Package ‘bayesm’

July 21, 2017

Version 3.1-0.1
Type Package
Title Bayesian Inference for Marketing/Micro-Econometrics
Depends R (>= 3.2.0)
Date 2017-06-23
Author Peter Rossi <perossichi@gmail.com>
Maintainer Peter Rossi <perossichi@gmail.com>
License GPL (>= 2)
Imports Rcpp (>= 0.12.0), utils, stats, graphics, grDevices
LinkingTo Rcpp, RcppArmadillo
Suggests knitr, rmarkdown
VignetteBuilder knitr
URL http://www.perossi.org/home/bsm-1

Description Covers many important models used in marketing and micro-econometrics applications.
The package includes:
Bayes Regression (univariate or multivariate dep var),
Bayes Seemingly Unrelated Regression (SUR),
Binary and Ordinal Probit,
Multinomial Logit (MNL) and Multinomial Probit (MNP),
Multivariate Probit,
Negative Binomial (Poisson) Regression,
Multivariate Mixtures of Normals (including clustering),
Dirichlet Process Prior Density Estimation with normal base,
Hierarchical Linear Models with normal prior and covariates,
Hierarchical Linear Models with a mixture of normals prior and covariates,
Hierarchical Multinomial Logits with a mixture of normals prior and covariates,
Hierarchical Multinomial Logits with a Dirichlet Process prior and covariates,
Hierarchical Negative Binomial Regression Models,
Bayesian analysis of choice-based conjoint data,
Bayesian treatment of linear instrumental variables models,
Analysis of Multivariate Ordinal survey data with scale usage heterogeneity (as in Rossi et al, JASA (01)), Bayesian Analysis of Aggregate Random Coefficient Logit Models as in BLP (see Jiang, Manchanda, Rossi 2009)


**RoxygenNote** 6.0.1

**NeedsCompilation** yes

**Repository** CRAN

**Date/Publication** 2017-07-21 15:05:52 UTC

**R topics documented:**

bank ................................................................. 3
breg ................................................................. 6
camera ............................................................ 7
cgetC ............................................................... 8
cheese ............................................................. 9
clusterMix ......................................................... 11
condMom .......................................................... 13
createX ............................................................ 14
customerSat ....................................................... 15
detailing .......................................................... 16
eMixMargDen ...................................................... 18
ghkvec ............................................................. 19
limnl ................................................................. 21
llmnp ............................................................... 22
llnllogit ........................................................... 24
lndChisq .......................................................... 25
lndIWishart ......................................................... 26
lndMvn ............................................................. 27
lndMvst ............................................................ 28
logMargDenNR ..................................................... 29
margarine ......................................................... 30
mixDen ............................................................. 33
mixDenBi ........................................................... 34
mnIHess ............................................................ 35
mnpProb ............................................................ 36
momMix ............................................................. 37
nmat ................................................................. 38
numEff .............................................................. 39
orangeJuice ....................................................... 40
plot.bayesm.hcoef ............................................... 43
plot.bayesm.mat ................................................ 44
<table>
<thead>
<tr>
<th>Function Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>plot.bayesm.nmix</td>
<td>45</td>
</tr>
<tr>
<td>rbayesBLP</td>
<td>46</td>
</tr>
<tr>
<td>rbINormGibbs</td>
<td>51</td>
</tr>
<tr>
<td>rbprobitGibbs</td>
<td>52</td>
</tr>
<tr>
<td>rdirichlet</td>
<td>53</td>
</tr>
<tr>
<td>rDPGibbs</td>
<td>54</td>
</tr>
<tr>
<td>rhierBinLogit</td>
<td>58</td>
</tr>
<tr>
<td>rhierLinearMixture</td>
<td>61</td>
</tr>
<tr>
<td>rhierLinearModel</td>
<td>64</td>
</tr>
<tr>
<td>rhierMnlDP</td>
<td>66</td>
</tr>
<tr>
<td>rhierMnlRwMixture</td>
<td>71</td>
</tr>
<tr>
<td>rhierNegbinRw</td>
<td>75</td>
</tr>
<tr>
<td>rivDP</td>
<td>78</td>
</tr>
<tr>
<td>rivGibbs</td>
<td>81</td>
</tr>
<tr>
<td>rmixGibbs</td>
<td>83</td>
</tr>
<tr>
<td>rmixture</td>
<td>85</td>
</tr>
<tr>
<td>rmnlIndepMetrop</td>
<td>86</td>
</tr>
<tr>
<td>rmmGibbs</td>
<td>88</td>
</tr>
<tr>
<td>rmultireg</td>
<td>90</td>
</tr>
<tr>
<td>rmvpgibbs</td>
<td>92</td>
</tr>
<tr>
<td>rmvst</td>
<td>94</td>
</tr>
<tr>
<td>rmegbinRw</td>
<td>95</td>
</tr>
<tr>
<td>rmixGibbs</td>
<td>98</td>
</tr>
<tr>
<td>rordprobitGibbs</td>
<td>100</td>
</tr>
<tr>
<td>rscaleUsage</td>
<td>103</td>
</tr>
<tr>
<td>rsurGibbs</td>
<td>105</td>
</tr>
<tr>
<td>rtrun</td>
<td>107</td>
</tr>
<tr>
<td>runireg</td>
<td>108</td>
</tr>
<tr>
<td>runiregGibbs</td>
<td>110</td>
</tr>
<tr>
<td>rwishart</td>
<td>111</td>
</tr>
<tr>
<td>Scotch</td>
<td>113</td>
</tr>
<tr>
<td>simnhlogit</td>
<td>114</td>
</tr>
<tr>
<td>summary.bayesm.mat</td>
<td>115</td>
</tr>
<tr>
<td>summary.bayesm.nmix</td>
<td>116</td>
</tr>
<tr>
<td>summary.bayesm.var</td>
<td>117</td>
</tr>
<tr>
<td>tuna</td>
<td>118</td>
</tr>
</tbody>
</table>

**Index**

bank  Bank Card Conjoint Data
Description

A panel dataset from a conjoint experiment in which two partial profiles of credit cards were presented to 946 respondents from a regional bank wanting to offer credit cards to customers outside of its normal operating region. Each respondent was presented with between 13 and 17 paired comparisons. The bank and attribute levels are disguised to protect the proprietary interests of the cooperating firm.

Usage

data(bank)

Format

The bank object is a list containing two data frames. The first, choiceAtt, provides choice attributes for the partial credit card profiles. The second, demo, provides demographic information on the respondents.

Details

In the choiceAtt data frame:

- `...$id` respondent id
- `...$choice` profile chosen
- `...$Med_FInt` medium fixed interest rate
- `...$Low_FInt` low fixed interest rate
- `...$Med_VInt` variable interest rate
- `...$Rewrd_2` reward level 2
- `...$Rewrd_3` reward level 3
- `...$Rewrd_4` reward level 4
- `...$Med_Fee` medium annual fee level
- `...$Low_Fee` low annual fee level
- `...$Bank_B` bank offering the credit card
- `...$Out_State` location of the bank offering the credit card
- `...$Med_Rebate` medium rebate level
- `...$High_Rebate` high rebate level
- `...$High_CredLine` high credit line level
- `...$Long_Grace` grace period

The profiles are coded as the difference in attribute levels. Thus, that 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.

In the demo data frame:

- `...$id` respondent id
- `...$age` respondent age in years
- `...$income` respondent income category
- `...$gender` female=1
bank

Source


References


http://www.perossi.org/home/bsm-1

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)

Data = list(lgtdata=lgtdata, Z=Z)
Mcmc = list(R=10000, sbeta=0.2, keep=20)

set.seed(66)
out = rhierBinLogit(Data=Data, Mcmc=Mcmc)
breg = 5000/20
summary(out$Deltadraw, burnin=breg)
summary(out$Vbetadraw, burnin=breg)

## plotting examples
if(0){

## 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

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

### Arguments

- `y`: `nx1` vector of values of dep variable
- `X`: `nxk` design matrix
- `betabar`: `kx1` vector for the prior mean of the regression coefficients
- `A`: `kxk` prior precision matrix

### Details

Model:  
\[ y = X'\beta + e \]  
with \[ e \sim N(0, 1) \].

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

### Value

`kx1` 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, Anderson School, UCLA, <perossichi@gmail.com>.

References

For further discussion, see Bayesian Statistics and Marketing by Rossi, Allenby, and McCulloch. [http://www.perossi.org/home/bsm-1](http://www.perossi.org/home/bsm-1)

Examples

```r
if(!is.naSys.getenv("LONG_TEST")) R=1000 else R=10

## simulate data
set.seed(66)
X = cbind(rep(1,n), runif(n)); beta = c(1,2)
y = X %*% beta + rnorm(n)

## set prior
betabar = c(0,0)
A = diag(c(0.05, 0.05))

## 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(0.01, 0.05, 0.50, 0.95, 0.99))
mat = rbind(betadraw,mat); rownames(mat)[1] = "beta"
print(mat)
```

camera  

*Conjoint Survey Data for Digital Cameras*

describe camera

description

Panel dataset from a conjoint survey for digital cameras with 332 respondents. Data exclude respondents that always answered none, always picked the same brand, always selected the highest priced offering, or who appeared to be answering randomly.

Usage

data(camera)
Format

A list of lists. Each inner list corresponds to one survey respondent and contains a numeric vector (y) of choice indicators and a numeric matrix (X) of covariates. Each respondent participated in 16 choice scenarios each including 4 camera options (and an outside option) for a total of 80 rows per respondent.

Details

The covariates included in each X matrix are:

```r
...$canon  an indicator for brand Canon
...$sony   an indicator for brand Sony
...$nikon  an indicator for brand Nikon
...$panasonic  an indicator for brand Panasonic
...$pixels an indicator for a higher pixel count
...$zoom   an indicator for a higher level of zoom
...$video  an indicator for the ability to capture video
...$swivel an indicator for a swivel video display
...$wifi   an indicator for wifi capability
...$price  in hundreds of U.S. dollars
```

Source


References

For analysis of a similar dataset, see Case Study 4, *Bayesian Statistics and Marketing* Rossi, Allenby, and McCulloch [http://www.perossi.org/home/bsm-1](http://www.perossi.org/home/bsm-1)
cheese

Arguments

- e: quadratic parameter ($0 < e < 1$)
- k: items are on a scale from $1, \ldots, k$

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, Anderson School, UCLA. <perossichi@gmail.com>.

References


See Also

- rscaleUsage

Examples

```r
cgetC(0.1, 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

```r
data(cheese)
```

Format

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

- `$RETAILER`: a list of 88 retailers
- `$VOLUME`: unit sales
- `$DISP`: percent ACV on display (a measure of advertising display activity)
- `$PRICE`: in U.S. dollars
Source


References

Chapter 3, *Bayesian Statistics and Marketing* by Rossi, Allenby, and McCulloch.

http://www.perossi.org/home/bsm-1

Examples

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

```r
## 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,2]=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)
```
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, nprint=BayesmConstant.nprint)

Arguments

zdraw \textit{Rxnobs} array of draws of indicators
cutoff cutoff probability for similarity (def: 0.9)
SILENT logical flag for silent operation (def: FALSE)
nprint print every nprint'th draw (def: 100)

Details

Define a similarity matrix, \( \text{Sim} \) with \( \text{Sim}[i,j]=1 \) 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 \( \text{loss}(E[\text{Sim}] - \text{Sim}(z)) \), where loss is absolute deviation.

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

Value

A list containing:

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, Anderson School, UCLA, <perossichi@gmail.com>.

References

For further discussion, see *Bayesian Statistics and Marketing* by Rossi, Allenby, and McCulloch Chapter 3.

http://www.perossi.org/home/bsm-1

See Also

rnmixGibbs

Examples

```r
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(1,0.5,0.5,1), ncol=2)
  Sigma2 = matrix(c(1,0.5,0.5,1), 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
  Data = list(y=dm$x)
  ncomp = 2
  Prior = list(ncomp=ncomp, a=c(rep(100,ncomp)))
  R = 2000
  Mcmc = list(R=R, keep=1)
  out = rnmixGibbs(Data=Data, Prior=Prior, Mcmc=Mmc)

  ## find clusters
  begin = 500
  end = R
  outclusterMix = clusterMix(out$mmix$zdraw[begin:end,])

  ## check on clustering versus "truth"
  ## note: there could be switched labels
  table(outclusterMix$cluster, dm$z)
}
```
condMom

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

Description

condMom compute moments of conditional distribution of the $i$th element of a multivariate normal given all others.

Usage

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

Arguments

- `x` vector of values to condition on; $i$th element not used
- `mu` mean vector with `length(x) = n`
- `sigi` inverse of covariance matrix; dimension $n \times n$
- `i` conditional distribution of $i$th element

Details

$x \sim MVN(mu, sigi^{-1})$.

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

Value

A list containing:

- `cmean` conditional mean
- `cvar` conditional variance

Warning

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

Author(s)

Peter Rossi, Anderson School, UCLA, <perossichi@gmail.com>.

References

For further discussion, see Bayesian Statistics and Marketing by Rossi, Allenby, and McCulloch. [http://www.perossi.org/home/bsm-1](http://www.perossi.org/home/bsm-1)
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 (rmnlnIndepMetrop 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: $nxp \times na$ matrix of alternative-specific vars
- Xd: $n \times nd$ 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 of dimension $n \times (p - DIFF) \times (INT + nd) + na$.

Note
rmnpGibbs assumes that the base alternative is the default.

Author(s)
Peter Rossi, Anderson School, UCLA, <perossichi@gmail.com>.
customerSat

References
For further discussion, see *Bayesian Statistics and Marketing* by Rossi, Allenby, and McCulloch.
http://www.perossi.org/home/bsm-1

See Also
rmnIndepMetrop, rmnpGibbs

Examples

```r
na=2; nd=1; p=3
vec = c(1, 1.5, 0.5, 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 (1 is "Poor" and 10 is "Excellent").

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 Number of Customers
- ...$q7 Reach of Advertising
- ...$q8 Long-term Exposure
- ...$q9 Distribution
- ...$q10 Distribution to Right Geographic Areas
**Source**


**References**

Case Study 3, *Bayesian Statistics and Marketing* by Rossi, Allenby, and McCulloch.  
http://www.perossi.org/home/bsm-1

**Examples**

```r
data(customerSat)
apply(as.matrix(customerSat),2,table)
## see also examples for 'rscaleUsage'
```

---

**detailing**  
*Physician Detailing Data*

**Description**

Monthly data on physician detailing (sales calls). 23 months of data for each of 1000 physicians; includes physician covariates.

**Usage**

```r
data(detailing)
```

**Format**

The `detailing` object is a list containing two data frames, `counts` and `demo`.

**Details**

In the `counts` data frame:

- `...$id` identifies the physician
- `...$scrips` the number of new prescriptions ordered by the physician for the drug detailed
- `...$detailing` the number of sales calls made to each physician per month
- `...$lagged_scripts` scripts value for prior month

In the `demo` data frame:

- `...$id` identifies the physician
- `...$generalphys` dummy for if doctor is a "general practitioner"
- `...$specialist` dummy for if the physician is a specialist in the therapeutic class for which the drug is intended
- `...$mean_samples` the mean number of free drug samples given the doctor over the sample period
Source


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)
}
rm(detailing, counts)
cat("Finished reading data", fill=TRUE)
}

Data = list(regdata=regdata, Z=Z)

nvar = ncol(X)      # Number of X variables
nz = ncol(Z)        # Number of Z variables
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)

### plotting examples
if(0){
  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, eMixMargDen computes the marginal densities for each component in the multivariate mixture on a user-supplied grid and then averages over the MCMC draws.

Usage

eMixMargDen(grid, probdraw, compdraw)
Arguments

grid array of grid points, grid[,i] are ordinates for ith dimension of the density
probdraw array where each row contains a draw of probabilities of the mixture component
compdraw list of lists of draws of mixture component moments

Details

length(compdraw) is the number of MCMC draws.
compdraw[i] is a list draws of mu and of the inverse Cholesky 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 decom 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, Anderson School, UCLA, <perossichi@gmail.com>.

