

Package ‘hbmem’

February 20, 2015

Type Package

Title Hierarchical Bayesian Analysis of Recognition Memory

Version 0.3

Date 2010-03-15

Author Michael S. Pratte

Maintainer Mike <prattems@gmail.com>

Depends R (>= 1.8.0), methods

Description Contains functions for fitting hierarchical versions of EVSD, UVSD, DPSD, DPSD with d' restricted to be positive, and our gamma signal detection model to recognition memory confidence-ratings data.

License LGPL (>= 2.0)

LazyLoad yes

URL <http://pcl.missouri.edu>

Repository CRAN

Date/Publication 2012-12-20 09:21:57

NeedsCompilation yes

R topics documented:

hbmem-package	2
dpsd-class	4
dpsdProbs	5
dpsdRNSample	6
dpsdRNSim	7
dpsdSample	8
dpsdSim	11
dpsdSim-class	12
gammaLikeSample	13
gammaProbs	15
gammaSample	16

gammaSim	18
normalSim	19
prm09	21
rtgamma	22
rtnorm	22
sampleGamma	23
sampleNorm	26
sampleNormb	29
sampleNormR	31
samplePosNorm	34
sampleSig2	37
sampleSig2b	38
uvsd-class	40
uvsdLogLike	41
uvsdProbs	42
uvsdSample	43
uvsdSim	46
uvsdSim-class	47

Index	48
--------------	-----------

hbmem-package

Hierarchical Models of Recognition Memory

Description

Contains functions for fitting hierarchical versions of EVSD, UVSD, DPSD, and our gamma signal detection model to recognition memory confidence-ratings data.

Details

Package:	hbmem
Type:	Package
Version:	0.3
Date:	2012-12-20
License:	LGPL
LazyLoad:	yes

Author(s)

Michael S. Pratte <prattems@gmail.com>

References

Morey, Pratte, and Rouder (2008); Pratte, Rouder, and Morey (2009); Pratte and Rouder (2012).

See Also

'uvsdSample' to fit hierarchical UVSD model, 'uvsdSim' to simulate data from the hierarchical UVSD model, 'dpsdSample' to fit the hierarchical DPSD model, 'dpsdSim' to simulate data from the hierarchical DPSD model, 'dpsdPosSim' and 'dpsdPosSample' for the DPSD model with positive sensitivity, and datasets from our publications.

Examples

```
#In this example data are simulated from EVSD
#They are then fit by both UVSD and DPSD

library(hbmem)
sim=uvsdSim(s2aS2=0,s2bS2=0) #Simulate data from hierarchical EVSD
dat=as.data.frame(cbind(sim@subj,sim@item,sim@Scond,sim@cond,sim@lag,sim@resp))
colnames(dat)=c("sub","item","Scond","cond","lag","resp")

M=10 #Set way low for speed
keep=2:M
#For real analysis we run 105000 iterations
#with the first 5000 serving as burnin, and
#only keep every 10th iteration for analysis,
#i.e., thinning the chains to mitigate autocorrelation.
evsd=uvsdSample(dat,M=M,keep=keep,equalVar=TRUE) #Fit EVSD
uvsd=uvsdSample(dat,M=M,keep=keep,freeSig2=TRUE) #Fit UVSD w/1 Sigma2
dpsd=dpsdSample(dat,M=M,keep=keep) #Fit DPSD

#Look at available information
slotNames(uvsd)
slotNames(dpsd)

#Compare DIC; smaller is better
evsd@DIC
uvsd@DIC
dpsd@DIC

#Effective parameters. Because there are no
#real effects on studied-item variance, the
#hierarchical models are drastically shrinking these
#effect parameters to zero, so that they do not
#count as full parameters.
evsd@pD
uvsd@pD
dpsd@pD

#PLOTS FROM UVSD FIT
par(mfrow=c(3,2),pch=19,pty='s')
#Make sure chains look OK
matplot(uvsd@blockN[,uvsd@muN],t='l',xlab="Iteration",ylab="Mu-N")
abline(h=sim@muN,col="blue")
matplot(uvsd@blockS[,uvsd@muS],t='l',xlab="Iteration",ylab="Mu-S")
abline(h=sim@muS,col="blue")
```

```

#Estimates of Alpha as function of true values
plot(uvsd@estN[uvsd@alphaN]~sim@alphaN,xlab="True
Alpha-N",ylab="Est. Alpha-N");abline(0,1,col="blue")
plot(uvsd@estS[uvsd@alphaS]~sim@alphaS,xlab="True
Alpha-S",ylab="Est. Alpha-S");abline(0,1,col="blue")
#Estimates of Beta as function of true values
plot(uvsd@estN[uvsd@betaN]~sim@betaN,xlab="True
Beta-N",ylab="Est. Beta-N");abline(0,1,col="blue")
plot(uvsd@estS[uvsd@betaS]~sim@betaS,xlab="True
Beta-S",ylab="Est. Beta-S");abline(0,1,col="blue")

###Look at Sigma2 and Recollection from UVSD and DPSD###
par(mfrow=c(2,3),pch=19,pty='s')
plot(sqrt(exp(uvsd@blockS2[,uvsd@muS])),
t='l',ylab="Sigma",main="Grand Mean")
abline(h=1,col="blue")
hist(uvsd@blockS2[,uvsd@s2alphaS],main="Participant Effect")
hist(uvsd@blockS2[,uvsd@s2betaS],main="Item Effect")

plot(pnorm(dpsd@blockR[,dpsd@muS]),
t='l',ylab="P(Recollection)",main="Grand Mean")
abline(h=0,col="blue")
hist(dpsd@blockR[,dpsd@s2alphaS],main="Participant Effect")
hist(dpsd@blockR[,dpsd@s2betaS],main="Item Effect")

#See what DPSD does with EVSD effects
par(mfrow=c(2,3))
plot(dpsd@estN[dpsd@alphaN]~sim@alphaN,xlab="True
Alpha-N",ylab="DPSD Alpha-N");abline(0,1,col="blue")
plot(dpsd@estS[dpsd@alphaS]~sim@alphaS,xlab="True
Alpha-S",ylab="DPSD Alpha-S");abline(0,1,col="blue")
plot(dpsd@estR[dpsd@alphaS]~sim@alphaS,xlab="True
Alpha-S",ylab="DPSD Alpha-R");abline(0,1,col="blue")

plot(dpsd@estN[dpsd@betaN]~sim@betaN,xlab="True
Beta-N",ylab="DPSD Beta-N");abline(0,1,col="blue")
plot(dpsd@estS[dpsd@betaS]~sim@betaS,xlab="True
Beta-S",ylab="DPSD Beta-S");abline(0,1,col="blue")
plot(dpsd@estR[dpsd@betaS]~sim@betaS,xlab="True
Beta-S",ylab="DPSD Beta-R");abline(0,1,col="blue")

```

dpsd-class

Class "dpsd" ~~~

Description

Holds all information returned from posterior simulations of dual-process models

Slots

muN: Object of class "numeric" ~~
 alphaN: Object of class "numeric" ~~
 betaN: Object of class "numeric" ~~
 s2alphaN: Object of class "numeric" ~~
 s2betaN: Object of class "numeric" ~~
 thetaN: Object of class "numeric" ~~
 muS: Object of class "numeric" ~~
 alphaS: Object of class "numeric" ~~
 betaS: Object of class "numeric" ~~
 s2alphaS: Object of class "numeric" ~~
 s2betaS: Object of class "numeric" ~~
 thetaS: Object of class "numeric" ~~
 estN: Object of class "numeric" ~~
 estS: Object of class "numeric" ~~
 estR: Object of class "numeric" ~~
 estCrit: Object of class "matrix" ~~
 blockN: Object of class "matrix" ~~
 blockS: Object of class "matrix" ~~
 blockR: Object of class "matrix" ~~
 s.crit: Object of class "array" ~~
 pD: Object of class "numeric" ~~
 DIC: Object of class "numeric" ~~
 M: Object of class "numeric" ~~
 keep: Object of class "numeric" ~~
 b0: Object of class "matrix" ~~
 b0Crit: Object of class "numeric" ~~

 dpsdProbs

Function dpsdProbs

Description

Returns the probability of making confidence ratings given parameters of DPSD.

Usage

dpsdProbs(r,d,crit)

Arguments

r	Probability of recollection.
d	Mean of the signal-detection distribution. In the common parameterization of the model, this would be zero for new-item trials, and d' for studied-item trials. In the PRM09 parameterization, these are d_n and d_s for new and studied-item trials, respectively.
crit	Criteria (not including $-\text{Inf}$ or Inf).

Details

For new-item trials, simply set $r=0$.

Value

The function returns the probability of making each response for the paramters given.

Author(s)

Michael S. Pratte

References

See Pratte, Rouder, & Morey (2009)

See Also

hbmern

Examples

```
#Low r
dpsdProbs(.2,1,c(-1,-.5,0,.5,1)) #studied
dpsdProbs(0,-1,c(-1,-.5,0,.5,1)) #new

#High r
dpsdProbs(.6,1,c(-1,-.5,0,.5,1)) #studied
dpsdProbs(0,-1,c(-1,-.5,0,.5,1)) #new
```

dpsdRNSample

Fit DPSD model with R restricted to be function of N

Description

This is a dual process model in which the person and item effects on probability of recollection are linear functions of those effects for the new-item distribuion.

