation parameter for AR(1) and exchangeable correlation
<code>rhomat</code>	MLE of matrix of latent correlation parameter for unstructured correlation
<code>mle</code>	MLE of all parameters for unstructured correlation
<code>cov</code>	estimated covariance matrix of the parameters

Author(s)

H. Joe, Statistics Department, UBC, with assistance of Laing Wei Chou

References

- Ashford, JR and Sowden, RR (1970). Multivariate probit analysis. *Biometrics*, 26, 535-546.
- Chaganty NR and Joe H (2004). Efficiency of the generalised estimating equations for binary response. *J Royal Statist Soc B*, 66, 851-860.
- Joe, H (1995). Approximations to multivariate normal rectangle probabilities based on conditional expectations, *J Amer Stat Assoc*, 90, 957-964.

Examples

```
# first test data set
data(binaryex)
x=binaryex$x
y=binaryex$y
id=binaryex$id

# various ways of using mprobit are shown here
# exchangeable dependence
out.exch = mprobit.exch(x,y,id)
print(out.exch)
```

```

# AR(1) dependence
out.ar = mprobit(x, y, id, corstr="ar")
print(out.ar)

# unstructured correlation matrix
out.unstr = mprobit.formula(y~x, binaryex$id, data=binaryex, corstr="unstr")
print(out.unstr)

# second test data set
data(binaryar)
dat=binaryar
x=dat$x
y=dat$y
id=dat$id
x2=x*x
dat$x2=x2

# exchangeable dependence
out2.exch = mprobit.exch(cbind(x, x2), y, id)
print(out2.exch)

# AR(1) dependence
out2.ar = mprobit.formula(y~x+x2, dat$id, data=dat, corstr="ar")
print(out2.ar)

# varying cluster sizes
set.seed(123)
ncl=nrow(dat)/4
idel=sort(sample(ncl, ncl/4))
idel=idel*4
datsub=dat[-idel,]
out3.ar = mprobit.formula(y~x+x2, datsub$id, data=datsub, corstr="ar")
print(out3.ar)

```

mvn.deriv

Derivatives of Multivariate Normal Rectangle Probabilities

Description

Derivatives of Multivariate Normal Rectangle Probabilities based on Approximations

Usage

```

mvn.deriv.margin(lb, ub, mu, sigma, k, ksign, type=1, eps=1.e-05, nsim=0)
mvn.deriv.rho(lb, ub, mu, sigma, j1, k1, type=1, eps=1.e-05, nsim=0)

```

Arguments

lb	vector of lower limits of integral/probability
ub	vector of upper limits of integral/probability

mu	mean vector
sigma	covariance matrix, it is assumed to be positive-definite
type	indicator, type=1 refers to the first order approximation, type=2 is the second order approximation.
eps	accuracy/tolerance for bivariate marginal rectangle probabilities
nsim	an optional integer if random permutations are used in the approximation for dimension ≥ 6 ; nsim=2000 recommended for dim ≥ 6
k	margin for which derivative is to be taken, that is, deriv of mvnapp(lb,ub,mu,sigma) with respect to lb[k] or ub[k];
ksign	=-1 for deriv of mvnapp(lb,ub,mu,sigma) with respect to lb[k] =+1 for deriv of mvnapp(lb,ub,mu,sigma) with respect to ub[k]
j1	correlation for which derivative is to be taken, that is, deriv of mvnapp(lb,ub,mu,sigma) with respect to rho[j1,k1], where rho is a correlation corresponding to sigma
k1	See above explanation with j1

Value

derivative with respect to margin lb[k], ub[k], or correlation rho[j1][k1] corresponding to sigma matrix

Author(s)

H. Joe, Statistics Department, UBC

References

Joe, H. (1995) JASA TODO, complete

See Also

[mvnapp](#)

