bank *Bank Card Conjoint Data of Allenby and Ginter (1995)*

Description

Data from a conjoint experiment in which two partial profiles of credit cards were presented to 946 respondents. The variable bank$choiceAtt$choice indicates which profile was chosen. The profiles are coded as the difference in attribute levels. Thus, a "-1" means the profile coded as a choice of "0" has the attribute. A value of 0 means that the attribute was not present in the comparison.

data on age,income and gender (female=1) are also recorded in bank$demo

Usage

```
data(bank)
```

Format

This R object is a list of two data frames, list(choiceAtt,demo).

List of 2

$ choiceAtt:'data.frame': 14799 obs. of 16 variables:
...$ id : int [1:14799] 1 1 1 1 1 1 1 1 1 1
...$ choice : int [1:14799] 1 1 1 1 1 1 1 1 0 1
...$ Med_FInt : int [1:14799] 1 1 1 0 0 0 0 0 0 0
...$ Low_FInt : int [1:14799] 0 0 0 0 0 0 0 0 0 0
...$ Med_VInt : int [1:14799] 0 0 0 0 0 0 0 0 0 0
...$ Rewrd_2 : int [1:14799] -1 1 0 0 0 0 0 1 -1 0
...$ Rewrd_3 : int [1:14799] 0 -1 1 0 0 0 0 0 1 -1
...$ Rewrd_4 : int [1:14799] 0 0 -1 0 0 0 0 0 0 1
...$ Med_Fee : int [1:14799] 0 0 0 1 1 -1 -1 0 0 0
...$ Low_Fee : int [1:14799] 0 0 0 0 0 1 1 0 0 0
...$ Bank_B : int [1:14799] 0 0 0 -1 1 -1 1 0 0 0
...$ Out_State : int [1:14799] 0 0 0 0 -1 0 -1 0 0 0
...$ Med_Rebate : int [1:14799] 0 0 0 0 0 0 0 0 0 0
...$ High_Rebate : int [1:14799] 0 0 0 0 0 0 0 0 0 0
...$ High_CredLine: int [1:14799] 0 0 0 0 0 0 0 -1 -1 -1
...$ Long_Grace : int [1:14799] 0 0 0 0 0 0 0 0 0 0

$ demo :'data.frame': 946 obs. of 4 variables:
...$ id : int [1:946] 1 2 3 4 6 7 8 9 10 11
...$ age : int [1:946] 60 40 75 40 30 30 50 50 50 40
...$ income: int [1:946] 20 40 30 40 30 60 50 100 50 40
...$ gender: int [1:946] 1 1 0 0 0 0 1 0 0 0

Details

Each respondent was presented with between 13 and 17 paired comparisons. Thus, this dataset has a panel structure.

Source

Allenby and Ginter (1995), "Using Extremes to Design Products and Segment Markets," *JMR*, 392-403.

References

Appendix A, *Bayesian Statistics and Marketing* by Rossi,Allenby and McCulloch.
http://faculty.chicagogsb.edu/peter.rossi/research/bsm.html

Examples

```
data(bank)
cat(" table of Binary Dep Var", fill=TRUE)
print(table(bank$choiceAtt[,2]))
cat(" table of Attribute Variables",fill=TRUE)
mat=apply(as.matrix(bank$choiceAtt[,3:16]),2,table)
print(mat)
cat(" means of Demographic Variables",fill=TRUE)
mat=apply(as.matrix(bank$demo[,2:3]),2,mean)
print(mat)

## example of processing for use with rhierBinLogit
##
if(0)
{
choiceAtt=bank$choiceAtt
Z=bank$demo

## center demo data so that mean of random-effects
## distribution can be interpreted as the average respondent

Z[,1]=rep(1,nrow(Z))
Z[,2]=Z[,2]-mean(Z[,2])
Z[,3]=Z[,3]-mean(Z[,3])
Z[,4]=Z[,4]-mean(Z[,4])
Z=as.matrix(Z)

hh=levels(factor(choiceAtt$id))
nhh=length(hh)
lgtdata=NULL
for (i in 1:nhh) {
        y=choiceAtt[choiceAtt[,1]==hh[i],2]
        nobs=length(y)
        X=as.matrix(choiceAtt[choiceAtt[,1]==hh[i],c(3:16)])
        lgtdata[[i]]=list(y=y,X=X)
                }
```

```
cat("Finished Reading data",fill=TRUE)
fsh()

Data=list(lgtdata=lgtdata,Z=Z)
Mcmc=list(R=10000,sbeta=0.2,keep=20)
set.seed(66)
out=rhierBinLogit(Data=Data,Mcmc=Mcmc)

begin=5000/20
end=10000/20

summary(out$Deltadraw,burnin=begin)
summary(out$Vbetadraw,burnin=begin)

if(0){
## plotting examples

## plot grand means of random effects distribution (first row of Delta)
index=4*c(0:13)+1
matplot(out$Deltadraw[,index],type="l",xlab="Iterations/20",ylab="",
main="Average Respondent Part-Worths")

## plot hierarchical coefs
plot(out$betadraw)

## plot log-likelihood
plot(out$llike,type="l",xlab="Iterations/20",ylab="",main="Log Likelihood")

}
}
```

`breg` *Posterior Draws from a Univariate Regression with Unit Error Variance*

Description

`breg` makes one draw from the posterior of a univariate regression (scalar dependent variable) given the error variance = 1.0. A natural conjugate, normal prior is used.

Usage

```
breg(y, X, betabar, A)
```

Arguments

`y`	vector of values of dep variable.
`X`	n (length(y)) x k Design matrix.
`betabar`	k x 1 vector. Prior mean of regression coefficients.
`A`	Prior precision matrix.

Details

model: $y = x'\beta + e$. $e \sim N(0, 1)$.

prior: $\beta \sim N(betabar, A^{-1})$.

Value

k x 1 vector containing a draw from the posterior distribution.

Warning

This routine is a utility routine that does **not** check the input arguments for proper dimensions and type.

In particular, X must be a matrix. If you have a vector for X, coerce it into a matrix with one column

Author(s)

Peter Rossi, Graduate School of Business, University of Chicago, ⟨Peter.Rossi@ChicagoGsb.edu⟩.

References

For further discussion, see *Bayesian Statistics and Marketing* by Rossi,Allenby and McCulloch.
http://faculty.chicagogsb.edu/peter.rossi/research/bsm.html

Examples

```
##

if(nchar(Sys.getenv("LONG_TEST")) != 0) {R=1000} else {R=10}

## simulate data
set.seed(66)
n=100
X=cbind(rep(1,n),runif(n)); beta=c(1,2)
y=X%*%beta+rnorm(n)
##
## set prior
A=diag(c(.05,.05)); betabar=c(0,0)
##
## make draws from posterior
betadraw=matrix(double(R*2),ncol=2)
for (rep in 1:R) {betadraw[rep,]=breg(y,X,betabar,A)}
##
## summarize draws
mat=apply(betadraw,2,quantile,probs=c(.01,.05,.5,.95,.99))
mat=rbind(beta,mat); rownames(mat)[1]="beta"; print(mat)
```

Description

`cgetC` obtains a list of censoring points, or cut-offs, used in the ordinal multivariate probit model of Rossi et al (2001). This approach uses a quadratic parameterization of the cut-offs. The model is useful for modeling correlated ordinal data on a scale from 1, ..., k with different scale usage patterns.

Usage

```
cgetC(e, k)
```

Arguments

`e`	quadratic parameter (>0 and less than 1)
`k`	items are on a scale from 1, . . . , k

Value

A vector of k+1 cut-offs.

Warning

This is a utility function which implements **no** error-checking.

Author(s)

Rob McCulloch and Peter Rossi, Graduate School of Business, University of Chicago. ⟨Peter.Rossi@ChicagoGsb.edu⟩.

References

Rossi et al (2001), "Overcoming Scale Usage Heterogeneity," *JASA*96, 20-31.

See Also

`rscaleUsage`

Examples

```
##
cgetC(.1,10)
```

cheese	*Sliced Cheese Data*

Description

Panel data with sales volume for a package of Borden Sliced Cheese as well as a measure of display activity and price. Weekly data aggregated to the "key" account or retailer/market level.

Usage

```
data(cheese)
```

Format

A data frame with 5555 observations on the following 4 variables.

`RETAILER` a list of 88 retailers

`VOLUME` unit sales

`DISP` a measure of display activity – per cent ACV on display

`PRICE` in $

Source

Boatwright et al (1999), "Account-Level Modeling for Trade Promotion," *JASA* 94, 1063-1073.

References

Chapter 3, *Bayesian Statistics and Marketing* by Rossi et al.
http://faculty.chicagogsb.edu/peter.rossi/research/bsm.html

Examples

```
data(cheese)
cat(" Quantiles of the Variables ",fill=TRUE)
mat=apply(as.matrix(cheese[,2:4]),2,quantile)
print(mat)

##
## example of processing for use with rhierLinearModel
##
if(0)
{

retailer=levels(cheese$RETAILER)
nreg=length(retailer)
nvar=3
regdata=NULL
```

```
for (reg in 1:nreg) {
        y=log(cheese$VOLUME[cheese$RETAILER==retailer[reg]])
        iota=c(rep(1,length(y)))
        X=cbind(iota,cheese$DISP[cheese$RETAILER==retailer[reg]],
                log(cheese$PRICE[cheese$RETAILER==retailer[reg]]))
        regdata[[reg]]=list(y=y,X=X)
}
Z=matrix(c(rep(1,nreg)),ncol=1)
nz=ncol(Z)
##
## run each individual regression and store results
##
lscoef=matrix(double(nreg*nvar),ncol=nvar)
for (reg in 1:nreg) {
        coef=lsfit(regdata[[reg]]$X,regdata[[reg]]$y,intercept=FALSE)$coef
        if (var(regdata[[reg]]$X[,2])==0)  { lscoef[reg,1]=coef[1]; lscoef[reg,3]=coef[2]}
        else {lscoef[reg,]=coef }
}

R=2000
Data=list(regdata=regdata,Z=Z)
Mcmc=list(R=R,keep=1)

set.seed(66)
out=rhierLinearModel(Data=Data,Mcmc=Mcmc)

cat("Summary of Delta Draws",fill=TRUE)
summary(out$Deltadraw)
cat("Summary of Vbeta Draws",fill=TRUE)
summary(out$Vbetadraw)

if(0){
#
# plot hier coefs
plot(out$betadraw)
}

}
```

`clusterMix` *Cluster Observations Based on Indicator MCMC Draws*

Description

`clusterMix` uses MCMC draws of indicator variables from a normal component mixture model to cluster observations based on a similarity matrix.

Usage

```
clusterMix(zdraw, cutoff = 0.9, SILENT = FALSE)
```

Arguments

`zdraw`	R x nobs array of draws of indicators
`cutoff`	cutoff probability for similarity (def=.9)
`SILENT`	logical flag for silent operation (def= FALSE)

Details

define a similarity matrix, Sim, Sim[i,j]=1 if observations i and j are in same component. Compute the posterior mean of Sim over indicator draws.

clustering is achieved by two means:

Method A: Find the indicator draw whose similarity matrix minimizes, loss(E[Sim]-Sim(z)), where loss is absolute deviation.

Method B: Define a Similarity matrix by setting any element of E[Sim] = 1 if E[Sim] > cutoff. Compute the clustering scheme associated with this "windsorized" Similarity matrix.

Value

`clustera`	indicator function for clustering based on method A above
`clusterb`	indicator function for clustering based on method B above

Warning

This routine is a utility routine that does **not** check the input arguments for proper dimensions and type.

Author(s)

Peter Rossi, Graduate School of Business, University of Chicago ⟨Peter.Rossi@ChicagoGsb.edu⟩.

References

For further discussion, see *Bayesian Statistics and Marketing* by Rossi, Allenby and McCulloch Chapter 3.
http://faculty.chicagogsb.edu/peter.rossi/research/bsm.html

See Also

`rnmixGibbs`

Examples

```
##
if(nchar(Sys.getenv("LONG_TEST")) != 0)
{
## simulate data from mixture of normals
n=500
pvec=c(.5,.5)
mu1=c(2,2)
mu2=c(-2,-2)
```

```
Sigma1=matrix(c(1,.5,.5,1),ncol=2)
Sigma2=matrix(c(1,.5,.5,1),ncol=2)
comps=NULL
comps[[1]]=list(mu1,backsolve(chol(Sigma1),diag(2)))
comps[[2]]=list(mu2,backsolve(chol(Sigma2),diag(2)))
dm=rmixture(n,pvec,comps)
## run MCMC on normal mixture
R=2000
Data=list(y=dm$x)
ncomp=2
Prior=list(ncomp=ncomp,a=c(rep(100,ncomp)))
Mcmc=list(R=R,keep=1)
out=rnmixGibbs(Data=Data,Prior=Prior,Mcmc=Mcmc)
begin=500
end=R
## find clusters
outclusterMix=clusterMix(out$zdraw[begin:end,])
##
## check on clustering versus "truth"
##  note: there could be switched labels
##
table(outclusterMix$clustera,dm$z)
table(outclusterMix$clusterb,dm$z)
}
##
```

`condMom` *Computes Conditional Mean/Var of One Element of MVN given All Others*

Description

`condMom` compute moments of conditional distribution of ith element of normal given all others.

Usage

```
condMom(x, mu, sigi, i)
```

Arguments

`x`	vector of values to condition on - ith element not used
`mu`	length(x) mean vector
`sigi`	length(x)-dim covariance matrix
`i`	conditional distribution of ith element

Details

$x \sim MVN(mu, Sigma)$.

`condMom` computes moments of x_i given x_{-i}.

Value

a list containing:

`cmean`	cond mean
`cvar`	cond variance

Warning

This routine is a utility routine that does **not** check the input arguments for proper dimensions and type.

Author(s)

Peter Rossi, Graduate School of Business, University of Chicago, ⟨Peter.Rossi@ChicagoGsb.edu⟩.

References

For further discussion, see *Bayesian Statistics and Marketing* by Rossi, Allenby and McCulloch.
`http://faculty.chicagogsb.edu/peter.rossi/research/bsm.html`

Examples

```
##
sig=matrix(c(1,.5,.5,.5,1,.5,.5,.5,1),ncol=3)
sigi=chol2inv(chol(sig))
mu=c(1,2,3)
x=c(1,1,1)
condMom(x,mu,sigi,2)
```

`createX` — *Create X Matrix for Use in Multinomial Logit and Probit Routines*

Description

`createX` makes up an X matrix in the form expected by Multinomial Logit (`rmnlIndepMetrop` and `rhierMnlRwMixture`) and Probit (`rmnpGibbs` and `rmvpGibbs`) routines. Requires an array of alternative specific variables and/or an array of "demographics" or variables constant across alternatives which may vary across choice occasions.

Usage

```
createX(p, na, nd, Xa, Xd, INT = TRUE, DIFF = FALSE, base = p)
```

Arguments

`p`	integer - number of choice alternatives
`na`	integer - number of alternative-specific vars in Xa
`nd`	integer - number of non-alternative specific vars
`Xa`	n x p*na matrix of alternative-specific vars
`Xd`	n x nd matrix of non-alternative specific vars
`INT`	logical flag for inclusion of intercepts
`DIFF`	logical flag for differencing wrt to base alternative
`base`	integer - index of base choice alternative

note: na,nd,Xa,Xd can be NULL to indicate lack of Xa or Xd variables.

Value

X matrix – n*(p-DIFF) x [(INT+nd)*(p-1) + na] matrix.

Author(s)

Peter Rossi, Graduate School of Business, University of Chicago, ⟨Peter.Rossi@ChicagoGsb.edu⟩.

References

For further discussion, see *Bayesian Statistics and Marketing* by Rossi, Allenby and McCulloch.
http://faculty.chicagogsb.edu/peter.rossi/research/bsm.html

See Also

`rmnlIndepMetrop`, `rmnpGibbs`

Examples

```
na=2; nd=1; p=3
vec=c(1,1.5,.5,2,3,1,3,4.5,1.5)
Xa=matrix(vec,byrow=TRUE,ncol=3)
Xa=cbind(Xa,-Xa)
Xd=matrix(c(-1,-2,-3),ncol=1)
createX(p=p,na=na,nd=nd,Xa=Xa,Xd=Xd)
createX(p=p,na=na,nd=nd,Xa=Xa,Xd=Xd,base=1)
createX(p=p,na=na,nd=nd,Xa=Xa,Xd=Xd,DIFF=TRUE)
createX(p=p,na=na,nd=nd,Xa=Xa,Xd=Xd,DIFF=TRUE,base=2)
createX(p=p,na=na,nd=NULL,Xa=Xa,Xd=NULL)
createX(p=p,na=NULL,nd=nd,Xa=NULL,Xd=Xd)
```

`customerSat` *Customer Satisfaction Data*

Description

Responses to a satisfaction survey for a Yellow Pages advertising product. All responses are on a 10 point scale from 1 to 10 (10 is "Excellent" and 1 is "Poor")

Usage

```
data(customerSat)
```

Format

A data frame with 1811 observations on the following 10 variables.

`q1` Overall Satisfaction

`q2` Setting Competitive Prices

`q3` Holding Price Increase to a Minimum

`q4` Appropriate Pricing given Volume

`q5` Demonstrating Effectiveness of Purchase

`q6` Reach a Large # of Customers

`q7` Reach of Advertising

`q8` Long-term Exposure

`q9` Distribution

`q10` Distribution to Right Geographic Areas

Source

Rossi et al (2001), "Overcoming Scale Usage Heterogeneity," *JASA* 96, 20-31.

References

Case Study 3, *Bayesian Statistics and Marketing* by Rossi et al.
http://faculty.chicagogsb.edu/peter.rossi/research/bsm.html

Examples

```
data(customerSat)
apply(as.matrix(customerSat),2,table)
```

detailing *Physician Detailing Data from Manchanda et al (2004)*

Description

Monthly data on detailing (sales calls) on 1000 physicians. 23 mos of data for each Physician. Includes physician covariates. Dependent Variable (`scripts`) is the number of new prescriptions ordered by the physician for the drug detailed.

Usage

```
data(detailing)
```

Format

This R object is a list of two data frames, list(counts,demo).

List of 2:

$ counts:'data.frame': 23000 obs. of 4 variables:
...$ id : int [1:23000] 1 1 1 1 1 1 1 1 1 1
...$ scripts : int [1:23000] 3 12 3 6 5 2 5 1 5 3
...$ detailing : int [1:23000] 1 1 1 2 1 0 2 2 1 1
...$ lagged_scripts: int [1:23000] 4 3 12 3 6 5 2 5 1 5

$ demo :'data.frame': 1000 obs. of 4 variables:
...$ id : int [1:1000] 1 2 3 4 5 6 7 8 9 10
...$ generalphys : int [1:1000] 1 0 1 1 0 1 1 1 1 1
...$ specialist: int [1:1000] 0 1 0 0 1 0 0 0 0 0
...$ mean_samples: num [1:1000] 0.722 0.491 0.339 3.196 0.348

Details

generalphys is dummy for if doctor is a "general practitioner," specialist is dummy for if the physician is a specialist in the theraputic class for which the drug is intended, mean_samples is the mean number of free drug samples given the doctor over the sample.

Source

Manchanda, P., P. K. Chintagunta and P. E. Rossi (2004), "Response Modeling with Non-Random Marketing Mix Variables," *Journal of Marketing Research* 41, 467-478.

Examples

```
data(detailing)
cat(" table of Counts Dep Var", fill=TRUE)
print(table(detailing$counts[,2]))
cat(" means of Demographic Variables",fill=TRUE)
mat=apply(as.matrix(detailing$demo[,2:4]),2,mean)
print(mat)
```

```
##
## example of processing for use with rhierNegbinRw
##
if(0)
{
data(detailing)
counts = detailing$counts
Z = detailing$demo

# Construct the Z matrix
Z[,1] = 1
Z[,2]=Z[,2]-mean(Z[,2])
Z[,3]=Z[,3]-mean(Z[,3])
Z[,4]=Z[,4]-mean(Z[,4])
Z=as.matrix(Z)
id=levels(factor(counts$id))
nreg=length(id)
nobs = nrow(counts$id)

regdata=NULL
for (i in 1:nreg) {
    X = counts[counts[,1] == id[i],c(3:4)]
    X = cbind(rep(1,nrow(X)),X)
    y = counts[counts[,1] == id[i],2]
    X = as.matrix(X)
    regdata[[i]]=list(X=X, y=y)
}
nvar=ncol(X)            # Number of X variables
nz=ncol(Z)              # Number of Z variables
rm(detailing,counts)
cat("Finished Reading data",fill=TRUE)
fsh()

Data = list(regdata=regdata, Z=Z)
deltabar = matrix(rep(0,nvar*nz),nrow=nz)
Vdelta = 0.01 * diag(nz)
nu = nvar+3
V = 0.01*diag(nvar)
a = 0.5
b = 0.1
Prior = list(deltabar=deltabar, Vdelta=Vdelta, nu=nu, V=V, a=a, b=b)

R = 10000
keep =1
s_beta=2.93/sqrt(nvar)
s_alpha=2.93
c=2
Mcmc = list(R=R, keep = keep, s_beta=s_beta, s_alpha=s_alpha, c=c)
out = rhierNegbinRw(Data, Prior, Mcmc)

# Unit level mean beta parameters
Mbeta = matrix(rep(0,nreg*nvar),nrow=nreg)
```

```
ndraws = length(out$alphadraw)
for (i in 1:nreg) { Mbeta[i,] = rowSums(out$Betadraw[i, , ])/ndraws }

cat(" Deltadraws ",fill=TRUE)
summary(out$Deltadraw)
cat(" Vbetadraws ",fill=TRUE)
summary(out$Vbetadraw)
cat(" alphadraws ",fill=TRUE)
summary(out$alphadraw)

if(0){
## plotting examples
plot(out$betadraw)
plot(out$alphadraw)
plot(out$Deltadraw)
}
}
```

`eMixMargDen` *Compute Marginal Densities of A Normal Mixture Averaged over MCMC Draws*

Description

`eMixMargDen` assumes that a multivariate mixture of normals has been fitted via MCMC (using `rnmixGibbs`). For each MCMC draw, the marginal densities for each component in the multivariate mixture are computed on a user-supplied grid and then averaged over draws.

