Package ‘ChoiceModelR’

January 3, 2012

Type Package

Title Choice Modeling in R

Version 1.0

Date 2011-11-08

Author Ryan Sermas `<DecisionAnalystR@decisionanalyst.com>`, assisted by
John V. Colias `<DecisionAnalystR@decisionanalyst.com>`

Author@R c(person(``Ryan", ``Sermas", role = ``aut", email =
``DecisionAnalystR@decisionanalyst.com), person("John``,"Colias``, "V``, role = "ctb``, email =
"DecisionAnalystR@decisionanlayst.com``), person("Decision Analyst, Inc.``, role = "cph"))

Maintainer Ryan Sermas `<DecisionAnalystR@decisionanalyst.com>`

Suggests bayesm, MASS, lattice, Matrix

Description Implements an MCMC algorithm to estimate a hierarchical
multinomial logit model with a normal heterogeneity
distribution. The algorithm uses a hybrid Gibbs Sampler with a
random walk metropolis step for the MNL coefficients for each
unit. Dependent variable may be discrete or continuous.
Independent variables may be discrete or continuous with
optional order constraints. Means of the distribution of
heterogeneity can optionally be modeled as a linear function of unit characteristics variables.

URL `http://www.decisionanalyst.com`

LazyLoad yes

R topics documented:

ChoiceModelR-package 2
choicemodelr 3
datar 9
sharedatar 12
truebetas 15

Index **16**

ChoiceModelR-package *Choice Modeling in R*

Description

Estimates coefficients of a Hierarchical Bayes Multinomial Logit Model

Details

Package:	ChoiceModelR
Type:	Package
Version:	1.0
Date:	2011-12-01
License:	GPL (>=3)
LazyLoad:	yes

The ChoiceModelR package includes the function choicemodelr that implements an MCMC algorithm to estimate a hierarchical multinomial logit model with a normal heterogeneity distribution. The algorithm uses a hybrid Gibbs Sampler with a random walk metropolis step for the MNL coefficients for each unit. Means of the distribution of heterogeneity can optionally be modeled as a linear function of unit descriptor variables.

The dependent variable can be either discrete or a share. If the dependent variable y_i is a share (0 to 1 inclusive), instead of discrete (1 ,..., nalt; where nalt is the number of alternatives in choice set), then each choice observation is replicated wgt times with alternative i chosen in wgt*y_i observations. Independent variables can be continuous or discrete, with order constraints imposed on estimated coefficients.

The basic structure of the code for this algorithm was derived from the rhierMnlRwMixture program of the bayesm package available at cran.r-project.org. Significant modifications were made to greatly reduce the run time, to allow constraints on estimated parameters, handle varying number of choice observations, handle varying number of choice alternatives within each choice scenario, and to optionally allow the dependent variable to be a share (between 0 and 1) instead of discrete (1 ,..., nalt; where nalt is the number of alternatives in choice set).

Author(s)

Ryan Sermas <DecisionAnalystR@decisionanalyst.com>, assisted by John V. Colias <DecisionAnalystR@decisionanalyst.com>

Maintainer: Ryan Sermas <DecisionAnalystR@decisionanalyst.com>

References

Rossi, Peter; Allenby, Greg M.; and McCulloch, Robert (2005), Bayesian Statistics and Marketing, John Wiley and Sons.