Usage

```
dpsdRNSample(dat, M = 5000, keep = (M/10):M, getDIC = TRUE, jump = 0.001)
```

Arguments

dat	Data frame that must include variables Scond,cond,sub,item,lag,resp. Scond indexes studied/new, whereas cond indexes conditions nested within the studied or new conditions. Indexes for Scond,cond, sub, item, and response must start at zero and have no gaps (i.e., no skipped subject numbers). Lags must be zero-centered.
M	Number of MCMC iterations.
keep	Which MCMC iterations should be included in estimates and returned. Use keep to both get ride of burn-in, and thin chains if necessary
getDIC	Logical. Should the function compute DIC value? This takes a while if M is large.
jump	The criteria and decorrelating steps utilize Matropolis-Hastings sampling routines, which require tuning. All MCMC functions should self-tune during the burnin period (iterations before keep), and they will alert you to the success of tuning. If acceptance rates are too low, "jump" should be decreased, if they are too high, "jump" should be increased. Alternatively, or in addition to adjusting "jump", simply increase the burnin period which will allow the function more time to self-tune.

References

Pratte and Rouder (2010)

dpsdRNSim *Function dpsdRNSim*

Description

Simulate data from DPSD model with R a function of N

Usage

```
dpsdRNSim(NN = 2, NS = 1, I = 30, J = 200, K = 6, muN = c(-0.7, -0.5), s2aN = 0.2, s2bN = 0.2, muS = 0, s
```

Arguments

NN	Number of new-item conditions.
NS	Number of studied-item conditions.
I	Number of participants.
J	Number of items.
K	Number of confidence ratings

muN	Mean of new-item distribuion
s2aN	Variance of participant effects on new-item distribution
s2bN	Variance of item effects on new-item distribution
muS	Mean of studied-item distribution
s2aS	Variance of participant effects on studied-item distribution
s2bS	Variance of item effects on studied-item distribution
muR	Mean of recollection (on probit space)
phiA	Linear slope of participant effect on recollection.
phiB	Linear slope of item effect on recollection.
crit	Matrix of criteria

References

See Pratte and Rouder (in review)

dpsdSample	<i>Function to fit hierarchical DPSD model to data.</i>
------------	---------------------------------------------------------

Description

Runs MCMC estimation for the hierarchical DPSD model.

Usage

```
dpsdSample(dat, M = 5000, keep = (M/10):M, getDIC = TRUE,
freeCrit=TRUE,Hier=TRUE, jump=.01)
```

Arguments

dat	Data frame that must include variables Scond,cond,sub,item,lag,resp. Scond indexes studied/new, whereas cond indexes conditions nested within the studied or new conditions. Indexes for Scond,cond, sub, item, and response must start at zero and have no gaps (i.e., no skipped subject numbers). Lags must be zero-centered.
M	Number of MCMC iterations.
keep	Which MCMC iterations should be included in estimates and returned. Use keep to both get ride of burn-in, and thin chains if necessary
getDIC	Logical. Should the function compute DIC value? This takes a while if M is large.
freeCrit	Logical. If true then criteria are estimated separately for each participant. Should be set to false if analyzing only one participant (e.g., if averaging over subjects).

Hier	Logical. If true then the variances of effects (e.g., item effects) are estimated from the data, i.e., effects are treated as random. If false then these variances are fixed to 2.0 (.5 for recollection effects), thus treating these effects as fixed. This option is there to allow for comparison with more traditional approaches, and to see the effects of imposing hierarchical structure. It should always be set to TRUE in real analysis, and is not even guaranteed to work if set to false.
jump	The criteria and decorrelating steps utilize Metropolis-Hastings sampling routines, which require tuning. All MCMC functions should self-tune during the burnin period (iterations before keep), and they will alert you to the success of tuning. If acceptance rates are too low, "jump" should be decreased, if they are too high, "jump" should be increased. Alternatively, or in addition to adjusting "jump", simply increase the burnin period which will allow the function more time to self-tune.

Value

The function returns an internally defined "uvsd" structure that includes the following components

mu	Indexes which element of blocks contain mu
alpha	Indexes which element of blocks contain participant effects, alpha
beta	Indexes which element of blocks contain item effects, beta
s2alpha	Indexes which element of blocks contain variance of participant effects (alpha).
s2beta	Indexes which element of blocks contain variance of item effects (beta).
theta	Indexes which element of blocks contain theta, the slope of the lag effect
estN	Posterior means of block parameters for new-item means
estS	Posterior means of block parameters for studied-item means
estR	Posterior means of block for Recollection means.
estCrit	Posterior means of criteria
blockN	Each iteration for each parameter in the new-item mean block. Rows index iteration, columns index parameter.
blockS	Same as blockN, but for the studied-item means
blockR	Same as blockN, but for the recollection-parameter means.
s.crit	Samples of each criteria.
pD	Number of effective parameters used in DIC. Note that this should be smaller than the actual number of parameters, as constraint from the hierarchical structure decreases the number of effective parameters.
DIC	DIC value. Smaller values indicate better fits. Note that DIC is notably biased toward complexity.
M	Number of MCMC iterations run
keep	MCMC iterations that were used for estimation and returned
b0	Metropolis-Hastings acceptance rates for decorrelating steps. These should be between .2 and .6. If they are not, the M, keep, or jump arguments need to be adjusted.
b0Crit	acceptance rates for criteria.

Author(s)

Michael S. Pratte

References

See Pratte, Rouder, & Morey (2009)

See Also

hbmem

Examples

```
#In this example we generate data from EVSD, then fit it with both
#hierarchical DPSD and DPSD assuming no participant or item effects.
library(hbmem)
sim=dpsdSim(I=30,J=200)
dat=as.data.frame(cbind(sim@subj,sim@item,sim@cond,sim@Scond,sim@lag,sim@resp))
colnames(dat)=c("sub","item","cond","Scond","lag","resp")
dat$lag[dat$Scond==1]=dat$lag[dat$Scond==1]-mean(dat$lag[dat$Scond==1])

M=10 #Too low for real analysis!
keep=2:M
DPSD=dpsdSample(dat,M=M)

#Look at all parameters
par(mfrow=c(3,3),pch=19,pty='s')

matplot(DPSD@blockN[,DPSD@muN],t='l',
ylab="muN")
abline(h=sim@muN,col="blue")
plot(DPSD@estN[DPSD@alphaN]~sim@alphaN)
abline(0,1,col="blue")
plot(DPSD@estN[DPSD@betaN]~sim@betaN)
abline(0,1,col="blue")

matplot(DPSD@blockS[,DPSD@muS],t='l',
ylab="muS")
abline(h=sim@muS,col="blue")
plot(DPSD@estS[DPSD@alphaS]~sim@alphaS)
abline(0,1,col="blue")
plot(DPSD@estS[DPSD@betaS]~sim@betaS)
abline(0,1,col="blue")

matplot(pnorm(DPSD@blockR[,DPSD@muS]),t='l',
ylab="P(recollection)")
abline(h=pnorm(sim@muR),col="blue")
plot(DPSD@estR[DPSD@alphaS]~sim@alphaR)
abline(0,1,col="blue")
plot(DPSD@estR[DPSD@betaS]~sim@betaR)
abline(0,1,col="blue")
```

dpsdSim *Function dpsdSim*

Description

Simulates data from a hierarchical DPSD model.

Usage

```
dpsdSim(NN=2, NS=1, I=30, J=200, K=6, muN=c(-.7, -.5), s2aN=.2, s2bN=.2,
muS=0, s2aS=.2, s2bS=.2, muR=qnorm(.25), s2aR=.2, s2bR=.2,
crit=matrix(rep(c(-1.6, -.5, 0, .5, 1.6), each=I), ncol=(K-1)))
```

Arguments

NN	Number of new-item conditions.
NS	Number of studied-item conditions.
I	Number of participants.
J	Number of items.
K	Number of response options.
muN	Mean of new-item distribution. If there are more than one new-item conditions this is a vector of means with length equal to NN.
s2aN	Variance of participant effects on mean of new-item distribution.
s2bN	Variance of item effects on mean of new-item distribution.
muS	Mean of studied-item distribution. If there are more than new-item conditions this is a vector of means with length equal to NN None studied-item conditions this is a vector of means with length equal to NS.
s2aS	Variance of participant effects on mean of studied-item distribution.
s2bS	Variance of item effects on mean of studied-item distribution.
muR	Mean recollection, on probit space.
s2aR	Variance of participant effects recollection.
s2bR	Variance of item effects on recollection.
crit	Matrix of criteria (not including -Inf or Inf). Columns correspond to criteria, rows correspond to participants.

Value

The function returns an internally defined "dpsdSim" structure.

Author(s)

Michael S. Pratte

References

See Pratte, Rouder, & Morey (2009)

See Also

hbmem

Examples

```
library(hbmem)
#Data from hiererchial model
sim=dpsdSim()
slotNames(sim)
#Scond indicates studied/new
#cond indicates which condition (e.g., deep/shallow)

table(sim@resp,sim@Scond,sim@cond)

#Usefull to make data.frame for passing to functions
dat=as.data.frame(cbind(sim@subj,sim@item,sim@Scond,sim@cond,sim@lag,sim@resp))
colnames(dat)=c("sub","item","Scond","cond","lag","resp")

table(dat$resp,dat$Scond,dat$cond)
```

dpsdSim-class

Class "dpsdSim"

Description

Class "dpsdSim" to hold objects from DPSD simulations.