Examples

```
# step size for numerical derivatives (accuracy of mvnapp etc may be about 1.e-4 to 1.e-5)
heps = 1.e-3

cat("compare numerical and analytical derivatives based on mvnapp\n")

cat("\ncase 1: dim=3\n");
m=3
mu=rep(0,m)
a=c(0,0,0)
b=c(1,1.5,2)
rr=matrix(c(1,.3,.3,.3,1,.4,.3,.4,1),m,m)
pr=mvnapp(a,b,mu,rr)$pr
# not checking ifail returned from mvnapp
cat("pr=mvnapp(avec,bvec,mu=0,sigma=corrmat)=",pr,"\n")
```

```

cat("derivative wrt a_k,b_k, k=1,...,"m,"\n")
for(k in 1:m)
{ cat(" k=", k, " lower\n")
  a2=a
  a2[k]=a[k]+heps
  pr2=mvnapp(a2,b,mu,rr)$pr
  da.numerical = (pr2-pr)/heps
  da.analytic= mvn.deriv.margin(a,b,mu,rr,k,-1)$deriv
  cat(" numerical: ", da.numerical, ", analytic: ", da.analytic,"\n")
  cat(" k=", k, " upper\n")
  b2=b
  b2[k]=b[k]+heps
  pr2=mvnapp(a,b2,mu,rr)$pr
  db.numerical = (pr2-pr)/heps
  db.analytic= mvn.deriv.margin(a,b,mu,rr,k,1)$deriv
  cat(" numerical: ", db.numerical, ", analytic: ", db.analytic,"\n")
}

cat("derivative wrt rho(j,k)\n")
for(j in 1:(m-1))
{ for(k in (j+1):m)
  { cat(" (j,k)=", j,k,"\n")
    rr2=rr
    rr2[j,k]=rr[j,k]+heps
    rr2[k,j]=rr[k,j]+heps
    pr2=mvnapp(a,b,mu,rr2)$pr
    drh.numerical = (pr2-pr)/heps
    drh.analytic= mvn.deriv.rho(a,b,mu,rr2,j,k)$deriv
    cat(" numerical: ", drh.numerical, ", analytic: ", drh.analytic,"\n")
  }
}

#=====

cat("\ncase 2: dim=5\n");
m=5
mu=rep(0,m)
a=c(0,0,0,-1,-1)
b=c(1,1.5,2,2,2)
rr=matrix(c(1,.3,.3,.3,.4, .3,1,.4,.4,.4, .3,.4,1,.4,.4,
.3,.4,.4,1,.4, .4,.4,.4,.4,1),m,m)
pr=mvnapp(a,b,mu,rr)$pr
# not checking ifail returned from mvnapp
cat("pr=mvnapp(avec,bvec,mu=0,sigma=corrmat)=",pr,"\n")

cat("derivative wrt a_k,b_k, k=1,...,"m,"\n")
for(k in 1:m)
{ cat(" k=", k, " lower\n")
  a2=a
  a2[k]=a[k]+heps
  pr2=mvnapp(a2,b,mu,rr)$pr
  da.numerical = (pr2-pr)/heps