Usage

```
eMixMargDen(grid, probdraw, compdraw)
```

Arguments

`grid`	array of grid points, grid[,i] are ordinates for ith component
`probdraw`	array - each row of which contains a draw of probabilities of mixture comp
`compdraw`	list of lists of draws of mixture comp moments

Details

length(compdraw) is number of MCMC draws.
compdraw[[i]] is a list draws of mu and inv Chol root for each of mixture components.
compdraw[[i]][[j]] is jth component. compdraw[[i]][[j]]$mu is mean vector; compdraw[[i]][[j]]$rooti is the UL decomp of $Sigma^{-1}$.

Value

an array of the same dimension as grid with density values.

Warning

This routine is a utility routine that does **not** check the input arguments for proper dimensions and type. To avoid errors, call with output from `rnmixGibbs`.

Author(s)

Peter Rossi, Graduate School of Business, University of Chicago, ⟨Peter.Rossi@ChicagoGsb.edu⟩.

References

For further discussion, see *Bayesian Statistics and Marketing* by Rossi, Allenby and McCulloch.
`http://faculty.chicagogsb.edu/peter.rossi/research/bsm.html`

See Also

`rnmixGibbs`

`fsh` *Flush Console Buffer*

Description

Flush contents of console buffer. This function only has an effect on the Windows GUI.

Usage

```
fsh()
```

Value

No value is returned.

Author(s)

Peter Rossi, Graduate School of Business, University of Chicago, ⟨Peter.Rossi@ChicagoGsb.edu⟩.

`ghkvec`	*Compute GHK approximation to Multivariate Normal Integrals*

Description

`ghkvec` computes the GHK approximation to the integral of a multivariate normal density over a half plane defined by a set of truncation points.

Usage

```
ghkvec(L, trunpt, above, r)
```

Arguments

`L`	lower triangular Cholesky root of Covariance matrix
`trunpt`	vector of truncation points
`above`	vector of indicators for truncation above(1) or below(0)
`r`	number of draws to use in GHK

Value

approximation to integral

Note

`ghkvec` can accept a vector of truncations and compute more than one integral. That is, length(trunpt)/length(above) number of different integrals, each with the same Sigma and mean 0 but different truncation points. See example below for an example with two integrals at different truncation points.

Author(s)

Peter Rossi, Graduate School of Business, University of Chicago, ⟨Peter.Rossi@ChicagoGsb.edu⟩.

References

For further discussion, see *Bayesian Statistics and Marketing* by Rossi,Allenby and McCulloch, Chapter 2.
http://faculty.chicagogsb.edu/peter.rossi/research/bsm.html

Examples

```
##

Sigma=matrix(c(1,.5,.5,1),ncol=2)
L=t(chol(Sigma))
trunpt=c(0,0,1,1)
above=c(1,1)
ghkvec(L,trunpt,above,100)
```

llmnl *Evaluate Log Likelihood for Multinomial Logit Model*

Description

llmnl evaluates log-likelihood for the multinomial logit model.

Usage

```
llmnl(beta,y, X)
```

Arguments

beta	k x 1 coefficient vector
y	n x 1 vector of obs on y (1,..., p)
X	n*p x k Design matrix (use createX to make)

Details

Let $mu_i = X_i\beta$, then $Pr(y_i = j) = exp(mu_{i,j})/\sum_k exp(mu_{i,k})$.
X_i is the submatrix of X corresponding to the ith observation. X has n*p rows.

Use createX to create X.

Value

value of log-likelihood (sum of log prob of observed multinomial outcomes).

Warning

This routine is a utility routine that does **not** check the input arguments for proper dimensions and type.

Author(s)

Peter Rossi, Graduate School of Business, University of Chicago, ⟨Peter.Rossi@ChicagoGsb.edu⟩.

References

For further discussion, see *Bayesian Statistics and Marketing* by Rossi, Allenby and McCulloch.
http://faculty.chicagogsb.edu/peter.rossi/research/bsm.html

See Also

createX, rmnlIndepMetrop

Examples

```
##
## Not run: ll=llmnl(beta,y,X)
```

`llmnp` *Evaluate Log Likelihood for Multinomial Probit Model*

Description

`llmnp` evaluates the log-likelihood for the multinomial probit model.

Usage

```
llmnp(beta, Sigma, X, y, r)
```

Arguments

`beta`	k x 1 vector of coefficients
`Sigma`	(p-1) x (p-1) Covariance matrix of errors
`X`	X is n*(p-1) x k array. X is from differenced system.
`y`	y is vector of n indicators of multinomial response (1, . . . , p).
`r`	number of draws used in GHK

Details

X is (p-1)*n x k matrix. Use `createX` with `DIFF=TRUE` to create X.

Model for each obs: $w = Xbeta + e$. $e \sim N(0, Sigma)$.

censoring mechanism:

if $y = j(j < p), w_j > max(w_{-j})$ and $w_j > 0$
if $y = p, w < 0$

To use GHK, we must transform so that these are rectangular regions e.g. if $y = 1, w_1 > 0$ and $w_1 - w_{-1} > 0$.

Define A_j such that if j=1,. . . ,p-1, $A_j w = A_j mu + A_j e > 0$ is equivalent to $y = j$. Thus, if y=j, we have $A_j e > -A_j mu$. Lower truncation is $-A_j mu$ and $cov = A_j Sigmat(A_j)$. For $j = p$, $e < -mu$.

Value

value of log-likelihood (sum of log prob of observed multinomial outcomes).

Warning

This routine is a utility routine that does **not** check the input arguments for proper dimensions and type.

Author(s)

Peter Rossi, Graduate School of Business, University of Chicago, ⟨Peter.Rossi@ChicagoGsb.edu⟩.

References

For further discussion, see *Bayesian Statistics and Marketing* by Rossi, Allenby and McCulloch, Chapters 2 and 4.
http://faculty.chicagogsb.edu/peter.rossi/research/bsm.html

See Also

createX, rmnpGibbs

Examples

```
##
## Not run: ll=llmnp(beta,Sigma,X,y,r)
```

---

llnhlogit *Evaluate Log Likelihood for non-homothetic Logit Model*

Description

`llmnp` evaluates log-likelihood for the Non-homothetic Logit model.

Usage

```
llnhlogit(theta, choice, lnprices, Xexpend)
```

Arguments

`theta`	parameter vector (see details section)
`choice`	n x 1 vector of choice (1, . . . , p)
`lnprices`	n x p array of log-prices
`Xexpend`	n x d array of vars predicting expenditure

Details

Non-homothetic logit model with: $ln(psi_i(U)) = alpha_i - e^{k_i}U$

Structure of theta vector
alpha: (p x 1) vector of utility intercepts.
k: (p x 1) vector of utility rotation parms.
gamma: (k x 1) – expenditure variable coefs.
tau: (1 x 1) – logit scale parameter.

Value

value of log-likelihood (sum of log prob of observed multinomial outcomes).

Warning

This routine is a utility routine that does **not** check the input arguments for proper dimensions and type.

Author(s)

Peter Rossi, Graduate School of Business, University of Chicago ⟨Peter.Rossi@ChicagoGsb.edu⟩.

References

For further discussion, see *Bayesian Statistics and Marketing* by Rossi, Allenby and McCulloch,Chapter 4.
http://faculty.chicagogsb.edu/peter.rossi/research/bsm.html

See Also

simnhlogit

Examples

```
##
## Not run: ll=llnhlogit(theta,choice,lnprices,Xexpend)
```

`lndIChisq` *Compute Log of Inverted Chi-Squared Density*

Description

`lndIChisq` computes the log of an Inverted Chi-Squared Density.

Usage

```
lndIChisq(nu, ssq, x)
```

Arguments

nu	d.f. parameter
ssq	scale parameter
x	ordinate for density evaluation

Details

$Z = \nu * ssq/\chi^2_\nu$, $Z \sim$ Inverted Chi-Squared.
`lndIChisq` computes the complete log-density, including normalizing constants.

Value

log density value

Warning

This routine is a utility routine that does **not** check the input arguments for proper dimensions and type.

Author(s)

Peter Rossi, Graduate School of Business, University of Chicago, ⟨Peter.Rossi@ChicagoGsb.edu⟩.

References

For further discussion, see *Bayesian Statistics and Marketing* by Rossi, Allenby and McCulloch, Chapter 2.
`http://faculty.chicagogsb.edu/peter.rossi/research/bsm.html`

See Also

`dchisq`

Examples

```
##
lndIChisq(3,1,2)
```

`lndIWishart` *Compute Log of Inverted Wishart Density*

Description

`lndIWishart` computes the log of an Inverted Wishart density.

Usage

```
lndIWishart(nu, V, IW)
```

Arguments

`nu`	d.f. parameter
`V`	"location" parameter
`IW`	ordinate for density evaluation

Details

$Z \sim$ Inverted Wishart(nu,V).
in this parameterization, $E[Z] = 1/(nu-k-1)V$, V is a k x k matrix `lndIWishart` computes the complete log-density, including normalizing constants.

Value

log density value

Warning

This routine is a utility routine that does **not** check the input arguments for proper dimensions and type.

Author(s)

Peter Rossi, Graduate School of Business, University of Chicago, ⟨Peter.Rossi@ChicagoGsb.edu⟩.

References

For further discussion, see *Bayesian Statistics and Marketing* by Rossi, Allenby and McCulloch, Chapter 2.
http://faculty.chicagogsb.edu/peter.rossi/research/bsm.html

See Also

rwishart

Examples

```
##
lndIWishart(5,diag(3),(diag(3)+.5))
```

`lndMvn` *Compute Log of Multivariate Normal Density*

Description

`lndMvn` computes the log of a Multivariate Normal Density.

Usage

```
lndMvn(x, mu, rooti)
```

Arguments

`x`	density ordinate
`mu`	mu vector
`rooti`	inv of Upper Triangular Cholesky root of Sigma

Details

$z \sim N(mu, \Sigma)$

Value

log density value

Warning

This routine is a utility routine that does **not** check the input arguments for proper dimensions and type.

Author(s)

Peter Rossi, Graduate School of Business, University of Chicago, ⟨Peter.Rossi@ChicagoGsb.edu⟩.

References

For further discussion, see *Bayesian Statistics and Marketing* by Rossi, Allenby and McCulloch, Chapter 2.
http://faculty.chicagogsb.edu/peter.rossi/research/bsm.html

See Also

lndMvst

Examples

```
##
Sigma=matrix(c(1,.5,.5,1),ncol=2)
lndMvn(x=c(rep(0,2)),mu=c(rep(0,2)),rooti=backsolve(chol(Sigma),diag(2)))
```

---

lndMvst — *Compute Log of Multivariate Student-t Density*

Description

`lndMvst` computes the log of a Multivariate Student-t Density.

Usage

```
lndMvst(x, nu, mu, rooti,NORMC)
```

Arguments

`x`	density ordinate
`nu`	d.f. parameter
`mu`	mu vector
`rooti`	inv of Cholesky root of Sigma
`NORMC`	include normalizing constant, def: FALSE

Details

$z \sim MVst(mu, nu, \Sigma)$

Value

log density value

Warning

This routine is a utility routine that does **not** check the input arguments for proper dimensions and type.

Author(s)

Peter Rossi, Graduate School of Business, University of Chicago, ⟨Peter.Rossi@ChicagoGsb.edu⟩.

References

For further discussion, see *Bayesian Statistics and Marketing* by Rossi, Allenby and McCulloch, Chapter 2.
http://faculty.chicagogsb.edu/peter.rossi/research/bsm.html

See Also

lndMvn

Examples

```
##
Sigma=matrix(c(1,.5,.5,1),ncol=2)
lndMvst(x=c(rep(0,2)),nu=4,mu=c(rep(0,2)),rooti=backsolve(chol(Sigma),diag(2)))
```

---

logMargDenNR — *Compute Log Marginal Density Using Newton-Raftery Approx*

Description

`logMargDenNR` computes log marginal density using the Newton-Raftery approximation.
Note: this approximation can be influenced by outliers in the vector of log-likelihoods. Use with **care** .

Usage

```
logMargDenNR(ll)
```

Arguments

`ll`	vector of log-likelihoods evaluated at length(ll) MCMC draws

Value

approximation to log marginal density value.

Warning

This routine is a utility routine that does **not** check the input arguments for proper dimensions and type.

Author(s)

Peter Rossi, Graduate School of Business, University of Chicago, ⟨Peter.Rossi@ChicagoGsb.edu⟩.

References

For further discussion, see *Bayesian Statistics and Marketing* by Rossi, Allenby and McCulloch, Chapter 6.
http://faculty.chicagogsb.edu/peter.rossi/research/bsm.html

`margarine` *Household Panel Data on Margarine Purchases*

Description

Panel data on purchases of margarine by 516 households. Demographic variables are included.

Usage

```
data(margarine)
```

Format

This is an R object that is a list of two data frames, list(choicePrice,demos)

List of 2
$ choicePrice:'data.frame': 4470 obs. of 12 variables:
... $ hhid : int [1:4470] 2100016 2100016 2100016 2100016
... $ choice : num [1:4470] 1 1 1 1 1 4 1 1 4 1
... $ PPk_Stk : num [1:4470] 0.66 0.63 0.29 0.62 0.5 0.58 0.29
... $ PBB_Stk : num [1:4470] 0.67 0.67 0.5 0.61 0.58 0.45 0.51
... $ PFl_Stk : num [1:4470] 1.09 0.99 0.99 0.99 0.99 0.99 0.99
... $ PHse_Stk: num [1:4470] 0.57 0.57 0.57 0.57 0.45 0.45 0.29
... $ PGen_Stk: num [1:4470] 0.36 0.36 0.36 0.36 0.33 0.33 0.33
... $ PImp_Stk: num [1:4470] 0.93 1.03 0.69 0.75 0.72 0.72 0.72
... $ PSS_Tub : num [1:4470] 0.85 0.85 0.79 0.85 0.85 0.85 0.85
... $ PPk_Tub : num [1:4470] 1.09 1.09 1.09 1.09 1.07 1.07 1.07
... $ PFl_Tub : num [1:4470] 1.19 1.19 1.19 1.19 1.19 1.19 1.19
... $ PHse_Tub: num [1:4470] 0.33 0.37 0.59 0.59 0.59 0.59 0.59

Pk is Parkay; BB is BlueBonnett, Fl is Fleischmanns, Hse is house, Gen is generic, Imp is Imperial, SS is Shed Spread. _Stk indicates stick, _Tub indicates Tub form.

$ demos :'data.frame': 516 obs. of 8 variables:
...$ hhid : num [1:516] 2100016 2100024 2100495 2100560
...$ Income : num [1:516] 32.5 17.5 37.5 17.5 87.5 12.5
...$ Fs3_4 : int [1:516] 0 1 0 0 0 0 0 0 0 0
...$ Fs5 : int [1:516] 0 0 0 0 0 0 0 0 1 0
...$ Fam_Size : int [1:516] 2 3 2 1 1 2 2 2 5 2
...$ college : int [1:516] 1 1 0 0 1 0 1 0 1 1
...$ whtcollar: int [1:516] 0 1 0 1 1 0 0 0 1 1
...$ retired : int [1:516] 1 1 1 0 0 1 0 1 0 0

Fs3_4 is dummy (family size 3-4). Fs5 is dummy for family size >= 5. college,whtcollar,retired are dummies reflecting these statuses.

Details

choice is a multinomial indicator of one of the 10 brands (in order listed under format). All prices are in $.

Source

Allenby and Rossi (1991), "Quality Perceptions and Asymmetric Switching Between Brands," *Marketing Science* 10, 185-205.

References

Chapter 5, *Bayesian Statistics and Marketing* by Rossi et al.
http://faculty.chicagogsb.edu/peter.rossi/research/bsm.html

Examples

```
data(margarine)
cat(" Table of Choice Variable ",fill=TRUE)
print(table(margarine$choicePrice[,2]))
cat(" Means of Prices",fill=TRUE)
mat=apply(as.matrix(margarine$choicePrice[,3:12]),2,mean)
print(mat)
cat(" Quantiles of Demographic Variables",fill=TRUE)
mat=apply(as.matrix(margarine$demos[,2:8]),2,quantile)
print(mat)

##
## example of processing for use with rhierMnlRwMixture
##
if(0)
{
select= c(1:5,7)  ## select brands
chPr=as.matrix(margarine$choicePrice)
## make sure to log prices
chPr=cbind(chPr[,1],chPr[,2],log(chPr[,2+select]))
```

```
demos=as.matrix(margarine$demos[,c(1,2,5)])

## remove obs for other alts
chPr=chPr[chPr[,2] <= 7,]
chPr=chPr[chPr[,2] != 6,]

## recode choice
chPr[chPr[,2] == 7,2]=6

hhidl=levels(as.factor(chPr[,1]))
lgtdata=NULL
nlgt=length(hhidl)
p=length(select)  ## number of choice alts
ind=1
for (i in 1:nlgt) {
   nobs=sum(chPr[,1]==hhidl[i])
   if(nobs >=5) {
      data=chPr[chPr[,1]==hhidl[i],]
      y=data[,2]
      names(y)=NULL
      X=createX(p=p,na=1,Xa=data[,3:8],nd=NULL,Xd=NULL,INT=TRUE,base=1)
       lgtdata[[ind]]=list(y=y,X=X,hhid=hhidl[i]); ind=ind+1
   }
}
nlgt=length(lgtdata)
##
## now extract demos corresponding to hhs in lgtdata
##
Z=NULL
nlgt=length(lgtdata)
for(i in 1:nlgt){
   Z=rbind(Z,demos[demos[,1]==lgtdata[[i]]$hhid,2:3])
}
##
## take log of income and family size and demean
##
Z=log(Z)
Z[,1]=Z[,1]-mean(Z[,1])
Z[,2]=Z[,2]-mean(Z[,2])

keep=5
R=20000
mcmc1=list(keep=keep,R=R)
out=rhierMnlRwMixture(Data=list(p=p,lgtdata=lgtdata,Z=Z),Prior=list(ncomp=1),Mcmc=mcmc1)

summary(out$Deltadraw)
summary(out$nmix)

if(0){
## plotting examples
plot(out$nmix)
plot(out$Deltadraw)}
}
```

mixDen	*Compute Marginal Density for Multivariate Normal Mixture*

Description

mixDen computes the marginal density for each component of a normal mixture at each of the points on a user-specifed grid.

Usage

```
mixDen(x, pvec, comps)
```

Arguments

x	array - ith column gives grid points for ith variable
pvec	vector of mixture component probabilites
comps	list of lists of components for normal mixture

Details

length(comps) is the number of mixture components. comps[[j]] is a list of parameters of the jth component. comps[[j]]$mu is mean vector; comps[[j]]$rooti is the UL decomp of $Sigma^{-1}$.

Value

an array of the same dimension as grid with density values.

Warning

This routine is a utility routine that does **not** check the input arguments for proper dimensions and type.

Author(s)

Peter Rossi, Graduate School of Business, University of Chicago ⟨Peter.Rossi@ChicagoGsb.edu⟩.

References

For further discussion, see *Bayesian Statistics and Marketing* by Rossi, Allenby and McCulloch, Chapter 3.
http://faculty.chicagogsb.edu/peter.rossi/research/bsm.html

See Also

rnmixGibbs

Examples

```
## Not run:
##
##  see examples in rnmixGibbs documentation
##
## End(Not run)
```

`mixDenBi` *Compute Bivariate Marginal Density for a Normal Mixture*

Description

`mixDenBi` computes the implied bivariate marginal density from a mixture of normals with specified mixture probabilities and component parameters.

Usage

```
mixDenBi(i, j, xi, xj, pvec, comps)
```

Arguments

`i`	index of first variable
`j`	index of second variable
`xi`	grid of values of first variable
`xj`	grid of values of second variable
`pvec`	normal mixture probabilities
`comps`	list of lists of components

Details

length(comps) is the number of mixture components. comps[[j]] is a list of parameters of the jth component. comps[[j]]$mu is mean vector; comps[[j]]$rooti is the UL decomp of $Sigma^{-1}$.

Value

an array (length(xi)=length(xj) x 2) with density value

Warning

This routine is a utility routine that does **not** check the input arguments for proper dimensions and type.

Author(s)

Peter Rossi, Graduate School of Business, University of Chicago ⟨Peter.Rossi@ChicagoGsb.edu⟩.

References

For further discussion, see *Bayesian Statistics and Marketing* by Rossi, Allenby and McCulloch, Chapter 3.
http://faculty.chicagogsb.edu/peter.rossi/research/bsm.html

See Also

rnmixGibbs, mixDen

Examples

```
## Not run:
##
##  see examples in rnmixGibbs documentation
##
## End(Not run)
```

`mnlHess` *Computes -Expected Hessian for Multinomial Logit*

Description

`mnlHess` computes -Expected[Hessian] for Multinomial Logit Model

Usage

```
mnlHess(beta,y, X)
```

Arguments

`beta`	k x 1 vector of coefficients
`y`	n x 1 vector of choices, (1, ...,p)
`X`	n*p x k Design matrix

Details

See `llmnl` for information on structure of X array. Use `createX` to make X.

Value

k x k matrix

Warning

This routine is a utility routine that does **not** check the input arguments for proper dimensions and type.

