Package 'MCMCpack'

October 12, 2022

Version 1.6-3

Date 2022-04-13

Title Markov Chain Monte Carlo (MCMC) Package

Author Andrew D. Martin [aut], Kevin M. Quinn [aut], Jong Hee Park [aut,cre], Ghislain Vieille-dent [ctb], Michael Malecki[ctb], Matthew Black-well [ctb], Keith Poole [ctb], Craig Reed [ctb], Ben Goodrich [ctb], Qiushi Yu [ctb], Ross Ihaka [cph], The R Development Core Team [cph], The R Foundation [cph], Pierre L'Ecuyer [cph], Makoto Matsumoto [cph], Takuji Nishimura [cph]

Maintainer Jong Hee Park <jongheepark@snu.ac.kr>

Depends R (>= 3.6), coda (>= 0.11-3), MASS, stats

Imports graphics, grDevices, lattice, methods, utils, mcmc, quantreg

Description Contains functions to perform Bayesian inference using posterior simulation for a number of statistical models. Most simulation is done in compiled C++ written in the Scythe Statistical Library Version 1.0.3. All models return 'coda' mcmc objects that can then be summarized using the 'coda' package. Some useful utility functions such as density functions, pseudo-random number generators for statistical distributions, a general purpose Metropolis sampling algorithm, and tools for visualization are provided.

License GPL-3

SystemRequirements gcc (>= 4.0)

URL https://CRAN.R-project.org/package=MCMCpack

NeedsCompilation yes

Repository CRAN

RoxygenNote 7.1.2

Encoding UTF-8

Date/Publication 2022-04-13 11:12:32 UTC

R **topics documented:**

BayesFactor	*Create an object of class BayesFactor from MCMCpack output*

Description

This function creates an object of class `BayesFactor` from MCMCpack output.

Usage

```
BayesFactor(...)

is.BayesFactor(BF)
```

Arguments

`...`	MCMCpack output objects. These have to be of class `mcmc` and have a `logmarglike` attribute. In what follows, we let `M` denote the total number of models to be compared.
`BF`	An object to be checked for membership in class `BayesFactor`.

Value

An object of class `BayesFactor`. A `BayesFactor` object has four attributes. They are: `BF.mat` an $M \times M$ matrix in which element i, j contains the Bayes factor for model i relative to model j; `BF.log.mat` an $M \times M$ matrix in which element i, j contains the natural log of the Bayes factor for model i relative to model j; `BF.logmarglike` an M vector containing the log marginal likelihoods for models 1 through M; and `BF.call` an M element list containing the calls used to fit models 1 through M.

See Also

MCMCregress

Examples

```
## Not run:
data(birthwt)

model1 <- MCMCregress(bwt~age+lwt+as.factor(race) + smoke + ht,
                  data=birthwt, b0=c(2700, 0, 0, -500, -500,
                                        -500, -500),
                  B0=c(1e-6, .01, .01, 1.6e-5, 1.6e-5, 1.6e-5,
                       1.6e-5), c0=10, d0=4500000,
                  marginal.likelihood="Chib95", mcmc=10000)

model2 <- MCMCregress(bwt~age+lwt+as.factor(race) + smoke,
                  data=birthwt, b0=c(2700, 0, 0, -500, -500,
                                        -500),
                  B0=c(1e-6, .01, .01, 1.6e-5, 1.6e-5, 1.6e-5),
                  c0=10, d0=4500000,
                  marginal.likelihood="Chib95", mcmc=10000)

model3 <- MCMCregress(bwt~as.factor(race) + smoke + ht,
                  data=birthwt, b0=c(2700, -500, -500,
                                        -500, -500),
                  B0=c(1e-6, 1.6e-5, 1.6e-5, 1.6e-5,
                       1.6e-5), c0=10, d0=4500000,
                  marginal.likelihood="Chib95", mcmc=10000)

BF <- BayesFactor(model1, model2, model3)
print(BF)
```

```
## End(Not run)
```

---

choicevar   *Handle Choice-Specific Covariates in Multinomial Choice Models*

Description

This function handles choice-specific covariates in multinomial choice models. See the example for an example of useage.

Usage

```
choicevar(var, varname, choicelevel)
```

Arguments

var	The is the name of the variable in the dataframe.
varname	The name of the new variable to be created.
choicelevel	The level of y that the variable corresponds to.

Value

The new variable used by the MCMCmnl() function.

See Also

MCMCmnl

Dirichlet   *The Dirichlet Distribution*

Description

Density function and random generation from the Dirichlet distribution.

Usage

```
ddirichlet(x, alpha)

rdirichlet(n, alpha)
```

Arguments

x	A vector containing a single deviate or matrix containing one random deviate per row.
alpha	Vector of shape parameters, or matrix of shape parameters corresponding to the number of draw.
n	Number of random vectors to generate.

Details

The Dirichlet distribution is the multidimensional generalization of the beta distribution.

Value

ddirichlet gives the density. rdirichlet returns a matrix with n rows, each containing a single Dirichlet random deviate.

Author(s)

Code is taken from Greg's Miscellaneous Functions (gregmisc). His code was based on code posted by Ben Bolker to R-News on 15 Dec 2000.

See Also

Beta

Examples

```
density <- ddirichlet(c(.1,.2,.7), c(1,1,1))
draws <- rdirichlet(20, c(1,1,1) )
```

---

dtomogplot   *Dynamic Tomography Plot*

Description

dtomogplot is used to produce a tomography plot (see King, 1997) for a series of temporally ordered, partially observed 2 x 2 contingency tables.

Usage

```
dtomogplot(
  r0,
  r1,
  c0,
  c1,
  time.vec = NA,
  delay = 0,
  xlab = "fraction of r0 in c0 (p0)",
  ylab = "fraction of r1 in c0 (p1)",
  color.palette = heat.colors,
  bgcol = "black",
  ...
)
```

Arguments

r0	An $(ntables \times 1)$ vector of row sums from row 0.
r1	An $(ntables \times 1)$ vector of row sums from row 1.
c0	An $(ntables \times 1)$ vector of column sums from column 0.
c1	An $(ntables \times 1)$ vector of column sums from column 1.
time.vec	Vector of time periods that correspond to the elements of r_0, r_1, c_0, and c_1.
delay	Time delay in seconds between the plotting of the tomography lines. Setting a positive delay is useful for visualizing temporal dependence.
xlab	The x axis label for the plot.
ylab	The y axis label for the plot.
color.palette	Color palette to be used to encode temporal patterns.
bgcol	The background color for the plot.
...	further arguments to be passed

Details

Consider the following partially observed 2 by 2 contingency table:

	$\mid Y = 0$	$\mid Y = 1$	$\mid$
$X = 0$	$\mid Y_0$	$\mid$	$\mid r_0$
$X = 1$	$\mid Y_1$	$\mid$	$\mid r_1$
	$\mid c_0$	$\mid c_1$	$\mid N$

where r_0, r_1, c_0, c_1, and N are non-negative integers that are observed. The interior cell entries are not observed. It is assumed that $Y_0 | r_0 \sim \mathcal{B}inomial(r_0, p_0)$ and $Y_1 | r_1 \sim \mathcal{B}inomial(r_1, p_1)$.

This function plots the bounds on the maximum likelihood estimates for (p0, p1) and color codes

them by the elements of time.vec.

References

Gary King, 1997. *A Solution to the Ecological Inference Problem*. Princeton: Princeton University Press.

Jonathan C. Wakefield. 2004. "Ecological Inference for 2 x 2 Tables." *Journal of the Royal Statistical Society, Series A*. 167(3): 385445.

Kevin Quinn. 2004. "Ecological Inference in the Presence of Temporal Dependence." In *Ecological Inference: New Methodological Strategies*. Gary King, Ori Rosen, and Martin A. Tanner (eds.). New York: Cambridge University Press.

See Also

[MCMChierEI](), [MCMCdynamicEI](),[tomogplot]()

Examples

```
## Not run:
## simulated data example 1
set.seed(3920)
n <- 100
r0 <- rpois(n, 2000)
r1 <- round(runif(n, 100, 4000))
p0.true <- pnorm(-1.5 + 1:n/(n/2))
p1.true <- pnorm(1.0 - 1:n/(n/4))
y0 <- rbinom(n, r0, p0.true)
y1 <- rbinom(n, r1, p1.true)
c0 <- y0 + y1
c1 <- (r0+r1) - c0

## plot data
dtomogplot(r0, r1, c0, c1, delay=0.1)

## simulated data example 2
set.seed(8722)
n <- 100
r0 <- rpois(n, 2000)
r1 <- round(runif(n, 100, 4000))
p0.true <- pnorm(-1.0 + sin(1:n/(n/4)))
p1.true <- pnorm(0.0 - 2*cos(1:n/(n/9)))
y0 <- rbinom(n, r0, p0.true)
y1 <- rbinom(n, r1, p1.true)
c0 <- y0 + y1
c1 <- (r0+r1) - c0

## plot data
dtomogplot(r0, r1, c0, c1, delay=0.1)

## End(Not run)
```

Euro2016 *Euro 2016 data*

Description

Data on head-to-head outcomes from the 2016 UEFA European Football Championship.

Format

This dataframe contains all of the head-to-head results from Euro 2016. This includes results from both the group stage and the knock-out rounds.

dummy.rater An artificial "dummy" rater equal to 1 for all matches. Included so that Euro2016 can be used directly with MCMCpack's models for pairwise comparisons.

team1 The home team

team2 The away team

winner The winner of the match. NA if a draw.

Source

<https://en.wikipedia.org/wiki/UEFA_Euro_2016>

HDPHMMnegbin *Markov Chain Monte Carlo for sticky HDP-HMM with a Negative Binomial outcome distribution*

Description

This function generates a sample from the posterior distribution of a (sticky) HDP-HMM with a Negative Binomial outcome distribution (Fox et al, 2011). The user supplies data and priors, and a sample from the posterior distribution is returned as an mcmc object, which can be subsequently analyzed with functions provided in the coda package.

Usage

```
HDPHMMnegbin(
  formula,
  data = parent.frame(),
  K = 10,
  b0 = 0,
  B0 = 1,
  a.theta = 50,
  b.theta = 5,
```

```
      a.alpha = 1,
      b.alpha = 0.1,
      a.gamma = 1,
      b.gamma = 0.1,
      e = 2,
      f = 2,
      g = 10,
      burnin = 1000,
      mcmc = 1000,
      thin = 1,
      verbose = 0,
      seed = NA,
      beta.start = NA,
      P.start = NA,
      rho.start = NA,
      rho.step,
      nu.start = NA,
      gamma.start = 0.5,
      theta.start = 0.98,
      ak.start = 100,
      ...
    )
```

Arguments

formula	Model formula.
data	Data frame.
K	The number of regimes under consideration. This should be larger than the hypothesized number of regimes in the data. Note that the sampler will likely visit fewer than K regimes.
b0	The prior mean of β. This can either be a scalar or a column vector with dimension equal to the number of betas. If this takes a scalar value, then that value will serve as the prior mean for all of the betas.
B0	The prior precision of β. This can either be a scalar or a square matrix with dimensions equal to the number of betas. If this takes a scalar value, then that value times an identity matrix serves as the prior precision of beta. Default value of 0 is equivalent to an improper uniform prior for beta.
a.theta, b.theta	
	Paramaters for the Beta prior on θ, which captures the strength of the self-transition bias.
a.alpha, b.alpha	
	Shape and scale parameters for the Gamma distribution on $\alpha + \kappa$.
a.gamma, b.gamma	
	Shape and scale parameters for the Gamma distribution on γ.
e	The hyperprior for the distribution ρ See details.
f	The hyperprior for the distribution ρ. See details.

g	The hyperprior for the distribution ρ. See details.
burnin	The number of burn-in iterations for the sampler.
mcmc	The number of Metropolis iterations for the sampler.
thin	The thinning interval used in the simulation. The number of mcmc iterations must be divisible by this value.
verbose	A switch which determines whether or not the progress of the sampler is printed to the screen. If verbose is greater than 0 the iteration number, the current beta vector, and the Metropolis acceptance rate are printed to the screen every verboseth iteration.
seed	The seed for the random number generator. If NA, the Mersenne Twister generator is used with default seed 12345; if an integer is passed it is used to seed the Mersenne twister. The user can also pass a list of length two to use the L'Ecuyer random number generator, which is suitable for parallel computation. The first element of the list is the L'Ecuyer seed, which is a vector of length six or NA (if NA a default seed of rep(12345,6) is used). The second element of list is a positive substream number. See the MCMCpack specification for more details.
beta.start	The starting value for the β vector. This can either be a scalar or a column vector with dimension equal to the number of betas. If this takes a scalar value, then that value will serve as the starting value for all of the betas. The default value of NA will use the maximum likelihood estimate of β as the starting value for all regimes.
P.start	Initial transition matrix between regimes. Should be a K by K matrix. If not provided, the default value will be place theta.start along the diagonal and the rest of the mass even distributed within rows.
rho.start	The starting value for the ρ variable. This can either be a scalar or a column vector with dimension equal to the number of regimes. If the value is scalar, it will be used for all regimes. The default value is a vector of ones.
rho.step	Tuning parameter for the slice sampling approach to sampling rho. Determines the size of the step-out used to find the correct slice to draw from. Lower values are more accurate, but will take longer (up to a fixed searching limit). Default is 0.1.
nu.start	The starting values for the random effect, ν. The default value is a vector of ones.
theta.start, ak.start, gamma.start	
	Scalar starting values for the θ, $\alpha + \kappa$, and γ parameters.
...	further arguments to be passed.

Details

HDPHMMnegbin simulates from the posterior distribution of a sticky HDP-HMM with a Negative Binomial outcome distribution, allowing for multiple, arbitrary changepoints in the model. The details of the model are discussed in Blackwell (2017). The implementation here is based on a weak-limit approximation, where there is a large, though finite number of regimes that can be switched between. Unlike other changepoint models in MCMCpack, the HDP-HMM approach allows for the state sequence to return to previous visited states.

The model takes the following form, where we show the fixed-limit version:

$$y_t \sim \mathcal{P}oisson(\nu_t \mu_t)$$

$$\mu_t = x_t'\beta_m, \ \ m = 1, \ldots, M$$

$$\nu_t \sim \mathcal{G}amma(\rho_m, \rho_m)$$

Where M is an upper bound on the number of states and β_m and ρ_m are parameters when a state is m at t.

The transition probabilities between states are assumed to follow a heirarchical Dirichlet process:

$$\pi_m \sim \mathcal{D}irichlet(\alpha\delta_1, \ldots, \alpha\delta_j + \kappa, \ldots, \alpha\delta_M)$$

$$\delta \sim \mathcal{D}irichlet(\gamma/M, \ldots, \gamma/M)$$

The κ value here is the sticky parameter that encourages self-transitions. The sampler follows Fox et al (2011) and parameterizes these priors with $\alpha + \kappa$ and $\theta = \kappa/(\alpha + \kappa)$, with the latter representing the degree of self-transition bias. Gamma priors are assumed for $(\alpha + \kappa)$ and γ.

We assume Gaussian distribution for prior of β:

$$\beta_m \sim \mathcal{N}(b_0, B_0^{-1}), \ \ m = 1, \ldots, M$$

The overdispersion parameters have a prior with the following form:

$$f(\rho_m | e, f, g) \propto \rho^{e-1}(\rho + g)^{-(e+f)}$$

The model is simulated via blocked Gibbs conditonal on the states. The β being simulated via the auxiliary mixture sampling method of Fuerhwirth-Schanetter et al. (2009). The ρ is updated via slice sampling. The ν_i are updated their (conjugate) full conditional, which is also Gamma. The states are updated as in Fox et al (2011), supplemental materials.

Value

An mcmc object that contains the posterior sample. This object can be summarized by functions provided by the coda package.

References

Andrew D. Martin, Kevin M. Quinn, and Jong Hee Park. 2011. "MCMCpack: Markov Chain Monte Carlo in R.", *Journal of Statistical Software*. 42(9): 1-21. doi: 10.18637/jss.v042.i09.

Daniel Pemstein, Kevin M. Quinn, and Andrew D. Martin. 2007. *Scythe Statistical Library 1.0.* http://scythe.lsa.umich.edu.

Sylvia Fruehwirth-Schnatter, Rudolf Fruehwirth, Leonhard Held, and Havard Rue. 2009. "Improved auxiliary mixture sampling for hierarchical models of non-Gaussian data", *Statistics and Computing* 19(4): 479-492. <doi:10.1007/s11222-008-9109-4>

Matthew Blackwell. 2017. "Game Changers: Detecting Shifts in Overdispersed Count Data," *Political Analysis* 26(2), 230-239. <doi:10.1017/pan.2017.42>

Emily B. Fox, Erik B. Sudderth, Michael I. Jordan, and Alan S. Willsky. 2011.. "A sticky HDP-HMM with application to speaker diarization." *The Annals of Applied Statistics*, 5(2A), 1020-1056. <doi:10.1214/10-AOAS395>

See Also

MCMCnegbinChange, HDPHMMpoisson

Examples

```
## Not run:
  n <- 150
  reg <- 3
  true.s <- gl(reg, n/reg, n)
  rho.true <- c(1.5, 0.5, 3)
  b1.true <- c(1, -2, 2)
  x1 <- runif(n, 0, 2)
  nu.true <- rgamma(n, rho.true[true.s], rho.true[true.s])
  mu <- nu.true * exp(1 + x1 * b1.true[true.s])
  y <- rpois(n, mu)

  posterior <- HDPHMMnegbin(y ~ x1, K = 10, verbose = 1000,
                        e = 2, f = 2, g = 10,
                        a.theta = 100, b.theta = 1,
                        b0 = rep(0, 2), B0 = (1/9) * diag(2),
                        rho.step = rep(0.75, times = 10),
                        seed = list(NA, 2),
                        theta.start = 0.95, gamma.start = 10,
                        ak.start = 10)

  plotHDPChangepoint(posterior, ylab="Density", start=1)

## End(Not run)
```

HDPHMMpoisson   *Markov Chain Monte Carlo for sticky HDP-HMM with a Poisson out-come distribution*

Description

This function generates a sample from the posterior distribution of a (sticky) HDP-HMM with a Poisson outcome distribution (Fox et al, 2011). The user supplies data and priors, and a sample from the posterior distribution is returned as an mcmc object, which can be subsequently analyzed with functions provided in the coda package.

Usage

```
HDPHMMpoisson(
  formula,
  data = parent.frame(),
  K = 10,
  b0 = 0,
  B0 = 1,
  a.alpha = 1,
  b.alpha = 0.1,
  a.gamma = 1,
  b.gamma = 0.1,
  a.theta = 50,
  b.theta = 5,
  burnin = 1000,
  mcmc = 1000,
  thin = 1,
  verbose = 0,
  seed = NA,
  beta.start = NA,
  P.start = NA,
  gamma.start = 0.5,
  theta.start = 0.98,
  ak.start = 100,
  ...
)
```

Arguments

formula	Model formula.
data	Data frame.
K	The number of regimes under consideration. This should be larger than the hypothesized number of regimes in the data. Note that the sampler will likely visit fewer than K regimes.

b0   The prior mean of β. This can either be a scalar or a column vector with dimension equal to the number of betas. If this takes a scalar value, then that value will serve as the prior mean for all of the betas.

B0   The prior precision of β. This can either be a scalar or a square matrix with dimensions equal to the number of betas. If this takes a scalar value, then that value times an identity matrix serves as the prior precision of beta. Default value of 0 is equivalent to an improper uniform prior for beta.

a.alpha, b.alpha
           Shape and scale parameters for the Gamma distribution on $\alpha + \kappa$.

a.gamma, b.gamma
           Shape and scale parameters for the Gamma distribution on γ.

a.theta, b.theta
           Paramaters for the Beta prior on θ, which captures the strength of the self-transition bias.

burnin   The number of burn-in iterations for the sampler.

mcmc   The number of Metropolis iterations for the sampler.

thin   The thinning interval used in the simulation. The number of mcmc iterations must be divisible by this value.

verbose   A switch which determines whether or not the progress of the sampler is printed to the screen. If verbose is greater than 0 the iteration number, the current beta vector, and the Metropolis acceptance rate are printed to the screen every verboseth iteration.

seed   The seed for the random number generator. If NA, the Mersenne Twister generator is used with default seed 12345; if an integer is passed it is used to seed the Mersenne twister. The user can also pass a list of length two to use the L'Ecuyer random number generator, which is suitable for parallel computation. The first element of the list is the L'Ecuyer seed, which is a vector of length six or NA (if NA a default seed of rep(12345,6) is used). The second element of list is a positive substream number. See the MCMCpack specification for more details.

beta.start   The starting value for the β vector. This can either be a scalar or a column vector with dimension equal to the number of betas. If this takes a scalar value, then that value will serve as the starting value for all of the betas. The default value of NA will use the maximum likelihood estimate of β as the starting value for all regimes.

P.start   Initial transition matrix between regimes. Should be a K by K matrix. If not provided, the default value will be place theta.start along the diagonal and the rest of the mass even distributed within rows.

theta.start, ak.start, gamma.start
           Scalar starting values for the θ, $\alpha + \kappa$, and γ parameters.

...   further arguments to be passed.

Details

HDPHMMpoisson simulates from the posterior distribution of a sticky HDP-HMM with a Poisson outcome distribution, allowing for multiple, arbitrary changepoints in the model. The details of

the model are discussed in Blackwell (2017). The implementation here is based on a weak-limit approximation, where there is a large, though finite number of regimes that can be switched between. Unlike other changepoint models in MCMCpack, the HDP-HMM approach allows for the state sequence to return to previous visited states.

The model takes the following form, where we show the fixed-limit version:

$$y_t \sim \mathcal{P}oisson(\mu_t)$$

$$\mu_t = x_t' \beta_m, \quad m = 1, \ldots, M$$

Where M is an upper bound on the number of states and β_m are parameters when a state is m at t.

The transition probabilities between states are assumed to follow a heirarchical Dirichlet process:

$$\pi_m \sim \mathcal{D}irichlet(\alpha\delta_1, \ldots, \alpha\delta_j + \kappa, \ldots, \alpha\delta_M)$$

$$\delta \sim \mathcal{D}irichlet(\gamma/M, \ldots, \gamma/M)$$

The κ value here is the sticky parameter that encourages self-transitions. The sampler follows Fox et al (2011) and parameterizes these priors with $\alpha + \kappa$ and $\theta = \kappa/(\alpha + \kappa)$, with the latter representing the degree of self-transition bias. Gamma priors are assumed for $(\alpha + \kappa)$ and γ.

We assume Gaussian distribution for prior of β:

$$\beta_m \sim \mathcal{N}(b_0, B_0^{-1}), \quad m = 1, \ldots, M$$

The model is simulated via blocked Gibbs conditonal on the states. The β being simulated via the auxiliary mixture sampling method of Fuerhwirth-Schanetter et al. (2009). The states are updated as in Fox et al (2011), supplemental materials.

Value

An mcmc object that contains the posterior sample. This object can be summarized by functions provided by the coda package.

References

Andrew D. Martin, Kevin M. Quinn, and Jong Hee Park. 2011. "MCMCpack: Markov Chain Monte Carlo in R.", *Journal of Statistical Software*. 42(9): 1-21. doi: 10.18637/jss.v042.i09.

Daniel Pemstein, Kevin M. Quinn, and Andrew D. Martin. 2007. *Scythe Statistical Library 1.0.* http://scythe.lsa.umich.edu.

Sylvia Fruehwirth-Schnatter, Rudolf Fruehwirth, Leonhard Held, and Havard Rue. 2009. "Improved auxiliary mixture sampling for hierarchical models of non-Gaussian data", *Statistics and Computing* 19(4): 479-492. <doi:10.1007/s11222-008-9109-4>

Matthew Blackwell. 2017. "Game Changers: Detecting Shifts in Overdispersed Count Data," *Political Analysis* 26(2), 230-239. <doi:10.1017/pan.2017.42>

Emily B. Fox, Erik B. Sudderth, Michael I. Jordan, and Alan S. Willsky. 2011.. "A sticky HDP-HMM with application to speaker diarization." *The Annals of Applied Statistics*, 5(2A), 1020-1056. <doi:10.1214/10-AOAS395>

See Also

[MCMCpoissonChange](), [HDPHMMnegbin]()

Examples

```
## Not run:
  n <- 150
  reg <- 3
  true.s <- gl(reg, n/reg, n)
  b1.true <- c(1, -2, 2)
  x1 <- runif(n, 0, 2)
  mu <- exp(1 + x1 * b1.true[true.s])
  y <- rpois(n, mu)

  posterior <- HDPHMMpoisson(y ~ x1, K = 10, verbose = 1000,
                        a.theta = 100, b.theta = 1,
                        b0 = rep(0, 2), B0 = (1/9) * diag(2),
                        seed = list(NA, 2),
                        theta.start = 0.95, gamma.start = 10,
                        ak.start = 10)

  plotHDPChangepoint(posterior, ylab="Density", start=1)

## End(Not run)
```

HDPHSMMnegbin   *Markov Chain Monte Carlo for HDP-HSMM with a Negative Bino-mial outcome distribution*

Description

This function generates a sample from the posterior distribution of a Hidden Semi-Markov Model with a Heirarchical Dirichlet Process and a Negative Binomial outcome distribution (Johnson and Willsky, 2013). The user supplies data and priors, and a sample from the posterior distribution is returned as an mcmc object, which can be subsequently analyzed with functions provided in the coda package.

Usage

```
HDPHSMMnegbin(
  formula,
  data = parent.frame(),
  K = 10,
  b0 = 0,
  B0 = 1,
  a.alpha = 1,
  b.alpha = 0.1,
```

```
    a.gamma = 1,
    b.gamma = 0.1,
    a.omega,
    b.omega,
    e = 2,
    f = 2,
    g = 10,
    r = 1,
    burnin = 1000,
    mcmc = 1000,
    thin = 1,
    verbose = 0,
    seed = NA,
    beta.start = NA,
    P.start = NA,
    rho.start = NA,
    rho.step,
    nu.start = NA,
    omega.start = NA,
    gamma.start = 0.5,
    alpha.start = 100,
    ...
)
```

Arguments

formula	Model formula.
data	Data frame.
K	The number of regimes under consideration. This should be larger than the hypothesized number of regimes in the data. Note that the sampler will likely visit fewer than K regimes.
b0	The prior mean of β. This can either be a scalar or a column vector with dimension equal to the number of betas. If this takes a scalar value, then that value will serve as the prior mean for all of the betas.
B0	The prior precision of β. This can either be a scalar or a square matrix with dimensions equal to the number of betas. If this takes a scalar value, then that value times an identity matrix serves as the prior precision of beta. Default value of 0 is equivalent to an improper uniform prior for beta.
a.alpha, b.alpha	
	Shape and scale parameters for the Gamma distribution on α.
a.gamma, b.gamma	
	Shape and scale parameters for the Gamma distribution on γ.
a.omega, b.omega	
	Paramaters for the Beta prior on ω, which determines the regime length distribution, which is Negative Binomial, with parameters r and omega.
e	The hyperprior for the distribution ρ See details.

f	The hyperprior for the distribution ρ. See details.
g	The hyperprior for the distribution ρ. See details.
r	Parameter of the Negative Binomial prior for regime durations. It is the target number of successful trials. Must be strictly positive. Higher values increase the variance of the duration distributions.
burnin	The number of burn-in iterations for the sampler.
mcmc	The number of Metropolis iterations for the sampler.
thin	The thinning interval used in the simulation. The number of mcmc iterations must be divisible by this value.
verbose	A switch which determines whether or not the progress of the sampler is printed to the screen. If verbose is greater than 0 the iteration number, the current beta vector, and the Metropolis acceptance rate are printed to the screen every verboseth iteration.
seed	The seed for the random number generator. If NA, the Mersenne Twister generator is used with default seed 12345; if an integer is passed it is used to seed the Mersenne twister. The user can also pass a list of length two to use the L'Ecuyer random number generator, which is suitable for parallel computation. The first element of the list is the L'Ecuyer seed, which is a vector of length six or NA (if NA a default seed of rep(12345,6) is used). The second element of list is a positive substream number. See the MCMCpack specification for more details.
beta.start	The starting value for the β vector. This can either be a scalar or a column vector with dimension equal to the number of betas. If this takes a scalar value, then that value will serve as the starting value for all of the betas. The default value of NA will use the maximum likelihood estimate of β as the starting value for all regimes.
P.start	Initial transition matrix between regimes. Should be a K by K matrix. If not provided, the default value will be uniform transition distributions.
rho.start	The starting value for the ρ variable. This can either be a scalar or a column vector with dimension equal to the number of regimes. If the value is scalar, it will be used for all regimes. The default value is a vector of ones.
rho.step	Tuning parameter for the slice sampling approach to sampling rho. Determines the size of the step-out used to find the correct slice to draw from. Lower values are more accurate, but will take longer (up to a fixed searching limit). Default is 0.1.
nu.start	The starting values for the random effect, ν. The default value is a vector of ones.
omega.start	A vector of starting values for the probability of success parameter in the Negative Binomial distribution that governs the duration distributions.
alpha.start, gamma.start	
	Scalar starting values for the α, and γ parameters.
...	further arguments to be passed.

Details

HDPHSMMnegbin simulates from the posterior distribution of a HDP-HSMM with a Negative Bino-
mial outcome distribution, allowing for multiple, arbitrary changepoints in the model. The details of
the model are discussed in Johnson & Willsky (2013). The implementation here is based on a weak-
limit approximation, where there is a large, though finite number of regimes that can be switched
between. Unlike other changepoint models in MCMCpack, the HDP-HSMM approach allows for the
state sequence to return to previous visited states.

The model takes the following form, where we show the fixed-limit version:

$$y_t \sim \mathcal{P}oisson(\nu_t \mu_t)$$

$$\mu_t = x_t'\beta_k, \quad k = 1, \ldots, K$$

$$\nu_t \sim \mathcal{G}amma(\rho_k, \rho_k)$$

Where K is an upper bound on the number of states and β_k and ρ_k are parameters when a state is k
at t.

In the HDP-HSMM, there is a super-state sequence that, for a given observation, is drawn from the
transition distribution and then a duration is drawn from a duration distribution to determin how
long that state will stay active. After that duration, a new super-state is drawn from the transition
distribution, where self-transitions are disallowed. The transition probabilities between states are
assumed to follow a heirarchical Dirichlet process:

$$\pi_k \sim \mathcal{D}irichlet(\alpha\delta_1, \ldots, \alpha\delta_K)$$

$$\delta \sim \mathcal{D}irichlet(\gamma/K, \ldots, \gamma/K)$$

In the algorithm itself, these π vectors are modified to remove self-transitions as discussed above.
There is a unique duration distribution for each regime with the following parameters:

$$D_k \sim \mathcal{N}egBin(r, \omega_k)$$

$$\omega_k \sim \mathcal{B}eta(a_{\omega,k}, b_{\omega,k})$$

We assume Gaussian distribution for prior of β:

$$\beta_k \sim \mathcal{N}(b_0, B_0^{-1}), \quad m = 1, \ldots, K$$

The overdispersion parameters have a prior with the following form:

$$f(\rho_k|e, f, g) \propto \rho^{e-1}(\rho + g)^{-(e+f)}$$

The model is simulated via blocked Gibbs conditonal on the states. The β being simulated via the
auxiliary mixture sampling method of Fuerhwirth-Schanetter et al. (2009). The ρ is updated via
slice sampling. The ν_t are updated their (conjugate) full conditional, which is also Gamma. The
states and their durations are drawn as in Johnson & Willsky (2013).

Value

An mcmc object that contains the posterior sample. This object can be summarized by functions provided by the coda package.

References

Andrew D. Martin, Kevin M. Quinn, and Jong Hee Park. 2011. "MCMCpack: Markov Chain Monte Carlo in R.", *Journal of Statistical Software*. 42(9): 1-21. doi: 10.18637/jss.v042.i09.

Daniel Pemstein, Kevin M. Quinn, and Andrew D. Martin. 2007. *Scythe Statistical Library 1.0.* http://scythe.lsa.umich.edu.

Sylvia Fruehwirth-Schnatter, Rudolf Fruehwirth, Leonhard Held, and Havard Rue. 2009. "Improved auxiliary mixture sampling for hierarchical models of non-Gaussian data", *Statistics and Computing* 19(4): 479-492. <doi:10.1007/s11222-008-9109-4>

Matthew Blackwell. 2017. "Game Changers: Detecting Shifts in Overdispersed Count Data," *Political Analysis* 26(2), 230-239. <doi:10.1017/pan.2017.42>

Matthew J. Johnson and Alan S. Willsky. 2013. "Bayesian Nonparametric Hidden Semi-Markov Models." *Journal of Machine Learning Research*, 14(Feb), 673-701.

See Also

MCMCnegbinChange, HDPHMMnegbin,

Examples

```
## Not run:
  n <- 150
  reg <- 3
  true.s <- gl(reg, n/reg, n)
  rho.true <- c(1.5, 0.5, 3)
  b1.true <- c(1, -2, 2)
  x1 <- runif(n, 0, 2)
  nu.true <- rgamma(n, rho.true[true.s], rho.true[true.s])
  mu <- nu.true * exp(1 + x1 * b1.true[true.s])
  y <- rpois(n, mu)

  posterior <- HDPHSMMnegbin(y ~ x1, K = 10, verbose = 1000,
                        e = 2, f = 2, g = 10,
                        b0 = 0, B0 = 1/9,
                        a.omega = 1, b.omega = 100, r = 1,
                        rho.step = rep(0.75, times = 10),
                        seed = list(NA, 2),
                        omega.start = 0.05, gamma.start = 10,
                        alpha.start = 5)

  plotHDPChangepoint(posterior, ylab="Density", start=1)

## End(Not run)
```

| HMMpanelFE | *Markov Chain Monte Carlo for the Hidden Markov Fixed-effects Model* |

Description

HMMpanelFE generates a sample from the posterior distribution of the fixed-effects model with varying individual effects model discussed in Park (2011). The code works for both balanced and unbalanced panel data as long as there is no missing data in the middle of each group. This model uses a multivariate Normal prior for the fixed effects parameters and varying individual effects, an Inverse-Gamma prior on the residual error variance, and Beta prior for transition probabilities. The user supplies data and priors, and a sample from the posterior distribution is returned as an mcmc object, which can be subsequently analyzed with functions provided in the coda package.

Usage

```
HMMpanelFE(
  subject.id,
  y,
  X,
  m,
  mcmc = 1000,
  burnin = 1000,
  thin = 1,
  verbose = 0,
  b0 = 0,
  B0 = 0.001,
  c0 = 0.001,
  d0 = 0.001,
  delta0 = 0,
  Delta0 = 0.001,
  a = NULL,
  b = NULL,
  seed = NA,
  ...
)
```

Arguments

subject.id	A numeric vector indicating the group number. It should start from 1.
y	The response variable.
X	The model matrix excluding the constant.
m	A vector of break numbers for each subject in the panel.
mcmc	The number of MCMC iterations after burn-in.
burnin	The number of burn-in iterations for the sampler.

thin	The thinning interval used in the simulation. The number of MCMC iterations must be divisible by this value.
verbose	A switch which determines whether or not the progress of the sampler is printed to the screen. If verbose is greater than 0, the iteration number and the posterior density samples are printed to the screen every verboseth iteration.
b0	The prior mean of β. This can either be a scalar or a column vector with dimension equal to the number of betas. If this takes a scalar value, then that value will serve as the prior mean for all of the betas.
B0	The prior precision of β. This can either be a scalar or a square matrix with dimensions equal to the number of betas. If this takes a scalar value, then that value times an identity matrix serves as the prior precision of beta. Default value of 0 is equivalent to an improper uniform prior for beta.
c0	$c_0/2$ is the shape parameter for the inverse Gamma prior on σ^2 (the variance of the disturbances). The amount of information in the inverse Gamma prior is something like that from c_0 pseudo-observations.
d0	$d_0/2$ is the scale parameter for the inverse Gamma prior on σ^2 (the variance of the disturbances). In constructing the inverse Gamma prior, d_0 acts like the sum of squared errors from the c_0 pseudo-observations.
delta0	The prior mean of α.
Delta0	The prior precision of α.
a	a is the shape1 beta prior for transition probabilities. By default, the expected duration is computed and corresponding a and b values are assigned. The expected duration is the sample period divided by the number of states.
b	b is the shape2 beta prior for transition probabilities. By default, the expected duration is computed and corresponding a and b values are assigned. The expected duration is the sample period divided by the number of states.
seed	The seed for the random number generator. If NA, current R system seed is used.
...	further arguments to be passed

Details

HMMpanelFE simulates from the fixed-effect hidden Markov pbject level:

$$\varepsilon_{it} \sim \mathcal{N}(\alpha_{im}, \sigma_{im}^2)$$

We assume standard, semi-conjugate priors:

$$\beta \sim \mathcal{N}(b_0, B_0^{-1})$$

And:

$$\sigma^{-2} \sim \mathcal{G}amma(c_0/2, d_0/2)$$

And:

$$\alpha \sim \mathcal{N}(delta_0, Delta_0^{-1})$$

β, α and σ^{-2} are assumed *a priori* independent.

And:
$$p_{mm} \sim \mathcal{Beta}(a, b), \quad m = 1, \ldots, M$$

Where M is the number of states.

OLS estimates are used for starting values.

Value

An mcmc object that contains the posterior sample. This object can be summarized by functions provided by the coda package. The object contains an attribute sigma storage matrix that contains time-varying residual variance, an attribute state storage matrix that contains posterior samples of hidden states, and an attribute delta storage matrix containing time-varying intercepts.

References

Jong Hee Park, 2012. "Unified Method for Dynamic and Cross-Sectional Heterogeneity: Introducing Hidden Markov Panel Models." *American Journal of Political Science*.56: 1040-1054. <doi: 10.1111/j.1540-5907.2012.00590.x>

Siddhartha Chib. 1998. "Estimation and comparison of multiple change-point models." *Journal of Econometrics*. 86: 221-241. <doi: 10.1016/S0304-4076(97)00115-2>

Andrew D. Martin, Kevin M. Quinn, and Jong Hee Park. 2011. "MCMCpack: Markov Chain Monte Carlo in R.", *Journal of Statistical Software*. 42(9): 1-21. doi: 10.18637/jss.v042.i09.

Examples

```
## Not run:
  ## data generating
  set.seed(1974)
  N <- 30
  T <- 80
  NT <- N*T

  ## true parameter values
  true.beta <- c(1, 1)
  true.sigma <- 3
  x1 <- rnorm(NT)
  x2 <- runif(NT, 2, 4)

  ## group-specific breaks
  break.point = rep(T/2, N); break.sigma=c(rep(1, N));
  break.list <- rep(1, N)

  X <- as.matrix(cbind(x1, x2), NT, );
  y <- rep(NA, NT)
  id  <-  rep(1:N, each=NT/N)
  K <-  ncol(X);
  true.beta <- as.matrix(true.beta, K, 1)

  ## compute the break probability
  ruler <- c(1:T)
```

```
    W.mat <- matrix(NA, T, N)
    for (i in 1:N){
      W.mat[, i] <- pnorm((ruler-break.point[i])/break.sigma[i])
    }
    Weight <- as.vector(W.mat)

    ## draw time-varying individual effects and sample y
    j = 1
    true.sigma.alpha <- 30
    true.alpha1 <- true.alpha2 <- rep(NA, N)
    for (i in 1:N){
      Xi <- X[j:(j+T-1), ]
      true.mean <- Xi  %*% true.beta
      weight <- Weight[j:(j+T-1)]
      true.alpha1[i] <- rnorm(1, 0, true.sigma.alpha)
      true.alpha2[i] <- -1*true.alpha1[i]
      y[j:(j+T-1)] <- ((1-weight)*true.mean + (1-weight)*rnorm(T, 0, true.sigma) +
            (1-weight)*true.alpha1[i]) +
            (weight*true.mean + weight*rnorm(T, 0, true.sigma) + weight*true.alpha2[i])
      j <- j + T
    }

    ## extract the standardized residuals from the OLS with fixed-effects
    FEols <- lm(y ~ X + as.factor(id) -1 )
    resid.all <- rstandard(FEols)
    time.id <- rep(1:80, N)

    ## model fitting
    G <- 100
    BF <- testpanelSubjectBreak(subject.id=id, time.id=time.id,
            resid= resid.all, max.break=3, minimum = 10,
            mcmc=G, burnin = G, thin=1, verbose=G,
            b0=0, B0=1/100, c0=2, d0=2, Time = time.id)

    ## get the estimated break numbers
    estimated.breaks <- make.breaklist(BF, threshold=3)

    ## model fitting
    out <- HMMpanelFE(subject.id = id, y, X=X, m =  estimated.breaks,
                mcmc=G, burnin=G, thin=1, verbose=G,
                b0=0, B0=1/100, c0=2, d0=2, delta0=0, Delta0=1/100)

    ## print out the slope estimate
    ## true values are 1 and 1
    summary(out)

    ## compare them with the result from the constant fixed-effects
    summary(FEols)

## End(Not run)
```

HMMpanelRE   *Markov Chain Monte Carlo for the Hidden Markov Random-effects*
                               Model

Description

HMMpanelRE generates a sample from the posterior distribution of the hidden Markov random-effects model discussed in Park (2011). The code works for panel data with the same starting point. The sampling of panel parameters is based on Algorithm 2 of Chib and Carlin (1999). This model uses a multivariate Normal prior for the fixed effects parameters and varying individual effects, an Inverse-Wishart prior on the random-effects parameters, an Inverse-Gamma prior on the residual error variance, and Beta prior for transition probabilities. The user supplies data and priors, and a sample from the posterior distribution is returned as an mcmc object, which can be subsequently analyzed with functions provided in the coda package.