`choicemodelr` *Choice Modeling in R*

Description

Estimates coefficients of a Hierarchical Bayes Multinomial Logit Model

Usage

```
choicemodelr(data, xcoding, demos, prior, mcmc, constraints, options)
```

Arguments

`data`	Required. A data frame. The column variables of the data frame are as follows, where natts is the number of attributes; i.e., independent variables: UnitID Set Alt X_1 ... X_natts y The first column contains the ID of the unit (e.g. customer or survey respondent). The second column contains the choice set number for the unit, where each choice set is an observation for the unit. The third column contains the alternative number within the choice set. The last column contains the dependent variable. If the dependent variable y is discrete, then the dependent variable takes a non-zero value only in the first row of the choice set data, and takes a value from 1 to the number of alternatives in the choice set. For example, the following 2 rows of the data frame data shows 2 choice sets for unitID=103322 , 3 alternatives per choice set, 3 independent variables X1 to X3, and a dependent variable y indicating choice of alternative 2 in the first choice set and alternative 3 in the second choice set. 103322 1 1 4 6 1 2 103322 1 2 1 1 1 0 103322 2 1 3 6 1 3 103322 2 2 4 8 1 0 The next example is identical to the first example, except that the dependent variable is a share, indicating 30 percent and 40 percent for alternatives 1 and 2 of choice set 1. 103322 1 1 4 6 1 0.3 103322 1 2 1 1 1 0.4 103322 2 1 3 6 1 0.5 103322 2 2 4 8 1 0.5
`xcoding`	Required. A vector that specifies the way in which each attribute will be coded: 0 = categorical (effects coded)

	1 = continuous (the program mean centers the variable across the levels appearing in the data) The order of attributes in xcoding must match the order of the attributes appearing in the data file.
`demos`	An “ni by nz” matrix of demographic variables or unit characteristics, where “ni” is the number of units and “nz” is the number of unit-level demographic or descriptor variables.
`prior`	list(mubar, Amu, df, v, deltabar, Ad) mubar = prior mean of the distribution of mu; must be a vector of length equal to the number of attributes (default is a vector of zeros) Amu = precision parameter (default is 0.01) df = prior degrees of freedom (default is 5, must be ≥ 2) v = prior variance (default is 2, must be ≥ 0) deltabar = prior mean of the distribution of delta; must be a vector of length equal to the number (nz) of unit descriptor variables in the upper level model (default is a vector of zeros with length nz) Ad = precision parameter; must be a vector of length equal to natts * nz (default is 0)
`mcmc`	Required. A list with 3 arguments: list(R, use, s). R = total number of iterations of the Markov chain Monte Carlo (MCMC chain) to be performed (R is required). use = the number of iterations to be used in parameter estimation (use is required). s = a scaling parameter that is used to adjust the standard deviation of random draws of unit-level parameters during the random walk metropolis step of the MCMC chain. Only specify s if you wish to keep a constant scaling parameter. (By default, s = 0.1 and is adjusted at each iteration to keep acceptance of random draws of unit parameters at approximately 30 percent.)
`constraints`	A list of matrices containing the values 0, 1, and -1. If specifying constraints, a constraints matrix must be specified for EVERY attribute. Simply declare a matrix of 0s for an unconstrained attribute.

Each matrix must be square with dimensions equal to the number of levels of the attribute it represents. For a continuous attribute declare a 1 x 1 matrix containing the appropriate value. The matrices are interpreted as follows:

- c1[i, j] = 1, beta_i > beta_j
- c1[i, j] = -1, beta_i < beta_j
- c1[i, j] = 0, no constraint

The lower-triangular and diagonal portions of the matrix have no meaning and values in these positions are ignored.

For example, for a model with 3 attributes, set constraints = list(c1, c2, c3).

c1 = matrix(c(0,-1,-1,-1,
0,0,-1,-1,
0,0,0,-1,
0,0,0,0), ncol = 4, byrow = TRUE)

c2 = matrix(c(0,1,1,1,1,1,1,1,1,

```
0,0,1,1,1,1,1,1,1,
0,0,0,1,1,1,1,1,1,
0,0,0,0,1,1,1,1,1,
0,0,0,0,0,1,1,1,1,
0,0,0,0,0,0,1,1,1,
0,0,0,0,0,0,0,1,1,
0,0,0,0,0,0,0,0,1,
0,0,0,0,0,0,0,0,0), ncol = 9, byrow = TRUE)


c3 = matrix(c(0,1,1,1,
0,0,1,1,
0,0,0,1,
0,0,0,0), ncol = 4, byrow = TRUE)
```