Author(s)

Peter Rossi, Graduate School of Business, University of Chicago, ⟨Peter.Rossi@ChicagoGsb.edu⟩.

References

For further discussion, see *Bayesian Statistics and Marketing* by Rossi, Allenby and McCulloch, Chapter 3.
http://faculty.chicagogsb.edu/peter.rossi/research/bsm.html

See Also

`llmnl`, `createX`, `rmnlIndepMetrop`

Examples

```
##
## Not run: mnlHess(beta,y,X)
```

`mnpProb` *Compute MNP Probabilities*

Description

`mnpProb` computes MNP probabilities for a given X matrix corresponding to one observation. This function can be used with output from `rmnpGibbs` to simulate the posterior distribution of market shares or fitted probabilties.

Usage

```
mnpProb(beta, Sigma, X, r)
```

Arguments

`beta`	MNP coefficients
`Sigma`	Covariance matrix of latents
`X`	X array for one observation – use `createX` to make
`r`	number of draws used in GHK (def: 100)

Details

see `rmnpGibbs` for definition of the model and the interpretation of the beta, Sigma parameters. Uses the GHK method to compute choice probabilities. To simulate a distribution of probabilities, loop over the beta, Sigma draws from `rmnpGibbs` output.

Value

p x 1 vector of choice probabilites

Author(s)

Peter Rossi, Graduate School of Business, University of Chicago, ⟨Peter.Rossi@ChicagoGsb.edu⟩.

References

For further discussion, see *Bayesian Statistics and Marketing* by Rossi,Allenby and McCulloch, Chapters 2 and 4.
`http://faculty.chicagogsb.edu/peter.rossi/research/bsm.html`

See Also

`rmnpGibbs`, `createX`

Examples

```
##
## example of computing MNP probabilites
##  here I'm thinking of Xa as having the prices of each of the 3 alternatives
Xa=matrix(c(1,.5,1.5),nrow=1)
X=createX(p=3,na=1,nd=NULL,Xa=Xa,Xd=NULL,DIFF=TRUE)
beta=c(1,-1,-2)  ## beta contains two intercepts and the price coefficient
Sigma=matrix(c(1,.5,.5,1),ncol=2)
mnpProb(beta,Sigma,X)
```

`momMix` *Compute Posterior Expectation of Normal Mixture Model Moments*

Description

`momMix` averages the moments of a normal mixture model over MCMC draws.

Usage

```
momMix(probdraw, compdraw)
```

Arguments

`probdraw`	R x ncomp list of draws of mixture probs
`compdraw`	list of length R of draws of mixture component moments

Details

R is the number of MCMC draws in argument list above.
ncomp is the number of mixture components fitted.
compdraw is a list of lists of lists with mixture components.
compdraw[[i]] is ith draw.
compdraw[[i]][[j]][[1]] is the mean parameter vector for the jth component, ith MCMC draw.
compdraw[[i]][[j]][[2]] is the UL decomposition of $Sigma^{-1}$ for the jth component, ith MCMC draw.

Value

a list of the following items ...

`mu`	Posterior Expectation of Mean
`sigma`	Posterior Expecation of Covariance Matrix
`sd`	Posterior Expectation of Vector of Standard Deviations
`corr`	Posterior Expectation of Correlation Matrix

Warning

This routine is a utility routine that does **not** check the input arguments for proper dimensions and type.

Author(s)

Peter Rossi, Graduate School of Business, University of Chicago, ⟨Peter.Rossi@ChicagoGsb.edu⟩.

References

For further discussion, see *Bayesian Statistics and Marketing* by Rossi, Allenby and McCulloch, Chapter 5.
`http://faculty.chicagogsb.edu/peter.rossi/research/bsm.html`

See Also

`rmixGibbs`

`nmat` *Convert Covariance Matrix to a Correlation Matrix*

Description

`nmat` converts a covariance matrix (stored as a vector, col by col) to a correlation matrix (also stored as a vector).

Usage

```
nmat(vec)
```

Arguments

`vec`	k x k Cov matrix stored as a k*k x 1 vector (col by col)

Details

This routine is often used with apply to convert an R x (k*k) array of covariance MCMC draws to correlations. As in `corrdraws=apply(vardraws,1,nmat)`

Value

k*k x 1 vector with correlation matrix

Warning

This routine is a utility routine that does **not** check the input arguments for proper dimensions and type.

Author(s)

Peter Rossi, Graduate School of Business, University of Chicago, ⟨Peter.Rossi@ChicagoGsb.edu⟩.

Examples

```
##
set.seed(66)
X=matrix(rnorm(200,4),ncol=2)
Varmat=var(X)
nmat(as.vector(Varmat))
```

---

numEff *Compute Numerical Standard Error and Relative Numerical Efficiency*

Description

`numEff` computes the numerical standard error for the mean of a vector of draws as well as the relative numerical efficiency (ratio of variance of mean of this time series process relative to iid sequence).

Usage

```
numEff(x, m = as.integer(min(length(x), (100/sqrt(5000)) * sqrt(length(x)))))
```

Arguments

`x`	R x 1 vector of draws
`m`	number of lags for autocorrelations

Details

default for number of lags is chosen so that if R = 5000, m =100 and increases as the sqrt(R).

Value

`stderr`	standard error of the mean of x
`f`	variance ratio (relative numerical efficiency)

Warning

This routine is a utility routine that does **not** check the input arguments for proper dimensions and type.

Author(s)

Peter Rossi, Graduate School of Business, University of Chicago, ⟨Peter.Rossi@ChicagoGsb.edu⟩.

References

For further discussion, see *Bayesian Statistics and Marketing* by Rossi, Allenby and McCulloch, Chapter 3.
http://faculty.chicagogsb.edu/peter.rossi/research/bsm.html

Examples

```
numEff(rnorm(1000),m=20)
numEff(rnorm(1000))
```

`orangeJuice` *Store-level Panel Data on Orange Juice Sales*

Description

yx, weekly sales of refrigerated orange juice at 83 stores.
storedemo, contains demographic information on those stores.

Usage

```
data(orangeJuice)
```

Format

This R object is a list of two data frames, list(yx,storedemo).

List of 2
$ yx :'data.frame': 106139 obs. of 19 variables:
. . . $ store : int [1:106139] 2 2 2 2 2 2 2 2 2 2
. . . $ brand : int [1:106139] 1 1 1 1 1 1 1 1 1 1
. . . $ week : int [1:106139] 40 46 47 48 50 51 52 53 54 57
. . . $ logmove : num [1:106139] 9.02 8.72 8.25 8.99 9.09
. . . $ constant: int [1:106139] 1 1 1 1 1 1 1 1 1 1
. . . $ price1 : num [1:106139] 0.0605 0.0605 0.0605 0.0605 0.0605
. . . $ price2 : num [1:106139] 0.0605 0.0603 0.0603 0.0603 0.0603
. . . $ price3 : num [1:106139] 0.0420 0.0452 0.0452 0.0498 0.0436
. . . $ price4 : num [1:106139] 0.0295 0.0467 0.0467 0.0373 0.0311
. . . $ price5 : num [1:106139] 0.0495 0.0495 0.0373 0.0495 0.0495

. . . $ price6 : num [1:106139] 0.0530 0.0478 0.0530 0.0530 0.0530
. . . $ price7 : num [1:106139] 0.0389 0.0458 0.0458 0.0458 0.0466
. . . $ price8 : num [1:106139] 0.0414 0.0280 0.0414 0.0414 0.0414
. . . $ price9 : num [1:106139] 0.0289 0.0430 0.0481 0.0423 0.0423
. . . $ price10 : num [1:106139] 0.0248 0.0420 0.0327 0.0327 0.0327
. . . $ price11 : num [1:106139] 0.0390 0.0390 0.0390 0.0390 0.0382
. . . $ deal : int [1:106139] 1 0 0 0 0 0 1 1 1 1
. . . $ feat : num [1:106139] 0 0 0 0 0 0 0 0 0 0
. . . $ profit : num [1:106139] 38.0 30.1 30.0 29.9 29.9

1 Tropicana Premium 64 oz; 2 Tropicana Premium 96 oz; 3 Florida's Natural 64 oz;
4 Tropicana 64 oz; 5 Minute Maid 64 oz; 6 Minute Maid 96 oz;
7 Citrus Hill 64 oz; 8 Tree Fresh 64 oz; 9 Florida Gold 64 oz;
10 Dominicks 64 oz; 11 Dominicks 128 oz.

$ storedemo:'data.frame': 83 obs. of 12 variables:
. . . $ STORE : int [1:83] 2 5 8 9 12 14 18 21 28 32
. . . $ AGE60 : num [1:83] 0.233 0.117 0.252 0.269 0.178
. . . $ EDUC : num [1:83] 0.2489 0.3212 0.0952 0.2222 0.2534
. . . $ ETHNIC : num [1:83] 0.1143 0.0539 0.0352 0.0326 0.3807
. . . $ INCOME : num [1:83] 10.6 10.9 10.6 10.8 10.0
. . . $ HHLARGE : num [1:83] 0.1040 0.1031 0.1317 0.0968 0.0572
. . . $ WORKWOM : num [1:83] 0.304 0.411 0.283 0.359 0.391
. . . $ HVAL150 : num [1:83] 0.4639 0.5359 0.0542 0.5057 0.3866
. . . $ SSTRDIST: num [1:83] 2.11 3.80 2.64 1.10 9.20
. . . $ SSTRVOL : num [1:83] 1.143 0.682 1.500 0.667 1.111
. . . $ CPDIST5 : num [1:83] 1.93 1.60 2.91 1.82 0.84
. . . $ CPWVOL5 : num [1:83] 0.377 0.736 0.641 0.441 0.106

Details

`store` store number

`brand` brand indicator

`week` week number

`logmove` log of the number of units sold

`constant` a vector of 1

`price1` price of brand 1

`deal` in-store coupon activity

`feature` feature advertisement

`STORE` store number

`AGE60` percentage of the population that is aged 60 or older

`EDUC` percentage of the population that has a college degree

`ETHNIC` percent of the population that is black or Hispanic

`INCOME` median income

`HHLARGE` percentage of households with 5 or more persons

`WORKWOM` percentage of women with full-time jobs

`HVAL150` percentage of households worth more than $150,000

`SSTRDIST` distance to the nearest warehouse store

`SSTRVOL` ratio of sales of this store to the nearest warehouse store

`CPDIST5` average distance in miles to the nearest 5 supermarkets

`CPWVOL5` ratio of sales of this store to the average of the nearest five stores

Source

Alan L. Montgomery (1997), "Creating Micro-Marketing Pricing Strategies Using Supermarket Scanner Data," *Marketing Science* 16(4) 315-337.

References

Chapter 5, *Bayesian Statistics and Marketing* by Rossi et al.
http://faculty.chicagogsb.edu/peter.rossi/research/bsm.html

Examples

```
## Example
## load data
data(orangeJuice)

## print some quantiles of yx data
cat("Quantiles of the Variables in yx data",fill=TRUE)
mat=apply(as.matrix(orangeJuice$yx),2,quantile)
print(mat)

## print some quantiles of storedemo data
cat("Quantiles of the Variables in storedemo data",fill=TRUE)
mat=apply(as.matrix(orangeJuice$storedemo),2,quantile)
print(mat)

## Example 2 processing for use with rhierLinearModel
##
##
if(0)
{

## select brand 1 for analysis
brand1=orangeJuice$yx[(orangeJuice$yx$brand==1),]

store = sort(unique(brand1$store))
nreg = length(store)
nvar=14

regdata=NULL
for (reg in 1:nreg) {
```

```
        y=brand1$logmove[brand1$store==store[reg]]
        iota=c(rep(1,length(y)))
        X=cbind(iota,log(brand1$price1[brand1$store==store[reg]]),
                     log(brand1$price2[brand1$store==store[reg]]),
                     log(brand1$price3[brand1$store==store[reg]]),
                     log(brand1$price4[brand1$store==store[reg]]),
                     log(brand1$price5[brand1$store==store[reg]]),
                     log(brand1$price6[brand1$store==store[reg]]),
                     log(brand1$price7[brand1$store==store[reg]]),
                     log(brand1$price8[brand1$store==store[reg]]),
                     log(brand1$price9[brand1$store==store[reg]]),
                     log(brand1$price10[brand1$store==store[reg]]),
                     log(brand1$price11[brand1$store==store[reg]]),
                     brand1$deal[brand1$store==store[reg]],
                     brand1$feat[brand1$store==store[reg]])
        regdata[[reg]]=list(y=y,X=X)
      }

## storedemo is standardized to zero mean.

Z=as.matrix(orangeJuice$storedemo[,2:12])
dmean=apply(Z,2,mean)
for (s in 1:nreg){
        Z[s,]=Z[s,]-dmean
}
iotaz=c(rep(1,nrow(Z)))
Z=cbind(iotaz,Z)
nz=ncol(Z)

Data=list(regdata=regdata,Z=Z)
Mcmc=list(R=R,keep=1)

out=rhierLinearModel(Data=Data,Mcmc=Mcmc)

summary(out$Deltadraw)
summary(out$Vbetadraw)

if(0){
## plotting examples
plot(out$betadraw)
}
}
```

plot.bayesm.hcoef — *Plot Method for Hierarchical Model Coefs*

Description

`plot.bayesm.hcoef` is an S3 method to plot 3 dim arrays of hierarchical coefficients. Arrays are of class bayesm.hcoef with dimensions: cross-sectional unit x coef x MCMC draw.

Usage

```
## S3 method for class 'bayesm.hcoef':
plot(x,burnin,...)
```

Arguments

`x`	An object of S3 class, bayesm.hcoef
`burnin`	no draws to burnin, def: .1*R
`...`	standard graphics parameters

Details

Typically, `plot.bayesm.hcoef` will be invoked by a call to the generic plot function as in `plot(object)` where object is of class bayesm.hcoef. All of the `bayesm` hierarchical routines return draws of hierarchical coefficients in this class (see example below). One can also simply invoke `plot.bayesm.hcoef` on any valid 3-dim array as in `plot.bayesm.hcoef(betadraws)`

`plot.bayesm.hcoef` is also exported for use as a standard function, as in `plot.bayesm.hcoef(array)`.

Author(s)

Peter Rossi, Graduate School of Business, University of Chicago, ⟨Peter.Rossi@ChicagoGsb.edu⟩.

See Also

`rhierMnlRwMixture`,`rhierLinearModel`, `rhierLinearMixture`,`rhierNegbinRw`

Examples

```
##
## not run
#  out=rhierLinearModel(Data,Prior,Mcmc)
#  plot(out$betadraws)
#
```

`plot.bayesm.mat` *Plot Method for Arrays of MCMC Draws*

Description

`plot.bayesm.mat` is an S3 method to plot arrays of MCMC draws. The columns in the array correspond to parameters and the rows to MCMC draws.

Usage

```
## S3 method for class 'bayesm.mat':
plot(x,names,burnin,tvalues,TRACEPLOT,DEN,INT,CHECK_NDRAWS, ...)
```

Arguments

`x`	An object of either S3 class, bayesm.mat, or S3 class, mcmc
`names`	optional character vector of names for coefficients
`burnin`	number of draws to discard for burn-in, def: .1*nrow(X)
`tvalues`	vector of true values
`TRACEPLOT`	logical, TRUE provide sequence plots of draws and acfs, def: TRUE
`DEN`	logical, TRUE use density scale on histograms, def: TRUE
`INT`	logical, TRUE put various intervals and points on graph, def: TRUE
`CHECK_NDRAWS`	logical, TRUE check that there are at least 100 draws, def: TRUE
`...`	standard graphics parameters

Details

Typically, `plot.bayesm.mat` will be invoked by a call to the generic plot function as in `plot(object)` where object is of class bayesm.mat. All of the `bayesm` MCMC routines return draws in this class (see example below). One can also simply invoke `plot.bayesm.mat` on any valid 2-dim array as in `plot.bayesm.mat(betadraws)`.

`plot.bayesm.mat` paints (by default) on the histogram:

green "[]" delimiting 95% Bayesian Credibility Interval
yellow "()" showing +/- 2 numerical standard errors
red "|" showing posterior mean

`plot.bayesm.mat` is also exported for use as a standard function, as in `plot.bayesm.mat(matrix)`

Author(s)

Peter Rossi, Graduate School of Business, University of Chicago, ⟨Peter.Rossi@ChicagoGsb.edu⟩.

Examples

```
##
## not run
#  out=runiregGibbs(Data,Prior,Mcmc)
#  plot(out$betadraw)
#
```

`plot.bayesm.nmix` *Plot Method for MCMC Draws of Normal Mixtures*

Description

`plot.bayesm.nmix` is an S3 method to plot aspects of the fitted density from a list of MCMC draws of normal mixture components. Plots of marginal univariate and bivariate densities are produced.

Usage

```
## S3 method for class 'bayesm.nmix':
plot(x,names,burnin,Grid,bi.sel,nstd,marg,Data,ngrid,ndraw, ...)
```

Arguments

`x`	An object of S3 class bayesm.nmix
`names`	optional character vector of names for each of the dimensions
`burnin`	number of draws to discard for burn-in, def: .1*nrow(X)
`Grid`	matrix of grid points for densities, def: mean +/- nstd std deviations (if Data no supplied), range of Data if supplied)
`bi.sel`	list of vectors, each giving pairs for bivariate distributions, def: list(c(1,2))
`nstd`	number of standard deviations for default Grid, def: 2
`marg`	logical, if TRUE display marginals, def: TRUE
`Data`	matrix of data points, used to paint histograms on marginals and for grid
`ngrid`	number of grid points for density estimates, def:50
`ndraw`	number of draws to average Mcmc estimates over, def:200
`...`	standard graphics parameters

Details

Typically, `plot.bayesm.nmix` will be invoked by a call to the generic plot function as in `plot(object)` where object is of class bayesm.nmix. These objects are lists of three components. The first component is an array of draws of mixture component probabilties. The second component is not used. The third is a lists of lists of lists with draws of each of the normal components.

`plot.bayesm.nmix` can also be used as a standard function, as in `plot.bayesm.nmix(list)`.

Author(s)

Peter Rossi, Graduate School of Business, University of Chicago, ⟨Peter.Rossi@ChicagoGsb.edu⟩.

See Also

`rnmixGibbs`, `rhierMnlRwMixture`, `rhierLinearMixture`, `rDPGibbs`

Examples

```
##
## not run
#  out=rnmixGibbs(Data,Prior,Mcmc)
#  plot(out,bi.sel=list(c(1,2),c(3,4),c(1,3)))
#        # plot bivariate distributions for dimension 1,2; 3,4; and 1,3
#
```

rbiNormGibbs	*Illustrate Bivariate Normal Gibbs Sampler*

Description

`rbiNormGibbs` implements a Gibbs Sampler for the bivariate normal distribution. Intermediate moves are shown and the output is contrasted with the iid sampler. i This function is designed for illustrative/teaching purposes.

Usage

```
rbiNormGibbs(initx = 2, inity = -2, rho, burnin = 100, R = 500)
```

Arguments

`initx`	initial value of parameter on x axis (def: 2)
`inity`	initial value of parameter on y axis (def: -2)
`rho`	correlation for bivariate normals
`burnin`	burn-in number of draws (def:100)
`R`	number of MCMC draws (def:500)

Details

(theta1,theta2)   N((0,0), Sigma=matrix(c(1,rho,rho,1),ncol=2))

Value

R x 2 array of draws

Author(s)

Peter Rossi, Graduate School of Business, University of Chicago, ⟨Peter.Rossi@ChicagoGsb.edu⟩.

References

For further discussion, see *Bayesian Statistics and Marketing* by Rossi, Allenby and McCulloch, Chapters 2 and 3.
http://faculty.chicagogsb.edu/peter.rossi/research/bsm.html

Examples

```
##
## Not run:  out=rbiNormGibbs(rho=.95)
```

`rbprobitGibbs` *Gibbs Sampler (Albert and Chib) for Binary Probit*

Description

`rbprobitGibbs` implements the Albert and Chib Gibbs Sampler for the binary probit model.

Usage

```
rbprobitGibbs(Data, Prior, Mcmc)
```

Arguments

`Data`	list(X,y)
`Prior`	list(betabar,A)
`Mcmc`	list(R,keep)

Details

Model: $z = X\beta + e$. $e \sim N(0, I)$. y=1, if z> 0.

Prior: $\beta \sim N(betabar, A^{-1})$.

List arguments contain

`X` Design Matrix

`y` n x 1 vector of observations, (0 or 1)

`betabar` k x 1 prior mean (def: 0)

`A` k x k prior precision matrix (def: .01I)

`R` number of MCMC draws

`keep` thinning parameter - keep every keepth draw (def: 1)

Value

`betadraw`	R/keep x k array of betadraws

Author(s)

Peter Rossi, Graduate School of Business, University of Chicago, ⟨Peter.Rossi@ChicagoGsb.edu⟩.

References

For further discussion, see *Bayesian Statistics and Marketing* by Rossi, Allenby and McCulloch, Chapter 3.
http://faculty.chicagogsb.edu/peter.rossi/research/bsm.html

See Also

`rmnpGibbs`

Examples

```
##
## rbprobitGibbs example
##
if(nchar(Sys.getenv("LONG_TEST")) != 0) {R=2000} else {R=10}

set.seed(66)
simbprobit=
function(X,beta) {
##  function to simulate from binary probit including x variable
y=ifelse((X%*%beta+rnorm(nrow(X)))<0,0,1)
list(X=X,y=y,beta=beta)
}

nobs=200
X=cbind(rep(1,nobs),runif(nobs),runif(nobs))
beta=c(0,1,-1)
nvar=ncol(X)
simout=simbprobit(X,beta)

Data1=list(X=simout$X,y=simout$y)
Mcmc1=list(R=R,keep=1)

out=rbprobitGibbs(Data=Data1,Mcmc=Mcmc1)

summary(out$betadraw,tvalues=beta)

if(0){
## plotting example
plot(out$betadraw,tvalues=beta)
}
```

`rdirichlet` *Draw From Dirichlet Distribution*

Description

`rdirichlet` draws from Dirichlet

Usage

```
rdirichlet(alpha)
```

Arguments

`alpha`	vector of Dirichlet parms (must be > 0)

Value

Vector of draws from Dirichlet

Warning

This routine is a utility routine that does **not** check the input arguments for proper dimensions and type.