Slots

Scond: Object of class "numeric" ~~
 cond: Object of class "numeric" ~~
 subj: Object of class "numeric" ~~
 item: Object of class "numeric" ~~
 lag: Object of class "numeric" ~~
 resp: Object of class "numeric" ~~
 muN: Object of class "numeric" ~~
 muS: Object of class "numeric" ~~
 muR: Object of class "numeric" ~~
 alphaN: Object of class "numeric" ~~
 betaN: Object of class "numeric" ~~
 alphaS: Object of class "numeric" ~~

betaS: Object of class "numeric" ~~
 alphaR: Object of class "numeric" ~~
 betaR: Object of class "numeric" ~~

gammaLikeSample *Function gammaLikeSample*

Description

Runs MCMC for the hierarchical Gamma Likelihood model

Usage

```
gammaLikeSample(dat, M = 10000, keep = (M/10):M, getDIC = TRUE,
  shape=2, jump=.005)
```

Arguments

dat	Data frame that must include variables cond,sub,item,lag,resp. Indexes for cond, sub, item, and response must start at zero and have no gaps (i.e., no skipped subject numbers). Lags must be zero-centered.
M	Number of MCMC iterations.
keep	Which MCMC iterations should be included in estimates and returned. Use keep to both get ride of burn-in, and thin chains if necessary
getDIC	Logical. should the function compute DIC value? This takes a while if M is large.
shape	Fixed shape across both new and studied distribuitions.
jump	The criteria and decorrelating steps utilize Matropolis-Hastings sampling routines, which require tuning. All MCMC functions should self tune during the burnin perior (iterations before keep), and they will alert you to the success of tuning. If acceptance rates are too low, "jump" should be decreased, if they are too hight, "jump" should be increased. Alternatively, or in addition to adjusting "jump", simply increase the burnin period which will allow the function more time to self-tune.

Value

The function returns an internally defined "uvsd" S4 class that includes the following components

mu	Indexes which element of blocks contain grand means, mu
alpha	Indexes which element of blocks contain participant effects, alpha
beta	Indexes which element of blocks contain item effects, beta
s2alpha	Indexes which element of blocks contain variance of participant effects (alpha).
s2beta	Indexes which element of blocks contain variance of item effects (beta).

theta	Indexes which element of blocks contain theta, the slope of the lag effect
estN	Posterior means of block parameters for new-item means
estS	Posterior means of block parameters for studied-item means
estS2	Not used for gamma model.
estCrit	Posterior means of criteria
blockN	Each iteration for each parameter in the new-item mean block. Rows index iteration, columns index parameter.
blocks	Same as blockN, but for the studied-item means
blocks2	Not used for gamma model.
s.crit	Samples of each criteria.
pD	Number of effective parameters used in DIC. Note that this should be smaller than the actual number of parameters, as constraint from the hierarchical structure decreases the number of effective parameters.
DIC	DIC value. Smaller values indicate better fits. Note that DIC is notably biased toward complexity.
M	Number of MCMC iterations run
keep	MCMC iterations that were used for estimation and returned
b0	Metropolis-Hastings acceptance rates for new-item distribution parameters. These should be between .2 and .6. If they are not, the M, keep, or jump need to be adjusted.
b0S2	Metropolis-Hastings acceptance rates for studied-item distribution parameters.
b0Crit	Metropolis-Hastings acceptance rates for criteria.

Author(s)

Michael S. Pratte

See Also

hbmem

Examples

```
#This function is broken, so
#no example that works.
#make data from gamma model
if(1==0)
{
library(hbmem)
sim=gammaLikeSim(I=50,J=400,muS=log(.5),s2aS=0,s2bS=0)
dat=as.data.frame(cbind(sim@subj,sim@item,sim@cond,sim@Scond,sim@lag,sim@resp))
colnames(dat)=c("sub","item","cond","Scond","lag","resp")
dat$lag=0

table(dat$resp,dat$Scond)
M=5000
```

```

keep=500:M
gamma=gammaLikeSample(dat,M=M,keep=keep,jump=.001)

par(mfrow=c(2,3),pch=19,pty='s')
matplot(exp(gamma@blockS[,gamma@muS]),t='l',xlab="Iteration",ylab="Mu-S")
abline(h=exp(sim@muS),col="blue")
#Estimates of Alpha as function of true values
plot(gamma@estS[gamma@alphaS]~sim@alphaS,xlab="True
Alpha-S",ylab="Est. Alpha-S");abline(0,1,col="blue")
#Estimates of Beta as function of true values
plot(gamma@estS[gamma@betaS]~sim@betaS,xlab="True
Beta-S",ylab="Est. Beta-S");abline(0,1,col="blue")

#Look at some criteria
for(i in 1:3){
matplot(t(exp(gamma@s.crit[i,2:7])),t='l')
abline(h=sim@crit[i,])
}

gamma@estS[c(gamma@s2alphaS,gamma@s2betaS)]
}

```

gammaProbs

Function gammaProbs

Description

Returns the probability of making confidence rating responses given parameters of gamma signal detection model.

Usage

```
gammaProbs(scale, shape, bounds)
```

Arguments

scale	Scale of gamma distribution.
shape	Shape of gamma distribution, usually fixed to 2.0
bounds	Criteria placed on strength axis.

gammaSample	<i>Function gammaSample</i>
-------------	-----------------------------

Description

Runs MCMC for the hierarchical Gamma model

Usage

```
gammaSample(dat, M = 10000, keep = (M/10):M, getDIC = TRUE,
  freeCrit=TRUE, shape=2, jump=.005)
```

Arguments

dat	Data frame that must include variables cond,sub,item,lag,resp. Indexes for cond, sub, item, and response must start at zero and have no gapes (i.e., no skipped subject numbers). Lags must be zero-centered.
M	Number of MCMC iterations.
keep	Which MCMC iterations should be included in estimates and returned. Use keep to both get ride of burn-in, and thin chains if necessary
getDIC	Logical. should the function compute DIC value? This takes a while if M is large.
freeCrit	Logical. If TRUE (default) individual criteria vary across people. If false, all participants have the same criteria (but note that overall response biases are still modeled in the means)
shape	Fixed shape across both new and studied distributions.
jump	The criteria and decorrelating steps utilize Matropolis-Hastings sampling routines, which require tuning. All MCMC functions should self tune during the burnin perior (iterations before keep), and they will alert you to the success of tuning. If acceptance rates are too low, "jump" should be decreased, if they are too high, "jump" should be increased. Alternatively, or in addition to adjusting "jump", simply increase the burnin period which will allow the function more time to self-tune.

Value

The function returns an internally defined "uvsd" S4 class that includes the following components

mu	Indexes which element of blocks contain grand means, mu
alpha	Indexes which element of blocks contain participant effects, alpha
beta	Indexes which element of blocks contain item effects, beta
s2alpha	Indexes which element of blocks contain variance of participant effects (alpha).
s2beta	Indexes which element of blocks contain variance of item effects (beta).
theta	Indexes which element of blocks contain theta, the slope of the lag effect

estN	Posterior means of block parameters for new-item means
estS	Posterior means of block parameters for studied-item means
estS2	Not used for gamma model.
estCrit	Posterior means of criteria
blockN	Each iteration for each parameter in the new-item mean block. Rows index iteration, columns index parameter.
blockS	Same as blockN, but for the studied-item means
blockS2	Not used for gamma model.
s.crit	Samples of each criteria.
pD	Number of effective parameters used in DIC. Note that this should be smaller than the actual number of parameters, as constraint from the hierarchical structure decreases the number of effective parameters.
DIC	DIC value. Smaller values indicate better fits. Note that DIC is notably biased toward complexity.
M	Number of MCMC iterations run
keep	MCMC iterations that were used for estimation and returned
b0	Metropolis-Hastings acceptance rates for new-item distribution parameters. These should be between .2 and .6. If they are not, the M, keep, or jump need to be adjusted.
b0S2	Metropolis-Hastings acceptance rates for studied-item distribution parameters.
b0Crit	Metropolis-Hastings acceptance rates for criteria.

Author(s)

Michael S. Pratte

See Also

hbmem

Examples

```
#make data from gamma model
library(hbmem)
sim=gammaSim(I=30,J=200)
dat=as.data.frame(cbind(sim@subj,sim@item,sim@cond,sim@Scond,sim@lag,sim@resp))
colnames(dat)=c("sub","item","cond","Scond","lag","resp")

M=10 #set very small for demo speed
keep=2:M
gamma=gammaSample(dat,M=M,keep=keep,jump=.01)

par(mfrow=c(3,2),pch=19,pty='s')
#Look at chains of MuN and MuS
matplot(gamma@blockN[,gamma@muN],t='l',xlab="Iteration",ylab="Mu-N")
abline(h=sim@muN,col="blue")
```

```

matplot(gamma@blockS[,gamma@muS],t='1',xlab="Iteration",ylab="Mu-S")
abline(h=sim@muS,col="blue")

#Estimates of Alpha as function of true values
plot(gamma@estN[gamma@alphaN]~sim@alphaN,xlab="True
Alpha-N",ylab="Est. Alpha-N");abline(0,1,col="blue")
plot(gamma@estS[gamma@alphaS]~sim@alphaS,xlab="True
Alpha-S",ylab="Est. Alpha-S");abline(0,1,col="blue")
#Estimates of Beta as function of true values
plot(gamma@estN[gamma@betaN]~sim@betaN,xlab="True
Beta-N",ylab="Est. Beta-N");abline(0,1,col="blue")
plot(gamma@estS[gamma@betaS]~sim@betaS,xlab="True
Beta-S",ylab="Est. Beta-S");abline(0,1,col="blue")

gamma@estN[c(gamma@s2alphaN,gamma@s2betaN)]
gamma@estS[c(gamma@s2alphaS,gamma@s2betaS)]

#Look at some criteria
par(mfrow=c(2,2))
for(i in 1:4)
matplot(t(gamma@s.crit[i,]),t='1')

```

gammaSim

Function gammaSim

Description

Simulates data from a hierarchical Gamma model.