`options` A list with 5 possible arguments: list(none, save, keep, wgt, restart).

none: set to TRUE to estimate a none parameter, and the data does not include a row for nono (i.e., no choice) (default is FALSE).

save: set to TRUE to save draws of betas, deltas, mu, rooti, and the log likelihood (default is FALSE).

keep = the thinning parameter defining the number of random draws to save (default is 10).

wgt = the choice-set weight parameter; possible values are 1 to 10. This parameter only needs to be specified if estimating a model using a share dependent variable (default is 5).

restart: Set to TRUE if restarting from a previous model estimation. To use this option, a model estimation must have been completed prior to the current run, and the restart.txt file must be in the working directory. All iterations from the previous run are treated as burn-in. When restarting, keep all arguments (except for R and use) identical to those of the previous run to avoid errors.

Details

Model:

Y_ij	~ MNL(beta_i*X_ij) for all i units and choice sets j
	(X_ij is nvar by 1, where nvar is the number of independent variables)
beta_i	= Z_i'delta + u_i
	(beta_i is 1 by nvar)
Z_i	= a column vector (nz by 1) of unit characteristics variables
delta	= a matrix (nz by nvar) of parameters where each column corresponds to a column of beta_i
u_i	~ N(mu,Sigma), a multivariate normal distribution
mu	= a vector of means of the distribution of heterogeneity of length nvar
Sigma	= Covariance matrix of the distribution of heterogeneity

Priors:

delta	~ N(deltabar, inverse(A_d))
mu	~ N(mubar, inverse(SigmaAmu)
Sigma_j	~ IW(nu,V)

deltabar	= nz by nvar vector of prior means = 0
Ad	= prior precision matrix for deltabar = .01I
mubar	= nvar by 1 prior mean vector for mu = vector of zeros
nu	= nuI is the degrees of freedom parameter for IW prior for Sigma
V	= pds location parameter for IW prior for Sigma
Amu	= prior precision for normal mean = .01

Value

`betadraw`	An ni by natt by floor(use/keep) array of MCMC random draws of unit-level multinomial logit model parameter estimates.
`betadraw.c`	An ni by natt by floor(use/keep) array of constrained MCMC random draws of unit-level multinomial logit model parameter estimates.
`deltadraw`	A floor(use/keep) by nz*natt array of MCMC random draws of parameter estimates on covariates to the distribution of heterogeneity.
`compdraw`	A list of floor(use/keep) MCMC random draws of estimates of means and roots for the multivariate normal distribution of heterogeneity.
`loglike`	A floor(use/keep) vector of likelihoods for the MCMC draws of multinomial logit parameters.
`Written to Console During Model Estimation`	During model estimation, the following statistics are written to the screen after each 100 iterations. The selection of these particular statistics was suggested by Sawtooth Software's technical paper, "The CBC/HB System for Hierarchical Bayes Estimation," Version 5.0 Technical Paper (2009). Following Sawtooth Software's approach for certain statistics, we use a weighted average with a weight of 0.01 for the last 100 iterations and 0.99 for previous iterations.
`Acceptance`	Percent of MCMC draws accepted in the Metropolis Hastings step.
`RLH`	nth root of the likelihood, where n is the average number of choice tasks (weighted average).
`Percent Certainty`	Percent difference between log likelihood and log likelihood of a chance model (weighted average).
`Average Variance`	Average variance of latest estimates of model coefficients across all units (weighted average).
`RMS`	Root mean squared of latest estimates of model coefficients across all units (weighted average).
`Graphic Output`	During model estimation, estimates of mu (mean of model coefficients from the distribution of heterogeneity) are plotted in the graphics window.
`Written to Disk`	At the end of model estimation, average of MCMC draws of unit-level model coefficients are written to Xbetas.csv. A log file, documenting run-time output is written to Rlog.txt. Latest MCMC draws are written to restart.txt.