Author(s)

Peter Rossi, Graduate School of Business, University of Chicago, ⟨Peter.Rossi@ChicagoGsb.edu⟩.

References

For further discussion, see *Bayesian Statistics and Marketing* by Rossi, Allenby and McCulloch, Chapter 2.
http://faculty.chicagogsb.edu/peter.rossi/research/bsm.html

Examples

```
##
set.seed(66)
rdirichlet(c(rep(3,5)))
```

`rDPGibbs` *Density Estimation with Dirichlet Process Prior and Normal Base*

Description

`rDPGibbs` implements a Gibbs Sampler to draw from the posterior for a normal mixture problem with a Dirichlet Process prior. A natural conjugate base prior is used along with priors on the hyper parameters of this distribution. One interpretation of this model is as a normal mixture with a random number of components that can grow with the sample size.

Usage

```
rDPGibbs(Prior, Data, Mcmc)
```

Arguments

`Prior`	list(Prioralpha,lambda_hyper)
`Data`	list(y)
`Mcmc`	list(R,keep,maxuniq,SCALE,gridsize)

Details

Model:
$y_i \sim N(mu_i, Sigma_i)$.

Priors:
$theta_i = (mu_i, Sigma_i) \sim DP(G_0(lambda), alpha)$
$G_0(lambda):$
$mu_i|Sigma_i \sim N(0, Sigma_i(x)a^{-1})$
$Sigma_i \sim IW(nu, nu * v * I)$
$lambda(a, nu, v):$
$a \sim$ uniform on grid[alim[1],alimb[2]]
$nu \sim$ uniform on grid[dim(data)-1 + exp(nulim[1]),dim(data)-1 +exp(nulim[2])]
$v \sim$ uniform on grid[vlim[1],vlim[2]]
$alpha \sim (1 - (alpha - alphamin)/(alphamax - alphamin))^power$
alpha= alphamin then expected number of components = Istarmin
alpha= alphamax then expected number of components = Istarmax

list arguments

Data:

- `y` N x k matrix of observations on k dimensional data

Prioralpha:

- `Istarmin` expected number of components at lower bound of support of alpha
- `Istarmax` expected number of components at upper bound of support of alpha
- `power` power parameter for alpha prior

lambda_hyper:

- `alim` defines support of a distribution,def:c(.01,2)
- `nulim` defines support of nu distribution, def:c(.01,3)
- `vlim` defines support of v distribution, def:c(.1,4)

Mcmc:

- `R` number of mcmc draws
- `keep` thinning parm, keep every keepth draw
- `maxuniq` storage constraint on the number of unique components
- `SCALE` should data be scaled by mean,std deviation before posterior draws, def: TRUE
- `gridsize` number of discrete points for hyperparameter priors,def: 20

output:
the basic output are draws from the predictive distribution of the data in the object, `nmix`. The average of these draws is the Bayesian analogue of a density estimate.

nmix:

`probdraw` R/keep x 1 matrix of 1s

`zdraw` R/keep x N matrix of draws of indicators of which component each obs is assigned to

`compdraw` R/keep list of draws of normals

Output of the components is in the form of a list of lists.
compdraw[[i]] is ith draw – list of lists.
compdraw[[i]][[1]] is list of parms for a draw from predictive.
compdraw[[i]][1]][[1]] is the mean vector. compdraw[[i]][[1]][[2]] is the inverse of Cholesky root. $Sigma$ = t(R)%*%R, R^{-1} = compdraw[[i]][[1]][[2]].

Value

`nmix`	a list containing: probdraw,zdraw,compdraw
`alphadraw`	vector of draws of DP process tightness parameter
`nudraw`	vector of draws of base prior hyperparameter
`adraw`	vector of draws of base prior hyperparameter
`vdraw`	vector of draws of base prior hyperparameter

Note

we parameterize the prior on $Sigma_i$ such that $mode(Sigma) = nu/(nu+2)vI$. The support of nu enforces valid IW density; $nulim[1] > 0$

We use the structure for `nmix` that is compatible with the `bayesm` routines for finite mixtures of normals. This allows us to use the same summary and plotting methods.

The default choices of alim,nulim, and vlim determine the location and approximate size of candidate "atoms" or possible normal components. The defaults are sensible given that we scale the data. Without scaling, you want to insure that alim is set for a wide enough range of values (remember a is a precision parameter) and the v is big enough to propose Sigma matrices wide enough to cover the data range.

A careful analyst should look at the posterior distribution of a, nu, v to make sure that the support is set correctly in alim, nulim, vlim. In other words, if we see the posterior bunched up at one end of these support ranges, we should widen the range and rerun.

If you want to force the procedure to use many small atoms, then set nulim to consider only large values and set vlim to consider only small scaling constants. Set Istarmax to a large number. This will create a very "lumpy" density estimate somewhat like the classical Kernel density estimates. Of course, this is not advised if you have a prior belief that densities are relatively smooth.

Author(s)

Peter Rossi, Graduate School of Business, University of Chicago, ⟨Peter.Rossi@ChicagoGsb.edu⟩.

See Also

rnmixGibbs,rmixture, rmixGibbs , eMixMargDen, momMix, mixDen, mixDenBi

Examples

```
if(nchar(Sys.getenv("LONG_TEST")) != 0) {R=2000} else {R=10}

## simulate univariate data from Chi-Sq

set.seed(66)
N=200
chisqdf=8; y1=as.matrix(rchisq(N,df=chisqdf))

## set arguments for rDPGibbs

Data1=list(y=y1)
Prioralpha=list(Istarmin=1,Istarmax=10,power=.8)
Prior1=list(Prioralpha=Prioralpha)

Mcmc=list(R=R,keep=1,maxuniq=200)

out1=rDPGibbs(Prior=Prior1,Data=Data1,Mcmc)

if(0){
## plotting examples
rgi=c(0,20); grid=matrix(seq(from=rgi[1],to=rgi[2],length.out=50),ncol=1)
deltax=(rgi[2]-rgi[1])/nrow(grid)
plot(out1$nmix,Grid=grid,Data=y1)
## plot true density with historgram
plot(range(grid[,1]),1.5*range(dchisq(grid[,1],df=chisqdf)),type="n",xlab=paste("Chisq ; ",N," obs",sep="
hist(y1,xlim=rgi,freq=FALSE,col="yellow",breaks=20,add=TRUE)
lines(grid[,1],dchisq(grid[,1],df=chisqdf)/(sum(dchisq(grid[,1],df=chisqdf))*deltax),col="blue",lwd=2)
}

## simulate bivariate data from the  "Banana" distribution (Meng and Barnard)
banana=function(A,B,C1,C2,N,keep=10,init=10)
{ R=init*keep+N*keep
   x1=x2=0
   bimat=matrix(double(2*N),ncol=2)
  for (r in 1:R)
  { x1=rnorm(1,mean=(B*x2+C1)/(A*(x2^2)+1),sd=sqrt(1/(A*(x2^2)+1)))
  x2=rnorm(1,mean=(B*x2+C2)/(A*(x1^2)+1),sd=sqrt(1/(A*(x1^2)+1)))
  if (r>init*keep && r%%keep==0) {mkeep=r/keep; bimat[mkeep-init,]=c(x1,x2)} }
return(bimat)
}

set.seed(66)
nvar2=2
A=0.5; B=0; C1=C2=3
y2=banana(A=A,B=B,C1=C1,C2=C2,1000)

Data2=list(y=y2)
```

```
Prioralpha=list(Istarmin=1,Istarmax=10,power=.8)
Prior2=list(Prioralpha=Prioralpha)
Mcmc=list(R=R,keep=1,maxuniq=200)

out2=rDPGibbs(Prior=Prior2,Data=Data2,Mcmc)

if(0){
## plotting examples

rx1=range(y2[,1]); rx2=range(y2[,2])
x1=seq(from=rx1[1],to=rx1[2],length.out=50)
x2=seq(from=rx2[1],to=rx2[2],length.out=50)
grid=cbind(x1,x2)

plot(out2$nmix,Grid=grid,Data=y2)

## plot true bivariate density
tden=matrix(double(50*50),ncol=50)
for (i in 1:50){ for (j in 1:50)
      {tden[i,j]=exp(-0.5*(A*(x1[i]^2)*(x2[j]^2)+(x1[i]^2)+(x2[j]^2)-2*B*x1[i]*x2[j]-2*C1*x1[i]-2*C2*x2[j
}
tden=tden/sum(tden)
image(x1,x2,tden,col=terrain.colors(100),xlab="",ylab="")
contour(x1,x2,tden,add=TRUE,drawlabels=FALSE)
title("True Density")
}
```

---

rhierBinLogit *MCMC Algorithm for Hierarchical Binary Logit*

Description

`rhierBinLogit` implements an MCMC algorithm for hierarchical binary logits with a normal heterogeneity distribution. This is a hybrid sampler with a RW Metropolis step for unit-level logit parameters.

`rhierBinLogit` is designed for use on choice-based conjoint data with partial profiles. The Design matrix is based on differences of characteristics between two alternatives. See Appendix A of *Bayesian Statistics and Marketing* for details.

Usage

```
rhierBinLogit(Data, Prior, Mcmc)
```

Arguments

`Data`	list(lgtdata,Z) (note: Z is optional)
`Prior`	list(Deltabar,ADelta,nu,V) (note: all are optional)
`Mcmc`	list(sbeta,R,keep) (note: all but R are optional)

Details

Model:
$y_{hi} = 1$ with $pr = exp(x'_{hi} beta_h)/(1 + exp(x'_{hi} beta_h)$. $beta_h$ is nvar x 1.
h=1,...,length(lgtdata) units or "respondents" for survey data.

$beta_h$= ZDelta[h,] + u_h.
Note: here ZDelta refers to Z%*%Delta, ZDelta[h,] is hth row of this product.
Delta is an nz x nvar array.

$u_h \sim N(0, V_{beta})$.

Priors:
$delta = vec(Delta) \sim N(vec(Deltabar), V_{beta}(x)ADelta^{-1})$
$V_{beta} \sim IW(nu, V)$

Lists contain:

- `lgtdata` list of lists with each cross-section unit MNL data
- `lgtdata[[h]]$y` n_h vector of binary outcomes (0,1)
- `lgtdata[[h]]$X` n_h by nvar design matrix for hth unit
- `Deltabar` nz x nvar matrix of prior means (def: 0)
- `ADelta` prior prec matrix (def: .01I)
- `nu` d.f. parm for IW prior on norm comp Sigma (def: nvar+3)
- `V` pds location parm for IW prior on norm comp Sigma (def: nuI)
- `sbeta` scaling parm for RW Metropolis (def: .2)
- `R` number of MCMC draws
- `keep` MCMC thinning parm: keep every keepth draw (def: 1)

Value

a list containing:

- `Deltadraw` R/keep x nz*nvar matrix of draws of Delta
- `betadraw` nlgt x nvar x R/keep array of draws of betas
- `Vbetadraw` R/keep x nvar*nvar matrix of draws of Vbeta
- `llike` R/keep vector of log-like values
- `reject` R/keep vector of reject rates over nlgt units

Note

Some experimentation with the Metropolis scaling paramter (sbeta) may be required.

Author(s)

Peter Rossi, Graduate School of Business, University of Chicago, ⟨Peter.Rossi@ChicagoGsb.edu⟩.

References

For further discussion, see *Bayesian Statistics and Marketing* by Rossi, Allenby and McCulloch, Chapter 5.
http://faculty.chicagogsb.edu/peter.rossi/research/bsm.html

See Also

rhierMnlRwMixture

Examples

```
##
if(nchar(Sys.getenv("LONG_TEST")) != 0) {R=10000} else {R=10}

set.seed(66)
nvar=5                          ## number of coefficients
nlgt=1000                       ## number of cross-sectional units
nobs=10                         ## number of observations per unit
nz=2                            ## number of regressors in mixing distribution

## set hyper-parameters
##     B=ZDelta + U

Z=matrix(c(rep(1,nlgt),runif(nlgt,min=-1,max=1)),nrow=nlgt,ncol=nz)
Delta=matrix(c(-2,-1,0,1,2,-1,1,-.5,.5,0),nrow=nz,ncol=nvar)
iota=matrix(1,nrow=nvar,ncol=1)
Vbeta=diag(nvar)+.5*iota%*%t(iota)

## simulate data
lgtdata=NULL

for (i in 1:nlgt)
{ beta=t(Delta)%*%Z[i,]+as.vector(t(chol(Vbeta))%*%rnorm(nvar))
  X=matrix(runif(nobs*nvar),nrow=nobs,ncol=nvar)
  prob=exp(X%*%beta)/(1+exp(X%*%beta))
  unif=runif(nobs,0,1)
  y=ifelse(unif<prob,1,0)
  lgtdata[[i]]=list(y=y,X=X,beta=beta)
}

out=rhierBinLogit(Data=list(lgtdata=lgtdata,Z=Z),Mcmc=list(R=R))

cat("Summary of Delta draws",fill=TRUE)
summary(out$Deltadraw,tvalues=as.vector(Delta))
cat("Summary of Vbeta draws",fill=TRUE)
summary(out$Vbetadraw,tvalues=as.vector(Vbeta[upper.tri(Vbeta,diag=TRUE)]))

if(0){
## plotting examples
plot(out$Deltadraw,tvalues=as.vector(Delta))
plot(out$betadraw)
plot(out$Vbetadraw,tvalues=as.vector(Vbeta[upper.tri(Vbeta,diag=TRUE)]))
```

}

rhierLinearMixture *Gibbs Sampler for Hierarchical Linear Model*

Description

rhierLinearMixture implements a Gibbs Sampler for hierarchical linear models with a mixture of normals prior.

Usage

```
rhierLinearMixture(Data, Prior, Mcmc)
```

Arguments

Data	list(regdata,Z) (Z optional).
Prior	list(deltabar,Ad,mubar,Amu,nu,V,nu.e,ssq,ncomp) (all but ncomp are optional).
Mcmc	list(R,keep) (R required).

Details

Model: length(regdata) regression equations.
$y_i = X_i beta_i + e_i$. $e_i \sim N(0, tau_i)$. nvar X vars in each equation.

Priors:
$tau_i \sim$ nu.e*$ssq_i/\chi^2_{nu.e}$. tau_i is the variance of e_i.

$beta_i$= ZDelta[i,] + u_i.
Note: here ZDelta refers to Z%*%D, ZDelta[i,] is ith row of this product.
Delta is an nz x nvar array.

$u_i \sim N(mu_{ind}, Sigma_{ind})$. $ind \sim$ multinomial(pvec).

$pvec \sim$ dirichlet (a)
$delta = vec(Delta) \sim N(deltabar, A_d^{-1})$
$mu_j \sim N(mubar, Sigma_j(x)Amu^{-1})$
$Sigma_j \sim$ IW(nu,V)

List arguments contain:

- `regdata` list of lists with X,y matrices for each of length(regdata) regressions
- `regdata[[i]]$X` X matrix for equation i
- `regdata[[i]]$y` y vector for equation i
- `deltabar` nz*nvar vector of prior means (def: 0)

`Ad` prior prec matrix for vec(Delta) (def: .01I)

`mubar` nvar x 1 prior mean vector for normal comp mean (def: 0)

`Amu` prior precision for normal comp mean (def: .01I)

`nu` d.f. parm for IW prior on norm comp Sigma (def: nvar+3)

`V` pds location parm for IW prior on norm comp Sigma (def: nuI)

`nu.e` d.f. parm for regression error variance prior (def: 3)

`ssq` scale parm for regression error var prior (def: var(y_i))

`ncomp` number of components used in normal mixture

`R` number of MCMC draws

`keep` MCMC thinning parm: keep every keepth draw (def: 1)

Value

a list containing

`taudraw`	R/keep x nreg array of error variance draws
`betadraw`	nreg x nvar x R/keep array of individual regression coef draws
`Deltadraw`	R/keep x nz x nvar array of Deltadraws
`nmix`	list of three elements, (probdraw, NULL, compdraw)

Note

More on `probdraw` component of nmix return value list:
this is an R/keep by ncomp array of draws of mixture component probs (pvec)
More on `compdraw` component of nmix return value list:

compdraw[[i]] the ith draw of components for mixtures

compdraw[[i]][[j]] ith draw of the jth normal mixture comp

compdraw[[i]][[j]][[1]] ith draw of jth normal mixture comp mean vector

compdraw[[i]][[j]][[2]] ith draw of jth normal mixture cov parm (rooti)

Note: Z should **not** include an intercept and should be centered for ease of interpretation.

Be careful in assessing prior parameter, Amu. .01 can be too small for some applications.
See Rossi et al, chapter 5 for full discussion.

Author(s)

Peter Rossi, Graduate School of Business, University of Chicago, ⟨Peter.Rossi@ChicagoGsb.edu⟩.

References

For further discussion, see *Bayesian Statistics and Marketing* by Rossi, Allenby and McCulloch, Chapter 3.
http://faculty.chicagogsb.edu/peter.rossi/research/bsm.html

See Also

rhierLinearModel

Examples

```
##
if(nchar(Sys.getenv("LONG_TEST")) != 0) {R=2000} else {R=10}

set.seed(66)
nreg=300; nobs=500; nvar=3; nz=2

Z=matrix(runif(nreg*nz),ncol=nz)
Z=t(t(Z)-apply(Z,2,mean))
Delta=matrix(c(1,-1,2,0,1,0),ncol=nz)
tau0=.1
iota=c(rep(1,nobs))

## create arguments for rmixture

tcomps=NULL
a=matrix(c(1,0,0,0.5773503,1.1547005,0,-0.4082483,0.4082483,1.2247449),ncol=3)
tcomps[[1]]=list(mu=c(0,-1,-2),rooti=a)
tcomps[[2]]=list(mu=c(0,-1,-2)*2,rooti=a)
tcomps[[3]]=list(mu=c(0,-1,-2)*4,rooti=a)
tpvec=c(.4,.2,.4)

regdata=NULL                                              # simulated data with Z
betas=matrix(double(nreg*nvar),ncol=nvar)
tind=double(nreg)

for (reg in 1:nreg) {
tempout=rmixture(1,tpvec,tcomps)
betas[reg,]=Delta%*%Z[reg,]+as.vector(tempout$x)
tind[reg]=tempout$z
X=cbind(iota,matrix(runif(nobs*(nvar-1)),ncol=(nvar-1)))
tau=tau0*runif(1,min=0.5,max=1)
y=X%*%betas[reg,]+sqrt(tau)*rnorm(nobs)
regdata[[reg]]=list(y=y,X=X,beta=betas[reg,],tau=tau)
}

## run rhierLinearMixture

Data1=list(regdata=regdata,Z=Z)
Prior1=list(ncomp=3)
Mcmc1=list(R=R,keep=1)
```

```
out1=rhierLinearMixture(Data=Data1,Prior=Prior1,Mcmc=Mcmc1)

cat("Summary of Delta draws",fill=TRUE)
summary(out1$Deltadraw,tvalues=as.vector(Delta))
cat("Summary of Normal Mixture Distribution",fill=TRUE)
summary(out1$nmix)

if(0){
## plotting examples
plot(out1$betadraw)
plot(out1$nmix)
plot(out1$Deltadraw)
}
```

`rhierLinearModel` *Gibbs Sampler for Hierarchical Linear Model*

Description

`rhierLinearModel` implements a Gibbs Sampler for hierarchical linear models with a normal prior.

Usage

```
rhierLinearModel(Data, Prior, Mcmc)
```

Arguments

`Data`	list(regdata,Z) (Z optional).
`Prior`	list(Deltabar,A,nu.e,ssq,nu,V) (optional).
`Mcmc`	list(R,keep) (R required).

Details

Model: length(regdata) regression equations.
$y_i = X_i beta_i + e_i$. $e_i \sim N(0, tau_i)$. nvar X vars in each equation.

Priors:
$tau_i \sim$ nu.e*$ssq_i/\chi^2_{nu.e}$. tau_i is the variance of e_i.
$beta_i \sim$ N(ZDelta[i,],V_{beta}).
Note: ZDelta is the matrix Z * Delta; [i,] refers to ith row of this product.

$vec(Delta)$ given $V_{beta} \sim N(vec(Deltabar), V_{beta}(x)A^{-1})$.
$V_{beta} \sim IW(nu, V)$.
$Delta, Deltabar$ are nz x nvar. A is nz x nz. V_{beta} is nvar x nvar.

Note: if you don't have any z vars, set Z=iota (nreg x 1).

List arguments contain:

`regdata`	list of lists with X,y matrices for each of length(regdata) regressions
`regdata[[i]]$X`	X matrix for equation i
`regdata[[i]]$y`	y vector for equation i
`Deltabar`	nz x nvar matrix of prior means (def: 0)
`A`	nz x nz matrix for prior precision (def: .01I)
`nu.e`	d.f. parm for regression error variance prior (def: 3)
`ssq`	scale parm for regression error var prior (def: var(y_i))
`nu`	d.f. parm for Vbeta prior (def: nvar+3)
`V`	Scale location matrix for Vbeta prior (def: nu*I)
`R`	number of MCMC draws
`keep`	MCMC thinning parm: keep every keepth draw (def: 1)

Value

a list containing

`betadraw`	nreg x nvar x R/keep array of individual regression coef draws
`taudraw`	R/keep x nreg array of error variance draws
`Deltadraw`	R/keep x nz x nvar array of Deltadraws
`Vbetadraw`	R/keep x nvar*nvar array of Vbeta draws

Author(s)

Peter Rossi, Graduate School of Business, University of Chicago, ⟨Peter.Rossi@ChicagoGsb.edu⟩.