References

For further discussion, see Bayesian Statistics and Marketing by Rossi, Allenby, and McCulloch.
http://www.perossi.org/home/bsm-1

See Also

rnmixGibbs

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, HALTON=TRUE, pn)

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

HALTON
  if TRUE, uses Halton sequence. If FALSE, uses runif random number generator (def: TRUE)

pn
  prime number used for Halton sequence (def: the smallest prime numbers, i.e. 2, 3, 5, ...)

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 variance and mean 0 but different truncation points. See 'examples' below for an example with two integrals at different truncation points.

The user can choose between two random number generators for the numerical integration: psuedo-random numbers by runif or quasi-random numbers by a Halton sequence. Generally, the quasi-random (Halton) sequence is more uniformly distributed within domain, so it shows lower error and improved convergence than the psuedo-random (runif) sequence (Morokoff and Caflisch, 1995).

For the prime numbers generating Halton sequence, we suggest to use the first smallest prime numbers. Halton (1960) and Kocis and Whiten (1997) prove that their discrepancy measures (how uniformly the sample points are distributed) have the upper bounds, which decrease as the generating prime number decreases.

Note: For a high dimensional integration (10 or more dimension), we suggest use of the psuedo-random number generator (runif) because, according to Kocis and Whiten (1997), Halton sequences may be highly correlated when the dimension is 10 or more.

Author(s)

Peter Rossi, Anderson School, UCLA, <perossichi@gmail.com>.
Keunwoo Kim, Anderson School, UCLA, <keunwoo.kim@gmail.com>.

References

For further discussion, see Bayesian Statistics and Marketing by Rossi, Allenby, and McCulloch, Chapter 2.
http://www.perossi.org/home/bsm-1

Examples

```r
Sigma = matrix(c(1, 0.5, 0.5, 1), ncol=2)
L = t(chol(Sigma))
trunpt = c(0,0,1,1)
above = c(1,1)

# drawn by Halton sequence
ghkvec(L, trunpt, above, r=100)

# use prime number 11 and 13
ghkvec(L, trunpt, above, r=100, HALTON=TRUE, pn=c(11,13))

# drawn by R::runif
ghkvec(L, trunpt, above, r=100, HALTON=FALSE)
```

**llmnl**  
*Evaluate Log Likelihood for Multinomial Logit Model*

**Description**

llmnl evaluates log-likelihood for the multinomial logit model.

**Usage**

```r
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 * k` design matrix (use `createX` to create `X`)

**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 \( i \)th 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, Anderson School, UCLA, <perossichi@gmail.com>.

References
For further discussion, see *Bayesian Statistics and Marketing* by Rossi, Allenby, and McCulloch. 
http://www.perossi.org/home/bsm-1

See Also
createX, rmnlindepmetrop

Examples
```r
## 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**: n*(p-1) x k array where X is from differenced system
- **y**: vector of n indicators of multinomial response (1, ..., p)
- **r**: number of draws used in GHK
**Details**

\[ X \text{ is } (p - 1) \times nxk \text{ matrix. Use } \text{createX with DIFF=TRUE} \text{ to create } X. \]

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

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 \( j = 1, \ldots, p - 1 \) then \( A_j w = A_j mu + A_j e > 0 \) is equivalent to \( y = j \).

Thus, if \( y = j \), we have \( A_j e > -A_j mu \). Lower truncation is \(-A_j mu \) and \( \text{cov} = A_j Sigma(A_j) \). 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, Anderson School, UCLA, <perossichi@gmail.com>.

**References**

For further discussion, see Chapters 2 and 4, *Bayesian Statistics and Marketing* by Rossi, Allenby, and McCulloch.

http://www.perossi.org/home/bsm-1

**See Also**

createX, rmnpgibbs

**Examples**

```r
## Not run: ll=llmnp(beta,Sigma,X,y,r)
```
llnhlogit  
Evaluate Log Likelihood for non-homothetic Logit Model

Description

llnhlogit evaluates log-likelihood for the Non-homothetic Logit model.

Usage

llnhlogit(theta, choice, lnprices, Xexpend)

Arguments

theta  
parameter vector (see details section)

choice  
nx1 vector of choice (1, ..., p)

lnprices  
nxp array of log-prices

Xexpend  
nxd array of vars predicting expenditure

Details

Non-homothetic logit model, \( Pr(i) = \exp(tauv_i)/\sum_j \exp(tauv_j) \)

\( v_i = alpha_i - e^{kappaStar_i}u^i - lnpi \)

\( tau \) is the scale parameter of extreme value error distribution.

\( u^i \) solves \( psi_i(u^i)E/p_i \).

\( ln(psi_i(U)) = alpha_i - e^{kappaStar_i}U \).

\( ln(E) = gamma'Xexpend. \)

Structure of theta vector:

alpha: px1 vector of utility intercepts.

kappaStar: px1 vector of utility rotation parms expressed on natural log scale.

gamma: kx1 – expenditure variable coefs.

tau: 1x1 – 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, Anderson School, UCLA, <perossichi@gmail.com>.
References

For further discussion, see Chapter 4, *Bayesian Statistics and Marketing* by Rossi, Allenby, and McCulloch.

http://www.perossi.org/home/bsm-1

See Also

simnhlogit

Examples

```r
N=1000; p=3; k=1
theta = c(rep(1,p), seq(from=-1,to=1,length=p), rep(2,k), 0.5)
lnprices = matrix(runif(N*p), ncol=p)
Xexpend = matrix(runif(N*k), ncol=k)
simdata = simnhlogit(theta, lnprices, Xexpend)

## evaluate likelihood at true theta
llstar = llnhlogit(theta, simdata$y, simdata$lnprices, simdata$Xexpend)
```

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 (this must be a matrix)

Details

\[ Z = \frac{nu \times ssq}{\chi^2_{nu}} \]  
with \( Z \sim \text{Inverted Chi-Squared}. \)

`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, Anderson School, UCLA, <perossichi@gmail.com>.

References

For further discussion, see Chapter 2, Bayesian Statistics and Marketing by Rossi, Allenby, and McCulloch.

http://www.perossi.org/home/bsm-1

See Also
dchisq

Examples

lndIChisq(3, 1, matrix(2))

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

$Z \sim \text{Inverted Wishart}(nu,V)$. 
In this parameterization, $E[Z] = 1/(nu - k - 1)V$, where $V$ is a $k \times k$ 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, Anderson School, UCLA, <perossichi@gmail.com>.

**References**

For further discussion, see Chapter 2, *Bayesian Statistics and Marketing* by Rossi, Allenby, and McCulloch.

[http://www.perossi.org/home/bsm-1](http://www.perossi.org/home/bsm-1)

**See Also**

rwishart

**Examples**

```r
lndIwishart(5, diag(3), diag(3)+0.5)
```

---

**lndMvn**

*Compute Log of Multivariate Normal Density*

**Description**

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

**Usage**

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

**Arguments**

- `x`: density ordinate
- `mu`: mu vector
- `rooti`: inv of upper triangular Cholesky root of Σ

**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, Anderson School, UCLA, <perossichi@gmail.com>.

References

For further discussion, see Chapter 2, *Bayesian Statistics and Marketing* by Rossi, Allenby, and McCulloch.
http://www.perossi.org/home/bsm-1

See Also

`IndMvst`

Examples

```r
Sigma = matrix(c(1, 0.5, 0.5, 1), ncol=2)
IndMvtn(x=c(rep(0,2)), mu=c(rep(0,2)), rooti=backsolve(chol(Sigma),diag(2)))
```

---

### `IndMvst`
*Compute Log of Multivariate Student-t Density*

**Description**

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

**Usage**

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

**Arguments**

- `x`: density ordinate
- `nu`: d.f. parameter
- `mu`: mu vector
- `rooti`: inv of Cholesky root of Σ
- `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, Anderson School, UCLA, <perossichi@gmail.com>.

References
For further discussion, see Chapter 2, Bayesian Statistics and Marketing by Rossi, Allenby, and McCulloch.
http://www.perossi.org/home/bsm-1

See Also
lindMvn

Examples
Sigma = matrix(c(1, 0.5, 0.5, 1), ncol=2)
lindMvst(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.

Usage
logMargDenNR(ll)

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

Value
Approximation to log marginal density value.
Warning

This approximation can be influenced by outliers in the vector of log-likelihoods; use with care. This routine is a utility routine that does not check the input arguments for proper dimensions and type.

Author(s)

Peter Rossi, Anderson School, UCLA, <perossichi@gmail.com>.

References

For further discussion, see Chapter 6, Bayesian Statistics and Marketing by Rossi, Allenby, and McCulloch. http://www.perossi.org/home/bsm-1

Description

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

Usage

data(margarine)

Format

The detailing object is a list containing two data frames, choicePrice and demos.

Details

In the choicePrice data frame:

...$hhid household ID
...$choice multinomial indicator of one of the 10 products

The products are indicated by brand and type.

Brands:

...$Pk Parkay
...$BB BlueBonnett
...$Fl Fleischmanns
...$Hse house
...$Gen generic
...$Imp Imperial
...$SS Shed Spread
Product type:

...$_5tk  stick
...$_Tub  tub

In the demos data frame:

...$Fs3_4  dummy for family size 3-4
...$Fs5    dummy for family size >= 5
...$college dummy for education status
...$whtcollar dummy for job status
...$retired dummy for retirement status

All prices are in U.S. dollars.

Source

References
Chapter 5, Bayesian Statistics and Marketing by Rossi, Allenby, and McCulloch.
http://www.perossi.org/home/bsm-1

Examples

```r
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)

mat=apply(as.matrix(margarine$choicePrice[,3:12]), 2, mean)
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$choicePrice[,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) {
  nob = sum(chPr[,1]==hhidl[i])
  if(nob >= 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
  }
}
lgt = length(lgtdata)

## now extract demos corresponding to hhs in lgdata
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
mcmc = list(keep=keep, R=R)

out = rhierMnlRwMixture(Data=list(p=p, lgdata=lgtdata, Z=Z),
                        Prior=list(ncomp=1), Mcmc=mcmc)

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

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

Compute Marginal Density for Multivariate Normal Mixture

Description

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

Usage

mixDen(x, pvec, comps)

Arguments

x array where ith column gives grid points for ith variable
pvec vector of mixture component probabilities
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, Anderson School, UCLA, <perossichi@gmail.com>.

References

For further discussion, see Chapter 3, Bayesian Statistics and Marketing by Rossi, Allenby, and McCulloch.
http://www.perossi.org/home/bsm-1
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**

```r
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, Anderson School, UCLA, <perossichi@gmail.com>.
References
For further discussion, see Chapter 3, *Bayesian Statistics and Marketing* by Rossi, Allenby, and McCulloch.
http://www.perossi.org/home/bsm-1

See Also
rnmixGibbs, mixDen

---

**mnlHess**

*Computes –Expected Hessian for Multinomial Logit*

**Description**

*mnlHess* computes expected Hessian \( E[H] \) for Multinomial Logit Model.

**Usage**

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

**Arguments**

- **beta**: \( k \times 1 \) vector of coefficients
- **y**: \( n \times 1 \) vector of choices, \((1, \ldots, p)\)
- **X**: \( n \times p \times k \) Design matrix

**Details**

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

**Value**

\( k \times k \) matrix

**Warning**

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

**Author(s)**

Peter Rossi, Anderson School, UCLA, <perossichi@gmail.com>.

**References**

For further discussion, see Chapter 3, *Bayesian Statistics and Marketing* by Rossi, Allenby, and McCulloch.
http://www.perossi.org/home/bsm-1
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 probabilities.

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

Value

px1 vector of choice probabilities

Author(s)

Peter Rossi, Anderson School, UCLA, <perossichi@gmail.com>.

References

For further discussion, see Chapters 2 and 4, Bayesian Statistics and Marketing by Rossi, Allenby, and McCulloch.

http://www.perossi.org/home/bsm-1
momMix

See Also
rmnpGibbs, createX

Examples

```r
## example of computing MNP probabilites
## here Xa has the prices of each of the 3 alternatives
Xa  = matrix(c(1.1, 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, 0.5, 0.5, 1), ncol=2)

mnpProb(beta, Sigma, X)
```

### Description

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

#### Usage

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

#### Arguments

- `probdraw`: A `Rxncmp` list of draws of mixture probs
- `compdraw`: A 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[i]`: is a list of lists with mixture components.
- `compdraw[[i]][[j]][[1]]`: is the mean parameter vector for the `j`th component, `i`th MCMC draw.
- `compdraw[[i]][[j]][[2]]`: is the UL decomposition of `Σ^{-1}` for the `j`th component, `i`th MCMC draw.

#### Value

A list containing:

- `mu`: posterior expectation of mean
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 \( k \times k \) Cov matrix stored as a \( k \times k \times 1 \) vector (col by col)

Details

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

Value

\( k \times k \times 1 \) vector with correlation matrix

sigma posterior expectation 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, Anderson School, UCLA, <perossichi@gmail.com>.

References

For further discussion, see Chapter 5, Bayesian Statistics and Marketing by Rossi, Allenby, and McCulloch.

http://www.perossi.org/home/bsm-1

See Also

rmixGibbs
**numEff**

**Warning**

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

**Author(s)**

Peter Rossi, Anderson School, UCLA, <perossichi@gmail.com>.

**Examples**

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

**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**

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

**Arguments**

- `x` \(Rx1\) 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**

A list containing:

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

Description
Weekly sales of refrigerated orange juice at 83 stores. Contains demographic information on those stores.

Usage
data(orangeJuice)

Format
The orangeJuice object is a list containing two data frames, yx and storedemo.

Details
In the yx data frame:

...$store   store number
...$brand   brand indicator
...$week    week number
...$logmove log of the number of units sold
...$constant a vector of 1s
...$price#   price of brand #
...$deal     in-store coupon activity
...$feature feature advertisement
...$profit   profit
The price variables correspond to the following brands:

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

In the storedemo data frame:

...$STORE store number
...$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


References

Chapter 5, *Bayesian Statistics and Marketing* by Rossi, Allenby, and McCulloch

http://www.perossi.org/home/bsm-1

Examples

```r
# 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)

## 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=T)

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

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

  ## plotting examples
  if(0){ plot(out$betadraw) }
}
plot.bayesm.hcoef

\}

---

plot.bayesm.hcoef  \hspace{1cm} \textit{Plot Method for Hierarchical Model Coefs}

\section*{Description}

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

\section*{Usage}

\begin{verbatim}
## S3 method for class 'bayesm.hcoef'
plot(x, names, burnin, ...)
\end{verbatim}

\section*{Arguments}

\begin{itemize}
\item \texttt{x} \hspace{1cm} An object of S3 class, \texttt{bayesm.hcoef}
\item \texttt{names} \hspace{1cm} a list of names for the variables in the hierarchical model
\item \texttt{burnin} \hspace{1cm} no draws to burnin (def: 0.1 * \texttt{R})
\item \texttt{...} \hspace{1cm} standard graphics parameters
\end{itemize}

\section*{Details}

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

\texttt{plot.bayesm.hcoef} is also exported for use as a standard function, as in \texttt{plot.bayesm.hcoef(array)}.

\section*{Author(s)}

Peter Rossi, Anderson School, UCLA, <perossichi@gmail.com>.

\section*{See Also}

\texttt{rhierMnlRwMixture}, \texttt{rhierLinearModel}, \texttt{rhierLinearMixture}, \texttt{rhierNegbinRw}

\section*{Examples}

\begin{verbatim}
## Not run: out=rhierLinearModel(Data,Prior,Mcmc); plot(out$betadraws)
\end{verbatim}
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

```r
## 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: `0.1 * 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, Anderson School, UCLA, <perossichi@gmail.com>.