Note

For further explanation of model and priors, see rhierMnlRwMixture of the bayesm package, authored by Peter Rossi, Ph.D., Anderson School, UCLA. For further discussion, see Rossi, Allenby and McCulloch (2005). The model specification is identical to that in bayesm, except that (a) the step length of the random walk metropolis algorithm was simplified to use increments of covariance (s**2)(Sigma), where “s” is a scaling parameter mentioned above and “Sigma” is the current draw of the covariance matrix of the distribution of heterogeneity and (b) the distribution of heterogeneity was simplified to a normal vs. a mixture of normals.

Author(s)

Ryan Sermas, assisted by John V. Colias Ph.D., at Decision Analyst, Inc. < `DecisionAnalystR@decisionanaly`

References

Rossi, Peter; Allenby, Greg M.; and McCulloch, Robert (2005), *Bayesian Statistics and Marketing*, John Wiley and Sons.

Sawtooth Software (2009), “The CBC/HB System for Hierarchical Bayes Estimation”, Version 5.0 Technical Paper, www.sawtoothsoftware.com.

Examples

```
# EXAMPLE 1: MULTINOMIAL LOGIT

# LOAD ARTIFICIAL (SIMULATED) DATA THAT WAS CREATED
# BY R CODE FOUND IN datar SECTION OF THE HELP FILES.

data(datar)
data(truebetas)

# USE choicemodelr TO ESTIMATE THE PARAMETERS OF THE CHOICE MODEL.
# FOR CONVERGENCE OF MCMC CHAIN, SET R = 4000 AND use = 2000.

xcoding = c(0, 0)
mcmc = list(R = 10, use = 10)

options = list(none=FALSE, save=TRUE, keep=1)

attlevels = c(5, 3)
constype =  c(0, 1)
constraints = vector("list", 2)

for (i in 1:length(attlevels)) {
constraints[[i]] = diag(0, attlevels[i])
if (constype[i] == 1) {
constraints[[i]][upper.tri(constraints[[i]])] = -1
}
else if (constype[i] == 2) {
constraints[[i]][upper.tri(constraints[[i]])] = 1
}
}

out = choicemodelr(datar, xcoding, mcmc = mcmc, options = options, constraints = constrai
# CALCULATE MEAN ABSOLUTE ERROR BETWEEN ESTIMATED AND TRUE BETAS.
estbetas = apply(out$betadraw.c,c(1,2),mean)
estbetas = cbind(estbetas[,1:4],0-apply(estbetas[,1:4],1,sum),estbetas[,5:6],0-apply(estb
colnames(estbetas) = c("A1B1", "A1B2", "A1B3", "A1B4", "A1B5", "A2B1", "A2B2", "A2B3")

MAE = mean(abs(estbetas - truebetas))
print(MAE)

# CALCULATE MEAN ABSOLUTE ERROR BETWEEN PROBABILITY
# DIFFERENCES USING ESTIMATED AND TRUE BETAS.

TrueProb = cbind(exp(truebetas[,1:5]) / apply(exp(truebetas[,1:5]),1,sum),
                 exp(truebetas[,6:8]) / apply(exp(truebetas[,6:8]),1,sum))
EstProb = cbind(exp(estbetas[,1:5]) / apply(exp(estbetas[,1:5]),1,sum),
                exp(estbetas[,6:8]) / apply(exp(estbetas[,6:8]),1,sum))
MAEProb = mean(abs(TrueProb - EstProb))

print(MAEProb)


# EXAMPLE 2: FRACTIONAL MULTINOMIAL LOGIT

# LOAD ARTIFICIAL (SIMULATED) FRACTIONAL MULTINOMIAL LOGIT DATA CREATED
# BY R CODE FOUND IN sharedatar SECTION OF THE HELP FILES.

data(sharedatar)
data(truebetas)

# USE choicemodelr TO ESTIMATE THE PARAMETERS OF THE CHOICE MODEL.
# FOR CONVERGENCE OF MCMC CHAIN, SET R = 2000 AND use = 1000.

xcoding = c(0, 0)
mcmc = list(R = 10, use = 10)

options = list(none=FALSE, save=TRUE, keep=1)

attlevels = c(5, 3)
constype =  c(0, 1)
constraints = vector("list", 2)

for (i in 1:length(attlevels)) {
constraints[[i]] = diag(0, attlevels[i])
if (constype[i] == 1) {
constraints[[i]][upper.tri(constraints[[i]])] = -1
}
else if (constype[i] == 2) {
constraints[[i]][upper.tri(constraints[[i]])] = 1
}
}

out = choicemodelr(sharedatar, xcoding, mcmc = mcmc, options = options, constraints = con

# CALCULATE MEAN ABSOLUTE ERROR BETWEEN ESTIMATED AND TRUE BETAS.
estbetas = apply(out$betadraw.c,c(1,2),mean)
estbetas = cbind(estbetas[,1:4],0-apply(estbetas[,1:4],1,sum),estbetas[,5:6],0-apply(estb
colnames(estbetas) = c("A1B1", "A1B2", "A1B3", "A1B4", "A1B5", "A2B1", "A2B2", "A2B3")
```

```
MAE = mean(abs(estbetas - truebetas))
print(MAE)

# CALCULATE MEAN ABSOLUTE ERROR BETWEEN PROBABILITY
# DIFFERENCES USING ESTIMATED AND TRUE BETAS.

TrueProb = cbind(exp(truebetas[,1:5]) / apply(exp(truebetas[,1:5]),1,sum),
                 exp(truebetas[,6:8]) / apply(exp(truebetas[,6:8]),1,sum))
EstProb = cbind(exp(estbetas[,1:5]) / apply(exp(estbetas[,1:5]),1,sum),
                exp(estbetas[,6:8]) / apply(exp(estbetas[,6:8]),1,sum))
MAEProb = mean(abs(TrueProb - EstProb))

print(MAEProb)
```