References

For further discussion, see *Bayesian Statistics and Marketing* by Rossi, Allenby and McCulloch, Chapter 3.
http://faculty.chicagogsb.edu/peter.rossi/research/bsm.html

See Also

rhierLinearMixture

Examples

```
##
if(nchar(Sys.getenv("LONG_TEST")) != 0) {R=2000} else {R=10}

nreg=100; nobs=100; nvar=3
Vbeta=matrix(c(1,.5,0,.5,2,.7,0,.7,1),ncol=3)
Z=cbind(c(rep(1,nreg)),3*runif(nreg)); Z[,2]=Z[,2]-mean(Z[,2])
nz=ncol(Z)
Delta=matrix(c(1,-1,2,0,1,0),ncol=2)
Delta=t(Delta) # first row of Delta is means of betas
Beta=matrix(rnorm(nreg*nvar),nrow=nreg)%*%chol(Vbeta)+Z%*%Delta
```

```
tau=.1
iota=c(rep(1,nobs))
regdata=NULL
for (reg in 1:nreg) { X=cbind(iota,matrix(runif(nobs*(nvar-1)),ncol=(nvar-1)))
        y=X%*%Beta[reg,]+sqrt(tau)*rnorm(nobs); regdata[[reg]]=list(y=y,X=X) }

Data1=list(regdata=regdata,Z=Z)
Mcmc1=list(R=R,keep=1)
out=rhierLinearModel(Data=Data1,Mcmc=Mcmc1)

cat("Summary of Delta draws",fill=TRUE)
summary(out$Deltadraw,tvalues=as.vector(Delta))
cat("Summary of Vbeta draws",fill=TRUE)
summary(out$Vbetadraw,tvalues=as.vector(Vbeta[upper.tri(Vbeta,diag=TRUE)]))

if(0){
## plotting examples
plot(out$betadraw)
plot(out$Deltadraw)
}
```

`rhierMnlDP` *MCMC Algorithm for Hierarchical Multinomial Logit with Dirichlet Process Prior Heterogeneity*

Description

`rhierMnlDP` is a MCMC algorithm for a hierarchical multinomial logit with a Dirichlet Process Prior for the distribution of heteorogeneity. A base normal model is used so that the DP can be interpreted as allowing for a mixture of normals with as many components as there are panel units. This is a hybrid Gibbs Sampler with a RW Metropolis step for the MNL coefficients for each panel unit. This procedure can be interpreted as a Bayesian semi-parameteric method in the sense that the DP prior can accomodate heterogeniety of an unknown form.

Usage

```
rhierMnlDP(Data, Prior, Mcmc)
```

Arguments

`Data`	list(p,lgtdata,Z) (Z is optional)
`Prior`	list(deltabar,Ad,Prioralpha,lambda_hyper) (all are optional)
`Mcmc`	list(s,w,R,keep) (R required)

Details

Model:
$y_i \sim MNL(X_i, beta_i)$. i=1,..., length(lgtdata). $theta_i$ is nvar x 1.

$beta_i$= ZDelta[i,] + u_i.
Note: here ZDelta refers to Z%*%D, ZDelta[i,] is ith row of this product.
Delta is an nz x nvar array.

$beta_i \sim N(mu_i, Sigma_i)$.

Priors:
$theta_i = (mu_i, Sigma_i) \sim DP(G_0(lambda), alpha)$
$G_0(lambda)$:
$mu_i | Sigma_i \sim N(0, Sigma_i(x)a^{-1})$
$Sigma_i \sim IW(nu, nu * v * I)$

$lambda(a, nu, v)$:
$a \sim$ uniform[alim[1],alimb[2]]
$nu \sim$ dim(data)-1 + exp(z)
$z \sim$ uniform[dim(data)-1+nulim[1],nulim[2]]
$v \sim$ uniform[vlim[1],vlim[2]]

$alpha \sim (1 - (alpha - alphamin)/(alphamax - alphamin))^power$
alpha= alphamin then expected number of components = Istarmin
alpha= alphamax then expected number of components = Istarmax

Lists contain:

Data:

`p` p is number of choice alternatives
`lgtdata` list of lists with each cross-section unit MNL data
`lgtdata[[i]]$y` n_i vector of multinomial outcomes (1,...,m)
`lgtdata[[i]]$X` n_i by nvar design matrix for ith unit

Prior:

`deltabar` nz*nvar vector of prior means (def: 0)
`Ad` prior prec matrix for vec(D) (def: .01I)

Prioralpha:

`Istarmin` expected number of components at lower bound of support of alpha def(1)
`Istarmax` expected number of components at upper bound of support of alpha (def: min(50,.1*nlgt))
`power` power parameter for alpha prior (def: .8)

lambda_hyper:

`alim` defines support of a distribution,def:c(.01,2)

`nulim` defines support of nu distribution, def:c(.01,3)

`vlim` defines support of v distribution, def:c(.1,4)

Mcmc:

`R` number of mcmc draws

`keep` thinning parm, keep every keepth draw

`maxuniq` storage constraint on the number of unique components

`gridsize` number of discrete points for hyperparameter priors,def: 20

Value

a list containing:

`Deltadraw`	R/keep x nz*nvar matrix of draws of Delta, first row is initial value
`betadraw`	nlgt x nvar x R/keep array of draws of betas
`nmix`	list of 3 components, probdraw, NULL, compdraw
`adraw`	R/keep draws of hyperparm a
`vdraw`	R/keep draws of hyperparm v
`nudraw`	R/keep draws of hyperparm nu
`Istardraw`	R/keep draws of number of unique components
`alphadraw`	R/keep draws of number of DP tightness parameter
`loglike`	R/keep draws of log-likelihood

Note

As is well known, Bayesian density estimation involves computing the predictive distribution of a "new" unit parameter, $theta_{n+1}$ (here "n"=nlgt). This is done by averaging the normal base distribution over draws from the distribution of $theta_{n+1}$ given $theta_1$, ..., $theta_n$,alpha,lambda,Data. To facilitate this, we store those draws from the predictive distribution of $theta_{n+1}$ in a list structure compatible with other `bayesm` routines that implement a finite mixture of normals.

More on nmix list:
contains the draws from the predictive distribution of a "new" observations parameters. These are simply the parameters of one normal distribution. We enforce compatibility with a mixture of k components in order to utilize generic summary plotting functions.

Therefore,`probdraw` is a vector of ones. `zdraw` (indicator draws) is omitted as it is not necessary for density estimation. `compdraw` contains the draws of the $theta_{n+1}$ as a list of list of lists.

More on `compdraw` component of return value list:

compdraw[[i]] ith draw of components for mixtures

compdraw[[i][[1]]] ith draw of the thetanp1 compdraw[[i][[1]][[1]]] ith draw of mean vector

compdraw[[i][[1]][[2]]] ith draw of parm (rooti)

We parameterize the prior on $Sigma_i$ such that $mode(Sigma) = nu/(nu + 2)vI$. The support of nu enforces a non-degenerate IW density; $nulim[1] > 0$.

The default choices of alim,nulim, and vlim determine the location and approximate size of candidate "atoms" or possible normal components. The defaults are sensible given a reasonable scaling of the X variables. You want to insure that alim is set for a wide enough range of values (remember a is a precision parameter) and the v is big enough to propose Sigma matrices wide enough to cover the data range.

A careful analyst should look at the posterior distribution of a, nu, v to make sure that the support is set correctly in alim, nulim, vlim. In other words, if we see the posterior bunched up at one end of these support ranges, we should widen the range and rerun.

If you want to force the procedure to use many small atoms, then set nulim to consider only large values and set vlim to consider only small scaling constants. Set alphamax to a large number. This will create a very "lumpy" density estimate somewhat like the classical Kernel density estimates. Of course, this is not advised if you have a prior belief that densities are relatively smooth.

Note: Z should **not** include an intercept and is centered for ease of interpretation.

Large R values may be required (>20,000).

Author(s)

Peter Rossi, Graduate School of Business, University of Chicago, ⟨Peter.Rossi@ChicagoGsb.edu⟩.

References

For further discussion, see *Bayesian Statistics and Marketing* by Rossi, Allenby and McCulloch, Chapter 5.
http://faculty.chicagogsb.edu/peter.rossi/research/bsm.html

See Also

rhierMnlRwMixture

Examples

```
##
if(nchar(Sys.getenv("LONG_TEST")) != 0) {R=20000} else {R=10}

set.seed(66)
p=3                                 # num of choice alterns
ncoef=3
nlgt=300                            # num of cross sectional units
nz=2
Z=matrix(runif(nz*nlgt),ncol=nz)
Z=t(t(Z)-apply(Z,2,mean))           # demean Z
ncomp=3                                 # no of mixture components
```

```
Delta=matrix(c(1,0,1,0,1,2),ncol=2)
comps=NULL
comps[[1]]=list(mu=c(0,-1,-2),rooti=diag(rep(2,3)))
comps[[2]]=list(mu=c(0,-1,-2)*2,rooti=diag(rep(2,3)))
comps[[3]]=list(mu=c(0,-1,-2)*4,rooti=diag(rep(2,3)))
pvec=c(.4,.2,.4)

simmnlwX= function(n,X,beta) {
  ##  simulate from MNL model conditional on X matrix
  k=length(beta)
  Xbeta=X%*%beta
  j=nrow(Xbeta)/n
  Xbeta=matrix(Xbeta,byrow=TRUE,ncol=j)
  Prob=exp(Xbeta)
  iota=c(rep(1,j))
  denom=Prob%*%iota
  Prob=Prob/as.vector(denom)
  y=vector("double",n)
  ind=1:j
  for (i in 1:n)
      {yvec=rmultinom(1,1,Prob[i,]); y[i]=ind%*%yvec}
  return(list(y=y,X=X,beta=beta,prob=Prob))
}

## simulate data with a mixture of 3 normals
simlgtdata=NULL
ni=rep(50,300)
for (i in 1:nlgt)
{  betai=Delta%*%Z[i,]+as.vector(rmixture(1,pvec,comps)$x)
   Xa=matrix(runif(ni[i]*p,min=-1.5,max=0),ncol=p)
   X=createX(p,na=1,nd=NULL,Xa=Xa,Xd=NULL,base=1)
   outa=simmnlwX(ni[i],X,betai)
   simlgtdata[[i]]=list(y=outa$y,X=X,beta=betai)
}

## plot betas
if(1){
## set if(1) above to produce plots
bmat=matrix(0,nlgt,ncoef)
for(i in 1:nlgt) {bmat[i,]=simlgtdata[[i]]$beta}
par(mfrow=c(ncoef,1))
for(i in 1:ncoef) hist(bmat[,i],breaks=30,col="magenta")
}

##   set Data and Mcmc lists
keep=5
Mcmc1=list(R=R,keep=keep)
Data1=list(p=p,lgtdata=simlgtdata,Z=Z)

out=rhierMnlDP(Data=Data1,Mcmc=Mcmc1)

cat("Summary of Delta draws",fill=TRUE)
summary(out$Deltadraw,tvalues=as.vector(Delta))
```

```
if(0) {
## plotting examples
plot(out$betadraw)
plot(out$nmix)
}
```

---

rhierMnlRwMixture — *MCMC Algorithm for Hierarchical Multinomial Logit with Mixture of Normals Heterogeneity*

Description

`rhierMnlRwMixture` is a MCMC algorithm for a hierarchical multinomial logit with a mixture of normals heterogeneity distribution. This is a hybrid Gibbs Sampler with a RW Metropolis step for the MNL coefficients for each panel unit.

Usage

```
rhierMnlRwMixture(Data, Prior, Mcmc)
```

Arguments

`Data`	list(p,lgtdata,Z) (Z is optional)
`Prior`	list(a,deltabar,Ad,mubar,Amu,nu,V,ncomp) (all but ncomp are optional)
`Mcmc`	list(s,w,R,keep) (R required)

Details

Model:
$y_i \sim MNL(X_i, beta_i)$. i=1,..., length(lgtdata). $theta_i$ is nvar x 1.

$beta_i$= ZDelta[i,] + u_i.
Note: here ZDelta refers to Z%*%D, ZDelta[i,] is ith row of this product.
Delta is an nz x nvar array.

$u_i \sim N(mu_{ind}, Sigma_{ind})$. $ind \sim$ multinomial(pvec).

Priors:
$pvec \sim$ dirichlet (a)
$delta = vec(Delta) \sim N(deltabar, A_d^{-1})$
$mu_j \sim N(mubar, Sigma_j(x)Amu^{-1})$
$Sigma_j \sim$ IW(nu,V)

Lists contain:

- `p` p is number of choice alternatives `lgtdata` list of lists with each cross-section unit MNL data

`lgtdata[[i]]$y` n_i vector of multinomial outcomes (1,. . . ,m)

`lgtdata[[i]]$X` n_i by nvar design matrix for ith unit

`a` vector of length ncomp of Dirichlet prior parms (def: rep(5,ncomp))

`deltabar` nz*nvar vector of prior means (def: 0)

`Ad` prior prec matrix for vec(D) (def: .01I)

`mubar` nvar x 1 prior mean vector for normal comp mean (def: 0)

`Amu` prior precision for normal comp mean (def: .01I)

`nu` d.f. parm for IW prior on norm comp Sigma (def: nvar+3)

`V` pds location parm for IW prior on norm comp Sigma (def: nuI)

`ncomp` number of components used in normal mixture

`s` scaling parm for RW Metropolis (def: 2.93/sqrt(nvar))

`w` fractional likelihood weighting parm (def: .1)

`R` number of MCMC draws

`keep` MCMC thinning parm: keep every keepth draw (def: 1)

Value

a list containing:

`Deltadraw`	R/keep x nz*nvar matrix of draws of Delta, first row is initial value
`betadraw`	nlgt x nvar x R/keep array of draws of betas
`nmix`	list of 3 components, probdraw, NULL, compdraw
`loglike`	log-likelihood for each kept draw (length R/keep)

Note

More on `probdraw` component of nmix list:
R/keep x ncomp matrix of draws of probs of mixture components (pvec)
More on `compdraw` component of return value list:

compdraw[[i]] the ith draw of components for mixtures

compdraw[[i][[j]]] ith draw of the jth normal mixture comp

compdraw[[i][[j]][[1]]] ith draw of jth normal mixture comp mean vector

compdraw[[i][[j]][[2]]] ith draw of jth normal mixture cov parm (rooti)

Note: Z should **not** include an intercept and is centered for ease of interpretation.

Be careful in assessing prior parameter, Amu. .01 is too small for many applications. See Rossi et al, chapter 5 for full discussion.

Note: as of version 2.0-2 of `bayesm`, the fractional weight parameter has been changed to a weight between 0 and 1. w is the fractional weight on the normalized pooled likelihood. This differs from what is in Rossi et al chapter 5, i.e.

$like_i^(1-w)xlike_pooled^((n_i/N)*w)$

Large R values may be required (>20,000).

Author(s)

Peter Rossi, Graduate School of Business, University of Chicago, ⟨Peter.Rossi@ChicagoGsb.edu⟩.

References

For further discussion, see *Bayesian Statistics and Marketing* by Rossi, Allenby and McCulloch, Chapter 5.
http://faculty.chicagogsb.edu/peter.rossi/research/bsm.html

See Also

rmnlIndepMetrop

Examples

```
##
if(nchar(Sys.getenv("LONG_TEST")) != 0) {R=10000} else {R=10}

set.seed(66)
p=3                                # num of choice alterns
ncoef=3
nlgt=300                           # num of cross sectional units
nz=2
Z=matrix(runif(nz*nlgt),ncol=nz)
Z=t(t(Z)-apply(Z,2,mean))          # demean Z
ncomp=3                                # no of mixture components
Delta=matrix(c(1,0,1,0,1,2),ncol=2)
comps=NULL
comps[[1]]=list(mu=c(0,-1,-2),rooti=diag(rep(1,3)))
comps[[2]]=list(mu=c(0,-1,-2)*2,rooti=diag(rep(1,3)))
comps[[3]]=list(mu=c(0,-1,-2)*4,rooti=diag(rep(1,3)))
pvec=c(.4,.2,.4)

simmnlwX= function(n,X,beta) {
  ##  simulate from MNL model conditional on X matrix
  k=length(beta)
  Xbeta=X%*%beta
  j=nrow(Xbeta)/n
  Xbeta=matrix(Xbeta,byrow=TRUE,ncol=j)
  Prob=exp(Xbeta)
  iota=c(rep(1,j))
  denom=Prob%*%iota
  Prob=Prob/as.vector(denom)
  y=vector("double",n)
```

```
  ind=1:j
  for (i in 1:n)
      {yvec=rmultinom(1,1,Prob[i,]); y[i]=ind%*%yvec}
  return(list(y=y,X=X,beta=beta,prob=Prob))
}

## simulate data
simlgtdata=NULL
ni=rep(50,300)
for (i in 1:nlgt)
{  betai=Delta%*%Z[i,]+as.vector(rmixture(1,pvec,comps)$x)
   Xa=matrix(runif(ni[i]*p,min=-1.5,max=0),ncol=p)
   X=createX(p,na=1,nd=NULL,Xa=Xa,Xd=NULL,base=1)
   outa=simmnlwX(ni[i],X,betai)
   simlgtdata[[i]]=list(y=outa$y,X=X,beta=betai)
}

## plot betas
if(0){
## set if(1) above to produce plots
bmat=matrix(0,nlgt,ncoef)
for(i in 1:nlgt) {bmat[i,]=simlgtdata[[i]]$beta}
par(mfrow=c(ncoef,1))
for(i in 1:ncoef) hist(bmat[,i],breaks=30,col="magenta")
}

##   set parms for priors and Z
Prior1=list(ncomp=5)

keep=5
Mcmc1=list(R=R,keep=keep)
Data1=list(p=p,lgtdata=simlgtdata,Z=Z)

out=rhierMnlRwMixture(Data=Data1,Prior=Prior1,Mcmc=Mcmc1)

cat("Summary of Delta draws",fill=TRUE)
summary(out$Deltadraw,tvalues=as.vector(Delta))
cat("Summary of Normal Mixture Distribution",fill=TRUE)
summary(out$nmix)

if(0) {
## plotting examples
plot(out$betadraw)
plot(out$nmix)
}
```

rhierNegbinRw	*MCMC Algorithm for Negative Binomial Regression*

Description

`rhierNegbinRw` implements an MCMC strategy for the hierarchical Negative Binomial (NBD) regression model. Metropolis steps for each unit level set of regression parameters are automatically tuned by optimization. Over-dispersion parameter (alpha) is common across units.

Usage

```
rhierNegbinRw(Data, Prior, Mcmc)
```

Arguments

`Data`	list(regdata,Z)
`Prior`	list(Deltabar,Adelta,nu,V,a,b)
`Mcmc`	list(R,keep,s_beta,s_alpha,c,Vbeta0,Delta0)

Details

Model: $y_i \sim$ NBD(mean=lambda, over-dispersion=alpha).
$lambda = exp(X_i beta_i)$

Prior: $beta_i \sim N(Delta' z_i, Vbeta)$.

$vec(Delta|Vbeta) \sim N(vec(Deltabar), Vbeta(x)Adelta)$.
$Vbeta \sim IW(nu, V)$.
$alpha \sim Gamma(a, b)$.
note: prior mean of $alpha = a/b$, $variance = a/(b^2)$

list arguments contain:

`regdata`	list of lists with data on each of nreg units
`regdata[[i]]$X`	nobs_i x nvar matrix of X variables
`regdata[[i]]$y`	nobs_i x 1 vector of count responses
`Z`	nreg x nz mat of unit chars (def: vector of ones)
`Deltabar`	nz x nvar prior mean matrix (def: 0)
`Adelta`	nz x nz pds prior prec matrix (def: .01I)
`nu`	d.f. parm for IWishart (def: nvar+3)
`V`	location matrix of IWishart prior (def: nuI)
`a`	Gamma prior parm (def: .5)
`b`	Gamma prior parm (def: .1)
`R`	number of MCMC draws
`keep`	MCMC thinning parm: keep every keepth draw (def: 1)
`s_beta`	scaling for beta| alpha RW inc cov (def: 2.93/sqrt(nvar))
`s_alpha`	scaling for alpha | beta RW inc cov (def: 2.93)
`c`	fractional likelihood weighting parm (def:2)
`Vbeta0`	starting value for Vbeta (def: I)
`Delta0`	starting value for Delta (def: 0)

Value

a list containing:

`llike`	R/keep vector of values of log-likelihood
`betadraw`	nreg x nvar x R/keep array of beta draws
`alphadraw`	R/keep vector of alpha draws
`acceptrbeta`	acceptance rate of the beta draws
`acceptralpha`	acceptance rate of the alpha draws

Note

The NBD regression encompasses Poisson regression in the sense that as alpha goes to infinity the NBD distribution tends to the Poisson.
For "small" values of alpha, the dependent variable can be extremely variable so that a large number of observations may be required to obtain precise inferences.

For ease of interpretation, we recommend demeaning Z variables.

Author(s)

Sridhar Narayanam & Peter Rossi, Graduate School of Business, University of Chicago, ⟨Peter.Rossi@ChicagoGsb.edu⟩.