Usage

```
HMMpanelRE(
  subject.id,
  time.id,
  y,
  X,
  W,
  m = 1,
  mcmc = 1000,
  burnin = 1000,
  thin = 1,
  verbose = 0,
  b0 = 0,
  B0 = 0.001,
  c0 = 0.001,
  d0 = 0.001,
  r0,
  R0,
  a = NULL,
  b = NULL,
  seed = NA,
  beta.start = NA,
  sigma2.start = NA,
  D.start = NA,
  P.start = NA,
  marginal.likelihood = c("none", "Chib95"),
  ...
)
```

Arguments

subject.id   A numeric vector indicating the group number. It should start from 1.

time.id	A numeric vector indicating the time unit. It should start from 1.
y	The dependent variable
X	The model matrix of the fixed-effects
W	The model matrix of the random-effects. W should be a subset of X.
m	The number of changepoints.
mcmc	The number of MCMC iterations after burn-in.
burnin	The number of burn-in iterations for the sampler.
thin	The thinning interval used in the simulation. The number of MCMC iterations must be divisible by this value.
verbose	A switch which determines whether or not the progress of the sampler is printed to the screen. If verbose is greater than 0, the iteration number and the posterior density samples are printed to the screen every verboseth iteration.
b0	The prior mean of β. This can either be a scalar or a column vector with dimension equal to the number of betas. If this takes a scalar value, then that value will serve as the prior mean for all of the betas.
B0	The prior precision of β. This can either be a scalar or a square matrix with dimensions equal to the number of betas. If this takes a scalar value, then that value times an identity matrix serves as the prior precision of beta. Default value of 0 is equivalent to an improper uniform prior for beta.
c0	$c_0/2$ is the shape parameter for the inverse Gamma prior on σ^2 (the variance of the disturbances). The amount of information in the inverse Gamma prior is something like that from c_0 pseudo-observations.
d0	$d_0/2$ is the scale parameter for the inverse Gamma prior on σ^2 (the variance of the disturbances). In constructing the inverse Gamma prior, d_0 acts like the sum of squared errors from the c_0 pseudo-observations.
r0	The shape parameter for the Inverse-Wishart prior on variance matrix for the random effects. Set r=q for an uninformative prior where q is the number of random effects
R0	The scale matrix for the Inverse-Wishart prior on variance matrix for the random effects. This must be a square q-dimension matrix. Use plausible variance regarding random effects for the diagonal of R.
a	a is the shape1 beta prior for transition probabilities. By default, the expected duration is computed and corresponding a and b values are assigned. The expected duration is the sample period divided by the number of states.
b	b is the shape2 beta prior for transition probabilities. By default, the expected duration is computed and corresponding a and b values are assigned. The expected duration is the sample period divided by the number of states.
seed	The seed for the random number generator. If NA, current R system seed is used.
beta.start	The starting values for the beta vector. This can either be a scalar or a column vector with dimension equal to the number of betas. The default value of NA will use draws from the Uniform distribution with the same boundary with the data as the starting value. If this is a scalar, that value will serve as the starting value mean for all of the betas. When there is no covariate, the log value of means should be used.

sigma2.start   The starting values for σ^2. This can either be a scalar or a column vector with dimension equal to the number of states.

D.start   The starting values for the beta vector. This can either be a scalar or a column vector with dimension equal to the number of betas. The default value of NA will use draws from the Uniform distribution with the same boundary with the data as the starting value. If this is a scalar, that value will serve as the starting value mean for all of the betas. When there is no covariate, the log value of means should be used.

P.start   The starting values for the transition matrix. A user should provide a square matrix with dimension equal to the number of states. By default, draws from the `Beta(0.9,0.1)` are used to construct a proper transition matrix for each raw except the last raw.

marginal.likelihood
                   How should the marginal likelihood be calculated? Options are: none in which case the marginal likelihood will not be calculated and `Chib95` in which case the method of Chib (1995) is used.

...   further arguments to be passed

Details

`HMMpanelRE` simulates from the random-effect hidden Markov panel model introduced by Park (2011).

The model takes the following form:

$$y_i = X_i \beta_m + W_i b_i + \varepsilon_i \ \ m = 1, \dots, M$$

Where each group i have k_i observations. Random-effects parameters are assumed to be time-varying at the system level:

$$b_i \sim \mathcal{N}_q(0, D_m)$$
$$\varepsilon_i \sim \mathcal{N}(0, \sigma_m^2 I_{k_i})$$

And the errors: We assume standard, conjugate priors:

$$\beta \sim \mathcal{N}_p(b0, B0)$$

And:

$$\sigma^2 \sim \mathcal{IG}amma(c0/2, d0/2)$$

And:

$$D \sim \mathcal{IW}ishart(r0, R0)$$

See Chib and Carlin (1999) for more details.

And:

$$p_{mm} \sim \mathcal{B}eta(a, b), \ \ m = 1, \dots, M$$

Where M is the number of states.

NOTE: We do not provide default parameters for the priors on the precision matrix for the random effects. When fitting one of these models, it is of utmost importance to choose a prior that reflects your prior beliefs about the random effects. Using the `dwish` and `rwish` functions might be useful in choosing these values.

Value

An mcmc object that contains the posterior sample. This object can be summarized by functions provided by the coda package. The object contains an attribute prob.state storage matrix that contains the probability of $state_i$ for each period, and the log-marginal likelihood of the model (logmarglike).

References

Jong Hee Park, 2012. "Unified Method for Dynamic and Cross-Sectional Heterogeneity: Introducing Hidden Markov Panel Models." *American Journal of Political Science*.56: 1040-1054. <doi: 10.1111/j.1540-5907.2012.00590.x>

Siddhartha Chib. 1998. "Estimation and comparison of multiple change-point models." *Journal of Econometrics*. 86: 221-241. <doi: 10.1016/S0304-4076(97)00115-2>

Andrew D. Martin, Kevin M. Quinn, and Jong Hee Park. 2011. "MCMCpack: Markov Chain Monte Carlo in R.", *Journal of Statistical Software*. 42(9): 1-21. doi: 10.18637/jss.v042.i09.

Examples

```
## Not run:
  ## data generating
  set.seed(1977)
  Q <- 3
  true.beta1  <-  c(1, 1, 1) ; true.beta2   <-  c(-1, -1, -1)
  true.sigma2 <-  c(2, 5); true.D1 <- diag(.5, Q); true.D2 <- diag(2.5, Q)
  N=30; T=100;
  NT <- N*T
  x1 <- runif(NT, 1, 2)
  x2 <- runif(NT, 1, 2)
  X <- cbind(1, x1, x2);   W <- X;   y <- rep(NA, NT)

  ## true break numbers are one and at the center
  break.point = rep(T/2, N); break.sigma=c(rep(1, N));
  break.list <- rep(1, N)
  id  <-  rep(1:N, each=NT/N)
  K <-  ncol(X);
  ruler <- c(1:T)

  ## compute the weight for the break
  W.mat <- matrix(NA, T, N)
  for (i in 1:N){
    W.mat[, i] <- pnorm((ruler-break.point[i])/break.sigma[i])
  }
  Weight <- as.vector(W.mat)

  ## data generating by weighting two means and variances
  j = 1
  for (i in 1:N){
    Xi <- X[j:(j+T-1), ]
    Wi <- W[j:(j+T-1), ]
```

```
        true.V1 <- true.sigma2[1]*diag(T) + Wi%*%true.D1%*%t(Wi)
        true.V2 <- true.sigma2[2]*diag(T) + Wi%*%true.D2%*%t(Wi)
       true.mean1 <- Xi%*%true.beta1
       true.mean2 <- Xi%*%true.beta2
       weight <- Weight[j:(j+T-1)]
       y[j:(j+T-1)] <- (1-weight)*true.mean1 + (1-weight)*chol(true.V1)%*%rnorm(T) +
          weight*true.mean2 + weight*chol(true.V2)%*%rnorm(T)
       j <- j + T
     }
     ## model fitting
     subject.id <- c(rep(1:N, each=T))
     time.id <- c(rep(1:T, N))

     ## model fitting
     G <- 100
     b0  <- rep(0, K) ; B0  <- solve(diag(100, K))
     c0  <- 2; d0  <- 2
     r0  <- 5; R0  <- diag(c(1, 0.1, 0.1))
     subject.id <- c(rep(1:N, each=T))
     time.id <- c(rep(1:T, N))
     out1 <- HMMpanelRE(subject.id, time.id, y, X, W, m=1,
                       mcmc=G, burnin=G, thin=1, verbose=G,
                       b0=b0, B0=B0, c0=c0, d0=d0, r0=r0, R0=R0)

     ## latent state changes
     plotState(out1)

     ## print mcmc output
     summary(out1)



## End(Not run)
```

InvGamma   *The Inverse Gamma Distribution*

Description

Density function and random generation from the inverse gamma distribution.

Usage

```
dinvgamma(x, shape, scale = 1)

rinvgamma(n, shape, scale = 1)
```

Arguments

x	Scalar location to evaluate density.
shape	Scalar shape parameter.
scale	Scalar scale parameter (default value one).
n	Number of draws from the distribution.

Details

An inverse gamma random variable with shape a and scale b has mean $\frac{b}{a-1}$ (assuming $a > 1$) and variance $\frac{b^2}{(a-1)^2(a-2)}$ (assuming $a > 2$).

Value

`dinvgamma` evaluates the density at `x`.

`rinvgamma` takes `n` draws from the inverse Gamma distribution. The parameterization is consistent with the Gamma Distribution in the stats package.

References

Andrew Gelman, John B. Carlin, Hal S. Stern, and Donald B. Rubin. 2004. *Bayesian Data Analysis*. 2nd Edition. Boca Raton: Chapman & Hall.

See Also

GammaDist

Examples

```
density <- dinvgamma(4.2, 1.1)
draws <- rinvgamma(10, 3.2)
```

InvWishart   *The Inverse Wishart Distribution*

Description

Density function and random generation from the Inverse Wishart distribution.

Usage

```
riwish(v, S)

diwish(W, v, S)
```

Arguments

v	Degrees of freedom (scalar).
S	Scale matrix $(p \times p)$.
W	Positive definite matrix W $(p \times p)$.

Details

The mean of an inverse Wishart random variable with v degrees of freedom and scale matrix S is $(v - p - 1)^{-1}S$.

Value

diwish evaluates the density at positive definite matrix W. riwish generates one random draw from the distribution.

Examples

```
density <- diwish(matrix(c(2,-.3,-.3,4),2,2), 3, matrix(c(1,.3,.3,1),2,2))
draw <- riwish(3, matrix(c(1,.3,.3,1),2,2))
```

---

make.breaklist   *Vector of break numbers*

Description

This function generates a vector of break numbers using the output of testpanelSubjectBreak. The function performs a pairwise comparison of models using Bayes Factors.

Usage

```
make.breaklist(BF, threshold = 3)
```

Arguments

BF	output of testpanelSubjectBreak.
threshold	The Bayes Factor threshold to pick the best model. If a Bayes factor of two models is smaller than threshold, the model with a smaller number of break is chosen to avoid the over-identification problem. Users can change threshold into any positive number. The default value of 3 is chosen as it indicates the existence of "substantial evidence" in favor of the model in the numerator according to Jeffreys' scale.

Value

Vector fo break numbers.

References

Jong Hee Park, 2012. "Unified Method for Dynamic and Cross-Sectional Heterogeneity: Introducing Hidden Markov Panel Models." *American Journal of Political Science*.56: 1040-1054. <doi: 10.1111/j.1540-5907.2012.00590.x>

Harold Jeffreys, 1961. The Theory of Probability. Oxford University Press.

See Also

testpanelSubjectBreak

MCbinomialbeta	*Monte Carlo Simulation from a Binomial Likelihood with a Beta Prior*

Description

This function generates a sample from the posterior distribution of a binomial likelihood with a Beta prior.

Usage

```
MCbinomialbeta(y, n, alpha = 1, beta = 1, mc = 1000, ...)
```

Arguments

y	The number of successes in the independent Bernoulli trials.
n	The number of independent Bernoulli trials.
alpha	Beta prior distribution alpha parameter.
beta	Beta prior distribution beta parameter.
mc	The number of Monte Carlo draws to make.
...	further arguments to be passed

Details

MCbinomialbeta directly simulates from the posterior distribution. This model is designed primarily for instructional use. π is the probability of success for each independent Bernoulli trial. We assume a conjugate Beta prior:

$$\pi \sim \mathcal{B}eta(\alpha, \beta)$$

y is the number of successes in n trials. By default, a uniform prior is used.

Value

An mcmc object that contains the posterior sample. This object can be summarized by functions provided by the coda package.

See Also

[plot.mcmc](), [summary.mcmc]()

Examples

```
## Not run:
posterior <- MCbinomialbeta(3,12,mc=5000)
summary(posterior)
plot(posterior)
grid <- seq(0,1,0.01)
plot(grid, dbeta(grid, 1, 1), type="l", col="red", lwd=3, ylim=c(0,3.6),
  xlab="pi", ylab="density")
lines(density(posterior), col="blue", lwd=3)
legend(.75, 3.6, c("prior", "posterior"), lwd=3, col=c("red", "blue"))

## End(Not run)
```

MCMCbinaryChange   *Markov Chain Monte Carlo for a Binary Multiple Changepoint Model*

Description

This function generates a sample from the posterior distribution of a binary model with multiple changepoints. The function uses the Markov chain Monte Carlo method of Chib (1998). The user supplies data and priors, and a sample from the posterior distribution is returned as an mcmc object, which can be subsequently analyzed with functions provided in the coda package.

Usage

```
MCMCbinaryChange(
  data,
  m = 1,
  c0 = 1,
  d0 = 1,
  a = NULL,
  b = NULL,
  burnin = 10000,
  mcmc = 10000,
  thin = 1,
  verbose = 0,
  seed = NA,
  phi.start = NA,
  P.start = NA,
  marginal.likelihood = c("none", "Chib95"),
  ...
)
```

Arguments

`data`	The data.
`m`	The number of changepoints.
`c0`	c_0 is the shape1 parameter for Beta prior on ϕ (the mean).
`d0`	d_0 is the shape2 parameter for Beta prior on ϕ (the mean).
`a`	a is the shape1 beta prior for transition probabilities. By default, the expected duration is computed and corresponding a and b values are assigned. The expected duration is the sample period divided by the number of states.
`b`	b is the shape2 beta prior for transition probabilities. By default, the expected duration is computed and corresponding a and b values are assigned. The expected duration is the sample period divided by the number of states.
`burnin`	The number of burn-in iterations for the sampler.
`mcmc`	The number of MCMC iterations after burn-in.
`thin`	The thinning interval used in the simulation. The number of MCMC iterations must be divisible by this value.
`verbose`	A switch which determines whether or not the progress of the sampler is printed to the screen. If verbose is greater than 0, the iteration number and the posterior density samples are printed to the screen every verboseth iteration.
`seed`	The seed for the random number generator. If NA, current R system seed is used.
`phi.start`	The starting values for the mean. The default value of NA will use draws from the Uniform distribution.
`P.start`	The starting values for the transition matrix. A user should provide a square matrix with dimension equal to the number of states. By default, draws from the Beta(0.9,0.1) are used to construct a proper transition matrix for each raw except the last raw.
`marginal.likelihood`	
	How should the marginal likelihood be calculated? Options are: none in which case the marginal likelihood will not be calculated, and Chib95 in which case the method of Chib (1995) is used.
`...`	further arguments to be passed

Details

`MCMCbinaryChange` simulates from the posterior distribution of a binary model with multiple change-points.

The model takes the following form:

$$Y_t \sim \mathcal{B}ernoulli(\phi_i), \quad i = 1, \ldots, k$$

Where k is the number of states.

We assume Beta priors for ϕ_i and for transition probabilities:

$$\phi_i \sim \mathcal{B}eta(c_0, d_0)$$

And:
$$p_{mm} \sim \mathcal{B}eta a b, \quad m = 1, \ldots, k$$

Where M is the number of states.

Value

An mcmc object that contains the posterior sample. This object can be summarized by functions provided by the coda package. The object contains an attribute prob.state storage matrix that contains the probability of $state_i$ for each period, and the log-marginal likelihood of the model (logmarglike).

References

Jong Hee Park. 2011. "Changepoint Analysis of Binary and Ordinal Probit Models: An Application to Bank Rate Policy Under the Interwar Gold Standard." *Political Analysis*. 19: 188-204. <doi:10.1093/pan/mpr007>

Andrew D. Martin, Kevin M. Quinn, and Jong Hee Park. 2011. "MCMCpack: Markov Chain Monte Carlo in R.", *Journal of Statistical Software*. 42(9): 1-21. doi: 10.18637/jss.v042.i09.

Siddhartha Chib. 1995. "Marginal Likelihood from the Gibbs Output." *Journal of the American Statistical Association*. 90: 1313-1321. <doi: 10.1080/01621459.1995.10476635>

See Also

MCMCpoissonChange, plotState, plotChangepoint

Examples

```
## Not run:
set.seed(19173)
true.phi<- c(0.5, 0.8, 0.4)

## two breaks at c(80, 180)
y1 <- rbinom(80, 1,  true.phi[1])
y2 <- rbinom(100, 1, true.phi[2])
y3 <- rbinom(120, 1, true.phi[3])
y  <- as.ts(c(y1, y2, y3))

model0 <- MCMCbinaryChange(y, m=0, c0=2, d0=2, mcmc=100, burnin=100, verbose=50,
       marginal.likelihood = "Chib95")
model1 <- MCMCbinaryChange(y, m=1, c0=2, d0=2, mcmc=100, burnin=100, verbose=50,
       marginal.likelihood = "Chib95")
model2 <- MCMCbinaryChange(y, m=2, c0=2, d0=2, mcmc=100, burnin=100, verbose=50,
       marginal.likelihood = "Chib95")
model3 <- MCMCbinaryChange(y, m=3, c0=2, d0=2, mcmc=100, burnin=100, verbose=50,
       marginal.likelihood = "Chib95")
model4 <- MCMCbinaryChange(y, m=4, c0=2, d0=2, mcmc=100, burnin=100, verbose=50,
       marginal.likelihood = "Chib95")
model5 <- MCMCbinaryChange(y, m=5, c0=2, d0=2, mcmc=100, burnin=100, verbose=50,
       marginal.likelihood = "Chib95")
```

```
    print(BayesFactor(model0, model1, model2, model3, model4, model5))

    ## plot two plots in one screen
    par(mfrow=c(attr(model2, "m") + 1, 1), mai=c(0.4, 0.6, 0.3, 0.05))
    plotState(model2, legend.control = c(1, 0.6))
    plotChangepoint(model2, verbose = TRUE, ylab="Density", start=1, overlay=TRUE)


## End(Not run)
```

MCMCdynamicEI   *Markov Chain Monte Carlo for Quinn's Dynamic Ecological Infer-
                         ence Model*

Description

MCMCdynamicEI is used to fit Quinn's dynamic ecological inference model for partially observed
2 x 2 contingency tables.

Usage

```
MCMCdynamicEI(
  r0,
  r1,
  c0,
  c1,
  burnin = 5000,
  mcmc = 50000,
  thin = 1,
  verbose = 0,
  seed = NA,
  W = 0,
  a0 = 0.825,
  b0 = 0.0105,
  a1 = 0.825,
  b1 = 0.0105,
  ...
)
```

Arguments

r0 $(ntables \times 1)$ vector of row sums from row 0.

r1 $(ntables \times 1)$ vector of row sums from row 1.

c0 $(ntables \times 1)$ vector of column sums from column 0.

c1 $(ntables \times 1)$ vector of column sums from column 1.

burnin   The number of burn-in scans for the sampler.

mcmc   The number of mcmc scans to be saved.

thin   The thinning interval used in the simulation. The number of mcmc iterations
                must be divisible by this value.

verbose   A switch which determines whether or not the progress of the sampler is printed
                to the screen. If verbose is greater than 0 then every verboseth iteration will
                be printed to the screen.

seed   The seed for the random number generator. If NA, the Mersenne Twister gener-
                ator is used with default seed 12345; if an integer is passed it is used to seed the
                Mersenne twister. The user can also pass a list of length two to use the L'Ecuyer
                random number generator, which is suitable for parallel computation. The first
                element of the list is the L'Ecuyer seed, which is a vector of length six or NA
                (if NA a default seed of rep(12345,6) is used). The second element of list is a
                positive substream number. See the MCMCpack specification for more details.

W   Weight (*not precision*) matrix structuring the temporal dependence among ele-
                ments of θ_0 and θ_1. The default value of 0 will construct a weight matrix that
                corresponds to random walk priors for θ_0 and θ_1. The default assumes that the
                tables are equally spaced throughout time and that the elements of $r0$, $r1$, $c0$,
                and $c1$ are temporally ordered.

a0   a0/2 is the shape parameter for the inverse-gamma prior on the σ_0^2 parameter.

b0   b0/2 is the scale parameter for the inverse-gamma prior on the σ_0^2 parameter.

a1   a1/2 is the shape parameter for the inverse-gamma prior on the σ_1^2 parameter.

b1   b1/2 is the scale parameter for the inverse-gamma prior on the σ_1^2 parameter.

...   further arguments to be passed

Details

Consider the following partially observed 2 by 2 contingency table for unit t where $t = 1, \ldots, ntables$:

	$\mid Y = 0$	$\mid Y = 1$	$\mid$
$X = 0$	$\mid Y_{0t}$	$\mid$	$\mid r_{0t}$
$X = 1$	$\mid Y_{1t}$	$\mid$	$\mid r_{1t}$
	$\mid c_{0t}$	$\mid c_{1t}$	$\mid N_t$

Where r_{0t}, r_{1t}, c_{0t}, c_{1t}, and N_t are non-negative integers that are observed. The interior cell entries
are not observed. It is assumed that $Y_{0t}|r_{0t} \sim \mathcal{B}inomial(r_{0t}, p_{0t})$ and $Y_{1t}|r_{1t} \sim \mathcal{B}inomial(r_{1t}, p_{1t})$.
Let $\theta_{0t} = log(p_{0t}/(1 - p_{0t}))$, and $\theta_{1t} = log(p_{1t}/(1 - p_{1t}))$.

The following prior distributions are assumed:

$$p(\theta_0|\sigma_0^2) \propto \sigma_0^{-ntables} \exp\left(-\frac{1}{2\sigma_0^2}\theta_0' P\theta_0\right)$$

and

$$p(\theta_1|\sigma_1^2) \propto \sigma_1^{-ntables} \exp\left(-\frac{1}{2\sigma_1^2}\theta_1' P\theta_1\right)$$

where $P_{ts} = -W_{ts}$ for t not equal to s and $P_{tt} = \sum_{s\neq t} W_{ts}$. The θ_{0t} is assumed to be a priori independent of θ_{1t} for all t. In addition, the following hyperpriors are assumed: $\sigma_0^2 \sim \mathcal{IG}(a_0/2, b_0/2)$, and $\sigma_1^2 \sim \mathcal{IG}(a_1/2, b_1/2)$.

Inference centers on p_0, p_1, σ_0^2, and σ_1^2. Univariate slice sampling (Neal, 2003) together with Gibbs sampling is used to sample from the posterior distribution.

Value

An mcmc object that contains the sample from the posterior distribution. This object can be summarized by functions provided by the coda package.

References

Kevin Quinn. 2004. "Ecological Inference in the Presence of Temporal Dependence." In *Ecological Inference: New Methodological Strategies*. Gary King, Ori Rosen, and Martin A. Tanner (eds.). New York: Cambridge University Press.

Andrew D. Martin, Kevin M. Quinn, and Jong Hee Park. 2011. "MCMCpack: Markov Chain Monte Carlo in R", *Journal of Statistical Software*. 42(9): 1-21. doi: 10.18637/jss.v042.i09.

Radford Neal. 2003. "Slice Sampling" (with discussion). *Annals of Statistics*, 31: 705-767.

Daniel Pemstein, Kevin M. Quinn, and Andrew D. Martin. 2007. *Scythe Statistical Library 1.0*. http://scythe.lsa.umich.edu.

Martyn Plummer, Nicky Best, Kate Cowles, and Karen Vines. 2006. "Output Analysis and Diagnostics for MCMC (CODA)", *R News*. 6(1): 7-11. https://CRAN.R-project.org/doc/Rnews/Rnews_2006-1.pdf.

Jonathan C. Wakefield. 2004. "Ecological Inference for 2 x 2 Tables." *Journal of the Royal Statistical Society, Series A*. 167(3): 385445.

See Also

MCMChierEI, plot.mcmc, summary.mcmc

Examples

```
    ## Not run:
## simulated data example 1
set.seed(3920)
n <- 100
r0 <- rpois(n, 2000)
r1 <- round(runif(n, 100, 4000))
p0.true <- pnorm(-1.5 + 1:n/(n/2))
p1.true <- pnorm(1.0 - 1:n/(n/4))
y0 <- rbinom(n, r0, p0.true)
y1 <- rbinom(n, r1, p1.true)
```

```
c0 <- y0 + y1
c1 <- (r0+r1) - c0

## plot data
dtomogplot(r0, r1, c0, c1, delay=0.1)

## fit dynamic model
post1 <- MCMCdynamicEI(r0,r1,c0,c1, mcmc=40000, thin=5, verbose=100,
                       seed=list(NA, 1))

## fit exchangeable hierarchical model
post2 <- MCMChierEI(r0,r1,c0,c1, mcmc=40000, thin=5, verbose=100,
                    seed=list(NA, 2))

p0meanDyn <- colMeans(post1)[1:n]
p1meanDyn <- colMeans(post1)[(n+1):(2*n)]
p0meanHier <- colMeans(post2)[1:n]
p1meanHier <- colMeans(post2)[(n+1):(2*n)]

## plot truth and posterior means
pairs(cbind(p0.true, p0meanDyn, p0meanHier, p1.true, p1meanDyn, p1meanHier))


## simulated data example 2
set.seed(8722)
n <- 100
r0 <- rpois(n, 2000)
r1 <- round(runif(n, 100, 4000))
p0.true <- pnorm(-1.0 + sin(1:n/(n/4)))
p1.true <- pnorm(0.0 - 2*cos(1:n/(n/9)))
y0 <- rbinom(n, r0, p0.true)
y1 <- rbinom(n, r1, p1.true)
c0 <- y0 + y1
c1 <- (r0+r1) - c0

## plot data
dtomogplot(r0, r1, c0, c1, delay=0.1)

## fit dynamic model
post1 <- MCMCdynamicEI(r0,r1,c0,c1, mcmc=40000, thin=5, verbose=100,
                       seed=list(NA, 1))

## fit exchangeable hierarchical model
post2 <- MCMChierEI(r0,r1,c0,c1, mcmc=40000, thin=5, verbose=100,
                    seed=list(NA, 2))

p0meanDyn <- colMeans(post1)[1:n]
p1meanDyn <- colMeans(post1)[(n+1):(2*n)]
p0meanHier <- colMeans(post2)[1:n]
p1meanHier <- colMeans(post2)[(n+1):(2*n)]

## plot truth and posterior means
pairs(cbind(p0.true, p0meanDyn, p0meanHier, p1.true, p1meanDyn, p1meanHier))
```

```
## End(Not run)
```

MCMCdynamicIRT1d_b   *Markov Chain Monte Carlo for Dynamic One Dimensional Item Re-*
sponse Theory Model

Description

This function generates a sample from the posterior distribution of a dynamic one dimensional item
response theory (IRT) model, with Normal random walk priors on the subject abilities (ideal points),
and multivariate Normal priors on the item parameters. The user supplies data and priors, and a
sample from the posterior distribution is returned as an mcmc object, which can be subsequently
analyzed with functions provided in the coda package.

Usage

```
MCMCdynamicIRT1d_b(
  datamatrix,
  item.time.map,
  theta.constraints = list(),
  burnin = 1000,
  mcmc = 20000,
  thin = 1,
  verbose = 0,
  seed = NA,
  theta.start = NA,
  alpha.start = NA,
  beta.start = NA,
  tau2.start = 1,
  a0 = 0,
  A0 = 0.1,
  b0 = 0,
  B0 = 0.1,
  c0 = -1,
  d0 = -1,
  e0 = 0,
  E0 = 1,
  store.ability = TRUE,
  store.item = TRUE,
  ...
)

MCMCdynamicIRT1d(
  datamatrix,
  item.time.map,
```

```
    theta.constraints = list(),
    burnin = 1000,
    mcmc = 20000,
    thin = 1,
    verbose = 0,
    seed = NA,
    theta.start = NA,
    alpha.start = NA,
    beta.start = NA,
    tau2.start = 1,
    a0 = 0,
    A0 = 0.1,
    b0 = 0,
    B0 = 0.1,
    c0 = -1,
    d0 = -1,
    e0 = 0,
    E0 = 1,
    store.ability = TRUE,
    store.item = TRUE,
    ...
)
```

Arguments

datamatrix   The matrix of data. Must be 0, 1, or missing values. The rows of datamatrix
                      correspond to subjects and the columns correspond to items.

item.time.map   A vector that relates each item to a time period. Each element of item.time.map
                      gives the time period of the corresponding column of datamatrix. It is assumed
                      that the minimum value of item.time.map is 1.

theta.constraints
                      A list specifying possible simple equality or inequality constraints on the ability
                      parameters. A typical entry in the list has one of three forms: varname=c which
                      will constrain the ability parameter for the subject named varname to be equal
                      to c, varname="+" which will constrain the ability parameter for the subject
                      named varname to be positive, and varname="-" which will constrain the ability
                      parameter for the subject named varname to be negative. If x is a matrix without
                      row names defaults names of "V1","V2", ... , etc will be used. See Rivers (2003)
                      for a thorough discussion of identification of IRT models.

burnin   The number of burn-in iterations for the sampler.

mcmc   The number of Gibbs iterations for the sampler.

thin   The thinning interval used in the simulation. The number of Gibbs iterations
                      must be divisible by this value.

verbose   A switch which determines whether or not the progress of the sampler is printed
                      to the screen. If verbose is greater than 0 then every verboseth iteration will
                      be printed to the screen.

seed   The seed for the random number generator. If NA, the Mersenne Twister generator is used with default seed 12345; if an integer is passed it is used to seed the Mersenne twister. The user can also pass a list of length two to use the L'Ecuyer random number generator, which is suitable for parallel computation. The first element of the list is the L'Ecuyer seed, which is a vector of length six or NA (if NA a default seed of rep(12345,6) is used). The second element of list is a positive substream number. See the MCMCpack specification for more details.

theta.start   The starting values for the subject abilities (ideal points). This can either be a scalar or a column vector with dimension equal to the number of voters. If this takes a scalar value, then that value will serve as the starting value for all of the thetas. The default value of NA will choose the starting values based on an eigenvalue-eigenvector decomposition of the aggreement score matrix formed from the datamatrix.

alpha.start   The starting values for the α difficulty parameters. This can either be a scalar or a column vector with dimension equal to the number of items. If this takes a scalar value, then that value will serve as the starting value for all of the alphas. The default value of NA will set the starting values based on a series of probit regressions that condition on the starting values of theta.

beta.start   The starting values for the β discrimination parameters. This can either be a scalar or a column vector with dimension equal to the number of items. If this takes a scalar value, then that value will serve as the starting value for all of the betas. The default value of NA will set the starting values based on a series of probit regressions that condition on the starting values of theta.

tau2.start   The starting values for the evolution variances (the variance of the random walk increments for the ability parameters / ideal points. Order corresponds to the rows of datamatrix.

a0   A vector containing the prior mean of each of the difficulty parameters α. Should have as many elements as items / roll calls. Order corresponds to the columns of datamatrix. If a scalar is passed it is assumed that all elements of a0 are equal to the scalar.

A0   A vector containing the prior precision (inverse variance) of each of the difficulty parameters α. Should have as many elements as items / roll calls. Order corresponds to the columns of datamatrix. If a scalar is passed it is assumed that all elements of A0 are equal to the scalar.

b0   A vector containing the prior mean of each of the discrimination parameters β. Should have as many elements as items / roll calls. Order corresponds to the columns of datamatrix. If a scalar is passed it is assumed that all elements of b0 are equal to the scalar.

B0   A vector containing the prior precision (inverse variance) of each of the discrimination parameters β. Should have as many elements as items / roll calls. Order corresponds to the columns of datamatrix. If a scalar is passed it is assumed that all elements of B0 are equal to the scalar.

c0 $c_{0/2}$ is the shape parameter for the inverse Gamma prior on τ^2 (the variance of the random walk increments). The amount of information in the inverse Gamma prior is something like that from c_0 pseudo-observations. c0 can be either a vector with an element for each subject or a scalar. If c0 is negative then τ^2 is not estimated– the values in tau2.start are used throughout the sampling.

d0 $d_{0/2}$ is the scale parameter for the inverse Gamma prior on τ^2 (the variance of
                the random walk increments). In constructing the inverse Gamma prior, d_0 acts
                like the sum of squared errors from the c_0 pseudo-observations. d0 can be either
                a vector with an element for each subject or a scalar. If d0 is negative then τ^2 is
                not estimated– the values in tau2.start are used throughout the sampling.

e0   A vector containing the prior mean of the initial ability parameter / ideal point
                for each subject. Should have as many elements as subjects. Order corresponds
                to the rows of datamatrix. If a scalar is passed it is assumed that all elements
                of e0 are equal to the scalar.

E0   A vector containing the prior variance of the initial ability parameter / ideal point
                for each subject. Should have as many elements as subjects. Order corresponds
                to the rows of datamatrix. If a scalar is passed it is assumed that all elements
                of E0 are equal to the scalar.

store.ability   A switch that determines whether or not to store the ability parameters for pos-
                terior analysis. *NOTE*: In situations with many individuals storing the ability
                parameters takes an enormous amount of memory, so store.ability should
                only be TRUE if the chain is thinned heavily, or for applications with a small
                number of individuals. By default, the item parameters are stored.

store.item   A switch that determines whether or not to store the item parameters for poste-
                rior analysis. *NOTE*: In situations with many items storing the item parameters
                takes an enormous amount of memory, so store.item should only be FALSE if
                the chain is thinned heavily, or for applications with a small number of items.
                By default, the item parameters are not stored.

...   further arguments to be passed

Details

MCMCdynamicIRT1d simulates from the posterior distribution using the algorithm of Martin and
Quinn (2002). The simulation proper is done in compiled C++ code to maximize efficiency. Please
consult the coda documentation for a comprehensive list of functions that can be used to analyze
the posterior sample.

The model takes the following form. We assume that each subject has an subject ability (ideal point)
denoted $\theta_{j,t}$ (where j indexes subjects and t indexes time periods) and that each item has a difficulty
parameter α_i and discrimination parameter β_i. The observed choice by subject j on item i is the
observed data matrix which is $(I \times J)$. We assume that the choice is dictated by an unobserved
utility:

$$z_{i,j,t} = -\alpha_i + \beta_i \theta_{j,t} + \varepsilon_{i,j,t}$$

Where the disturbances are assumed to be distributed standard Normal. The parameters of interest
are the subject abilities (ideal points) and the item parameters.

We assume the following priors. For the subject abilities (ideal points):

$$\theta_{j,t} \sim \mathcal{N}(\theta_{j,t-1}, \tau_j^2)$$

with

$$\theta_{j,0} \sim \mathcal{N}(e0, E0)$$

.

The evolution variance has the following prior:

$$\tau_j^2 \sim \mathcal{IG}(c0/2, d0/2)$$

.

For the item parameters in the standard model, the prior is:

$$\alpha_i \sim \mathcal{N}(a0, A0^{-1})$$

and

$$\beta_i \sim \mathcal{N}(b0, B0^{-1})$$

.

The model is identified by the proper priors on the item parameters and constraints placed on the ability parameters.

As is the case with all measurement models, make sure that you have plenty of free memory, especially when storing the item parameters.

Value

An mcmc object that contains the posterior sample. This object can be summarized by functions provided by the coda package.

Author(s)

Kevin M. Quinn

References

Andrew D. Martin and Kevin M. Quinn. 2002. "Dynamic Ideal Point Estimation via Markov Chain Monte Carlo for the U.S. Supreme Court, 1953-1999." *Political Analysis.* 10: 134-153. <doi:10.1093/pan/10.2.134>

Andrew D. Martin, Kevin M. Quinn, and Jong Hee Park. 2011. "MCMCpack: Markov Chain Monte Carlo in R.", *Journal of Statistical Software*. 42(9): 1-21. doi: 10.18637/jss.v042.i09.

See Also

plot.mcmc,summary.mcmc, MCMCirt1d

Examples

```
   ## Not run:
data(Rehnquist)

## assign starting values
theta.start <- rep(0, 9)
theta.start[2] <- -3 ## Stevens
theta.start[7] <- 2  ## Thomas

out <- MCMCdynamicIRT1d(t(Rehnquist[,1:9]),
                        item.time.map=Rehnquist$time,
                        theta.start=theta.start,
                        mcmc=50000, burnin=20000, thin=5,
                        verbose=500, tau2.start=rep(0.1, 9),
                        e0=0, E0=1,
                        a0=0, A0=1,
                        b0=0, B0=1, c0=-1, d0=-1,
                        store.item=FALSE,
                        theta.constraints=list(Stevens="-", Thomas="+"))

summary(out)

## End(Not run)
```

MCMCfactanal | *Markov Chain Monte Carlo for Normal Theory Factor Analysis Model*

Description

This function generates a sample from the posterior distribution of a normal theory factor analysis model. Normal priors are assumed on the factor loadings and factor scores while inverse Gamma priors are assumed for the uniquenesses. The user supplies data and parameters for the prior distributions, and a sample from the posterior distribution is returned as an mcmc object, which can be subsequently analyzed with functions provided in the coda package.

Usage

```
MCMCfactanal(
  x,
  factors,
  lambda.constraints = list(),
  data = NULL,
  burnin = 1000,
  mcmc = 20000,
  thin = 1,
  verbose = 0,
```

```
        seed = NA,
        lambda.start = NA,
        psi.start = NA,
        l0 = 0,
        L0 = 0,
        a0 = 0.001,
        b0 = 0.001,
        store.scores = FALSE,
        std.var = TRUE,
        ...
    )
```

Arguments

x	Either a formula or a numeric matrix containing the manifest variables.
factors	The number of factors to be fitted.
lambda.constraints	
	List of lists specifying possible simple equality or inequality constraints on the factor loadings. A typical entry in the list has one of three forms: varname=list(d,c) which will constrain the dth loading for the variable named varname to be equal to c, varname=list(d,"+") which will constrain the dth loading for the variable named varname to be positive, and varname=list(d, "-") which will constrain the dth loading for the variable named varname to be negative. If x is a matrix without column names defaults names of "V1","V2", ... , etc will be used.
data	A data frame.
burnin	The number of burn-in iterations for the sampler.
mcmc	The number of iterations for the sampler.
thin	The thinning interval used in the simulation. The number of iterations must be divisible by this value.
verbose	A switch which determines whether or not the progress of the sampler is printed to the screen. If verbose is greater than 0 the iteration number and the factor loadings and uniquenesses are printed to the screen every verboseth iteration.
seed	The seed for the random number generator. If NA, the Mersenne Twister generator is used with default seed 12345; if an integer is passed it is used to seed the Mersenne twister. The user can also pass a list of length two to use the L'Ecuyer random number generator, which is suitable for parallel computation. The first element of the list is the L'Ecuyer seed, which is a vector of length six or NA (if NA a default seed of rep(12345,6) is used). The second element of list is a positive substream number. See the MCMCpack specification for more details.
lambda.start	Starting values for the factor loading matrix Lambda. If lambda.start is set to a scalar the starting value for all unconstrained loadings will be set to that scalar. If lambda.start is a matrix of the same dimensions as Lambda then the lambda.start matrix is used as the starting values (except for equality-constrained elements). If lambda.start is set to NA (the default) then starting values for unconstrained elements are set to 0, and starting values for inequality

constrained elements are set to either 0.5 or -0.5 depending on the nature of the constraints.

psi.start   Starting values for the uniquenesses. If `psi.start` is set to a scalar then the starting value for all diagonal elements of `Psi` are set to this value. If `psi.start` is a k-vector (where k is the number of manifest variables) then the staring value of `Psi` has `psi.start` on the main diagonal. If `psi.start` is set to `NA` (the default) the starting values of all the uniquenesses are set to 0.5.

l0   The means of the independent Normal prior on the factor loadings. Can be either a scalar or a matrix with the same dimensions as `Lambda`.

L0   The precisions (inverse variances) of the independent Normal prior on the factor loadings. Can be either a scalar or a matrix with the same dimensions as `Lambda`.

a0   Controls the shape of the inverse Gamma prior on the uniqueness. The actual shape parameter is set to `a0/2`. Can be either a scalar or a k-vector.

b0   Controls the scale of the inverse Gamma prior on the uniquenesses. The actual scale parameter is set to `b0/2`. Can be either a scalar or a k-vector.

store.scores   A switch that determines whether or not to store the factor scores for posterior analysis. *NOTE: This takes an enormous amount of memory, so should only be used if the chain is thinned heavily, or for applications with a small number of observations*. By default, the factor scores are not stored.

std.var   If `TRUE` (the default) the manifest variables are rescaled to have zero mean and unit variance. Otherwise, the manifest variables are rescaled to have zero mean but retain their observed variances.

...   further arguments to be passed

Details

The model takes the following form:

$$x_i = \Lambda\phi_i + \epsilon_i$$

$$\epsilon_i \sim \mathcal{N}(0, \Psi)$$

where x_i is the k-vector of observed variables specific to observation i, Λ is the $k \times d$ matrix of factor loadings, ϕ_i is the d-vector of latent factor scores, and Ψ is a diagonal, positive definite matrix. Traditional factor analysis texts refer to the diagonal elements of Ψ as uniquenesses.

The implementation used here assumes independent conjugate priors for each element of Λ each ϕ_i, and each diagonal element of Ψ. More specifically we assume:

$$\Lambda_{ij} \sim \mathcal{N}(l_{0_{ij}}, L_{0_{ij}}^{-1}), i = 1, \dots, k, j = 1, \dots, d$$

$$\phi_i \sim \mathcal{N}(0, I), i = 1, \dots, n$$

$$\Psi_{ii} \sim \mathcal{IG}(a_{0_i}/2, b_{0_i}/2), i = 1, \dots, k$$

`MCMCfactanal` simulates from the posterior distribution using standard Gibbs sampling. The simulation proper is done in compiled C++ code to maximize efficiency. Please consult the coda documentation for a comprehensive list of functions that can be used to analyze the posterior sample.

As is the case with all measurement models, make sure that you have plenty of free memory, especially when storing the scores.

Value

An mcmc object that contains the sample from the posterior distribution. This object can be summarized by functions provided by the coda package.

References

Andrew D. Martin, Kevin M. Quinn, and Jong Hee Park. 2011. "MCMCpack: Markov Chain Monte Carlo in R.", *Journal of Statistical Software*. 42(9): 1-21. doi: 10.18637/jss.v042.i09.

Daniel Pemstein, Kevin M. Quinn, and Andrew D. Martin. 2007. *Scythe Statistical Library 1.0.* http://scythe.lsa.umich.edu.

Martyn Plummer, Nicky Best, Kate Cowles, and Karen Vines. 2006. "Output Analysis and Diagnostics for MCMC (CODA)", *R News*. 6(1): 7-11. https://CRAN.R-project.org/doc/Rnews/Rnews_2006-1.pdf.

See Also

plot.mcmc,summary.mcmc,factanal

Examples

```
## Not run:
### An example using the formula interface
data(swiss)
posterior <- MCMCfactanal(~Agriculture+Examination+Education+Catholic
                +Infant.Mortality, factors=2,
                lambda.constraints=list(Examination=list(1,"+"),
                   Examination=list(2,"-"), Education=c(2,0),
                   Infant.Mortality=c(1,0)),
                verbose=0, store.scores=FALSE, a0=1, b0=0.15,
                data=swiss, burnin=5000, mcmc=50000, thin=20)
plot(posterior)
summary(posterior)

### An example using the matrix interface
Y <- cbind(swiss$Agriculture, swiss$Examination,
           swiss$Education, swiss$Catholic,
           swiss$Infant.Mortality)
colnames(Y) <- c("Agriculture", "Examination", "Education", "Catholic",
                "Infant.Mortality")
post <- MCMCfactanal(Y, factors=2,
                    lambda.constraints=list(Examination=list(1,"+"),
                       Examination=list(2,"-"), Education=c(2,0),
```

```
                         Infant.Mortality=c(1,0)),
                  verbose=0, store.scores=FALSE, a0=1, b0=0.15,
                  burnin=5000, mcmc=50000, thin=20)

## End(Not run)
```

MCMChierEI   *Markov Chain Monte Carlo for Wakefield's Hierarchial Ecological In-*
                            ference Model

Description

'MCMChierEI' is used to fit Wakefield's hierarchical ecological inference model for partially ob-
served 2 x 2 contingency tables.