Usage

```

gammaSim(NN=1,NS=2,I=30,J=200,K=6,muN=log(.65),s2aN=.2,s2bN=.2,
muS=log(c(.8,1.2)),s2aS=.2,s2bS=.2,lagEffect=-.001,shape=2,
crit=matrix(rep(c(.3,.6,1,1.2,1.6),each=I),ncol=(K-1)))

```

Arguments

NN	Number of conditions for new words.
NS	Number of conditions for studied words.
I	Number of participants.
J	Number of items.
K	Number of response options.
muN	Mean of new-item distribution. If NN is greater than 1, then muN must be a vector of length NN.
s2aN	Variance of participant effects on mean of new-item distribution.
s2bN	Variance of item effects on mean of new-item distribution.

muS	Mean of studied-item distribution. If NS is greater than 1, then muS must be a vector of length NS.
s2aS	Variance of participant effects on mean of studied-item distribution.
s2bS	Variance of item effects on mean of studied-item distribution.
lagEffect	Linear slope of lag effect on log of studied-item scale.
shape	Common shape for both new and studied distributions.
crit	Matrix of criteria (not including -Inf or Inf). Columns correspond to criteria, rows correspond to participants.

Value

The function returns an internally defined "uvsdSim" structure.

Author(s)

Michael S. Pratte

References

See Pratte, Rouder, & Morey (2009)

See Also

hbmem

Examples

```
library(hbmem)
#Data from hiererchial model
sim=gammaSim()
slotNames(sim)
table(sim@resp,sim@cond,sim@Scond)

#Usefull to make data.frame for passing to model-fitting functions
dat=as.data.frame(cbind(sim@subj,sim@item,sim@cond,sim@Scond,sim@lag,sim@resp))
colnames(dat)=c("sub","item","cond","Scond","lag","resp")

table(dat$resp,dat$cond,dat$Scond)
```

normalSim

Function normalSim

Description

Simulates data from a hierarchical linear normal model.

Usage

```
normalSim(N=1, I=30, J=300, mu=0, s2a=.2, s2b=.2, muS2=0, s2aS2=0, s2bS2=0)
```

Arguments

N	Number of conditions.
I	Number of participants.
J	Number of items.
mu	Grand mean
s2a	Variance of subject effect on the mean
s2b	Variance of item effect on the mean
muS2	Overall variance of data on log scale
s2aS2	Variance of subject effect on variance
s2bS2	Variance of item effect on variance

Value

The function returns a data frame with subject (subj), item, lag, and response (resp) columns. Lag is a vector of zeros (i.e., no lag effect).

Author(s)

Michael S. Pratte

See Also

hbmem

Examples

```
library(hbmem)
I=20
J=50
R=I*J
dat=normalSim(I=I, J=J, mu=10, s2a=1, s2b=1, muS2=log(1), s2aS2=0, s2bS2=0)
summary(dat)
```

prn09

PRN09 Data

Description

Confidence ratings data from Pratte, Rouder, \& Morey (2009).

Usage

```
data(prn09)
```

Format

A flat-field data frame (each row is a trial) with the following variables

cond 0=new; 1=studied

sub index of subject starting at 0

item index of item starting at 0

lag index of lag, zero-centered

resp which response was made; 0="sure new"

Details

Participants studied a list of 240 words, and were then tested on the 240 studied and on 240 new words. At test, participants made one of six confidence ratings ranging from "sure new" to "sure studied". Note that to apply the models to these data the "Scond" variable should be set to "cond", and the "cond" variable should be all zeros. This is a backwards-compatibility issue.

Source

Pratte, Rouder, \& Morey (2009). Separating Mnemonic Process from Participant and Item Effects in the Assessment of ROC Asymmetries. *Journal of Experimental Psychology: Learning, Memory, and Cognition*.

Examples

```
library(hbmem)
data(prn09)
table(prn09$resp,prn09$cond)
#Turn it into data suitable for
#analysis with HBEM functions:
newdat=prn09
newdat$Scond=newdat$cond
newdat$cond=0
summary(newdat)
```

`rtgamma`*Function rtgamma*

Description

Returns random draws from truncated gamma distribution.

Usage

```
rtgamma(N, shape, scale, a, b)
```

Arguments

<code>N</code>	Number of samples.
<code>shape</code>	Shape of gamma distribution.
<code>scale</code>	Scale of gamma distribution.
<code>a</code>	Lower truncation point.
<code>b</code>	Upper truncation point.

`rtnorm`*Function rtnorm*

Description

Returns random samples from a truncated normal distribution.

Usage

```
rtnorm(N, mu, sigma, a, b)
```

Arguments

<code>N</code>	Number of samples to return.
<code>mu</code>	A vector of length <code>N</code> that contains distribution means for each draw.
<code>sigma</code>	A vector of length <code>N</code> that contains distribution standard deviations for each draw.
<code>a</code>	Vector of length <code>N</code> of lower truncation points; may be <code>-Inf</code> .
<code>b</code>	Vector of length <code>N</code> of upper truncation point; may be <code>Inf</code> .

Details

This function is currently unstable for drawing from regions with extremely low probabilities. If this happens it should print a warning, and return a draw from a uniform distribution between `a` and `b`. See example below for how to break it.

Value

Returns 'N' random draws.

Author(s)

Michael S. Pratte

See Also

hbmem

Examples

```
library(hbmem)
#Draw one
rtnorm(1,0,1,0,.2)

#Draw 50
N=500
mu=rep(0,N)
sigma=rep(1,N)
a=rep(1,N)
b=rep(2,N)
x=rtnorm(N,mu,sigma,a,b)
hist(x)

#Break it
rtnorm(1,0,1,1000,1001)
```

sampleGamma

Function sampleGamma

Description

Samples posterior of mean parameters of the hierarchical linear model on the log scale parameter of a gamma distribution. Usually used within an MCMC loop.

Usage

```
sampleGamma(sample, y, cond, subj, item,
lag, N, I, J, R, ncond, nsub, nitem, s2mu, s2a, s2b, met, shape,
sampLag, pos=FALSE)
```

Arguments

sample	Block of linear model parameters from previous iteration.
y	Vector of data
cond	Vector fo condition index, starting at zero.
subj	Vector of subject index, starting at zero.
item	Vector of item index, starting at zero.
lag	Vector of lag index, zero-centered.
N	Numer of conditions.
I	Number of subjects.
J	Number of items.
R	Total number of trials.
ncond	Vector of length (N) containing number of trials per condition.
nsub	Vector of length (I) containing number of trials per each subject.
nitem	Vector of length (J) containing number of trials per each item.
s2mu	Prior variance on the grand mean mu; usually set to some large number.
s2a	Shape parameter of inverse gamma prior placed on effect variances.
s2b	Rate parameter of inverse gamma prior placed on effect variances. Setting both s2a AND s2b to be small (e.g., .01, .01) makes this an uninformative prior.
met	Vector of tuning parameter for metropolis-hastings steps. Here, all sampling (except variances of alpha and beta) and decorrelating steps utilize the M-H sampling algorithm. This should be adjusted so that $.2 < b_0 < .6$.
shape	Single shape of Gamma distribution.
sampLag	Logical. Whether or not to sample the lag effect.
pos	Logical. If true, the model on scale is $1 + \exp(\mu + \alpha + \beta)$. That is, the scale is always greater than one.

Value

The function returns a list. The first element of the list is the newly sampled block of parameters. The second element contains a vector of 0s and 1s indicating which of the decorrelating steps were accepted.

Author(s)

Michael S. Pratte

See Also

hbmern

Examples

```

library(hbmem)
N=2
shape=2
I=30
J=50
R=I*J
#make some data
mu=log(c(1,2))
alpha=rnorm(I,0,.2)
beta=rnorm(J,0,.2)
theta=-.001
cond=sample(0:(N-1),R,replace=TRUE)
subj=rep(0:(I-1),each=J)
item=NULL
for(i in 1:I)
item=c(item,sample(0:(J-1),J,replace=FALSE))
lag=rnorm(R,0,100)
lag=lag-mean(lag)
resp=1:R
for(r in 1:R)
{
  scale=1+exp(mu[cond[r]+1]+alpha[subj[r]+1]+beta[item[r]+1]+theta*lag[r])
  resp[r]=rgamma(1,shape=shape,scale=scale)
}

ncond=table(cond)
nsub=table(subj)
nitem=table(item)

M=10
keep=2:M
B=N+I+J+3
s.block=matrix(0,nrow=M,ncol=B)
met=rep(.08,B)
b0=rep(0,B)
jump=.0005
for(m in 2:M)
{
  tmp=sampleGamma(s.block[m-1,],resp,cond,subj,item,lag,
N,I,J,R,ncond,nsub,nitem,5,.01,.01,met,2,1,pos=TRUE)
  s.block[m,]=tmp[[1]]
  b0=b0 + tmp[[2]]
  #Auto-tuning of metropolis decorrelating steps
  if(m>20 & m<min(keep))
  {
    met=met+(b0/m<.4)*rep(-jump,B) +(b0/m>.6)*rep(jump,B)
    met[met<jump]=jump
  }
  if(m==min(keep)) b0=rep(0,B)
}

```

```

b0/length(keep) #check acceptance rate

hbest=colMeans(s.block[keep,])

par(mfrow=c(2,2),pch=19,pty='s')
matplot(s.block[keep,1:N],t='l')
abline(h=mu,col="green")
acf(s.block[keep,1])
plot(hbest[(N+1):(I+N)]~alpha)
abline(0,1,col="green")
plot(hbest[(I+N+1):(I+J+N)]~beta)
abline(0,1,col="green")

#variance of participant effect
mean(s.block[keep,(N+I+J+1)])
#variance of item effect
mean(s.block[keep,(N+I+J+2)])
#estimate of lag effect
mean(s.block[keep,(N+I+J+3)])

```

sampleNorm

Function sampleNorm

Description

Samples posterior of mean parameters of the hierarchical linear normal model with a single Σ_2 . Usually used within an MCMC loop.