References

For further discussion, see *Bayesian Statistics and Marketing* by Rossi, Allenby and McCulloch, Chapter 5.
http://faculty.chicagogsb.edu/peter.rossi/research/bsm.html

See Also

rnegbinRw

Examples

```
##
if(nchar(Sys.getenv("LONG_TEST")) != 0) {R=2000} else {R=10}
##
set.seed(66)
simnegbin =
function(X, beta, alpha) {
#   Simulate from the Negative Binomial Regression
lambda = exp(X %*% beta)
y=NULL
for (j in 1:length(lambda))
    y = c(y,rnbinom(1,mu = lambda[j],size = alpha))
return(y)
}

nreg = 100        # Number of cross sectional units
T = 50            # Number of observations per unit
```

```
nobs = nreg*T
nvar=2              # Number of X variables
nz=2                # Number of Z variables

# Construct the Z matrix
Z = cbind(rep(1,nreg),rnorm(nreg,mean=1,sd=0.125))

Delta = cbind(c(4,2), c(0.1,-1))
alpha = 5
Vbeta = rbind(c(2,1),c(1,2))

# Construct the regdata (containing X)
simnegbindata = NULL
for (i in 1:nreg) {
    betai = as.vector(Z[i,]%*%Delta) + chol(Vbeta)%*%rnorm(nvar)
    X = cbind(rep(1,T),rnorm(T,mean=2,sd=0.25))
    simnegbindata[[i]] = list(y=simnegbin(X,betai,alpha), X=X,beta=betai)
}

Beta = NULL
for (i in 1:nreg) {Beta=rbind(Beta,matrix(simnegbindata[[i]]$beta,nrow=1))}

Data1 = list(regdata=simnegbindata, Z=Z)
Mcmc1 = list(R=R)

out = rhierNegbinRw(Data=Data1, Mcmc=Mcmc1)

cat("Summary of Delta draws",fill=TRUE)
summary(out$Deltadraw,tvalues=as.vector(Delta))
cat("Summary of Vbeta draws",fill=TRUE)
summary(out$Vbetadraw,tvalues=as.vector(Vbeta[upper.tri(Vbeta,diag=TRUE)]))
cat("Summary of alpha draws",fill=TRUE)
summary(out$alpha,tvalues=alpha)

if(0){
## plotting examples
plot(out$betadraw)
plot(out$alpha,tvalues=alpha)
plot(out$Deltadraw,tvalues=as.vector(Delta))
}
```

`rivDP` *Linear "IV" Model with DP Process Prior for Errors*

Description

`rivDP` is a Gibbs Sampler for a linear structural equation with an arbitrary number of instruments. `rivDP` uses a mixture of normals for the structural and reduced form equation implemented with a Dirichlet Process Prior.

Usage

```
rivDP(Data, Prior, Mcmc)
```

Arguments

`Data`	list(z,w,x,y)
`Prior`	list(md,Ad,mbg,Abg,lambda,Prioralpha) (optional)
`Mcmc`	list(R,keep,SCALE) (R required)

Details

Model:
$x = z'delta + e1$.
$y = beta * x + w'gamma + e2$.
$e1, e2 \sim N(theta_i)$. $theta_i$ represents $mu_i, Sigma_i$

Note: Error terms have non-zero means. DO NOT include intercepts in the z or w matrices. This is different from `rivGibbs` which requires intercepts to be included explicitly.

Priors:
$delta \sim N(md, Ad^{-1})$. $vec(beta, gamma) \sim N(mbg, Abg^{-1})$

$theta_i \sim \ G$

$G \sim DP(alpha, G_0)$

G_0 is the natural conjugate prior for $(mu, Sigma)$:
$Sigma \sim IW(nu, vI)$ and $mu|Sigma \sim N(0, 1/amuSigma)$
These parameters are collected together in the list `lambda`. It is highly recommended that you use the default settings for these hyper-parameters.

$alpha \sim (1 - (alpha - alpha_{min})/(alpha_{max} - alphamin))^{power}$
where $alpha_{min}$ and $alpha_{max}$ are set using the arguments in the reference below. It is highly recommended that you use the default values for the hyperparameters of the prior on alpha

List arguments contain:

- `z` matrix of obs on instruments
- `y` vector of obs on lhs var in structural equation
- `x` ”endogenous” var in structural eqn
- `w` matrix of obs on ”exogenous” vars in the structural eqn
- `md` prior mean of delta (def: 0)
- `Ad` pds prior prec for prior on delta (def: .01I)
- `mbg` prior mean vector for prior on beta,gamma (def: 0)
- `Abg` pds prior prec for prior on beta,gamma (def: .01I)
- `lambda` list of hyperparameters for theta prior- use default settings

`Prioralpha` list of hyperparameters for theta prior- use default settings

`R` number of MCMC draws

`keep` MCMC thinning parm: keep every keepth draw (def: 1)

`SCALE` scale data, def: TRUE

`gridsize` gridsize parm for alpha draws (def: 20)

output includes object `nmix` of class "bayesm.nmix" which contains draws of predictive distribution of errors (a Bayesian analogue of a density estimate for the error terms).
nmix:

`probdraw` not used

`zdraw` not used

`compdraw` list R/keep of draws from bivariate predictive for the errors

note: in compdraw list, there is only one component per draw

Value

a list containing:

`deltadraw`	R/keep x dim(delta) array of delta draws
`betadraw`	R/keep x 1 vector of beta draws
`gammadraw`	R/keep x dim(gamma) array of gamma draws
`Istardraw`	R/keep x 1 array of drawsi of the number of unique normal components
`alphadraw`	R/keep x 1 array of draws of Dirichlet Process tightness parameter
`nmix`	R/keep x list of draws for predictive distribution of errors

Author(s)

Peter Rossi, Graduate School of Business, University of Chicago, ⟨Peter.Rossi@ChicagoGsb.edu⟩.

References

For further discussion, see "A Semi-Parametric Bayesian Approach to the Instrumental Variable Problem," by Conley, Hansen, McCulloch and Rossi, Journal of Econometrics (2008).

See Also

`rivGibbs`

Examples

```
##
if(nchar(Sys.getenv("LONG_TEST")) != 0) {R=2000} else {R=10}

##
## simulate scaled log-normal errors and run
##
set.seed(66)
k=10
delta=1.5
Sigma=matrix(c(1,.6,.6,1),ncol=2)
N=1000
tbeta=4
set.seed(66)
scalefactor=.6
root=chol(scalefactor*Sigma)
mu=c(1,1)
##
## compute interquartile ranges
##
ninterq=qnorm(.75)-qnorm(.25)
error=matrix(rnorm(100000*2),ncol=2)
error=t(t(error)+mu)
Err=t(t(exp(error))-exp(mu+.5*scalefactor*diag(Sigma)))
lnNinterq=quantile(Err[,1],prob=.75)-quantile(Err[,1],prob=.25)
##
## simulate data
##
error=matrix(rnorm(N*2),ncol=2)%*%root
error=t(t(error)+mu)
Err=t(t(exp(error))-exp(mu+.5*scalefactor*diag(Sigma)))
#
# scale appropriately
Err[,1]=Err[,1]*ninterq/lnNinterq
Err[,2]=Err[,2]*ninterq/lnNinterq
z=matrix(runif(k*N),ncol=k)
x=z%*%(delta*c(rep(1,k)))+Err[,1]
y=x*tbeta+Err[,2]

# set intial values for MCMC
Data = list(); Mcmc=list()
Data$z = z;  Data$x=x; Data$y=y

# start MCMC and keep results
Mcmc$maxuniq=100
Mcmc$R=R
end=Mcmc$R
begin=100

out=rivDP(Data=Data,Mcmc=Mcmc)

cat("Summary of Beta draws",fill=TRUE)
```

```
summary(out$betadraw,tvalues=tbeta)

if(0){
## plotting examples
plot(out$betadraw,tvalues=tbeta)
plot(out$nmix)  ## plot "fitted" density of the errors
##

}
```

---

rivGibbs *Gibbs Sampler for Linear "IV" Model*

Description

`rivGibbs` is a Gibbs Sampler for a linear structural equation with an arbitrary number of instruments.

Usage

```
rivGibbs(Data, Prior, Mcmc)
```

Arguments

`Data`	list(z,w,x,y)
`Prior`	list(md,Ad,mbg,Abg,nu,V) (optional)
`Mcmc`	list(R,keep) (R required)

Details

Model:
$x = z'delta + e1$.
$y = beta * x + w'gamma + e2$.
$e1, e2 \sim N(0, Sigma)$.

Note: if intercepts are desired in either equation, include vector of ones in z or w

Priors:
$delta \sim N(md, Ad^{-1})$. $vec(beta, gamma) \sim N(mbg, Abg^{-1})$
$Sigma \sim$ IW(nu,V)

List arguments contain:

- `z` matrix of obs on instruments
- `y` vector of obs on lhs var in structural equation
- `x` "endogenous" var in structural eqn
- `w` matrix of obs on "exogenous" vars in the structural eqn
- `md` prior mean of delta (def: 0)
- `Ad` pds prior prec for prior on delta (def: .01I)

`mbg`	prior mean vector for prior on beta,gamma (def: 0)
`Abg`	pds prior prec for prior on beta,gamma (def: .01I)
`nu`	d.f. parm for IW prior on Sigma (def: 5)
`V`	pds location matrix for IW prior on Sigma (def: nuI)
`R`	number of MCMC draws
`keep`	MCMC thinning parm: keep every keepth draw (def: 1)

Value

a list containing:

`deltadraw`	R/keep x dim(delta) array of delta draws
`betadraw`	R/keep x 1 vector of beta draws
`gammadraw`	R/keep x dim(gamma) array of gamma draws
`Sigmadraw`	R/keep x 4 array of Sigma draws

Author(s)

Rob McCulloch and Peter Rossi, Graduate School of Business, University of Chicago, ⟨Peter.Rossi@ChicagoGsb.edu⟩.

References

For further discussion, see *Bayesian Statistics and Marketing* by Rossi, Allenby and McCulloch, Chapter 5.
http://faculty.chicagogsb.edu/peter.rossi/research/bsm.html

Examples

```
##
if(nchar(Sys.getenv("LONG_TEST")) != 0) {R=2000} else {R=10}

set.seed(66)
simIV = function(delta,beta,Sigma,n,z,w,gamma) {
eps = matrix(rnorm(2*n),ncol=2) %*% chol(Sigma)
x = z %*% delta + eps[,1]; y = beta*x +  eps[,2] + w%*%gamma
list(x=as.vector(x),y=as.vector(y)) }
n = 200 ; p=1 # number of instruments
z = cbind(rep(1,n),matrix(runif(n*p),ncol=p))
w = matrix(1,n,1)
rho=.8
Sigma = matrix(c(1,rho,rho,1),ncol=2)
delta = c(1,4); beta = .5; gamma = c(1)
simiv = simIV(delta,beta,Sigma,n,z,w,gamma)

Mcmc1=list();  Data1 = list()
Data1$z = z; Data1$w=w; Data1$x=simiv$x; Data1$y=simiv$y
Mcmc1$R = R
Mcmc1$keep=1
```

```
out=rivGibbs(Data=Data1,Mcmc=Mcmc1)

cat("Summary of Beta draws",fill=TRUE)
summary(out$betadraw,tvalues=beta)
cat("Summary of Sigma draws",fill=TRUE)
summary(out$Sigmadraw,tvalues=as.vector(Sigma[upper.tri(Sigma,diag=TRUE)]))

if(0){
## plotting examples
plot(out$betadraw)
}
```

`rmixGibbs` *Gibbs Sampler for Normal Mixtures w/o Error Checking*

Description

`rmixGibbs` makes one draw using the Gibbs Sampler for a mixture of multivariate normals.

Usage

```
rmixGibbs(y, Bbar, A, nu, V, a, p, z, comps)
```

Arguments

`y`	data array - rows are obs
`Bbar`	prior mean for mean vector of each norm comp
`A`	prior precision parameter
`nu`	prior d.f. parm
`V`	prior location matrix for covariance priro
`a`	Dirichlet prior parms
`p`	prior prob of each mixture component
`z`	component identities for each observation – ”indicators”
`comps`	list of components for the normal mixture

Details

`rmixGibbs` is not designed to be called directly. Instead, use `rnmixGibbs` wrapper function.

Value

a list containing:

`p`	draw mixture probabilities
`z`	draw of indicators of each component
`comps`	new draw of normal component parameters

Warning

This routine is a utility routine that does **not** check the input arguments for proper dimensions and type.

Author(s)

Rob McCulloch and Peter Rossi, Graduate School of Business, University of Chicago, ⟨Peter.Rossi@ChicagoGsb.edu⟩.

References

For further discussion, see *Bayesian Statistics and Marketing* by Allenby, McCulloch, and Rossi, Chapter 5.
`http://faculty.chicagogsb.edu/peter.rossi/research/bsm.html`

See Also

`rnmixGibbs`

`rmixture` *Draw from Mixture of Normals*

Description

`rmixture` simulates iid draws from a Multivariate Mixture of Normals

Usage

```
rmixture(n, pvec, comps)
```

Arguments

`n`	number of observations
`pvec`	ncomp x 1 vector of prior probabilities for each mixture component
`comps`	list of mixture component parameters

Details

comps is a list of length, ncomp = length(pvec). comps[[j]][[1]] is mean vector for the jth component. comps[[j]][[2]] is the inverse of the cholesky root of Sigma for that component

Value

A list containing . . .

`x`	An n x length(comps[[1]][[1]]) array of iid draws
`z`	A n x 1 vector of indicators of which component each draw is taken from

Warning

This routine is a utility routine that does **not** check the input arguments for proper dimensions and type.

Author(s)

Peter Rossi, Graduate School of Business, University of Chicago, ⟨Peter.Rossi@ChicagoGsb.edu⟩.

See Also

`rnmixGibbs`

`rmnlIndepMetrop` *MCMC Algorithm for Multinomial Logit Model*

Description

`rmnIndepMetrop` implements Independence Metropolis for the MNL.

Usage

```
rmnlIndepMetrop(Data, Prior, Mcmc)
```

Arguments

`Data`	list(p,y,X)
`Prior`	list(A,betabar) optional
`Mcmc`	list(R,keep,nu)

Details

Model: y ∼ MNL(X,beta). $Pr(y = j) = exp(x_j' beta)/\sum_k e^{x_k' beta}$.

Prior: $beta \sim N(betabar, A^{-1})$

list arguments contain:

- `p` number of alternatives
- `y` nobs vector of multinomial outcomes (1,. . . , p)
- `X` nobs*p x nvar matrix
- `A` nvar x nvar pds prior prec matrix (def: .01I)
- `betabar` nvar x 1 prior mean (def: 0)
- `R` number of MCMC draws
- `keep` MCMC thinning parm: keep every keepth draw (def: 1)
- `nu` degrees of freedom parameter for independence t density (def: 6)

Value

a list containing:

`betadraw`	R/keep x nvar array of beta draws
`loglike`	R/keep vector of loglike values for each draw
`acceptr`	acceptance rate of Metropolis draws

Author(s)

Peter Rossi, Graduate School of Business, University of Chicago, ⟨Peter.Rossi@ChicagoGsb.edu⟩.

References

For further discussion, see *Bayesian Statistics and Marketing* by Rossi, Allenby and McCulloch, Chapter 5.
http://faculty.chicagogsb.edu/peter.rossi/research/bsm.html

See Also

rhierMnlRwMixture

Examples

```
##

if(nchar(Sys.getenv("LONG_TEST")) != 0) {R=2000} else {R=10}

set.seed(66)
n=200; p=3; beta=c(1,-1,1.5,.5)

simmnl= function(p,n,beta) {
  #   note: create X array with 2 alt.spec vars
  k=length(beta)
  X1=matrix(runif(n*p,min=-1,max=1),ncol=p)
  X2=matrix(runif(n*p,min=-1,max=1),ncol=p)
  X=createX(p,na=2,nd=NULL,Xd=NULL,Xa=cbind(X1,X2),base=1)
  Xbeta=X%*%beta # now do probs
  p=nrow(Xbeta)/n
  Xbeta=matrix(Xbeta,byrow=TRUE,ncol=p)
  Prob=exp(Xbeta)
  iota=c(rep(1,p))
  denom=Prob%*%iota
  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%*%yvec }
   return(list(y=y,X=X,beta=beta,prob=Prob))
}
```

```
simout=simmnl(p,n,beta)

Data1=list(y=simout$y,X=simout$X,p=p); Mcmc1=list(R=R,keep=1)
out=rmnlIndepMetrop(Data=Data1,Mcmc=Mcmc1)

cat("Summary of beta draws",fill=TRUE)
summary(out$betadraw,tvalues=beta)

if(0){
## plotting examples
plot(out$betadraw)
}
```

`rmnpGibbs` *Gibbs Sampler for Multinomial Probit*

Description

`rmnpGibbs` implements the McCulloch/Rossi Gibbs Sampler for the multinomial probit model.

Usage

```
rmnpGibbs(Data, Prior, Mcmc)
```

Arguments

`Data`	list(p, y, X)
`Prior`	list(betabar,A,nu,V) (optional)
`Mcmc`	list(beta0,sigma0,R,keep) (R required)

Details

model:
$w_i = X_i\beta + e$. $e \sim N(0, Sigma)$. note: w_i, e are (p-1) x 1.
$y_i = j$, if $w_{ij} > max(0, w_{i,-j})$ j=1,...,p-1. $w_{i,-j}$ means elements of w_i other than the jth.
$y_i = p$, if all $w_i < 0$.

priors:
$beta \sim N(betabar, A^{-1})$
$Sigma \sim$ IW(nu,V)

to make up X matrix use createX with `DIFF=TRUE`.

List arguments contain

- `p` number of choices or possible multinomial outcomes

`y`	n x 1 vector of multinomial outcomes
`X`	n*(p-1) x k Design Matrix
`betabar`	k x 1 prior mean (def: 0)
`A`	k x k prior precision matrix (def: .01I)
`nu`	d.f. parm for IWishart prior (def: (p-1) + 3)
`V`	pds location parm for IWishart prior (def: nu*I)
`beta0`	initial value for beta
`sigma0`	initial value for sigma
`R`	number of MCMC draws
`keep`	thinning parameter - keep every keepth draw (def: 1)

Value

a list containing:

`betadraw`	R/keep x k array of betadraws
`sigmadraw`	R/keep x (p-1)*(p-1) array of sigma draws – each row is in vector form

Note

beta is not identified. beta/sqrt($sigma_{11}$) and Sigma/$sigma_{11}$ are. See Allenby et al or example below for details.

Author(s)

Peter Rossi, Graduate School of Business, University of Chicago, ⟨Peter.Rossi@ChicagoGsb.edu⟩.

References

For further discussion, see *Bayesian Statistics and Marketing* by Rossi, Allenby and McCulloch, Chapter 4.
http://faculty.chicagogsb.edu/peter.rossi/research/bsm.html

See Also

rmvpGibbs

Examples

```
##
if(nchar(Sys.getenv("LONG_TEST")) != 0) {R=2000} else {R=10}

set.seed(66)
p=3
n=500
beta=c(-1,1,1,2)
Sigma=matrix(c(1,.5,.5,1),ncol=2)
k=length(beta)
```

```
X1=matrix(runif(n*p,min=0,max=2),ncol=p); X2=matrix(runif(n*p,min=0,max=2),ncol=p)
X=createX(p,na=2,nd=NULL,Xa=cbind(X1,X2),Xd=NULL,DIFF=TRUE,base=p)

simmnp= function(X,p,n,beta,sigma) {
  indmax=function(x) {which(max(x)==x)}
  Xbeta=X%*%beta
  w=as.vector(crossprod(chol(sigma),matrix(rnorm((p-1)*n),ncol=n)))+ Xbeta
  w=matrix(w,ncol=(p-1),byrow=TRUE)
  maxw=apply(w,1,max)
  y=apply(w,1,indmax)
  y=ifelse(maxw < 0,p,y)
  return(list(y=y,X=X,beta=beta,sigma=sigma))
}

simout=simmnp(X,p,500,beta,Sigma)

Data1=list(p=p,y=simout$y,X=simout$X)
Mcmc1=list(R=R,keep=1)

out=rmnpGibbs(Data=Data1,Mcmc=Mcmc1)

cat(" Summary of Betadraws ",fill=TRUE)
betatilde=out$betadraw/sqrt(out$sigmadraw[,1])
attributes(betatilde)$class="bayesm.mat"
summary(betatilde,tvalues=beta)

cat(" Summary of Sigmadraws ",fill=TRUE)
sigmadraw=out$sigmadraw/out$sigmadraw[,1]
attributes(sigmadraw)$class="bayesm.var"
summary(sigmadraw,tvalues=as.vector(Sigma[upper.tri(Sigma,diag=TRUE)]))

if(0){
## plotting examples
plot(betatilde,tvalues=beta)
}
```

`rmultireg` *Draw from the Posterior of a Multivariate Regression*

Description

`rmultireg` draws from the posterior of a Multivariate Regression model with a natural conjugate prior.

Usage

```
rmultireg(Y, X, Bbar, A, nu, V)
```

Arguments

`Y`	n x m matrix of observations on m dep vars
`X`	n x k matrix of observations on indep vars (supply intercept)
`Bbar`	k x m matrix of prior mean of regression coefficients
`A`	k x k Prior precision matrix
`nu`	d.f. parameter for Sigma
`V`	m x m pdf location parameter for prior on Sigma

Details

Model: $Y = XB + U$. $cov(u_i) = Sigma$. B is k x m matrix of coefficients. $Sigma$ is m x m covariance.

Priors: $beta$ given $Sigma \sim N(betabar, Sigma(x)A^{-1})$. $betabar = vec(Bbar)$; $beta = vec(B)$
$Sigma \sim$ IW(nu,V).

Value

A list of the components of a draw from the posterior

`B`	draw of regression coefficient matrix
`Sigma`	draw of Sigma

Warning

This routine is a utility routine that does **not** check the input arguments for proper dimensions and type.

Author(s)

Peter Rossi, Graduate School of Business, University of Chicago, ⟨Peter.Rossi@ChicagoGsb.edu⟩.