Examples

```r
## 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: 0.1 * 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 probabilities. 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, Anderson School, UCLA, <perossichi@gmail.com>.

See Also

rnmixGibbs, rhierMnlRwMixture, rhierLinearMixture, rDPGibbs
Examples

```r
## not run
# out = rnmixGibbs(Data, Prior, Mcmc)

## plot bivariate distributions for dimension 1,2; 3,4; and 1,3
# plot(out,bi.sel=list(c(1,2),c(3,4),c(1,3)))
```

---

**rbayesBLP**

*Bayesian Analysis of Random Coefficient Logit Models Using Aggregate Data*

**Description**

rbayesBLP implements a hybrid MCMC algorithm for aggregate level sales data in a market with differentiated products. bayesm version 3.1-0 and prior versions contain an error when using instruments with this function; this will be fixed in a future version.

**Usage**

```r
rbayesBLP(Data, Prior, Mcmc)
```

**Arguments**

- `Data` list(X, share, J, Z)
- `Prior` list(sigmasqR, theta_hat, A, deltabar, Ad, nu0, s0_sq, VOmega)
- `Mcmc` list(R, keep, nprint, H, initial_theta_bar, initial_r, initial_tau_sq, initial_Omega, initial_delta, s, cand_cov, tol)

**Value**

A list containing:

- `thetabardraw` $K \times R/keep$ matrix of random coefficient mean draws
- `Sigmadraw` $K \times K\times R/keep$ matrix of random coefficient variance draws
- `rdraw` $K \times K\times R/keep$ matrix of $r$ draws (same information as in Sigmadraw)
- `tausqdraw` $R/keep\times 1$ vector of aggregate demand shock variance draws
- `Omegadraw` $2 \times 2 \times R/keep$ matrix of correlated endogenous shock variance draws
- `deltadraw` $I \times R/keep$ matrix of endogenous structural equation coefficient draws
- `acceptrate` scalar of acceptance rate of Metropolis-Hasting
- `s` scale parameter used for Metropolis-Hasting
- `cand_cov` var-cov matrix used for Metropolis-Hasting

**Argument Details**

Data = `list(X, share, J, Z)` [Z optional]
J: number of alternatives, excluding an outside option
X: $J \times TxK$ matrix (no outside option, which is normalized to 0).
   If IV is used, the last column of X is the endogeneous variable.
share: $J \times T$ vector (no outside option).
   Note that both the share vector and the X matrix are organized by the $jt$ index.
   $j$ varies faster than $t$, i.e. $(j = 1, t = 1), (j = 2, t = 1), ..., (j = J, T = 1), ..., (j = J, t = T)$
Z: $J \times TxI$ matrix of instrumental variables (optional)

Prior = list(sigmasqR, theta_hat, A, deltabar, Ad, nu0, s0_sq, VOmega) [optional]

sigmasqR: $K \times (K + 1)/2$ vector for $r$ prior variance (def: diffuse prior for $\Sigma$)
theta_hat: $K$ vector for $\theta_{bar}$ prior mean (def: 0 vector)
A: $K \times K$ matrix for $\theta_{bar}$ prior precision (def: $0.01 \times \text{diag}(K)$)
deltabar: $I$ vector for $\delta$ prior mean (def: 0 vector)
Ad: $I \times I$ matrix for $\delta$ prior precision (def: $0.01 \times \text{diag}(I)$)
nu0: d.f. parameter for $\tau_{s,q}$ and $\Omega$ prior (def: $K+1$)
s0_sq: scale parameter for $\tau_{s,q}$ prior (def: 1)
VOmega: $2 \times 2$ matrix parameter for $\Omega$ prior (def: $\text{matrix}(c(1,0.5,0.5,1,2,2))$

Mcmc = list(R, keep, nprint, H, initial_theta_bar, initial_r, initial_tau_sq, initial_Omega, initial_delta, s, cand_cov, tol) [only R and H required]

R: number of MCMC draws
keep: MCMC thinning parameter – keep every keepth draw (def: 1)
nprint: print the estimated time remaining for every nprint’th draw (def: 100, set to 0 for no print)
H: number of random draws used for Monte-Carlo integration
initial_theta_bar: initial value of $\theta_{bar}$ (def: 0 vector)
initial_r: initial value of $r$ (def: 0 vector)
initial_tau_sq: initial value of $\tau_{s,q}$ (def: 0.1)
initial_Omega: initial value of $\Omega$ (def: diag(2))
initial_delta: initial value of $\delta$ (def: 0 vector)
s: scale parameter of Metropolis-Hasting increment (def: automatically tuned)
cand_cov: var-cov matrix of Metropolis-Hasting increment (def: automatically tuned)
tol: convergence tolerance for the contraction mapping (def: $1e-6$)

Model Details

Model and Priors (without IV):

$u_{jt} = X_{jt}\theta_i + \eta_{jt} + e_{jt}$
$e_{jt} \sim \text{type I Extreme Value (logit)}$
$\theta_i \sim N(\theta_{bar}, \Sigma)$
$\eta_{jt} \sim N(0, \tau_{s,q})$

This structure implies a logit model for each consumer ($\theta$). Aggregate shares (share) are produced by integrating this consumer level logit model over the assumed normal distribution of $\theta$.

$r \sim N(0, \text{diag}(\text{sigmasqR}))$
$\theta_{bar} \sim N(\theta_{bar}, A^{-1})$
\[ \tau_{sq} \sim n_{00} + s_{0sq}/\chi^2(n_{00}) \]

Note: we observe the aggregate level market share, not individual level choices.

Note: \( r \) is the vector of nonzero elements of cholesky root of \( \Sigma \). Instead of \( \Sigma \) we draw \( r \), which is one-to-one correspondence with the positive-definite \( \Sigma \).

**Model and Priors (with IV)::**

\[ u_{ijt} = X_{jt} \theta_i + \eta_{jt} + e_{ijt} \]
\[ e_{ijt} \sim \text{type I Extreme Value (logit)} \]
\[ \theta_i \sim N(\theta_{bar}, \Sigma) \]

\[ X_{jt} = [X_{cjojt}, X_{endojt}] \]
\[ X_{cendojt} = Z_{jt} \delta_{jt} + \zeta_{jt} \]
\[ \text{vec}(\zeta_{jt}, \eta_{jt}) \sim N(0, \Omega) \]

\[ r \sim N(0, \text{diag}(\text{sigmasqR})) \]
\[ \theta_{bar} \sim N(\theta_{hat}, A^{-1}) \]
\[ \delta \sim N(\delta_{bar}, Ad^{-1}) \]
\[ \Omega \sim IW(n_{00}, VOmega) \]

**MCMC and Tuning Details:**

**MCMC Algorithm::** Step 1 (\( \Sigma \)):
- Given \( \theta_{bar} \) and \( \tau_{sq} \), draw \( r \) via Metropolis-Hasting.
- Covert the drawn \( r \) to \( \Sigma \).

Note: if user does not specify the Metropolis-Hasting increment parameters (\( s \) and \( \text{cand\_cov} \)), \( rbayesBLP \) automatically tunes the parameters.

Step 2 without IV (\( \theta_{bar}, \tau_{sq} \)):
- Given \( \Sigma \), draw \( \theta_{bar} \) and \( \tau_{sq} \) via Gibbs sampler.

Step 2 with IV (\( \theta_{bar}, \delta, \Omega \)):
- Given \( \Sigma \), draw \( \theta_{bar}, \delta \), and \( \Omega \) via IV Gibbs sampler.

**Tuning Metropolis-Hastings algorithm::** \( r\_\text{cand} = r\_\text{old} + s*\text{N}(0,\text{cand\_cov}) \)

Fix the candidate covariance matrix as \( \text{cand\_cov0} = \text{diag}(\text{rep}(0.1, K), \text{rep}(1, K*(K-1)/2)) \).
Start from \( s_0 = 2.38/\text{sqrt}(\text{dim}(r)) \)

Repeat{
Run 500 MCMC chain.
If acceptance rate < 30% => update \( s_1 = s_0/5 \).
If acceptance rate > 50% => update \( s_1 = s_0*3 \).
(Store r draws if acceptance rate is 20~80%.)
\( s_0 = s_1 \)
} until acceptance rate is 30~50%
Scale matrix $C = s_1 \sqrt{c_{and}}$
Correlation matrix $R = \text{Corr}(r \text{ draws})$
Use $C^*R^*C$ as $s^2c_{and}$.

Author(s)

Keunwoo Kim, Anderson School, UCLA, <keunwoo.kim@gmail.com>.

References

For further discussion, see Bayesian Analysis of Random Coefficient Logit Models Using Aggregate Data by Jiang, Manchanda, and Rossi, Journal of Econometrics, 2009.


Examples

```r
if(!char(Sys.getenv("LONG_TEST")) != "") {

## Simulate aggregate level data
simulData <- function(para, others, Hbatch) {
  # Hbatch does the integration for computing market shares
  #   in batches of size Hbatch

  ## parameters
  theta_bar <- para$theta_bar
  Sigma <- para$Sigma
  tau_sq <- para$tau_sq

  T <- others$T
  J <- others$J
  p <- others$p
  H <- others$H
  K <- J + p

  ## build X
  X <- matrix(rnorm(T*J*p), T*J, p)
  inter <- NULL
  for (t in 1:T) { inter <- rbind(inter, diag(J)) }
  X <- cbind(inter, X)

  ## draw eta ~ N(0, tau_sq)
  eta <- rnorm(T*J)*sqrt(tau_sq)
  X <- cbind(X, eta)

  share <- rep(0, J*H)
  for (HH in 1:(H/Hbatch)){
    ## draw theta ~ N(theta_bar, Sigma)
    cho <- chol(Sigma)
    theta <- matrix(rnorm(K*Hbatch), nrow=K, ncol=Hbatch)
    theta <- t(cho)**%*%theta + theta_bar

    ## utility
```
V <- X%*%rbind(theta, 1)
expV <- exp(V)
expSum <- matrix(colSums(matrix(expV, J, T+Hbatch)), T, Hbatch)
choiceProb <- expV / (1 + expSum)
share <- share + rowSums(choiceProb) / H

## the last K+1'th column is eta, which is unobservable.
X <- X[,c(1:K)]
return (list(X=X, share=share))

## true parameter
theta_bar_true <- c(-2, -3, -4, -5)
Sigma_true <- rbind(c(3,2,1.5,1), c(2,4,-1,1.5), c(1.5,-1,4,-0.5), c(1,1.5,-0.5,3))
cho <- chol(Sigma_true)
r_true <- c(log(diag(cho)), cho[1,2:4], cho[2,3:4], cho[3,4])
tau_sq_true <- 1

## simulate data
set.seed(66)
T <- 300
J <- 3
p <- 1
K <- 4
H <- 1000000
Hbatch <- 5000

dat <- simulData(para=list(theta_bar=theta_bar_true, Sigma=Sigma_true, tau_sq=tau_sq_true),
others=list(T=T, J=J, p=p, H=H), Hbatch)
X <- dat$X
share <- dat$share

## Mcmc run
R <- 2000
H <- 50
Data1 <- list(X=X, share=share, J=J)
Mcmc1 <- list(R=R, H=H, nprint=0)
set.seed(66)
out <- rbayesBLP(Data=Data1, Mcmc=Mcmc1)

## acceptance rate
out$acceptrate

## summary of draws
summary(out$thetabardraw)
summary(out$sigmadraw)
summary(out$tausqdraw)

## plotting draws
plot(out$thetabardraw)
plot(out$sigmadraw)
Description

rbiNormGibbs implements a Gibbs Sampler for the bivariate normal distribution. Intermediate moves are plotted and the output is contrasted with the iid sampler. 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

\((\theta_1, \theta_2) \sim N((0,0), \Sigma)\) with \(\Sigma = \text{matrix}(c(1, \rho, \rho, 1), \text{ncol}=2)\)

Value

Rx2 matrix of draws

Author(s)

Peter Rossi, Anderson School, UCLA, <perossichi@gmail.com>.

References

For further discussion, see Chapters 2 and 3, Bayesian Statistics and Marketing by Rossi, Allenby, and McCulloch.

http://www.perossi.org/home/bsm-1

Examples

```r
## Not run: out=rbiNormGibbs(rho=0.95)
```
rbprobitGibbs \hspace{1em} 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 \hspace{1em} \text{list(y, X)}
Prior \hspace{1em} \text{list(betabar, A)}
Mcmc \hspace{1em} \text{list(R, keep, nprint)}

Details

Model and Priors: \( z = X\beta + e \) with \( e \sim N(0, I) \)
\( y = 1 \) if \( z > 0 \)
\( \beta \sim N(\text{betabar}, A^{-1}) \)

Argument Details: Data = list(y, X)
\( y \): \( nx1 \) vector of 0/1 outcomes
\( X \): \( nxk \) design matrix

Prior = list(betabar, A) \( \text{[optional]} \)
\( \text{betabar} \): \( kx1 \) prior mean (def: 0)
\( A \): \( kxk \) prior precision matrix (def: 0.01*I)

Mcmc = list(R, keep, nprint) \( \text{[only R required]} \)
\( R \): number of MCMC draws
\( \text{keep} \): MCMC thinning parameter – keep every keepth draw (def: 1)
\( \text{nprint} \): print the estimated time remaining for every nprint’th draw (def: 100, set to 0 for no print)

Value

A list containing:

betadraw \hspace{1em} R/keepxk matrix of betadraws
**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, Anderson School, UCLA, <perossichi@gmail.com>.

References

For further discussion, see Chapter 2, Bayesian Statistics and Marketing by Rossi, Allenby, and McCulloch.
http://www.perossi.org/home/bsm-1

Examples

```r
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

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

Arguments

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

Model and Priors:  
\[ y_i \sim N(\mu_i, \Sigma_i) \]
\[ \theta_i = (\mu_i, \Sigma_i) \sim DP(G_0(\lambda), \alpha) \]

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

We parameterize the prior on \( \Sigma_i \) such that \( \text{mode}(\Sigma) = nu/(nu + 2)vI \). 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.

Argument Details:  
Data = list(y)  
y:  \[ nxk \] matrix of observations on k dimensional data

Prior = list(Prioralpha, lambda_hyper) [optional]

Prioralpha:  list(Istarmin, Istarmax, power)
$Istarmin$: is expected number of components at lower bound of support of alpha (def: 1)
$Istarmax$: is expected number of components at upper bound of support of alpha (def: min(50, 0.1*nrow(y)))
$power$: is the power parameter for alpha prior (def: 0.8)
lambda_hyper: list(alim, nulim, vlim)
$alim$: defines support of a distribution (def: c(0.01, 10))
$nulim$: defines support of nu distribution (def: c(0.01, 3))
$vlim$: defines support of v distribution (def: c(0.1, 4))

Mcmc = list(R, keep, nprint, maxuniq, SCALE, gridsize) [only R required]
R: number of MCMC draws  
keep: MCMC thinning parameter – keep every keepth draw (def: 1)  
nprint: print the estimated time remaining for every nprint'th draw (def: 100, set to 0 for no print)  
maxuniq: storage constraint on the number of unique components (def: 200)  
SCALE: should data be scaled by mean,std deviation before posterior draws (def: TRUE)  
gridsize: number of discrete points for hyperparameter priors (def: 20)  

**Details:**  
nmix is a list with 3 components. Several functions in the bayesm package that involve a Dirichlet Process or mixture-of-normals return nmix. Across these functions, a common structure is used for nmix in order to utilize generic summary and plotting functions.