Usage

```

sampleNorm(sample, y, cond, subj, item, lag, N, I, J, R, ncond, nsub,
nitem, s2mu, s2a, s2b, meta, metb, sigma2, samplag=TRUE,Hier=TRUE)

```

Arguments

sample	Block of linear model parameters from previous iteration.
y	Vector of data
cond	Vector of condition index, starting at zero.
subj	Vector of subject index, starting at zero.
item	Vector of item index, starting at zero.
lag	Vector of lag index, zero-centered.
N	Number of conditions.
I	Number of subjects.

J	Number of items.
R	Total number of trials.
ncond	Vector of length (N) containing number of trials per each condition.
nsub	Vector of length (I) containing number of trials per each subject.
nitem	Vector of length (J) containing number of trials per each item.
s2mu	Prior variance on the grand mean mu; usually set to some large number.
s2a	Shape parameter of inverse gamma prior placed on effect variances.
s2b	Rate parameter of inverse gamma prior placed on effect variances. Setting both s2a AND s2b to be small (e.g., .01, .01) makes this an uninformative prior.
meta	Matrix of tuning parameter for metropolis-hastings decorrelating step on mu and alpha. This should be adjusted so that $.2 < b_0 < .6$.
metb	Tuning parameter for decorrelating step on alpha and beta.
sigma2	Variance of distribution.
sampLag	Logical. Whether or not to sample the lag effect.
Hier	Logical. If TRUE then effect variances are estimated from data. If FALSE then these values are set to whatever value is in the s2alpha and s2beta slots of sample. This should always be set to TRUE.

Value

The function returns a list. The first element of the list is the newly sampled block of parameters. The second element contains a vector of 0s and 1s indicating which of the decorrelating steps were accepted.

Author(s)

Michael S. Pratte

References

See Pratte, Rouder, & Morey (2009)

See Also

hbmem

Examples

```
library(hbmem)
N=2
t.mu=c(1,2)
I=20
J=50
R=I*J
#make some data
tmp=normalSim(N=N, I=I, J=J, mu=t.mu, s2a=2, s2b=2, muS2=log(1), s2aS2=0, s2bS2=0)
dat=tmp[[1]]
```

```

t.alpha=tmp[[2]]
t.beta=tmp[[3]]

ncond=table(dat$cond)
nsub=table(dat$sub)
nitem=table(dat$item)

M=10
keep=2:M
B=N+I+J+3
s.block=matrix(0,nrow=M,ncol=B)
met=c(.1,.1);b0=c(0,0)
jump=.001
for(m in 2:M)
{
tmp=sampleNorm(s.block[m-1,],dat$resp,dat$cond,dat$subj,dat$item,dat$lag,
N,I,J,R,ncond,nsub,nitem,5,.01,.01,met[1],met[2],1,1,1)
s.block[m,]=tmp[[1]]
b0=b0 + tmp[[2]]

#Auto-tuning of metropolis decorrelating steps
if(m>20 & m<min(keep))
{
met=met+(b0/m<.2)*c(-jump,-jump) +(b0/m>.3)*c(jump,jump)
met[met<jump]=jump
}
}

b0/M #check acceptance rate

hbest=colMeans(s.block[keep,])

par(mfrow=c(2,2),pch=19,pty='s')
matplot(s.block[keep,1:N],t='l')
abline(h=t.mu,col="green")
abline(h=tapply(dat$resp,dat$cond,mean),col="orange")
acf(s.block[keep,1])
plot(hbest[(N+1):(I+N)]~t.alpha)
abline(0,1,col="green")
plot(hbest[(I+N+1):(I+J+N)]~t.beta)
abline(0,1,col="green")

#variance of participant effect
mean(s.block[keep,(N+I+J+1)])
#variance of item effect
mean(s.block[keep,(N+I+J+2)])
#estimate of lag effect
mean(s.block[keep,(N+I+J+3)])

```

sampleNormb	<i>Function sampleNormb</i>
-------------	-----------------------------

Description

Same as sampleNorm, but assumes an additive model on sigma2, and takes the block of sigma2 parameters as argument

Usage

```
sampleNormb(sample, y, cond, subj, item, lag, N, I, J, R, ncond, nsub, nitem, s2mu, s2a, s2b, meta, metb, blockSigma2
```

Arguments

sample	Block of linear model parameters from previous iteration.
y	Vector of data
cond	Vector of condition index, starting at zero.
subj	Vector of subject index, starting at zero.
item	Vector of item index, starting at zero.
lag	Vector of lag index, zero-centered.
N	Number of conditions.
I	Number of subjects.
J	Number of items.
R	Total number of trials.
ncond	Vector of length (N) containing number of trials per each condition.
nsub	Vector of length (I) containing number of trials per each subject.
nitem	Vector of length (J) containing number of trials per each item.
s2mu	Prior variance on the grand mean mu; usually set to some large number.
s2a	Shape parameter of inverse gamma prior placed on effect variances.
s2b	Rate parameter of inverse gamma prior placed on effect variances. Setting both s2a AND s2b to be small (e.g., .01, .01) makes this an uninformative prior.
meta	Matrix of tuning parameter for metropolis-hastings decorrelating step on mu and alpha. This should be adjusted so that $.2 < b0 < .6$.
metb	Tuning parameter for decorrelating step on alpha and beta.
blockSigma2	Block of parameters for Sigma2 (on log scale). Like all blocks, first element is the overall mean, followed by participant effects and then item effects.
sampLag	Logical. Whether or not to sample the lag effect.
Hier	Logical. If TRUE then effect variances are estimated from data. If false, then these values are fixed to whatever is in the s2alpha and s2beta slots of sample. This value should always be TRUE unless you know what you are doing.

Value

The function returns a list. The first element of the list is the newly sampled block of parameters. The second element contains a vector of 0s or 1s indicating which of the decorrelating steps were accepted.

Author(s)

Michael S. Pratte

See Also

hbmemb, sampleSig2b

Examples

```
library(hbmemb)
N=2
I=50
J=200
B=N+I+J+3
R = I * J

mu=c(3,5)
muS2=log(c(1,2))
alpha = rnorm(I, 0, sqrt(.2))
beta = rnorm(J, 0, sqrt(.2))
alphaS2 = rnorm(I, 0, sqrt(.2))
betaS2 = rnorm(J, 0, sqrt(.2))
cond=sample(0:(N-1),R,replace=TRUE)
subj = rep(0:(I - 1), each = J)
item = rep(0:(J - 1), I)
lag = rep(0, R)
lag=runif(R,-500,500)
lag=lag-mean(lag)
resp = 1:R
for (r in 1:R) {
  mean = mu[cond[r] + 1] + alpha[subj[r] + 1] + beta[item[r] + 1]
  sd = sqrt(exp(muS2[cond[r]+1] + alphaS2[subj[r] + 1] +
betaS2[item[r] + 1] + .005*lag[r]))
  resp[r] = rnorm(1, mean, sd)
}
sim=(as.data.frame(cbind(cond,subj, item, lag, resp)))
attach(sim)
plot(resp~lag)

#####MCMC SETUP#####
blockS=blockS2=matrix(0,nrow=10,ncol=B)
blockS[,B-1]=blockS[,B-2]=blockS2[,B-1]=blockS2[,B-2]=.5
b0mean=c(0,0)
b0S2=rep(0,B)
met=rep(.01,B)
jump=.0001
```

```

ncond=table(cond)
nsub=table(subj)
nitem=table(item)

for(m in 2:10) #way to low for real analysis
{
  tmp=sampleNormb(blockS[m-1,],resp,cond,subj,item,lag,N,I,J,I*J,ncond,nsub,nitem,10,.01,.01,.02,.005,blockS2)
  blockS[m,]=tmp[[1]]
  b0mean=b0mean+tmp[[2]]

  tmp=sampleSig2b(blockS2[m-1,],resp,cond,subj,item,lag,N,I,J,I*J,ncond,nsub,nitem,10,.01,.01,met,blockS[m,],)
  blockS2[m,]=tmp[[1]]
  b0S2=b0S2+tmp[[2]]
  if(m<10) met=met+(b0S2/m<.3)*-jump +(b0S2/m>.5)*jump
  met[met<jump]=jump
#met[B]=.0001
}
b0mean/m
b0S2/m

s=colMeans(blockS)
s2=colMeans(blockS2)

par(mfrow=c(3,3))
matplot(blockS[,1:N],t='l')
abline(h=mu)
plot(s[(N+1):(I+N)]~alpha);abline(0,1)
plot(s[(I+N+1):(I+J+N)]~beta);abline(0,1)

matplot(blockS2[,1:N],t='l')
abline(h=muS2)
plot(s2[(N+1):(I+N)]~alphaS2);abline(0,1)
plot(s2[(I+N+1):(I+N+J)]~betaS2);abline(0,1)

plot(blockS2[,B-2],t='l')
plot(blockS2[,B-1],t='l')
plot(blockS2[,B],t='l')

```

sampleNormR

Function sampleNormR

Description

Samples posterior of mean parameters of the hierarchical linear normal model with the effects a linear function of some other variable.