References

For further discussion, see *Bayesian Statistics and Marketing* by Rossi, Allenby and McCulloch, Chapter 2.
http://faculty.chicagogsb.edu/peter.rossi/research/bsm.html

Examples

```
##
if(nchar(Sys.getenv("LONG_TEST")) != 0) {R=2000} else {R=10}

set.seed(66)
n=200
m=2
X=cbind(rep(1,n),runif(n))
k=ncol(X)
B=matrix(c(1,2,-1,3),ncol=m)
```

```
Sigma=matrix(c(1,.5,.5,1),ncol=m); RSigma=chol(Sigma)
Y=X%*%B+matrix(rnorm(m*n),ncol=m)%*%RSigma

betabar=rep(0,k*m);Bbar=matrix(betabar,ncol=m)
A=diag(rep(.01,k))
nu=3; V=nu*diag(m)

betadraw=matrix(double(R*k*m),ncol=k*m)
Sigmadraw=matrix(double(R*m*m),ncol=m*m)
for (rep in 1:R)
   {out=rmultireg(Y,X,Bbar,A,nu,V);betadraw[rep,]=out$B
    Sigmadraw[rep,]=out$Sigma}

cat(" Betadraws ",fill=TRUE)
mat=apply(betadraw,2,quantile,probs=c(.01,.05,.5,.95,.99))
mat=rbind(as.vector(B),mat); rownames(mat)[1]="beta"
print(mat)
cat(" Sigma draws",fill=TRUE)
mat=apply(Sigmadraw,2,quantile,probs=c(.01,.05,.5,.95,.99))
mat=rbind(as.vector(Sigma),mat); rownames(mat)[1]="Sigma"
print(mat)
```

`rmvpGibbs` *Gibbs Sampler for Multivariate Probit*

Description

`rmvpGibbs` implements the Edwards/Allenby Gibbs Sampler for the multivariate probit model.

Usage

```
rmvpGibbs(Data, Prior, Mcmc)
```

Arguments

`Data`	list(p,y,X)
`Prior`	list(betabar,A,nu,V) (optional)
`Mcmc`	list(beta0,sigma0,R,keep) (R required)

Details

model:
$w_i = X_i beta + e.$ $e \sim$ N(0,Sigma). note: w_i is p x 1.
$y_{ij} = 1$, if $w_{ij} > 0$, else $y_i = 0$. j=1,...,p.

priors:
$beta \sim N(betabar, A^{-1})$

$Sigma \sim$ IW(nu,V)

to make up X matrix use `createX`

List arguments contain

- `p` dimension of multivariate probit
- `X` n*p x k Design Matrix
- `y` n*p x 1 vector of 0,1 outcomes
- `betabar` k x 1 prior mean (def: 0)
- `A` k x k prior precision matrix (def: .01I)
- `nu` d.f. parm for IWishart prior (def: (p-1) + 3)
- `V` pds location parm for IWishart prior (def: nu*I)
- `beta0` initial value for beta
- `sigma0` initial value for sigma
- `R` number of MCMC draws
- `keep` thinning parameter - keep every keepth draw (def: 1)

Value

a list containing:

`betadraw`	R/keep x k array of betadraws
`sigmadraw`	R/keep x p*p array of sigma draws – each row is in vector form

Note

beta and Sigma are not identifed. Correlation matrix and the betas divided by the appropriate standard deviation are. See Allenby et al for details or example below.

Author(s)

Peter Rossi, Graduate School of Business, University of Chicago, ⟨Peter.Rossi@ChicagoGsb.edu⟩.

References

For further discussion, see *Bayesian Statistics and Marketing* by Rossi, Allenby and McCulloch, Chapter 4.
http://faculty.chicagogsb.edu/peter.rossi/research/bsm.html

See Also

`rmnpGibbs`

Examples

```
##
if(nchar(Sys.getenv("LONG_TEST")) != 0) {R=2000} else {R=10}

set.seed(66)
p=3
n=500
beta=c(-2,0,2)
Sigma=matrix(c(1,.5,.5,.5,1,.5,.5,.5,1),ncol=3)
k=length(beta)
I2=diag(rep(1,p)); xadd=rbind(I2)
for(i in 2:n) { xadd=rbind(xadd,I2)}; X=xadd

simmvp= function(X,p,n,beta,sigma) {
  w=as.vector(crossprod(chol(sigma),matrix(rnorm(p*n),ncol=n)))+ X%*%beta
  y=ifelse(w<0,0,1)
  return(list(y=y,X=X,beta=beta,sigma=sigma))
}

simout=simmvp(X,p,500,beta,Sigma)

Data1=list(p=p,y=simout$y,X=simout$X)
Mcmc1=list(R=R,keep=1)
out=rmvpGibbs(Data=Data1,Mcmc=Mcmc1)

ind=seq(from=0,by=p,length=k)
inda=1:3
ind=ind+inda
cat(" Betadraws ",fill=TRUE)
betatilde=out$betadraw/sqrt(out$sigmadraw[,ind])
attributes(betatilde)$class="bayesm.mat"
summary(betatilde,tvalues=beta/sqrt(diag(Sigma)))

rdraw=matrix(double((R)*p*p),ncol=p*p)
rdraw=t(apply(out$sigmadraw,1,nmat))
attributes(rdraw)$class="bayesm.var"
tvalue=nmat(as.vector(Sigma))
dim(tvalue)=c(p,p)
tvalue=as.vector(tvalue[upper.tri(tvalue,diag=TRUE)])
cat(" Draws of Correlation Matrix ",fill=TRUE)
summary(rdraw,tvalues=tvalue)

if(0){
plot(betatilde,tvalues=beta/sqrt(diag(Sigma)))
}
```

`rmvst` *Draw from Multivariate Student-t*

Description

`rmvst` draws from a Multivariate student-t distribution.

Usage

```
rmvst(nu, mu, root)
```

Arguments

`nu`	d.f. parameter
`mu`	mean vector
`root`	Upper Tri Cholesky Root of Sigma

Value

length(mu) draw vector

Warning

This routine is a utility routine that does **not** check the input arguments for proper dimensions and type.

Author(s)

Peter Rossi, Graduate School of Business, University of Chicago, ⟨Peter.Rossi@ChicagoGsb.edu⟩.

References

For further discussion, see *Bayesian Statistics and Marketing* by Rossi, Allenby and McCulloch.
http://faculty.chicagogsb.edu/peter.rossi/research/bsm.html

See Also

`lndMvst`

Examples

```
##
set.seed(66)
rmvst(nu=5,mu=c(rep(0,2)),root=chol(matrix(c(2,1,1,2),ncol=2)))
```

rnegbinRw	*MCMC Algorithm for Negative Binomial Regression*

Description

`rnegbinRw` implements a Random Walk Metropolis Algorithm for the Negative Binomial (NBD) regression model. beta | alpha and alpha | beta are drawn with two different random walks.

Usage

```
rnegbinRw(Data, Prior, Mcmc)
```

Arguments

`Data`	list(y,X)
`Prior`	list(betabar,A,a,b)
`Mcmc`	list(R,keep,s_beta,s_alpha,beta0

Details

Model: $y \sim NBD(mean = lambda, over - dispersion = alpha)$.
$lambda = exp(x'beta)$

Prior: $beta \sim N(betabar, A^{-1})$
$alpha \sim Gamma(a, b)$.
note: prior mean of $alpha = a/b$, $variance = a/(b^2)$

list arguments contain:

`y`	nobs vector of counts (0,1,2,...)
`X`	nobs x nvar matrix
`betabar`	nvar x 1 prior mean (def: 0)
`A`	nvar x nvar pds prior prec matrix (def: .01I)
`a`	Gamma prior parm (def: .5)
`b`	Gamma prior parm (def: .1)
`R`	number of MCMC draws
`keep`	MCMC thinning parm: keep every keepth draw (def: 1)
`s_beta`	scaling for beta| alpha RW inc cov matrix (def: 2.93/sqrt(nvar)
`s_alpha`	scaling for alpha | beta RW inc cov matrix (def: 2.93)

Value

a list containing:

`betadraw`	R/keep x nvar array of beta draws
`alphadraw`	R/keep vector of alpha draws
`llike`	R/keep vector of log-likelihood values evaluated at each draw
`acceptrbeta`	acceptance rate of the beta draws
`acceptralpha`	acceptance rate of the alpha draws

Note

The NBD regression encompasses Poisson regression in the sense that as alpha goes to infinity the NBD distribution tends toward the Poisson.
For "small" values of alpha, the dependent variable can be extremely variable so that a large number of observations may be required to obtain precise inferences.

Author(s)

Sridhar Narayanam & Peter Rossi, Graduate School of Business, University of Chicago, ⟨Peter.Rossi@ChicagoGsb.edu⟩.

References

For further discussion, see *Bayesian Statistics and Marketing* by Rossi, Allenby, McCulloch.
http://faculty.chicagogsb.edu/peter.rossi/research/bsm.html

See Also

rhierNegbinRw

Examples

```
##
if(nchar(Sys.getenv("LONG_TEST")) != 0)  {R=1000} else {R=10}

set.seed(66)
simnegbin =
function(X, beta, alpha) {
#   Simulate from the Negative Binomial Regression
lambda = exp(X %*% beta)
y=NULL
for (j in 1:length(lambda))
    y = c(y,rnbinom(1,mu = lambda[j],size = alpha))
return(y)
}

nobs = 500
nvar=2            # Number of X variables
alpha = 5
Vbeta = diag(nvar)*0.01
```

```
# Construct the regdata (containing X)
simnegbindata = NULL
beta = c(0.6,0.2)
X = cbind(rep(1,nobs),rnorm(nobs,mean=2,sd=0.5))
simnegbindata = list(y=simnegbin(X,beta,alpha), X=X, beta=beta)

Data1 = simnegbindata
Mcmc1 = list(R=R)

out = rnegbinRw(Data=Data1,Mcmc=Mcmc1)

cat("Summary of alpha/beta draw",fill=TRUE)
summary(out$alphadraw,tvalues=alpha)
summary(out$betadraw,tvalues=beta)

if(0){
## plotting examples
plot(out$betadraw)
}
```

`rnmixGibbs` *Gibbs Sampler for Normal Mixtures*

Description

`rnmixGibbs` implements a Gibbs Sampler for normal mixtures.

Usage

```
rnmixGibbs(Data, Prior, Mcmc)
```

Arguments

`Data`	list(y)
`Prior`	list(Mubar,A,nu,V,a,ncomp) (only ncomp required)
`Mcmc`	list(R,keep) (R required)

Details

Model:
$y_i \sim N(mu_{ind_i}, Sigma_{ind_i})$.
ind $\sim$ iid multinomial(p). p is a ncomp x 1 vector of probs.

Priors:
$mu_j \sim N(mubar, Sigma_j(x)A^{-1})$. $mubar = vec(Mubar)$.
$Sigma_j \sim$ IW(nu,V).
note: this is the natural conjugate prior – a special case of multivariate regression.
$p \sim$ Dirchlet(a).

Output of the components is in the form of a list of lists.
compsdraw[[i]] is ith draw – list of ncomp lists.
compsdraw[[i]][[j]] is list of parms for jth normal component.
jcomp=compsdraw[[i]][j]]. Then jth comp $\sim N(jcomp[[1]], Sigma)$, $Sigma =$ t(R)%*%R, $R^{-1} =$ jcomp[[2]].

List arguments contain:

y	n x k array of data (rows are obs)
Mubar	1 x k array with prior mean of normal comp means (def: 0)
A	1 x 1 precision parameter for prior on mean of normal comp (def: .01)
nu	d.f. parameter for prior on Sigma (normal comp cov matrix) (def: k+3)
V	k x k location matrix of IW prior on Sigma (def: nuI)
a	ncomp x 1 vector of Dirichlet prior parms (def: rep(5,ncomp))
ncomp	number of normal components to be included
R	number of MCMC draws
keep	MCMC thinning parm: keep every keepth draw (def: 1)

Value

`nmix`

a list containing: probdraw,zdraw,compdraw

Note

more details on contents of nmix:

probdraw R/keep x ncomp array of mixture prob draws

zdraw R/keep x nobs array of indicators of mixture comp identity for each obs

compdraw R/keep lists of lists of comp parm draws

In this model, the component normal parameters are not-identified due to label-switching. However, the fitted mixture of normals density is identified as it is invariant to label-switching. See Allenby et al, chapter 5 for details. Use `eMixMargDen` or `momMix` to compute posterior expectation or distribution of various identified parameters.

Author(s)

Peter Rossi, Graduate School of Business, University of Chicago, ⟨Peter.Rossi@ChicagoGsb.edu⟩.

References

For further discussion, see *Bayesian Statistics and Marketing* by Rossi, Allenby and McCulloch, Chapter 3.
`http://faculty.chicagogsb.edu/peter.rossi/research/bsm.html`

See Also

rmixture, rmixGibbs ,eMixMargDen, momMix, mixDen, mixDenBi

Examples

```
##
if(nchar(Sys.getenv("LONG_TEST")) != 0) {R=2000} else {R=10}

set.seed(66)
dim=5;  k=3   # dimension of simulated data and number of "true" components
sigma = matrix(rep(0.5,dim^2),nrow=dim);diag(sigma)=1
sigfac = c(1,1,1);mufac=c(1,2,3); compsmv=list()
for(i in 1:k) compsmv[[i]] = list(mu=mufac[i]*1:dim,sigma=sigfac[i]*sigma)
comps = list() # change to "rooti" scale
for(i in 1:k) comps[[i]] = list(mu=compsmv[[i]][[1]],rooti=solve(chol(compsmv[[i]][[2]])))
pvec=(1:k)/sum(1:k)

nobs=500
dm = rmixture(nobs,pvec,comps)

Data1=list(y=dm$x)
ncomp=9
Prior1=list(ncomp=ncomp)
Mcmc1=list(R=R,keep=1)
out=rnmixGibbs(Data=Data1,Prior=Prior1,Mcmc=Mcmc1)

cat("Summary of Normal Mixture Distribution",fill=TRUE)
summary(out)
tmom=momMix(matrix(pvec,nrow=1),list(comps))
mat=rbind(tmom$mu,tmom$sd)
cat(" True Mean/Std Dev",fill=TRUE)
print(mat)

if(0){
##
## plotting examples
##
plot(out,Data=dm$x)
}
```

---

rordprobitGibbs   *Gibbs Sampler for Ordered Probit*

Description

rordprobitGibbs implements a Gibbs Sampler for the ordered probit model.

Usage

```
rordprobitGibbs(Data, Prior, Mcmc)
```

Arguments

`Data`	list(X, y, k)
`Prior`	list(betabar, A, dstarbar, Ad)
`Mcmc`	list(R, keep, s, change, draw)

Details

Model: $z = X\beta + e$. $e \sim N(0, I)$. y=1,..,k. cutoff=c(c [1] ,..c [k+1]).
y=k, if c [k] <= z < c [k+1] .

Prior: $\beta \sim N(betabar, A^{-1})$. $dstar \sim N(dstarbar, Ad^{-1})$.

List arguments contain

`X` n x nvar Design Matrix

`y` n x 1 vector of observations, (1,...,k)

`k` the largest possible value of y

`betabar` nvar x 1 prior mean (def: 0)

`A` nvar x nvar prior precision matrix (def: .01I)

`dstarbar` ndstar x 1 prior mean, ndstar=k-2 (def: 0)

`Ad` ndstar x ndstar prior precision matrix (def:I)

`s` scaling parm for RW Metropolis (def: 2.93/sqrt(nvar))

`R` number of MCMC draws

`keep` thinning parameter - keep every keepth draw (def: 1)

Value

`betadraw`	R/keep x k matrix of betadraws
`cutdraw`	R/keep x (k-1) matrix of cutdraws
`dstardraw`	R/keep x (k-2) matrix of dstardraws
`accept`	a value of acceptance rate in RW Metropolis

Note

set c[1]=-100. c[k+1]=100. c[2] is set to 0 for identification.

The relationship between cut-offs and dstar is
c[3] = exp(dstar[1]), c[4]=c[3]+exp(dstar[2]),..., c[k] = c[k-1] + exp(datsr[k-2])

Be careful in assessing prior parameter, Ad. .1 is too small for many applications.

Author(s)

Peter Rossi, Graduate School of Business, University of Chicago, ⟨Peter.Rossi@ChicagoGsb.edu⟩.

References

Bayesian Statistics and Marketing by Rossi, Allenby and McCulloch
http://faculty.chicagogsb.edu/peter.rossi/research/bsm.html

See Also

rbprobitGibbs

Examples

```
##
## rordprobitGibbs example
##
if(nchar(Sys.getenv("LONG_TEST")) != 0) {R=2000} else {R=10}

## simulate data for ordered probit model

   simordprobit=function(X, betas, cutoff){
    z = X%*%betas + rnorm(nobs)
    y = cut(z, br = cutoff, right=TRUE, include.lowest = TRUE, labels = FALSE)
    return(list(y = y, X = X, k=(length(cutoff)-1), betas= betas, cutoff=cutoff ))
   }

   set.seed(66)
   nobs=300
   X=cbind(rep(1,nobs),runif(nobs, min=0, max=5),runif(nobs,min=0, max=5))
   k=5
   betas=c(0.5, 1, -0.5)
   cutoff=c(-100, 0, 1.0, 1.8, 3.2,  100)
   simout=simordprobit(X, betas, cutoff)
   Data=list(X=simout$X,y=simout$y, k=k)

## set Mcmc for ordered probit model

   Mcmc=list(R=R)
   out=rordprobitGibbs(Data=Data,Mcmc=Mcmc)

   cat(" ", fill=TRUE)
   cat("acceptance rate= ",accept=out$accept,fill=TRUE)

## outputs of betadraw and cut-off draws

   cat(" Summary of betadraws",fill=TRUE)
   summary(out$betadraw,tvalues=betas)
   cat(" Summary of cut-off draws",fill=TRUE)
   summary(out$cutdraw,tvalues=cutoff[2:k])

if(0){
## plotting examples
plot(out$cutdraw)
}
```

rscaleUsage	*MCMC Algorithm for Multivariate Ordinal Data with Scale Usage Heterogeneity.*

Description

`rscaleUsage` implements an MCMC algorithm for multivariate ordinal data with scale usage heterogeneity.

Usage

```
rscaleUsage(Data,Prior, Mcmc)
```

Arguments

`Data`	list(k,x)
`Prior`	list(nu,V,mubar,Am,gsigma,gl11,gl22,gl12,Lambdanu,LambdaV,ge) (optional)
`Mcmc`	list(R,keep,ndghk,printevery,e,y,mu,Sigma,sigma,tau,Lambda) (optional)

Details

Model: n=nrow(x) individuals respond to m=ncol(x) questions. all questions are on a scale 1, . . . , k. for respondent i and question j,
$x_{ij} = d$, if $c_{d-1} \leq y_{ij} \leq c_d$.
d=1,. . . ,k. $c_d = a + bd + ed^2$.

$y_i = mu + tau_i * iota + sigma_i * z_i$. $z_i \sim N(0, Sigma)$.

Priors:
$(tau_i, ln(sigma_i)) \sim N(phi, Lamda)$. $phi = (0, lambda_{22})$.
mu $\sim N(mubar, Am^-1)$.
Sigma $\sim$ IW(nu,V).
Lambda $\sim$ IW(Lambdanu,LambdaV).
e $\sim$ unif on a grid.

Value

a list containing:

`Sigmadraw`	R/keep x m*m array of Sigma draws
`mudraw`	R/keep x m array of mu draws
`taudraw`	R/keep x n array of tau draws
`sigmadraw`	R/keep x n array of sigma draws
`Lambdadraw`	R/keep x 4 array of Lamda draws
`edraw`	R/keep x 1 array of e draws

Warning

tau_i, $sigma_i$ are identified from the scale usage patterns in the m questions asked per respondent (# cols of x). Do not attempt to use this on data sets with only a small number of total questions!

Note

It is **highly** recommended that the user choose the default settings. This means not specifying the argument `Prior` and setting `R` in Mcmc and `Data` only. If you wish to change prior settings and/or the grids used, please read the case study in Allenby et al carefully.

Author(s)

Rob McCulloch and Peter Rossi, Graduate School of Business, University of Chicago, ⟨Peter.Rossi@ChicagoGsb.edu⟩.

References

For further discussion, see *Bayesian Statistics and Marketing* by Rossi, Allenby, and McCulloch, Case Study on Scale Usage Heterogeneity.
http://faculty.chicagogsb.edu/peter.rossi/research/bsm.html

Examples

```
##
if(nchar(Sys.getenv("LONG_TEST")) != 0) {R=1000} else {R=1}
{
data(customerSat)
surveydat = list(k=10,x=as.matrix(customerSat))

Mcmc1 = list(R=R)
set.seed(66)
out=rscaleUsage(Data=surveydat,Mcmc=Mcmc1)

summary(out$mudraw)

}
```

`rsurGibbs` *Gibbs Sampler for Seemingly Unrelated Regressions (SUR)*

Description

`rsurGibbs` implements a Gibbs Sampler to draw from the posterior of the Seemingly Unrelated Regression (SUR) Model of Zellner

Usage

```
rsurGibbs(Data, Prior, Mcmc)
```

Arguments

`Data`	list(regdata)
`Prior`	list(betabar,A, nu, V)
`Mcmc`	list(R,keep)

Details

Model: $y_i = X_i beta_i + e_i$. i=1,...,m. m regressions.
(e(1,k), ..., e(m,k)) $\sim N(0, Sigma)$. k=1, ..., nobs.

We can also write as the stacked model:
$y = Xbeta + e$ where y is a nobs*m long vector and k=length(beta)=sum(length(betai)).

Note: we must have the same number of observations in each equation but we can have different numbers of X variables

Priors: $beta \sim N(betabar, A^{-1})$. $Sigma \sim IW(nu, V)$.