Usage

```
MCMChierEI(
  r0,
  r1,
  c0,
  c1,
  burnin = 5000,
  mcmc = 50000,
  thin = 1,
  verbose = 0,
  seed = NA,
  m0 = 0,
  M0 = 2.287656,
  m1 = 0,
  M1 = 2.287656,
  a0 = 0.825,
  b0 = 0.0105,
  a1 = 0.825,
  b1 = 0.0105,
  ...
)
```

Arguments

r0	$(ntables \times 1)$ vector of row sums from row 0.
r1	$(ntables \times 1)$ vector of row sums from row 1.
c0	$(ntables \times 1)$ vector of column sums from column 0.
c1	$(ntables \times 1)$ vector of column sums from column 1.
burnin	The number of burn-in scans for the sampler.

mcmc	The number of mcmc scans to be saved.
thin	The thinning interval used in the simulation. The number of mcmc iterations must be divisible by this value.
verbose	A switch which determines whether or not the progress of the sampler is printed to the screen. If verbose is greater than 0 then every verboseth iteration will be printed to the screen.
seed	The seed for the random number generator. If NA, the Mersenne Twister generator is used with default seed 12345; if an integer is passed it is used to seed the Mersenne twister. The user can also pass a list of length two to use the L'Ecuyer random number generator, which is suitable for parallel computation. The first element of the list is the L'Ecuyer seed, which is a vector of length six or NA (if NA a default seed of rep(12345,6) is used). The second element of list is a positive substream number. See the MCMCpack specification for more details.
m0	Prior mean of the μ_0 parameter.
M0	Prior variance of the μ_0 parameter.
m1	Prior mean of the μ_1 parameter.
M1	Prior variance of the μ_1 parameter.
a0	a0/2 is the shape parameter for the inverse-gamma prior on the σ_0^2 parameter.
b0	b0/2 is the scale parameter for the inverse-gamma prior on the σ_0^2 parameter.
a1	a1/2 is the shape parameter for the inverse-gamma prior on the σ_1^2 parameter.
b1	b1/2 is the scale parameter for the inverse-gamma prior on the σ_1^2 parameter.
...	further arguments to be passed

Details

Consider the following partially observed 2 by 2 contingency table for unit t where $t = 1, \ldots, ntables$:

	$\mid Y = 0$	$\mid Y = 1$	$\mid$
$X = 0$	$\mid Y_{0t}$	$\mid$	$\mid r_{0t}$
$X = 1$	$\mid Y_{1t}$	$\mid$	$\mid r_{1t}$
	$\mid c_{0t}$	$\mid c_{1t}$	$\mid N_t$

Where r_{0t}, r_{1t}, c_{0t}, c_{1t}, and N_t are non-negative integers that are observed. The interior cell entries are not observed. It is assumed that $Y_{0t}|r_{0t} \sim \mathcal{B}inomial(r_{0t}, p_{0t})$ and $Y_{1t}|r_{1t} \sim \mathcal{B}inomial(r_{1t}, p_{1t})$. Let $\theta_{0t} = log(p_{0t}/(1 - p_{0t}))$, and $\theta_{1t} = log(p_{1t}/(1 - p_{1t}))$.

The following prior distributions are assumed: $\theta_{0t} \sim \mathcal{N}(\mu_0, \sigma_0^2)$, $\theta_{1t} \sim \mathcal{N}(\mu_1, \sigma_1^2)$. θ_{0t} is assumed to be a priori independent of θ_{1t} for all t. In addition, we assume the following hyperpriors: $\mu_0 \sim \mathcal{N}(m_0, M_0)$, $\mu_1 \sim \mathcal{N}(m_1, M_1)$, $\sigma_0^2 \sim \mathcal{IG}(a_0/2, b_0/2)$, and $\sigma_1^2 \sim \mathcal{IG}(a_1/2, b_1/2)$.

The default priors have been chosen to make the implied prior distribution for p_0 and p_1 *approximately* uniform on (0,1).

Inference centers on p_0, p_1, μ_0, μ_1, σ_0^2, and σ_1^2. Univariate slice sampling (Neal, 2003) along with Gibbs sampling is used to sample from the posterior distribution.

See Section 5.4 of Wakefield (2003) for discussion of the priors used here. `MCMChierEI` departs from the Wakefield model in that the `mu0` and `mu1` are here assumed to be drawn from independent normal distributions whereas Wakefield assumes they are drawn from logistic distributions.

Value

An mcmc object that contains the sample from the posterior distribution. This object can be summarized by functions provided by the coda package.

References

Jonathan C. Wakefield. 2004. "Ecological Inference for 2 x 2 Tables." *Journal of the Royal Statistical Society, Series A*. 167(3): 385445.

Radford Neal. 2003. "Slice Sampling" (with discussion). *Annals of Statistics*, 31: 705-767.

Andrew D. Martin, Kevin M. Quinn, and Jong Hee Park. 2011. "MCMCpack: Markov Chain Monte Carlo in R.", *Journal of Statistical Software*. 42(9): 1-21. doi: 10.18637/jss.v042.i09.

Daniel Pemstein, Kevin M. Quinn, and Andrew D. Martin. 2007. *Scythe Statistical Library 1.0.* http://scythe.lsa.umich.edu.

Martyn Plummer, Nicky Best, Kate Cowles, and Karen Vines. 2006. "Output Analysis and Diagnostics for MCMC (CODA)", *R News*. 6(1): 7-11. https://CRAN.R-project.org/doc/Rnews/Rnews_2006-1.pdf.

See Also

MCMCdynamicEI, plot.mcmc,summary.mcmc

Examples

```
   ## Not run:
## simulated data example
set.seed(3920)
n <- 100
r0 <- round(runif(n, 400, 1500))
r1 <- round(runif(n, 100, 4000))
p0.true <- pnorm(rnorm(n, m=0.5, s=0.25))
p1.true <- pnorm(rnorm(n, m=0.0, s=0.10))
y0 <- rbinom(n, r0, p0.true)
y1 <- rbinom(n, r1, p1.true)
c0 <- y0 + y1
c1 <- (r0+r1) - c0

## plot data
tomogplot(r0, r1, c0, c1)

## fit exchangeable hierarchical model
post <- MCMChierEI(r0,r1,c0,c1, mcmc=40000, thin=5, verbose=100,
```

```
                        seed=list(NA, 1))

p0meanHier <- colMeans(post)[1:n]
p1meanHier <- colMeans(post)[(n+1):(2*n)]

## plot truth and posterior means
pairs(cbind(p0.true, p0meanHier, p1.true, p1meanHier))

## End(Not run)
```

MCMChlogit   *Markov Chain Monte Carlo for the Hierarchical Binomial Linear Re-*
                           gression Model using the logit link function

---

Description

MCMChlogit generates a sample from the posterior distribution of a Hierarchical Binomial Linear
Regression Model using the logit link function and Algorithm 2 of Chib and Carlin (1999). This
model uses a multivariate Normal prior for the fixed effects parameters, an Inverse-Wishart prior
on the random effects variance matrix, and an Inverse-Gamma prior on the variance modelling
over-dispersion. The user supplies data and priors, and a sample from the posterior distribution is
returned as an mcmc object, which can be subsequently analyzed with functions provided in the
coda package.

Usage

```
MCMChlogit(
  fixed,
  random,
  group,
  data,
  burnin = 5000,
  mcmc = 10000,
  thin = 10,
  verbose = 1,
  seed = NA,
  beta.start = NA,
  sigma2.start = NA,
  Vb.start = NA,
  mubeta = 0,
  Vbeta = 1e+06,
  r,
  R,
  nu = 0.001,
  delta = 0.001,
  FixOD = 0,
  ...
)
```

Arguments

fixed	A two-sided linear formula of the form 'y~x1+...+xp' describing the fixed-effects part of the model, with the response on the left of a '~' operator and the p fixed terms, separated by '+' operators, on the right. Response variable y must be 0 or 1 (Binomial process).
random	A one-sided formula of the form '~x1+...+xq' specifying the model for the random effects part of the model, with the q random terms, separated by '+' operators.
group	String indicating the name of the grouping variable in data, defining the hierarchical structure of the model.
data	A data frame containing the variables in the model.
burnin	The number of burnin iterations for the sampler.
mcmc	The number of Gibbs iterations for the sampler. Total number of Gibbs iterations is equal to burnin+mcmc. burnin+mcmc must be divisible by 10 and superior or equal to 100 so that the progress bar can be displayed.
thin	The thinning interval used in the simulation. The number of mcmc iterations must be divisible by this value.
verbose	A switch (0,1) which determines whether or not the progress of the sampler is printed to the screen. Default is 1: a progress bar is printed, indicating the step (in %) reached by the Gibbs sampler.
seed	The seed for the random number generator. If NA, the Mersenne Twister generator is used with default seed 12345; if an integer is passed it is used to seed the Mersenne twister.
beta.start	The starting values for the β vector. This can either be a scalar or a p-length vector. The default value of NA will use the OLS β estimate of the corresponding Gaussian Linear Regression without random effects. If this is a scalar, that value will serve as the starting value mean for all of the betas.
sigma2.start	Scalar for the starting value of the residual error variance. The default value of NA will use the OLS estimates of the corresponding Gaussian Linear Regression without random effects.
Vb.start	The starting value for variance matrix of the random effects. This must be a square q-dimension matrix. Default value of NA uses an identity matrix.
mubeta	The prior mean of β. This can either be a scalar or a p-length vector. If this takes a scalar value, then that value will serve as the prior mean for all of the betas. The default value of 0 will use a vector of zeros for an uninformative prior.
Vbeta	The prior variance of β. This can either be a scalar or a square p-dimension matrix. If this takes a scalar value, then that value times an identity matrix serves as the prior variance of beta. Default value of 1.0E6 will use a diagonal matrix with very large variance for an uninformative flat prior.
r	The shape parameter for the Inverse-Wishart prior on variance matrix for the random effects. r must be superior or equal to q. Set r=q for an uninformative prior. See the NOTE for more details

R   The scale matrix for the Inverse-Wishart prior on variance matrix for the ran-
           dom effects. This must be a square q-dimension matrix. Use plausible variance
           regarding random effects for the diagonal of R. See the NOTE for more details

nu   The shape parameter for the Inverse-Gamma prior on the residual error variance.
           Default value is nu=delta=0.001 for uninformative prior.

delta   The rate (1/scale) parameter for the Inverse-Gamma prior on the residual error
           variance. Default value is nu=delta=0.001 for uninformative prior.

FixOD   A switch (0,1) which determines whether or not the variance for over-dispersion
           (sigma2) should be fixed (1) or not (0). Default is 0, parameter sigma2 is esti-
           mated. If FixOD=1, sigma2 is fixed to the value provided for sigma2.start.

...   further arguments to be passed

Details

MCMChlogit simulates from the posterior distribution sample using the blocked Gibbs sampler of
Chib and Carlin (1999), Algorithm 2. The simulation is done in compiled C++ code to maximize
efficiency. Please consult the coda documentation for a comprehensive list of functions that can be
used to analyze the posterior sample.

The model takes the following form:

$$y_i \sim \mathcal{B}ernoulli(\theta_i)$$

With latent variables $\phi(\theta_i)$, ϕ being the logit link function:

$$\phi(\theta_i) = X_i\beta + W_ib_i + \varepsilon_i$$

Where each group i have k_i observations.

Where the random effects:

$$b_i \sim \mathcal{N}_q(0, V_b)$$

And the over-dispersion terms:

$$\varepsilon_i \sim \mathcal{N}(0, \sigma^2 I_{k_i})$$

We assume standard, conjugate priors:

$$\beta \sim \mathcal{N}_p(\mu_\beta, V_\beta)$$

And:

$$\sigma^2 \sim \mathcal{IG}amma(\nu, 1/\delta)$$

And:

$$V_b \sim \mathcal{IW}ishart(r, rR)$$

See Chib and Carlin (1999) for more details.

NOTE: We do not provide default parameters for the priors on the precision matrix for the random effects. When fitting one of these models, it is of utmost importance to choose a prior that reflects your prior beliefs about the random effects. Using the `dwish` and `rwish` functions might be useful in choosing these values.

Value

mcmc   An mcmc object that contains the posterior sample. This object can be summa-
                rized by functions provided by the coda package. The posterior sample of the
                deviance D, with $D = -2 \log(\prod_i P(y_i|\theta_i))$, is also provided.

theta.pred   Predictive posterior mean for the inverse-logit of the latent variables. The ap-
                proximation of Diggle et al. (2004) is used to marginalized with respect to
                over-dispersion terms:

$$E[\theta_i|\beta, b_i, \sigma^2] = \phi^{-1}((X_i\beta + W_i b_i)/\sqrt{(16\sqrt{3}/15\pi)^2\sigma^2 + 1})$$

Author(s)

Ghislain Vieilledent <ghislain.vieilledent@cirad.fr>

References

Siddhartha Chib and Bradley P. Carlin. 1999. "On MCMC Sampling in Hierarchical Longitudinal Models." *Statistics and Computing.* 9: 17-26.

Daniel Pemstein, Kevin M. Quinn, and Andrew D. Martin. 2007. *Scythe Statistical Library 1.0.* http://scythe.lsa.umich.edu.

Andrew D. Martin and Kyle L. Saunders. 2002. "Bayesian Inference for Political Science Panel Data." Paper presented at the 2002 Annual Meeting of the American Political Science Association.

Martyn Plummer, Nicky Best, Kate Cowles, and Karen Vines. 2006. "Output Analysis and Diagnostics for MCMC (CODA)", *R News.* 6(1): 7-11. https://CRAN.R-project.org/doc/Rnews/Rnews_2006-1.pdf.

Diggle P., Heagerty P., Liang K., and Zeger S. 2004. "Analysis of Longitudinal Data." *Oxford University Press*, 2sd Edition.

See Also

plot.mcmc, summary.mcmc

Examples

```
## Not run:
#========================================
# Hierarchical Binomial Linear Regression
#========================================

#== inv.logit function
inv.logit <- function(x, min=0, max=1) {
    p <- exp(x)/(1+exp(x))
    p <- ifelse( is.na(p) & !is.na(x), 1, p ) # fix problems with +Inf
    return(p*(max-min)+min)
}

#== Generating data

# Constants
nobs <- 1000
nspecies <- 20
species <- c(1:nspecies,sample(c(1:nspecies),(nobs-nspecies),replace=TRUE))

# Covariates
X1 <- runif(n=nobs,min=-10,max=10)
X2 <- runif(n=nobs,min=-10,max=10)
X <- cbind(rep(1,nobs),X1,X2)
W <- X

# Target parameters
# beta
beta.target <- matrix(c(0.3,0.2,0.1),ncol=1)
# Vb
Vb.target <- c(0.5,0.05,0.05)
# b
b.target <- cbind(rnorm(nspecies,mean=0,sd=sqrt(Vb.target[1])),
                  rnorm(nspecies,mean=0,sd=sqrt(Vb.target[2])),
                  rnorm(nspecies,mean=0,sd=sqrt(Vb.target[3])))

# Response
theta <- vector()
Y <- vector()
for (n in 1:nobs) {
  theta[n] <- inv.logit(X[n,]%*%beta.target+W[n,]%*%b.target[species[n],])
  Y[n] <- rbinom(n=1,size=1,prob=theta[n])
}

# Data-set
Data <- as.data.frame(cbind(Y,theta,X1,X2,species))
plot(Data$X1,Data$theta)

#== Call to MCMChlogit
model <- MCMChlogit(fixed=Y~X1+X2, random=~X1+X2, group="species",
            data=Data, burnin=5000, mcmc=1000, thin=1,verbose=1,
```

```
              seed=NA, beta.start=0, sigma2.start=1,
              Vb.start=1, mubeta=0, Vbeta=1.0E6,
              r=3, R=diag(c(1,0.1,0.1)), nu=0.001, delta=0.001, FixOD=1)

#== MCMC analysis

# Graphics
pdf("Posteriors-MCMChlogit.pdf")
plot(model$mcmc)
dev.off()

# Summary
summary(model$mcmc)

# Predictive posterior mean for each observation
model$theta.pred

# Predicted-Observed
plot(Data$theta,model$theta.pred)
abline(a=0,b=1)

## #Not run
## #You can also compare with lme4 results
## #== lme4 resolution
## library(lme4)
## model.lme4 <- lmer(Y~X1+X2+(1+X1+X2|species),data=Data,family="binomial")
## summary(model.lme4)
## plot(fitted(model.lme4),model$theta.pred,main="MCMChlogit/lme4")
## abline(a=0,b=1)

## End(Not run)
```

MCMChpoisson	*Markov Chain Monte Carlo for the Hierarchical Poisson Linear Regression Model using the log link function*

Description

MCMChpoisson generates a sample from the posterior distribution of a Hierarchical Poisson Linear Regression Model using the log link function and Algorithm 2 of Chib and Carlin (1999). This model uses a multivariate Normal prior for the fixed effects parameters, an Inverse-Wishart prior on the random effects variance matrix, and an Inverse-Gamma prior on the variance modelling over-dispersion. The user supplies data and priors, and a sample from the posterior distribution is returned as an mcmc object, which can be subsequently analyzed with functions provided in the coda package.

Usage

```
MCMChpoisson(
```

```
    fixed,
    random,
    group,
    data,
    burnin = 5000,
    mcmc = 10000,
    thin = 10,
    verbose = 1,
    seed = NA,
    beta.start = NA,
    sigma2.start = NA,
    Vb.start = NA,
    mubeta = 0,
    Vbeta = 1e+06,
    r,
    R,
    nu = 0.001,
    delta = 0.001,
    FixOD = 0,
    ...
)
```

Arguments

fixed	A two-sided linear formula of the form 'y~x1+...+xp' describing the fixed-effects part of the model, with the response on the left of a '~' operator and the p fixed terms, separated by '+' operators, on the right. Response variable y must be 0 or 1 (Binomial process).
random	A one-sided formula of the form '~x1+...+xq' specifying the model for the random effects part of the model, with the q random terms, separated by '+' operators.
group	String indicating the name of the grouping variable in data, defining the hierarchical structure of the model.
data	A data frame containing the variables in the model.
burnin	The number of burnin iterations for the sampler.
mcmc	The number of Gibbs iterations for the sampler. Total number of Gibbs iterations is equal to burnin+mcmc. burnin+mcmc must be divisible by 10 and superior or equal to 100 so that the progress bar can be displayed.
thin	The thinning interval used in the simulation. The number of mcmc iterations must be divisible by this value.
verbose	A switch (0,1) which determines whether or not the progress of the sampler is printed to the screen. Default is 1: a progress bar is printed, indicating the step (in %) reached by the Gibbs sampler.
seed	The seed for the random number generator. If NA, the Mersenne Twister generator is used with default seed 12345; if an integer is passed it is used to seed the Mersenne twister.

beta.start   The starting values for the β vector. This can either be a scalar or a p-length
                      vector. The default value of NA will use the OLS β estimate of the correspond-
                      ing Gaussian Linear Regression without random effects. If this is a scalar, that
                      value will serve as the starting value mean for all of the betas.

sigma2.start   Scalar for the starting value of the residual error variance. The default value of
                      NA will use the OLS estimates of the corresponding Gaussian Linear Regression
                      without random effects.

Vb.start   The starting value for variance matrix of the random effects. This must be a
                      square q-dimension matrix. Default value of NA uses an identity matrix.

mubeta   The prior mean of β. This can either be a scalar or a p-length vector. If this takes
                      a scalar value, then that value will serve as the prior mean for all of the betas.
                      The default value of 0 will use a vector of zeros for an uninformative prior.

Vbeta   The prior variance of β. This can either be a scalar or a square p-dimension
                      matrix. If this takes a scalar value, then that value times an identity matrix
                      serves as the prior variance of beta. Default value of 1.0E6 will use a diagonal
                      matrix with very large variance for an uninformative flat prior.

r   The shape parameter for the Inverse-Wishart prior on variance matrix for the
                      random effects. r must be superior or equal to q. Set r=q for an uninformative
                      prior. See the NOTE for more details.

R   The scale matrix for the Inverse-Wishart prior on variance matrix for the ran-
                      dom effects. This must be a square q-dimension matrix. Use plausible variance
                      regarding random effects for the diagonal of R. See the NOTE for more details.

nu   The shape parameter for the Inverse-Gamma prior on the residual error variance.
                      Default value is nu=delta=0.001 for uninformative prior.

delta   The rate (1/scale) parameter for the Inverse-Gamma prior on the residual error
                      variance. Default value is nu=delta=0.001 for uninformative prior.

FixOD   A switch (0,1) which determines whether or not the variance for over-dispersion
                      (sigma2) should be fixed (1) or not (0). Default is 0, parameter sigma2 is esti-
                      mated. If FixOD=1, sigma2 is fixed to the value provided for sigma2.start.

...   further arguments to be passed

Details

MCMChpoisson simulates from the posterior distribution sample using the blocked Gibbs sampler of
Chib and Carlin (1999), Algorithm 2. The simulation is done in compiled C++ code to maximize
efficiency. Please consult the coda documentation for a comprehensive list of functions that can be
used to analyze the posterior sample.

The model takes the following form:

$$y_i \sim \mathcal{P}oisson(\lambda_i)$$

With latent variables $\phi(\lambda_i)$, ϕ being the log link function:

$$\phi(\lambda_i) = X_i\beta + W_ib_i + \varepsilon_i$$

Where each group i have k_i observations.

Where the random effects:

$$b_i \sim \mathcal{N}_q(0, V_b)$$

And the over-dispersion terms:

$$\varepsilon_i \sim \mathcal{N}(0, \sigma^2 I_{k_i})$$

We assume standard, conjugate priors:

$$\beta \sim \mathcal{N}_p(\mu_\beta, V_\beta)$$

And:

$$\sigma^2 \sim \mathcal{IG}amma(\nu, 1/\delta)$$

And:

$$V_b \sim \mathcal{IW}ishart(r, rR)$$

See Chib and Carlin (1999) for more details.

NOTE: We do not provide default parameters for the priors on the precision matrix for the random effects. When fitting one of these models, it is of utmost importance to choose a prior that reflects your prior beliefs about the random effects. Using the dwish and rwish functions might be useful in choosing these values.

Value

mcmc	An mcmc object that contains the posterior sample. This object can be summarized by functions provided by the coda package. The posterior sample of the deviance D, with $D = -2\log(\prod_i P(y_i|\lambda_i))$, is also provided.
lambda.pred	Predictive posterior mean for the exponential of the latent variables. The approximation of Diggle et al. (2004) is used to marginalized with respect to over-dispersion terms:

$$E[\lambda_i|\beta, b_i, \sigma^2] = \phi^{-1}((X_i\beta + W_i b_i) + 0.5\sigma^2)$$

Author(s)

Ghislain Vieilledent <ghislain.vieilledent@cirad.fr>

References

Siddhartha Chib and Bradley P. Carlin. 1999. "On MCMC Sampling in Hierarchical Longitudinal Models." *Statistics and Computing.* 9: 17-26.

Daniel Pemstein, Kevin M. Quinn, and Andrew D. Martin. 2007. *Scythe Statistical Library 1.0.* `http://scythe.lsa.umich.edu`.

Andrew D. Martin and Kyle L. Saunders. 2002. "Bayesian Inference for Political Science Panel Data." Paper presented at the 2002 Annual Meeting of the American Political Science Association.

Martyn Plummer, Nicky Best, Kate Cowles, and Karen Vines. 2006. "Output Analysis and Diagnostics for MCMC (CODA)", *R News.* 6(1): 7-11. `https://CRAN.R-project.org/doc/Rnews/Rnews_2006-1.pdf`.

See Also

`plot.mcmc`, `summary.mcmc`

Examples

```
## Not run:
#========================================
# Hierarchical Poisson Linear Regression
#========================================

#== Generating data

# Constants
nobs <- 1000
nspecies <- 20
species <- c(1:nspecies,sample(c(1:nspecies),(nobs-nspecies),replace=TRUE))

# Covariates
X1 <- runif(n=nobs,min=-1,max=1)
X2 <- runif(n=nobs,min=-1,max=1)
X <- cbind(rep(1,nobs),X1,X2)
W <- X

# Target parameters
# beta
beta.target <- matrix(c(0.1,0.1,0.1),ncol=1)
# Vb
Vb.target <- c(0.05,0.05,0.05)
# b
b.target <- cbind(rnorm(nspecies,mean=0,sd=sqrt(Vb.target[1])),
                  rnorm(nspecies,mean=0,sd=sqrt(Vb.target[2])),
                  rnorm(nspecies,mean=0,sd=sqrt(Vb.target[3])))

# Response
lambda <- vector()
Y <- vector()
for (n in 1:nobs) {
```

```
    lambda[n] <- exp(X[n,]%*%beta.target+W[n,]%*%b.target[species[n],])
    Y[n] <- rpois(1,lambda[n])
}

# Data-set
Data <- as.data.frame(cbind(Y,lambda,X1,X2,species))
plot(Data$X1,Data$lambda)

#== Call to MCMChpoisson
model <- MCMChpoisson(fixed=Y~X1+X2, random=~X1+X2, group="species",
                data=Data, burnin=5000, mcmc=1000, thin=1,verbose=1,
                seed=NA, beta.start=0, sigma2.start=1,
                Vb.start=1, mubeta=0, Vbeta=1.0E6,
                r=3, R=diag(c(0.1,0.1,0.1)), nu=0.001, delta=0.001, FixOD=1)

#== MCMC analysis

# Graphics
pdf("Posteriors-MCMChpoisson.pdf")
plot(model$mcmc)
dev.off()

# Summary
summary(model$mcmc)

# Predictive posterior mean for each observation
model$lambda.pred

# Predicted-Observed
plot(Data$lambda,model$lambda.pred)
abline(a=0,b=1)

## #Not run
## #You can also compare with lme4 results
## #== lme4 resolution
## library(lme4)
## model.lme4 <- lmer(Y~X1+X2+(1+X1+X2|species),data=Data,family="poisson")
## summary(model.lme4)
## plot(fitted(model.lme4),model$lambda.pred,main="MCMChpoisson/lme4")
## abline(a=0,b=1)

## End(Not run)
```

MCMChregress   *Markov Chain Monte Carlo for the Hierarchical Gaussian Linear Re-*
                            gression Model

Description

MCMChregress generates a sample from the posterior distribution of a Hierarchical Gaussian Linear Regression Model using Algorithm 2 of Chib and Carlin (1999). This model uses a multivariate Normal prior for the fixed effects parameters, an Inverse-Wishart prior on the random effects variance matrix, and an Inverse-Gamma prior on the residual error variance. The user supplies data and priors, and a sample from the posterior distribution is returned as an mcmc object, which can be subsequently analyzed with functions provided in the coda package.

Usage

```
MCMChregress(
  fixed,
  random,
  group,
  data,
  burnin = 1000,
  mcmc = 10000,
  thin = 10,
  verbose = 1,
  seed = NA,
  beta.start = NA,
  sigma2.start = NA,
  Vb.start = NA,
  mubeta = 0,
  Vbeta = 1e+06,
  r,
  R,
  nu = 0.001,
  delta = 0.001,
  ...
)
```

Arguments

fixed	A two-sided linear formula of the form 'y~x1+...+xp' describing the fixed-effects part of the model, with the response on the left of a '~' operator and the p fixed terms, separated by '+' operators, on the right.
random	A one-sided formula of the form '~x1+...+xq' specifying the model for the random effects part of the model, with the q random terms, separated by '+' operators.
group	String indicating the name of the grouping variable in data, defining the hierarchical structure of the model.
data	A data frame containing the variables in the model.
burnin	The number of burnin iterations for the sampler.
mcmc	The number of Gibbs iterations for the sampler. Total number of Gibbs iterations is equal to burnin+mcmc. burnin+mcmc must be divisible by 10 and superior or equal to 100 so that the progress bar can be displayed.

thin	The thinning interval used in the simulation. The number of mcmc iterations must be divisible by this value.
verbose	A switch (0,1) which determines whether or not the progress of the sampler is printed to the screen. Default is 1: a progress bar is printed, indicating the step (in %) reached by the Gibbs sampler.
seed	The seed for the random number generator. If NA, the Mersenne Twister generator is used with default seed 12345; if an integer is passed it is used to seed the Mersenne twister.
beta.start	The starting values for the β vector. This can either be a scalar or a p-length vector. The default value of NA will use the OLS β estimate of the corresponding Gaussian Linear Regression without random effects. If this is a scalar, that value will serve as the starting value mean for all of the betas.
sigma2.start	Scalar for the starting value of the residual error variance. The default value of NA will use the OLS estimates of the corresponding Gaussian Linear Regression without random effects.
Vb.start	The starting value for variance matrix of the random effects. This must be a square q-dimension matrix. Default value of NA uses an identity matrix.
mubeta	The prior mean of β. This can either be a scalar or a p-length vector. If this takes a scalar value, then that value will serve as the prior mean for all of the betas. The default value of 0 will use a vector of zeros for an uninformative prior.
Vbeta	The prior variance of β. This can either be a scalar or a square p-dimension matrix. If this takes a scalar value, then that value times an identity matrix serves as the prior variance of beta. Default value of 1.0E6 will use a diagonal matrix with very large variance for an uninformative flat prior.
r	The shape parameter for the Inverse-Wishart prior on variance matrix for the random effects. r must be superior or equal to q. Set r=q for an uninformative prior. See the NOTE for more details
R	The scale matrix for the Inverse-Wishart prior on variance matrix for the random effects. This must be a square q-dimension matrix. Use plausible variance regarding random effects for the diagonal of R. See the NOTE for more details
nu	The shape parameter for the Inverse-Gamma prior on the residual error variance. Default value is nu=delta=0.001 for uninformative prior.
delta	The rate (1/scale) parameter for the Inverse-Gamma prior on the residual error variance. Default value is nu=delta=0.001 for uninformative prior.
...	further arguments to be passed

Details

MCMChregress simulates from the posterior distribution sample using the blocked Gibbs sampler of Chib and Carlin (1999), Algorithm 2. The simulation is done in compiled C++ code to maximize efficiency. Please consult the coda documentation for a comprehensive list of functions that can be used to analyze the posterior sample.

The model takes the following form:

$$y_i = X_i\beta + W_ib_i + \varepsilon_i$$

Where each group i have k_i observations.

Where the random effects:

$$b_i \sim \mathcal{N}_q(0, V_b)$$

And the errors:

$$\varepsilon_i \sim \mathcal{N}(0, \sigma^2 I_{k_i})$$

We assume standard, conjugate priors:

$$\beta \sim \mathcal{N}_p(\mu_\beta, V_\beta)$$

And:
$$\sigma^2 \sim \mathcal{IGamma}(\nu, 1/\delta)$$

And:

$$V_b \sim \mathcal{IWishart}(r, rR)$$

See Chib and Carlin (1999) for more details.

NOTE: We do not provide default parameters for the priors on the precision matrix for the random effects. When fitting one of these models, it is of utmost importance to choose a prior that reflects your prior beliefs about the random effects. Using the dwish and rwish functions might be useful in choosing these values.

Value

mcmc	An mcmc object that contains the posterior sample. This object can be summarized by functions provided by the coda package. The posterior sample of the deviance D, with $D = -2\log(\prod_i P(y_i|\beta, b_i, \sigma^2))$, is also provided.
Y.pred	Predictive posterior mean for each observation.

Author(s)

Ghislain Vieilledent <ghislain.vieilledent@cirad.fr>

References

Siddhartha Chib and Bradley P. Carlin. 1999. "On MCMC Sampling in Hierarchical Longitudinal Models." *Statistics and Computing.* 9: 17-26.

Daniel Pemstein, Kevin M. Quinn, and Andrew D. Martin. 2007. *Scythe Statistical Library 1.0.* http://scythe.lsa.umich.edu.

Andrew D. Martin and Kyle L. Saunders. 2002. "Bayesian Inference for Political Science Panel Data." Paper presented at the 2002 Annual Meeting of the American Political Science Association.

Martyn Plummer, Nicky Best, Kate Cowles, and Karen Vines. 2006. "Output Analysis and Diagnostics for MCMC (CODA)", *R News.* 6(1): 7-11. https://CRAN.R-project.org/doc/Rnews/Rnews_2006-1.pdf.

See Also

plot.mcmc, summary.mcmc

Examples

```
## Not run:
#========================================
# Hierarchical Gaussian Linear Regression
#========================================

#== Generating data

# Constants
nobs <- 1000
nspecies <- 20
species <- c(1:nspecies,sample(c(1:nspecies),(nobs-nspecies),replace=TRUE))

# Covariates
X1 <- runif(n=nobs,min=0,max=10)
X2 <- runif(n=nobs,min=0,max=10)
X <- cbind(rep(1,nobs),X1,X2)
W <- X

# Target parameters
# beta
beta.target <- matrix(c(0.1,0.3,0.2),ncol=1)
# Vb
Vb.target <- c(0.5,0.2,0.1)
# b
b.target <- cbind(rnorm(nspecies,mean=0,sd=sqrt(Vb.target[1])),
                  rnorm(nspecies,mean=0,sd=sqrt(Vb.target[2])),
                  rnorm(nspecies,mean=0,sd=sqrt(Vb.target[3])))
# sigma2
sigma2.target <- 0.02

# Response
Y <- vector()
for (n in 1:nobs) {
  Y[n] <- rnorm(n=1,
                mean=X[n,]%*%beta.target+W[n,]%*%b.target[species[n],],
                sd=sqrt(sigma2.target))
}

# Data-set
Data <- as.data.frame(cbind(Y,X1,X2,species))
plot(Data$X1,Data$Y)

#== Call to MCMChregress
model <- MCMChregress(fixed=Y~X1+X2, random=~X1+X2, group="species",
               data=Data, burnin=1000, mcmc=1000, thin=1,verbose=1,
               seed=NA, beta.start=0, sigma2.start=1,
```

```
                Vb.start=1, mubeta=0, Vbeta=1.0E6,
                r=3, R=diag(c(1,0.1,0.1)), nu=0.001, delta=0.001)

#== MCMC analysis

# Graphics
pdf("Posteriors-MCMChregress.pdf")
plot(model$mcmc)
dev.off()

# Summary
summary(model$mcmc)

# Predictive posterior mean for each observation
model$Y.pred

# Predicted-Observed
plot(Data$Y,model$Y.pred)
abline(a=0,b=1)

## End(Not run)
```

MCMCirt1d   *Markov Chain Monte Carlo for One Dimensional Item Response The-*
                             ory Model

Description

This function generates a sample from the posterior distribution of a one dimensional item response
theory (IRT) model, with Normal priors on the subject abilities (ideal points), and multivariate
Normal priors on the item parameters. The user supplies data and priors, and a sample from the
posterior distribution is returned as an mcmc object, which can be subsequently analyzed with
functions provided in the coda package.

Usage

```
MCMCirt1d(
  datamatrix,
  theta.constraints = list(),
  burnin = 1000,
  mcmc = 20000,
  thin = 1,
  verbose = 0,
  seed = NA,
  theta.start = NA,
  alpha.start = NA,
  beta.start = NA,
  t0 = 0,
```

```
    T0 = 1,
    ab0 = 0,
    AB0 = 0.25,
    store.item = FALSE,
    store.ability = TRUE,
    drop.constant.items = TRUE,
    ...
)
```

Arguments

datamatrix The matrix of data. Must be 0, 1, or missing values. The rows of `datamatrix` correspond to subjects and the columns correspond to items.

theta.constraints

A list specifying possible simple equality or inequality constraints on the ability parameters. A typical entry in the list has one of three forms: `varname=c` which will constrain the ability parameter for the subject named `varname` to be equal to c, `varname="+"` which will constrain the ability parameter for the subject named `varname` to be positive, and `varname="-"` which will constrain the ability parameter for the subject named `varname` to be negative. If x is a matrix without row names defaults names of "V1","V2", ... , etc will be used. See Rivers (2003) for a thorough discussion of identification of IRT models.

burnin   The number of burn-in iterations for the sampler.

mcmc   The number of Gibbs iterations for the sampler.

thin   The thinning interval used in the simulation. The number of Gibbs iterations must be divisible by this value.

verbose   A switch which determines whether or not the progress of the sampler is printed to the screen. If `verbose` is greater than 0 then every `verbose`th iteration will be printed to the screen.

seed   The seed for the random number generator. If NA, the Mersenne Twister generator is used with default seed 12345; if an integer is passed it is used to seed the Mersenne twister. The user can also pass a list of length two to use the L'Ecuyer random number generator, which is suitable for parallel computation. The first element of the list is the L'Ecuyer seed, which is a vector of length six or NA (if NA a default seed of rep(12345,6) is used). The second element of list is a positive substream number. See the MCMCpack specification for more details.

theta.start The starting values for the subject abilities (ideal points). This can either be a scalar or a column vector with dimension equal to the number of voters. If this takes a scalar value, then that value will serve as the starting value for all of the thetas. The default value of NA will choose the starting values based on an eigenvalue-eigenvector decomposition of the aggreement score matrix formed from the `datamatrix`.

alpha.start The starting values for the α difficulty parameters. This can either be a scalar or a column vector with dimension equal to the number of items. If this takes a scalar value, then that value will serve as the starting value for all of the alphas. The default value of NA will set the starting values based on a series of probit regressions that condition on the starting values of theta.

beta.start   The starting values for the β discrimination parameters. This can either be a
                  scalar or a column vector with dimension equal to the number of items. If this
                  takes a scalar value, then that value will serve as the starting value for all of the
                  betas. The default value of NA will set the starting values based on a series of
                  probit regressions that condition on the starting values of theta.

t0   A scalar parameter giving the prior mean of the subject abilities (ideal points).

T0   A scalar parameter giving the prior precision (inverse variance) of the subject
                  abilities (ideal points).

ab0   The prior mean of (alpha, beta). Can be either a scalar or a 2-vector. If a
                  scalar both means will be set to the passed value. The prior mean is assumed to
                  be the same across all items.

AB0   The prior precision of (alpha, beta).This can either be ascalar or a 2 by 2
                  matrix. If this takes a scalar value, then that value times an identity matrix serves
                  as the prior precision. The prior precision is assumed to be the same across all
                  items.

store.item   A switch that determines whether or not to store the item parameters for poste-
                  rior analysis. *NOTE: In situations with many items storing the item parameters
                  takes an enormous amount of memory, so* store.item *should only be* FALSE *if
                  the chain is thinned heavily, or for applications with a small number of items.*
                  By default, the item parameters are not stored.

store.ability   A switch that determines whether or not to store the ability parameters for pos-
                  terior analysis. *NOTE: In situations with many individuals storing the ability
                  parameters takes an enormous amount of memory, so* store.ability *should
                  only be* TRUE *if the chain is thinned heavily, or for applications with a small
                  number of individuals.* By default, the item parameters are stored.

drop.constant.items
                  A switch that determines whether or not items that have no variation should be
                  deleted before fitting the model. Default = TRUE.

...   further arguments to be passed

Details

If you are interested in fitting K-dimensional item response theory models, or would rather identify
the model by placing constraints on the item parameters, please see MCMCirtKd.

MCMCirt1d simulates from the posterior distribution using standard Gibbs sampling using data aug-
mentation (a Normal draw for the subject abilities, a multivariate Normal draw for the item parame-
ters, and a truncated Normal draw for the latent utilities). The simulation proper is done in compiled
C++ code to maximize efficiency. Please consult the coda documentation for a comprehensive list
of functions that can be used to analyze the posterior sample.

The model takes the following form. We assume that each subject has an subject ability (ideal point)
denoted θ_j and that each item has a difficulty parameter α_i and discrimination parameter β_i. The
observed choice by subject j on item i is the observed data matrix which is $(I \times J)$. We assume
that the choice is dictated by an unobserved utility:

$$z_{i,j} = -\alpha_i + \beta_i \theta_j + \varepsilon_{i,j}$$

Where the errors are assumed to be distributed standard Normal. The parameters of interest are the subject abilities (ideal points) and the item parameters.

We assume the following priors. For the subject abilities (ideal points):

$$\theta_j \sim \mathcal{N}(t_0, T_0^{-1})$$

For the item parameters, the prior is:

$$[\alpha_i, \beta_i]' \sim \mathcal{N}_2(ab_0, AB_0^{-1})$$

The model is identified by the proper priors on the item parameters and constraints placed on the ability parameters.

As is the case with all measurement models, make sure that you have plenty of free memory, especially when storing the item parameters.

Value

An mcmc object that contains the sample from the posterior distribution. This object can be summarized by functions provided by the coda package.

References

James H. Albert. 1992. "Bayesian Estimation of Normal Ogive Item Response Curves Using Gibbs Sampling." *Journal of Educational Statistics*. 17: 251-269.

Joshua Clinton, Simon Jackman, and Douglas Rivers. 2004. "The Statistical Analysis of Roll Call Data." *American Political Science Review*. 98: 355-370.

Valen E. Johnson and James H. Albert. 1999. "Ordinal Data Modeling." Springer: New York.

Andrew D. Martin, Kevin M. Quinn, and Jong Hee Park. 2011. "MCMCpack: Markov Chain Monte Carlo in R.", *Journal of Statistical Software*. 42(9): 1-21. doi: 10.18637/jss.v042.i09.

Daniel Pemstein, Kevin M. Quinn, and Andrew D. Martin. 2007. *Scythe Statistical Library 1.0*. http://scythe.lsa.umich.edu.

Martyn Plummer, Nicky Best, Kate Cowles, and Karen Vines. 2006. "Output Analysis and Diagnostics for MCMC (CODA)", *R News*. 6(1): 7-11. https://CRAN.R-project.org/doc/Rnews/Rnews_2006-1.pdf.

Douglas Rivers. 2004. "Identification of Multidimensional Item-Response Models." Stanford University, typescript.

See Also

plot.mcmc, summary.mcmc, MCMCirtKd

Examples

```
    ## Not run:
    ## US Supreme Court Example with inequality constraints
    data(SupremeCourt)
    posterior1 <- MCMCirt1d(t(SupremeCourt),
                    theta.constraints=list(Scalia="+", Ginsburg="-"),
                    B0.alpha=.2, B0.beta=.2,
                    burnin=500, mcmc=100000, thin=20, verbose=500,
                    store.item=TRUE)
    geweke.diag(posterior1)
    plot(posterior1)
    summary(posterior1)

    ## US Senate Example with equality constraints
    data(Senate)
    Sen.rollcalls <- Senate[,6:677]
    posterior2 <- MCMCirt1d(Sen.rollcalls,
                    theta.constraints=list(KENNEDY=-2, HELMS=2),
                    burnin=2000, mcmc=100000, thin=20, verbose=500)
    geweke.diag(posterior2)
    plot(posterior2)
    summary(posterior2)

  ## End(Not run)
```

MCMCirtHier1d   *Markov Chain Monte Carlo for Hierarchical One Dimensional Item*
                       Response Theory Model, Covariates Predicting Latent Ideal Point
                       (Ability)

Description

This function generates a sample from the posterior distribution of a one dimensional item response theory (IRT) model, with multivariate Normal priors on the item parameters, and a Normal-Inverse Gamma hierarchical prior on subject ideal points (abilities). The user supplies item-response data, subject covariates, and priors. Note that this identification strategy obviates the constraints used on theta in MCMCirt1d. A sample from the posterior distribution is returned as an mcmc object, which can be subsequently analyzed with functions provided in the coda package.