`datar` *Arificial (Simulated) Choice Data for choicemodelr*

Description

Artificial (simulated) choice data for 300 units with a discrete dependent variable. The choice data has a maximum of 50 choice sets per unit (varies from unit to unit). The choice sets have a maximum of 5 alternatives per choice set (varies from choice set to choice set).

Usage

```
data(datar)
```

Format

The format is: num [1:61342, 1:6] 1 1 1 1 1 1 1 1 1 1 ... - attr(*, "dimnames")=List of 2 ..$: NULL ..$: chr [1:6] "" "" "" "" ...

Source

Choice data was simulated using the code in the example.

Examples

```
data(datar)
head(datar)

# datar DATA SET WAS CREATED USING THE FOLLOWING CODE.

if (0) {

# LOAD LIBRARIES REQUIRED TO CREATE THE SIMULATED DATA.  YOU MAY NEED TO INSTALL THESE PA
library(MASS)
library(lattice)
library(Matrix)
library(bayesm)

set.seed(88)
```

```
# CREATE FUNCTION TO SIMULATE ARTIFICIAL MULTINOMIAL CHOICE DATA BASED SIMULATED TRUE BET

simmnlv2 = function(p,n,beta)
{
#
#   p. rossi 2004
#   Modified by John Colias 2011
#
# Purpose: simulate from MNL (including X values)
#
# Arguments:
#   p is number of alternatives
#   n is number of obs
#   beta is true parm value
#
# Output:
#   list of X  (note: we include full set of intercepts and 2 unif(-1,1) X vars)
#   y  (indicator of choice-- 1, ...,p
#   prob is a n x p matrix of choice probs
#
#   note: first choice alternative has intercept set to zero
#
k=length(beta)
x1=runif(n*p,min=-1,max=1)
x2=runif(n*p,min=-1,max=1)
x3=runif(n*p,min=-1,max=1)
I2=diag(rep(1,p-1))
zero=rep(0,p-1)
xadd=rbind(zero,I2)
for(i in 2:n) {
        xadd=rbind(xadd,zero,I2)
}

xlast3 = cbind(x1,x2,x3)
xmax = apply(xlast3,1,max)
xcat = (xlast3 == xmax)*1
X=cbind(xadd,xcat)

# now construct probabilities
Xbeta=X
p=nrow(Xbeta)/n
Xbeta=matrix(Xbeta,byrow=TRUE,ncol=p)
Prob=exp(Xbeta)
iota=c(rep(1,p))
denom=Prob
Prob=Prob/as.vector(denom)
# draw y
y=vector("double",n)
ind=1:p
for (i in 1:n)
        {
        yvec=rmultinom(1,1,Prob[i,])
        y[i]=ind
        }

return(list(y=y,X=X,beta=beta,prob=Prob))
}
```

```
# DEFINE DIMENSIONS OF ARTIFICIAL DATA.

nunits = 300     # number of units
cmax = 50        # maximum number of cards per unit
amax = 5         # maximum number of alternatives per card