Usage

```

sampleNormR(sample, phi,blockD,y,subj, item, lag, I, J, R, nsub, nitem,
             s2mu, s2a, s2b, meta, metb, sigma2, sampLag)

```

Arguments

sample	Block of linear model parameters from previous iteration.
y	Vector of data
phi	Vector of linear slopes on effects.
blockD	Block of parameters that will serve as the means of random effects
subj	Vector of subject index, starting at zero.
item	Vector of item index, starting at zero.
lag	Vector of lag index, zero-centered.
I	Number of subjects.
J	Number of items.
R	Total number of trials.
nsub	Vector of length (I) containing number of trials per each subject.
nitem	Vector of length (J) containing number of trials per each item.
s2mu	Prior variance on the grand mean mu; usually set to some large number.
s2a	Shape parameter of inverse gamma prior placed on effect variances.
s2b	Rate parameter of inverse gamma prior placed on effect variances. Setting both s2a AND s2b to be small (e.g., .01, .01) makes this an uninformative prior.
meta	Matrix of tuning parameter for metropolis-hastings decorrelating step on mu and alpha. This should be adjusted so that $.2 < b_0 < .6$.
metb	Tuning parameter for decorrelating step on alpha and beta.
sigma2	Variance of distribution.
sampLag	Logical. Whether or not to sample the lag effect.

Value

The function returns a list. The first element of the list is the newly sampled block of parameters. The THIRD element contains a vector of 0s and 1s indicating which of the decorrelating steps were accepted.

Author(s)

Michael S. Pratte

References

Not published yet.

See Also

hbmem

Examples

```

library(hbmem)

I=50
J=100
M=10
B=I+J+4
mu=.5
muS2=0
s2a=.2
s2b=.2
s2aS2=0
s2bS2=0

phi=c(.2,.08)
blockD=rep(0,B)
blockD[2:(I+1)]=rnorm(I,0,.5)
blockD[(I+2):(I+J+1)]=rnorm(J,0,.5)

R = I * J
alpha = rnorm(I, phi[1]*blockD[2:(I+1)], sqrt(s2a))
beta = rnorm(J, phi[2]*blockD[(I+2):(I+J+1)], sqrt(s2b))
alphaS2 = rnorm(I, 0, sqrt(s2aS2))
betaS2 = rnorm(J, 0, sqrt(s2bS2))
subj = rep(0:(I - 1), each = J)
item = rep(0:(J - 1), I)
lag = rep(0, R)
resp = 1:R
for (r in 1:R) {
  mean = mu + alpha[subj[r] + 1] + beta[item[r] + 1]
  sd = sqrt(exp(muS2 + alphaS2[subj[r] + 1] + betaS2[item[r] + 1]))
  resp[r] = rnorm(1, mean, sd)
}
sim=(as.data.frame(cbind(subj, item, lag, resp)))

blockR=matrix(0,M,B)
blockR[1,c(I+J+2,I+J+3)]=c(.1,.1)
met=c(.1,.1)
b0=c(0,0)

for(m in 2:M)
{
tmp=sampleNormR(blockR[m-1,],phi,blockD,sim$resp,sim$subj,sim$item,sim$lag,I,J,I*J,table(sim$subj),table(sim$item))
blockR[m,]=tmp[[1]]
b0=b0+tmp[[3]]
}

est=colMeans(blockR)

par(defpar(2,3))

```

```

plot(blockR[,1],t='l')
abline(h=mu,col="blue")
plot(blockR[,I+J+2],t='l')
abline(h=s2a,col="blue")
plot(blockR[,I+J+3],t='l')
abline(h=s2b,col="blue")

plot(est[2:(I+1)]~alpha);abline(0,1,col="blue")
plot(est[(I+2):(I+J+1)]~beta);abline(0,1,col="blue")

#Compare estimates from regular normal ones:

s.block=matrix(0,nrow=M,ncol=B)
met=c(.1,.1);b0=c(0,0)
for(m in 2:M)
{
tmp=sampleNorm(s.block[m-1,],sim$resp,rep(0,length(sim$resp)),sim$subj,sim$item,sim$lag,1,I,J,R,R,table(sim$subj,
table(sim$item),100,.01,.01,met[1],met[2],1,1)
s.block[m,]=tmp[[1]]
b0=b0 + tmp[[2]]
}

est2=colMeans(s.block)

par(defpar(1,2))
plot(est[2:(I+1)]~est2[2:(I+1)]);abline(0,1,col="blue")
plot(est[(I+2):(I+J+1)]~est2[(I+2):(I+J+1)]);abline(0,1,col="blue")

```

samplePosNorm

Function samplePosNorm

Description

Samples posterior of mean parameters of the positive hierarchical linear normal model with a single Sigma^2 $(x = N(\exp(\mu + \alpha_i + \beta_j), \text{sigma}^2))$.

Usage

```
samplePosNorm(sample, y, cond, sub, item, lag, N, I, J, R,
sig2mu, a, b, met, sigma2, samplag)
```

Arguments

sample	Block of linear model parameters from previous iteration.
y	Vector of data
cond	Vector of condition index.

sub	Vector of subject index, starting at zero.
item	Vector of item index, starting at zero.
lag	Vector of lag index, zero-centered.
N	Number of conditions.
I	Number of subjects.
J	Number of items.
R	Total number of trials.
sig2mu	Prior variance on the grand mean mu; usually set to some large number.
a	Shape parameter of inverse gamma prior placed on effect variances.
b	Rate parameter of inverse gamma prior placed on effect variances. Setting both s2a AND s2b to be small (e.g., .01, .01) makes this an uninformative prior.
met	Vector of tuning parameter for metropolis-hastings sampling. There is one tuning parameter for mu, each of I alphas, each of J betas, s2alpha,s2beta,and theta. Those for s2alpha and s2beta are placeholders, as these parameters are sampled with gibbs.
sigma2	Variance of distribution.
sampLag	Logical. Whether or not to sample the lag effect.

Value

The function returns a list. The first element of the list is the newly sampled block of parameters. The second element contains a vector of 0s and 1s indicating which of the decorrelating steps were accepted.

Author(s)

Michael S. Pratte

References

Not Published yet

See Also

hbmem

Examples

```
library(hbmem)

N=3
I=50
J=100
R=N*I*J
t.sigma2=3
t.mu=c(-1,0,1)
t.sig2alpha=.2
```

```

t.sig2beta=.6
t.alpha=rnorm(I,0,sqrt(t.sig2alpha))
t.beta =rnorm(J,0,sqrt(t.sig2beta))
t.theta=-.5
cond=sample((0:(N-1)),R,replace=TRUE)
sub=rep(rep(0:(I-1),each=J),N)
item=rep(rep(0:(J-1),I),N)
lag=scale(rnorm(R,0,sqrt(t.sigma2)/10))

tmean=1:R
for(r in 1:R) tmean[r]=exp(t.mu[cond[r]+1]+t.alpha[sub[r]+1]+t.beta[item[r]+1]+t.theta*lag[r])
y=rnorm(R,tmean,sqrt(t.sigma2))

M=10 #Way too low for real analysis!
B=N+I+J+3
block=matrix(0,nrow=M,ncol=B)
met=rep(.1,B);jump=.0001
b0=rep(0,B)
keep=2:M
for(m in 2:M)
{
  tmp= samplePosNorm(block[m-1,],y,cond,sub,item,lag,N,I,J,R,1,.01,.01,met,t.sigma2,1)
  block[m,]=tmp[[1]]
  b0=b0+tmp[[2]]

  if(m<keep[1])
  {
    met=met+(b0/m<.3)*-jump +(b0/m>.5)*jump
    met[met<jump]=jump
  }
  #if(m%%100==0) print(m)
}

est=colMeans(block[keep,])
b0/M

par(mfrow=c(3,2))
est.mu=est[1:N]
matplot(exp(block[keep,1:N]),t='l',main="Mu",ylab="Mu")
abline(h=exp(t.mu),col="blue")
#abline(h=tapply(y,cond,mean),col="green")
acf(block[keep,1],main="ACF of Mu")

est.alpha=est[(N+1):(N+I)]
plot(est.alpha~t.alpha,ylab="Est. Alpha",xlab="True Alpha");abline(0,1)
est.beta=est[(N+I+1):(N+I+J)]
plot(est.beta~t.beta,ylab="Est. Beta",xlab="True Beta");abline(0,1)

est.theta=est[N+I+J+3]
plot(block[keep,(N+I+J+3)],t='l',main="Theta",ylab="Theta")
abline(h=t.theta,col="blue")

plot(density(block[keep,(N+I+J+1)]),col="red",main="Posterior of Variances",xlim=c(0,1))

```

```
abline(v=t.sig2alpha,col="red")
lines(density(block[keep,(N+I+J+2)]),col="blue")
abline(v=t.sig2beta,col="blue")
```

sampleSig2

Function sampleSig2

Description

Samples posterior of the variance of a normal distribution which has an additive structure on the mean, and a single variance for all values. Usually used within MCMC loop.