Usage

```
MCMCirtHier1d(
  datamatrix,
  Xjdata,
  burnin = 1000,
  mcmc = 20000,
```

```
    thin = 1,
    verbose = 0,
    seed = NA,
    theta.start = NA,
    a.start = NA,
    b.start = NA,
    beta.start = NA,
    b0 = 0,
    B0 = 0.01,
    c0 = 0.001,
    d0 = 0.001,
    ab0 = 0,
    AB0 = 0.25,
    store.item = FALSE,
    store.ability = TRUE,
    drop.constant.items = TRUE,
    marginal.likelihood = c("none", "Chib95"),
    px = TRUE,
    px_a0 = 10,
    px_b0 = 10,
    ...
)
```

Arguments

datamatrix	The matrix of data. Must be 0, 1, or missing values. The rows of datamatrix correspond to subjects and the columns correspond to items.
Xjdata	A data.frame containing second-level predictor covariates for ideal points θ. Predictors are modeled as a linear regression on the mean vector of θ; the posterior sample contains regression coefficients β and common variance σ^2. See Rivers (2003) for a thorough discussion of identification of IRT models.
burnin	The number of burn-in iterations for the sampler.
mcmc	The number of Gibbs iterations for the sampler.
thin	The thinning interval used in the simulation. The number of Gibbs iterations must be divisible by this value.
verbose	A switch which determines whether or not the progress of the sampler is printed to the screen. If verbose is greater than 0 then every verboseth iteration will be printed to the screen.
seed	The seed for the random number generator. If NA, the Mersenne Twister generator is used with default seed 12345; if an integer is passed it is used to seed the Mersenne twister. The user can also pass a list of length two to use the L'Ecuyer random number generator, which is suitable for parallel computation. The first element of the list is the L'Ecuyer seed, which is a vector of length six or NA (if NA a default seed of rep(12345,6) is used). The second element of list is a positive substream number. See the MCMCpack specification for more details.
theta.start	The starting values for the subject abilities (ideal points). This can either be a scalar or a column vector with dimension equal to the number of voters. If

	this takes a scalar value, then that value will serve as the starting value for all of the thetas. The default value of NA will choose the starting values based on an eigenvalue-eigenvector decomposition of the agreement score matrix formed from the `datamatrix`.
a.start	The starting values for the a difficulty parameters. This can either be a scalar or a column vector with dimension equal to the number of items. If this takes a scalar value, then that value will serve as the starting value for all a. The default value of NA will set the starting values based on a series of probit regressions that condition on the starting values of theta.
b.start	The starting values for the b discrimination parameters. This can either be a scalar or a column vector with dimension equal to the number of items. If this takes a scalar value, then that value will serve as the starting value for all b. The default value of NA will set the starting values based on a series of probit regressions that condition on the starting values of theta.
beta.start	The starting values for the β regression coefficients that predict the means of ideal points θ. This can either be a scalar or a column vector with length equal to the number of covariates. If this takes a scalar value, then that value will serve as the starting value for all of the betas. The default value of NA will set the starting values based on a linear regression of the covariates on (either provided or generated) `theta.start`.
b0	The prior mean of β. Can be either a scalar or a vector of length equal to the number of subject covariates. If a scalar all means with be set to the passed value.
B0	The prior precision of β. This can either be a scalar or a square matrix with dimensions equal to the number of betas. If this takes a scalar value, then that value times an identity matrix serves as the prior precision of beta. A default proper but diffuse value of .01 ensures finite marginal likelihood for model comparison. A value of 0 is equivalent to an improper uniform prior for beta.
c0	$c_0/2$ is the shape parameter for the inverse Gamma prior on σ^2 (the variance of θ). The amount of information in the inverse Gamma prior is something like that from c_0 pseudo-observations.
d0	$d_0/2$ is the scale parameter for the inverse Gamma prior on σ^2 (the variance of θ). In constructing the inverse Gamma prior, d_0 acts like the sum of squared errors from the c_0 pseudo-observations.
ab0	The prior mean of (a, b). Can be either a scalar or a 2-vector. If a scalar both means will be set to the passed value. The prior mean is assumed to be the same across all items.
AB0	The prior precision of (a, b).This can either be ascalar or a 2 by 2 matrix. If this takes a scalar value, then that value times an identity matrix serves as the prior precision. The prior precision is assumed to be the same across all items.
store.item	A switch that determines whether or not to store the item parameters for posterior analysis. *NOTE: In situations with many items storing the item parameters takes an enormous amount of memory, so* `store.item` *should only be* TRUE *if the chain is thinned heavily, or for applications with a small number of items.* By default, the item parameters are not stored.

store.ability   A switch that determines whether or not to store the ability parameters for posterior analysis. *NOTE: In situations with many individuals storing the ability parameters takes an enormous amount of memory, so* store.ability *should only be* TRUE *if the chain is thinned heavily, or for applications with a small number of individuals*. By default, ability parameters are stored.

drop.constant.items
                   A switch that determines whether or not items that have no variation should be deleted before fitting the model. Default = TRUE.

marginal.likelihood
                   Should the marginal likelihood of the second-level model on ideal points be calculated using the method of Chib (1995)? It is stored as an attribute of the posterior mcmc object and suitable for comparison using BayesFactor.

px   Use Parameter Expansion to reduce autocorrelation in the chain? PX introduces an unidentified parameter $alpha$ for the residual variance in the latent data (Liu and Wu 1999). Default = TRUE

px_a0   Prior shape parameter for the inverse-gamma distribution on $alpha$, the residual variance of the latent data. Default=10.

px_b0   Prior scale parameter for the inverse-gamma distribution on $alpha$, the residual variance of the latent data. Default = 10

...   further arguments to be passed

Details

If you are interested in fitting K-dimensional item response theory models, or would rather identify the model by placing constraints on the item parameters, please see MCMCirtKd.

MCMCirtHier1d simulates from the posterior distribution using standard Gibbs sampling using data augmentation (a Normal draw for the subject abilities, a multivariate Normal draw for (second-level) subject ability predictors, an Inverse-Gamma draw for the (second-level) variance of subject abilities, a multivariate Normal draw for the item parameters, and a truncated Normal draw for the latent utilities). The simulation proper is done in compiled C++ code to maximize efficiency. Please consult the coda documentation for a comprehensive list of functions that can be used to analyze the posterior sample.

The model takes the following form. We assume that each subject has an subject ability (ideal point) denoted θ_j and that each item has a difficulty parameter a_i and discrimination parameter b_i. The observed choice by subject j on item i is the observed data matrix which is $(I \times J)$. We assume that the choice is dictated by an unobserved utility:

$$z_{i,j} = -\alpha_i + \beta_i \theta_j + \varepsilon_{i,j}$$

Where the errors are assumed to be distributed standard Normal. This constitutes the measurement or level-1 model. The subject abilities (ideal points) are modeled by a second level Normal linear predictor for subject covariates Xjdata, with common variance σ^2. The parameters of interest are the subject abilities (ideal points), item parameters, and second-level coefficients.

We assume the following priors. For the subject abilities (ideal points):

$$\theta_j \sim \mathcal{N}(\mu_\theta, T_0^{-1})$$

For the item parameters, the prior is:

$$[a_i, b_i]' \sim \mathcal{N}_2(ab_0, AB_0^{-1})$$

The model is identified by the proper priors on the item parameters and constraints placed on the ability parameters.

As is the case with all measurement models, make sure that you have plenty of free memory, especially when storing the item parameters.

Value

An `mcmc` object that contains the sample from the posterior distribution. This object can be summarized by functions provided by the coda package. If `marginal.likelihood = "Chib95"` the object will have attribute `logmarglike`.

Author(s)

Michael Malecki, <mike@crunch.io>, https://github.com/malecki.

References

James H. Albert. 1992. "Bayesian Estimation of Normal Ogive Item Response Curves Using Gibbs Sampling." *Journal of Educational Statistics*. 17: 251–269.

Joshua Clinton, Simon Jackman, and Douglas Rivers. 2004. "The Statistical Analysis of Roll Call Data." *American Political Science Review* 98: 355–370.

Valen E. Johnson and James H. Albert. 1999. "Ordinal Data Modeling." Springer: New York.

Liu, Jun S. and Ying Nian Wu. 1999. "Parameter Expansion for Data Augmentation." *Journal of the American Statistical Association* 94: 1264–1274.

Andrew D. Martin, Kevin M. Quinn, and Jong Hee Park. 2011. "MCMCpack: Markov Chain Monte Carlo in R.", *Journal of Statistical Software*. 42(9): 1-21. doi: 10.18637/jss.v042.i09.

Daniel Pemstein, Kevin M. Quinn, and Andrew D. Martin. 2007. *Scythe Statistical Library 1.0.* http://scythe.lsa.umich.edu.

Martyn Plummer, Nicky Best, Kate Cowles, and Karen Vines. 2006. "Output Analysis and Diagnostics for MCMC (CODA)", *R News*. 6(1): 7-11. https://CRAN.R-project.org/doc/Rnews/Rnews_2006-1.pdf.

Douglas Rivers. 2004. "Identification of Multidimensional Item-Response Models." Stanford University, typescript.

See Also

plot.mcmc,summary.mcmc, MCMCirtKd

Examples

```
   ## Not run:
data(SupremeCourt)

Xjdata <- data.frame(presparty= c(1,1,0,1,1,1,1,0,0),
                     sex= c(0,0,1,0,0,0,0,1,0))

## Parameter Expansion reduces autocorrelation.
  posterior1 <- MCMCirtHier1d(t(SupremeCourt),
                   burnin=50000, mcmc=10000, thin=20,
                   verbose=10000,
                   Xjdata=Xjdata,
                   marginal.likelihood="Chib95",
  px=TRUE)

## But, you can always turn it off.
  posterior2 <- MCMCirtHier1d(t(SupremeCourt),
                   burnin=50000, mcmc=10000, thin=20,
                   verbose=10000,
                   Xjdata=Xjdata,
                   #marginal.likelihood="Chib95",
  px=FALSE)
## Note that the hierarchical model has greater autocorrelation than
## the naive IRT model.
  posterior0 <- MCMCirt1d(t(SupremeCourt),
                       theta.constraints=list(Scalia="+", Ginsburg="-"),
                       B0.alpha=.2, B0.beta=.2,
                       burnin=50000, mcmc=100000, thin=100, verbose=10000,
                       store.item=FALSE)

## Randomly 10% Missing -- this affects the expansion parameter, increasing
## the variance of the (unidentified) latent parameter alpha.

  scMiss <- SupremeCourt
  scMiss[matrix(as.logical(rbinom(nrow(SupremeCourt)*ncol(SupremeCourt), 1, .1)),
     dim(SupremeCourt))] <- NA

  posterior1.miss <- MCMCirtHier1d(t(scMiss),
                   burnin=80000, mcmc=10000, thin=20,
                   verbose=10000,
                   Xjdata=Xjdata,
                   marginal.likelihood="Chib95",
  px=TRUE)


## End(Not run)
```

MCMCirtKd   *Markov Chain Monte Carlo for K-Dimensional Item Response Theory*
                                    Model

Description

This function generates a sample from the posterior distribution of a K-dimensional item response
theory (IRT) model, with standard normal priors on the subject abilities (ideal points), and normal
priors on the item parameters. The user supplies data and priors, and a sample from the posterior
distribution is returned as an mcmc object, which can be subsequently analyzed with functions
provided in the coda package.

Usage

```
MCMCirtKd(
  datamatrix,
  dimensions,
  item.constraints = list(),
  burnin = 1000,
  mcmc = 10000,
  thin = 1,
  verbose = 0,
  seed = NA,
  alphabeta.start = NA,
  b0 = 0,
  B0 = 0,
  store.item = FALSE,
  store.ability = TRUE,
  drop.constant.items = TRUE,
  ...
)
```

Arguments

datamatrix   The matrix of data. Must be 0, 1, or missing values. It is of dimensionality
                    subjects by items.

dimensions   The number of dimensions in the latent space.

item.constraints

                    List of lists specifying possible equality or simple inequality constraints on the
                    item parameters. A typical entry in the list has one of three forms: `rowname=list(d,c)`
                    which will constrain the dth item parameter for the item named rowname to be
                    equal to c, `rowname=list(d,"+")` which will constrain the dth item parameter
                    for the item named rowname to be positive, and`rowname=list(d, "-")` which
                    will constrain the dth item parameter for the item named rowname to be neg-
                    ative. If x is a matrix without row names defaults names of "V1", "V2", ... ,
                    etc will be used. In a K dimensional model, the first item parameter for item
                    i is the difficulty parameter (α_i), the second item parameter is the discrimation
                    parameter on dimension 1 ($\beta_{i,1}$), the third item parameter is the discrimation

parameter on dimension 2 ($\beta_{i,2}$), ..., and the (K+1)th item parameter is the discrimation parameter on dimension K ($\beta_{i,1}$). The item difficulty parameters (α) should generally not be constrained.

burnin   The number of burn-in iterations for the sampler.

mcmc   The number of iterations for the sampler.

thin   The thinning interval used in the simulation. The number of iterations must be divisible by this value.

verbose   A switch which determines whether or not the progress of the sampler is printed to the screen. If verbose is greater than 0 then every verboseth iteration will be printed to the screen.

seed   The seed for the random number generator. If NA, the Mersenne Twister generator is used with default seed 12345; if an integer is passed it is used to seed the Mersenne twister. The user can also pass a list of length two to use the L'Ecuyer random number generator, which is suitable for parallel computation. The first element of the list is the L'Ecuyer seed, which is a vector of length six or NA (if NA a default seed of rep(12345,6) is used). The second element of list is a positive substream number. See the MCMCpack specification for more details.

alphabeta.start
                The starting values for the α and β difficulty and discrimination parameters. If alphabeta.start is set to a scalar the starting value for all unconstrained item parameters will be set to that scalar. If alphabeta.start is a matrix of dimension $(K + 1) \times items$ then the alphabeta.start matrix is used as the starting values (except for equality-constrained elements). If alphabeta.start is set to NA (the default) then starting values for unconstrained elements are set to values generated from a series of proportional odds logistic regression fits, and starting values for inequality constrained elements are set to either 1.0 or -1.0 depending on the nature of the constraints.

b0   The prior means of the α and β difficulty and discrimination parameters, stacked for all items. If a scalar is passed, it is used as the prior mean for all items.

B0   The prior precisions (inverse variances) of the independent normal prior on the item parameters. Can be either a scalar or a matrix of dimension $(K + 1) \times items$.

store.item   A switch that determines whether or not to store the item parameters for posterior analysis. *NOTE: In applications with many items this takes an enormous amount of memory. If you have many items and want to want to store the item parameters you may want to thin the chain heavily.* By default, the item parameters are not stored.

store.ability   A switch that determines whether or not to store the subject abilities for posterior analysis. *NOTE: In applications with many subjects this takes an enormous amount of memory. If you have many subjects and want to want to store the ability parameters you may want to thin the chain heavily.* By default, the ability parameters are all stored.

drop.constant.items
                A switch that determines whether or not items that have no variation should be deleted before fitting the model. Default = TRUE.

...   further arguments to be passed

Details

MCMCirtKd simulates from the posterior distribution using standard Gibbs sampling using data augmentation (a normal draw for the subject abilities, a multivariate normal draw for the item parameters, and a truncated normal draw for the latent utilities). The simulation proper is done in compiled C++ code to maximize efficiency. Please consult the coda documentation for a comprehensive list of functions that can be used to analyze the posterior sample.

The default number of burnin and mcmc iterations is much smaller than the typical default values in MCMCpack. This is because fitting this model is extremely computationally expensive. It does not mean that this small of a number of scans will yield good estimates. The priors of this model need to be proper for identification purposes. The user is asked to provide prior means and precisions *(not variances)* for the item parameters and the subject parameters.

The model takes the following form. We assume that each subject has an ability (ideal point) denoted θ_j $(K \times 1)$, and that each item has a difficulty parameter α_i and discrimination parameter β_i $(K \times 1)$. The observed choice by subject j on item i is the observed data matrix which is $(I \times J)$. We assume that the choice is dictated by an unobserved utility:

$$z_{i,j} = -\alpha_i + \beta_i'\theta_j + \varepsilon_{i,j}$$

Where the $\varepsilon_{i,j}$s are assumed to be distributed standard normal. The parameters of interest are the subject abilities (ideal points) and the item parameters.

We assume the following priors. For the subject abilities (ideal points) we assume independent standard normal priors:

$$\theta_{j,k} \sim \mathcal{N}(0,1)$$

These cannot be changed by the user. For each item parameter, we assume independent normal priors:

$$[\alpha_i, \beta_i]' \sim \mathcal{N}_{(K+1)}(b_{0,i}, B_{0,i})$$

Where $B_{0,i}$ is a diagonal matrix. One can specify a separate prior mean and precision for each item parameter.

The model is identified by the constraints on the item parameters (see Jackman 2001). The user cannot place constraints on the subject abilities. This identification scheme differs from that in MCMCirt1d, which uses constraints on the subject abilities to identify the model. In our experience, using subject ability constraints for models in greater than one dimension does not work particularly well.

As is the case with all measurement models, make sure that you have plenty of free memory, especially when storing the item parameters.

Value

An mcmc object that contains the posterior sample. This object can be summarized by functions provided by the coda package.

References

James H. Albert. 1992. "Bayesian Estimation of Normal Ogive Item Response Curves Using Gibbs Sampling." *Journal of Educational Statistics*. 17: 251-269.

Joshua Clinton, Simon Jackman, and Douglas Rivers. 2004. "The Statistical Analysis of Roll Call Data." *American Political Science Review*. 98: 355-370.

Simon Jackman. 2001. "Multidimensional Analysis of Roll Call Data via Bayesian Simulation." *Political Analysis*. 9: 227-241.

Valen E. Johnson and James H. Albert. 1999. *Ordinal Data Modeling*. Springer: New York.

Andrew D. Martin, Kevin M. Quinn, and Jong Hee Park. 2011. "MCMCpack: Markov Chain Monte Carlo in R.", *Journal of Statistical Software*. 42(9): 1-21. doi: 10.18637/jss.v042.i09.

Daniel Pemstein, Kevin M. Quinn, and Andrew D. Martin. 2007. *Scythe Statistical Library 1.0*. http://scythe.lsa.umich.edu.

Martyn Plummer, Nicky Best, Kate Cowles, and Karen Vines. 2006. "Output Analysis and Diagnostics for MCMC (CODA)", *R News*. 6(1): 7-11. https://CRAN.R-project.org/doc/Rnews/Rnews_2006-1.pdf.

Douglas Rivers. 2004. "Identification of Multidimensional Item-Response Models." Stanford University, typescript.

See Also

plot.mcmc,summary.mcmc, MCMCirt1d, MCMCordfactanal

Examples

```
## Not run:
data(SupremeCourt)
# note that the rownames (the item names) are "1", "2", etc
posterior1 <- MCMCirtKd(t(SupremeCourt), dimensions=1,
              burnin=5000, mcmc=50000, thin=10,
              B0=.25, store.item=TRUE,
              item.constraints=list("1"=list(2,"-")))
plot(posterior1)
summary(posterior1)


data(Senate)
Sen.rollcalls <- Senate[,6:677]
posterior2 <- MCMCirtKd(Sen.rollcalls, dimensions=2,
              burnin=5000, mcmc=50000, thin=10,
              item.constraints=list(rc2=list(2,"-"), rc2=c(3,0),
                                    rc3=list(3,"-")),
              B0=.25)
plot(posterior2)
summary(posterior2)

## End(Not run)
```

MCMCirtKdRob   *Markov Chain Monte Carlo for Robust K-Dimensional Item Response Theory Model*

Description

This function generates a posterior sample from a Robust K-dimensional item response theory (IRT) model with logistic link, independent standard normal priors on the subject abilities (ideal points), and independent normal priors on the item parameters. The user supplies data and priors, and a sample from the posterior distribution is returned as an mcmc object, which can be subsequently analyzed with functions provided in the coda package.

Usage

```
MCMCirtKdRob(
  datamatrix,
  dimensions,
  item.constraints = list(),
  ability.constraints = list(),
  burnin = 500,
  mcmc = 5000,
  thin = 1,
  interval.method = "step",
  theta.w = 0.5,
  theta.mp = 4,
  alphabeta.w = 1,
  alphabeta.mp = 4,
  delta0.w = NA,
  delta0.mp = 3,
  delta1.w = NA,
  delta1.mp = 3,
  verbose = FALSE,
  seed = NA,
  theta.start = NA,
  alphabeta.start = NA,
  delta0.start = NA,
  delta1.start = NA,
  b0 = 0,
  B0 = 0,
  k0 = 0.1,
  k1 = 0.1,
  c0 = 1,
  d0 = 1,
  c1 = 1,
  d1 = 1,
  store.item = TRUE,
  store.ability = FALSE,
```

```
    drop.constant.items = TRUE,
    ...
)
```

Arguments

datamatrix   The matrix of data. Must be 0, 1, or missing values. It is of dimensionality
                subjects by items.

dimensions   The number of dimensions in the latent space.

item.constraints

                List of lists specifying possible equality or simple inequality constraints on the
                item parameters. A typical entry in the list has one of three forms: rowname=list(d,c)
                which will constrain the dth item parameter for the item named rowname to be
                equal to c, rowname=list(d,"+") which will constrain the dth item parameter
                for the item named rowname to be positive, and rowname=list(d, "-") which
                will constrain the dth item parameter for the item named rowname to be nega-
                tive. If datamatrix is a matrix without row names defaults names of "V1", "V2",
                ... , etc will be used. In a K-dimensional model, the first item parameter for item
                i is the difficulty parameter (α_i), the second item parameter is the discrimation
                parameter on dimension 1 ($\beta_{i,1}$), the third item parameter is the discrimation
                parameter on dimension 2 ($\beta_{i,2}$), ..., and the $(K + 1)$th item parameter is the
                discrimation parameter on dimension K ($\beta_{i,K}$). The item difficulty parameters
                (α) should generally not be constrained.

ability.constraints

                List of lists specifying possible equality or simple inequality constraints on the
                ability parameters. A typical entry in the list has one of three forms: colname=list(d,c)
                which will constrain the dth ability parameter for the subject named colname to
                be equal to c, colname=list(d,"+") which will constrain the dth ability pa-
                rameter for the subject named colname to be positive, and colname=list(d,
                "-") which will constrain the dth ability parameter for the subject named col-
                name to be negative. If datamatrix is a matrix without column names defaults
                names of "V1", "V2", ... , etc will be used.

burnin   The number of burn-in iterations for the sampler.

mcmc   The number of iterations for the sampler after burn-in.

thin   The thinning interval used in the simulation. The number of iterations must be
                divisible by this value.

interval.method

                Method for finding the slicing interval. Can be equal to either step in which case
                the stepping out algorithm of Neal (2003) is used or doubling in which case
                the doubling procedure of Neal (2003) is used. The stepping out method tends
                to be faster on a per-iteration basis as it typically requires few function calls.
                The doubling method expands the initial interval more quickly which makes the
                Markov chain mix somewhat more quickly– at least in some situations.

theta.w   The initial width of the slice sampling interval for each ability parameter (the
                elements of θ)

theta.mp   The parameter governing the maximum possible width of the slice interval for
                 each ability parameter (the elements of θ). If `interval.method="step"` then
                 the maximum width is `theta.w * theta.mp`.

                 If `interval.method="doubling"` then the maximum width is `theta.w * 2^theta.mp`.

alphabeta.w   The initial width of the slice sampling interval for each item parameter (the
                 elements of α and β)

alphabeta.mp   The parameter governing the maximum possible width of the slice interval for
                 each item parameters (the elements of α and β). If `interval.method="step"`
                 then the maximum width is `alphabeta.w * alphabeta.mp`.

                 If `interval.method="doubling"` then the maximum width is `alphabeta.w * 2^alphabeta.mp`.

delta0.w   The initial width of the slice sampling interval for δ_0

delta0.mp   The parameter governing the maximum possible width of the slice interval for
                 δ_0. If `interval.method="step"` then the maximum width is `delta0.w * delta0.mp`.
                 If `interval.method="doubling"` then the maximum width is `delta0.w * 2^delta0.mp`.

delta1.w   The initial width of the slice sampling interval for δ_1

delta1.mp   The parameter governing the maximum possible width of the slice interval for
                 δ_1. If `interval.method="step"` then the maximum width is `delta1.w * delta1.mp`.
                 If `interval.method="doubling"` then the maximum width is `delta1.w * 2^delta1.mp`.

verbose   A switch which determines whether or not the progress of the sampler is printed
                 to the screen. If verbose > 0, the iteration number with be printed to the screen
                 every verbose'th iteration.

seed   The seed for the random number generator. If NA, the Mersenne Twister gener-
                 ator is used with default seed 12345; if an integer is passed it is used to seed the
                 Mersenne twister. The user can also pass a list of length two to use the L'Ecuyer
                 random number generator, which is suitable for parallel computation. The first
                 element of the list is the L'Ecuyer seed, which is a vector of length six or NA
                 (if NA a default seed of `rep(12345,6)` is used). The second element of list is a
                 positive substream number. See the MCMCpack specification for more details.

theta.start   The starting values for the ability parameters θ. Can be either a scalar or a matrix
                 with number of rows equal to the number of subjects and number of columns
                 equal to the dimension K of the latent space. If `theta.start=NA` then starting
                 values will be chosen that are 0 for unconstrained subjects, -0.5 for subjects
                 with negative inequality constraints and 0.5 for subjects with positive inequality
                 constraints.

alphabeta.start

                 The starting values for the α and β difficulty and discrimination parameters.
                 If `alphabeta.start` is set to a scalar the starting value for all unconstrained
                 item parameters will be set to that scalar. If `alphabeta.start` is a matrix of
                 dimension $(K + 1) \times items$ then the `alphabeta.start` matrix is used as the
                 starting values (except for equality-constrained elements). If `alphabeta.start`
                 is set to NA (the default) then starting values for unconstrained elements are set
                 to values generated from a series of proportional odds logistic regression fits,
                 and starting values for inequality constrained elements are set to either 1.0 or
                 -1.0 depending on the nature of the constraints.

delta0.start	The starting value for the δ_0 parameter.
delta1.start	The starting value for the δ_1 parameter.
b0	The prior means of the α and β difficulty and discrimination parameters, stacked for all items. If a scalar is passed, it is used as the prior mean for all items.
B0	The prior precisions (inverse variances) of the independent Normal prior on the item parameters. Can be either a scalar or a matrix of dimension $(K + 1) \times items$.
k0	δ_0 is constrained to lie in the interval between 0 and k0.
k1	δ_1 is constrained to lie in the interval between 0 and k1.
c0	Parameter governing the prior for δ_0. δ_0 divided by k0 is assumed to be follow a beta distribution with first parameter c0.
d0	Parameter governing the prior for δ_0. δ_0 divided by k0 is assumed to be follow a beta distribution with second parameter d0.
c1	Parameter governing the prior for δ_1. δ_1 divided by k1 is assumed to be follow a beta distribution with first parameter c1.
d1	Parameter governing the prior for δ_1. δ_1 divided by k1 is assumed to be follow a beta distribution with second parameter d1.
store.item	A switch that determines whether or not to store the item parameters for posterior analysis. *NOTE: This typically takes an enormous amount of memory, so should only be used if the chain is thinned heavily, or for applications with a small number of items.* By default, the item parameters are not stored.
store.ability	A switch that determines whether or not to store the subject abilities for posterior analysis. By default, the item parameters are all stored.
drop.constant.items	
	A switch that determines whether or not items that have no variation should be deleted before fitting the model. Default = TRUE.
...	further arguments to be passed

Details

MCMCirtKdRob simulates from the posterior using the slice sampling algorithm of Neal (2003). The simulation proper is done in compiled C++ code to maximize efficiency. Please consult the coda documentation for a comprehensive list of functions that can be used to analyze the posterior sample.

The model takes the following form. We assume that each subject has an subject ability (ideal point) denoted θ_j $(K \times 1)$, and that each item has a scalar difficulty parameter α_i and discrimination parameter β_i $(K \times 1)$. The observed choice by subject j on item i is the observed data matrix which is $(I \times J)$.

The probability that subject j answers item i correctly is assumed to be:

$$\pi_{ij} = \delta_0 + (1 - \delta_0 - \delta_1)/(1 + \exp(\alpha_i - \beta_i\theta_j))$$

This model was discussed in Bafumi et al. (2005).

We assume the following priors. For the subject abilities (ideal points) we assume independent standard Normal priors:

$$\theta_{j,k} \sim \mathcal{N}(0, 1)$$

These cannot be changed by the user. For each item parameter, we assume independent Normal priors:

$$[\alpha_i, \beta_i]' \sim \mathcal{N}_{(K+1)}(b_{0,i}, B_{0,i})$$

Where $B_{0,i}$ is a diagonal matrix. One can specify a separate prior mean and precision for each item parameter. We also assume $\delta_0/k_0 \sim \mathcal{B}eta(c_0, d_0)$ and $\delta_1/k_1 \sim \mathcal{B}eta(c_1, d_1)$.

The model is identified by constraints on the item parameters and / or ability parameters. See Rivers (2004) for a discussion of identification of IRT models.

As is the case with all measurement models, make sure that you have plenty of free memory, especially when storing the item parameters.

Value

An mcmc object that contains the posterior sample. This object can be summarized by functions provided by the coda package.

References

James H. Albert. 1992. "Bayesian Estimation of Normal Ogive Item Response Curves Using Gibbs Sampling." *Journal of Educational Statistics*. 17: 251-269.

Joseph Bafumi, Andrew Gelman, David K. Park, and Noah Kaplan. 2005. "Practical Issues in Implementing and Understanding Bayesian Ideal Point Estimation." *Political Analysis*.

Joshua Clinton, Simon Jackman, and Douglas Rivers. 2004. "The Statistical Analysis of Roll Call Data." *American Political Science Review*. 98: 355-370.

Simon Jackman. 2001. "Multidimensional Analysis of Roll Call Data via Bayesian Simulation." *Political Analysis*. 9: 227-241.

Valen E. Johnson and James H. Albert. 1999. *Ordinal Data Modeling*. Springer: New York.

Andrew D. Martin, Kevin M. Quinn, and Jong Hee Park. 2011. "MCMCpack: Markov Chain Monte Carlo in R.", *Journal of Statistical Software*. 42(9): 1-21. doi: 10.18637/jss.v042.i09.

Daniel Pemstein, Kevin M. Quinn, and Andrew D. Martin. 2007. *Scythe Statistical Library 1.0*. http://scythe.lsa.umich.edu.

Radford Neal. 2003. "Slice Sampling" (with discussion). *Annals of Statistics*, 31: 705-767.

Martyn Plummer, Nicky Best, Kate Cowles, and Karen Vines. 2006. "Output Analysis and Diagnostics for MCMC (CODA)", *R News*. 6(1): 7-11. https://CRAN.R-project.org/doc/Rnews/Rnews_2006-1.pdf.

Douglas Rivers. 2004. "Identification of Multidimensional Item-Response Models." Stanford University, typescript.

See Also

plot.mcmc,summary.mcmc, MCMCirt1d, MCMCirtKd

Examples

```
## Not run:
## Court example with ability (ideal point) and
##  item (case) constraints
data(SupremeCourt)
post1 <- MCMCirtKdRob(t(SupremeCourt), dimensions=1,
                burnin=500, mcmc=5000, thin=1,
                B0=.25, store.item=TRUE, store.ability=TRUE,
                ability.constraints=list("Thomas"=list(1,"+"),
                "Stevens"=list(1,-4)),
                item.constraints=list("1"=list(2,"-")),
                verbose=50)
plot(post1)
summary(post1)

## Senate example with ability (ideal point) constraints
data(Senate)
namestring <- as.character(Senate$member)
namestring[78] <- "CHAFEE1"
namestring[79] <- "CHAFEE2"
namestring[59] <- "SMITH.NH"
namestring[74] <- "SMITH.OR"
rownames(Senate) <- namestring
post2 <- MCMCirtKdRob(Senate[,6:677], dimensions=1,
                        burnin=1000, mcmc=5000, thin=1,
                        ability.constraints=list("KENNEDY"=list(1,-4),
                                "HELMS"=list(1, 4), "ASHCROFT"=list(1,"+"),
                                "BOXER"=list(1,"-"), "KERRY"=list(1,"-"),
                                "HATCH"=list(1,"+")),
                        B0=0.1, store.ability=TRUE, store.item=FALSE,
                        verbose=5, k0=0.15, k1=0.15,
                        delta0.start=0.13, delta1.start=0.13)

plot(post2)
summary(post2)

## End(Not run)
```

MCMClogit   *Markov Chain Monte Carlo for Logistic Regression*

Description

This function generates a sample from the posterior distribution of a logistic regression model using
a random walk Metropolis algorithm. The user supplies data and priors, and a sample from the
posterior distribution is returned as an mcmc object, which can be subsequently analyzed with
functions provided in the coda package.

Usage

```
MCMClogit(
  formula,
  data = NULL,
  burnin = 1000,
  mcmc = 10000,
  thin = 1,
  tune = 1.1,
  verbose = 0,
  seed = NA,
  beta.start = NA,
  b0 = 0,
  B0 = 0,
  user.prior.density = NULL,
  logfun = TRUE,
  marginal.likelihood = c("none", "Laplace"),
  ...
)
```

Arguments

formula   Model formula.

data   Data frame.

burnin   The number of burn-in iterations for the sampler.

mcmc   The number of Metropolis iterations for the sampler.

thin   The thinning interval used in the simulation. The number of mcmc iterations
                must be divisible by this value.

tune   Metropolis tuning parameter. Can be either a positive scalar or a k-vector, where
                k is the length of β.Make sure that the acceptance rate is satisfactory (typically
                between 0.20 and 0.5) before using the posterior sample for inference.

verbose   A switch which determines whether or not the progress of the sampler is printed
                to the screen. If verbose is greater than 0 the iteration number, the current
                beta vector, and the Metropolis acceptance rate are printed to the screen every
                verboseth iteration.

seed   The seed for the random number generator. If NA, the Mersenne Twister gener-
                ator is used with default seed 12345; if an integer is passed it is used to seed the
                Mersenne twister. The user can also pass a list of length two to use the L'Ecuyer
                random number generator, which is suitable for parallel computation. The first
                element of the list is the L'Ecuyer seed, which is a vector of length six or NA

(if NA a default seed of rep(12345,6) is used). The second element of list is a positive substream number. See the MCMCpack specification for more details.

beta.start   The starting value for the β vector. This can either be a scalar or a column vector with dimension equal to the number of betas. If this takes a scalar value, then that value will serve as the starting value for all of the betas. The default value of NA will use the maximum likelihood estimate of β as the starting value.

b0   If user.prior.density==NULL b0 is the prior mean of β under a multivariate normal prior. This can either be a scalar or a column vector with dimension equal to the number of betas. If this takes a scalar value, then that value will serve as the prior mean for all of the betas.

B0   If user.prior.density==NULL B0 is the prior precision of β under a multivariate normal prior. This can either be a scalar or a square matrix with dimensions equal to the number of betas. If this takes a scalar value, then that value times an identity matrix serves as the prior precision of β. Default value of 0 is equivalent to an improper uniform prior for beta.

user.prior.density
    If non-NULL, the prior (log)density up to a constant of proportionality. This must be a function defined in R whose first argument is a continuous (possibly vector) variable. This first argument is the point in the state space at which the prior (log)density is to be evaluated. Additional arguments can be passed to user.prior.density() by inserting them in the call to MCMClogit(). See the Details section and the examples below for more information.

logfun   Logical indicating whether use.prior.density() returns the natural log of a density function (TRUE) or a density (FALSE).

marginal.likelihood
    How should the marginal likelihood be calculated? Options are: none in which case the marginal likelihood will not be calculated or Laplace in which case the Laplace approximation (see Kass and Raftery, 1995) is used.

...   further arguments to be passed

Details

MCMClogit simulates from the posterior distribution of a logistic regression model using a random walk Metropolis algorithm. The simulation proper is done in compiled C++ code to maximize efficiency. Please consult the coda documentation for a comprehensive list of functions that can be used to analyze the posterior sample.

The model takes the following form:

$$y_i \sim \mathcal{B}ernoulli(\pi_i)$$

Where the inverse link function:

$$\pi_i = \frac{\exp(x_i'\beta)}{1 + \exp(x_i'\beta)}$$

By default, we assume a multivariate Normal prior on β:

$$\beta \sim \mathcal{N}(b_0, B_0^{-1})$$

Additionally, arbitrary user-defined priors can be specified with the user.prior.density argument.

If the default multivariate normal prior is used, the Metropolis proposal distribution is centered at the current value of β and has variance-covariance $V = T(B_0 + C^{-1})^{-1}T$, where T is a the diagonal positive definite matrix formed from the tune, B_0 is the prior precision, and C is the large sample variance-covariance matrix of the MLEs. This last calculation is done via an initial call to glm.

If a user-defined prior is used, the Metropolis proposal distribution is centered at the current value of β and has variance-covariance $V = TCT$, where T is a the diagonal positive definite matrix formed from the tune and C is the large sample variance-covariance matrix of the MLEs. This last calculation is done via an initial call to glm.

Value

An mcmc object that contains the posterior sample. This object can be summarized by functions provided by the coda package.

References

Andrew D. Martin, Kevin M. Quinn, and Jong Hee Park. 2011. "MCMCpack: Markov Chain Monte Carlo in R.", *Journal of Statistical Software*. 42(9): 1-21. doi: 10.18637/jss.v042.i09.

Daniel Pemstein, Kevin M. Quinn, and Andrew D. Martin. 2007. *Scythe Statistical Library 1.0.* http://scythe.lsa.umich.edu.

Martyn Plummer, Nicky Best, Kate Cowles, and Karen Vines. 2006. "Output Analysis and Diagnostics for MCMC (CODA)", *R News*. 6(1): 7-11. https://CRAN.R-project.org/doc/Rnews/Rnews_2006-1.pdf.

See Also

plot.mcmc,summary.mcmc, glm

Examples

```
## Not run:
## default improper uniform prior
data(birthwt)
posterior <- MCMClogit(low~age+as.factor(race)+smoke, data=birthwt)
plot(posterior)
summary(posterior)


## multivariate normal prior
data(birthwt)
posterior <- MCMClogit(low~age+as.factor(race)+smoke, b0=0, B0=.001,
```

```
                               data=birthwt)
    plot(posterior)
    summary(posterior)


    ## user-defined independent Cauchy prior
    logpriorfun <- function(beta){
      sum(dcauchy(beta, log=TRUE))
    }

    posterior <- MCMClogit(low~age+as.factor(race)+smoke,
                           data=birthwt, user.prior.density=logpriorfun,
                           logfun=TRUE)
    plot(posterior)
    summary(posterior)


    ## user-defined independent Cauchy prior with additional args
    logpriorfun <- function(beta, location, scale){
      sum(dcauchy(beta, location, scale, log=TRUE))
    }

    posterior <- MCMClogit(low~age+as.factor(race)+smoke,
                           data=birthwt, user.prior.density=logpriorfun,
                           logfun=TRUE, location=0, scale=10)
    plot(posterior)
    summary(posterior)



  ## End(Not run)
```

MCMCmetrop1R   *Metropolis Sampling from User-Written R function*

Description

This function allows a user to construct a sample from a user-defined continuous distribution using a random walk Metropolis algorithm.

Usage

```
MCMCmetrop1R(
  fun,
  theta.init,
  burnin = 500,
  mcmc = 20000,
  thin = 1,
  tune = 1,
```

```
        verbose = 0,
        seed = NA,
        logfun = TRUE,
        force.samp = FALSE,
        V = NULL,
        optim.method = "BFGS",
        optim.lower = -Inf,
        optim.upper = Inf,
        optim.control = list(fnscale = -1, trace = 0, REPORT = 10, maxit = 500),
        ...
    )
```

Arguments

fun	The unnormalized (log)density of the distribution from which to take a sample. This must be a function defined in R whose first argument is a continuous (possibly vector) variable. This first argument is the point in the state space at which the (log)density is to be evaluated. Additional arguments can be passed to `fun()` by inserting them in the call to `MCMCmetrop1R()`. See the Details section and the examples below for more information.
theta.init	Starting values for the sampling. Must be of the appropriate dimension. It must also be the case that fun(theta.init,...) is greater than -Inf if fun() is a logdensity or greater than 0 if fun() is a density.
burnin	The number of burn-in iterations for the sampler.
mcmc	The number of MCMC iterations after burnin.
thin	The thinning interval used in the simulation. The number of MCMC iterations must be divisible by this value.
tune	The tuning parameter for the Metropolis sampling. Can be either a positive scalar or a k-vector, where k is the length of θ.
verbose	A switch which determines whether or not the progress of the sampler is printed to the screen. If verbose is greater than 0 the iteration number, the θ vector, the function value, and the Metropolis acceptance rate are sent to the screen every verboseth iteration.
seed	The seed for the random number generator. If NA, the Mersenne Twister generator is used with default seed 12345; if an integer is passed it is used to seed the Mersenne twister. The user can also pass a list of length two to use the L'Ecuyer random number generator, which is suitable for parallel computation. The first element of the list is the L'Ecuyer seed, which is a vector of length six or NA (if NA a default seed of rep(12345,6) is used). The second element of list is a positive substream number. See the MCMCpack specification for more details.
logfun	Logical indicating whether fun returns the natural log of a density function (TRUE) or a density (FALSE).
force.samp	Logical indicating whether the sampling should proceed if the Hessian calculated from the initial call to optim routine to maximize the (log)density is not negative definite. If force.samp==TRUE and the Hessian from optim is non-negative definite, the Hessian is rescaled by subtracting small values from

it's main diagonal until it is negative definite. Sampling proceeds using this rescaled Hessian in place of the original Hessian from optim. By default, if force.samp==FALSE and the Hessian from optim is non-negative definite, an error message is printed and the call to MCMCmetrop1R is terminated.