# CREATE SIGMA FOR MULTIVARIATE NORMAL DISTRIBUTION OF HETEROGENEITY.
sigma = 0.2*matrix(runif(49),7,7)
tsigma = t(sigma)
sigma[lower.tri(sigma)] = tsigma[lower.tri(tsigma)]
sigma = nearPD(sigma)$mat

# DEFINE MEANS FOR MULTIVARIATE NORMAL DISTRIBUTION OF HETEROGENEITY.
avgbeta = c(.5,-1.5,.9,1.0,-1, -0.5, 1.5)

# DRAW BETAS FOR EACH UNIT.
# LAST THREE BETAS ARE 3 LEVELS OF ONE ATTRIBUTE
# THAT IS NON-DECREASING IN VALUE.

betatemp = mvrnorm(n=nunits, avgbeta, sigma)
beta = betatemp[,1:5]
beta = cbind(beta,beta[,5]+exp(betatemp[,6]))
beta = cbind(beta,beta[,6]+exp(betatemp[,7]))
tbeta = cbind(beta[,1:4],0) - apply(cbind(beta[,1:4],0),1,mean)
beta[,1:4] = tbeta[,1:4]
tbeta = beta[,5:7] - apply(beta[,5:7],1,mean)
beta[,5:7] = tbeta

# CREATE MULTINOMIAL LOGIT y AND X FOR EACH UNIT ASSUMING beta IS "TRUE".
datah=NULL
for (i in 1:nunits) {
datah[[i]] = simmnlv2(amax,cmax,beta[i,])
}

# SAMPLE cmax-2, cmax-1, or cmax CARDS
# FOR EACH UNIT TO CREATE DATA WITH VARYING
# NUMBER OF CHOICE CARDS PER UNIT.
# SAMPLE amax-2, amax-1, or amax ALTERNATIVES
# FOR EACH CHOICE CARD OF EACH UNIT
# TO CREATE DATA WITH VARYING NUMBER OF
# ALTERNATIVES PER CHOICE CARD.
ny = NULL
datar = NULL
for (i in 1:nunits) {
      if (i == 1) {
         cat("Please wait ... this may take a few minutes.", fill = TRUE)
         cat("", fill = TRUE) }

# SAMPLE CHOICE CARDS.
cards = sample(c(1:cmax),sample(c(cmax-2,cmax-1,cmax),1))
cnum = 0
for (c in cards) {
    cnum = cnum + 1
    cond = 0
# KEEP SAMPLING ALTERNATIVES UNTIL THE CHOSEN ALTERNATIVE IS WITHIN THE SAMPLED ALTERNATI
    while (cond==0) {
        alts = sample(c(1:amax),sample(c(amax-2,amax-1,amax),1))
        depvar = datah[[i]]$y[c]
        if (is.element(depvar,alts)) {
            cond = 1
            depvar = sum((depvar==alts)*c(1:length(alts))) } }
    anum = 0
    for (a in alts) {
        anum = anum + 1
        if (anum > 1) {depvar = 0}
        xx = datah[[i]]$X[(c-1)*amax+a,]
        xa = xx[1:(length(xx)-3)]
        if (sum(xa)==0) {xa = length(xx) - 2}
        xb = which.max(xx[(length(xx)-2):length(xx)])
        datar = rbind(datar,c(i,cnum,anum,xa,xb,depvar)) } } }

truebetas = cbind(beta[,1:4],0-apply(beta[,1:4],1,sum),beta[,5:7])
colnames(truebetas) = c("A1B1", "A1B2", "A1B3", "A1B4", "A1B5", "A2B1", "A2B2", "A2B3")

# END OF CODE TO CREATE ARTIFICIAL DATA.
}
```