```

```

da.analytic= mvn.deriv.margin(a,b,mu,rr,k,-1)$deriv
cat("  numerical: ", da.numerical, ", analytic: ", da.analytic,"\n")
cat("  k=", k, " upper\n")
b2=b
b2[k]=b[k]+heps
pr2=mvnapp(a,b2,mu,rr)$pr
db.numerical = (pr2-pr)/heps
db.analytic= mvn.deriv.margin(a,b,mu,rr,k,1)$deriv
cat("  numerical: ", db.numerical, ", analytic: ", db.analytic,"\n")
}

cat("derivative wrt rho(j,k): first order approx\n")
for(j in 1:(m-1))
{ for(k in (j+1):m)
  { cat("  (j,k)=", j,k,"\n")
    rr2=rr
    rr2[j,k]=rr[j,k]+heps
    rr2[k,j]=rr[k,j]+heps
    pr2=mvnapp(a,b,mu,rr2)$pr
    drh.numerical = (pr2-pr)/heps
    drh.analytic= mvn.deriv.rho(a,b,mu,rr2,j,k)$deriv
    cat("  numerical: ", drh.numerical, ", analytic: ", drh.analytic,"\n")
  }
}

cat("\nsecond order approx\n")
pr=mvnapp(a,b,mu,rr,type=2)$pr
cat("pr=mvnapp(avec,bvec,mu=0,sigma=corrmat,type=2)=",pr,"\n")

cat("derivative wrt a_k,b_k, k=1,...,m,\n")
for(k in 1:m)
{ cat("  k=", k, " lower\n")
  a2=a
  a2[k]=a[k]+heps
  pr2=mvnapp(a2,b,mu,rr,type=2)$pr
  da.numerical = (pr2-pr)/heps
  da.analytic= mvn.deriv.margin(a,b,mu,rr,k,-1,type=2)$deriv
  cat("  numerical: ", da.numerical, ", analytic: ", da.analytic,"\n")
  cat("  k=", k, " upper\n")
  b2=b
  b2[k]=b[k]+heps
  pr2=mvnapp(a,b2,mu,rr,type=2)$pr
  db.numerical = (pr2-pr)/heps
  db.analytic= mvn.deriv.margin(a,b,mu,rr,k,1,type=2)$deriv
  cat("  numerical: ", db.numerical, ", analytic: ", db.analytic,"\n")
}

cat("derivative wrt rho(j,k): second order approx\n")
for(j in 1:(m-1))
{ for(k in (j+1):m)
  { cat("  (j,k)=", j,k,"\n")
    rr2=rr
    rr2[j,k]=rr[j,k]+heps

```

```

rr2[k, j]=rr[k, j]+heps
pr2=mvnapp(a, b, mu, rr2, type=2)$pr
drh.numerical = (pr2-pr)/heps
drh.analytic= mvn.deriv.rho(a, b, mu, rr2, j, k, type=2)$deriv
cat("  numerical: ", drh.numerical, ", analytic: ", drh.analytic, "\n")
}
}

```

mvnapp

MVN Rectangle Probabilities

Description

Approximation to multivariate normal rectangle probabilities using methods in Joe (1995), JASA

Usage

```
mvnapp(lb, ub, mu, sigma, type=1, eps=1.e-05, nsim=0)
```

Arguments

lb	vector of lower limits of integral/probability
ub	vector of upper limits of integral/probability
mu	mean vector
sigma	covariance matrix, it is assumed to be positive-definite
type	indicator, type=1 refers to the first order approximation, type=2 is the second order approximation.
eps	accuracy/tolerance for bivariate marginal rectangle probabilities
nsim	an optional integer if random permutations are used in the approximation for dimension ≥ 6 ; nsim=2000 recommended for dim ≥ 6

Value

prob	rectangle probability with approximation
esterr	indicator of accuracy in the approximation
ifail	= 0 if no problems >= 1 if problems from using Schervish's code in dimensions 2 to 4.

Author(s)

H. Joe, Statistics Department, UBC

References

Joe, H (1995). Approximations to multivariate normal rectangle probabilities based on conditional expectations. *J. Amer. Statist. Assoc.*, 90, 957-964.

Schervish, M.J. (1984). Multivariate normal probabilities with error bound. *Appl. Statist.*, 33, 81-94.

See Also

[pmnorm](#).