Please note that a non-negative Hessian at the mode is often an indication that the function of interest is not a proper density. Thus, force.samp *should only be set equal to* TRUE *with great caution.*

V	The variance-covariance matrix for the Gaussian proposal distribution. Must be a square matrix or NULL. If a square matrix, V must have dimension equal to the length of theta.init. If NULL, V is calculated from tune and an initial call to optim. See the Details section below for more information. Unless the log-posterior is expensive to compute it will typically be best to use the default V = NULL.
optim.method	The value of the method parameter sent to optim during an initial maximization of fun. See optim for more details.
optim.lower	The value of the lower parameter sent to optim during an initial maximization of fun. See optim for more details.
optim.upper	The value of the upper parameter sent to optim during an initial maximization of fun. See optim for more details.
optim.control	The value of the control parameter sent to optim during an initial maximization of fun. See optim for more details.
...	Additional arguments.

Details

MCMCmetrop1R produces a sample from a user-defined distribution using a random walk Metropolis algorithm with multivariate normal proposal distribution. See Gelman et al. (2003) and Robert & Casella (2004) for details of the random walk Metropolis algorithm.

The proposal distribution is centered at the current value of θ and has variance-covariance V. If V is specified by the user to be NULL then V is calculated as: $V = T(-1 \cdot H)^{-1}T$, where T is a the diagonal positive definite matrix formed from the tune and H is the approximate Hessian of fun evaluated at its mode. This last calculation is done via an initial call to optim.

Value

An mcmc object that contains the posterior sample. This object can be summarized by functions provided by the coda package.

References

Siddhartha Chib; Edward Greenberg. 1995. "Understanding the Metropolis-Hastings Algorithm." *The American Statistician*, 49, 327-335.

Andrew Gelman, John B. Carlin, Hal S. Stern, and Donald B. Rubin. 2003. *Bayesian Data Analysis.* 2nd Edition. Boca Raton: Chapman & Hall/CRC.

Andrew D. Martin, Kevin M. Quinn, and Jong Hee Park. 2011. "MCMCpack: Markov Chain Monte Carlo in R.", *Journal of Statistical Software*. 42(9): 1-21. doi: 10.18637/jss.v042.i09.

Daniel Pemstein, Kevin M. Quinn, and Andrew D. Martin. 2007. *Scythe Statistical Library 1.0.*
http://scythe.lsa.umich.edu.

Martyn Plummer, Nicky Best, Kate Cowles, and Karen Vines. 2006. "Output Analysis and Diagnostics for MCMC (CODA)", *R News*. 6(1): 7-11. https://CRAN.R-project.org/doc/Rnews/Rnews_2006-1.pdf.

Christian P. Robert and George Casella. 2004. *Monte Carlo Statistical Methods*. 2nd Edition. New York: Springer.

See Also

plot.mcmc, summary.mcmc, optim, metrop

Examples

```
## Not run:

  ## logistic regression with an improper uniform prior
  ## X and y are passed as args to MCMCmetrop1R

  logitfun <- function(beta, y, X){
    eta <- X %*% beta
    p <- 1.0/(1.0+exp(-eta))
    sum( y * log(p) + (1-y)*log(1-p) )
  }

  x1 <- rnorm(1000)
  x2 <- rnorm(1000)
  Xdata <- cbind(1,x1,x2)
  p <- exp(.5 - x1 + x2)/(1+exp(.5 - x1 + x2))
  yvector <- rbinom(1000, 1, p)

  post.samp <- MCMCmetrop1R(logitfun, theta.init=c(0,0,0),
                            X=Xdata, y=yvector,
                            thin=1, mcmc=40000, burnin=500,
                            tune=c(1.5, 1.5, 1.5),
                            verbose=500, logfun=TRUE)

  raftery.diag(post.samp)
  plot(post.samp)
  summary(post.samp)
  ## ##################################################


  ##  negative binomial regression with an improper unform prior
  ## X and y are passed as args to MCMCmetrop1R
  negbinfun <- function(theta, y, X){
    k <- length(theta)
    beta <- theta[1:(k-1)]
    alpha <- exp(theta[k])
    mu <- exp(X %*% beta)
```

```
        log.like <- sum(
                       lgamma(y+alpha) - lfactorial(y) - lgamma(alpha) +
                       alpha * log(alpha/(alpha+mu)) +
                       y * log(mu/(alpha+mu))
                       )
    }

    n <- 1000
    x1 <- rnorm(n)
    x2 <- rnorm(n)
    XX <- cbind(1,x1,x2)
    mu <- exp(1.5+x1+2*x2)*rgamma(n,1)
    yy <- rpois(n, mu)

    post.samp <- MCMCmetrop1R(negbinfun, theta.init=c(0,0,0,0), y=yy, X=XX,
                              thin=1, mcmc=35000, burnin=1000,
                              tune=1.5, verbose=500, logfun=TRUE,
                              seed=list(NA,1))
    raftery.diag(post.samp)
    plot(post.samp)
    summary(post.samp)
    ## ###################################################


    ## sample from a univariate normal distribution with
    ## mean 5 and standard deviation 0.1
    ##
    ## (MCMC obviously not necessary here and this should
    ##  really be done with the logdensity for better
    ##  numerical accuracy-- this is just an illustration of how
    ##  MCMCmetrop1R works with a density rather than logdensity)

    post.samp <- MCMCmetrop1R(dnorm, theta.init=5.3, mean=5, sd=0.1,
                            thin=1, mcmc=50000, burnin=500,
                            tune=2.0, verbose=5000, logfun=FALSE)

    summary(post.samp)


  ## End(Not run)
```

MCMCmixfactanal   *Markov Chain Monte Carlo for Mixed Data Factor Analysis Model*

Description

This function generates a sample from the posterior distribution of a mixed data (both continuous and ordinal) factor analysis model. Normal priors are assumed on the factor loadings and factor

scores, improper uniform priors are assumed on the cutpoints, and inverse gamma priors are assumed for the error variances (uniquenesses). The user supplies data and parameters for the prior distributions, and a sample from the posterior distribution is returned as an mcmc object, which can be subsequently analyzed with functions provided in the coda package.

Usage

```
MCMCmixfactanal(
  x,
  factors,
  lambda.constraints = list(),
  data = parent.frame(),
  burnin = 1000,
  mcmc = 20000,
  thin = 1,
  tune = NA,
  verbose = 0,
  seed = NA,
  lambda.start = NA,
  psi.start = NA,
  l0 = 0,
  L0 = 0,
  a0 = 0.001,
  b0 = 0.001,
  store.lambda = TRUE,
  store.scores = FALSE,
  std.mean = TRUE,
  std.var = TRUE,
  ...
)
```

Arguments

x   A one-sided formula containing the manifest variables. Ordinal (including di-
                     chotomous) variables must be coded as ordered factors. Each level of these
                     ordered factors must be present in the data passed to the function. NOTE:
                     data input is different in MCMCmixfactanal than in either MCMCfactanal or
                     MCMCordfactanal.

factors   The number of factors to be fitted.

lambda.constraints
                     List of lists specifying possible equality or simple inequality constraints on the
                     factor loadings. A typical entry in the list has one of three forms: varname=list(d,c)
                     which will constrain the dth loading for the variable named varname to be equal
                     to c, varname=list(d,"+") which will constrain the dth loading for the vari-
                     able named varname to be positive, and varname=list(d, "-") which will con-
                     strain the dth loading for the variable named varname to be negative. If x is a
                     matrix without column names defaults names of "V1", "V2", ... , etc will be
                     used. Note that, unlike MCMCfactanal, the Λ matrix used here has factors+1

	columns. The first column of Λ corresponds to negative item difficulty parameters for ordinal manifest variables and mean parameters for continuous manifest variables and should generally not be constrained directly by the user.
data	A data frame.
burnin	The number of burn-in iterations for the sampler.
mcmc	The number of iterations for the sampler.
thin	The thinning interval used in the simulation. The number of iterations must be divisible by this value.
tune	The tuning parameter for the Metropolis-Hastings sampling. Can be either a scalar or a k-vector (where k is the number of manifest variables). tune must be strictly positive.
verbose	A switch which determines whether or not the progress of the sampler is printed to the screen. If verbose is great than 0 the iteration number and the Metropolis-Hastings acceptance rate are printed to the screen every verboseth iteration.
seed	The seed for the random number generator. If NA, the Mersenne Twister generator is used with default seed 12345; if an integer is passed it is used to seed the Mersenne twister. The user can also pass a list of length two to use the L'Ecuyer random number generator, which is suitable for parallel computation. The first element of the list is the L'Ecuyer seed, which is a vector of length six or NA (if NA a default seed of rep(12345,6) is used). The second element of list is a positive substream number. See the MCMCpack specification for more details.
lambda.start	Starting values for the factor loading matrix Lambda. If lambda.start is set to a scalar the starting value for all unconstrained loadings will be set to that scalar. If lambda.start is a matrix of the same dimensions as Lambda then the lambda.start matrix is used as the starting values (except for equality-constrained elements). If lambda.start is set to NA (the default) then starting values for unconstrained elements in the first column of Lambda are based on the observed response pattern, the remaining unconstrained elements of Lambda are set to 0, and starting values for inequality constrained elements are set to either 1.0 or -1.0 depending on the nature of the constraints.
psi.start	Starting values for the error variance (uniqueness) matrix. If psi.start is set to a scalar then the starting value for all diagonal elements of Psi that represent error variances for continuous variables are set to this value. If psi.start is a k-vector (where k is the number of manifest variables) then the staring value of Psi has psi.start on the main diagonal with the exception that entries corresponding to error variances for ordinal variables are set to 1.. If psi.start is set to NA (the default) the starting values of all the continuous variable uniquenesses are set to 0.5. Error variances for ordinal response variables are always constrained (regardless of the value of psi.start to have an error variance of 1 in order to achieve identification.
l0	The means of the independent Normal prior on the factor loadings. Can be either a scalar or a matrix with the same dimensions as Lambda.
L0	The precisions (inverse variances) of the independent Normal prior on the factor loadings. Can be either a scalar or a matrix with the same dimensions as Lambda.
a0	Controls the shape of the inverse Gamma prior on the uniqueness. The actual shape parameter is set to a0/2. Can be either a scalar or a k-vector.

b0   Controls the scale of the inverse Gamma prior on the uniquenesses. The actual
                       scale parameter is set to b0/2. Can be either a scalar or a k-vector.

store.lambda   A switch that determines whether or not to store the factor loadings for posterior
                       analysis. By default, the factor loadings are all stored.

store.scores   A switch that determines whether or not to store the factor scores for posterior
                       analysis. *NOTE: This takes an enormous amount of memory, so should only be
                       used if the chain is thinned heavily, or for applications with a small number of
                       observations*. By default, the factor scores are not stored.

std.mean   If TRUE (the default) the continuous manifest variables are rescaled to have zero
                       mean.

std.var   If TRUE (the default) the continuous manifest variables are rescaled to have unit
                       variance.

...   further arguments to be passed

Details

The model takes the following form:

Let $i = 1, \ldots, N$ index observations and $j = 1, \ldots, K$ index response variables within an observation. An observed variable x_{ij} can be either ordinal with a total of C_j categories or continuous. The distribution of X is governed by a $N \times K$ matrix of latent variables X^* and a series of cutpoints γ. X^* is assumed to be generated according to:

$$x_i^* = \Lambda \phi_i + \epsilon_i$$

$$\epsilon_i \sim \mathcal{N}(0, \Psi)$$

where x_i^* is the k-vector of latent variables specific to observation i, Λ is the $k \times d$ matrix of factor loadings, and ϕ_i is the d-vector of latent factor scores. It is assumed that the first element of ϕ_i is equal to 1 for all i.

If the jth variable is ordinal, the probability that it takes the value c in observation i is:

$$\pi_{ijc} = \Phi(\gamma_{jc} - \Lambda_j' \phi_i) - \Phi(\gamma_{j(c-1)} - \Lambda_j' \phi_i)$$

If the jth variable is continuous, it is assumed that $x_{ij}^* = x_{ij}$ for all i.

The implementation used here assumes independent conjugate priors for each element of Λ and each ϕ_i. More specifically we assume:

$$\Lambda_{ij} \sim \mathcal{N}(l_{0_{ij}}, L_{0_{ij}}^{-1}), i = 1, \ldots, k, j = 1, \ldots, d$$

$$\phi_{i(2:d)} \sim \mathcal{N}(0, I), i = 1, \ldots, n$$

MCMCmixfactanal simulates from the posterior distribution using a Metropolis-Hastings within Gibbs sampling algorithm. The algorithm employed is based on work by Cowles (1996). Note that

the first element of ϕ_i is a 1. As a result, the first column of Λ can be interpretated as negative item difficulty parameters. Further, the first element γ_1 is normalized to zero, and thus not returned in the mcmc object. The simulation proper is done in compiled C++ code to maximize efficiency. Please consult the coda documentation for a comprehensive list of functions that can be used to analyze the posterior sample.

As is the case with all measurement models, make sure that you have plenty of free memory, especially when storing the scores.

Value

An mcmc object that contains the posterior sample. This object can be summarized by functions provided by the coda package.

References

Kevin M. Quinn. 2004. "Bayesian Factor Analysis for Mixed Ordinal and Continuous Responses." *Political Analysis*. 12: 338-353.

Andrew D. Martin, Kevin M. Quinn, and Jong Hee Park. 2011. "MCMCpack: Markov Chain Monte Carlo in R.", *Journal of Statistical Software*. 42(9): 1-21. doi: 10.18637/jss.v042.i09.

M. K. Cowles. 1996. "Accelerating Monte Carlo Markov Chain Convergence for Cumulative-link Generalized Linear Models." *Statistics and Computing.* 6: 101-110.

Valen E. Johnson and James H. Albert. 1999. "Ordinal Data Modeling." Springer: New York.

Daniel Pemstein, Kevin M. Quinn, and Andrew D. Martin. 2007. *Scythe Statistical Library 1.0.* http://scythe.lsa.umich.edu.

Martyn Plummer, Nicky Best, Kate Cowles, and Karen Vines. 2006. "Output Analysis and Diagnostics for MCMC (CODA)", *R News*. 6(1): 7-11. https://CRAN.R-project.org/doc/Rnews/Rnews_2006-1.pdf.

See Also

plot.mcmc, summary.mcmc, factanal, MCMCfactanal, MCMCordfactanal, MCMCirt1d, MCMCirtKd

Examples

```
## Not run:
data(PErisk)

post <- MCMCmixfactanal(~courts+barb2+prsexp2+prscorr2+gdpw2,
                        factors=1, data=PErisk,
                        lambda.constraints = list(courts=list(2,"-")),
                        burnin=5000, mcmc=1000000, thin=50,
                        verbose=500, L0=.25, store.lambda=TRUE,
                        store.scores=TRUE, tune=1.2)
plot(post)
summary(post)
```

```
library(MASS)
data(Cars93)
attach(Cars93)
new.cars <- data.frame(Price, MPG.city, MPG.highway,
                Cylinders, EngineSize, Horsepower,
                RPM, Length, Wheelbase, Width, Weight, Origin)
rownames(new.cars) <- paste(Manufacturer, Model)
detach(Cars93)

# drop obs 57 (Mazda RX 7) b/c it has a rotary engine
new.cars <- new.cars[-57,]
# drop 3 cylinder cars
new.cars <- new.cars[new.cars$Cylinders!=3,]
# drop 5 cylinder cars
new.cars <- new.cars[new.cars$Cylinders!=5,]

new.cars$log.Price <- log(new.cars$Price)
new.cars$log.MPG.city <- log(new.cars$MPG.city)
new.cars$log.MPG.highway <- log(new.cars$MPG.highway)
new.cars$log.EngineSize <- log(new.cars$EngineSize)
new.cars$log.Horsepower <- log(new.cars$Horsepower)

new.cars$Cylinders <- ordered(new.cars$Cylinders)
new.cars$Origin    <- ordered(new.cars$Origin)



post <- MCMCmixfactanal(~log.Price+log.MPG.city+
                log.MPG.highway+Cylinders+log.EngineSize+
                log.Horsepower+RPM+Length+
                Wheelbase+Width+Weight+Origin, data=new.cars,
                lambda.constraints=list(log.Horsepower=list(2,"+"),
                log.Horsepower=c(3,0), weight=list(3,"+")),
                factors=2,
                burnin=5000, mcmc=500000, thin=100, verbose=500,
                L0=.25, tune=3.0)
plot(post)
summary(post)


## End(Not run)
```

MCMCmnl   *Markov Chain Monte Carlo for Multinomial Logistic Regression*

Description

This function generates a sample from the posterior distribution of a multinomial logistic regression model using either a random walk Metropolis algorithm or a slice sampler. The user supplies data

and priors, and a sample from the posterior distribution is returned as an mcmc object, which can be subsequently analyzed with functions provided in the coda package.

Usage

```
MCMCmnl(
  formula,
  baseline = NULL,
  data = NULL,
  burnin = 1000,
  mcmc = 10000,
  thin = 1,
  mcmc.method = "IndMH",
  tune = 1,
  tdf = 6,
  verbose = 0,
  seed = NA,
  beta.start = NA,
  b0 = 0,
  B0 = 0,
  ...
)
```

Arguments

formula   Model formula.

If the choicesets do not vary across individuals, the y variable should be a factor or numeric variable that gives the observed choice of each individual. If the choicesets do vary across individuals, y should be a $n \times p$ matrix where n is the number of individuals and p is the maximum number of choices in any choiceset. Here each column of y corresponds to a particular observed choice and the elements of y should be either 0 (not chosen but available), 1 (chosen), or -999 (not available).

Choice-specific covariates have to be entered using the syntax: choicevar(cvar, "var", "choice") where cvar is the name of a variable in data, "var" is the name of the new variable to be created, and "choice" is the level of y that cvar corresponds to. Specifying each choice-specific covariate will typically require p calls to the choicevar function in the formula.

Individual specific covariates can be entered into the formula normally.

See the examples section below to see the syntax used to fit various models.

baseline   The baseline category of the response variable.

baseline should be set equal to one of the observed levels of the response variable. If left equal to NULL then the baseline level is set to the alpha-numerically first element of the response variable. If the choicesets vary across individuals, the baseline choice must be in the choiceset of each individual.

data   The data frame used for the analysis. Each row of the dataframe should correspond to an individual who is making a choice.

burnin	The number of burn-in iterations for the sampler.
mcmc	The number of iterations to run the sampler past burn-in.
thin	The thinning interval used in the simulation. The number of mcmc iterations must be divisible by this value.
mcmc.method	Can be set to either "IndMH" (default), "RWM", or "slice" to perform independent Metropolis-Hastings sampling, random walk Metropolis sampling or slice sampling respectively.
tune	Metropolis tuning parameter. Can be either a positive scalar or a k-vector, where k is the length of β. Make sure that the acceptance rate is satisfactory (typically between 0.20 and 0.5) before using the posterior sample for inference.
tdf	Degrees of freedom for the multivariate-t proposal distribution when mcmc.method is set to "IndMH". Must be positive.
verbose	A switch which determines whether or not the progress of the sampler is printed to the screen. If verbose is greater than 0 the iteration number, the current beta vector, and the Metropolis acceptance rate are printed to the screen every verboseth iteration.
seed	The seed for the random number generator. If NA, the Mersenne Twister generator is used with default seed 12345; if an integer is passed it is used to seed the Mersenne twister. The user can also pass a list of length two to use the L'Ecuyer random number generator, which is suitable for parallel computation. The first element of the list is the L'Ecuyer seed, which is a vector of length six or NA (if NA a default seed of rep(12345,6) is used). The second element of list is a positive substream number. See the MCMCpack specification for more details.
beta.start	The starting value for the β vector. This can either be a scalar or a column vector with dimension equal to the number of betas. If this takes a scalar value, then that value will serve as the starting value for all of the betas. The default value of NA will use the maximum likelihood estimate of β as the starting value.
b0	The prior mean of β. This can either be a scalar or a column vector with dimension equal to the number of betas. If this takes a scalar value, then that value will serve as the prior mean for all of the betas.
B0	The prior precision of β. This can either be a scalar or a square matrix with dimensions equal to the number of betas. If this takes a scalar value, then that value times an identity matrix serves as the prior precision of β. Default value of 0 is equivalent to an improper uniform prior for beta.
...	Further arguments to be passed.

Details

MCMCmnl simulates from the posterior distribution of a multinomial logistic regression model using either a random walk Metropolis algorithm or a univariate slice sampler. The simulation proper is done in compiled C++ code to maximize efficiency. Please consult the coda documentation for a comprehensive list of functions that can be used to analyze the posterior sample.

The model takes the following form:

$$y_i \sim \mathcal{Multinomial}(\pi_i)$$

where:

$$\pi_{ij} = \frac{\exp(x'_{ij}\beta)}{\sum_{k=1}^{p}\exp(x'_{ik}\beta)}$$

We assume a multivariate Normal prior on β:

$$\beta \sim \mathcal{N}(b_0, B_0^{-1})$$

The Metropolis proposal distribution is centered at the current value of β and has variance-covariance $V = T(B_0 + C^{-1})^{-1}T$, where T is a the diagonal positive definite matrix formed from the tune, B_0 is the prior precision, and C is the large sample variance-covariance matrix of the MLEs. This last calculation is done via an initial call to optim.

Value

An mcmc object that contains the posterior sample. This object can be summarized by functions provided by the coda package.

References

Andrew D. Martin, Kevin M. Quinn, and Jong Hee Park. 2011. "MCMCpack: Markov Chain Monte Carlo in R.", *Journal of Statistical Software*. 42(9): 1-21. doi: 10.18637/jss.v042.i09.

Daniel Pemstein, Kevin M. Quinn, and Andrew D. Martin. 2007. *Scythe Statistical Library 1.0*. http://scythe.lsa.umich.edu.

Radford Neal. 2003. "Slice Sampling" (with discussion). *Annals of Statistics*, 31: 705-767.

Martyn Plummer, Nicky Best, Kate Cowles, and Karen Vines. 2006. "Output Analysis and Diagnostics for MCMC (CODA)", *R News*. 6(1): 7-11. https://CRAN.R-project.org/doc/Rnews/Rnews_2006-1.pdf.

Siddhartha Chib, Edward Greenberg, and Yuxin Chen. 1998. "MCMC Methods for Fitting and Comparing Multinomial Response Models."

See Also

plot.mcmc,summary.mcmc, multinom

Examples

```
## Not run:
data(Nethvote)

## just a choice-specific X var
post1 <- MCMCmnl(vote ~
            choicevar(distD66, "sqdist", "D66") +
            choicevar(distPvdA, "sqdist", "PvdA") +
            choicevar(distVVD, "sqdist", "VVD") +
            choicevar(distCDA, "sqdist", "CDA"),
            baseline="D66", mcmc.method="IndMH", B0=0,
```

```
              verbose=500, mcmc=100000, thin=10, tune=1.0,
              data=Nethvote)

plot(post1)
summary(post1)



## just individual-specific X vars
post2<- MCMCmnl(vote ~
              relig + class + income + educ + age + urban,
              baseline="D66", mcmc.method="IndMH", B0=0,
              verbose=500, mcmc=100000, thin=10, tune=0.5,
              data=Nethvote)

plot(post2)
summary(post2)



## both choice-specific and individual-specific X vars
post3 <- MCMCmnl(vote ~
              choicevar(distD66, "sqdist", "D66") +
              choicevar(distPvdA, "sqdist", "PvdA") +
              choicevar(distVVD, "sqdist", "VVD") +
              choicevar(distCDA, "sqdist", "CDA") +
              relig + class + income + educ + age + urban,
              baseline="D66", mcmc.method="IndMH", B0=0,
              verbose=500, mcmc=100000, thin=10, tune=0.5,
              data=Nethvote)

plot(post3)
summary(post3)



## numeric y variable
nethvote$vote <- as.numeric(nethvote$vote)
post4 <- MCMCmnl(vote ~
              choicevar(distD66, "sqdist", "2") +
              choicevar(distPvdA, "sqdist", "3") +
              choicevar(distVVD, "sqdist", "4") +
              choicevar(distCDA, "sqdist", "1") +
              relig + class + income + educ + age + urban,
              baseline="2", mcmc.method="IndMH", B0=0,
              verbose=500, mcmc=100000, thin=10, tune=0.5,
              data=Nethvote)

plot(post4)
summary(post4)
```

```
## Simulated data example with nonconstant choiceset
n <- 1000
y <- matrix(0, n, 4)
colnames(y) <- c("a", "b", "c", "d")
xa <- rnorm(n)
xb <- rnorm(n)
xc <- rnorm(n)
xd <- rnorm(n)
xchoice <- cbind(xa, xb, xc, xd)
z <- rnorm(n)
for (i in 1:n){
  ## randomly determine choiceset (c is always in choiceset)
  choiceset <- c(3, sample(c(1,2,4), 2, replace=FALSE))
  numer <- matrix(0, 4, 1)
  for (j in choiceset){
    if (j == 3){
      numer[j] <- exp(xchoice[i, j] )
    }
    else {
      numer[j] <- exp(xchoice[i, j] - z[i] )
    }
  }
  p <- numer / sum(numer)
  y[i,] <- rmultinom(1, 1, p)
  y[i,-choiceset] <- -999
}

post5 <- MCMCmnl(y~choicevar(xa, "x", "a") +
                 choicevar(xb, "x", "b") +
                 choicevar(xc, "x", "c") +
                 choicevar(xd, "x", "d") + z,
                 baseline="c", verbose=500,
                 mcmc=100000, thin=10, tune=.85)

plot(post5)
summary(post5)


## End(Not run)
```

MCMCnegbin   *Markov Chain Monte Carlo for Negative Binomial Regression*

Description

This function generates a sample from the posterior distribution of a Negative Binomial regression model via auxiliary mixture sampling. The user supplies data and priors, and a sample from the posterior distribution is returned as an mcmc object, which can be subsequently analyzed with functions provided in the coda package.

Usage

```
MCMCnegbin(
  formula,
  data = parent.frame(),
  b0 = 0,
  B0 = 1,
  e = 2,
  f = 2,
  g = 10,
  burnin = 1000,
  mcmc = 1000,
  thin = 1,
  verbose = 0,
  seed = NA,
  beta.start = NA,
  rho.start = NA,
  rho.step = 0.1,
  nu.start = NA,
  marginal.likelihood = c("none", "Chib95"),
  ...
)
```

Arguments

formula	Model formula.
data	Data frame.
b0	The prior mean of β. This can either be a scalar or a column vector with dimension equal to the number of betas. If this takes a scalar value, then that value will serve as the prior mean for all of the betas.
B0	The prior precision of β. This can either be a scalar or a square matrix with dimensions equal to the number of betas. If this takes a scalar value, then that value times an identity matrix serves as the prior precision of β. Default value of 0 is equivalent to an improper uniform prior for beta.
e	The hyperprior for the distribution ρ. See details.
f	The hyperprior for the distribution ρ. See details.
g	The hyperprior for the distribution ρ. See details.
burnin	The number of burn-in iterations for the sampler.
mcmc	The number of Metropolis iterations for the sampler.
thin	The thinning interval used in the simulation. The number of mcmc iterations must be divisible by this value.
verbose	A switch which determines whether or not the progress of the sampler is printed to the screen. If verbose is greater than 0 the iteration number, the current beta vector, and the Metropolis acceptance rate are printed to the screen every verboseth iteration.

seed
: The seed for the random number generator. If NA, the Mersenne Twister generator is used with default seed 12345; if an integer is passed it is used to seed the Mersenne twister. The user can also pass a list of length two to use the L'Ecuyer random number generator, which is suitable for parallel computation. The first element of the list is the L'Ecuyer seed, which is a vector of length six or NA (if NA a default seed of rep(12345,6) is used). The second element of list is a positive substream number. See the MCMCpack specification for more details.

beta.start
: The starting value for the β vector. This can either be a scalar or a column vector with dimension equal to the number of betas. If this takes a scalar value, then that value will serve as the starting value for all of the betas. The default value of NA will use the maximum likelihood estimate of β as the starting value.

rho.start
: The starting value for the ρ variable. The default value is 1.

rho.step
: Tuning parameter for the slice sampling approach to sampling rho. Determines the size of the step-out used to find the correct slice to draw from. Lower values are more accurate, but will take longer (up to a fixed searching limit). Default is 0.1.

nu.start
: The starting values for the random effect, ν. The default value is a vector of ones.

marginal.likelihood
: How should the marginal likelihood be calculated? Options are: none in which case the marginal likelihood will not be calculated or Laplace in which case the Laplace approximation (see Kass and Raftery, 1995) is used.

...
: further arguments to be passed.

Details

MCMCnegbin simulates from the posterior distribution of a Negative Binomial regression model using a combination of auxiliary mixture sampling and slice sampling. The simulation proper is done in compiled C++ code to maximize efficiency. Please consult the coda documentation for a comprehensive list of functions that can be used to analyze the posterior sample.

The model takes the following form:

$$y_i \sim \mathcal{P}oisson(\nu_i \mu_i)$$

Where the inverse link function:

$$\mu_i = \exp(x_i'\beta)$$

We assume a multivariate Normal prior on β:

$$\beta \sim \mathcal{N}(b_0, B_0^{-1})$$

The unit-level random effect that handles overdispersion is assumed to be distributed Gamma:

$$\nu_i \sim \mathcal{G}amma(\rho, \rho)$$

The overdispersion parameter has a prior with the following form:

$$f(\rho|e, f, g) \propto \rho^{e-1}(\rho + g)^{-(e+f)}$$

The model is simulated via blocked Gibbs, with the the β being simulated via the auxiliary mixture sampling method of Fuerhwirth-Schanetter et al. (2009). The ρ is updated via slice sampling. The ν_i are updated their (conjugate) full conditional, which is also Gamma.

Value

An mcmc object that contains the posterior sample. This object can be summarized by functions provided by the coda package.

References

Andrew D. Martin, Kevin M. Quinn, and Jong Hee Park. 2011. "MCMCpack: Markov Chain Monte Carlo in R.", *Journal of Statistical Software*. 42(9): 1-21. doi: 10.18637/jss.v042.i09.

Daniel Pemstein, Kevin M. Quinn, and Andrew D. Martin. 2007. *Scythe Statistical Library 1.0.* http://scythe.lsa.umich.edu.

Martyn Plummer, Nicky Best, Kate Cowles, and Karen Vines. 2006. "Output Analysis and Diagnostics for MCMC (CODA)", *R News*. 6(1): 7-11. https://CRAN.R-project.org/doc/Rnews/Rnews_2006-1.pdf.

Sylvia Fruehwirth-Schnatter, Rudolf Fruehwirth, Leonhard Held, and Havard Rue. 2009. "Improved auxiliary mixture sampling for hierarchical models of non-Gaussian data", *Statistics and Computing* 19(4): 479-492. <doi:10.1007/s11222-008-9109-4>

See Also

plot.mcmc,summary.mcmc, glm.nb

Examples

```
## Not run:
  n <- 150
  mcmcs <- 5000
  burnin <- 5000
  thin <- 5
  x1 <- runif(n, 0, 2)
  rho.true <- 1.5
  nu.true <- rgamma(n, rho.true, rho.true)
  mu <- nu.true * exp(1 + x1 * 1)
  y <- rpois(n, mu)
  posterior <- MCMCnegbin(y ~ x1)
  plot(posterior)
  summary(posterior)

## End(Not run)
```

MCMCnegbinChange   *Markov Chain Monte Carlo for Negative Binomial Regression Changepoint Model*

Description

This function generates a sample from the posterior distribution of a Negative Binomial regression model with multiple changepoints. For the changepoints, the sampler uses the Markov Chain Monte Carlo method of Chib (1998). The user supplies data and priors, and a sample from the posterior distribution is returned as an mcmc object, which can be subsequently analyzed with functions provided in the coda package.

Usage

```
MCMCnegbinChange(
  formula,
  data = parent.frame(),
  m = 1,
  fixed.m = TRUE,
  b0 = 0,
  B0 = 1,
  a = NULL,
  b = NULL,
  e = 2,
  f = 2,
  g = 10,
  burnin = 1000,
  mcmc = 1000,
  thin = 1,
  verbose = 0,
  seed = NA,
  beta.start = NA,
  P.start = NA,
  rho.start = NA,
  rho.step,
  nu.start = NA,
  marginal.likelihood = c("none", "Chib95"),
  ...
)
```

Arguments

formula   Model formula.

data   Data frame.

m   The number of changepoints.

fixed.m	A logical indicator for whether or not the number of changepoints in the sampler should be exactly equal to m or if that is simply an upper bound. Setting fixed.m to FALSE is equivalent to assuming a weak-limit approximation to a Dirichlet process mixture.
b0	The prior mean of β. This can either be a scalar or a column vector with dimension equal to the number of betas. If this takes a scalar value, then that value will serve as the prior mean for all of the betas.
B0	The prior precision of β. This can either be a scalar or a square matrix with dimensions equal to the number of betas. If this takes a scalar value, then that value times an identity matrix serves as the prior precision of beta. Default value of 0 is equivalent to an improper uniform prior for beta.
a	a is the shape1 beta prior for transition probabilities. By default, the expected duration is computed and corresponding a and b values are assigned. The expected duration is the sample period divided by the number of states.
b	b is the shape2 beta prior for transition probabilities. By default, the expected duration is computed and corresponding a and b values are assigned. The expected duration is the sample period divided by the number of states.
e	The hyperprior for the distribution ρ See details.
f	The hyperprior for the distribution ρ. See details.
g	The hyperprior for the distribution ρ. See details.
burnin	The number of burn-in iterations for the sampler.
mcmc	The number of Metropolis iterations for the sampler.
thin	The thinning interval used in the simulation. The number of mcmc iterations must be divisible by this value.
verbose	A switch which determines whether or not the progress of the sampler is printed to the screen. If verbose is greater than 0 the iteration number, the current beta vector, and the Metropolis acceptance rate are printed to the screen every verboseth iteration.
seed	The seed for the random number generator. If NA, the Mersenne Twister generator is used with default seed 12345; if an integer is passed it is used to seed the Mersenne twister. The user can also pass a list of length two to use the L'Ecuyer random number generator, which is suitable for parallel computation. The first element of the list is the L'Ecuyer seed, which is a vector of length six or NA (if NA a default seed of rep(12345,6) is used). The second element of list is a positive substream number. See the MCMCpack specification for more details.
beta.start	The starting value for the β vector. This can either be a scalar or a column vector with dimension equal to the number of betas. If this takes a scalar value, then that value will serve as the starting value for all of the betas. The default value of NA will use the maximum likelihood estimate of β as the starting value for all regimes.
P.start	The starting values for the transition matrix. A user should provide a square matrix with dimension equal to the number of states. By default, draws from the Beta(0.9, 0.1) are used to construct a proper transition matrix for each raw except the last raw.

rho.start	The starting value for the ρ variable. This can either be a scalar or a column vector with dimension equal to the number of regimes. If the value is scalar, it will be used for all regimes. The default value is a vector of ones.
rho.step	Tuning parameter for the slice sampling approach to sampling rho. Determines the size of the step-out used to find the correct slice to draw from. Lower values are more accurate, but will take longer (up to a fixed searching limit). Default is 0.1.
nu.start	The starting values for the random effect, ν. The default value is a vector of ones.
marginal.likelihood	
	How should the marginal likelihood be calculated? Options are: none in which case the marginal likelihood will not be calculated or Laplace in which case the Laplace approximation (see Kass and Raftery, 1995) is used.
...	further arguments to be passed.

Details

MCMCnegbinChangesimulates from the posterior distribution of a Negative Binomial regression model with multiple changepoints using the methods of Chib (1998) and Fruehwirth-Schnatter et al (2009). The details of the model are discussed in Blackwell (2017).

The model takes the following form:

$$y_t \sim \mathcal{P}oisson(\nu_t \mu_t)$$

$$\mu_t = x'_t \beta_m, \quad m = 1, \ldots, M$$

$$\nu_t \sim \mathcal{G}amma(\rho_m, \rho_m)$$

Where M is the number of states and β_m and ρ_m are parameters when a state is m at t.

We assume Gaussian distribution for prior of β:

$$\beta_m \sim \mathcal{N}(b_0, B_0^{-1}), \quad m = 1, \ldots, M$$

And:

$$p_{mm} \sim \mathcal{B}eta(a, b), \quad m = 1, \ldots, M$$

Where M is the number of states.

The overdispersion parameters have a prior with the following form:

$$f(\rho_m | e, f, g) \propto \rho^{e-1} (\rho + g)^{-(e+f)}$$

The model is simulated via blocked Gibbs conditonal on the states. The β being simulated via the auxiliary mixture sampling method of Fuerhwirth-Schanetter et al. (2009). The ρ is updated via slice sampling. The ν_i are updated their (conjugate) full conditional, which is also Gamma. The states are updated as in Chib (1998)

Value

An mcmc object that contains the posterior sample. This object can be summarized by functions provided by the coda package.

References

Andrew D. Martin, Kevin M. Quinn, and Jong Hee Park. 2011. "MCMCpack: Markov Chain Monte Carlo in R.", *Journal of Statistical Software*. 42(9): 1-21. doi: 10.18637/jss.v042.i09.

Daniel Pemstein, Kevin M. Quinn, and Andrew D. Martin. 2007. *Scythe Statistical Library 1.0.* http://scythe.lsa.umich.edu.

Sylvia Fruehwirth-Schnatter, Rudolf Fruehwirth, Leonhard Held, and Havard Rue. 2009. "Improved auxiliary mixture sampling for hierarchical models of non-Gaussian data", *Statistics and Computing* 19(4): 479-492. <doi:10.1007/s11222-008-9109-4>

Matthew Blackwell. 2017. "Game Changers: Detecting Shifts in Overdispersed Count Data," *Political Analysis* 26(2), 230-239. <doi:10.1017/pan.2017.42>

See Also

MCMCpoissonChange, plotState, plotChangepoint

Examples

```
## Not run:
  n <- 150
  reg <- 3
  true.s <- gl(reg, n/reg, n)
  rho.true <- c(1.5, 0.5, 3)
  b0.true <- c(1, 3, 1)
  b1.true <- c(1, -2, 2)
  x1 <- runif(n, 0, 2)
  nu.true <- rgamma(n, rho.true[true.s], rho.true[true.s])
  mu <- nu.true * exp(b0.true[true.s] + x1 * b1.true[true.s])
  y <- rpois(n, mu)

  posterior <- MCMCnegbinChange(y ~ x1, m = 2, verbose = 1000,
                        marginal.likelihood = "Chib95",
                        e = 2, f = 2, g = 10,
                        b0 = rep(0, 2), B0 = (1/9) * diag(2),
                        rho.step = rep(0.75, times = 3),
                        seed = list(NA, 2))

  par(mfrow=c(attr(posterior, "m") + 1, 1), mai=c(0.4, 0.6, 0.3, 0.05))
  plotState(posterior, legend.control = c(1, 0.6))
  plotChangepoint(posterior, verbose = TRUE, ylab="Density",
 start=1, overlay=TRUE)


  open.ended <- MCMCnegbinChange(y ~ x1, m = 10, verbose = 1000,
                        fixed.m = FALSE, mcmc = 2000, burnin = 2000,
```

```
                            e = 2, f = 2, g = 10,
                            a = 100, b = 1,
                            b0 = rep(0, 2), B0 = (1/9) * diag(2),
                            rho.step = 0.75,
                            seed = list(NA, 2))

    plotState(open.ended, legend.control = c(1, 0.6))

## End(Not run)
```

MCMCoprobit   *Markov Chain Monte Carlo for Ordered Probit Regression*

Description

This function generates a sample from the posterior distribution of an ordered probit regression model using the data augmentation approach of Albert and Chib (1993), with cut-points sampled according to Cowles (1996) or Albert and Chib (2001). The user supplies data and priors, and a sample from the posterior distribution is returned as an mcmc object, which can be subsequently analyzed with functions provided in the coda package.

Usage

```
MCMCoprobit(
  formula,
  data = parent.frame(),
  burnin = 1000,
  mcmc = 10000,
  thin = 1,
  tune = NA,
  tdf = 1,
  verbose = 0,
  seed = NA,
  beta.start = NA,
  b0 = 0,
  B0 = 0,
  a0 = 0,
  A0 = 0,
  mcmc.method = c("Cowles", "AC"),
  ...
)
```

Arguments

formula	Model formula.
data	Data frame.

burnin	The number of burn-in iterations for the sampler.
mcmc	The number of MCMC iterations for the sampler.
thin	The thinning interval used in the simulation. The number of Gibbs iterations must be divisible by this value.
tune	The tuning parameter for the Metropolis-Hastings step. Default of NA corresponds to a choice of 0.05 divided by the number of categories in the response variable.
tdf	Degrees of freedom for the multivariate-t proposal distribution when mcmc.method is set to "IndMH". Must be positive.
verbose	A switch which determines whether or not the progress of the sampler is printed to the screen. If verbose is greater than 0 the iteration number, the beta vector, and the Metropolis-Hastings acceptance rate are printed to the screen every verboseth iteration.
seed	The seed for the random number generator. If NA, the Mersenne Twister generator is used with default seed 12345; if an integer is passed it is used to seed the Mersenne twister. The user can also pass a list of length two to use the L'Ecuyer random number generator, which is suitable for parallel computation. The first element of the list is the L'Ecuyer seed, which is a vector of length six or NA (if NA a default seed of rep(12345,6) is used). The second element of list is a positive substream number. See the MCMCpack specification for more details.
beta.start	The starting value for the β vector. This can either be a scalar or a column vector with dimension equal to the number of betas. If this takes a scalar value, then that value will serve as the starting value for all of the betas. The default value of NA will use rescaled estimates from an ordered logit model.
b0	The prior mean of β. This can either be a scalar or a column vector with dimension equal to the number of betas. If this takes a scalar value, then that value will serve as the prior mean for all of the betas.
B0	The prior precision of β. This can either be a scalar or a square matrix with dimensions equal to the number of betas. If this takes a scalar value, then that value times an identity matrix serves as the prior precision of β. Default value of 0 is equivalent to an improper uniform prior on β.
a0	The prior mean of γ. This can either be a scalar or a column vector with dimension equal to the number of betas. If this takes a scalar value, then that value will serve as the prior mean for all of the betas.
A0	The prior precision of γ. This can either be a scalar or a square matrix with dimensions equal to the number of betas. If this takes a scalar value, then that value times an identity matrix serves as the prior precision of γ. Default value of 0 is equivalent to an improper uniform prior on γ.
mcmc.method	Can be set to either "Cowles" (default) or "AC" to perform posterior sampling of cutpoints based on Cowles (1996) or Albert and Chib (2001) respectively.
...	further arguments to be passed

Details

MCMCoprobit simulates from the posterior distribution of a ordered probit regression model using data augmentation. The simulation proper is done in compiled C++ code to maximize efficiency.

Please consult the coda documentation for a comprehensive list of functions that can be used to analyze the posterior sample.

The observed variable y_i is ordinal with a total of C categories, with distribution governed by a latent variable:

$$z_i = x_i'\beta + \varepsilon_i$$

The errors are assumed to be from a standard Normal distribution. The probabilities of observing each outcome is governed by this latent variable and $C - 1$ estimable cutpoints, which are denoted γ_c. The probability that individual i is in category c is computed by:

$$\pi_{ic} = \Phi(\gamma_c - x_i'\beta) - \Phi(\gamma_{c-1} - x_i'\beta)$$

These probabilities are used to form the multinomial distribution that defines the likelihoods.