Usage

```
sampleSig2(sig2,block,y,cond,sub,item,lag,N,ncond,I,J,a,b)
```

Arguments

sig2	Sample of sig2 from previous iteration.
block	Vector of parameters for mean of distribution
y	Vector of data
cond	Vector that indexes condition (e.g., deep vs. shallow)
sub	Vector of subject index, starting at zero.
item	Vector of item index, starting at zero.
lag	Vector of lag index, zero-centered.
N	Number of conditions.
ncond	Number of trials per condition.
I	Number of subjects.
J	Number of items.
a	Shape parameter for inverse gamma prior on Sigma2.
b	Rate parameter for inverse gamma prior on Sigma2. Setting 'a' and 'b' to small values (e.g., .01, .01) makes the prior non-informative.

Value

The function returns a new sample of Sigma2.

Author(s)

Michael S. Pratte

See Also

hbmern

Examples

```

library(hbmern)
true.mean=c(0,0)
true.sigma2=c(10,20)
N=2
I=1
J=1
R=1000
cond=rep(0:1,R/2)
ncond=table(cond)
sub=rep(0,R)
item=rep(0,R)
lag=rep(0,R)

#make some data
dat=rnorm(R,true.mean[cond+1],sqrt(true.sigma2[cond+1]))
true.block=c(true.mean,rep(0,(I+J+3)))

a=b=.01

M=10
s.sigma2=matrix(1,M,N)

for(m in 2:M)
{
s.sigma2[m,]=sampleSig2(s.sigma2[m-1,],true.block,dat,cond,sub,item,lag,N,ncond,I, J,a,b)
}

par(mfrow=c(1,1),pty='s')

matplot(s.sigma2,t='l')

abline(h=true.sigma2,col="blue")
abline(h=colMeans(s.sigma2),col="green") #post mean

```

sampleSig2b

*Function sampleSig2b***Description**

Samples posterior of the variance of a normal distribution which has the same additive structure on the mean and the log of variance. Usually used within MCMC loop.

Usage

```
sampleSig2b(sample,y,cond,sub,item,lag,N,I,J,R,ncond,nsub,nitem,s2mu,s2a,s2b,met,blockMean,sampLag=
```

Arguments

sample	Previous sample of block variances.
y	Vector of data
cond	Vector of condition index, starting at zero.
sub	Vector of subject index, starting at zero.
item	Vector of item index, starting at zero.
lag	Vector of lag index, zero-centered.
N	Number of conditions.
I	Number of subjects.
J	Number of items.
R	Total number of trials.
ncond	Vector of length (N) containing number of trials per each condition.
nsub	Vector of length (I) containing number of trials per each subject.
nitem	Vector of length (J) containing number of trials per each item.
s2mu	Prior variance on the grand mean mu; usually set to some large number.
s2a	Shape parameter of inverse gamma prior placed on effect variances.
s2b	Rate parameter of inverse gamma prior placed on effect variances. Setting both s2a AND s2b to be small (e.g., .01, .01) makes this an uninformative prior.
met	Vector of metropolis-hastings tuning parameters.
blockMean	Block of parameters for the mean of the distribution.
sampLag	Logical. Whether or not to sample the lag effect.
Hier	Logical. If TRUE then effect variances are estimated from data. If FALSE then these values are set to whatever value is in the s2alpha and s2beta slots of sample. This should always be set to TRUE.

Details

This function is for a model with an additive structure on the log of the variance of a normal distribution. This model is under development, the code is buggy, and it might not even work in the end.

Value

The function returns a new sample of a block of Sigma2 parameters.

Author(s)

Michael S. Pratte

See Also

hbmemb, sampleNormb

Examples

#See sampleNormb for example

uvsd-class	<i>Class "uvsd"</i>
------------	---------------------

Description

This class holds objects that are returned from uvsdSample.

Slots

muN: Object of class "numeric" ~~
 alphaN: Object of class "numeric" ~~
 betaN: Object of class "numeric" ~~
 s2alphaN: Object of class "numeric" ~~
 s2betaN: Object of class "numeric" ~~
 thetaN: Object of class "numeric" ~~
 muS: Object of class "numeric" ~~
 alphaS: Object of class "numeric" ~~
 betaS: Object of class "numeric" ~~
 s2alphaS: Object of class "numeric" ~~
 s2betaS: Object of class "numeric" ~~
 thetaS: Object of class "numeric" ~~
 estN: Object of class "numeric" ~~
 estS: Object of class "numeric" ~~
 estS2: Object of class "numeric" ~~
 estCrit: Object of class "matrix" ~~
 blockN: Object of class "matrix" ~~
 blockS: Object of class "matrix" ~~
 blockS2: Object of class "matrix" ~~
 s.crit: Object of class "array" ~~
 pD: Object of class "numeric" ~~
 DIC: Object of class "numeric" ~~
 M: Object of class "numeric" ~~
 keep: Object of class "numeric" ~~
 b0: Object of class "matrix" ~~
 b0S2: Object of class "numeric" ~~
 b0Crit: Object of class "numeric" ~~

uvsdLogLike

*Function uvsdLogLike***Description**

Computes log likelihood for UVSD model

Usage

```
uvsdLogLike(R, NN, NS, I, J, K, dat, cond, Scond, subj, item, lag, blockN, blockS, blockS2, crit)
```

Arguments

R	Total number of trials.
NN	Number of new-item conditions.
NS	Number of studied-item conditions.
I	Number of subjects.
J	Number of items.
K	Number of response options.
dat	Vector of responses, ranging from 0:(K-1).
cond	Vector of condition index.
Scond	Vector of new/studied condition index; 0=new, 1=studied.
subj	Vector of subject index, starting at 0 with no missing subject numbers.
item	Vector of item index, starting at 0 with no missing item numbers.
lag	Vector of lag index.
blockN	Block of parameters for new-item means.
blockS	Block of parameters for studied-item means.
blockS2	Block of parameters for Sigma2 values. If there is only one Sigma2 for all participants and items, then the first element of blockS2 should contain this value, and the other elements fo blockS2 should be zero.
crit	VECTOR of criteria including -Inf and Inf for top and bottom critieria, respectively. Vector contains the (K+1) criteria for the first subjects, followed by those for the second subject, etc.

Value

The function returns the log likelihood.

Author(s)

Michael S. Pratte

References

See Pratte, Rouder, & Morey (2009)

See Also

hbmemb

uvsdProbs

Function uvsdProbs

Description

Returns the probability of making confidence ratings given parameters of UVSD.

Usage

```
uvsdProbs(mean, sd, bounds)
```

Arguments

mean	Mean of the signal-detection distribution. In the common parameterization of the model, this would be zero for new-item trials, and d' for studied-item trials. In the PRM09 parameterization, these are d_n and d_s for new and studied-item trials, respectively.
sd	Standard deviation of the distribution
bounds	Criteria (not including $-\text{Inf}$ or Inf).

Value

The function returns the probability of making each response for the parameters given.

Author(s)

Michael S. Pratte

References

See Pratte, Rouder, & Morey (2009)

See Also

hbmemb

Examples

```
uvsdProbs(-1,1,c(-1,-.5,0,.5,1)) #New condition
uvsdProbs(1,1.3,c(-1,-.5,0,.5,1)) #Studied condition
```

uvsdSample	<i>Function uvsdSample</i>
------------	----------------------------

Description

Runs MCMC estimation for the hierarchical UVSD model.

Usage

```
uvsdSample(dat, M = 10000, keep = (M/10):M, getDIC = TRUE,
freeCrit=TRUE, equalVar=FALSE, freeSig2=FALSE, Hier=TRUE, jump=.0001)
```

Arguments

dat	Data frame that must include variables Scond,cond,sub,item,lag,resp. Scond indexes studied/new, whereas cond indexes conditions nested within the studied or new conditions. Indexes for Scond,cond, sub, item, and response must start at zero and have no gaps (i.e., no skipped subject numbers). Lags must be zero-centered.
M	Number of MCMC iterations.
keep	Which MCMC iterations should be included in estimates and returned. Use keep to both get ride of burn-in, and thin chains if necessary
getDIC	Logical. should the function compute DIC value? This takes a while if M is large.
freeCrit	Logical. If TRUE (default) individual criteria vary across people. If false, all participants have the same criteria. This should be set to false if there is only one participant, e.g., if averaging data over subjects.
equalVar	Logical. If FALSE (default), unequal-variance model is fit. If TRUE, equal-variance model is fit.
freeSig2	Logical. If FALSE (default), one sigma is fit for all participants and items (as in Pratte, et al., 2009). If TRUE, then an additive model is placed on the log of sigma2 (as in Pratte and Rouder (2010).
Hier	Logical. If TRUE then the variances of effects (e.g., item effects) are estimated from the data, i.e., effects are treated as random. If FALSE then these variances are fixed to 2.0 (.5 for recollection effects), thus treating these effects as fixed. This option is there to allow for compairson with more traditional approaches, and to see the effects of imposing hierarcical structure. It should always be set to TRUE in real analysis, and is not even guaranteed to work if set to false.
jump	The criteria and decorrelating steps utilize Matropolis-Hastings sampling routines, which require tuning. All MCMC functions should self tune during the burnin perior (iterations before keep), and they will alert you to the success of tuning. If acceptance rates are too low, "jump" should be decreased, if they are too high, "jump" should be increased. Alternatively, or in addition to adjusting "jump", simply increase the burnin period which will allow the function more time to self-tune.