Examples

```
m<-2
rh<-0.5
a<-c(-1,-1)
b<-c(1,1)
mu<-rep(0,m)
s<-matrix(c(1,rh,rh,1),2,2)
print(pmnorm(a,b,mu,s))
print(mvnapp(a,b,mu,s))
print(mvnapp(a,b,mu,s,type=2))
print(mvnapp(a,b,mu,s,type=2,nsim=3))
```

```
m<-3
rh<-0.3
a<-c(-1,-1,-2)
b<-c(1,1,.5)
mu<-rep(0,m)
s<-matrix(c(1,.5,.6,.5,1,.7,.6,.7,1),3,3)
print(pmnorm(a,b,mu,s))
print(mvnapp(a,b,mu,s))
print(mvnapp(a,b,mu,s,type=2))
print(mvnapp(a,b,mu,s,type=2,nsim=3))
```

```
m<-4
rh<-0.1
a<-c(-1,-2.5,-2,-1.5)
b<-c(1.68,1.11,.5,.25)
mu<-rep(0,m)
s<-matrix(c(1,.5,.3,.4,.5,1,.5,.4,.3,.5,1,.4,.4,.4,.4,1),m,m)
print(pmnorm(a,b,mu,s))
print(mvnapp(a,b,mu,s))
print(mvnapp(a,b,mu,s,type=2))
print(mvnapp(a,b,mu,s,type=2,nsim=3))
```

```
m<-5
rh<-0.4
a<-rep(-1,m)
b<-rep(2,m)
mu<-rep(0,m)
s<-matrix(c(1,rh,rh,rh,rh,rh,1,rh,rh,rh,rh,rh,1,rh,rh,rh,rh,rh,1,
```

```

      rh, rh, rh, rh, rh, 1), m, m)
print(mvnapp(a, b, mu, s))
print(mvnapp(a, b, mu, s, type=2))
print(mvnapp(a, b, mu, s, type=2, nsim=3))

m<-6
a<-c(-1, -1, -1, -1.5, -1, -2)
b<-rep(7, m)
mu<-rep(0, m)
s<-matrix(c(1, rh, rh, rh, rh, rh, rh, rh, 1, rh, rh, rh, rh, rh, rh, 1, rh, rh, rh, rh, rh, rh, 1,
            rh, rh, rh, rh, rh, rh, 1, rh, rh, rh, rh, rh, rh, 1), m, m)
print(mvnapp(a, b, mu, s))
print(mvnapp(a, b, mu, s, type=2))
print(mvnapp(a, b, mu, s, type=2, nsim=3))

```

ordinalex

Multivariate Ordinal Test Data Set

Description

A simulated data set with an ordinal response y (3 categories), a covariate x , and a cluster group variable id . Used in examples for `ordprobit` function. The values used for the simulation are: `cutpt1=-0.5`, `cutpt2=0.3`, `beta1=0.4`, latent exchangeable correlation = 0.6

Usage

```
data(ordinalex)
```

ordprobit

Maximum Likelihood for Repeated Measures Multivariate Ordinal Probit Model

Description

Maximum Likelihood for Repeated Measures Multivariate Ordinal Probit: Exchangeable, AR(1) and Unstructured Correlation Matrices. Quasi-Newton minimization of negative log-likelihood is used with approximation of Joe (1995) for rectangle probabilities for AR(1) and unstructured correlation, and one-dimensional Romberg integration for exchangeable correlation.

Usage

```

ordprobit(x, y, id, corstr="exch", iprint=0, startpar=0)
  or
ordprobit.formula(formula, id, data, corstr="exch", iprint=0, startpar=0)
  or
ordprobit.exch(x, y, id, iprint=0, startpar=0)
ordprobit.ar(x, y, id, iprint=0, startpar=0)
ordprobit.unstr(x, y, id, iprint=0, startpar=0)

```

Arguments

<code>x</code>	vector or matrix of explanatory variables. Each row corresponds to an observation and each column to a variable. The number of rows of <code>x</code> should equal the number of data values in <code>y</code> , and there should be fewer columns than rows. Missing values are not allowed.
<code>y</code>	numeric vector containing the ordinal response. The values must be in the range 1,2,..., number of categories. Missing values are not allowed.
<code>id</code>	group or cluster id, should be a vector of positive integers. If AR(1) correlation, records are assumed to be ordered within each cluster id. If unstructured correlation, the cluster size must be constant, and records are assumed to be ordered the same way with each cluster id (i.e., <i>j</i> th record within each cluster refers to a common time/condition for the repeated measurement). For AR(1) and exchangeable correlation, cluster size can vary. For the formula version, include the data structure.
<code>formula</code>	For the formula version of <code>ordprobit</code> , a formula expression as for other regression models, of the form "response ~ predictors".
<code>data</code>	For the formula version of <code>ordprobit</code> , the data frame which contains the variables in the formula, and the cluster id variable.
<code>corstr</code>	For <code>ordprobit</code> as a front end to the other three functions, <code>corstr="exch"</code> means exchangeable correlation (the default); <code>corstr="ar"</code> means AR(1) correlation; <code>corstr="unstr"</code> means unstructured correlation (in which case cluster size must be a constant).
<code>iprint</code>	logical indicator, default is FALSE, for whether the iterations for numerical maximum likelihood should be printed.
<code>startpar</code>	initial parameter vector in the order: regression coefficients, correlation parameter(s). If not supplied, default = 0, <code>startpar</code> will be generated by the automatically..