MCMCoprobit provides two ways to sample the cutpoints. Cowles (1996) proposes a sampling scheme that groups sampling of a latent variable with cutpoints. In this case, for identification the first element γ_1 is normalized to zero. Albert and Chib (2001) show that we can sample cutpoints indirectly without constraints by transforming cutpoints into real-valued parameters (α).

Value

An mcmc object that contains the posterior sample. This object can be summarized by functions provided by the coda package.

References

Albert, J. H. and S. Chib. 1993. "Bayesian Analysis of Binary and Polychotomous Response Data." *J. Amer. Statist. Assoc.* 88, 669-679

M. K. Cowles. 1996. "Accelerating Monte Carlo Markov Chain Convergence for Cumulative-link Generalized Linear Models." *Statistics and Computing.* 6: 101-110.

Andrew D. Martin, Kevin M. Quinn, and Jong Hee Park. 2011. "MCMCpack: Markov Chain Monte Carlo in R.", *Journal of Statistical Software*. 42(9): 1-21. doi: 10.18637/jss.v042.i09.

Valen E. Johnson and James H. Albert. 1999. *Ordinal Data Modeling*. Springer: New York.

Albert, James and Siddhartha Chib. 2001. "Sequential Ordinal Modeling with Applications to Survival Data." *Biometrics.* 57: 829-836.

Daniel Pemstein, Kevin M. Quinn, and Andrew D. Martin. 2007. *Scythe Statistical Library 1.0.* http://scythe.lsa.umich.edu.

Martyn Plummer, Nicky Best, Kate Cowles, and Karen Vines. 2006. "Output Analysis and Diagnostics for MCMC (CODA)", *R News*. 6(1): 7-11. https://CRAN.R-project.org/doc/Rnews/Rnews_2006-1.pdf.

See Also

plot.mcmc,summary.mcmc

Examples

```
    ## Not run:
    x1 <- rnorm(100); x2 <- rnorm(100);
    z <- 1.0 + x1*0.1 - x2*0.5 + rnorm(100);
    y <- z; y[z < 0] <- 0; y[z >= 0 & z < 1] <- 1;
    y[z >= 1 & z < 1.5] <- 2; y[z >= 1.5] <- 3;
    out1 <- MCMCoprobit(y ~ x1 + x2, tune=0.3)
    out2 <- MCMCoprobit(y ~ x1 + x2, tune=0.3, tdf=3, verbose=1000, mcmc.method="AC")
    summary(out1)
    summary(out2)
    plot(out1)
    plot(out2)

  ## End(Not run)
```

MCMCoprobitChange   *Markov Chain Monte Carlo for Ordered Probit Changepoint Regression Model*

Description

This function generates a sample from the posterior distribution of an ordered probit regression model with multiple parameter breaks. The function uses the Markov chain Monte Carlo method of Chib (1998). The user supplies data and priors, and a sample from the posterior distribution is returned as an mcmc object, which can be subsequently analyzed with functions provided in the coda package.

Usage

```
MCMCoprobitChange(
  formula,
  data = parent.frame(),
  m = 1,
  burnin = 1000,
  mcmc = 1000,
  thin = 1,
  tune = NA,
  verbose = 0,
  seed = NA,
  beta.start = NA,
  gamma.start = NA,
  P.start = NA,
  b0 = NULL,
  B0 = NULL,
  a = NULL,
  b = NULL,
```

```
    marginal.likelihood = c("none", "Chib95"),
    gamma.fixed = 0,
    ...
)
```

Arguments

formula	Model formula.
data	Data frame.
m	The number of changepoints.
burnin	The number of burn-in iterations for the sampler.
mcmc	The number of MCMC iterations after burnin.
thin	The thinning interval used in the simulation. The number of MCMC iterations must be divisible by this value.
tune	The tuning parameter for the Metropolis-Hastings step. Default of NA corresponds to a choice of 0.05 divided by the number of categories in the response variable.
verbose	A switch which determines whether or not the progress of the sampler is printed to the screen. If verbose is greater than 0 the iteration number, the β vector, and the error variance are printed to the screen every verboseth iteration.
seed	The seed for the random number generator. If NA, the Mersenne Twister generator is used with default seed 12345; if an integer is passed it is used to seed the Mersenne twister. The user can also pass a list of length two to use the L'Ecuyer random number generator, which is suitable for parallel computation. The first element of the list is the L'Ecuyer seed, which is a vector of length six or NA (if NA a default seed of rep(12345,6) is used). The second element of list is a positive substream number. See the MCMCpack specification for more details.
beta.start	The starting values for the β vector. This can either be a scalar or a column vector with dimension equal to the number of betas. The default value of of NA will use the MLE estimate of β as the starting value. If this is a scalar, that value will serve as the starting value mean for all of the betas.
gamma.start	The starting values for the γ vector. This can either be a scalar or a column vector with dimension equal to the number of gammas. The default value of of NA will use the MLE estimate of γ as the starting value. If this is a scalar, that value will serve as the starting value mean for all of the gammas.
P.start	The starting values for the transition matrix. A user should provide a square matrix with dimension equal to the number of states. By default, draws from the Beta(0.9, 0.1) are used to construct a proper transition matrix for each raw except the last raw.
b0	The prior mean of β. This can either be a scalar or a column vector with dimension equal to the number of betas. If this takes a scalar value, then that value will serve as the prior mean for all of the betas.
B0	The prior precision of β. This can either be a scalar or a square matrix with dimensions equal to the number of betas. If this takes a scalar value, then that value times an identity matrix serves as the prior precision of beta. Default value of 0 is equivalent to an improper uniform prior for beta.

a a is the shape1 beta prior for transition probabilities. By default, the expected
                               duration is computed and corresponding a and b values are assigned. The ex-
                               pected duration is the sample period divided by the number of states.

b b is the shape2 beta prior for transition probabilities. By default, the expected
                               duration is computed and corresponding a and b values are assigned. The ex-
                               pected duration is the sample period divided by the number of states.

marginal.likelihood
                               How should the marginal likelihood be calculated? Options are: none in which
                               case the marginal likelihood will not be calculated, and Chib95 in which case
                               the method of Chib (1995) is used.

gamma.fixed   1 if users want to constrain γ values to be constant. By default, γ values are
                               allowed to vary across regimes.

...   further arguments to be passed

Details

MCMCoprobitChange simulates from the posterior distribution of an ordinal probit regression model
with multiple parameter breaks. The simulation of latent states is based on the linear approximation
method discussed in Park (2011).

The model takes the following form:

$$\Pr(y_t = 1) = \Phi(\gamma_{c,m} - x_i'\beta_m) - \Phi(\gamma_{c-1,m} - x_i'\beta_m) \ \ m = 1, \ldots, M$$

Where M is the number of states, and $\gamma_{c,m}$ and β_m are paramters when a state is m at t.

We assume Gaussian distribution for prior of β:

$$\beta_m \sim \mathcal{N}(b_0, B_0^{-1}), \ \ m = 1, \ldots, M$$

And:

$$p_{mm} \sim \mathcal{B}eta(a, b), \ \ m = 1, \ldots, M$$

Where M is the number of states.

Note that when the fitted changepoint model has very few observations in any of states, the marginal
likelihood outcome can be "nan," which indicates that too many breaks are assumed given the model
and data.

Value

An mcmc object that contains the posterior sample. This object can be summarized by functions
provided by the coda package. The object contains an attribute prob.state storage matrix that
contains the probability of $state_i$ for each period, the log-likelihood of the model (loglike), and
the log-marginal likelihood of the model (logmarglike).

References

Jong Hee Park. 2011. "Changepoint Analysis of Binary and Ordinal Probit Models: An Application to Bank Rate Policy Under the Interwar Gold Standard." *Political Analysis*. 19: 188-204. <doi:10.1093/pan/mpr007>

Andrew D. Martin, Kevin M. Quinn, and Jong Hee Park. 2011. "MCMCpack: Markov Chain Monte Carlo in R.", *Journal of Statistical Software*. 42(9): 1-21. doi: 10.18637/jss.v042.i09.

Siddhartha Chib. 1998. "Estimation and comparison of multiple change-point models." *Journal of Econometrics*. 86: 221-241.

See Also

plotState, plotChangepoint

Examples

```
set.seed(1909)
N <- 200
x1 <- rnorm(N, 1, .5);

## set a true break at 100
z1 <- 1 + x1[1:100] + rnorm(100);
z2 <- 1 -0.2*x1[101:200] + rnorm(100);
z <- c(z1,  z2);
y <- z

## generate y
y[z < 1] <- 1;
y[z >= 1 & z < 2] <- 2;
y[z >= 2] <- 3;

## inputs
formula <- y ~ x1

## fit multiple models with a varying number of breaks
out1 <- MCMCoprobitChange(formula, m=1,
      mcmc=100, burnin=100, thin=1, tune=c(.5, .5), verbose=100,
     b0=0, B0=0.1, marginal.likelihood = "Chib95")
out2 <- MCMCoprobitChange(formula, m=2,
      mcmc=100, burnin=100, thin=1, tune=c(.5, .5, .5), verbose=100,
     b0=0, B0=0.1, marginal.likelihood = "Chib95")

## Do model comparison
## NOTE: the chain should be run longer than this example!
BayesFactor(out1, out2)

## draw plots using the "right" model
plotState(out1)
plotChangepoint(out1)
```

MCMCordfactanal   *Markov Chain Monte Carlo for Ordinal Data Factor Analysis Model*

Description

This function generates a sample from the posterior distribution of an ordinal data factor analysis model. Normal priors are assumed on the factor loadings and factor scores while improper uniform priors are assumed on the cutpoints. The user supplies data and parameters for the prior distributions, and a sample from the posterior distribution is returned as an mcmc object, which can be subsequently analyzed with functions provided in the coda package.

Usage

```
MCMCordfactanal(
  x,
  factors,
  lambda.constraints = list(),
  data = parent.frame(),
  burnin = 1000,
  mcmc = 20000,
  thin = 1,
  tune = NA,
  verbose = 0,
  seed = NA,
  lambda.start = NA,
  l0 = 0,
  L0 = 0,
  store.lambda = TRUE,
  store.scores = FALSE,
  drop.constantvars = TRUE,
  ...
)
```

Arguments

x   Either a formula or a numeric matrix containing the manifest variables.

factors   The number of factors to be fitted.

lambda.constraints

List of lists specifying possible equality or simple inequality constraints on the factor loadings. A typical entry in the list has one of three forms: varname=list(d,c) which will constrain the dth loading for the variable named varname to be equal to c, varname=list(d,"+") which will constrain the dth loading for the variable named varname to be positive, and varname=list(d, "-") which will constrain the dth loading for the variable named varname to be negative. If x is a matrix without column names defaults names of "V1", "V2", ... , etc will be used. Note that, unlike MCMCfactanal, the Λ matrix used here has factors+1

columns. The first column of Λ corresponds to negative item difficulty parameters and should generally not be constrained.

data	A data frame.
burnin	The number of burn-in iterations for the sampler.
mcmc	The number of iterations for the sampler.
thin	The thinning interval used in the simulation. The number of iterations must be divisible by this value.
tune	The tuning parameter for the Metropolis-Hastings sampling. Can be either a scalar or a k-vector. Must be strictly positive.
verbose	A switch which determines whether or not the progress of the sampler is printed to the screen. If verbose is greater than 0 the iteration number and the Metropolis-Hastings acceptance rate are printed to the screen every verboseth iteration.
seed	The seed for the random number generator. If NA, the Mersenne Twister generator is used with default seed 12345; if an integer is passed it is used to seed the Mersenne twister. The user can also pass a list of length two to use the L'Ecuyer random number generator, which is suitable for parallel computation. The first element of the list is the L'Ecuyer seed, which is a vector of length six or NA (if NA a default seed of rep(12345,6) is used). The second element of list is a positive substream number. See the MCMCpack specification for more details.
lambda.start	Starting values for the factor loading matrix Lambda. If lambda.start is set to a scalar the starting value for all unconstrained loadings will be set to that scalar. If lambda.start is a matrix of the same dimensions as Lambda then the lambda.start matrix is used as the starting values (except for equality-constrained elements). If lambda.start is set to NA (the default) then starting values for unconstrained elements in the first column of Lambda are based on the observed response pattern, the remaining unconstrained elements of Lambda are set to , and starting values for inequality constrained elements are set to either 1.0 or -1.0 depending on the nature of the constraints.
l0	The means of the independent Normal prior on the factor loadings. Can be either a scalar or a matrix with the same dimensions as Lambda.
L0	The precisions (inverse variances) of the independent Normal prior on the factor loadings. Can be either a scalar or a matrix with the same dimensions as Lambda.
store.lambda	A switch that determines whether or not to store the factor loadings for posterior analysis. By default, the factor loadings are all stored.
store.scores	A switch that determines whether or not to store the factor scores for posterior analysis. *NOTE: This takes an enormous amount of memory, so should only be used if the chain is thinned heavily, or for applications with a small number of observations.* By default, the factor scores are not stored.
drop.constantvars	
	A switch that determines whether or not manifest variables that have no variation should be deleted before fitting the model. Default = TRUE.
...	further arguments to be passed

Details

The model takes the following form:

Let $i = 1, \ldots, N$ index observations and $j = 1, \ldots, K$ index response variables within an observation. The typical observed variable x_{ij} is ordinal with a total of C_j categories. The distribution of X is governed by a $N \times K$ matrix of latent variables X^* and a series of cutpoints γ. X^* is assumed to be generated according to:

$$x_i^* = \Lambda \phi_i + \epsilon_i$$

$$\epsilon_i \sim \mathcal{N}(0, I)$$

where x_i^* is the k-vector of latent variables specific to observation i, Λ is the $k \times d$ matrix of factor loadings, and ϕ_i is the d-vector of latent factor scores. It is assumed that the first element of ϕ_i is equal to 1 for all i.

The probability that the jth variable in observation i takes the value c is:

$$\pi_{ijc} = \Phi(\gamma_{jc} - \Lambda_j' \phi_i) - \Phi(\gamma_{j(c-1)} - \Lambda_j' \phi_i)$$

The implementation used here assumes independent conjugate priors for each element of Λ and each ϕ_i. More specifically we assume:

$$\Lambda_{ij} \sim \mathcal{N}(l_{0_{ij}}, L_{0_{ij}}^{-1}), i = 1, \ldots, k, j = 1, \ldots, d$$

$$\phi_{i(2:d)} \sim \mathcal{N}(0, I), i = 1, \ldots, n$$

The standard two-parameter item response theory model with probit link is a special case of the model sketched above.

`MCMCordfactanal` simulates from the posterior distribution using a Metropolis-Hastings within Gibbs sampling algorithm. The algorithm employed is based on work by Cowles (1996). Note that the first element of ϕ_i is a 1. As a result, the first column of Λ can be interpreted as item difficulty parameters. Further, the first element γ_1 is normalized to zero, and thus not returned in the mcmc object. The simulation proper is done in compiled C++ code to maximize efficiency. Please consult the coda documentation for a comprehensive list of functions that can be used to analyze the posterior sample.

As is the case with all measurement models, make sure that you have plenty of free memory, especially when storing the scores.

Value

An mcmc object that contains the posterior sample. This object can be summarized by functions provided by the coda package.

References

Shawn Treier and Simon Jackman. 2008. "Democracy as a Latent Variable." *American Journal of Political Science*. 52: 201-217.

Andrew D. Martin, Kevin M. Quinn, and Jong Hee Park. 2011. "MCMCpack: Markov Chain Monte Carlo in R.", *Journal of Statistical Software*. 42(9): 1-21. doi: 10.18637/jss.v042.i09.

M. K. Cowles. 1996. "Accelerating Monte Carlo Markov Chain Convergence for Cumulative-link Generalized Linear Models." *Statistics and Computing.* 6: 101-110.

Valen E. Johnson and James H. Albert. 1999. "Ordinal Data Modeling." Springer: New York.

Daniel Pemstein, Kevin M. Quinn, and Andrew D. Martin. 2007. *Scythe Statistical Library 1.0.* http://scythe.lsa.umich.edu.

Martyn Plummer, Nicky Best, Kate Cowles, and Karen Vines. 2006. "Output Analysis and Diagnostics for MCMC (CODA)", *R News*. 6(1): 7-11. https://CRAN.R-project.org/doc/Rnews/Rnews_2006-1.pdf.

See Also

plot.mcmc, summary.mcmc, factanal, MCMCfactanal, MCMCirt1d, MCMCirtKd

Examples

```
## Not run:
data(painters)
new.painters <- painters[,1:4]
cuts <- apply(new.painters, 2, quantile, c(.25, .50, .75))
for (i in 1:4){
   new.painters[new.painters[,i]<cuts[1,i],i] <- 100
  new.painters[new.painters[,i]<cuts[2,i],i] <- 200
  new.painters[new.painters[,i]<cuts[3,i],i] <- 300
  new.painters[new.painters[,i]<100,i] <- 400
}

posterior <- MCMCordfactanal(~Composition+Drawing+Colour+Expression,
                     data=new.painters, factors=1,
                     lambda.constraints=list(Drawing=list(2,"+")),
                     burnin=5000, mcmc=500000, thin=200, verbose=500,
                     L0=0.5, store.lambda=TRUE,
                     store.scores=TRUE, tune=1.2)
plot(posterior)
summary(posterior)

## End(Not run)
```

Description

This function generates a sample from the posterior distribution of a model for pairwise comparisons data with a probit link. Thurstone's model is a special case of this model when the α parameter is fixed at 1.

Usage

```
MCMCpaircompare(
  pwc.data,
  theta.constraints = list(),
  alpha.fixed = FALSE,
  burnin = 1000,
  mcmc = 20000,
  thin = 1,
  verbose = 0,
  seed = NA,
  alpha.start = NA,
  a = 0,
  A = 0.25,
  store.theta = TRUE,
  store.alpha = FALSE,
  ...
)
```

Arguments

pwc.data   A data.frame containing the pairwise comparisons data. Each row of `pwc.data`
                   corresponds to a single pairwise comparison. `pwc.data` needs to have exactly
                   four columns. The first column contains a unique identifier for the rater. Column
                   two contains the unique identifier for the first item being compared. Column
                   three contains the unique identifier for the second item being compared. Column
                   four contains the unique identifier of the item selected from the two items being
                   compared. If a tie occurred, the entry in the fourth column should be NA. For
                   applications without raters (such as sports competitions) all entries in the first
                   column should be set to a single value and `alpha.fixed` (see below) should be
                   set to TRUE. **The identifiers in columns 2 through 4 must start with a letter.
                   Examples are provided below.**

theta.constraints

                   A list specifying possible simple equality or inequality constraints on the item
                   parameters. A typical entry in the list has one of three forms: `itemname=c`
                   which will constrain the item parameter for the item named `itemname` to be
                   equal to c, `itemname="+"` which will constrain the item parameter for the item

named `itemname` to be positive, and `itemname="-"` which will constrain the item parameter for the item named `itemname` to be negative.

alpha.fixed	Should alpha be fixed to a constant value of 1 for all raters? Default is FALSE. If set to FALSE, an alpha value is estimated for each rater.
burnin	The number of burn-in iterations for the sampler.
mcmc	The number of Gibbs iterations for the sampler.
thin	The thinning interval used in the simulation. The number of Gibbs iterations must be divisible by this value.
verbose	A switch which determines whether or not the progress of the sampler is printed to the screen. If `verbose` is greater than 0 output is printed to the screen every `verbose`th iteration.
seed	The seed for the random number generator. If NA, the Mersenne Twister generator is used with default seed 12345; if an integer is passed it is used to seed the Mersenne twister. The user can also pass a list of length two to use the L'Ecuyer random number generator, which is suitable for parallel computation. The first element of the list is the L'Ecuyer seed, which is a vector of length six or NA (if NA a default seed of `rep(12345,6)` is used). The second element of list is a positive substream number. See the MCMCpack specification for more details.
alpha.start	The starting value for the alpha vector. This can either be a scalar or a column vector with dimension equal to the number of alphas. If this takes a scalar value, then that value will serve as the starting value for all of the alphas. The default value of NA will set the starting value of each alpha parameter to 1.
a	The prior mean of alpha. Must be a scalar. Default is 0.
A	The prior precision of alpha. Must be a positive scalar. Default is 0.25 (prior variance is 4).
store.theta	Should the theta draws be returned? Default is TRUE.
store.alpha	Should the alpha draws be returned? Default is FALSE.
...	further arguments to be passed

Details

`MCMCpaircompare` uses the data augmentation approach of Albert and Chib (1993). The user supplies data and priors, and a sample from the posterior is returned as an `mcmc` object, which can be subsequently analyzed in the `coda` package.

The simulation is done in compiled C++ code to maximize efficiency.

Please consult the `coda` package documentation for a comprehensive list of functions that can be used to analyze the posterior sample.

The model takes the following form:

$$i = 1, ..., I \quad (raters)$$

$$j = 1, ..., J \quad (items)$$

$$Y_{ijj'} = 1 \ if \ i \ chooses \ j \ over \ j'$$

$$Y_{ijj'} = 0 \ if \ i \ chooses \ j' \ over \ j$$

$$Y_{ijj'} = NA \ if \ i \ chooses \ neither$$

$$Pr(Y_{ijj'} = 1) = \Phi(\alpha_i[\theta_j - \theta_{j'}])$$

The following Gaussian priors are assumed:

$$\alpha_i \sim \mathcal{N}(a, A^{-1})$$

$$\theta_j \sim \mathcal{N}(0, 1)$$

For identification, some θ_js are truncated above or below 0, or fixed to constants.

Value

An mcmc object that contains the posterior sample. This object can be summarized by functions provided by the coda package.

References

Albert, J. H. and S. Chib. 1993. "Bayesian Analysis of Binary and Polychotomous Response Data." *J. Amer. Statist. Assoc.* 88, 669-679

Yu, Qiushi and Kevin M. Quinn. 2021. "A Multidimensional Pairwise Comparison Model for Heterogeneous Perception with an Application to Modeling the Perceived Truthfulness of Public Statements on COVID-19." University of Michigan Working Paper.

Andrew D. Martin, Kevin M. Quinn, and Jong Hee Park. 2011. "MCMCpack: Markov Chain Monte Carlo in R.", *Journal of Statistical Software*. 42(9): 1-21. doi: 10.18637/jss.v042.i09.

Daniel Pemstein, Kevin M. Quinn, and Andrew D. Martin. 2007. *Scythe Statistical Library 1.0.* http://scythe.lsa.umich.edu.

Martyn Plummer, Nicky Best, Kate Cowles, and Karen Vines. 2006. "Output Analysis and Diagnostics for MCMC (CODA)", *R News*. 6(1): 7-11. https://CRAN.R-project.org/doc/Rnews/Rnews_2006-1.pdf.

See Also

plot.mcmc,summary.mcmc, MCMCpaircompare2d, MCMCpaircompare2dDP

Examples

```
  ## Not run:
  ## Euro 2016 example
  data(Euro2016)

posterior1 <- MCMCpaircompare(pwc.data=Euro2016,
                              theta.constraints=list(Ukraine="-",
                                                     Portugal="+"),
                              alpha.fixed=TRUE,
                              verbose=10000,
```

```
                                    burnin=10000, mcmc=500000, thin=100,
                                    store.theta=TRUE, store.alpha=FALSE)

## alternative identification constraints
posterior2 <- MCMCpaircompare(pwc.data=Euro2016,
                              theta.constraints=list(Ukraine="-",
                                                      Portugal=1),
                              alpha.fixed=TRUE,
                              verbose=10000,
                              burnin=10000, mcmc=500000, thin=100,
                              store.theta=TRUE, store.alpha=FALSE)




## a synthetic data example with estimated rater-specific parameters
set.seed(123)

I <- 65  ## number of raters
J <- 50 ## number of items to be compared


## raters 1 to 5 have less sensitivity to stimuli than raters 6 through I
alpha.true <- c(rnorm(5, m=0.2, s=0.05), rnorm(I - 5, m=1, s=0.1))
theta.true <- sort(rnorm(J, m=0, s=1))

n.comparisons <- 125 ## number of pairwise comparisons for each rater

## generate synthetic data according to the assumed model
rater.id <- NULL
item.1.id <- NULL
item.2.id <- NULL
choice.id <- NULL
for (i in 1:I){
    for (c in 1:n.comparisons){
        rater.id <- c(rater.id, i+100)
        item.numbers <- sample(1:J, size=2, replace=FALSE)
        item.1 <- item.numbers[1]
        item.2 <- item.numbers[2]
        item.1.id <- c(item.1.id, item.1)
        item.2.id <- c(item.2.id, item.2)
        eta <- alpha.true[i] * (theta.true[item.1] - theta.true[item.2])
        prob.item.1.chosen <- pnorm(eta)
        u <- runif(1)
        if (u <= prob.item.1.chosen){
            choice.id <- c(choice.id, item.1)
        }
        else{
            choice.id <- c(choice.id, item.2)
```

```
        }
    }
}
item.1.id <- paste("item", item.1.id+100, sep=".")
item.2.id <- paste("item", item.2.id+100, sep=".")
choice.id <- paste("item", choice.id+100, sep=".")

sim.data <- data.frame(rater.id, item.1.id, item.2.id, choice.id)


## fit the model
posterior <- MCMCpaircompare(pwc.data=sim.data,
                             theta.constraints=list(item.101=-2,
                                                    item.150=2),
                             alpha.fixed=FALSE,
                             verbose=10000,
                             a=0, A=0.5,
                             burnin=10000, mcmc=200000, thin=100,
                             store.theta=TRUE, store.alpha=TRUE)

theta.draws <- posterior[, grep("theta", colnames(posterior))]
alpha.draws <- posterior[, grep("alpha", colnames(posterior))]

theta.post.med <- apply(theta.draws, 2, median)
alpha.post.med <- apply(alpha.draws, 2, median)

theta.post.025 <- apply(theta.draws, 2, quantile, prob=0.025)
theta.post.975 <- apply(theta.draws, 2, quantile, prob=0.975)
alpha.post.025 <- apply(alpha.draws, 2, quantile, prob=0.025)
alpha.post.975 <- apply(alpha.draws, 2, quantile, prob=0.975)

## compare estimates to truth
par(mfrow=c(1,2))
plot(theta.true, theta.post.med, xlim=c(-2.5, 2.5), ylim=c(-2.5, 2.5),
     col=rgb(0,0,0,0.3))
segments(x0=theta.true, x1=theta.true,
         y0=theta.post.025, y1=theta.post.975,
         col=rgb(0,0,0,0.3))
abline(0, 1, col=rgb(1,0,0,0.5))

plot(alpha.true, alpha.post.med, xlim=c(0, 1.2), ylim=c(0, 3),
     col=rgb(0,0,0,0.3))
segments(x0=alpha.true, x1=alpha.true,
         y0=alpha.post.025, y1=alpha.post.975,
         col=rgb(0,0,0,0.3))
abline(0, 1, col=rgb(1,0,0,0.5))


## End(Not run)
```

MCMCpaircompare2d	*Markov Chain Monte Carlo for the Two-Dimensional Pairwise Comparisons Model in Yu and Quinn (2021)*

Description

This function generates a sample from the posterior distribution of a model for pairwise comparisons data with a probit link. Unlike standard models for pairwise comparisons data, in this model the latent attribute of each item being compared is a vector in two-dimensional Euclidean space.

Usage

```
MCMCpaircompare2d(
  pwc.data,
  theta.constraints = list(),
  burnin = 1000,
  mcmc = 20000,
  thin = 1,
  verbose = 0,
  seed = NA,
  gamma.start = NA,
  theta.start = NA,
  store.theta = TRUE,
  store.gamma = TRUE,
  tune = 0.3,
  procrustes = FALSE,
  ...
)
```

Arguments

pwc.data
: A data.frame containing the pairwise comparisons data. Each row of `pwc.data` corresponds to a single pairwise comparison. `pwc.data` needs to have exactly four columns. The first column contains a unique identifier for the rater. Column two contains the unique identifier for the first item being compared. Column three contains the unique identifier for the second item being compared. Column four contains the unique identifier of the item selected from the two items being compared. If a tie occurred, the entry in the fourth column should be NA. **The identifiers in columns 2 through 4 must start with a letter. Examples are provided below.**

theta.constraints
: A list specifying possible simple equality or inequality constraints on the item parameters. A typical entry in the list has one of three forms: `itemname=list(d,c)` which will constrain the dth dimension of theta for the item named `itemname` to be equal to c, `itemname=list(d,"+")` which will constrain the dth dimension of theta for the item named `itemname` to be positive, and `itemname=list(d,`

"-") which will constrain the dth dimension of theta for the item named `itemname` to be negative.

burnin   The number of burn-in iterations for the sampler.

mcmc   The number of Gibbs iterations for the sampler.

thin   The thinning interval used in the simulation. The number of Gibbs iterations must be divisible by this value.

verbose   A switch which determines whether or not the progress of the sampler is printed to the screen. If verbose is greater than 0 output is printed to the screen every `verbose`th iteration.

seed   The seed for the random number generator. If NA, the Mersenne Twister generator is used with default seed 12345; if an integer is passed it is used to seed the Mersenne twister. The user can also pass a list of length two to use the L'Ecuyer random number generator, which is suitable for parallel computation. The first element of the list is the L'Ecuyer seed, which is a vector of length six or NA (if NA a default seed of rep(12345,6) is used). The second element of list is a positive substream number. See the MCMCpack specification for more details.

gamma.start   The starting value for the gamma vector. This can either be a scalar or a column vector with dimension equal to the number of raters. If this takes a scalar value, then that value will serve as the starting value for all of the gammas. The default value of NA will set the starting value of each gamma parameter to $\pi/4$.

theta.start   Starting values for the theta. Can be either a numeric scalar, a J by 2 matrix (where J is the number of items compared), or NA. If a scalar, all theta values are set to that value (except elements already specified via theta.contraints. If NA, then non constrained elements of theta are set equal to 0, elements constrained to be positive are set equal to 0.5, elements constrained to be negative are set equal to -0.5 and elements with equality constraints are set to satisfy those constraints.

store.theta   Should the theta draws be returned? Default is TRUE.

store.gamma   Should the gamma draws be returned? Default is TRUE.

tune   Tuning parameter for the random walk Metropolis proposal for each gamma_i. tune is the width of the uniform proposal centered at the current value of gamma_i. Must be a positive scalar.

procrustes   Should the theta and gamma draws be post-processed with a Procrustes transformation? Default is FALSE. The Procrustes target matrix is derived from the constrained elements of theta. Each row of theta that has both theta values constrained is part of the of the target matrix. Elements with equality constraints are set to those values. Elements constrained to be positive are set to 1. Elements constrained to be negative are set to -1. If procrustes is set to TRUE theta.constraints must be set so that there are at least three rows of theta that have both elements of theta constrained.

...   further arguments to be passed

Details

MCMCpaircompare2d uses the data augmentation approach of Albert and Chib (1993) in conjunction with Gibbs and Metropolis-within-Gibbs steps to fit the model. The user supplies data and a sample

from the posterior is returned as an `mcmc` object, which can be subsequently analyzed in the coda package.

The simulation is done in compiled C++ code to maximize efficiency.

Please consult the `coda` package documentation for a comprehensive list of functions that can be used to analyze the posterior sample.

The model takes the following form:

$$i = 1, ..., I \quad (raters)$$

$$j = 1, ..., J \quad (items)$$

$$Y_{ijj'} = 1 \ if \ i \ chooses \ j \ over \ j'$$

$$Y_{ijj'} = 0 \ if \ i \ chooses \ j' \ over \ j$$

$$Y_{ijj'} = NA \ if \ i \ chooses \ neither$$

$$\Pr(Y_{ijj'} = 1) = \Phi(\mathbf{z}_i'[\boldsymbol{\theta}_j - \boldsymbol{\theta}_{j'}])$$

$$\mathbf{z}_i = [\cos(\gamma_i), \ \sin(\gamma_i)]'$$

The following priors are assumed:

$$\gamma_i \sim \mathcal{U}nif(0, \ \pi/2)$$

$$\boldsymbol{\theta}_j \sim \mathcal{N}_2(\mathbf{0}, \mathbf{I}_2)$$

For identification, some $\boldsymbol{\theta}_j$s are truncated above or below 0, or fixed to constants.

Value

An mcmc object that contains the posterior sample. This object can be summarized by functions provided by the coda package.

Author(s)

Qiushi Yu <yuqiushi@umich.edu> and Kevin M. Quinn <kmq@umich.edu>

References

Albert, J. H. and S. Chib. 1993. "Bayesian Analysis of Binary and Polychotomous Response Data." *J. Amer. Statist. Assoc.* 88, 669-679

Yu, Qiushi and Kevin M. Quinn. 2021. "A Multidimensional Pairwise Comparison Model for Heterogeneous Perceptions with an Application to Modeling the Perceived Truthfulness of Public Statements on COVID-19." University of Michigan Working Paper.

Andrew D. Martin, Kevin M. Quinn, and Jong Hee Park. 2011. "MCMCpack: Markov Chain Monte Carlo in R.", *Journal of Statistical Software*. 42(9): 1-21. doi: 10.18637/jss.v042.i09.

Daniel Pemstein, Kevin M. Quinn, and Andrew D. Martin. 2007. *Scythe Statistical Library 1.0.* http://scythe.lsa.umich.edu.

Martyn Plummer, Nicky Best, Kate Cowles, and Karen Vines. 2006. "Output Analysis and Diagnostics for MCMC (CODA)", *R News*. 6(1): 7-11. https://CRAN.R-project.org/doc/Rnews/Rnews_2006-1.pdf.

See Also

plot.mcmc,summary.mcmc, MCMCpaircompare, MCMCpaircompare2dDP

Examples

```
  ## Not run:
## a synthetic data example
set.seed(123)

I <- 65  ## number of raters
J <- 50 ## number of items to be compared


## raters 1 to 5 put most weight on dimension 1
## raters 6 to 10 put most weight on dimension 2
## raters 11 to I put substantial weight on both dimensions
gamma.true <- c(runif(5, 0, 0.1),
             runif(5, 1.47, 1.57),
             runif(I-10, 0.58, 0.98) )
theta1.true <- rnorm(J, m=0, s=1)
theta2.true <- rnorm(J, m=0, s=1)
theta1.true[1] <- 2
theta2.true[1] <- 2
theta1.true[2] <- -2
theta2.true[2] <- -2
theta1.true[3] <-  2
theta2.true[3] <- -2



n.comparisons <- 125 ## number of pairwise comparisons for each rater

## generate synthetic data according to the assumed model
rater.id <- NULL
item.1.id <- NULL
item.2.id <- NULL
choice.id <- NULL
for (i in 1:I){
    for (c in 1:n.comparisons){
        rater.id <- c(rater.id, i+100)
        item.numbers <- sample(1:J, size=2, replace=FALSE)
        item.1 <- item.numbers[1]
        item.2 <- item.numbers[2]
        item.1.id <- c(item.1.id, item.1)
        item.2.id <- c(item.2.id, item.2)
        z <- c(cos(gamma.true[i]), sin(gamma.true[i]))
        eta <- z[1] * (theta1.true[item.1] - theta1.true[item.2])  +
            z[2] * (theta2.true[item.1] - theta2.true[item.2])
        prob.item.1.chosen <- pnorm(eta)
        u <- runif(1)
        if (u <= prob.item.1.chosen){
```

```
              choice.id <- c(choice.id, item.1)
          }
          else{
              choice.id <- c(choice.id, item.2)
          }
      }
  }
item.1.id <- paste("item", item.1.id+100, sep=".")
item.2.id <- paste("item", item.2.id+100, sep=".")
choice.id <- paste("item", choice.id+100, sep=".")

sim.data <- data.frame(rater.id, item.1.id, item.2.id, choice.id)


## fit the model
posterior <- MCMCpaircompare2d(pwc.data=sim.data,
                             theta.constraints=list(item.101=list(1,2),
                                                    item.101=list(2,2),
                                                    item.102=list(1,-2),
                                                    item.102=list(2,-2),
                                                    item.103=list(1,"+"),
                                                    item.103=list(2,"-")),
                             verbose=1000,
                             burnin=500, mcmc=20000, thin=10,
                             store.theta=TRUE, store.gamma=TRUE, tune=0.5)




theta1.draws <- posterior[, grep("theta1", colnames(posterior))]
theta2.draws <- posterior[, grep("theta2", colnames(posterior))]
gamma.draws <- posterior[, grep("gamma", colnames(posterior))]

theta1.post.med <- apply(theta1.draws, 2, median)
theta2.post.med <- apply(theta2.draws, 2, median)
gamma.post.med <- apply(gamma.draws, 2, median)

theta1.post.025 <- apply(theta1.draws, 2, quantile, prob=0.025)
theta1.post.975 <- apply(theta1.draws, 2, quantile, prob=0.975)
theta2.post.025 <- apply(theta2.draws, 2, quantile, prob=0.025)
theta2.post.975 <- apply(theta2.draws, 2, quantile, prob=0.975)
gamma.post.025 <- apply(gamma.draws, 2, quantile, prob=0.025)
gamma.post.975 <- apply(gamma.draws, 2, quantile, prob=0.975)



## compare estimates to truth
par(mfrow=c(2,2))
plot(theta1.true, theta1.post.med, xlim=c(-2.5, 2.5), ylim=c(-2.5, 2.5),
     col=rgb(0,0,0,0.3))
segments(x0=theta1.true, x1=theta1.true,
         y0=theta1.post.025, y1=theta1.post.975,
```

```
        col=rgb(0,0,0,0.3))
abline(0, 1, col=rgb(1,0,0,0.5))

plot(theta2.true, theta2.post.med, xlim=c(-2.5, 2.5), ylim=c(-2.5, 2.5),
     col=rgb(0,0,0,0.3))
segments(x0=theta2.true, x1=theta2.true,
         y0=theta2.post.025, y1=theta2.post.975,
         col=rgb(0,0,0,0.3))
abline(0, 1, col=rgb(1,0,0,0.5))

plot(gamma.true, gamma.post.med, xlim=c(0, 1.6), ylim=c(0, 1.6),
     col=rgb(0,0,0,0.3))
segments(x0=gamma.true, x1=gamma.true,
         y0=gamma.post.025, y1=gamma.post.975,
         col=rgb(0,0,0,0.3))
abline(0, 1, col=rgb(1,0,0,0.5))


## plot point estimates
plot(theta1.post.med, theta2.post.med,
     xlim=c(-2.5, 2.5), ylim=c(-2.5, 2.5),
     col=rgb(0,0,0,0.3))
for (i in 1:length(gamma.post.med)){
    arrows(x0=0, y0=0,
           x1=cos(gamma.post.med[i]),
           y1=sin(gamma.post.med[i]),
           col=rgb(1,0,0,0.2), len=0.05, lwd=0.5)
}

## End(Not run)
```

MCMCpaircompare2dDP   *Markov Chain Monte Carlo for the Two-Dimensional Pairwise Comparisons Model with Dirichlet Process Prior in Yu and Quinn (2021)*

Description

This function generates a sample from the posterior distribution of a model for pairwise comparisons data with a probit link. Unlike standard models for pairwise comparisons data, in this model the latent attribute of each item being compared is a vector in two-dimensional Euclidean space.

Usage

```
MCMCpaircompare2dDP(
  pwc.data,
  theta.constraints = list(),
  burnin = 1000,
  mcmc = 20000,
  thin = 1,
```

```
    verbose = 0,
    seed = NA,
    gamma.start = NA,
    theta.start = NA,
    store.theta = TRUE,
    store.gamma = FALSE,
    tune = 0.3,
    procrustes = FALSE,
    alpha.start = 1,
    cluster.max = 100,
    cluster.mcmc = 500,
    alpha.fixed = TRUE,
    a0 = 1,
    b0 = 1,
    ...
)
```

Arguments

pwc.data   A data.frame containing the pairwise comparisons data. Each row of `pwc.data`
                    corresponds to a single pairwise comparison. `pwc.data` needs to have exactly
                    four columns. The first column contains a unique identifier for the rater. Column
                    two contains the unique identifier for the first item being compared. Column
                    three contains the unique identifier for the second item being compared. Column
                    four contains the unique identifier of the item selected from the two items being
                    compared. If a tie occurred, the entry in the fourth column should be NA. **The
                    identifiers in columns 2 through 4 must start with a letter. Examples are
                    provided below.**

theta.constraints
                    A list specifying possible simple equality or inequality constraints on the item
                    parameters. A typical entry in the list has one of three forms: `itemname=list(d,c)`
                    which will constrain the dth dimension of theta for the item named `itemname` to
                    be equal to c, `itemname=list(d,"+")` which will constrain the dth dimension
                    of theta for the item named `itemname` to be positive, and `itemname=list(d,
                    "-")` which will constrain the dth dimension of theta for the item named `itemname`
                    to be negative.

burnin   The number of burn-in iterations for the sampler.

mcmc   The number of Gibbs iterations for the sampler.

thin   The thinning interval used in the simulation. The number of Gibbs iterations
                    must be divisible by this value.

verbose   A switch which determines whether or not the progress of the sampler is printed
                    to the screen. If verbose is greater than 0 output is printed to the screen every
                    verboseth iteration.

seed   The seed for the random number generator. If NA, the Mersenne Twister gener-
                    ator is used with default seed 12345; if an integer is passed it is used to seed the
                    Mersenne twister. The user can also pass a list of length two to use the L'Ecuyer
                    random number generator, which is suitable for parallel computation. The first

	element of the list is the L'Ecuyer seed, which is a vector of length six or NA (if NA a default seed of rep(12345,6) is used). The second element of list is a positive substream number. See the MCMCpack specification for more details.
gamma.start	The starting value for the gamma vector. This can either be a scalar or a column vector with dimension equal to the number of raters. If this takes a scalar value, then that value will serve as the starting value for all of the gammas. The default value of NA will set the starting value of each gamma parameter to $\pi/4$.
theta.start	Starting values for the theta. Can be either a numeric scalar, a J by 2 matrix (where J is the number of items compared), or NA. If a scalar, all theta values are set to that value (except elements already specified via theta.contraints. If NA, then non constrained elements of theta are set equal to 0, elements constrained to be positive are set equal to 0.5, elements constrained to be negative are set equal to -0.5 and elements with equality constraints are set to satisfy those constraints.
store.theta	Should the theta draws be returned? Default is TRUE.
store.gamma	Should the gamma draws be returned? Default is TRUE.
tune	Tuning parameter for the random walk Metropolis proposal for each gamma_i. tune is the width of the uniform proposal centered at the current value of gamma_i. Must be a positive scalar.
procrustes	Should the theta and gamma draws be post-processed with a Procrustes transformation? Default is FALSE. The Procrustes target matrix is derived from the constrained elements of theta. Each row of theta that has both theta values constrained is part of the of the target matrix. Elements with equality constraints are set to those values. Elements constrained to be positive are set to 1. Elements constrained to be negative are set to -1. If procrustes is set to TRUE theta.constraints must be set so that there are at least three rows of theta that have both elements of theta constrained.
alpha.start	The starting value for the DP concentration parameter alpha. Must be a positive scalar. Defaults to 1. If alpha.fixed is set equal to TRUE, then alpha is held fixed at alpha.start.
cluster.max	The maximum number of clusters allowed in the approximation to the DP prior for gamma. Defaults to 100. Must be a positive integer.
cluster.mcmc	The number of additional MCMC iterations that are done to sample each cluster-specific gamma value within one main MCMC iteration. Must be a positive integer. Defaults to 500. Setting this to a lower value speeds runtime at the cost of (possibly) worse mixing.
alpha.fixed	Logical value indicating whether the DP concentration parameter alpha be held fixed (TRUE) or estimated (FALSE).
a0	The shape parameter of the gamma prior for alpha. This is the same parameterization of the gamma distribution as R's internal rgamma() function. Only relevant if alpha.fixed is set equal to FALSE. Defaults to 1.
b0	The rate parameter of the gamma prior for alpha. This is the same parameterization of the gamma distribution as R's internal rgamma() function. Only relevant if alpha.fixed is set equal to FALSE. Defaults to 1.
...	further arguments to be passed

Details

`MCMCpaircompare2d` uses the data augmentation approach of Albert and Chib (1993) in conjunction with Gibbs and Metropolis-within-Gibbs steps to fit the model. The user supplies data and a sample from the posterior is returned as an `mcmc` object, which can be subsequently analyzed in the coda package.