---

sharedatar *Arificial (Simulated) Fractional Choice Data for choicemodelr*

Description

Artificial (simulated) fractional choice data for 300 units with a share dependent variable. The choice data has 50 choice sets per unit. The choice sets have 5 alternatives per choice set.

Usage

```
data(sharedatar)
```

Format

The format is: num [1:75000, 1:6] 1 1 1 1 1 1 1 1 1 1 ... - attr(*, "dimnames")=List of 2 ..$: NULL ..$: chr [1:6] "" "" "" "" ...

Source

Fractional choice data was simulated using the code in the example.

Examples

```
data(sharedatar)
head(sharedatar)

#  sharedatar WAS CREATED USING THE FOLLOWING CODE.

if (0) {

# LOAD LIBRARIES REQUIRED TO CREATE THE SIMULATED DATA.
# YOU MAY NEED TO INSTALL THESE PACKAGES.
```

```
library(MASS)
library(lattice)
library(Matrix)
library(bayesm)

set.seed(88)

# CREATE FUNCTION TO SIMULATE ARTIFICIAL MULTINOMIAL
# FRACTIONAL CHOICE DATA BASED SIMULATED TRUE BETAS.

simmnlv3 = function(p,n,l,beta)
{
#
#   p. rossi 2004
#   Modified by John Colias 2011
#
# Purpose: simulate from Fractional MNL (including X values)
#
# Arguments:
#   p is number of alternatives
#   n is number of obs
#   l is number of draws to construct the share
#   beta is true parm value
#
# Output:
#   list of X  (note: we include full set of intercepts and 2 unif(-1,1) X vars)
#   y  (indicator of choice-- 1, ...,p
#   prob is a n x p matrix of choice probs
#
#   note: first choice alternative has intercept set to zero
#
k=length(beta)
x1=runif(n*p,min=-1,max=1)
x2=runif(n*p,min=-1,max=1)
x3=runif(n*p,min=-1,max=1)
I2=diag(rep(1,p-1))
zero=rep(0,p-1)
xadd=rbind(zero,I2)
for(i in 2:n) {
        xadd=rbind(xadd,zero,I2)
}

xlast3 = cbind(x1,x2,x3)
xmax = apply(xlast3,1,max)
xcat = (xlast3 == xmax)*1
X=cbind(xadd,xcat)

# now construct probabilities
Xbeta=X
p=nrow(Xbeta)/n
Xbeta=matrix(Xbeta,byrow=T,ncol=p)
Prob=exp(Xbeta)
iota=c(rep(1,p))
denom=Prob
Prob=Prob/as.vector(denom)
# draw y
y=array(double(1),dim=c(n,p,l))
```

```
for (i in 1:n)
        {
        for (l in 1:l) {
        yvec=rmultinom(1,1,Prob[i,])
        y[i,,l] = yvec
        }
        }
return(list(y=apply(y,c(1,2),mean),X=X,beta=beta,prob=Prob))
}

# DEFINE DIMENSIONS OF ARTIFICIAL DATA.

nunits = 300     # number of units
cnum = 50        # number of cards per unit
anum = 5         # number of alternatives per card
lnum = 50        # number of draws to construct the shares for each card

# CREATE SIGMA FOR MULTIVARIATE NORMAL DISTRIBUTION OF HETEROGENEITY.
sigma = 0.2*matrix(runif(49),7,7)