Details

To get an initial version working, there are constraints: (a) For AR(1) and unstructured correlation, the maximum cluster size is 19 (although the joint probabilities get to be too small well before this limit is reached); (b) The maximum total number of parameters (regression and latent correlation parameters combined is 23). So this means a smaller upper bound on the number of predictors for the unstructured correlation.

Also the performance of the quasi-Newton algorithm gets worse as the number of parameters increase, particular with the sample size (number of clusters) is too small.

The default starting point used by the code should usually be OK. If the returned `cov=estimated covariance matrix` is the identity matrix, then the quasi-Newton iterations did not finish cleanly with a gradient vector that is near zero. If this case, it would be useful to use `iprint=1` to print the iterations, and try different starting points in `startpar` to check on the sensitivity to the starting point. The SE estimates are better if the quasi-Newton iterations finish with gradient vector closer to zero.

Value

list of MLE of parameters and their associated standard errors, in the order cutpt1,...,cutpt(number of categ-1),b1,...,b(number of covariates), rho(s); order of rhos is r12,r13,...,r23,...,r(d-1,d) for unstructured

negloglik	value of negative log-likelihood, evaluated at MLE
cutpts	MLE of ordered cutpoint parameters
beta	MLE of regression parameters
rho	MLE of latent correlation parameter for AR(1) and exchangeable correlation
rhomat	MLE of matrix of latent correlation parameter for unstructured correlation
mle	MLE of all parameters for unstructured correlation
cov	estimated covariance matrix of the parameters

Author(s)

H. Joe, Statistics Department, UBC, with assistance of Laing Wei Chou

References

Anderson, J.A. and Pemberton, J.D. (1985). The grouped continuous model for multivariate ordered categorical variables and covariate adjustment. *Biometrics*, 41, 875-885.

Joe, H (1995). Approximations to multivariate normal rectangle probabilities based on conditional expectations, *J Amer Stat Assoc*, 90, 957-964.

Examples

```
data(ordinalex)
x=ordinalex$x
y=ordinalex$y
id=ordinalex$id

# various ways of using ordprobit are shown here
# exchangeable dependence
ord.exch = ordprobit.exch(x,y,id)
print(ord.exch)

# AR(1) dependence
ord.ar = ordprobit(x,y,id,corstr="ar")
print(ord.ar)

# unstructured correlation matrix
ord.unstr = ordprobit.formula(y~x,ordinalex$id,data=ordinalex,corstr="unstr")
print(ord.unstr)
```

ordprobit.univar *Maximum Likelihood for Ordinal Probit Model*

Description

Maximum Likelihood for Ordinal Probit: Newton-Raphson minimization of negative log-likelihood.

Usage

```
ordprobit.univar(x, y, iprint=0, maxiter=20, toler=1.e-6)
```

Arguments

x	vector or matrix of explanatory variables. Each row corresponds to an observation and each column to a variable. The number of rows of x should equal the number of data values in y, and there should be fewer columns than rows. Missing values are not allowed.
y	numeric vector containing the ordinal response. The values must be in the range 1,2,..., number of categories. Missing values are not allowed.
iprint	logical indicator, default is FALSE, for whether the iterations for numerical maximum likelihood should be printed.
maxiter	maximum number of Newton-Raphson iterations, default = 20.
toler	tolerance for convergence in Newton-Raphson iterations, default = 1.e-6.