List arguments contain

- `regdata` list of lists, regdata[[i]]=list(y=yi,X=Xi)
- `betabar` k x 1 prior mean (def: 0)
- `A` k x k prior precision matrix (def: .01I)
- `nu` d.f. parm for Inverted Wishart prior (def: m+3)
- `V` scale parm for Inverted Wishart prior (def: nu*I)
- `R` number of MCMC draws
- `keep` thinning parameter - keep every keepth draw

Value

list of MCMC draws

`betadraw`	R x k array of betadraws
`Sigmadraw`	R x (m*m) array of Sigma draws

Author(s)

Peter Rossi, Graduate School of Business, University of Chicago, ⟨Peter.Rossi@ChicagoGsb.edu⟩.

References

For further discussion, see *Bayesian Statistics and Marketing* by Rossi, Allenby and McCulloch, Chapter 3.
http://faculty.chicagogsb.edu/peter.rossi/research/bsm.html

See Also

`rmultireg`

Examples

```
if(nchar(Sys.getenv("LONG_TEST")) != 0) {R=1000} else {R=10}
##
## simulate data from SUR
set.seed(66)
beta1=c(1,2)
beta2=c(1,-1,-2)
nobs=100
nreg=2
iota=c(rep(1,nobs))
X1=cbind(iota,runif(nobs))
X2=cbind(iota,runif(nobs),runif(nobs))
Sigma=matrix(c(.5,.2,.2,.5),ncol=2)
U=chol(Sigma)
E=matrix(rnorm(2*nobs),ncol=2)
y1=X1%*%beta1+E[,1]
y2=X2%*%beta2+E[,2]
##
## run Gibbs Sampler
regdata=NULL
regdata[[1]]=list(y=y1,X=X1)
regdata[[2]]=list(y=y2,X=X2)

Mcmc1=list(R=R)

out=rsurGibbs(Data=list(regdata=regdata),Mcmc=Mcmc1)

cat("Summary of beta draws",fill=TRUE)
summary(out$betadraw,tvalues=c(beta1,beta2))
cat("Summary of Sigmadraws",fill=TRUE)
summary(out$Sigmadraw,tvalues=as.vector(Sigma[upper.tri(Sigma,diag=TRUE)]))

if(0){
plot(out$betadraw,tvalues=c(beta1,beta2))
}
```

`rtrun` *Draw from Truncated Univariate Normal*

Description

`rtrun` draws from a truncated univariate normal distribution

Usage

```
rtrun(mu, sigma, a, b)
```

Arguments

`mu`	mean
`sigma`	sd
`a`	lower bound
`b`	upper bound

Details

Note that due to the vectorization of the rnorm,qnorm commands in R, all arguments can be vectors of equal length. This makes the inverse CDF method the most efficient to use in R.

Value

draw (possibly a vector)

Warning

This routine is a utility routine that does **not** check the input arguments for proper dimensions and type.

Author(s)

Peter Rossi, Graduate School of Business, University of Chicago, ⟨Peter.Rossi@ChicagoGsb.edu⟩.

References

For further discussion, see *Bayesian Statistics and Marketing* by Rossi, Allenby and McCulloch, Chapter 2.
http://faculty.chicagogsb.edu/peter.rossi/research/bsm.html

Examples

```
##
set.seed(66)
rtrun(mu=c(rep(0,10)),sigma=c(rep(1,10)),a=c(rep(0,10)),b=c(rep(2,10)))
```

`runireg` *IID Sampler for Univariate Regression*

Description

`runireg` implements an iid sampler to draw from the posterior of a univariate regression with a conjugate prior.

Usage

```
runireg(Data, Prior, Mcmc)
```

Arguments

`Data`	list(y,X)
`Prior`	list(betabar,A, nu, ssq)
`Mcmc`	list(R,keep)

Details

Model: $y = Xbeta + e$. $e \sim N(0, sigmasq)$.

Priors: $beta \sim N(betabar, sigmasq * A^{-1})$. $sigmasq \sim (nu * ssq)/chisq_{nu}$. List arguments contain

- `X` n x k Design Matrix
- `y` n x 1 vector of observations
- `betabar` k x 1 prior mean (def: 0)
- `A` k x k prior precision matrix (def: .01I)
- `nu` d.f. parm for Inverted Chi-square prior (def: 3)
- `ssq` scale parm for Inverted Chi-square prior (def: var(y))
- `R` number of draws
- `keep` thinning parameter - keep every keepth draw

Value

list of iid draws

`betadraw`	R x k array of betadraws
`sigmasqdraw`	R vector of sigma-sq draws

Author(s)

Peter Rossi, Graduate School of Business, University of Chicago, ⟨Peter.Rossi@ChicagoGsb.edu⟩.

References

For further discussion, see *Bayesian Statistics and Marketing* by Rossi, Allenby and McCulloch, Chapter 2.
`http://faculty.chicagogsb.edu/peter.rossi/research/bsm.html`

See Also

`runiregGibbs`

Examples

```
if(nchar(Sys.getenv("LONG_TEST")) != 0) {R=2000} else {R=10}
set.seed(66)
n=200
X=cbind(rep(1,n),runif(n)); beta=c(1,2); sigsq=.25
y=X%*%beta+rnorm(n,sd=sqrt(sigsq))

out=runireg(Data=list(y=y,X=X),Mcmc=list(R=R))

cat("Summary of beta/sigma-sq draws",fill=TRUE)
summary(out$betadraw,tvalues=beta)
summary(out$sigmasqdraw,tvalues=sigsq)

if(0){
## plotting examples
plot(out$betadraw)
}
```

`runiregGibbs` *Gibbs Sampler for Univariate Regression*

Description

`runiregGibbs` implements a Gibbs Sampler to draw from posterior of a univariate regression with a conditionally conjugate prior.

Usage

```
runiregGibbs(Data, Prior, Mcmc)
```

Arguments

`Data`	list(y,X)
`Prior`	list(betabar,A, nu, ssq)
`Mcmc`	list(sigmasq,R,keep)

Details

Model: $y = Xbeta + e$. $e \sim N(0, sigmasq)$.

Priors: $beta \sim N(betabar, A^{-1})$. $sigmasq \sim (nu * ssq)/chisq_{nu}$. List arguments contain

`X` n x k Design Matrix

`y` n x 1 vector of observations

`betabar` k x 1 prior mean (def: 0)

`A` k x k prior precision matrix (def: .01I)

`nu`	d.f. parm for Inverted Chi-square prior (def: 3)
`ssq`	scale parm for Inverted Chi-square prior (def:var(y))
`R`	number of MCMC draws
`keep`	thinning parameter - keep every keepth draw

Value

list of MCMC draws

`betadraw`	R x k array of betadraws
`sigmasqdraw`	R vector of sigma-sq draws

Author(s)

Peter Rossi, Graduate School of Business, University of Chicago, ⟨Peter.Rossi@ChicagoGsb.edu⟩.

References

For further discussion, see *Bayesian Statistics and Marketing* by Rossi, Allenby and McCulloch, Chapter 3.
http://faculty.chicagogsb.edu/peter.rossi/research/bsm.html

See Also

runireg

Examples

```
if(nchar(Sys.getenv("LONG_TEST")) != 0) {R=1000} else {R=10}
set.seed(66)
n=100
X=cbind(rep(1,n),runif(n)); beta=c(1,2); sigsq=.25
y=X%*%beta+rnorm(n,sd=sqrt(sigsq))

Data1=list(y=y,X=X); Mcmc1=list(R=R)

out=runiregGibbs(Data=Data1,Mcmc=Mcmc1)

cat("Summary of beta and Sigma draws",fill=TRUE)
summary(out$betadraw,tvalues=beta)
summary(out$sigmasqdraw,tvalues=sigsq)

if(0){
## plotting examples
plot(out$betadraw)
}
```

`rwishart`	*Draw from Wishart and Inverted Wishart Distribution*

Description

`rwishart` draws from the Wishart and Inverted Wishart distributions.

Usage

```
rwishart(nu, V)
```

Arguments

`nu`	d.f. parameter
`V`	pds location matrix

Details

In the parameterization used here, $W \sim W(nu, V)$, $E[W] = nuV$.

If you want to use an Inverted Wishart prior, you *must invert the location matrix* before calling `rwishart`, e.g.
$Sigma \sim$ IW(nu,V); $Sigma^{-1} \sim W(nu, V^{-1})$.

Value

`W`	Wishart draw
`IW`	Inverted Wishart draw
`C`	Upper tri root of W
`CI`	inv(C), W^{-1} = CICI'

Warning

This routine is a utility routine that does **not** check the input arguments for proper dimensions and type.

Author(s)

Peter Rossi, Graduate School of Business, University of Chicago, ⟨Peter.Rossi@ChicagoGsb.edu⟩.

References

For further discussion, see *Bayesian Statistics and Marketing* by Rossi, Allenby and McCulloch, Chapter 2.
http://faculty.chicagogsb.edu/peter.rossi/research/bsm.html

Examples

```
##
set.seed(66)
rwishart(5,diag(3))$IW
```

`Scotch` *Survey Data on Brands of Scotch Consumed*

Description

from Simmons Survey. Brands used in last year for those respondents who report consuming scotch.

Usage

```
data(Scotch)
```

Format

A data frame with 2218 observations on the following 21 variables. All variables are coded 1 if consumed in last year, 0 if not.

`Chivas.Regal` a numeric vector

`Dewar.s.White.Label` a numeric vector

`Johnnie.Walker.Black.Label` a numeric vector

`J...B` a numeric vector

`Johnnie.Walker.Red.Label` a numeric vector

`Other.Brands` a numeric vector

`Glenlivet` a numeric vector

`Cutty.Sark` a numeric vector

`Glenfiddich` a numeric vector

`Pinch..Haig.` a numeric vector

`Clan.MacGregor` a numeric vector

`Ballantine` a numeric vector

`Macallan` a numeric vector

`Passport` a numeric vector

`Black...White` a numeric vector

`Scoresby.Rare` a numeric vector

`Grants` a numeric vector

`Ushers` a numeric vector

`White.Horse` a numeric vector

`Knockando` a numeric vector

`the.Singleton` a numeric vector

Source

Edwards, Y. and G. Allenby (2003), "Multivariate Analysis of Multiple Response Data," *JMR* 40, 321-334.

References

Chapter 4, *Bayesian Statistics and Marketing* by Rossi et al.
http://faculty.chicagogsb.edu/peter.rossi/research/bsm.html

Examples

```
data(Scotch)
cat(" Frequencies of Brands", fill=TRUE)
mat=apply(as.matrix(Scotch),2,mean)
print(mat)
##
## use Scotch data to run Multivariate Probit Model
##
if(0){
##

y=as.matrix(Scotch)
p=ncol(y); n=nrow(y)
dimnames(y)=NULL
y=as.vector(t(y))
y=as.integer(y)
I_p=diag(p)
X=rep(I_p,n)
X=matrix(X,nrow=p)
X=t(X)

R=2000
Data=list(p=p,X=X,y=y)
Mcmc=list(R=R)
set.seed(66)
out=rmvpGibbs(Data=Data,Mcmc=Mcmc)

ind=(0:(p-1))*p + (1:p)
cat(" Betadraws ",fill=TRUE)
mat=apply(out$betadraw/sqrt(out$sigmadraw[,ind]),2,quantile,probs=c(.01,.05,.5,.95,.99))
attributes(mat)$class="bayesm.mat"
summary(mat)
rdraw=matrix(double((R)*p*p),ncol=p*p)
rdraw=t(apply(out$sigmadraw,1,nmat))
attributes(rdraw)$class="bayesm.var"
cat(" Draws of Correlation Matrix ",fill=TRUE)
summary(rdraw)

}
```

`simnhlogit` *Simulate from Non-homothetic Logit Model*

Description

`simnhlogit` simulates from the non-homothetic logit model

Usage

```
simnhlogit(theta, lnprices, Xexpend)
```

Arguments

`theta`	coefficient vector
`lnprices`	n x p array of prices
`Xexpend`	n x k array of values of expenditure variables

Details

For detail on parameterization, see `llnhlogit`.

Value

a list containing:

`y`	n x 1 vector of multinomial outcomes (1, . . . , p)
`Xexpend`	expenditure variables
`lnprices`	price array
`theta`	coefficients
`prob`	n x p array of choice probabilities

Warning

This routine is a utility routine that does **not** check the input arguments for proper dimensions and type.

Author(s)

Peter Rossi, Graduate School of Business, University of Chicago, ⟨Peter.Rossi@ChicagoGsb.edu⟩.

References

For further discussion, see *Bayesian Statistics and Marketing* by Rossi, Allenby and McCulloch, Chapter 4.
http://faculty.chicagogsb.edu/peter.rossi/research/bsm.html

See Also

`llnhlogit`

summary.bayesm.mat *Summarize Mcmc Parameter Draws*

Description

`summary.bayesm.mat` is an S3 method to summarize marginal distributions given an array of draws

Usage

```
## S3 method for class 'bayesm.mat':
summary(object, names, burnin = trunc(0.1 * nrow(X)), tvalues, QUANTILES = TRUE, TRAILER = TRUE
```

Arguments

`object`	`object` (hereafter `X`) is an array of draws, usually an object of class "bayesm.mat"
`names`	optional character vector of names for the columns of `X`
`burnin`	number of draws to burn-in, def: .1*nrow(X)
`tvalues`	optional vector of "true" values for use in simulation examples
`QUANTILES`	logical for should quantiles be displayed, def: TRUE
`TRAILER`	logical for should a trailer be displayed, def: TRUE
`...`	optional arguments for generic function

Details

Typically, `summary.bayesm.nmix` will be invoked by a call to the generic summary function as in `summary(object)` where object is of class bayesm.mat. Mean, Std Dev, Numerical Standard error (of estimate of posterior mean), relative numerical efficiency (see `numEff`) and effective sample size are displayed. If QUANTILES=TRUE, quantiles of marginal distirbutions in the columns of X are displayed.

`summary.bayesm.mat` is also exported for direct use as a standard function, as in `summary.bayesm.mat(matrix)`.

Author(s)

Peter Rossi, Graduate School of Business, University of Chicago, ⟨Peter.Rossi@ChicagoGsb.edu⟩.

See Also

`summary.bayesm.var`, `summary.bayesm.nmix`

Examples

```
##
## not run
#   out=rmnpGibbs(Data,Prior,Mcmc)
#   summary(out$betadraw)
#
```

`summary.bayesm.nmix` *Summarize Draws of Normal Mixture Components*

Description

`summary.bayesm.nmix` is an S3 method to display summaries of the distribution implied by draws of Normal Mixture Components. Posterior means and Variance-Covariance matrices are displayed.

Note: 1st and 2nd moments may not be very interpretable for mixtures of normals. This summary function can take a minute or so. The current implementation is not efficient.

Usage

```
## S3 method for class 'bayesm.nmix':
summary(object, names,burnin = trunc(0.1 * nrow(probdraw)), ...)
```

Arguments

`object`	an object of class "bayesm.nmix" – a list of lists of draws
`names`	optional character vector of names fo reach dimension of the density
`burnin`	number of draws to burn-in, def: .1*nrow(probdraw)
`...`	parms to send to summary

Details

an object of class "bayesm.nmix" is a list of three components:

probdraw a matrix of R/keep rows by dim of normal mix of mixture prob draws

second comp not used

compdraw list of list of lists with draws of mixture comp parms

Author(s)

Peter Rossi, Graduate School of Business, University of Chicago, ⟨Peter.Rossi@ChicagoGsb.edu⟩.

See Also

`summary.bayesm.mat`, `summary.bayesm.var`

Examples

```
##
## not run
#   out=rnmix(Data,Prior,Mcmc)
#   summary(out)
#
```

`summary.bayesm.var`   *Summarize Draws of Var-Cov Matrices*

Description

`summary.bayesm.var` is an S3 method to summarize marginal distributions given an array of draws

Usage

```
## S3 method for class 'bayesm.var':
summary(object, names, burnin = trunc(0.1 * nrow(Vard)), tvalues, QUANTILES = FALSE , ...)
```

Arguments

`object`	`object` (herafter, `Vard`) is an array of draws of a covariance matrix
`names`	optional character vector of names for the columns of `Vard`
`burnin`	number of draws to burn-in, def: .1*nrow(Vard)
`tvalues`	optional vector of "true" values for use in simulation examples
`QUANTILES`	logical for should quantiles be displayed, def: TRUE
`...`	optional arguments for generic function

Details

Typically, `summary.bayesm.var` will be invoked by a call to the generic summary function as in summary(object) where object is of class bayesm.var. Mean, Std Dev, Numerical Standard error (of estimate of posterior mean), relative numerical efficiency (see `numEff`) and effective sample size are displayed. If QUANTILES=TRUE, quantiles of marginal distirbutions in the columns of Vard are displayed.

`Vard` is an array of draws of a covariance matrix stored as vectors. Each row is a different draw.
The posterior mean of the vector of standard deviations and the correlation matrix are also displayed

Author(s)

Peter Rossi, Graduate School of Business, University of Chicago, ⟨Peter.Rossi@ChicagoGsb.edu⟩.

See Also

summary.bayesm.mat, summary.bayesm.nmix

Examples

```
##
## not run
#  out=rmnpGibbs(Data,Prior,Mcmc)
#  summary(out$sigmadraw)
#
```

---

tuna *Data on Canned Tuna Sales*

Description

Volume of canned tuna sales as well as a measure of display activity, log price and log wholesale price. Weekly data aggregated to the chain level. This data is extracted from the Dominick's Finer Foods database maintained by the University of Chicago http://http://research.chicagogsb.edu/marketing/databases/dominicks/dataset.aspx. Brands are seven of the top 10 UPCs in the canned tuna product category.

Usage

```
data(tuna)
```

Format

A data frame with 338 observations on the following 30 variables.

WEEK a numeric vector

MOVE1 unit sales of Star Kist 6 oz.

MOVE2 unit sales of Chicken of the Sea 6 oz.

MOVE3 unit sales of Bumble Bee Solid 6.12 oz.

MOVE4 unit sales of Bumble Bee Chunk 6.12 oz.

MOVE5 unit sales of Geisha 6 oz.

MOVE6 unit sales of Bumble Bee Large Cans.

MOVE7 unit sales of HH Chunk Lite 6.5 oz.

NSALE1 a measure of display activity of Star Kist 6 oz.

NSALE2 a measure of display activity of Chicken of the Sea 6 oz.

NSALE3 a measure of display activity of Bumble Bee Solid 6.12 oz.

NSALE4 a measure of display activity of Bumble Bee Chunk 6.12 oz.

NSALE5 a measure of display activity of Geisha 6 oz.

NSALE6 a measure of display activity of Bumble Bee Large Cans.

NSALE7 a measure of display activity of HH Chunk Lite 6.5 oz.

LPRICE1 log of price of Star Kist 6 oz.

LPRICE2 log of price of Chicken of the Sea 6 oz.

LPRICE3 log of price of Bumble Bee Solid 6.12 oz.

LPRICE4 log of price of Bumble Bee Chunk 6.12 oz.

LPRICE5 log of price of Geisha 6 oz.

LPRICE6 log of price of Bumble Bee Large Cans.

LPRICE7 log of price of HH Chunk Lite 6.5 oz.

LWHPRIC1 log of wholesale price of Star Kist 6 oz.

LWHPRIC2 log of wholesale price of Chicken of the Sea 6 oz.

LWHPRIC3 log of wholesale price of Bumble Bee Solid 6.12 oz.

LWHPRIC4 log of wholesale price of Bumble Bee Chunk 6.12 oz.

LWHPRIC5 log of wholesale price of Geisha 6 oz.

LWHPRIC6 log of wholesale price of Bumble Bee Large Cans.

LWHPRIC7 log of wholesale price of HH Chunk Lite 6.5 oz.

FULLCUST total customers visits

Source

Chevalier, A. Judith, Anil K. Kashyap and Peter E. Rossi (2003), "Why Don't Prices Rise During Periods of Peak Demand? Evidence from Scanner Data," *The American Economic Review* , 93(1), 15-37.

References

Chapter 7, *Bayesian Statistics and Marketing* by Rossi et al.
http://faculty.chicagogsb.edu/peter.rossi/research/bsm.html

Examples

```
data(tuna)
cat(" Quantiles of sales",fill=TRUE)
mat=apply(as.matrix(tuna[,2:5]),2,quantile)
print(mat)

##
## example of processing for use with rivGibbs
##
if(0)
{
  data(tuna)
  t = dim(tuna)[1]
  customers = tuna[,30]
  sales = tuna[,2:8]
  lnprice = tuna[,16:22]
```

```
  lnwhPrice= tuna[,23:29]
  share=sales/mean(customers)
  shareout=as.vector(1-rowSums(share))
  lnprob=log(share/shareout)

# create w matrix

  I1=as.matrix(rep(1, t))
  I0=as.matrix(rep(0, t))
  intercept=rep(I1, 4)
  brand1=rbind(I1, I0, I0, I0)
  brand2=rbind(I0, I1, I0, I0)
  brand3=rbind(I0, I0, I1, I0)
  w=cbind(intercept, brand1, brand2, brand3)

## choose brand 1 to 4

  y=as.vector(as.matrix(lnprob[,1:4]))
  X=as.vector(as.matrix(lnprice[,1:4]))
  lnwhPrice=as.vector(as.matrix (lnwhPrice[1:4]))
  z=cbind(w, lnwhPrice)

  Data=list(z=z, w=w, x=X, y=y)
  Mcmc=list(R=R, keep=1)
  set.seed(66)
  out=rivGibbs(Data=Data,Mcmc=Mcmc)

  cat(" betadraws ",fill=TRUE)
  summary(out$betadraw)

if(0){
## plotting examples
plot(out$betadraw)
}
}
```