**Value**  
A list containing:

- **nmix**  
  a list containing: probdraw, zdraw, compdraw (see “nmix Details” section)

- **alphadraw**  
  \(R/keep\times 1\) vector of alpha draws

- **nudraw**  
  \(R/keep\times 1\) vector of nu draws

- **adraw**  
  \(R/keep\times 1\) vector of a draws

- **vdraw**  
  \(R/keep\times 1\) vector of v draws

**Author(s)**  
Peter Rossi, Anderson School, UCLA, <perossichi@gmail.com>.

**See Also**

- **rnmixgibbs**, **rmixture**, **rmixGibbs**, **eMixMargDen**, **momMix**, **mixDen**, **mixDenBi**

**Examples**

```r
if(nchar(Sys.getenv("LONG_TEST"))) != 0) {R=2000} else {R=10}
set.seed(66)

## simulate univariate data from Chi-Sq

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=0.8)
Prior1 = list(Prioralpha=Prioralpha)
Mcmc = list(R=R, keep=1, maxuniq=200)
```
out1 = rDPGibbs(Prior=Prior1, Data=Data1, Mcmc=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 histogram
    plot(range(grid[,1]), 1.5*range(dchisq(grid[,1],df=chisqdf)),
         type="n", xlab=paste("Chisq ; N, obs", sep=""), ylab="")
    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)
mvar2 = 2
A = .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=0.8)
Prior2 = list(Prioralpha=Prioralpha)
Mcmc = list(R=R, keep=1, maxuniq=200)

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

if(0){
    ## plotting examples
rhierBinLogit

MCMC Algorithm for Hierarchical Binary Logit

Description

This function has been deprecated. Please use rhierMnlRwMixture instead.

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)
Prior    list(Deltabar, ADelta, nu, V)
Mcmc    list(R, keep, sbeta)
Details

**Model and Priors:** \( y_h = 1 \) with \( P_r = \exp(x'_h \beta_h)/(1 + \exp(x'_h \beta_h)) \) and \( \beta_h \) is \( nvar \times 1 \) \( h = 1, \ldots, \text{length}(\text{lgtdata}) \) units (or "respondents" for survey data)

\[ \beta_h = Z_{\Delta[h]} + u_h \]

Note: here \( Z_{\Delta} \) refers to \( Z_{\% \% \% \% \% \% \% \% \% \% \% \} \) \( \) the \( h \)th row of this product

\( \Delta \) is an \( nz \times nvar \) array

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

\[ \delta = \text{vec}(\Delta) \sim N(\text{vec}(\Delta_{\bar{b}}), V_{\beta}(x)A\Delta^{-1}) \]

\[ V_{\beta} \sim IW(\nu, V) \]

**Argument Details:** Data = \text{list}(\text{lgtdata}, Z) \{Z optional\}

\[ \text{lgtdata: list of lists with each cross-section unit MNL data} \]

\[ \text{lgtdata}[[h]]y: n_h \times 1 \text{ vector of binary outcomes (0,1)} \]

\[ \text{lgtdata}[[h]]x: n_h \times nvar \text{ design matrix for } h \text{'th unit} \]

\[ Z: nreg \times nz \text{ mat of unit chars (def: vector of ones)} \]

Prior = \text{list}(\Delta_{\bar{b}}, A\Delta, \nu, V) \{optional\}

\[ \Delta_{\bar{b}}: nz \times nvar \text{ matrix of prior means (def: 0)} \]

\[ A\Delta: \text{prior precision matrix (def: 0.01I)} \]

\[ \nu: \text{d.f. parameter for IW prior on normal component Sigma (def: nvar+3)} \]

\[ V: \text{pds location parm for IW prior on normal component Sigma (def: nul)} \]

Mcmc = \text{list}(R, keep, sbeta) \{only R required\}

\[ R: \text{number of MCMC draws} \]

\[ \text{keep: MCMC thinning parm – keep every keepth draw (def: 1)} \]

\[ \text{sbeta: scaling parm for RW Metropolis (def: 0.2)} \]

**Value**

A list containing:

\[ \text{Deltadraw } R/\text{keep} \times nz \times nvar \text{ matrix of draws of Delta} \]

\[ \text{betadraw } nlgt \times nvar \times R/\text{keep} \text{ array of draws of betas} \]

\[ \text{Vbetadraw } R/\text{keep} \times nz \times nvar \text{ matrix of draws of Vbeta} \]

\[ \text{llike } R/\text{keep} \times 1 \text{ vector of log-like values} \]

\[ \text{reject } R/\text{keep} \times 1 \text{ vector of reject rates over nlgt units} \]

**Note**

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

**Author(s)**

Peter Rossi, Anderson School, UCLA, <perossichi@gmail.com>. 
References

For further discussion, see Chapter 5, *Bayesian Statistics and Marketing* by Rossi, Allenby, and McCulloch.

http://www.perossi.org/home/bsm-1

See Also

rhierMnlRwMixture

Examples

```r
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

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, -0.5, 0.5, 0), nrow=nz, ncol=nvar)
iota = matrix(1, nrow=nvar, ncol=1)
Vbeta = diag(nvar) + 0.5*iota %*% t(iota)

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)
}

Data1 = list(lgtdata=lgtdata, Z=Z)
Mcmc1 = list(R=R)

out = rhierBinLogit(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$Deltadraw,tvalues=as.vector(Delta))
  plot(out$betadraw)
  plot(out$Vbetadraw,tvalues=as.vector(Vbeta[upper.tri(Vbeta, diag=TRUE)]))
}
```
rhierLinearMixture

Gibbs Sampler for Hierarchical Linear Model with Mixture-of-Normals Heterogeneity

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)
Prior list(deltabar, Ad, mubar, Amu, nu, V, nu.e, ssq, ncomp)
Mcmc list(R, keep, nprint)

Details

Model and Priors:  nreg regression equations with nvar as the number of X vars in each equation

\[ y_i = X_i \beta_i + e_i \text{ with } e_i \sim N(0, \tau_i) \]
\[ \tau_i \sim \nu.e \ast \text{ssq}/\chi^2_{\nu.e} \text{ where } \tau_i \text{ is the variance of } e_i \]
\[ B = Z\Delta + U \text{ or } \beta_i = \Delta^t Z[i,i]^t + u_i \]
\[ \Delta \text{ is an } n \times n \text{ var matrix} \]

Z should not include an intercept and should be centered for ease of interpretation. The mean of each of the nreg \( \beta \)s is the mean of the normal mixture. Use \text{summary()} to compute this mean from the \text{compdraw} output.

\[ u_i \sim N(\mu_{ind}, \Sigma_{ind}) \]
\[ ind \sim \text{multinomial(pvec)} \]

\[ pvec \sim \text{dirichlet}(a) \]
\[ \text{delta} = \text{vec}(\Delta) \sim N(\text{deltabar}, A_d^{-1}) \]
\[ \mu_j \sim N(\text{mubar}, \Sigma_j(x)A_{mu}^{-1}) \]
\[ \Sigma_j \sim \text{IW}(nu, V) \]

Be careful in assessing the prior parameter \text{Amu}: 0.01 can be too small for some applications. See chapter 5 of Rossi et al for full discussion.

Argument Details:  Data = list(regdata, Z) [Z optional]

regdata:  list of lists with X and y matrices for each of nreg=length(regdata) regressions
regdata[[i]]$x: \ n_{x \times nvar} \text{ design matrix for equation } i
regdata[[i]]$y: \ n_{x \times 1} \text{ vector of observations for equation } i
Z: \ n_{reg \times n} \text{ matrix of unit characteristics (def: vector of ones)}

Prior = list(deltabar, Ad, mubar, Amu, nu, V, nu.e, ssq, ncomp) [all but ncomp are optional]

deltabar: \ n_{\times nvar} \text{ vector of prior means (def: 0)}
Ad: \text{ prior precision matrix for vec(Delta) (def: 0.01*I)}
mubar: \ nvar \times 1 \text{ prior mean vector for normal component mean (def: 0)}
Amu: \text{ prior precision for normal component mean (def: 0.01)}
nu: \text{ d.f. parameter for IW prior on normal component Sigma (def: nvar+3)}
V: \text{ PDS location parameter for IW prior on normal component Sigma (def: nu*I)}
nu.e: \text{ d.f. parameter for regression error variance prior (def: 3)}
ssq: \text{ scale parameter for regression error variance prior (def: var(y_i))}
a: \text{ Dirichlet prior parameter (def: 5)}
ncomp: \text{ number of components used in normal mixture}

Mcmc = list(R, keep, nprint) [only R required]

R: \text{ number of MCMC draws}
keep: \text{ MCMC thinning parm – keep every keepth draw (def: 1)}
nprint: \text{ print the estimated time remaining for every nprint’th draw (def: 100, set to 0 for no print)}

nmix Details: nmix is a list with 3 components. Several functions in the bayesm package that
involve a Dirichlet Process or mixture-of-normals return nmix. Across these functions, a common
structure is used for nmix in order to utilize generic summary and plotting functions.

probdraw: \ ncomp \times R/keep \text{ matrix that reports the probability that each draw came from a particular component}
zdraw: \ R/keep \times nobs \text{ matrix that indicates which component each draw is assigned to (here, null)}
compdraw: \text{ A list of } R/keep \text{ lists of } ncomp \text{ lists. Each of the inner-most lists has 2 elements: a vector of draws for mu and a}

Value

A list containing:
taudraw \ R/keep \times nreg \text{ matrix of error variance draws}
betadraw \ nreg \times nvar \times R/keep \text{ array of individual regression coef draws}
Deltadraw \ R/keep \times nvar \times nvar \text{ matrix of Deltadraws}
nmix \text{ a list containing: probdraw, zdraw, compdraw (see “nmix Details” section)}

Author(s)

Peter Rossi, Anderson School, UCLA, <perossichi@gmail.com>.

References

For further discussion, see Chapter 5, Bayesian Statistics and Marketing by Rossi, Allenby, and McCulloch.
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 = 0.1
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(0.4, 0.2, 0.4)

## simulated data with Z
regdata = NULL
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)
rhierLinearModel

Gibbs Sampler for Hierarchical Linear Model with Normal Heterogeneity

Description

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

Usage

rhierLinearModel(Data, Prior, Mcmc)

Arguments

Data = list(regdata, Z)
Prior = list(Deltabar, A, nu.e, ssq, nu, V)
Mcmc = list(R, keep, nprint)

Details

Model and Priors: nreg regression equations with nvar X variables in each equation
\[ y_i = X_i \beta_i + e_i \] with \( e_i \sim N(0, \tau_i) \)
\[ \tau_i \sim nu.e * ssq_i / \chi^2_{nu.e} \] where \( \tau_i \) is the variance of \( e_i \)
\[ \beta_i \sim N(Z\Delta[i, ], V_\beta) \]

Note: Z\Delta is the matrix Z * \Delta and [i, ] refers to ith row of this product

\[ vec(\Delta) \text{ given } V_\beta \sim N(vec(Deltabar), V_\beta(x)A^{-1}) \]
\[ V_\beta \sim IW(nu, V) \]
\[ Delta, Deltabar \text{ are } nzxvar; A \text{ is } nznz; V_\beta \text{ is } nvarxvar. \]

Note: if you don’t have any Z variables, omit Z in the Data argument and a vector of ones will be inserted; the matrix \( \Delta \) will be 1xvar and should be interpreted as the mean of all unit \( \beta \)s.

Argument Details: Data = list(regdata, Z) [Z optional]
rhierLinearModel

regdata: list of lists with X and y matrices for each of nreg=length(regdata) regressions
regdata[[i]]$X: n_i x nvar design matrix for equation i
regdata[[i]]$y: n_i x 1 vector of observations for equation i
Z: nreg x nz matrix of unit characteristics (def: vector of ones)

Prior = list(Deltabar, A, nu.e, ssq, nu, V) [optional]

Deltabar: n.nz x nvar matrix of prior means (def: 0)
A: nz x nz matrix for prior precision (def: 0.01I)
nu.e: d.f. parameter for regression error variance prior (def: 3)
ssq: scale parameter for regression error var prior (def: var(y_i))
nu: d.f. parameter for Vbeta prior (def: nvar+3)
V: Scale location matrix for Vbeta prior (def: nu*I)

Mcmc = list(R, keep, nprint) [only R required]
R: number of MCMC draws
keep: MCMC thinning parm – keep every keepth draw (def: 1)
nprint: print the estimated time remaining for every nprintth draw (def: 100, set to 0 for no print)

Value
A list containing:

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

Author(s)
Peter Rossi, Anderson School, UCLA, <perossichi@gmail.com>.

References
For further discussion, see Chapter 3, Bayesian Statistics and Marketing by Rossi, Allenby, and McCulloch.
http://www.perossi.org/home/bsm-1

See Also
rhierLinearMixture

Examples
if(nchar(Sys.getenv("LONG_TEST")) != 0) {R=2000} else {R=10}
set.seed(66)
nreg = 100
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 heterogeneity. 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-parametric method in the sense that the DP prior can accommodate heterogeneity of an unknown form.
Usage

rhierMnlDP(Data, Prior, Mcmc)

Arguments

Data list(lgtdata, Z, p)
Prior list(deltabar, Ad, Prioralpha, lambda_hyper)
Mcmc list(R, keep, nprint, s, w, maxunique, gridsize)

Details

Model and Priors: \( y_i \sim MNL(X_i, \beta_i) \) with \( i = 1, \ldots, \text{length(lgtdata)} \) and where \( \theta_i \) is
\( \beta_i = Z \Delta[i,] + u_i \)
\( \Delta \) is the matrix \( Z \ast \Delta; [i,] \) refers to \( i \)th row of this product
\( \Delta \) is an \( n_{x \text{ngt}} \) matrix
\( \beta_i \sim N(\mu_i, \Sigma_i) \)
\( \theta_i = (\mu_i, \Sigma_i) \sim DP(G_0(\lambda), \alpha) \)

\( G_0(\lambda) : \)
\( \mu_i | \Sigma_i \sim N(0, \Sigma_i(x) a^{-1}) \)
\( \Sigma_i \sim IW(nu, nu * v * I) \)
\( \delta = \text{vec}(\Delta) \sim N(\text{deltabar}, A_d^{-1}) \)

\( \lambda(a, nu, v) : \)
\( a \sim \text{uniform}[\text{alim[1]}, \text{alim[2]]} \)
\( nu \sim \text{dim(data)}-1 + \exp(z) \)
\( z \sim \text{uniform}[\text{dim(data)}-1+nulim[1], \text{nulim[2]}] \)
\( v \sim \text{uniform}[\text{vlim[1]}, \text{vlim[2]}] \)
\( \alpha \sim (1 - (\alpha - \text{alphamin})/(\text{alphamax} - \text{alphamin}))^{\text{power}} \)
\( \alpha = \text{alphamin} \) then expected number of components = Istarmin
\( \alpha = \text{alphamax} \) then expected number of components = Istarmax

\( Z \) should NOT include an intercept and is centered for ease of interpretation. The mean of each of the n1gt \( \beta s \) is the mean of the normal mixture. Use summary() to compute this mean from the compdraw output.

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; \( \text{nulim[1]} > 0 \).

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.

**Argument Details:** Data = list(lgtdata, Z, p) [Z optional]

- lgtdata: list of lists with each cross-section unit MNL data
- lgtdata[[i]]$y: $ nx1 vector of multinomial outcomes (1,..., m)
- lgtdata[[i]]$X: $ nxnvar design matrix for ith unit
- Z: nregxnz matrix of unit characteristics (def: vector of ones)
- p: number of choice alternatives

Prior = list(deltabar, Ad, Prioralpha, lambda_hyper) [optional]

- deltabar: nz * nvarx1 vector of prior means (def: 0)
- Ad: prior precision matrix for vec(D) (def: 0.01*I)
- Prioralpha: list(Istarmin, Istarmax, power)
- $Istarmin$: 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, 0.1*nlgd))
- $power$: power parameter for alpha prior (def: 0.8)
- lambda_hyper: list(alim, nulim, vlim)
- $alim$: defines support of a distribution (def: c(0.01, 2))
- $nulim$: defines support of nu distribution (def: c(0.001, 3))
- $vlim$: defines support of v distribution (def: c(0.1, 4))

Mcmc = list(R, keep, nprint, s, w, maxunique, gridsize) [only R required]

- R: number of MCMC draws
- keep: MCMC thinning parameter – keep every keepth draw (def: 1)
- nprint: print the estimated time remaining for every nprint’th draw (def: 100, set to 0 for no print)
- s: scaling parameter for RW Metropolis (def: 2.93/sqrt(nvar))
- w: fractional likelihood weighting parameter (def: 0.1)
- maxunique: storage constraint on the number of unique components (def: 200)
- gridsize: number of discrete points for hyperparameter priors, (def: 20)

**nmix Details:** nmix is a list with 3 components. Several functions in the bayesm package that
involve a Dirichlet Process or mixture-of-normals return nmix. Across these functions, a common
structure is used for nmix in order to utilize generic summary and plotting functions.