Details

If ordprobit for repeated measures ordinal probit fails to converge from the simple starting point in that function, this function ordprobit.univar should provide a better starting point. It is also equivalent to ordprobit with an identity latent correlation matrix.

The ordinal probit model is similar to the ordinal logit model (proportion odds logistic regression : polr in library MASS), The parameter estimate of ordinal logit are roughly 1.8 to 2 times those of ordinal probit (the signs of the parameters in polr may be different, as this function may be using a different orientation for the latent variable).

Value

list of MLE of parameters and their associated standard errors, in the order cutpt1,...,cutpt(number of categ-1),b1,...,b(number of covariates).

negloglik	value of negative log-likelihood, evaluated at MLE
cutpts	MLE of ordered cutpoint parameters
beta	MLE of regression parameters
cov	estimated covariance matrix of the parameters

References

Anderson, J.A. and Pemberton, J.D. (1985). The grouped continuous model for multivariate ordered categorical variables and covariate adjustment. *Biometrics*, 41, 875-885.

Examples

```
data(ordinalex)
x=as.vector(ordinalex$x)
y=ordinalex$y
ord.univar = ordprobit.univar(x,y)
print(ord.univar)
startp=c(ord.univar$cutpts,ord.univar$beta,0.5)
ord.exch <- ordprobit.exch(x,y,ordinalex$id,iprint=0,startpar=startp)
print(ord.exch)
```

pnmnorm

MVN Rectangle Probabilities

Description

Multivariate normal rectangle probabilities using Schervish's method

Usage

```
pnmnorm(lb, ub, mu, sigma, eps = 1.e-05)
```

Arguments

lb	vector of lower limits of integral/probability
ub	vector of upper limits of integral/probability
mu	mean vector of the multivariate normal density
sigma	covariance matrix, it is assumed to be positive-definite
eps	tolerance for integration

Value

out	probability of the multivariate normal rectangle region
perr	estimated accuracy
ifault	return codes from the referenced paper = 0 if no problems = 1 or 2 if eps too small = 3 if dimension is not between 1 and 6 inclusive = 4 if covariance matrix is not positive-definite

Author(s)

H. Joe, Statistics Department, UBC

References

Schervish, M.J. (1984). Multivariate normal probabilities with error bound. Appl. Statist., 33, 81-94.

See Also

[mvnapp](#).

Examples

```
rh<-0.3
m<-2
a<-c(-1,-1)
b<-c(1,1)
mu<-rep(0,m)
s<-matrix(c(1,rh,rh,1),2,2)
print(pmnorm(a,b,mu,s))
```

```
m<-3
a<-c(-1,-1,-2)
b<-c(1,1,.5)
mu<-rep(0,m)
s<-matrix(c(1,rh,rh,rh,1,rh,rh,rh,1),3,3)
print(pmnorm(a,b,mu,s))
```

```
m<-4
a<-c(-1,-2.5,-2,-1.5)
b<-c(1.68,1.11,.5,.25)
mu<-rep(0,m)
s<-matrix(c(1,rh,rh,rh,rh,1,rh,rh,rh,rh,1,rh,rh,rh,rh,1),4,4)
print(pmnorm(a,b,mu,s))
```

Index

*Topic **datasets**

binaryar, 1
binaryex, 2
ordinalex, 12

*Topic **models**

exchmvn, 2
mprobit, 4
mvn.deriv, 7
mvnapp, 10
ordprobit, 13
ordprobit.univar, 15
pmnorm, 16

binaryar, 1
binaryex, 2

exchmvn, 2
exchmvn.deriv.margin (*exchmvn*), 2
exchmvn.deriv.rho (*exchmvn*), 2

mprobit, 4
mvn.deriv, 7
mvnapp, 8, 10, 17

ordinalex, 12
ordprobit, 13
ordprobit.univar, 15

pmnorm, 11, 16