tsigma = t(sigma)
sigma[lower.tri(sigma)] = tsigma[lower.tri(tsigma)]
sigma = nearPD(sigma)$mat

# DEFINE MEANS FOR MULTIVARIATE NORMAL DISTRIBUTION OF HETEROGENEITY.
avgbeta = c(.5,-1.5,.9,1.0,-1, -0.5, 1.5)

# DRAW BETAS FOR EACH UNIT.
# LAST THREE BETAS ARE 3 LEVELS OF ONE ATTRIBUTE
# THAT IS NON-DECREASING IN VALUE.

betatemp = mvrnorm(n=nunits, avgbeta, sigma)
beta = betatemp[,1:5]
beta = cbind(beta,beta[,5]+exp(betatemp[,6]))
beta = cbind(beta,beta[,6]+exp(betatemp[,7]))
tbeta = cbind(beta[,1:4],0) - apply(cbind(beta[,1:4],0),1,mean)
beta[,1:4] = tbeta[,1:4]
tbeta = beta[,5:7] - apply(beta[,5:7],1,mean)
beta[,5:7] = tbeta

# CREATE MULTINOMIAL LOGIT y AND X FOR EACH UNIT ASSUMING beta IS "TRUE".
datah=NULL
for (i in 1:nunits) {
datah[[i]] = simmnlv3(anum,cnum,lnum,beta[i,])
}

sharedatar = NULL
for (i in 1:nunits) {
    if (i == 1) {
        cat("Please wait ... this may take a few minutes.", fill = TRUE)
        cat("", fill = TRUE) }
    for (c in 1:cnum) {
        depvar = datah[[i]]$y[c,]
            for (a in 1:anum) {
            xx = datah[[i]]$X[(c-1)*anum+a,]
            xa = xx[1:(length(xx)-3)]
            if (sum(xa)==0) {xa = length(xx) - 2}
            xb = which.max(xx[(length(xx)-2):length(xx)])
            sharedatar = rbind(sharedatar,c(i,c,a,xa,xb,depvar[a])) } } }

# END OF CODE TO CREATE ARTIFICIAL DATA.
}
```

`truebetas` *True betas used to simulate data in the choice data set named datar, which is used in the example.*

Description

True betas are effects-coded betas for two variables for 300 units. The first variable is a four-level categorical and the second variable is a three-level categorical variable. The latter is constrained to be non-decreasing. These betas were used to simulate the choice data in the example data set named datar.

Usage

```
data(truebetas)
```

Format

The format is: num [1:300, 1:8] 0.7314 0.0484 0.1874 0.3961 0.5678 ... - attr(*, "dimnames")=List of 2 ..$: NULL ..$: chr [1:8] "A1B1" "A1B2" "A1B3" "A1B4" ...

Source

The true betas were created using the code in the example.

Examples

```
data(truebetas)
head(truebetas)
```

Index

∗Topic **datasets**
    datar, 9
    sharedatar, 12
    truebetas, 15
∗Topic **models**
    choicemodelr, 3
    ChoiceModelR-package, 2
∗Topic **multivariate**
    choicemodelr, 3
    ChoiceModelR-package, 2
∗Topic **package**
    choicemodelr, 3
    ChoiceModelR-package, 2

choicemodelr, 3
ChoiceModelR-package, 2

datar, 9

sharedatar, 12

truebetas, 15