- probdraw: ncomp*R/keep matrix that reports the probability that each draw came from a particular component (here, a one-column matrix of 1s)
- zdraw: R/keepxnobs matrix that indicates which component each draw is assigned to (here, null)
- compdraw: A list of R/keep lists of ncomp lists. Each of the inner-most lists has 2 elements: a vector of draws for mu and a

**Value**

A list containing:

- Deltadraw R/keepnxz * nvar matrix of draws of Delta, first row is initial value
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. Large R values may be required (>20,000).

Author(s)

Peter Rossi, Anderson School, UCLA, <perossichi@gmail.com>.

References

For further discussion, see Chapter 5, Bayesian Statistics and Marketing by Rossi, Allenby, and McCulloch.

http://www.perossi.org/home/bsm-1

See Also

rhiemnlrwmixture

Examples

```r
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(0.4, 0.2, 0.4)

## simulate from MNL model conditional on X matrix
simmnlwx = function(n,X,beta) {
  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(0){
  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))

## plotting examples
if(0) {
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(lgtdata, Z, p)
- **Prior**: list(a, deltabar, Ad, mubar, nu, V, a, ncomp, SignRes)
- **Mcmc**: list(R, keep, nprint, s, w)

**Details**

**Model and Priors:**

\[ y_i \sim MNL(X_i, \beta_i) \] with \( i = 1, \ldots, \text{length(lgtdata)} \) and where \( \beta_i \) is \( nvar \times 1 \)

\[ \beta_i = Z \Delta[i,] + u_i \]

Note: \( Z \Delta \) is the matrix \( Z \times \Delta \) and \( [i,] \) refers to \( i \)th row of this product

\( \Delta \) is an \( nz \times nvar \) array

\[ u_i \sim N(\mu_{ind}, \Sigma_{ind}) \] with \( ind \sim \text{multinomial(pvec)} \)

\( pvec \sim \text{dirichlet(a)} \)

\[ \text{delta = vec(\Delta) \sim N(deltabar, A_d^{-1})} \]

\[ \mu_j \sim N(mubar, \Sigma_j(x)Amu^{-1}) \]

\[ \Sigma_j \sim IW(nu, V) \]

Note: \( Z \) should NOT include an intercept and is centered for ease of interpretation. The mean of each of the \( n\text{lgt} \) \( \beta \)s is the mean of the normal mixture. Use `summary()` to compute this mean from the `compdraw` output.

Be careful in assessing prior parameter Amu: 0.01 is too small for many applications. See chapter 5 of Rossi et al for full discussion.

**Argument Details:**

**Data = list(lgtdata, Z, p)** [Z optional]

- **lgtdata**: list of \( n\text{lgt}=\text{length(lgtdata)} \) lists with each cross-section unit MNL data
- **lgtdata[[i]]$y**: \( n_i \times 1 \) vector of multinomial outcomes (1, ..., m)
- **lgtdata[[i]]$X**: \( n_i \times \text{pvar} \) design matrix for \( i \)th unit
- **Z**: \( nreg \times nz \) matrix of unit chars (def: vector of ones)
- **p**: number of choice alternatives
Prior = list(a, deltabar, Ad, mubar, Amu, nu, V, a, ncomp, SignRes) [all but ncomp are optional]

a: \text{ncomp\times 1} vector of Dirichlet prior parameters (def: rep(5,ncomp))
deltabar: \text{nz \times nvar\times 1} vector of prior means (def: 0)
Ad: prior precision matrix for vec(D) (def: 0.01*I)
mubar: \text{nvar\times 1} prior mean vector for normal component mean (def: 0 if unrestricted; 2 if restricted)
Amu: prior precision for normal component mean (def: 0.01 if unrestricted; 0.1 if restricted)
nu: d.f. parameter for IW prior on normal component Sigma (def: nvar+3 if unrestricted; nvar+15 if restricted)
V: PDS location parameter for IW prior on normal component Sigma (def: nu*I if unrestricted; nu*D if restricted)
ncomp: number of components used in normal mixture
SignRes: \text{nvar\times 1} vector of sign restrictions on the coefficient estimates (def: rep(0,nvar))

Mcmc = list(R, keep, nprint, s, w) [only R required]

R: number of MCMC draws
keep: MCMC thinning parameter – keep every keepth draw (def: 1)
nprint: print the estimated time remaining for every nprint’th draw (def: 100, set to 0 for no print)
s: scaling parameter for RW Metropolis (def: 2.93/sqrt(nvar))
w: fractional likelihood weighting parameter (def: 0.1)

Sign Restrictions: If \beta_k has a sign restriction: \beta_k = SignRes[k] \times \exp(\beta \ast i, k)

To use sign restrictions on the coefficients, SignRes must be an \text{nvar\times 1} vector containing values of either 0, -1, or 1. The value 0 means there is no sign restriction, -1 ensures that the coefficient is negative, and 1 ensures that the coefficient is positive. For example, if SignRes = c(\emptyset, 1, -1), the first coefficient is unconstrained, the second will be positive, and the third will be negative.

The sign restriction is implemented such that if the the k’th \beta has a non-zero sign restriction (i.e., it is constrained), we have \beta_k = SignRes[k] \times \exp(\beta \ast k).

The sign restrictions (if used) will be reflected in the betaDdraw output. However, the unconstrained mixture components are available in nmix. Important: Note that draws from nmix are distributed according to the mixture of normals but not the coefficients in betaDdraw.

Care should be taken when selecting priors on any sign restricted coefficients. See related vignette for additional information.

nmix Details: nmix is a list with 3 components. Several functions in the bayesm package that involve a Dirichlet Process or mixture-of-normals return nmix. Across these functions, a common structure is used for nmix in order to utilize generic summary and plotting functions.

probdraw: \text{ncomp\times R\times keep} matrix that reports the probability that each draw came from a particular component
zdraw: \text{R\times keep\times nobs} matrix that indicates which component each draw is assigned to (here, null)
compdraw: A list of R\times keep lists of ncomp lists. Each of the inner-most lists has 2 elements: a vector of draws for mu and a

Value

A list containing:

Deltadraw \text{R\times keep\times nz \times nvar} matrix of draws of Delta, first row is initial value
betadraw \text{nlgtxnvar\times R\times keep} array of beta draws
**rhierMnlRwMixture**

- `rmix`: a list containing: `probdraw`, `zdraw`, `compdraw` (see “`rmix` Details” section)
- `loglike`: \( \bar{R}/\text{keep} \) vector of log-likelihood for each kept draw
- `SignRes`: \( nvar \times 1 \) vector of sign restrictions

**Note**

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.

\[
\text{like}_i^{(1-w)} x \text{like}_\text{pooled}^{(\frac{n_i}{N}+w)}
\]

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

**Author(s)**

Peter Rossi, Anderson School, UCLA, <perossi@gmail.com>.

**References**

For further discussion, see Chapter 5, *Bayesian Statistics and Marketing* by Rossi, Allenby, and McCulloch.

[http://www.prossi.org/home/bsm-1](http://www.prossi.org/home/bsm-1)

**See Also**

`rmnlIndepMetrop`

**Examples**

```r
if(nchar(Sys.getenv("LONG_TEST")) != 0) {R=100000} else (R=10)
sset.seed(66)

p = 3  # num of choice alterns
ncoef = 3
nlgt = 300  # num of cross sectional units
nz = 2
Z = matrix(runif(nz*nlt),ncol=nz)
Z = t(t(Z) - apply(Z,2,mean))  # demean Z
ncomp = 3  # num of mixture components
Delta = matrix(c(1,0,1,0,1,2),ncol=2)
comp=NUNLL
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(0.4, 0.2, 0.4)

## simulate from MNL model conditional on X matrix
simmnlwX= function(n,X,beta) {
  k = length(bbeta)
  Xbeta = X%*%beta
```
```r
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){
  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
Mmc1 = list(R=R, keep=keep)
Data1 = list(p=p, lgtdata=simlgtdata, Z=Z)

## fit model without sign constraints
out1 = rhierMnlRwMixture(Data=Data1, Prior=Prior1, Mcmc=Mmc1)

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)

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

rhierNegbinRw

MCMC Algorithm for Hierarchical Negative Binomial Regression

Description

rhierNegbinRw implements an MCMC algorithm 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, nprint, s_beta, s_alpha, alpha, c, Vbeta0, Delta0) 

Details

Model and Priors: \( y_i \sim \text{NBD}(\text{mean}=\lambda, \text{over-dispersion}=\alpha) \)

\( \lambda = \exp(X_i \beta_i) \)

\( \beta_i \sim N(\Delta z_i, V_{beta}) \)

\( \text{vec}(\Delta|V_{beta}) \sim N(\text{vec}(\text{Deltabar}), V_{beta}(x)\text{Adelta}) \)

\( V_{beta} \sim IW(nu, V) \)

\( \alpha \sim Gamma(a, b) \) (unless Mcmc$alpha specified)

Note: prior mean of alpha = a/b, variance = a/(b^2)

Argument Details: Data = list(regdata, Z) [Z optional]
regdata: list of lists with data on each of nreg units
regdata[[i]]$X: nobs$xvar matrix of X variables
regdata[[i]]$y: nobs$x1 vector of count responses
Z: nreg$xz matrix of unit characteristics (def: vector of ones)

Prior = list(Deltabar, Adelta, nu, V, a, b) [optional]

Deltabar: nzxnvar prior mean matrix (def: 0)
Adelta: nzxnz PDS prior precision matrix (def: 0.01*I)
nu: d.f. parameter for Inverted Wishart prior (def: nvar+3)
V: location matrix of Inverted Wishart prior (def: nu*I)
a: Gamma prior parameter (def: 0.5)
b: Gamma prior parameter (def: 0.1)

Mcmc = list(R, keep, nprint, s_beta, s_alpha, alpha, c, Vbeta0, Delta0) [only R required]

R: number of MCMC draws
keep: MCMC thinning parameter – keep every keep'th draw (def: 1)
nprint: print the estimated time remaining for every nprint'th draw (def: 100, set to 0 for no print)
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)
alpha: over-dispersion parameter (def: alpha ~ Gamma(a,b))
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/keepx1 vector of values of log-likelihood
betadraw: nreg$xvarxR/keep array of beta draws
alphadraw: R/keepx1 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 Narayanan (Stanford GSB) and Peter Rossi (Anderson School, UCLA), <perossichi@gmail.com>.
References

For further discussion, see Chapter 5, Bayesian Statistics and Marketing by Rossi, Allenby, and McCulloch.

http://www.perossi.org/home/bsm-1

See Also

rnegbinRw

Examples

if(nchar(Sys.getenv("LONG_TEST")) != 0) {R=2000} else {R=10}
set.seed(66)

# Simulate from the Negative Binomial Regression
simnegbin = function(X, beta, alpha) {
  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)
Mmc1 = list(R=R)

out = rhiernegbinRw(Data=Data1, Mmc=Mmc1)

cat("Summary of Delta draws", fill=TRUE)
summary(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 equations implemented with a 
Dirichlet Process prior.

Usage

rivDP(Data, Prior, Mcmc)

Arguments

Data: list(y, x, w, z)
Prior: list(md, Ad, mbg, Abg, lambda, Prioralpha, lambda_hyper)
Mcmc: list(R, keep, nprint, maxuniq, SCALE, gridsize)

Details

Model and Priors: 

\[ \begin{align*} 
\mathbf{x} &= \mathbf{z}'\delta + \mathbf{e} \\
y &= \beta \ast \mathbf{x} + \mathbf{w}'\gamma + \mathbf{e} \\
\mathbf{e} &\sim N(\mathbf{0}) \\
\delta &\sim N(\mathbf{md}, \mathbf{Ad}^{-1}) \\
\mathbf{vec}(\beta, \gamma) &\sim N(\mathbf{mbg}, \mathbf{Abg}^{-1}) \\
\theta_i &\sim G \\
G &\sim DP(\alpha, G_0) \\
\alpha &\sim (1 - (\alpha - \alpha_{\text{min}})/(\alpha_{\text{max}} - \alpha_{\text{min}}))^{\text{power}} 
\end{align*} \]

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.

\[ \begin{align*} 
\delta &\sim N(\mathbf{md}, \mathbf{Ad}^{-1}) \\
\mathbf{vec}(\beta, \gamma) &\sim N(\mathbf{mbg}, \mathbf{Abg}^{-1}) \\
\theta_i &\sim G \\
G &\sim DP(\alpha, G_0) \\
\alpha &\sim (1 - (\alpha - \alpha_{\text{min}})/(\alpha_{\text{max}} - \alpha_{\text{min}}))^{\text{power}} 
\end{align*} \]

where \( \alpha_{\text{min}} \) and \( \alpha_{\text{max}} \) 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.
$G_0$ is the natural conjugate prior for $(\mu, \Sigma)$: $\Sigma \sim IW(\nu, vI)$ and $\mu | \Sigma \sim N(0, \Sigma(x)a^{-1})$

These parameters are collected together in the list $\lambda$. It is highly recommended that you use the default settings for these hyper-parameters.

$\lambda(a, \nu, v)$:

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

**Argument Details:** Data = list($y$, $x$, $w$, $z$)

- $y$: $nx1$ vector of obs on LHS variable in structural equation
- $x$: $nx1$ vector of obs on "endogenous" variable in structural equation
- $w$: $nxj$ matrix of obs on "exogenous" variables in the structural equation
- $z$: $nxp$ matrix of obs on instruments

Prior = list($md$, $Ad$, $mbg$, $Abg$, $lambda$, $Prioralpha$, $lambda\_hyper$) [optional]

- $md$: $p$-length prior mean of delta (def: 0)
- $Ad$: $pxp$ PDS prior precision matrix for prior on delta (def: 0.01*I)
- $mbg$: $(j+1)$-length prior mean vector for prior on beta,gamma (def: 0)
- $Abg$: $(j+1)x(j+1)$ PDS prior precision matrix for prior on beta,gamma (def: 0.01*I)
- $Prioralpha$: list($Istarmin$, $Istarmax$, $power$)
- $Istarmin$: is expected number of components at lower bound of support of alpha (def: 1)
- $Istarmax$: is expected number of components at upper bound of support of alpha (def: floor(0.1*length($y$)))
- $power$: is the power parameter for alpha prior (def: 0.8)
- $lambda\_hyper$: list($alim$, $nulim$, $vlim$)
- $alim$: defines support of a distribution (def: c(0.01, 10))
- $nulim$: defines support of nu distribution (def: c(0.01, 3))
- $vlim$: defines support of v distribution (def: c(0.1, 4))

Mcmc = list($R$, $keep$, $nprint$, $maxuniq$, $SCALE$, $gridsize$) [only $R$ required]

- $R$: number of MCMC draws
- $keep$: MCMC thinning parameter: keep every keepth draw (def: 1)
- $nprint$: print the estimated time remaining for every nprint’th draw (def: 100, set to 0 for no print)
- $maxuniq$: storage constraint on the number of unique components (def: 200)
- $SCALE$: scale data (def: TRUE)
- $gridsize$: gridsize parameter for alpha draws (def: 20)

**nmix Details:** $nmix$ is a list with 3 components. Several functions in the bayesm package that involve a Dirichlet Process or mixture-of-normals return $nmix$. Across these functions, a common structure is used for $nmix$ in order to utilize generic summary and plotting functions.