The simulation is done in compiled C++ code to maximize efficiency.

Please consult the `coda` package documentation for a comprehensive list of functions that can be used to analyze the posterior sample.

The model takes the following form:

$$i = 1, ..., I \quad (raters)$$

$$j = 1, ..., J \quad (items)$$

$$Y_{ijj'} = 1 \ if \ i \ chooses \ j \ over \ j'$$

$$Y_{ijj'} = 0 \ if \ i \ chooses \ j' \ over \ j$$

$$Y_{ijj'} = NA \ if \ i \ chooses \ neither$$

$$\Pr(Y_{ijj'} = 1) = \Phi(\mathbf{z}_i'[\boldsymbol{\theta}_j - \boldsymbol{\theta}_{j'}])$$

$$\mathbf{z}_i = [\cos(\gamma_i), \ \sin(\gamma_i)]'$$

The following priors are assumed:

$$\gamma_i \sim G$$

$$G \sim \mathcal{DP}(\alpha G_0)$$

$$G_0 = \mathcal{U}nif(0, \pi/2)$$

$$\alpha \sim \mathcal{G}amma(a_0, b_0)$$

$$\boldsymbol{\theta}_j \sim \mathcal{N}_2(\mathbf{0}, \mathbf{I}_2)$$

For identification, some $\boldsymbol{\theta}_j$s are truncated above or below 0, or fixed to constants.

Value

An mcmc object that contains the posterior sample. This object can be summarized by functions provided by the coda package. Most of the column names of the mcmc object are self explanatory. Note however that the columns with names of the form "cluster.[raterID]" give the cluster membership of each rater at each stored MCMC iteration. Because of the possibility of label switching, the particular values of these cluster membership variables are not meaningful. What is meaningful is whether two raters share the same cluster membership value at a particular MCMC iteration. This indicates that those two raters were clustered together during that iteration. Finally, note that the "n.clusters" column gives the number of distinct gamma values at each iteration, i.e. the number of clusters at that iteration.

Author(s)

Qiushi Yu <yuqiushi@umich.edu> and Kevin M. Quinn <kmq@umich.edu>

References

Albert, J. H. and S. Chib. 1993. "Bayesian Analysis of Binary and Polychotomous Response Data." *J. Amer. Statist. Assoc.* 88, 669-679

Yu, Qiushi and Kevin M. Quinn. 2021. "A Multidimensional Pairwise Comparison Model for Heterogeneous Perceptions with an Application to Modeling the Perceived Truthfulness of Public Statements on COVID-19." University of Michigan Working Paper.

Andrew D. Martin, Kevin M. Quinn, and Jong Hee Park. 2011. "MCMCpack: Markov Chain Monte Carlo in R.", *Journal of Statistical Software*. 42(9): 1-21. doi: 10.18637/jss.v042.i09.

Daniel Pemstein, Kevin M. Quinn, and Andrew D. Martin. 2007. *Scythe Statistical Library 1.0.* http://scythe.lsa.umich.edu.

Martyn Plummer, Nicky Best, Kate Cowles, and Karen Vines. 2006. "Output Analysis and Diagnostics for MCMC (CODA)", *R News*. 6(1): 7-11. https://CRAN.R-project.org/doc/Rnews/Rnews_2006-1.pdf.

See Also

plot.mcmc,summary.mcmc, MCMCpaircompare, MCMCpaircompare2dDP

Examples

```
## Not run:
## a synthetic data example
set.seed(123)

I <- 65  ## number of raters
J <- 50 ## number of items to be compared

## 3 clusters:
## raters 1 to 5 put most weight on dimension 1
## raters 6 to 10 put most weight on dimension 2
## raters 11 to I put substantial weight on both dimensions
gamma.true <- c(rep(0.05, 5),
            rep(1.50, 5),
            rep(0.7, I-10) )
theta1.true <- rnorm(J, m=0, s=1)
theta2.true <- rnorm(J, m=0, s=1)
theta1.true[1] <- 2
theta2.true[1] <- 2
theta1.true[2] <- -2
theta2.true[2] <- -2
theta1.true[3] <-  2
theta2.true[3] <- -2


n.comparisons <- 125 ## number of pairwise comparisons for each rater

## generate synthetic data according to the assumed model
rater.id <- NULL
item.1.id <- NULL
```

```
item.2.id <- NULL
choice.id <- NULL
for (i in 1:I){
    for (c in 1:n.comparisons){
        rater.id <- c(rater.id, i+100)
        item.numbers <- sample(1:J, size=2, replace=FALSE)
        item.1 <- item.numbers[1]
        item.2 <- item.numbers[2]
        item.1.id <- c(item.1.id, item.1)
        item.2.id <- c(item.2.id, item.2)
        z <- c(cos(gamma.true[i]), sin(gamma.true[i]))
        eta <- z[1] * (theta1.true[item.1] - theta1.true[item.2])  +
            z[2] * (theta2.true[item.1] - theta2.true[item.2])
        prob.item.1.chosen <- pnorm(eta)
        u <- runif(1)
        if (u <= prob.item.1.chosen){
            choice.id <- c(choice.id, item.1)
        }
        else{
            choice.id <- c(choice.id, item.2)
        }
    }
}
item.1.id <- paste("item", item.1.id+100, sep=".")
item.2.id <- paste("item", item.2.id+100, sep=".")
choice.id <- paste("item", choice.id+100, sep=".")

sim.data <- data.frame(rater.id, item.1.id, item.2.id, choice.id)


## fit the model (should be run for more than 10500 iterations)
posterior <- MCMCpaircompare2dDP(pwc.data=sim.data,
                                 theta.constraints=list(item.101=list(1,2),
                                                        item.101=list(2,2),
                                                        item.102=list(1,-2),
                                                        item.102=list(2,-2),
                                                        item.103=list(1,"+"),
                                                        item.103=list(2,"-")),
                                 verbose=100,
                                 burnin=500, mcmc=10000, thin=5,
                                 cluster.mcmc=10,
                                 store.theta=TRUE, store.gamma=TRUE,
                                 tune=0.1)




theta1.draws <- posterior[, grep("theta1", colnames(posterior))]
theta2.draws <- posterior[, grep("theta2", colnames(posterior))]
gamma.draws <- posterior[, grep("gamma", colnames(posterior))]

theta1.post.med <- apply(theta1.draws, 2, median)
```

```
theta2.post.med <- apply(theta2.draws, 2, median)
gamma.post.med <- apply(gamma.draws, 2, median)

theta1.post.025 <- apply(theta1.draws, 2, quantile, prob=0.025)
theta1.post.975 <- apply(theta1.draws, 2, quantile, prob=0.975)
theta2.post.025 <- apply(theta2.draws, 2, quantile, prob=0.025)
theta2.post.975 <- apply(theta2.draws, 2, quantile, prob=0.975)
gamma.post.025 <- apply(gamma.draws, 2, quantile, prob=0.025)
gamma.post.975 <- apply(gamma.draws, 2, quantile, prob=0.975)



## compare estimates to truth
par(mfrow=c(2,2))
plot(theta1.true, theta1.post.med, xlim=c(-2.5, 2.5), ylim=c(-2.5, 2.5),
     col=rgb(0,0,0,0.3))
segments(x0=theta1.true, x1=theta1.true,
         y0=theta1.post.025, y1=theta1.post.975,
         col=rgb(0,0,0,0.3))
abline(0, 1, col=rgb(1,0,0,0.5))

plot(theta2.true, theta2.post.med, xlim=c(-2.5, 2.5), ylim=c(-2.5, 2.5),
     col=rgb(0,0,0,0.3))
segments(x0=theta2.true, x1=theta2.true,
         y0=theta2.post.025, y1=theta2.post.975,
         col=rgb(0,0,0,0.3))
abline(0, 1, col=rgb(1,0,0,0.5))

plot(gamma.true, gamma.post.med, xlim=c(0, 1.6), ylim=c(0, 1.6),
     col=rgb(0,0,0,0.3))
segments(x0=gamma.true, x1=gamma.true,
         y0=gamma.post.025, y1=gamma.post.975,
         col=rgb(0,0,0,0.3))
abline(0, 1, col=rgb(1,0,0,0.5))


## plot point estimates
plot(theta1.post.med, theta2.post.med,
     xlim=c(-2.5, 2.5), ylim=c(-2.5, 2.5),
     col=rgb(0,0,0,0.3))
for (i in 1:length(gamma.post.med)){
    arrows(x0=0, y0=0,
           x1=cos(gamma.post.med[i]),
           y1=sin(gamma.post.med[i]),
           col=rgb(1,0,0,0.2), len=0.05, lwd=0.5)
}


## End(Not run)
```

MCMCpoisson   *Markov Chain Monte Carlo for Poisson Regression*

Description

This function generates a sample from the posterior distribution of a Poisson regression model using a random walk Metropolis algorithm. The user supplies data and priors, and a sample from the posterior distribution is returned as an mcmc object, which can be subsequently analyzed with functions provided in the coda package.

Usage

```
MCMCpoisson(
  formula,
  data = NULL,
  burnin = 1000,
  mcmc = 10000,
  thin = 1,
  tune = 1.1,
  verbose = 0,
  seed = NA,
  beta.start = NA,
  b0 = 0,
  B0 = 0,
  marginal.likelihood = c("none", "Laplace"),
  ...
)
```

Arguments

formula	Model formula.
data	Data frame.
burnin	The number of burn-in iterations for the sampler.
mcmc	The number of Metropolis iterations for the sampler.
thin	The thinning interval used in the simulation. The number of mcmc iterations must be divisible by this value.
tune	Metropolis tuning parameter. Can be either a positive scalar or a k-vector, where k is the length of β.Make sure that the acceptance rate is satisfactory (typically between 0.20 and 0.5) before using the posterior sample for inference.
verbose	A switch which determines whether or not the progress of the sampler is printed to the screen. If verbose is greater than 0 the iteration number, the current beta vector, and the Metropolis acceptance rate are printed to the screen every verboseth iteration.
seed	The seed for the random number generator. If NA, the Mersenne Twister generator is used with default seed 12345; if an integer is passed it is used to seed the Mersenne twister. The user can also pass a list of length two to use the L'Ecuyer random number generator, which is suitable for parallel computation. The first element of the list is the L'Ecuyer seed, which is a vector of length six or NA (if NA a default seed of rep(12345,6) is used). The second element of list is a positive substream number. See the MCMCpack specification for more details.

beta.start   The starting value for the β vector. This can either be a scalar or a column vector
                  with dimension equal to the number of betas. If this takes a scalar value, then
                  that value will serve as the starting value for all of the betas. The default value
                  of NA will use the maximum likelihood estimate of β as the starting value.

b0   The prior mean of β. This can either be a scalar or a column vector with dimen-
                  sion equal to the number of betas. If this takes a scalar value, then that value
                  will serve as the prior mean for all of the betas.

B0   The prior precision of β. This can either be a scalar or a square matrix with
                  dimensions equal to the number of betas. If this takes a scalar value, then that
                  value times an identity matrix serves as the prior precision of β. Default value
                  of 0 is equivalent to an improper uniform prior for beta.

marginal.likelihood
                  How should the marginal likelihood be calculated? Options are: none in which
                  case the marginal likelihood will not be calculated or Laplace in which case the
                  Laplace approximation (see Kass and Raftery, 1995) is used.

...   further arguments to be passed.

Details

MCMCpoisson simulates from the posterior distribution of a Poisson regression model using a ran-
dom walk Metropolis algorithm. The simulation proper is done in compiled C++ code to maximize
efficiency. Please consult the coda documentation for a comprehensive list of functions that can be
used to analyze the posterior sample.

The model takes the following form:

$$y_i \sim \mathcal{P}oisson(\mu_i)$$

Where the inverse link function:

$$\mu_i = \exp(x_i'\beta)$$

We assume a multivariate Normal prior on β:

$$\beta \sim \mathcal{N}(b_0, B_0^{-1})$$

The Metropois proposal distribution is centered at the current value of θ and has variance-covariance
$V = T(B_0 + C^{-1})^{-1}T$ where T is a the diagonal positive definite matrix formed from the tune,
B_0 is the prior precision, and C is the large sample variance-covariance matrix of the MLEs. This
last calculation is done via an initial call to glm.

Value

An mcmc object that contains the posterior sample. This object can be summarized by functions
provided by the coda package.

References

Andrew D. Martin, Kevin M. Quinn, and Jong Hee Park. 2011. "MCMCpack: Markov Chain Monte Carlo in R.", *Journal of Statistical Software*. 42(9): 1-21. doi: 10.18637/jss.v042.i09.

Daniel Pemstein, Kevin M. Quinn, and Andrew D. Martin. 2007. *Scythe Statistical Library 1.0.* http://scythe.lsa.umich.edu.

Martyn Plummer, Nicky Best, Kate Cowles, and Karen Vines. 2006. "Output Analysis and Diagnostics for MCMC (CODA)", *R News*. 6(1): 7-11. https://CRAN.R-project.org/doc/Rnews/Rnews_2006-1.pdf.

See Also

plot.mcmc, summary.mcmc, glm

Examples

```
## Not run:
counts <- c(18,17,15,20,10,20,25,13,12)
outcome <- gl(3,1,9)
treatment <- gl(3,3)
posterior <- MCMCpoisson(counts ~ outcome + treatment)
plot(posterior)
summary(posterior)

## End(Not run)
```

MCMCpoissonChange   *Markov Chain Monte Carlo for a Poisson Regression Changepoint Model*

Description

This function generates a sample from the posterior distribution of a Poisson regression model with multiple changepoints. The function uses the Markov chain Monte Carlo method of Chib (1998). The user supplies data and priors, and a sample from the posterior distribution is returned as an mcmc object, which can be subsequently analyzed with functions provided in the coda package.

Usage

```
MCMCpoissonChange(
  formula,
  data = parent.frame(),
  m = 1,
  b0 = 0,
  B0 = 1,
  a = NULL,
```

```
  b = NULL,
  c0 = NA,
  d0 = NA,
  lambda.mu = NA,
  lambda.var = NA,
  burnin = 1000,
  mcmc = 1000,
  thin = 1,
  verbose = 0,
  seed = NA,
  beta.start = NA,
  P.start = NA,
  marginal.likelihood = c("none", "Chib95"),
  ...
)
```

Arguments

formula	Model formula.
data	Data frame.
m	The number of changepoints.
b0	The prior mean of β. This can either be a scalar or a column vector with dimension equal to the number of betas. If this takes a scalar value, then that value will serve as the prior mean for all of the betas.
B0	The prior precision of β. This can either be a scalar or a square matrix with dimensions equal to the number of betas. If this takes a scalar value, then that value times an identity matrix serves as the prior precision of beta. Default value of 0 is equivalent to an improper uniform prior for beta.
a	a is the shape1 beta prior for transition probabilities. By default, the expected duration is computed and corresponding a and b values are assigned. The expected duration is the sample period divided by the number of states.
b	b is the shape2 beta prior for transition probabilities. By default, the expected duration is computed and corresponding a and b values are assigned. The expected duration is the sample period divided by the number of states.
c0	c_0 is the shape parameter for Gamma prior on λ (the mean). When there is no covariate, this should be provided by users. No default value is provided.
d0	d_0 is the scale parameter for Gamma prior on λ (the mean). When there is no covariate, this should be provided by users. No default value is provided.
lambda.mu	The mean of the Gamma prior on λ. $sigma.mu$ and $sigma.var$ allow users to choose the Gamma prior by choosing its mean and variance.
lambda.var	The variacne of the Gamma prior on λ. $sigma.mu$ and $sigma.var$ allow users to choose the Gamma prior by choosing its mean and variance.
burnin	The number of burn-in iterations for the sampler.
mcmc	The number of MCMC iterations after burn-in.

thin	The thinning interval used in the simulation. The number of MCMC iterations must be divisible by this value.
verbose	A switch which determines whether or not the progress of the sampler is printed to the screen. If verbose is greater than 0, the iteration number and the posterior density samples are printed to the screen every verboseth iteration.
seed	The seed for the random number generator. If NA, current R system seed is used.
beta.start	The starting values for the beta vector. This can either be a scalar or a column vector with dimension equal to the number of betas. The default value of NA will use draws from the Uniform distribution with the same boundary with the data as the starting value. If this is a scalar, that value will serve as the starting value mean for all of the betas. When there is no covariate, the log value of means should be used.
P.start	The starting values for the transition matrix. A user should provide a square matrix with dimension equal to the number of states. By default, draws from the Beta(0.9, 0.1) are used to construct a proper transition matrix for each raw except the last raw.
marginal.likelihood	
	How should the marginal likelihood be calculated? Options are: none in which case the marginal likelihood will not be calculated, and Chib95 in which case the method of Chib (1995) is used.
...	further arguments to be passed

Details

MCMCpoissonChange simulates from the posterior distribution of a Poisson regression model with multiple changepoints using the methods of Chib (1998) and Fruhwirth-Schnatter and Wagner (2006). The details of the model are discussed in Park (2010).

The model takes the following form:

$$y_t \sim \mathcal{P}oisson(\mu_t)$$

$$\mu_t = x_t'\beta_m, \quad m = 1, \ldots, M$$

Where M is the number of states and β_m is paramters when a state is m at t.

We assume Gaussian distribution for prior of β:

$$\beta_m \sim \mathcal{N}(b_0, B_0^{-1}), \quad m = 1, \ldots, M$$

And:

$$p_{mm} \sim \mathcal{B}eta(a, b), \quad m = 1, \ldots, M$$

Where M is the number of states.

Value

An mcmc object that contains the posterior sample. This object can be summarized by functions provided by the coda package. The object contains an attribute prob.state storage matrix that contains the probability of $state_i$ for each period, and the log-marginal likelihood of the model (logmarglike).

References

Jong Hee Park. 2010. "Structural Change in the U.S. Presidents' Use of Force Abroad." *American Journal of Political Science* 54: 766-782. <doi:10.1111/j.1540-5907.2010.00459.x>

Sylvia Fruhwirth-Schnatter and Helga Wagner 2006. "Auxiliary Mixture Sampling for Parameter-driven Models of Time Series of Counts with Applications to State Space Modelling." *Biometrika*. 93:827–841.

Siddhartha Chib. 1998. "Estimation and comparison of multiple change-point models." *Journal of Econometrics*. 86: 221-241. <doi: 10.1016/S0304-4076(97)00115-2>

Andrew D. Martin, Kevin M. Quinn, and Jong Hee Park. 2011. "MCMCpack: Markov Chain Monte Carlo in R.", *Journal of Statistical Software*. 42(9): 1-21. doi: 10.18637/jss.v042.i09.

Siddhartha Chib. 1995. "Marginal Likelihood from the Gibbs Output." *Journal of the American Statistical Association*. 90: 1313-1321. <doi: 10.1080/01621459.1995.10476635>

See Also

MCMCbinaryChange, plotState, plotChangepoint

Examples

```
## Not run:
set.seed(11119)
n <- 150
x1 <- runif(n, 0, 0.5)
true.beta1 <- c(1,  1)
true.beta2 <- c(1, -2)
true.beta3 <- c(1,  2)

## set true two breaks at (50, 100)
true.s <- rep(1:3, each=n/3)
mu1 <- exp(1 + x1[true.s==1]*1)
mu2 <- exp(1 + x1[true.s==2]*-2)
mu3 <- exp(1 + x1[true.s==3]*2)

y <- as.ts(c(rpois(n/3, mu1), rpois(n/3, mu2), rpois(n/3, mu3)))
formula = y ~ x1

## fit multiple models with a varying number of breaks
model0 <-  MCMCpoissonChange(formula, m=0,
        mcmc = 1000, burnin = 1000, verbose = 500,
        b0 = rep(0, 2), B0 = 1/5*diag(2), marginal.likelihood = "Chib95")
model1 <-  MCMCpoissonChange(formula, m=1,
```

```
          mcmc = 1000, burnin = 1000, verbose = 500,
          b0 = rep(0, 2), B0 = 1/5*diag(2), marginal.likelihood = "Chib95")
model2 <- MCMCpoissonChange(formula, m=2,
          mcmc = 1000, burnin = 1000, verbose = 500,
          b0 = rep(0, 2), B0 = 1/5*diag(2), marginal.likelihood = "Chib95")
model3 <- MCMCpoissonChange(formula, m=3,
          mcmc = 1000, burnin = 1000, verbose = 500,
          b0 = rep(0, 2), B0 = 1/5*diag(2), marginal.likelihood = "Chib95")
model4 <- MCMCpoissonChange(formula, m=4,
          mcmc = 1000, burnin = 1000, verbose = 500,
          b0 = rep(0, 2), B0 = 1/5*diag(2), marginal.likelihood = "Chib95")
model5 <- MCMCpoissonChange(formula, m=5,
          mcmc = 1000, burnin = 1000, verbose = 500,
          b0 = rep(0, 2), B0 = 1/5*diag(2), marginal.likelihood = "Chib95")

## find the most reasonable one
print(BayesFactor(model0, model1, model2, model3, model4, model5))

## draw plots using the "right" model
par(mfrow=c(attr(model2, "m") + 1, 1), mai=c(0.4, 0.6, 0.3, 0.05))
plotState(model2, legend.control = c(1, 0.6))
plotChangepoint(model2, verbose = TRUE, ylab="Density", start=1, overlay=TRUE)

## No covariate case
model2.1 <- MCMCpoissonChange(y ~ 1, m = 2, c0 = 2, d0 = 1,
          mcmc = 1000, burnin = 1000, verbose = 500,
          marginal.likelihood = "Chib95")
print(BayesFactor(model2, model2.1))

## End(Not run)
```

MCMCprobit   *Markov Chain Monte Carlo for Probit Regression*

Description

This function generates a sample from the posterior distribution of a probit regression model using
the data augmentation approach of Albert and Chib (1993). The user supplies data and priors, and
a sample from the posterior distribution is returned as an mcmc object, which can be subsequently
analyzed with functions provided in the coda package.

Usage

```
MCMCprobit(
  formula,
  data = NULL,
  burnin = 1000,
  mcmc = 10000,
```

```
        thin = 1,
        verbose = 0,
        seed = NA,
        beta.start = NA,
        b0 = 0,
        B0 = 0,
        bayes.resid = FALSE,
        marginal.likelihood = c("none", "Laplace", "Chib95"),
        ...
    )
```

Arguments

formula	Model formula.
data	Data frame.
burnin	The number of burn-in iterations for the sampler.
mcmc	The number of Gibbs iterations for the sampler.
thin	The thinning interval used in the simulation. The number of Gibbs iterations must be divisible by this value.
verbose	A switch which determines whether or not the progress of the sampler is printed to the screen. If verbose is greater than 0 the iteration number and the betas are printed to the screen every verboseth iteration.
seed	The seed for the random number generator. If NA, the Mersenne Twister generator is used with default seed 12345; if an integer is passed it is used to seed the Mersenne twister. The user can also pass a list of length two to use the L'Ecuyer random number generator, which is suitable for parallel computation. The first element of the list is the L'Ecuyer seed, which is a vector of length six or NA (if NA a default seed of rep(12345,6) is used). The second element of list is a positive substream number. See the MCMCpack specification for more details.
beta.start	The starting value for the β vector. This can either be a scalar or a column vector with dimension equal to the number of betas. If this takes a scalar value, then that value will serve as the starting value for all of the betas. The default value of NA will use the maximum likelihood estimate of β as the starting value.
b0	The prior mean of β. This can either be a scalar or a column vector with dimension equal to the number of betas. If this takes a scalar value, then that value will serve as the prior mean for all of the betas.
B0	The prior precision of β. This can either be a scalar or a square matrix with dimensions equal to the number of betas. If this takes a scalar value, then that value times an identity matrix serves as the prior precision of β. Default value of 0 is equivalent to an improper uniform prior on β.
bayes.resid	Should latent Bayesian residuals (Albert and Chib, 1995) be returned? Default is FALSE meaning no residuals should be returned. Alternatively, the user can specify an array of integers giving the observation numbers for which latent residuals should be calculated and returned. TRUE will return draws of latent residuals for all observations.

marginal.likelihood

How should the marginal likelihood be calculated? Options are: none in which case the marginal likelihood will not be calculated, Laplace in which case the Laplace approximation (see Kass and Raftery, 1995) is used, or Chib95 in which case Chib (1995) method is used.

...   further arguments to be passed

Details

MCMCprobit simulates from the posterior distribution of a probit regression model using data augmentation. The simulation proper is done in compiled C++ code to maximize efficiency. Please consult the coda documentation for a comprehensive list of functions that can be used to analyze the posterior sample.

The model takes the following form:

$$y_i \sim \mathcal{B}ernoulli(\pi_i)$$

Where the inverse link function:

$$\pi_i = \Phi(x_i'\beta)$$

We assume a multivariate Normal prior on β:

$$\beta \sim \mathcal{N}(b_0, B_0^{-1})$$

See Albert and Chib (1993) for estimation details.

Value

An mcmc object that contains the posterior sample. This object can be summarized by functions provided by the coda package.

References

Albert, J. H. and S. Chib. 1993. "Bayesian Analysis of Binary and Polychotomous Response Data." *J. Amer. Statist. Assoc.* 88, 669-679

Albert, J. H. and S. Chib. 1995. "Bayesian Residual Analysis for Binary Response Regression Models." *Biometrika.* 82, 747-759.

Andrew D. Martin, Kevin M. Quinn, and Jong Hee Park. 2011. "MCMCpack: Markov Chain Monte Carlo in R.", *Journal of Statistical Software*. 42(9): 1-21. doi: 10.18637/jss.v042.i09.

Siddhartha Chib. 1995. "Marginal Likelihood from the Gibbs Output." *Journal of the American Statistical Association*. 90: 1313-1321. <doi: 10.1080/01621459.1995.10476635>

Daniel Pemstein, Kevin M. Quinn, and Andrew D. Martin. 2007. *Scythe Statistical Library 1.0*. http://scythe.lsa.umich.edu.

Martyn Plummer, Nicky Best, Kate Cowles, and Karen Vines. 2006. "Output Analysis and Diagnostics for MCMC (CODA)", *R News*. 6(1): 7-11. https://CRAN.R-project.org/doc/Rnews/Rnews_2006-1.pdf.

See Also

[plot.mcmc](),[summary.mcmc](), [glm]()

Examples

```
## Not run:
data(birthwt)
out1 <- MCMCprobit(low~as.factor(race)+smoke, data=birthwt,
 b0 = 0, B0 = 10, marginal.likelihood="Chib95")
out2 <- MCMCprobit(low~age+as.factor(race), data=birthwt,
 b0 = 0, B0 = 10,  marginal.likelihood="Chib95")
out3 <- MCMCprobit(low~age+as.factor(race)+smoke, data=birthwt,
 b0 = 0, B0 = 10,  marginal.likelihood="Chib95")
BayesFactor(out1, out2, out3)
plot(out3)
summary(out3)

## End(Not run)
```

MCMCprobitChange   *Markov Chain Monte Carlo for a linear Gaussian Multiple Change-*
                          point Model

Description

This function generates a sample from the posterior distribution of a linear Gaussian model with
multiple changepoints. The function uses the Markov chain Monte Carlo method of Chib (1998).
The user supplies data and priors, and a sample from the posterior distribution is returned as an
mcmc object, which can be subsequently analyzed with functions provided in the coda package.

Usage

```
MCMCprobitChange(
  formula,
  data = parent.frame(),
  m = 1,
  burnin = 10000,
  mcmc = 10000,
  thin = 1,
  verbose = 0,
  seed = NA,
  beta.start = NA,
  P.start = NA,
  b0 = NULL,
  B0 = NULL,
  a = NULL,
```

```
    b = NULL,
    marginal.likelihood = c("none", "Chib95"),
    ...
)
```

Arguments

formula   Model formula.

data   Data frame.

m   The number of changepoints.

burnin   The number of burn-in iterations for the sampler.

mcmc   The number of MCMC iterations after burnin.

thin   The thinning interval used in the simulation. The number of MCMC iterations must be divisible by this value.

verbose   A switch which determines whether or not the progress of the sampler is printed to the screen. If verbose is greater than 0 the iteration number, the β vector, and the error variance are printed to the screen every verboseth iteration.

seed   The seed for the random number generator. If NA, the Mersenne Twister generator is used with default seed 12345; if an integer is passed it is used to seed the Mersenne twister. The user can also pass a list of length two to use the L'Ecuyer random number generator, which is suitable for parallel computation. The first element of the list is the L'Ecuyer seed, which is a vector of length six or NA (if NA a default seed of rep(12345,6) is used). The second element of list is a positive substream number. See the MCMCpack specification for more details.

beta.start The starting values for the β vector. This can either be a scalar or a column vector with dimension equal to the number of betas. The default value of of NA will use the MLE estimate of β as the starting value. If this is a scalar, that value will serve as the starting value mean for all of the betas.

P.start   The starting values for the transition matrix. A user should provide a square matrix with dimension equal to the number of states. By default, draws from the Beta(0.9, 0.1) are used to construct a proper transition matrix for each raw except the last raw.

b0   The prior mean of β. This can either be a scalar or a column vector with dimension equal to the number of betas. If this takes a scalar value, then that value will serve as the prior mean for all of the betas.

B0   The prior precision of β. This can either be a scalar or a square matrix with dimensions equal to the number of betas. If this takes a scalar value, then that value times an identity matrix serves as the prior precision of beta. Default value of 0 is equivalent to an improper uniform prior for beta.

a a is the shape1 beta prior for transition probabilities. By default, the expected duration is computed and corresponding a and b values are assigned. The expected duration is the sample period divided by the number of states.

b b is the shape2 beta prior for transition probabilities. By default, the expected duration is computed and corresponding a and b values are assigned. The expected duration is the sample period divided by the number of states.

marginal.likelihood

How should the marginal likelihood be calculated? Options are: `none` in which case the marginal likelihood will not be calculated, and `Chib95` in which case the method of Chib (1995) is used.

...   further arguments to be passed

Details

`MCMCprobitChange` simulates from the posterior distribution of a probit regression model with multiple parameter breaks. The simulation is based on Chib (1998) and Park (2011).

The model takes the following form:

$$\Pr(y_t = 1) = \Phi(x_i'\beta_m) \ \ m = 1, \ldots, M$$

Where M is the number of states, and β_m is a parameter when a state is m at t.

We assume Gaussian distribution for prior of β:

$$\beta_m \sim \mathcal{N}(b_0, B_0^{-1}), \ \ m = 1, \ldots, M$$

And:

$$p_{mm} \sim \mathcal{B}eta(a, b), \ \ m = 1, \ldots, M$$

Where M is the number of states.

Value

An mcmc object that contains the posterior sample. This object can be summarized by functions provided by the coda package. The object contains an attribute `prob.state` storage matrix that contains the probability of $state_i$ for each period, the log-likelihood of the model (`loglike`), and the log-marginal likelihood of the model (`logmarglike`).

References

Jong Hee Park. 2011. "Changepoint Analysis of Binary and Ordinal Probit Models: An Application to Bank Rate Policy Under the Interwar Gold Standard." *Political Analysis*. 19: 188-204. <doi:10.1093/pan/mpr007>

Andrew D. Martin, Kevin M. Quinn, and Jong Hee Park. 2011. "MCMCpack: Markov Chain Monte Carlo in R.", *Journal of Statistical Software*. 42(9): 1-21. doi: 10.18637/jss.v042.i09.

Siddhartha Chib. 1998. "Estimation and comparison of multiple change-point models." *Journal of Econometrics*. 86: 221-241.

Albert, J. H. and S. Chib. 1993. "Bayesian Analysis of Binary and Polychotomous Response Data." *J. Amer. Statist. Assoc.* 88, 669-679

See Also

[plotState](), [plotChangepoint]()

Examples

```
## Not run:
set.seed(1973)
x1 <- rnorm(300, 0, 1)
true.beta <- c(-.5, .2, 1)
true.alpha <- c(.1, -1., .2)
X <- cbind(1, x1)

## set two true breaks at 100 and 200
true.phi1 <- pnorm(true.alpha[1] + x1[1:100]*true.beta[1])
true.phi2 <- pnorm(true.alpha[2] + x1[101:200]*true.beta[2])
true.phi3 <-  pnorm(true.alpha[3] + x1[201:300]*true.beta[3])

## generate y
y1 <- rbinom(100, 1, true.phi1)
y2 <- rbinom(100, 1, true.phi2)
y3 <- rbinom(100, 1, true.phi3)
Y <- as.ts(c(y1, y2, y3))

## fit multiple models with a varying number of breaks
out0 <- MCMCprobitChange(formula=Y~X-1, data=parent.frame(), m=0,
                    mcmc=1000, burnin=1000, thin=1, verbose=1000,
                b0 = 0, B0 = 0.1, a = 1, b = 1,  marginal.likelihood = c("Chib95"))
out1 <- MCMCprobitChange(formula=Y~X-1, data=parent.frame(), m=1,
                    mcmc=1000, burnin=1000, thin=1, verbose=1000,
                b0 = 0, B0 = 0.1, a = 1, b = 1,  marginal.likelihood = c("Chib95"))
out2 <- MCMCprobitChange(formula=Y~X-1, data=parent.frame(), m=2,
                    mcmc=1000, burnin=1000, thin=1, verbose=1000,
                b0 = 0, B0 = 0.1, a = 1, b = 1,  marginal.likelihood = c("Chib95"))
out3 <- MCMCprobitChange(formula=Y~X-1, data=parent.frame(), m=3,
                    mcmc=1000, burnin=1000, thin=1, verbose=1000,
                b0 = 0, B0 = 0.1, a = 1, b = 1,  marginal.likelihood = c("Chib95"))

## find the most reasonable one
BayesFactor(out0, out1, out2, out3)

## draw plots using the "right" model
plotState(out2)
plotChangepoint(out2)

## End(Not run)
```

MCMCquantreg   *Bayesian quantile regression using Gibbs sampling*

Description

This function fits quantile regression models under Bayesian inference. The function samples from the posterior distribution using Gibbs sampling with data augmentation. A multivariate normal prior

is assumed for β. The user supplies the prior parameters. A sample of the posterior distribution is returned as an mcmc object, which can then be analysed by functions in the coda package.

Usage

```
MCMCquantreg(
  formula,
  data = NULL,
  tau = 0.5,
  burnin = 1000,
  mcmc = 10000,
  thin = 1,
  verbose = 0,
  seed = sample(1:1e+06, 1),
  beta.start = NA,
  b0 = 0,
  B0 = 0,
  ...
)
```

Arguments

formula	Model formula.
data	Data frame.
tau	The quantile of interest. Must be between 0 and 1. The default value of 0.5 corresponds to median regression.
burnin	The number of burn-in iterations for the sampler.
mcmc	The number of MCMC iterations after burnin.
thin	The thinning interval used in the simulation. The number of MCMC iterations must be divisible by this value.
verbose	A switch which determines whether or not the progress of the sampler is printed to the screen. If verbose is greater than 0 the iteration number and the most recently sampled values of β and σ are printed to the screen every verboseth iteration.
seed	The seed for the random number generator. If NA, the Mersenne Twister generator is used with default seed 12345; if an integer is passed it is used to seed the Mersenne twister. The default value for this argument is a random integer between 1 and 1,000,000. This default value ensures that if the function is used again with a different value of τ, it is extremely unlikely that the seed will be identical. The user can also pass a list of length two to use the L'Ecuyer random number generator, which is suitable for parallel computation. The first element of the list is the L'Ecuyer seed, which is a vector of length six or NA (if NA a default seed of rep(12345,6) is used). The second element of list is a positive substream number. See the MCMCpack specification for more details.
beta.start	The starting values for β. This can either be a scalar or a column vector with dimension equal to the dimension of β. The default value of NA will use the

OLS estimate $\hat{\beta}$ with $\hat{\sigma}\Phi^{-1}(\tau)$ added on to the first element of $\hat{\beta}$ as the starting value. ($\hat{\sigma}^2$ denotes the usual unbiased estimator of σ^2 under ordinary mean regression and $\Phi^{-1}(\tau)$ denotes the inverse of the cumulative density function of the standard normal distribution.) Note that the default value assume that an intercept is included in the model. If a scalar is given, that value will serve as the starting value for all β.

b0   The prior mean of β. This can either be a scalar or a column vector with dimension equal to the dimension of

β. If this takes a scalar value, then that value will serve as the prior mean for all β.

B0   The prior precision of β. This can either be a scalar or a square matrix with dimensions equal to the number of betas. If this takes a scalar value, then that value times an identity matrix serves as the prior precision of β. Default value of 0 is equivalent to an improper uniform prior for β.

. . .   further arguments to be passed

Details

MCMCquantreg simulates from the posterior distribution using Gibbs sampling with data augmentation (see http://people.brunel.ac.uk/~mastkky/). β are drawn from a multivariate normal distribution. The augmented data are drawn conditionally from the inverse Gaussian distribution. The simulation is carried out in compiled C++ code to maximise efficiency. Please consult the coda documentation for a comprehensive list of functions that can be used to analyse the posterior sample.

We assume the model

$$Q_\tau(y_i|x_i) = x_i'\beta$$

where $Q_\tau(y_i|x_i)$ denotes the conditional τth quantile of y_i given x_i, and $\beta = \beta(\tau)$ are the regression parameters possibly dependent on τ. The likelihood is formed based on assuming independent Asymmetric Laplace distributions on the y_i with skewness parameter τ and location parameters $x_i'\beta$. This assumption ensures that the likelihood function is maximised by the τth conditional quantile of the response variable. We assume standard, semi-conjugate priors on β:

$$\beta \sim \mathcal{N}(b_0, B_0^{-1})$$

Only starting values for β are allowed for this sampler.

Value

An mcmc object that contains the posterior sample. This object can be summarised by functions provided by the coda package.

Author(s)

Craig Reed

References

Daniel Pemstein, Kevin M. Quinn, and Andrew D. Martin. 2007. *Scythe Statistical Library 1.2.* http://scythe.lsa.umich.edu.

Craig Reed and Keming Yu. 2009. "An Efficient Gibbs Sampler for Bayesian Quantile Regression." Technical Report.

Keming Yu and Jin Zhang. 2005. "A Three Parameter Asymmetric Laplace Distribution and it's extensions." *Communications in Statistics - Theory and Methods*, 34, 1867-1879.

Martyn Plummer, Nicky Best, Kate Cowles, and Karen Vines. 2006. "Output Analysis and Diagnostics for MCMC (CODA)", *R News*. 6(1): 7-11. https://CRAN.R-project.org/doc/Rnews/Rnews_2006-1.pdf.

See Also

MCMCregress, plot.mcmc, summary.mcmc, lm, rq

Examples

```
## Not run:

x<-rep(1:10,5)
y<-rnorm(50,mean=x)
posterior_50 <- MCMCquantreg(y~x)
posterior_95 <- MCMCquantreg(y~x, tau=0.95, verbose=10000,
    mcmc=50000, thin=10, seed=2)
plot(posterior_50)
plot(posterior_95)
raftery.diag(posterior_50)
autocorr.plot(posterior_95)
summary(posterior_50)
summary(posterior_95)

## End(Not run)
```

MCMCregress   *Markov Chain Monte Carlo for Gaussian Linear Regression*

Description

This function generates a sample from the posterior distribution of a linear regression model with Gaussian errors using Gibbs sampling (with a multivariate Gaussian prior on the beta vector, and an inverse Gamma prior on the conditional error variance). The user supplies data and priors, and a sample from the posterior distribution is returned as an mcmc object, which can be subsequently analyzed with functions provided in the coda package.

Usage

```
MCMCregress(
  formula,
  data = NULL,
  burnin = 1000,
  mcmc = 10000,
  thin = 1,
  verbose = 0,
  seed = NA,
  beta.start = NA,
  b0 = 0,
  B0 = 0,
  c0 = 0.001,
  d0 = 0.001,
  sigma.mu = NA,
  sigma.var = NA,
  marginal.likelihood = c("none", "Laplace", "Chib95"),
  ...
)
```

Arguments

formula	Model formula.
data	Data frame.
burnin	The number of burn-in iterations for the sampler.
mcmc	The number of MCMC iterations after burnin.
thin	The thinning interval used in the simulation. The number of MCMC iterations must be divisible by this value.
verbose	A switch which determines whether or not the progress of the sampler is printed to the screen. If verbose is greater than 0 the iteration number, the β vector, and the error variance are printed to the screen every verboseth iteration.
seed	The seed for the random number generator. If NA, the Mersenne Twister generator is used with default seed 12345; if an integer is passed it is used to seed the Mersenne twister. The user can also pass a list of length two to use the L'Ecuyer random number generator, which is suitable for parallel computation. The first element of the list is the L'Ecuyer seed, which is a vector of length six or NA (if NA a default seed of rep(12345,6) is used). The second element of list is a positive substream number. See the MCMCpack specification for more details.
beta.start	The starting values for the β vector. This can either be a scalar or a column vector with dimension equal to the number of betas. The default value of of NA will use the OLS estimate of β as the starting value. If this is a scalar, that value will serve as the starting value mean for all of the betas.
b0	The prior mean of β. This can either be a scalar or a column vector with dimension equal to the number of betas. If this takes a scalar value, then that value will serve as the prior mean for all of the betas.