Value

The function returns an internally defined "uvsd" S4 class that includes the following components

mu	Indexes which element of blocks contain grand means, mu
alpha	Indexes which element of blocks contain participant effects, alpha
beta	Indexes which element of blocks contain item effects, beta
s2alpha	Indexes which element of blocks contain variance of participant effects (alpha).
s2beta	Indexes which element of blocks contain variance of item effects (beta).
theta	Indexes which element of blocks contain theta, the slope of the lag effect
estN	Posterior means of block parameters for new-item means
estS	Posterior means of block parameters for studied-item means
estS2	Posterior means of block for studied-item variances.
estCrit	Posterior means of criteria
blockN	Each iteration for each parameter in the new-item mean block. Rows index iteration, columns index parameter.
blockS	Same as blockN, but for the studied-item means
blockS2	Same as blockN, but for variances of studied-item distribution. If equalVar=TRUE, then these values are all zero. If UVSD is fit but freeSig2=FALSE, then only the first element is non-zero (mu).
s.crit	Samples of each criteria.
pD	Number of effective parameters used in DIC. Note that this should be smaller than the actual number of parameters, as constraint from the hierarchical structure decreases the number of effective parameters.
DIC	DIC value. Smaller values indicate better fits. Note that DIC is notably biased toward complexity.
M	Number of MCMC iterations run
keep	MCMC iterations that were used for estimation and returned
b0	Metropolis-Hastings acceptance rates for decorrelating steps. These should be between .2 and .6. If they are not, the M, keep, or jump need to be adjusted.
b0S2	If additive model is placed on Sigma2 (i.e., freeSigma2=TRUE), then all parameters on S2 must be tuned. b0S2 are the acceptance probabilities for these parameters.

Author(s)

Michael S. Pratte

References

See Pratte, Rouder, & Morey (2009)

See Also

hbmemb

Examples

```

#In this example we generate data from UVSD with a different muN,muS,and
#Sigma2 for every person and item. These data are then fit with
#hierarchical UVSD allowing participant or item effects on log(sigma2).

library(hbmem)
sim=uvsdSim(NN=1,muN=-.5,NS=2,muS=c(.5,1),I=30,J=300,s2aN = .2, s2bN = .2,
muS2=log(c(1.3,1.5)),s2aS=.2,s2bS=.2,s2aS2=.2,s2bS2=.2)
dat=as.data.frame(cbind(sim@subj,sim@item,sim@cond,sim@Scond,sim@lag,sim@resp))
colnames(dat)=c("sub","item","cond","Scond","lag","resp")

M=10 #Way too low for real analysis
keep=2:M
uvsd=uvsdSample(dat,M=M,keep=keep,equalVar=FALSE,freeSig2=TRUE,jump=.0001,Hier=1)

par(mfrow=c(3,2),pch=19,pty='s')
#Look at chains of MuN and MuS
matplot(uvsd@blockN[,uvsd@muN],t='l',xlab="Iteration",ylab="Mu-N")
abline(h=sim@muN,col="blue")
matplot(uvsd@blockS[,uvsd@muS],t='l',xlab="Iteration",ylab="Mu-S")
abline(h=sim@muS,col="blue")

#Estimates of strength effects as function of true values
plot(uvsd@estN[uvsd@alphaN]~sim@alphaN,xlab="True
Alpha-N",ylab="Est. Alpha-N");abline(0,1,col="blue")
plot(uvsd@estS[uvsd@alphaS]~sim@alphaS,xlab="True
Alpha-S",ylab="Est. Alpha-S");abline(0,1,col="blue")
plot(uvsd@estN[uvsd@betaN]~sim@betaN,xlab="True
Beta-N",ylab="Est. Beta-N");abline(0,1,col="blue")
plot(uvsd@estS[uvsd@betaS]~sim@betaS,xlab="True
Beta-S",ylab="Est. Beta-S");abline(0,1,col="blue")

#Sigma^2 effects
#Note that Sigma^2 is biased high with
#few participants and items. This bias
#goes away with larger sample sizes.
par(mfrow=c(2,2),pch=19,pty='s')
matplot(sqrt(exp(uvsd@blockS2[,uvsd@muS])),t='l',xlab="Iteration",ylab="Mu-Sigma2")
abline(h=sqrt(exp(sim@muS2)),col="blue")
plot(uvsd@blockS2[,uvsd@thetaS],t='l')

plot(uvsd@estS2[uvsd@alphaS]~sim@alphaS2,xlab="True
Alpha-Sigma2",ylab="Est. Alpha-Sigma2");abline(0,1,col="blue")
plot(uvsd@estS2[uvsd@betaS]~sim@betaS2,xlab="True
Beta-Sigma2",ylab="Est. Beta-Sigma2");abline(0,1,col="blue")

#Look at some criteria
par(mfrow=c(2,2))
for(i in 1:4)
matplot(t(uvsd@s.crit[i,,]),t='l')

```

uvsdSim

*Function uvsdSim***Description**

Simulates data from a hierarchical UVSD model.

Usage

```
uvsdSim(NN = 2, NS = 1, I = 30, J = 200, K = 6, muN = c(-0.5,
  -0.2), s2aN = 0.2, s2bN = 0.2, muS = 0.5, s2aS = 0.2, s2bS = 0.2,
  muS2 = log(1), s2aS2 = 0, s2bS2 = 0, lagEffect = -0.001, crit = matrix(rep(c(-1.5, -0.5, 0, 0.5, 1.5),
```

Arguments

NN	Number of conditions for new words.
NS	Number of conditions for studied words.
I	Number of participants.
J	Number of items.
K	Number of response options.
muN	Mean of new-item distribution. If NN is greater than 1, then muN must be a vector of length NN.
s2aN	Variance of participant effects on mean of new-item distribution.
s2bN	Variance of item effects on mean of new-item distribution.
muS	Mean of studied-item distribution. If NS is greater than 1, then muS must be a vector of length NS.
s2aS	Variance of participant effects on mean of studied-item distribution.
s2bS	Variance of item effects on mean of studied-item distribution.
lagEffect	Magnitude of linear lag effect on both studied-item distribution and $\log(\sigma^2)$.
muS2	Mean variance of studied-item distribution, σ^2
s2aS2	Variance of participant effects σ^2 .
s2bS2	Variance of item effects on σ^2 .
crit	Matrix of criteria (not including -Inf or Inf). Columns correspond to criteria, rows correspond to participants.

Value

The function returns an internally defined "uvsdSim" structure.

Author(s)

Michael S. Pratte

References

See Pratte, Rouder, & Morey (2009)

See Also

hbmem

Examples

```
library(hbmem)
#Data from hiererchial model
sim=uvsdSim()
slotNames(sim)
table(sim@resp,sim@Scond,sim@cond)

#Usefull to make data.frame for passing to model-fitting functions
dat=as.data.frame(cbind(sim@subj,sim@item,sim@cond,sim@Scond,sim@lag,sim@resp))
colnames(dat)=c("sub","item","cond","Scond","lag","resp")

table(dat$resp,dat$Scond,dat$cond)
```

uvsdSim-class

Class "uvsdSim"

Description

Class that holds objects from function uvsdSim()

Slots

Scond: Object of class "numeric" ~~
cond: Object of class "numeric" ~~
subj: Object of class "numeric" ~~
item: Object of class "numeric" ~~
lag: Object of class "numeric" ~~
resp: Object of class "numeric" ~~
muN: Object of class "numeric" ~~
muS: Object of class "numeric" ~~
muS2: Object of class "numeric" ~~
alphaN: Object of class "numeric" ~~
betaN: Object of class "numeric" ~~
alphaS: Object of class "numeric" ~~
betaS: Object of class "numeric" ~~
alphaS2: Object of class "numeric" ~~
betaS2: Object of class "numeric" ~~
crit: Object of class "matrix" ~~

Index

*Topic **classes**

- dpsd-class, 4
- dpsdSim-class, 12
- uvsd-class, 40
- uvsdSim-class, 47

*Topic **datasets**

- prm09, 21

*Topic **models**

- dpsdRNSample, 6
- dpsdSample, 8
- dpsdSim, 11
- gammaSample, 16

*Topic **package**

- hbmemb-package, 2

- dpsd-class, 4
- dpsdProbs, 5
- dpsdRNSample, 6
- dpsdRNSim, 7
- dpsdSample, 8
- dpsdSim, 11
- dpsdSim-class, 12

- gammaLikeSample, 13
- gammaProbs, 15
- gammaSample, 16
- gammaSim, 18

- hbmemb (hbmemb-package), 2
- hbmemb-package, 2

- normalSim, 19

- prm09, 21

- rtgamma, 22
- rtnorm, 22

- sampleGamma, 23
- sampleNorm, 26
- sampleNormb, 29

- sampleNormR, 31
- samplePosNorm, 34
- sampleSig2, 37
- sampleSig2b, 38

- uvsd-class, 40
- uvsdLogLike, 41
- uvsdProbs, 42
- uvsdSample, 43
- uvsdSim, 46
- uvsdSim-class, 47