- $probdraw$: $necomp|xR/keep$ matrix that reports the probability that each draw came from a particular component (here, a one-column matrix of 1s)
- $zdraw$: $R/keepxnxobs$ matrix that indicates which component each draw is assigned to (here, null)
- $compdraw$: A list of $R/keep$ lists of $necomp$ lists. Each of the inner-most lists has 2 elmenes: a vector of draws for $mu$ and a
Value
A list containing:

- `deltadraw` \( R/\text{keepxp} \) array of delta draws
- `betadraw` \( R/\text{keepx} \) vector of beta draws
- `alphadraw` \( R/\text{keepx} \) vector of draws of Dirichlet Process tightness parameter
- `istardraw` \( R/\text{keepx} \) vector of draws of the number of unique normal components
- `gammadraw` \( R/\text{keepxj} \) array of gamma draws
- `nmix` a list containing: `probdraw`, `zdraw`, `compdraw` (see “nmix Details” section)

Author(s)
Peter Rossi, Anderson School, UCLA, <perossichi@gmail.com>.

References
See also, Chapter 4, *Bayesian Non- and Semi-parametric Methods and Applications* by Peter Rossi.

See Also
rivGibbs

Examples
```r
if(nchar(Sys.getenv("LONG_TEST")) != 0) {R=2000} else {R=10}
set.seed(66)

## simulate scaled log-normal errors and run
k = 10
delta = 1.5
Sigma = matrix(c(1, 0.6, 0.6, 1), ncol=2)
N = 1000
tbeta = 4
scalefactor = 0.6
root = chol(scalefactor*Sigma)
mu = c(1,1)

## compute interquartile ranges
ninterq = qnorm(0.75) - qnorm(0.25)
error = matrix(rnorm(100000*2), ncol=2)%*%root
terror = t(t(error)+mu)
Err = t(t(exp(error))-exp(mu+0.5*scalefactor*diag(Sigma)))
lnNinterq = quantile(Err[,1], prob=0.75) - quantile(Err[,1], prob=0.25)

## simulate data
error = matrix(rnorm(N*2), ncol=2)%*%root
terror = t(t(error)+mu)
```
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(y, x, w, z)

Prior  
list(md, Ad, mbg, Abg, nu, V)

Mcmc  
list(R, keep, nprint)
Details

Model and Priors:  
\[ x = z\delta + e_1 \]
\[ y = \beta^* x + w^\gamma + e_2 \]
\[ e_1, e_2 \sim N(0, \Sigma) \]
Note: if intercepts are desired in either equation, include vector of ones in \( z \) or \( w \)
\[ \delta \sim N(md, Ad^{-1}) \]
\[ vec(\beta, \gamma) \sim N(mbg, Abg^{-1}) \]
\[ \Sigma \sim IW(nu, V) \]

Argument Details:  
Data = list(y, x, w, z)

- \( y \): \( nx1 \) vector of obs on LHS variable in structural equation
- \( x \): \( nx1 \) vector of obs on "endogenous" variable in structural equation
- \( w \): \( nxj \) matrix of obs on "exogenous" variables in the structural equation
- \( z \): \( nxp \) matrix of obs on instruments

Prior = list(md, Ad, mbg, Abg, nu, V) [optional]

- \( md \): \( p \)-length prior mean of delta (def: 0)
- \( Ad \): \( pxp \) PDS prior precision matrix for prior on delta (def: 0.01*I)
- \( mbg \): \( (j + 1) \)-length prior mean vector for prior on beta, gamma (def: 0)
- \( Abg \): \( (j + 1)x(j + 1) \) PDS prior precision matrix for prior on beta, gamma (def: 0.01*I)
- \( nu \): d.f. parameter for Inverted Wishart prior on Sigma (def: 5)
- \( V \): \( 2x2 \) location matrix for Inverted Wishart prior on Sigma (def: nu*I)

Mcmc = list(R, keep, nprint) [only R required]

- \( R \): number of MCMC draws
- \( keep \): MCMC thinning parameter: keep every keepth draw (def: 1)
- \( nprint \): print the estimated time remaining for every nprint'th draw (def: 100, set to 0 for no print)

Value

A list containing:
- \( \text{deltadraw} \): \( R/keepxp \) matrix of delta draws
- \( \text{betadraw} \): \( R/keepx1 \) vector of beta draws
- \( \text{gammadraw} \): \( R/keepxj \) matrix of gamma draws
- \( \text{sigmadraw} \): \( R/keepx4 \) matrix of Sigma draws – each row is the vector form of Sigma

Author(s)

Rob McCulloch and Peter Rossi, Anderson School, UCLA, <perossichi@gmail.com>.

References

For further discussion, see Chapter 5, *Bayesian Statistics and Marketing* by Rossi, Allenby, and McCulloch.
Examples

```r
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 = 0.8
Sigma = matrix(c(1,rho,rho,1), ncol=2)
delta = c(1,4)
beta = 0.5
gamma = c(1)
simiv = simIV(delta, beta, Sigma, n, z, w, gamma)

Data1 = list(); Data1$z = z; Data1$w=w; Data1$x=simiv$x; Data1$y=simiv$y
Mcmc1=list(); 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)]))

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

Description

`rmixGibbs` makes one draw using the Gibbs Sampler for a mixture of multivariate normals. `rmixGibbs` is not designed to be called directly. Instead, use `rnmixGibbs` wrapper function.

Usage

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

- **y**: data array where 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 prior
- **a**: Dirichlet prior parms
- **p**: prior prob of each mixture component
- **z**: component identities for each observation – "indicators"

Value

- a list containing:
  - **p**: draw of 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 (Arizona State University) and Peter Rossi (Anderson School, UCLA), <perossichi@gmail.com>.

References

For further discussion, see Chapter 5 *Bayesian Statistics and Marketing* by Rossi, Allenby, and McCulloch.
http://www.perossi.org/home/bsm-1

See Also

rnmixGibbs
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 with ncomp = length(pvec).
comps[[j]][[1]] is mean vector for the jth component.
comps[[j]][[2]] is the inverse of the cholesky root of Σ for jth component

Value

A list containing:

x: an n x length(comps[[1]][[1]]) array of iid draws
z: an 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, Anderson School, UCLA, <perossichi@gmail.com>.

See Also

rnmixGibbs
**rmnlIndepMetrop**  
*MCMC Algorithm for Multinomial Logit Model*

**Description**

`rmnlIndepMetrop` implements Independence Metropolis algorithm for the multinomial logit (MNL) model.

**Usage**

`rmnlIndepMetrop(Data, Prior, Mcmc)`

**Arguments**

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

**Details**

**Model and Priors:** $y \sim \text{MNL}(X, \beta)$

$Pr(y = j) = \frac{\exp(x_j^T \beta)}{\sum_k \exp(x_k^T \beta)}$  

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

**Argument Details:**

- **Data**  
  - `y`: $n \times 1$ vector of multinomial outcomes (1, ..., p)  
  - `X`: $n \times pxk$ matrix  
  - `p`: number of alternatives

- **Prior**  
  - `A`: $k \times k$ prior precision matrix (def: 0.01*I)  
  - `betabar`: $k \times 1$ prior mean (def: 0)

- **Mcmc**  
  - `R`: number of MCMC draws  
  - `keep`: MCMC thinning parameter – keep every `keep`th draw (def: 1)  
  - `nprint`: print the estimated time remaining for every `nprint`th draw (def: 100, set to 0 for no print)  
  - `nu`: d.f. parameter for independent t density (def: 6)

**Value**

A list containing:

- `betadraw`: $R/keep \times pxk$ matrix of beta draws
loglike \( R/\)keepe1 vector of log-likelihood values evaluated at each draw
acceptr acceptance rate of Metropolis draws

Author(s)

Peter Rossi, Anderson School, UCLA, <perossichi@gmail.com>.

References

For further discussion, see Chapter 3, *Bayesian Statistics and Marketing* by Rossi, Allenby, and McCulloch.
http://www.perossi.org/home/bsm-1

See Also

* rhierMnlRwMixture

Examples

```r
if(nchar(Sys.getenv("LONG_TEST")) != 0) {R=2000} else {R=10}
set.seed(66)

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))
}

n = 200
p = 3
beta = c(1, -1, 1.5, 0.5)

simout = simmnl(p,n,beta)
```
**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(y, X, p)`
  - `
y: nx1 vector of multinomial outcomes (1, ..., p)
  
- **Prior**
  - `list(betabar, A, nu, V)`
  - `
  
  - `betabar`: mean vector
  - `A`: precision matrix
  - `nu`: degrees of freedom
  - `V`: scale matrix

- **Mcmc**
  - `list(R, keep, nprint, beta0, sigma0)`
  - `
  
  - `R`: number of iterations
  - `keep`: number of iterations to keep
  - `nprint`: number of iterations to print
  - `beta0`: initial value for beta
  - `sigma0`: initial value for sigma

**Details**

**Model and Priors:**

\[ w_i = X_i \beta + e \] with \( e \sim N(0, \Sigma) \). Note: \(w_i\) and \(e\) are \((p - 1) \times 1\).

\[ y_i = j \text{ if } w_{ij} > \max(0, w_{i,-j}) \text{ for } j = 1, \ldots, p-1 \text{ and where } w_{i,-j} \text{ means elements of } w_i \text{ other than the } j\text{th}. \]

\[ y_i = p, \text{ if all } w_i < 0 \]

\( \beta \) is not identified. However, \( \beta/\sqrt{\sigma_{11}} \) and \( \Sigma/\sigma_{11} \) are identified. See reference or example below for details.

\[ \beta \sim N(\text{betabar}, A^{-1}) \]

\[ \Sigma \sim IW(\text{nu}, V) \]

**Argument Details:**

- **Data** = `list(y, X, p)`
  - `y`: \(nx1\) vector of multinomial outcomes (1, ..., p)
  - `X`: \(n \times (p - 1) \times k\) design matrix. To make \(X\) matrix use `createX` with `DIFF=TRUE`
  - `p`: number of alternatives

- **Prior** = `list(betabar, A, nu, V)` [optional]
betabar: \( k \times 1 \) prior mean (def: 0)
A: \( k \times k \) prior precision matrix (def: 0.01*I)
u: d.f. parameter for Inverted Wishart prior (def: (p-1)+3)
V: PDS location parameter for Inverted Wishart prior (def: nu*I)

\[
\text{Mcmc} = \text{list}(R, \text{keep}, \text{nprint}, \text{beta0}, \text{sigma0}) \quad \text{[only R required]}
\]

R: number of MCMC draws
keep: MCMC thinning parameter – keep every \( k \)-th draw (def: 1)
nprint: print the estimated time remaining for every \( n \)-th draw (def: 100, set to 0 for no print)

beta0: initial value for beta (def: 0)
sigma0: initial value for sigma (def: I)

Value
A list containing:

- betadraw \( R/\text{keep} \times k \) matrix of betadraws
- sigmadraw \( R/\text{keep} \times (p-1) \times (p-1) \) matrix of sigma draws – each row is the vector form of sigma

Author(s)
Peter Rossi, Anderson School, UCLA, <perossichi@gmail.com>.

References
For further discussion, see Chapter 4, *Bayesian Statistics and Marketing* by Rossi, Allenby, and McCulloch.
http://www.perossi.org/home/bsm-1

See Also
rmvpgibbs

Examples
if(nchar(Sys.getenv("LONG_TEST")) != 0) (R=2000) else (R=10)
set.seed(66)

simnmp = 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)) + Xbeta
  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))
}
rmultireg

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

Model:
\[ Y = XB + U \] with \( \text{cov}(u_i) = \Sigma \)
\( B \) is \( k \times m \) matrix of coefficients; \( \Sigma \) is \( m \times m \) covariance matrix.

Priors:
\[ \beta \mid \Sigma \sim N(\text{betabar}, \Sigma(x)A^{-1}) \]
\( \text{betabar} = \text{vec}(B\text{bar}); \beta = \text{vec}(B) \)
\[ \Sigma \sim \text{IW}(\nu, V) \]

Value

A list of the components of a draw from the posterior

- \( \beta \): 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, Anderson School, UCLA, <perossichi@gmail.com>.

References

For further discussion, see Chapter 2, *Bayesian Statistics and Marketing* by Rossi, Allenby, and McCulloch.

\[ \text{http://www.perossi.org/home/bsm-1} \]

Examples

```r
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, 0.5, 0.5, 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(0.01,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(0.01, 0.05, 0.5, 0.95, 0.99))
mat = rbind(as.vector(b), mat)
rownames(mat)[1] = "beta"
print(mat)

cat(" Sigma draws ", fill=TRUE)
mat = apply(Sigmadraw, 2, quantile, probs=c(0.01, 0.05, 0.5, 0.95, 0.99))
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(y, X, p)
- **Prior**
  - list(betabar, A, nu, V)
- **Mcmc**
  - list(R, keep, nprint, beta0, sigma0)

**Details**

**Model and Priors:** \( w_i = X_i \beta + e \) with \( e \sim N(0, \Sigma) \). Note: \( w_i \) is \( px1 \).
\( y_{ij} = 1 \) if \( w_{ij} > 0 \), else \( y_i = 0 \). \( j = 1, \ldots, p \)

beta and Sigma are not identified. Correlation matrix and the betas divided by the appropriate standard deviation are. See reference or example below for details.

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

To make \( X \) matrix use createX
Argument Details: Data = list(y, X, p)

X: n * pxk Design Matrix
y: n * px1 vector of 0/1 outcomes
p: dimension of multivariate probit

Prior = list(betabar, A, nu, V) [optional]

betabar: kx1 prior mean (def: 0)
A: kxk prior precision matrix (def: 0.01*I)
u: d.f. parameter for Inverted Wishart prior (def: (p-1)+3)
V: PDS location parameter for Inverted Wishart prior (def: nu*I)

Mcmc = list(R, keep, nprint, beta0, sigma0) [only R required]

R: number of MCMC draws
keep: MCMC thinning parameter – keep every keepth draw (def: 1)
nprint: print the estimated time remaining for every nprint’th draw (def: 100, set to 0 for no print)
beta0: initial value for beta
sigma0: initial value for sigma

Value
A list containing:

betadraw R/keepxk matrix of betadraws
sigmadraw R/keepxkp * p matrix of sigma draws – each row is the vector form of sigma

Author(s)
Peter Rossi, Anderson School, UCLA, <perossichi@gmail.com>.

References
For further discussion, see Chapter 4, Bayesian Statistics and Marketing by Rossi, Allenby, and McCulloch.
http://www.perossi.org/home/bsm-1

See Also
rmnpGibbs

Examples

if(nchar(Sys.getenv("LONG_TEST")) != 0) {R=2000} else {R=10}
set.seed(66)
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)
rmvst

\hspace{10pt} \textbf{Description} \\
\hspace{15pt} \textit{rmvst} draws from a multivariate student-t distribution.

\hspace{10pt} \textbf{Usage} \\
\hspace{15pt} \texttt{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, Anderson School, UCLA, <perossichi@gmail.com>.

References

For further discussion, see Bayesian Statistics and Marketing by Rossi, Allenby, and McCulloch.
http://www.perossi.org/home/bsm-1

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 where $\beta | \alpha$ 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, nprint, s_beta, s_alpha, beta0, alpha)

Details

Model and Priors: \( y \sim NBD(\text{mean} = \lambda, \text{dispersion} = \alpha) \)
\( \lambda = \exp(x'\beta) \)
\( \beta \sim N(\text{betabar}, A^{-1}) \)
\( \alpha \sim \text{Gamma}(a,b) \) (unless \( \text{Mcmc}\alpha \) specified)
Note: prior mean of \( \alpha = a/b \), variance = \( a/(b^2) \)

Argument Details: Data = list(y, X)

\( y: \) nx1 vector of counts (0, 1, 2, ...)
\( X: \) nxk design matrix

Prior = list(betabar, A, a, b) [optional]

betabar: kx1 prior mean (def: 0)
A: kxk PDS prior precision matrix (def: 0.01*I)
a: Gamma prior parameter (not used if \( \text{Mcmc}\alpha \) specified) (def: 0.5)
b: Gamma prior parameter (not used if \( \text{Mcmc}\alpha \) specified) (def: 0.1)

Mcmc = list(R, keep, nprint, s_beta, s_alpha, beta0, alpha) [only R required]

R: number of MCMC draws
keep: MCMC thinning parameter – keep every keepth draw (def: 1)
nprint: print the estimated time remaining for every nprint’th draw (def: 100, set to 0 for no print)
s_beta: scaling for beta | alpha RW inc cov matrix (def: 2.93/sqrt(k))
s_alpha: scaling for alpha | beta RW inc cov matrix (def: 2.93)
a: over-dispersion parameter (def: \( \alpha \sim \text{Gamma}(a,b) \))

Value

A list containing:

betadraw \( R/keepxk \) matrix of beta draws
alphadraw \( R/keepx1 \) vector of alpha draws
llike \( R/keepx1 \) 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 Narayanan (Stanford GSB) and Peter Rossi (Anderson School, UCLA).<perossichi@gmail.com>.

References
For further discussion, see Bayesian Statistics and Marketing by Rossi, Allenby, McCulloch. http://www.perossi.org/home/bsm-1

See Also
rhierNegbinRw

Examples
if(nchar(Sys.getenv("LONG_TEST")) != 0) {R=R000} else {R=R10}
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=list(R=R))

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

Mcmc

list(R, keep, nprint, Loglike)

Details

Model and Priors:

- \( y_i \sim N(\mu_{ind}, \Sigma_{ind}) \)
- \( \text{ind} \sim \text{iid multinomial}(p) \) with \( p \) an \( ncomp \times 1 \) vector of probs
- \( \mu_j \sim N(mubar, \Sigma_j(x)A^{-1}) \) with \( mubar = \text{vec}(Mubar) \)
- \( \Sigma_j \sim \text{IW}(nu, V) \)

Note: this is the natural conjugate prior – a special case of multivariate regression

\( p \sim \text{Dirichlet}(a) \)

Argument Details: Data = list(y)

- \( y \): \( nxk \) matrix of data (rows are obs)

Prior = list(Mubar, A, nu, V, a, ncomp) [only ncomp required]

- Mubar: \( 1xk \) vector with prior mean of normal component means (def: 0)
- A: \( 1x1 \) precision parameter for prior on mean of normal component (def: 0.01)
- nu: d.f. parameter for prior on Sigma (normal component cov matrix) (def: k+3)
- V: \( kxk \) location matrix of IW prior on Sigma (def: nu*I)
- a: \( ncomp \times 1 \) vector of Dirichlet prior parameters (def: rep(5, ncomp))
- ncomp: number of normal components to be included

Mcmc = list(R, keep, nprint, Loglike) [only R required]

- R: number of MCMC draws
**nmixGibbs**

keep: MCMC thinning parameter – keep every keepth draw (def: 1)
nprint: print the estimated time remaining for every nprint’th draw (def: 100, set to 0 for no print)
LogLike: logical flag for whether to compute the log-likelihood (def: FALSE)

**Details:** `nmix` is a list with 3 components. Several functions in the `bayesm` package that involve a Dirichlet Process or mixture-of-normals return `nmix`. Across these functions, a common structure is used for `nmix` in order to utilize generic summary and plotting functions.

**probdraw:** `ncomp x R/keep` matrix that reports the probability that each draw came from a particular component
**zdraw:** `R/keepxnobs` matrix that indicates which component each draw is assigned to
**compdraw:** A list of `R/keep` lists of `ncomp` lists. Each of the inner-most lists has 2 elemens: a vector of draws for `mu` and a

**Value**

A list containing:

1. `R/keepx1` vector of log-likelihood values
2. `nmix` a list containing: `probdraw`, `zdraw`, `compdraw` (see “nmix Details” section)

**Note**

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 chapter 5 of Rossi et al below for details.

Use `emixMargDen` or `momMix` to compute posterior expectation or distribution of various unidentified parameters.

**Author(s)**

Peter Rossi, Anderson School, UCLA, <perossichi@gmail.com>.

**References**

For further discussion, see Chapter 3, *Bayesian Statistics and Marketing* by Rossi, Allenby, and McCulloch.

http://www.perossi.org/home/bsm-1

**See Also**

`rmixture`, `rmixGibbs`, `emixMargDen`, `momMix`, `mixDen`, `mixDenBi`

**Examples**

```r
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)

# change to "root" scale
comps = list()
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 = nmixGibbs(Data=Data1, Prior=Prior1, Mcmc=Mcmc1)

cat("Summary of Normal Mixture Distribution", fill=TRUE)
summary(out$nmix)

tmom = momMix(matrix(pvec,nrow=1), list(comps))
mat = rbind(tmom$mu, tmom$sd)
cat(" True Mean/Std Dev", fill=TRUE)
print(mat)

## plotting examples
if(0) (plot(out$nmix, Data=dm$x))

---

rordprobitGibbs  

Gibbs Sampler for Ordered Probit

Description

rordprobitGibbs implements a Gibbs Sampler for the ordered probit model with a RW Metropolis step for the cut-offs.

Usage

rordprobitGibbs(Data, Prior, Mcmc)

Arguments

Data  
Prior  
Mcmc

list(y, X, k)
list(betabar, A, dstarbar, Ad)
list(R, keep, nprint, s)
Details

Model and Priors: \[ z = X\beta + e \text{ with } e \sim N(0, I) \]
y = k if \( c[k] \leq z < c[k+1] \) with \( k = 1, \ldots, K \)
cutoffs = \{c[1], \ldots, c[K+1]\}
\( \beta \sim N(\text{betabar}, A^{-1}) \)
dstar \( \sim N(\text{dstarbar}, A_d^{-1}) \)

Be careful in assessing prior parameter \( A_d \): 0.1 is too small for many applications.

Argument Details: Data = list(y, X, k)
\[ y: \text{n} \times 1 \text{ vector of observations, } (1, \ldots, k) \]
\[ X: \text{n} \times p \text{ Design Matrix} \]
\[ k: \text{the largest possible value of } y \]

Prior = list(\text{betabar, A, dstarbar, A_d}) \text{ [optional]} \]
\[ \text{betabar}: \text{p} \times 1 \text{ prior mean (def: 0)} \]
\[ A: \text{p} \times \text{p} \text{ prior precision matrix (def: 0.01*I)} \]
\[ \text{dstarbar}: \text{n} \times 1 \text{ prior mean, where } \text{n}\text{dstar} = k - 2 \text{ (def: 0)} \]
\[ A_d: \text{n} \times \text{ndstar} \times \text{ndstar} \text{ prior precision matrix (def: I)} \]

Mcmc = list(R, keep, nprint, s) \text{ [only R required]} \]
\[ R: \text{number of MCMC draws} \]
\[ \text{keep}: \text{MCMC thinning parameter – keep every keepth draw (def: 1)} \]
\[ \text{nprint}: \text{print the estimated time remaining for every nprint’th draw (def: 100, set to 0 for no print)} \]
\[ s: \text{scaling parameter for RW Metropolis (def: } 2.93/\sqrt{p}) \]

Value

A list containing:
\[ \text{betadraw} \text{ R/keepxp matrix of betadraws} \]
\[ \text{cutdraw} \text{ R/keepxp(k - 1) matrix of cutdraws} \]
\[ \text{dstardraw} \text{ R/keepxp(k - 2) matrix of dstardraws} \]
\[ \text{accept} \text{ acceptance rate of Metropolis draws for cut-offs} \]

Note

set \( c[1] = -100 \) and \( c[K+1] = 100 \). \( c[2] \) is set to 0 for identification.

The relationship between cut-offs and dstar is:
\[ c[3] = \exp(dstar[1]), \]
\[ c[4] = c[3] + \exp(dstar[2]), \ldots, \]
\[ c[K] = c[K-1] + \exp(dstar[K-2]) \]
Author(s)

Peter Rossi, Anderson School, UCLA, <perossichi@gmail.com>.

References

Bayesian Statistics and Marketing by Rossi, Allenby, and McCulloch
http://www.perossi.org/home/bsm-1

See Also

rbprobitGibbs

Examples

```r
if(nchar(Sys.getenv("LONG_TEST")) != 0) {R=2000} else {R=10}
set.seed(66)

## 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 ))
}
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 Mmc for ordered probit model
Mmc = list(R=R)
out = rordprobitGibbs(Data=Data, Mmc=Mmc)

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])

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

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

Usage

rscaleUsage(Data, Prior, Mcmc)

Arguments

Data list(x, k)
Prior list(nu, V, mubar, Am, gs, Lambdanu, LambdaV)
Mcmc list(R, keep, nprint, ndghk, e, y, mu, Sigma, sigma, tau, Lambda)

Details

Model and Priors: $n = \text{ncol}(x)$ individuals respond to $p = \text{ncol}(x)$ questions; all questions are on a scale $1, \ldots, k$ for respondent $i$ and question $j$.

$x_{ij} = d$ if $c_{d-1} \leq y_{ij} \leq c_d$ where $d = 1, \ldots, k$ and $c_d = a + bd + cd^2$

$y_i = \mu + \tau_i \ast \iota + \sigma_{ma} \ast z_i$ with $z_i \sim N(0, \Sigma)$
$(\tau_i, \ln(\sigma_{ma})) \sim N(\phi, \Lambda)$
$\phi = (0, \text{Lambda}_{22})$
$\mu \sim N(\text{mubar}, A^{-1})$
$\Sigma \sim IW(\nu, V)$
$\Lambda \sim IW(\text{Lambdanu}, \text{LambdaV})$
$e \sim \text{unif on a grid}$

It is highly recommended that the user choose the default prior settings. If you wish to change prior settings and/or the grids used, please carefully read the case study listed in the reference below.

Argument Details: Data = list(x, k)

x: nxp matrix of discrete responses
k: number of discrete rating scale options

Prior = list(nu, V, mubar, Am, gs, Lambdanu, LambdaV) {optional}

nu: d.f. parameter for Sigma prior (def: $p + 3$)
V: scale location matrix for Sigma prior (def: nu*I)
mubar: \( px1 \) vector of prior means (def: \( \text{rep}(k/2,p) \))
Am: \( pxp \) prior precision matrix (def: 0.01*I)
gs: grid size for sigma (def: 100)
Lambdanu: d.f. parameter for Lambda prior (def: 20)
LambdaV: scale location matrix for Lambda prior (def: \((\text{Lambdanu} - 3)\ast \text{Lambda})

Mmc = list(R, keep, nprint, ndghk, e, y, mu, Sigma, sigma, tau, Lambda) \[\text{only \ required}\]

R: number of MCMC draws (def: 1000)
keep: MCMC thinning parameter – keep every keepth draw (def: 1)
nprint: print the estimated time remaining for every nprint’th draw (def: 100, set to 0 for no print)
ndghk: number of draws for a GHK integration (def: 100)
e: initial value (def: 0)
y: initial values (def: x)
mu: initial values (def: \( \text{apply}(y,2,\text{mean}) \), a p-length vector)
Sigma: initial value (def: \( \text{var}(y) \))
sigma: initial values (def: \( \text{rep}(1,n) \))
tau: initial values (def: \( \text{rep}(0,n) \))
Lambda: initial values (def: \( \text{matrix}(c(4,0,0,.5),ncol=2) \))

Value
A list containing:

\( \text{Sigmadraw} \quad R/\text{keepxp} \ast p \) matrix of Sigma draws – each row is the vector form of Sigma
\( \text{mudraw} \quad R/\text{keepxp} \) matrix of mu draws
\( \text{taudraw} \quad R/\text{keepxn} \) matrix of tau draws
\( \text{sigmadraw} \quad R/\text{keepxn} \) matrix of sigma draws
\( \text{Lambdadraw} \quad R/\text{keepx4} \) matrix of Lambda draws
\( \text{edraw} \quad R/\text{keepx1} \) vector of e draws

Warning
\( \text{tau, sigma,} \) are identified from the scale usage patterns in the \( p \) questions asked per respondent (\# cols of \( x \)). Do not attempt to use this on datasets with only a small number of total questions.

Author(s)
Rob McCulloch (Arizona State University) and Peter Rossi (Anderson School, UCLA), <perossichi@gmail.com>.

References
For further discussion, see Case Study 3 on Overcoming Scale Usage Heterogeneity, Bayesian Statistics and Marketing by Rossi, Allenby, and McCulloch.
http://www.perossi.org/home/bsm-1
rsurGibbs

**Examples**

```r
if(nchar(Sys.getenv("LONG_TEST")) != 0) {R=1000} else {R=5}
set.seed(66)

data(customerSat)
surveydat = list(k=10, x=as.matrix(customerSat))

out = rscaleUsage(Data=surveydat, Mcmc=list(R=R))
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**

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

**Arguments**

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

**Details**

**Model and Priors:** $y_i = X_i \beta_i + e_i$ with $i = 1, \ldots, m$ for $m$ regressions

$e(k,1) \ldots e(k,m))^T \sim N(0, \Sigma)$ with $k = 1, \ldots, n$

Can be written as a stacked model:

$y = X \beta + e$ where $y$ is a $\text{nobs} \times m$ vector and $p = \text{length(beta)} = \text{sum(length(beta_i))}$

Note: must have the same number of observations ($n$) in each equation but can have a different number of $X$ variables ($p_i$) for each equation where $p = \sum p_i$.

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

**Argument Details:** Data = list(regdata)

- **regdata:** list of lists, regdata[[i]] = list(y=y_i, X=X_i), where y_i is n x 1 and X_i is n x p_i

**Prior** = list(betabar, A, nu, V) [optional]

- **betabar:** p x 1 prior mean (def: 0)
- **A:** p x p prior precision matrix (def: 0.01*I)
nu:      d.f. parameter for Inverted Wishart prior (def: m+3)
V:       \( \kappa \) scale parameter for Inverted Wishart prior (def: \( \nu I \))

\[
\text{Mmc}c = \text{list}(\text{R}, \text{keep}) \quad \text{[only R required]}
\]

R:       number of MCMC draws
keep:    MCMC thinning parameter – keep every \text{keep}th draw (def: 1)
nprint:  print the estimated time remaining for every \text{nprint}th draw (def: 100, set to 0 for no print)

Value
A list containing:

- \text{betadraw} \( R \times p \) matrix of betadraws
- \text{sigmadraw} \( R \times (m \times m) \) array of Sigma draws

Author(s)
Peter Rossi, Anderson School, UCLA, <perossichi@gmail.com>.

References
For further discussion, see Chapter 3, Bayesian Statistics and Marketing by Rossi, Allenby, and McCulloch.
http://www.perossi.org/home/bsm-1

See Also
rmultireg

Examples
if(nchar(Sys.getenv("LONG_TEST")) != 0) {R=1000} else {R=10}
set.seed(66)

## simulate data from SUR
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(0.5, 0.2, 0.2, 0.5), ncol=2)
U = chol(Sigma)
E = matrix(rnorm(2*nobs), ncol=2)%*%U
y1 = X1%*%beta1 + E[,1]
y2 = X2%*%beta2 + E[,2]

## run Gibbs Sampler
regdata = NULL
rtrun = \{ \begin{align*} \text{regdata[1]} & = \text{list}(y=y_1, X=X_1) \\ \text{regdata[2]} & = \text{list}(y=y_2, X=X_2) \\ \text{out} & = \text{rsurGibbs(Data=list(regdata=regdata), Mcmc=list(R=R))} \end{align*} \}

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)])) 

## plotting examples 
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**: standard deviation
- **a**: lower bound
- **b**: upper bound

**Details**

Note that due to the vectorization of the *rnorm* and *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.

**Note also that rtrun is currently affected by the numerical accuracy of the inverse CDF function when truncation points are far out in the tails of the distribution, where “far out” means \(|a - \mu|/\sigma > 6\) and/or \(|b - \mu|/\sigma > 6\). A fix will be implemented in a future version of bayesm.**
runireg

Author(s)
Peter Rossi, Anderson School, UCLA, <perossichi@gmail.com>.

References
For further discussion, see Chapter 2, *Bayesian Statistics and Marketing* by Rossi, Allenby, and McCulloch.
http://www.perossi.org/home/bsm-1

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, nprint)

Details
Model and Priors:  
y = Xβ + e with e ~ N(0, σ^2)
β ~ N(betabar, σ^2 * A^{-1})
σ^2 ∼ (nu * ssq)/χ^2_{nu}

Argument Details:  
Data = list(y, X)
y:  nx1 vector of observations
X:  nxk design matrix

Prior = list(betabar, A, nu, ssq) [optional]

betabar:  kx1 prior mean (def: 0)
A:  kxk prior precision matrix (def: 0.01*I)
nu: d.f. parameter for Inverted Chi-square prior (def: 3)
ssq: scale parameter for Inverted Chi-square prior (def: var(y))

Mcmc = list(R, keep, nprint) [only R required]

R: number of draws
keep: thinning parameter – keep every keepth draw (def: 1)
nprint: print the estimated time remaining for every nprint’th draw (def: 100, set to 0 for no print)

Value
A list containing:
betadraw R x k matrix of betadraws
sigmasqdraw R x 1 vector of sigma-sq draws

Author(s)
Peter Rossi, Anderson School, UCLA, <perossichi@gmail.com>.

References
For further discussion, see Chapter 2, Bayesian Statistics and Marketing by Rossi, Allenby, and McCulloch.
http://www.perossi.org/home/bsm-1

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 = 0.25
y = X%*%beta + rnorm(n, sd=sqrt(sigsq))

out = runireg(Data=list(y=y, X=X), Mcmc=list(R=R))

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

## plotting examples
if(0)(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, nprint)

Details

Model and Priors:  
y = Xβ + e with e ~ N(0, σ²)
β ~ N(betabar, A⁻¹)
σ² ~ (nu * ssq)/χ²

Argument Details:  

Data = list(y, X)
y: n x 1 vector of observations
X: n x k design matrix

Prior = list(betabar, A, nu, ssq) [optional]

betabar: k x 1 prior mean (def: 0)
A: k x k prior precision matrix (def: 0.01*I)
nu: d.f. parameter for Inverted Chi-square prior (def: 3)
ssq: scale parameter for Inverted Chi-square prior (def: var(y))

Mcmc = list(sigmasq, R, keep, nprint) [only R required]

sigmasq: value for σ² for first Gibbs sampler draw of βσ²
R: number of MCMC draws
keep: MCMC thinning parameter – keep every keepth draw (def: 1)
nprint: print the estimated time remaining for every nprint'th draw (def: 100, set to 0 for no print)

Value

A list containing:
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, $W \sim W(nu, V)$ with $E[W] = nuV$.

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

A list containing:

- W: Wishart draw
- IW: Inverted Wishart draw
- C: Upper tri root of W
- CI: $\text{inv}(C)$, $W^{-1} = \text{CIC}^\prime$

Warning

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

Author(s)

Peter Rossi, Anderson School, UCLA, <perossichi@gmail.com>.

References

For further discussion, see Chapter 2, *Bayesian Statistics and Marketing* by Rossi, Allenby, and McCulloch.

http://www.perossi.org/home/bsm-1

Examples

```r
set.seed(66)
rwishart(5, diag(3))$IW
```
Survey Data on Brands of Scotch Consumed

Description

Data from Simmons Survey. Brands used in last year for those respondents who report consuming scotch.

Usage

```r
data(Scotch)
```

Format

A data frame with 2218 observations on 21 brand variables. All variables are numeric vectors that are coded 1 if consumed in last year, 0 if not.

Source


References

Chapter 4, *Bayesian Statistics and Marketing* by Rossi, Allenby, and McCulloch [http://www.perossi.org/home/bsm-1](http://www.perossi.org/home/bsm-1)

Examples

```r
data(Scotch)
cat(" Frequencies of Brands", fill=TRUE)
mat = apply(as.matrix(Scotch), 2, mean)
print(mat)
```

```r
## 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)
1_p = diag(p)
X = rep(1_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 = rmvGibbs(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(0.01, 0.05, 0.5, 0.95, 0.99))
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)

Description

simnhlogit simulates from the non-homothetic logit model.

Usage

simnhlogit(theta, lnprices, Xexpend)

Arguments

theta coefficient vector
lnprices n xp array of prices
Xexpend n xk array of values of expenditure variables

Details

For details on parameterization, see llnhlogit.

Value

A list containing:
y nx1 vector of multinomial outcomes (1,...,p)
Xexpend expenditure variables
lnprices price array
theta coefficients
prob n xp 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, Anderson School, UCLA, <perossichi@gmail.com>.

**References**

For further discussion, see Chapter 4, *Bayesian Statistics and Marketing* by Rossi, Allenby, and McCulloch.  
http://www.perossi.org/home/bsm-1

**See Also**

`llnhlogit`

**Examples**

```r
N = 1000
p = 3
k = 1
theta = c(rep(1,p), seq(from=-1,to=1,length=p), rep(2,k), 0.5)
lnprices = matrix(runif(N*p), ncol=p)
Xexpend = matrix(runif(N*k), ncol=k)
simdata = simhlogit(theta, lnprices, Xexpend)
```

**Summary**

**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**

```r
## 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: 0.1 * 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 distributions 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).

summary.bayesm.mat(matrix) returns (invisibly) the array of the various summary statistics for further use. To assess this array use stats=summary(Drawmat).

Author(s)

Peter Rossi, Anderson School, UCLA, <perossichi@gmail.com>.

See Also

summary.bayesm.var, summary.bayesm.nmix

Examples

## Not run: out=rmpGibbs(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 for each dimension of the density
- **burnin**: number of draws to burn-in (def: 0.1 * nrow(probdraw))
- **...**: parms to send to summary

Details

An object of class `bayesm.nmix` is a list of three components:

- **probdraw**: a matrix of \( R/\text{keep} \) 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, Anderson School, UCLA, <perossichi@gmail.com>.

See Also

`summary.bayesm.mat`, `summary.bayesm.var`

Examples

```r
## Not run: out=rnmix(Data,Prior,Mcmc); summary(out)
```

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(Yard)), tvalues, QUANTILES = FALSE, ...)
Arguments

object: object (hereafter, 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: \(0.1 \times nrow(Vard)\)).
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 distributions 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, Anderson School, UCLA, <perossichi@gmail.com>.

See Also

summary.bayesm.mat, summary.bayesm.nmix

Examples

```r
## Not run: out=rnmnpGibbs(Data,Prior,Mcmc); summary(out$sigmadraw)
```

tuna  Canned Tuna Sales Data

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 Kilts Center for Marketing at the University of Chicago’s Booth
School of Business. 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 30 variables.

...$WEEK       a numeric vector
...$MOVE#      unit sales of brand #
...$NSALE#     a measure of display activity of brand #
...$LPRICE#    log of price of brand #
...$LWHPRIC#   log of wholesale price of brand #
...$FULLCUST   total customers visits

The brands are:

1. Star Kist 6 oz.
2. Chicken of the Sea 6 oz.
5. Geisha 6 oz.
7. HH Chunk Lite 6.5 oz.

Source


References

Chapter 7, Bayesian Statistics and Marketing by Rossi, Allenby, and McCulloch.
http://www.perossi.org/home/bsm-1

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)
```

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

*Topic array
createX, 14
nmat, 38

*Topic datasets
bank, 3
camera, 7
cheese, 9
customerSat, 15
detailing, 16
margarine, 30
orangeJuice, 40
Scotch, 113
tuna, 118

*Topic distribution
breg, 6
condMom, 13
g congest, 19
lndlChisq, 25
lndlWishart, 26
lndlMvn, 27
lndlMvst, 28
loglMargDenNR, 29
rbinNormGibbs, 51
rdirichlet, 53
rmixture, 85
rmvst, 94
rtrun, 107
rwishart, 111

*Topic hplot
plot.bayesm.hcoef, 43
plot.bayesm.mat, 44
plot.bayesm.nmix, 45

*Topic models
breg, 6
clusterMix, 11
eMixMargDen, 18
llmnl, 21
llmp, 22
llnhlogit, 24
mixDen, 33
mixDenBi, 34
mnlHess, 35
mnpProb, 36
rbayesBLP, 46
rbprobibGibbs, 52
rhierBinLogit, 58
rhierMnlDP, 66
rhierMnlRwMixture, 71
rhierNegbinRw, 75
rivDP, 78
rivGibbs, 81
rmlIndDepMetrhop, 86
rmpGibbs, 88
rmvpgibbs, 92
rnegbinRw, 95
rordprobibGibbs, 100
rscaleUsage, 103
simnhlogit, 114

*Topic multivariate
clusterMix, 11
eMixMargDen, 18
mixDen, 33
mixDenBi, 34
mmomMix, 37
rDpgibbs, 54
rmxGibbs, 83
rmixture, 85
rmvpgibbs, 92
rmvst, 94
rmxMixGibbs, 98
rwishart, 111

*Topic plot
summary.bayesm.nmix, 116

*Topic regression
breg, 6
rhierLinearMixture, 61
rhierLinearModel, 64
rmultireg, 90
<table>
<thead>
<tr>
<th>Topic</th>
<th>Function(s)</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ts</td>
<td>numEff, 39</td>
<td>39</td>
</tr>
<tr>
<td>univar</td>
<td>summary.bayesm.mat, 115, summary.bayesm.var, 117</td>
<td>115, 117</td>
</tr>
<tr>
<td>utilities</td>
<td>cgetC, 8, createX, 14, nmat, 38, numEff, 39</td>
<td>8, 14, 38, 39</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>bank, 3</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>breg, 6</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>camera, 7</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>cgetC, 8</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>cheese, 9</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>clusterMix, 11</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>condMom, 13</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>createX, 14, 21–23, 35–37, 88</td>
<td>14, 21, 35, 37</td>
</tr>
<tr>
<td></td>
<td>customerSat, 15</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>dchisq, 26</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>detailing, 16</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>eMixMargDen, 18, 56, 99</td>
<td>18, 56, 99</td>
</tr>
<tr>
<td></td>
<td>ghkvec, 19</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>llnml, 21, 35, 36</td>
<td>21, 35, 36</td>
</tr>
<tr>
<td></td>
<td>llmmp, 22</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>lnlnhlogit, 24, 115</td>
<td>24, 115</td>
</tr>
<tr>
<td></td>
<td>lndichi, 25</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>lndmvwishart, 26</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td>lndmvst, 28, 28, 95</td>
<td>28, 28, 95</td>
</tr>
<tr>
<td></td>
<td>logMargaDenNR, 29</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>margarine, 30</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>mixDen, 33, 35, 56, 99</td>
<td>33, 35, 56, 99</td>
</tr>
<tr>
<td></td>
<td>mixDenBi, 34, 56, 99</td>
<td>34, 56, 99</td>
</tr>
<tr>
<td></td>
<td>mnlHess, 35</td>
<td>35</td>
</tr>
<tr>
<td></td>
<td>mnpProb, 36</td>
<td>36</td>
</tr>
<tr>
<td></td>
<td>momMix, 37, 56, 99</td>
<td>37, 56, 99</td>
</tr>
<tr>
<td></td>
<td>nmat, 38</td>
<td>38</td>
</tr>
<tr>
<td></td>
<td>numEff, 39</td>
<td>39</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Topic</th>
<th>Function(s)</th>
<th>Page(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>orangeJuice, 40</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>plot.bayesm.hcoef, 43</td>
<td>43</td>
</tr>
<tr>
<td></td>
<td>plot.bayesm.mat, 44</td>
<td>44</td>
</tr>
<tr>
<td></td>
<td>plot.bayesm.mrix, 45</td>
<td>45</td>
</tr>
<tr>
<td></td>
<td>rbayesBLP, 46</td>
<td>46</td>
</tr>
<tr>
<td></td>
<td>rbinormgibbs, 51</td>
<td>51</td>
</tr>
<tr>
<td></td>
<td>rpbprobitgibbs, 52, 102</td>
<td>52, 102</td>
</tr>
<tr>
<td></td>
<td>rdirichlet, 53</td>
<td>53</td>
</tr>
<tr>
<td></td>
<td>rDPGibbs, 45, 54</td>
<td>45, 54</td>
</tr>
<tr>
<td></td>
<td>rhierBinLogit, 58</td>
<td>58</td>
</tr>
<tr>
<td></td>
<td>rhierLinearMixture, 43, 45, 61, 65</td>
<td>43, 45, 61, 65</td>
</tr>
<tr>
<td></td>
<td>rhierLinearModel, 43, 63, 64</td>
<td>43, 63, 64</td>
</tr>
<tr>
<td></td>
<td>rhierMnlDP, 66</td>
<td>66</td>
</tr>
<tr>
<td></td>
<td>rhierMnlRwMixture, 14, 43, 45, 50, 69, 71, 87</td>
<td>14, 43, 45, 50, 69, 71, 87</td>
</tr>
<tr>
<td></td>
<td>rhierNegbinRw, 43, 75, 97</td>
<td>43, 75, 97</td>
</tr>
<tr>
<td></td>
<td>rivDP, 78</td>
<td>78</td>
</tr>
<tr>
<td></td>
<td>rivGibbs, 81</td>
<td>81</td>
</tr>
<tr>
<td></td>
<td>rmixGibbs, 38, 56, 83, 99</td>
<td>38, 56, 83, 99</td>
</tr>
<tr>
<td></td>
<td>rmxnorm, 56, 99</td>
<td>56, 99</td>
</tr>
<tr>
<td></td>
<td>rmmlmldenmetrop, 14, 15, 22, 36, 73, 86</td>
<td>14, 15, 22, 36, 73, 86</td>
</tr>
<tr>
<td></td>
<td>rmpnGibbs, 14, 15, 23, 36, 53, 88, 93</td>
<td>14, 15, 23, 36, 53, 88, 93</td>
</tr>
<tr>
<td></td>
<td>rmultireg, 90, 106</td>
<td>90, 106</td>
</tr>
<tr>
<td></td>
<td>rmpvGibbs, 14, 89, 92</td>
<td>14, 89, 92</td>
</tr>
<tr>
<td></td>
<td>rmvst, 94</td>
<td>94</td>
</tr>
<tr>
<td></td>
<td>rnegbinRw, 77, 95</td>
<td>77, 95</td>
</tr>
<tr>
<td></td>
<td>rmmixGibbs, 12, 19, 34, 35, 45, 56, 84, 85, 98</td>
<td>12, 19, 34, 35, 45, 56, 84, 85, 98</td>
</tr>
<tr>
<td></td>
<td>rordprobitGibbs, 100</td>
<td>100</td>
</tr>
<tr>
<td></td>
<td>rscaleUsage, 9, 103</td>
<td>9, 103</td>
</tr>
<tr>
<td></td>
<td>rsurgibbs, 105</td>
<td>105</td>
</tr>
<tr>
<td></td>
<td>rtrun, 107</td>
<td>107</td>
</tr>
<tr>
<td></td>
<td>runireg, 108, 111</td>
<td>108, 111</td>
</tr>
<tr>
<td></td>
<td>runiregGibbs, 109, 110</td>
<td>109, 110</td>
</tr>
<tr>
<td></td>
<td>rwishart, 27, 111</td>
<td>27, 111</td>
</tr>
</tbody>
</table>

Scotch, 113
simnhlogit, 25, 114
summary.bayesm.mat, 115, 117, 118
summary.bayesm.nmix, 116, 116, 118
summary.bayesm.var, 116, 117, 117

tuna, 118