B0	The prior precision of β. This can either be a scalar or a square matrix with dimensions equal to the number of betas. If this takes a scalar value, then that value times an identity matrix serves as the prior precision of beta. Default value of 0 is equivalent to an improper uniform prior for beta.
c0	$c_0/2$ is the shape parameter for the inverse Gamma prior on σ^2 (the variance of the disturbances). The amount of information in the inverse Gamma prior is something like that from c_0 pseudo-observations.
d0	$d_0/2$ is the scale parameter for the inverse Gamma prior on σ^2 (the variance of the disturbances). In constructing the inverse Gamma prior, d_0 acts like the sum of squared errors from the c_0 pseudo-observations.
sigma.mu	The mean of the inverse Gamma prior on σ^2. $sigma.mu$ and $sigma.var$ allow users to choose the inverse Gamma prior by choosing its mean and variance.
sigma.var	The variacne of the inverse Gamma prior on σ^2. $sigma.mu$ and $sigma.var$ allow users to choose the inverse Gamma prior by choosing its mean and variance.
marginal.likelihood	
	How should the marginal likelihood be calculated? Options are: none in which case the marginal likelihood will not be calculated, Laplace in which case the Laplace approximation (see Kass and Raftery, 1995) is used, and Chib95 in which case the method of Chib (1995) is used.
...	further arguments to be passed.

Details

MCMCregress simulates from the posterior distribution using standard Gibbs sampling (a multivariate Normal draw for the betas, and an inverse Gamma draw for the conditional error variance). The simulation proper is done in compiled C++ code to maximize efficiency. Please consult the coda documentation for a comprehensive list of functions that can be used to analyze the posterior sample.

The model takes the following form:

$$y_i = x_i'\beta + \varepsilon_i$$

Where the errors are assumed to be Gaussian:

$$\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$$

We assume standard, semi-conjugate priors:

$$\beta \sim \mathcal{N}(b_0, B_0^{-1})$$

And:

$$\sigma^{-2} \sim \mathcal{G}amma(c_0/2, d_0/2)$$

Where β and σ^{-2} are assumed *a priori* independent. Note that only starting values for β are allowed because simulation is done using Gibbs sampling with the conditional error variance as the first block in the sampler.

Value

An mcmc object that contains the posterior sample. This object can be summarized by functions provided by the coda package.

References

Andrew D. Martin, Kevin M. Quinn, and Jong Hee Park. 2011. "MCMCpack: Markov Chain Monte Carlo in R.", *Journal of Statistical Software*. 42(9): 1-21. doi: 10.18637/jss.v042.i09.

Siddhartha Chib. 1995. "Marginal Likelihood from the Gibbs Output." *Journal of the American Statistical Association*. 90: 1313-1321.

Robert E. Kass and Adrian E. Raftery. 1995. "Bayes Factors." *Journal of the American Statistical Association*. 90: 773-795.

Daniel Pemstein, Kevin M. Quinn, and Andrew D. Martin. 2007. *Scythe Statistical Library 1.0*. http://scythe.lsa.umich.edu.

Martyn Plummer, Nicky Best, Kate Cowles, and Karen Vines. 2006. "Output Analysis and Diagnostics for MCMC (CODA)", *R News*. 6(1): 7-11. https://CRAN.R-project.org/doc/Rnews/Rnews_2006-1.pdf.

See Also

plot.mcmc, summary.mcmc, lm

Examples

```
## Not run:
line   <- list(X = c(-2,-1,0,1,2), Y = c(1,3,3,3,5))
posterior  <- MCMCregress(Y~X, b0=0, B0 = 0.1,
     sigma.mu = 5, sigma.var = 25, data=line, verbose=1000)
plot(posterior)
raftery.diag(posterior)
summary(posterior)

## End(Not run)
```

MCMCregressChange | *Markov Chain Monte Carlo for a linear Gaussian Multiple Change-point Model*

Description

This function generates a sample from the posterior distribution of a linear Gaussian model with multiple changepoints. The function uses the Markov chain Monte Carlo method of Chib (1998). The user supplies data and priors, and a sample from the posterior distribution is returned as an mcmc object, which can be subsequently analyzed with functions provided in the coda package.

Usage

```
MCMCregressChange(
  formula,
  data = parent.frame(),
  m = 1,
  b0 = 0,
  B0 = 0,
  c0 = 0.001,
  d0 = 0.001,
  sigma.mu = NA,
  sigma.var = NA,
  a = NULL,
  b = NULL,
  mcmc = 1000,
  burnin = 1000,
  thin = 1,
  verbose = 0,
  seed = NA,
  beta.start = NA,
  P.start = NA,
  random.perturb = FALSE,
  WAIC = FALSE,
  marginal.likelihood = c("none", "Chib95"),
  ...
)
```

Arguments

formula	Model formula.
data	Data frame.
m	The number of changepoints.
b0	The prior mean of β. This can either be a scalar or a column vector with dimension equal to the number of betas. If this takes a scalar value, then that value will serve as the prior mean for all of the betas.
B0	The prior precision of β. This can either be a scalar or a square matrix with dimensions equal to the number of betas. If this takes a scalar value, then that value times an identity matrix serves as the prior precision of beta. Default value of 0 is equivalent to an improper uniform prior for beta.
c0	$c_0/2$ is the shape parameter for the inverse Gamma prior on σ^2 (the variance of the disturbances). The amount of information in the inverse Gamma prior is something like that from c_0 pseudo-observations.
d0	$d_0/2$ is the scale parameter for the inverse Gamma prior on σ^2 (the variance of the disturbances). In constructing the inverse Gamma prior, d_0 acts like the sum of squared errors from the c_0 pseudo-observations.
sigma.mu	The mean of the inverse Gamma prior on σ^2. $sigma.mu$ and $sigma.var$ allow users to choose the inverse Gamma prior by choosing its mean and variance.

sigma.var	The variacne of the inverse Gamma prior on σ^2. $sigma.mu$ and $sigma.var$ allow users to choose the inverse Gamma prior by choosing its mean and variance.
a	a is the shape1 beta prior for transition probabilities. By default, the expected duration is computed and corresponding a and b values are assigned. The expected duration is the sample period divided by the number of states.
b	b is the shape2 beta prior for transition probabilities. By default, the expected duration is computed and corresponding a and b values are assigned. The expected duration is the sample period divided by the number of states.
mcmc	The number of MCMC iterations after burnin.
burnin	The number of burn-in iterations for the sampler.
thin	The thinning interval used in the simulation. The number of MCMC iterations must be divisible by this value.
verbose	A switch which determines whether or not the progress of the sampler is printed to the screen. If verbose is greater than 0 the iteration number, the β vector, and the error variance are printed to the screen every verboseth iteration.
seed	The seed for the random number generator. If NA, the Mersenne Twister generator is used with default seed 12345; if an integer is passed it is used to seed the Mersenne twister. The user can also pass a list of length two to use the L'Ecuyer random number generator, which is suitable for parallel computation. The first element of the list is the L'Ecuyer seed, which is a vector of length six or NA (if NA a default seed of rep(12345,6) is used). The second element of list is a positive substream number. See the MCMCpack specification for more details.
beta.start	The starting values for the β vector. This can either be a scalar or a column vector with dimension equal to the number of betas. The default value of of NA will use the MLE estimate of β as the starting value. If this is a scalar, that value will serve as the starting value mean for all of the betas.
P.start	The starting values for the transition matrix. A user should provide a square matrix with dimension equal to the number of states. By default, draws from the Beta(0.9, 0.1) are used to construct a proper transition matrix for each raw except the last raw.
random.perturb	If TRUE, randomly sample hidden states whenever regularly sampled hidden states have at least one single observation state (SOS). SOS is a sign of overfitting in non-ergodic hidden Markov models.
WAIC	Compute the Widely Applicable Information Criterion (Watanabe 2010).
marginal.likelihood	
	How should the marginal likelihood be calculated? Options are: none in which case the marginal likelihood will not be calculated, and Chib95 in which case the method of Chib (1995) is used.
...	further arguments to be passed

Details

MCMCregressChange simulates from the posterior distribution of the linear regression model with multiple changepoints.

The model takes the following form:

$$y_t = x_t'\beta_i + I(s_t = i)\varepsilon_t, \ \ i = 1, \ldots, k$$

Where k is the number of states and $I(s_t = i)$ is an indicator function that becomes 1 when a state at t is i and otherwise 0.

The errors are assumed to be Gaussian in each regime:

$$I(s_t = i)\varepsilon_t \sim \mathcal{N}(0, \sigma_i^2)$$

We assume standard, semi-conjugate priors:

$$\beta_i \sim \mathcal{N}(b_0, B_0^{-1}), \ \ i = 1, \ldots, k$$

And:

$$\sigma_i^{-2} \sim \mathcal{G}amma(c_0/2, d_0/2), \ \ i = 1, \ldots, k$$

Where β_i and σ_i^{-2} are assumed *a priori* independent.

The simulation proper is done in compiled C++ code to maximize efficiency.

Value

An mcmc object that contains the posterior sample. This object can be summarized by functions provided by the coda package. The object contains an attribute prob.state storage matrix that contains the probability of $state_i$ for each period, the log-likelihood of the model (loglike), and the log-marginal likelihood of the model (logmarglike).

References

Jong Hee Park, 2012. "Unified Method for Dynamic and Cross-Sectional Heterogeneity: Introducing Hidden Markov Panel Models." *American Journal of Political Science*.56: 1040-1054. <doi: 10.1111/j.1540-5907.2012.00590.x>

Sumio Watanabe. 2010. "Asymptotic equivalence of Bayes cross validation and widely applicable information criterion in singular learning theory" *Journal of Machine Learning Research*. 11: 3571-3594.

Siddhartha Chib. 1995. "Marginal Likelihood from the Gibbs Output." *Journal of the American Statistical Association*. 90: 1313-1321. <doi: 10.1016/S0304-4076(97)00115-2>

Siddhartha Chib. 1998. "Estimation and comparison of multiple change-point models." *Journal of Econometrics*. 86: 221-241. <doi: 10.1080/01621459.1995.10476635>

Andrew D. Martin, Kevin M. Quinn, and Jong Hee Park. 2011. "MCMCpack: Markov Chain Monte Carlo in R.", *Journal of Statistical Software*. 42(9): 1-21. doi: 10.18637/jss.v042.i09.

See Also

[plotState](), [plotChangepoint]()

Examples

```
## Not run:
set.seed(1119)
n <- 100
x1 <- runif(n)
true.beta1 <- c(2, -2)
true.beta2 <- c(0,  2)
true.Sigma <- c(1, 2)
true.s <- rep(1:2, each=n/2)

mu1 <- cbind(1, x1[true.s==1])%*%true.beta1
mu2 <- cbind(1, x1[true.s==2])%*%true.beta2

y <- as.ts(c(rnorm(n/2, mu1, sd=sqrt(true.Sigma[1])), rnorm(n/2, mu2, sd=sqrt(true.Sigma[2]))))
formula=y ~ x1

ols1 <- lm(y[true.s==1] ~x1[true.s==1])
ols2 <- lm(y[true.s==2] ~x1[true.s==2])

## prior
b0 <- 0
B0 <- 0.1
sigma.mu=sd(y)
sigma.var=var(y)

## models
model0 <-  MCMCregressChange(formula, m=0, b0=b0, B0=B0, mcmc=100, burnin=100,
            sigma.mu=sigma.mu, sigma.var=sigma.var, marginal.likelihood="Chib95")
model1 <-  MCMCregressChange(formula, m=1, b0=b0, B0=B0, mcmc=100, burnin=100,
            sigma.mu=sigma.mu, sigma.var=sigma.var, marginal.likelihood="Chib95")
model2 <-  MCMCregressChange(formula, m=2, b0=b0, B0=B0, mcmc=100, burnin=100,
            sigma.mu=sigma.mu, sigma.var=sigma.var, marginal.likelihood="Chib95")
model3 <-  MCMCregressChange(formula, m=3, b0=b0, B0=B0, mcmc=100, burnin=100,
            sigma.mu=sigma.mu, sigma.var=sigma.var, marginal.likelihood="Chib95")
model4 <-  MCMCregressChange(formula, m=4, b0=b0, B0=B0, mcmc=100, burnin=100,
            sigma.mu=sigma.mu, sigma.var=sigma.var, marginal.likelihood="Chib95")
model5 <-  MCMCregressChange(formula, m=5, b0=b0, B0=B0, mcmc=100, burnin=100,
            sigma.mu=sigma.mu, sigma.var=sigma.var, marginal.likelihood="Chib95")

print(BayesFactor(model0, model1, model2, model3, model4, model5))
plotState(model1)
plotChangepoint(model1)


## End(Not run)
```

MCMCresidualBreakAnalysis

> *Break Analysis of Univariate Time Series using Markov Chain Monte Carlo*

Description

This function performs a break analysis for univariate time series data using a linear Gaussian changepoint model. The code is written mainly for an internal use in testpanelSubjectBreak.

Usage

```
MCMCresidualBreakAnalysis(
  resid,
  m = 1,
  b0 = 0,
  B0 = 0.001,
  c0 = 0.1,
  d0 = 0.1,
  a = NULL,
  b = NULL,
  mcmc = 1000,
  burnin = 1000,
  thin = 1,
  verbose = 0,
  seed = NA,
  beta.start = NA,
  P.start = NA,
  random.perturb = FALSE,
  WAIC = FALSE,
  marginal.likelihood = c("none", "Chib95"),
  ...
)
```

Arguments

resid	Univariate time series
m	The number of breaks.
b0	The prior mean of β. This can either be a scalar or a column vector with dimension equal to the number of betas. If this takes a scalar value, then that value will serve as the prior mean for all of the betas.
B0	The prior precision of β. This can either be a scalar or a square matrix with dimensions equal to the number of betas. If this takes a scalar value, then that value times an identity matrix serves as the prior precision of beta. Default value of 0 is equivalent to an improper uniform prior for beta.
c0	$c_0/2$ is the shape parameter for the inverse Gamma prior on σ^2 (the variance of the disturbances). The amount of information in the inverse Gamma prior is something like that from c_0 pseudo-observations.

d0	$d_0/2$ is the scale parameter for the inverse Gamma prior on σ^2 (the variance of the disturbances). In constructing the inverse Gamma prior, d_0 acts like the sum of squared errors from the c_0 pseudo-observations.
a	a is the shape1 beta prior for transition probabilities. By default, the expected duration is computed and corresponding a and b values are assigned. The expected duration is the sample period divided by the number of states.
b	b is the shape2 beta prior for transition probabilities. By default, the expected duration is computed and corresponding a and b values are assigned. The expected duration is the sample period divided by the number of states.
mcmc	The number of MCMC iterations after burnin.
burnin	The number of burn-in iterations for the sampler.
thin	The thinning interval used in the simulation. The number of MCMC iterations must be divisible by this value.
verbose	A switch which determines whether or not the progress of the sampler is printed to the screen. If verbose is greater than 0 the iteration number, the β vector, and the error variance are printed to the screen every verboseth iteration.
seed	The seed for the random number generator. If NA, the Mersenne Twister generator is used with default seed 12345; if an integer is passed it is used to seed the Mersenne twister. The user can also pass a list of length two to use the L'Ecuyer random number generator, which is suitable for parallel computation. The first element of the list is the L'Ecuyer seed, which is a vector of length six or NA (if NA a default seed of rep(12345,6) is used). The second element of list is a positive substream number. See the MCMCpack specification for more details.
beta.start	The starting values for the β vector. This can either be a scalar or a column vector with dimension equal to the number of betas. The default value of of NA will use the OLS estimate of β as the starting value. If this is a scalar, that value will serve as the starting value mean for all of the betas.
P.start	The starting values for the transition matrix. A user should provide a square matrix with dimension equal to the number of states. By default, draws from the Beta(0.9, 0.1) are used to construct a proper transition matrix for each raw except the last raw.
random.perturb	If TRUE, randomly sample hidden states whenever regularly sampled hidden states have at least one single observation state. It's one method to avoid overfitting in a non-ergodic hidden Markov models. See Park and Sohn (2017).
WAIC	Compute the Widely Applicable Information Criterion (Watanabe 2010).
marginal.likelihood	How should the marginal likelihood be calculated? Options are: none in which case the marginal likelihood will not be calculated, and Chib95 in which case the method of Chib (1995) is used.
...	further arguments to be passed

Details

MCMCresidualBreakAnalysis simulates from the posterior distribution using standard Gibbs sampling (a multivariate Normal draw for the betas, and an inverse Gamma draw for the conditional

error variance). The simulation proper is done in compiled C++ code to maximize efficiency. Please consult the coda documentation for a comprehensive list of functions that can be used to analyze the posterior sample.

The model takes the following form:

$$y_i \sim \mathcal{N}(\beta_m, \sigma_m^2) \ \ m = 1, \ldots, M$$

We assume standard, semi-conjugate priors:

$$\beta \sim \mathcal{N}(b_0, B_0^{-1})$$

And:

$$\sigma^{-2} \sim \mathcal{G}amma(c_0/2, d_0/2)$$

Where β and σ^{-2} are assumed *a priori* independent.

And:

$$p_{mm} \sim \mathcal{B}eta(a, b), \ \ m = 1, \ldots, M$$

Where M is the number of states.

Value

An mcmc object that contains the posterior sample. This object can be summarized by functions provided by the coda package.

References

Jong Hee Park and Yunkyu Sohn. 2017. "Detecting Structural Changes in Network Data: An Application to Changes in Military Alliance Networks, 1816-2012". Working Paper.

Jong Hee Park, 2012. "Unified Method for Dynamic and Cross-Sectional Heterogeneity: Introducing Hidden Markov Panel Models." *American Journal of Political Science*.56: 1040-1054. <doi: 10.1111/j.1540-5907.2012.00590.x>

Sumio Watanabe. 2010. "Asymptotic equivalence of Bayes cross validation and widely applicable information criterion in singular learning theory" *Journal of Machine Learning Research*. 11: 3571-3594.

Siddhartha Chib. 1995. "Marginal Likelihood from the Gibbs Output." *Journal of the American Statistical Association*. 90: 1313-1321. <doi: 10.1016/S0304-4076(97)00115-2>

Siddhartha Chib. 1998. "Estimation and comparison of multiple change-point models." *Journal of Econometrics*. 86: 221-241. <doi: 10.1080/01621459.1995.10476635>

See Also

plot.mcmc, summary.mcmc, lm

Examples

```
## Not run:
line   <- list(X = c(-2,-1,0,1,2), Y = c(1,3,3,3,5))
ols <- lm(Y~X)
residual <-   rstandard(ols)
posterior <- MCMCresidualBreakAnalysis(residual, m = 1, data=line, mcmc=1000, verbose=200)
plotState(posterior)
summary(posterior)

## End(Not run)
```

MCMCSVDreg *Markov Chain Monte Carlo for SVD Regression*

Description

This function generates a sample from the posterior distribution of a linear regression model with Gaussian errors in which the design matrix has been decomposed with singular value decomposition.The sampling is done via the Gibbs sampling algorithm. The user supplies data and priors, and a sample from the posterior distribution is returned as an mcmc object, which can be subsequently analyzed with functions provided in the coda package.

Usage

```
MCMCSVDreg(
  formula,
  data = NULL,
  burnin = 1000,
  mcmc = 10000,
  thin = 1,
  verbose = 0,
  seed = NA,
  tau2.start = 1,
  g0 = 0,
  a0 = 0.001,
  b0 = 0.001,
  c0 = 2,
  d0 = 2,
  w0 = 1,
  beta.samp = FALSE,
  intercept = TRUE,
  ...
)
```

Arguments

formula	Model formula. Predictions are returned for elements of y that are coded as NA.
data	Data frame.
burnin	The number of burn-in iterations for the sampler.
mcmc	The number of MCMC iterations after burnin.
thin	The thinning interval used in the simulation. The number of MCMC iterations must be divisible by this value.
verbose	A switch which determines whether or not the progress of the sampler is printed to the screen. If verbose is greater than 0 the iteration number, the β vector, and the error variance are printed to the screen every verboseth iteration.
seed	The seed for the random number generator. If NA, the Mersenne Twister generator is used with default seed 12345; if an integer is passed it is used to seed the Mersenne twister. The user can also pass a list of length two to use the L'Ecuyer random number generator, which is suitable for parallel computation. The first element of the list is the L'Ecuyer seed, which is a vector of length six or NA (if NA a default seed of rep(12345,6) is used). The second element of list is a positive substream number. See the MCMCpack specification for more details.
tau2.start	The starting values for the τ^2 vector. Can be either a scalar or a vector. If a scalar is passed then that value will be the starting value for all elements of τ^2.
g0	The prior mean of γ. This can either be a scalar or a column vector with dimension equal to the number of gammas. If this takes a scalar value, then that value will serve as the prior mean for all of the betas.
a0	$a_0/2$ is the shape parameter for the inverse Gamma prior on σ^2 (the variance of the disturbances). The amount of information in the inverse Gamma prior is something like that from a_0 pseudo-observations.
b0	$b_0/2$ is the scale parameter for the inverse Gamma prior on σ^2 (the variance of the disturbances). In constructing the inverse Gamma prior, b_0 acts like the sum of squared errors from the a_0 pseudo-observations.
c0	$c_0/2$ is the shape parameter for the inverse Gamma prior on τ_i^2.
d0	$d_0/2$ is the scale parameter for the inverse Gamma prior on τ_i^2.
w0	The prior probability that $\gamma_i = 0$. Can be either a scalar or an N vector where N is the number of observations.
beta.samp	Logical indicating whether the sampled elements of beta should be stored and returned.
intercept	Logical indicating whether the original design matrix should include a constant term.
...	further arguments to be passed

Details

The model takes the following form:

$$y = X\beta + \varepsilon$$

Where the errors are assumed to be iid Gaussian:

$$\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$$

Let N denote the number of rows of X and P the number of columns of X. Unlike the standard regression setup where $N >> P$ here it is the case that $P >> N$.

To deal with this problem a singular value decomposition of X' is performed: $X' = ADF$ and the regression model becomes

$$y = F'D\gamma + \varepsilon$$

where $\gamma = A'\beta$

We assume the following priors:

$$\sigma^{-2} \sim \mathcal{G}amma(a_0/2, b_0/2)$$

$$\tau^{-2} \sim \mathcal{G}amma(c_0/2, d_0/2)$$

$$\gamma_i \sim w0_i\delta_0 + (1 - w0_i)\mathcal{N}(g0_i, \sigma^2\tau_i^2/d_i^2)$$

where δ_0 is a unit point mass at 0 and d_i is the ith diagonal element of D.

Value

An mcmc object that contains the posterior sample. This object can be summarized by functions provided by the coda package.

References

Mike West, Josheph Nevins, Jeffrey Marks, Rainer Spang, and Harry Zuzan. 2000. "DNA Microarray Data Analysis and Regression Modeling for Genetic Expression Profiling." Duke ISDS working paper.

Gottardo, Raphael, and Adrian Raftery. 2004. "Markov chain Monte Carlo with mixtures of singular distributions." Statistics Department, University of Washington, Technical Report 470.

Andrew D. Martin, Kevin M. Quinn, and Jong Hee Park. 2011. "MCMCpack: Markov Chain Monte Carlo in R.", *Journal of Statistical Software*. 42(9): 1-21. doi: 10.18637/jss.v042.i09.

Daniel Pemstein, Kevin M. Quinn, and Andrew D. Martin. 2007. *Scythe Statistical Library 1.0.* http://scythe.lsa.umich.edu.

Martyn Plummer, Nicky Best, Kate Cowles, and Karen Vines. 2006. "Output Analysis and Diagnostics for MCMC (CODA)", *R News*. 6(1): 7-11. https://CRAN.R-project.org/doc/Rnews/Rnews_2006-1.pdf.

See Also

plot.mcmc, summary.mcmc, lm

MCMCtobit	*Markov Chain Monte Carlo for Gaussian Linear Regression with a Censored Dependent Variable*

Description

This function generates a sample from the posterior distribution of a linear regression model with Gaussian errors using Gibbs sampling (with a multivariate Gaussian prior on the beta vector, and an inverse Gamma prior on the conditional error variance). The dependent variable may be censored from below, from above, or both. The user supplies data and priors, and a sample from the posterior distribution is returned as an mcmc object, which can be subsequently analyzed with functions provided in the coda package.

Usage

```
MCMCtobit(
  formula,
  data = NULL,
  below = 0,
  above = Inf,
  burnin = 1000,
  mcmc = 10000,
  thin = 1,
  verbose = 0,
  seed = NA,
  beta.start = NA,
  b0 = 0,
  B0 = 0,
  c0 = 0.001,
  d0 = 0.001,
  ...
)
```

Arguments

formula	A model formula.
data	A dataframe.
below	The point at which the dependent variable is censored from below. The default is zero. To censor from above only, specify that below = -Inf.
above	The point at which the dependent variable is censored from above. To censor from below only, use the default value of Inf.
burnin	The number of burn-in iterations for the sampler.
mcmc	The number of MCMC iterations after burnin.
thin	The thinning interval used in the simulation. The number of MCMC iterations must be divisible by this value.

verbose	A switch which determines whether or not the progress of the sampler is printed to the screen. If verbose is greater than 0 the iteration number, the β vector, and the error variance is printed to the screen every verboseth iteration.
seed	The seed for the random number generator. If NA, the Mersenne Twister generator is used with default seed 12345; if an integer is passed it is used to seed the Mersenne twister. The user can also pass a list of length two to use the L'Ecuyer random number generator, which is suitable for parallel computation. The first element of the list is the L'Ecuyer seed, which is a vector of length six or NA (if NA a default seed of rep(12345,6) is used). The second element of list is a positive substream number. See the MCMCpack specification for more details.
beta.start	The starting values for the β vector. This can either be a scalar or a column vector with dimension equal to the number of betas. The default value of of NA will use the OLS estimate of β as the starting value. If this is a scalar, that value will serve as the starting value mean for all of the betas.
b0	The prior mean of β. This can either be a scalar or a column vector with dimension equal to the number of betas. If this takes a scalar value, then that value will serve as the prior mean for all of the betas.
B0	The prior precision of β. This can either be a scalar or a square matrix with dimensions equal to the number of betas. If this takes a scalar value, then that value times an identity matrix serves as the prior precision of beta. Default value of 0 is equivalent to an improper uniform prior for beta.
c0	$c_0/2$ is the shape parameter for the inverse Gamma prior on σ^2 (the variance of the disturbances). The amount of information in the inverse Gamma prior is something like that from c_0 pseudo-observations.
d0	$d_0/2$ is the scale parameter for the inverse Gamma prior on σ^2 (the variance of the disturbances). In constructing the inverse Gamma prior, d_0 acts like the sum of squared errors from the c_0 pseudo-observations.
...	further arguments to be passed

Details

MCMCtobit simulates from the posterior distribution using standard Gibbs sampling (a multivariate Normal draw for the betas, and an inverse Gamma draw for the conditional error variance). MCMCtobit differs from MCMCregress in that the dependent variable may be censored from below, from above, or both. The simulation proper is done in compiled C++ code to maximize efficiency. Please consult the coda documentation for a comprehensive list of functions that can be used to analyze the posterior sample.

The model takes the following form:

$$y_i = x_i'\beta + \varepsilon_i,$$

where the errors are assumed to be Gaussian:

$$\varepsilon_i \sim \mathcal{N}(0, \sigma^2).$$

Let c_1 and c_2 be the two censoring points, and let y_i^* be the partially observed dependent variable. Then,

$$y_i = y_i^* \text{ if } c_1 < y_i^* < c_2,$$

$$y_i = c_1 \text{ if } c_1 \geq y_i^*,$$

$$y_i = c_2 \text{ if } c_2 \leq y_i^*.$$

We assume standard, semi-conjugate priors:

$$\beta \sim \mathcal{N}(b_0, B_0^{-1}),$$

and:

$$\sigma^{-2} \sim \mathcal{G}amma(c_0/2, d_0/2),$$

where β and σ^{-2} are assumed *a priori* independent. Note that only starting values for β are allowed because simulation is done using Gibbs sampling with the conditional error variance as the first block in the sampler.

Value

An mcmc object that contains the posterior sample. This object can be summarized by functions provided by the coda package.

Author(s)

Ben Goodrich, <goodrich.ben@gmail.com>, http://www.columbia.edu/~bg2382/

References

Andrew D. Martin, Kevin M. Quinn, and Jong Hee Park. 2011. "MCMCpack: Markov Chain Monte Carlo in R.", *Journal of Statistical Software*. 42(9): 1-21. doi: 10.18637/jss.v042.i09.

Daniel Pemstein, Kevin M. Quinn, and Andrew D. Martin. 2007. *Scythe Statistical Library 1.0*. http://scythe.lsa.umich.edu.

Martyn Plummer, Nicky Best, Kate Cowles, and Karen Vines. 2006. "Output Analysis and Diagnostics for MCMC (CODA)", *R News*. 6(1): 7-11. https://CRAN.R-project.org/doc/Rnews/Rnews_2006-1.pdf.

Siddhartha Chib. 1992. "Bayes inference in the Tobit censored regression model." *Journal of Econometrics*. 51:79-99.

James Tobin. 1958. "Estimation of relationships for limited dependent variables." *Econometrica*. 26:24-36.

See Also

plot.mcmc, summary.mcmc, survreg, MCMCregress

Examples

```
## Not run:
library(survival)
example(tobin)
summary(tfit)
tfit.mcmc <- MCMCtobit(durable ~ age + quant, data=tobin, mcmc=30000,
                          verbose=1000)
plot(tfit.mcmc)
raftery.diag(tfit.mcmc)
summary(tfit.mcmc)

## End(Not run)
```

MCmultinomdirichlet	*Monte Carlo Simulation from a Multinomial Likelihood with a Dirichlet Prior*

Description

This function generates a sample from the posterior distribution of a multinomial likelihood with a Dirichlet prior.

Usage

```
MCmultinomdirichlet(y, alpha0, mc = 1000, ...)
```

Arguments

y	A vector of data (number of successes for each category).
alpha0	The vector of parameters of the Dirichlet prior.
mc	The number of Monte Carlo draws to make.
...	further arguments to be passed

Details

MCmultinomdirichlet directly simulates from the posterior distribution. This model is designed primarily for instructional use. π is the parameter of interest of the multinomial distribution. It is of dimension $(d \times 1)$. We assume a conjugate Dirichlet prior:

$$\pi \sim \mathcal{D}irichlet(\alpha_0)$$

y is a $(d \times 1)$ vector of observed data.

Value

An mcmc object that contains the posterior sample. This object can be summarized by functions provided by the coda package.

See Also

plot.mcmc, summary.mcmc

Examples

```
## Not run:
## Example from Gelman, et. al. (1995, p. 78)
posterior <- MCmultinomdirichlet(c(727,583,137), c(1,1,1), mc=10000)
bush.dukakis.diff <- posterior[,1] - posterior[,2]
cat("Pr(Bush > Dukakis): ",
    sum(bush.dukakis.diff > 0) / length(bush.dukakis.diff), "\n")
hist(bush.dukakis.diff)

## End(Not run)
```

MCnormalnormal   *Monte Carlo Simulation from a Normal Likelihood (with known variance) with a Normal Prior*

Description

This function generates a sample from the posterior distribution of a Normal likelihood (with known variance) with a Normal prior.

Usage

```
MCnormalnormal(y, sigma2, mu0, tau20, mc = 1000, ...)
```

Arguments

y	The data.
sigma2	The known variance of y.
mu0	The prior mean of mu.
tau20	The prior variance of mu.
mc	The number of Monte Carlo draws to make.
...	further arguments to be passed

Details

`MCnormalnormal` directly simulates from the posterior distribution. This model is designed primarily for instructional use. μ is the parameter of interest of the Normal distribution. We assume a conjugate normal prior:

$$\mu \sim \mathcal{N}(\mu_0, \tau_0^2)$$

y is a vector of observed data.

Value

An mcmc object that contains the posterior sample. This object can be summarized by functions provided by the coda package.

See Also

[plot.mcmc](), [summary.mcmc]()

Examples

```
## Not run:
y <- c(2.65, 1.80, 2.29, 2.11, 2.27, 2.61, 2.49, 0.96, 1.72, 2.40)
posterior <- MCMCpack:::MCnormalnormal(y, 1, 0, 1, 5000)
summary(posterior)
plot(posterior)
grid <- seq(-3,3,0.01)
plot(grid, dnorm(grid, 0, 1), type="l", col="red", lwd=3, ylim=c(0,1.4),
   xlab="mu", ylab="density")
lines(density(posterior), col="blue", lwd=3)
legend(-3, 1.4, c("prior", "posterior"), lwd=3, col=c("red", "blue"))

## End(Not run)
```

MCpoissongamma	*Monte Carlo Simulation from a Poisson Likelihood with a Gamma Prior*

Description

This function generates a sample from the posterior distribution of a Poisson likelihood with a Gamma prior.

Usage

```
MCpoissongamma(y, alpha, beta, mc = 1000, ...)
```

Arguments

`y`	A vector of counts (must be non-negative).
`alpha`	Gamma prior distribution shape parameter.
`beta`	Gamma prior distribution scale parameter.
`mc`	The number of Monte Carlo draws to make.
`...`	further arguments to be passed

Details

`MCpoissongamma` directly simulates from the posterior distribution. This model is designed primarily for instructional use. λ is the parameter of interest of the Poisson distribution. We assume a conjugate Gamma prior:

$$\lambda \sim \mathcal{G}amma(\alpha, \beta)$$

y is a vector of counts.

Value

An mcmc object that contains the posterior sample. This object can be summarized by functions provided by the coda package.

See Also

plot.mcmc, summary.mcmc

Examples

```
## Not run:
data(quine)
posterior <- MCpoissongamma(quine$Days, 15, 1, 5000)
summary(posterior)
plot(posterior)
grid <- seq(14,18,0.01)
plot(grid, dgamma(grid, 15, 1), type="l", col="red", lwd=3, ylim=c(0,1.3),
  xlab="lambda", ylab="density")
lines(density(posterior), col="blue", lwd=3)
legend(17, 1.3, c("prior", "posterior"), lwd=3, col=c("red", "blue"))

## End(Not run)
```

---

mptable	*Calculate the marginal posterior probabilities of predictors being included in a quantile regression model.*

Description

This function extracts the marginal probability table produced by summary.qrssvs.

Usage

```
mptable(qrssvs)
```

Arguments

qrssvs   An object of class qrssvs. Typically this will be the gamma component of the list returned by SSVSquantreg.

Value

A table with the predictors listed together with their posterior marginal posterior probability of inclusion.

Author(s)

Craig Reed

See Also

[SSVSquantreg](#)

Examples

```
## Not run:
set.seed(1)
epsilon<-rnorm(100)
set.seed(2)
x<-matrix(rnorm(1000),100,10)
y<-x[,1]+x[,10]+epsilon
qrssvs<-SSVSquantreg(y~x)
mptable(qrssvs$gamma)

## End(Not run)
```

Nethvote	*Dutch Voting Behavior in 1989*

Description

Dutch Voting Behavior in 1989.

Format

A data frame with 1754 observations and 11 variables from the 1989 Dutch Parliamentary Election Study (Anker and Oppenhuis, 1993). Each observation is a survey respondent. These data are a subset of one of five multiply imputed datasets used in Quinn and Martin (2002). For more information see Quinn and Martin (2002).

vote   A factor giving the self-reported vote choice of each respondent. The levels are CDA (Christen Democratisch Appel), D66 (Democraten 66), Pvda (Partij van de Arbeid), and VVD (Volkspartij voor Vrijheid en Democratie).

distD66   A numeric variable giving the squared ideological distance between the respondent and the D66. Larger values indicate ideological dissimilarity between the respondent and the party.

distPvdA   A numeric variable giving the squared ideological distance between the respondent and the PvdA. Larger values indicate ideological dissimilarity between the respondent and the party.

distVVD   A numeric variable giving the squared ideological distance between the respondent and the VVD. Larger values indicate ideological dissimilarity between the respondent and the party.

distCDA   A numeric variable giving the squared ideological distance between the respondent and the CDA. Larger values indicate ideological dissimilarity between the respondent and the party.

relig   An indicator variable equal to 0 if the respondent is not religious and 1 if the respondent is religious.

class   Social class of respondent. 0 is the lowest social class, 4 is the highest social class.

income   Income of respondent. 0 is lowest and 6 is highest.

educ   Education of respondent. 0 is lowest and 4 is highest.

age   Age category of respondent. 0 is lowest and 12 is highest.

urban   Indicator variable equal to 0 if the respondent is not a resident of an urban area and 1 if the respondent is a resident of an urban area.

Source

H. Anker and E.V. Oppenhuis. 1993. "Dutch Parliamentary Election Study." (computer file). Dutch Electoral Research Foundation and Netherlands Central Bureau of Statistics, Amsterdam.

References

Kevin M. Quinn and Andrew D. Martin. 2002. "An Integrated Computational Model of Multiparty Electoral Competition." *Statistical Science*. 17: 405-419.

NoncenHypergeom   *The Noncentral Hypergeometric Distribution*

Description

Evaluates the density at a single point or all points, and generate random draws from the Noncentral Hypergeometric distribution.

Usage

```
dnoncenhypergeom(x = NA, n1, n2, m1, psi)

rnoncenhypergeom(n, n1, n2, m1, psi)
```

Arguments

x	The location to evaluate the density. If x is NA, then a matrix is returned with the density evaluated at all possible points.
n1	The size of group one.
n2	The size of group two.
m1	The observed number of positive outcomes (in both groups).
psi	Odds ratio.
n	The number of draws to make from the distribution.

Details

The Noncentral Hypergeometric is particularly useful for conditional inference for (2×2) tables. We use the parameterization and algorithms of Liao and Rosen (2001). The underlying R code is based on their published code. See their article for details of the parameterization.

Value

dnoncenhypergeom evaluates the density at point x, or a matrix with the first column containing the possible values of the random variable, and the second column containing the probabilities.

rnoncenhypergeom returns a list of n random draws from the distribution.

Source

J. G. Liao and Ori Rosen. 2001. "Fast and Stable Algorithms for Computing and Sampling From the Noncentral Hypergeometric Distribution." *The American Statistician.* 55: 366-369.

Examples

```
density <- dnoncenhypergeom(NA, 500, 500, 500, 6.0)
draws <- rnoncenhypergeom(10, 500, 500, 500, 6.0)
```

PErisk	*Political Economic Risk Data from 62 Countries in 1987*

Description

Political Economic Risk Data from 62 Countries in 1987.

Format

A data frame with 62 observations on the following 9 variables. All data points are from 1987. See Quinn (2004) for more details.

country a factor with levels `Argentina` through `Zimbabwe`

courts an ordered factor with levels `0 < 1.courts` is an indicator of whether the country in question is judged to have an independent judiciary. From Henisz (2002).

barb2 a numeric vector giving the natural log of the black market premium in each country. The black market premium is coded as the black market exchange rate (local currency per dollar) divided by the official exchange rate minus 1. From Marshall, Gurr, and Harff (2002).

prsexp2 an ordered factor with levels `0 < 1 < 2 < 3 < 4 < 5`, giving the lack of expropriation risk. From Marshall, Gurr, and Harff (2002).

prscorr2 an ordered factor with levels `0 < 1 < 2 < 3 < 4 < 5`, measuring the lack of corruption. From Marshall, Gurr, and Harff (2002).

gdpw2 a numeric vector giving the natural log of real GDP per worker in 1985 international prices. From Alvarez et al. (1999).

Source

Mike Alvarez, Jose Antonio Cheibub, Fernando Limongi, and Adam Przeworski. 1999. "ACLP Political and Economic Database."

Witold J. Henisz. 2002. "The Political Constraint Index (POLCON) Dataset."

Monty G. Marshall, Ted Robert Gurr, and Barbara Harff. 2002. "State Failure Task Force Problem Set."

References

Kevin M. Quinn. 2004. "Bayesian Factor Analysis for Mixed Ordinal and Continuous Response." *Political Analyis*. 12: 338-353.

---

plot.qrssvs *Plot output from quantile regression stochastic search variable selec-tion (QR-SSVS).*

Description

This function produces a Trellis plot of the predictors on the y-axis versus the marginal posterior probability of inclusion on the x-axis.

Usage

```
## S3 method for class 'qrssvs'
plot(x, ...)
```

Arguments

x   An object of class qrssvs. Typically this will be the gamma component of the
           list returned by SSVSquantreg.
...   Further arguments

Value

An object with class "trellis". The associated [update](#) and [print](#) methods are documented in the
"Lattice" package.

Author(s)

Craig Reed

References

Deepayan Sarkar. 2008. *lattice: Lattice Graphics.* R package version 0.17-17

See Also

[SSVSquantreg](#), [mptable](#), [Lattice](#) for a brief introduction to lattice displays and links to further
documentation.

Examples

```
## Not run:
set.seed(1)
epsilon<-rnorm(100)
set.seed(2)
x<-matrix(rnorm(1000),100,10)
y<-x[,1]+x[,10]+epsilon
qrssvs<-SSVSquantreg(y~x)
```

```
plot(qrssvs$gamma)
## Modify the graph by increasing the fontsize on the axes
qrssvsplot<-plot(qrssvs$gamma)
update(qrssvsplot, scales=list(cex=3))

## End(Not run)
```

---

plotChangepoint   *Posterior Density of Regime Change Plot*

Description

Plot the posterior density of regime change.

Usage

```
plotChangepoint(
  mcmcout,
  main = "Posterior Density of Regime Change Probabilities",
  xlab = "Time",
  ylab = "",
  verbose = FALSE,
  start = 1,
  overlay = FALSE
)
```

Arguments

mcmcout	The mcmc object containing the posterior density sample from a changepoint model. Note that this must have a prob.state attribute.
main	Title of the plot
xlab	Label for the x-axis.
ylab	Label for the y-axis.
verbose	If verbose=TRUE, expected changepoints are printed.
start	The time of the first observation to be shown in the time series plot.
overlay	If overlay=TRUE, the probability of each regime change is drawn separately, which will be useful to draw multiple plots in one screen. See the example in MCMCpoissonChange. Otherwise, multiple plots of regime change probabilities will be drawn.

See Also

[MCMCpoissonChange](), [MCMCbinaryChange]()

---

plotHDPChangepoint *Posterior Changepoint Probabilities from HDP-HMM*

Description

Plot the posterior density of regime change.

Usage

```
plotHDPChangepoint(
  mcmcout,
  main = "Posterior Changepoint Probabilities",
  xlab = "Time",
  ylab = "",
  start = 1
)
```

Arguments

mcmcout	The mcmc object containing the posterior density sample from a changepoint model. Note that this must be from a HDP-HMM sampler.
main	Title of the plot
xlab	Label for the x-axis.
ylab	Label for the y-axis.
start	The time of the first observation to be shown in the time series plot.

See Also

[HDPHMMpoisson](), [HDPHMMnegbin](), [HDPHSMMnegbin]()

---

plotState *Changepoint State Plot*

Description

Plot the posterior probability that each time point is in each state.

Usage

```
plotState(
  mcmcout,
  main = "Posterior Regime Probability",
  ylab = expression(paste("Pr(", S[t], "= k |", Y[t], ")")),
  legend.control = NULL,
  cex = 0.8,
  lwd = 1.2,
  start = 1
)
```

Arguments

mcmcout	The mcmc object containing the posterior density sample from a changepoint model. Note that this must have a prob.state attribute.
main	Title of the plot.
ylab	Label for the y-axis.
legend.control	Control the location of the legend. It is necessary to pass both the x and y locations; i.e., c(x,y).
cex	Control point size.
lwd	Line width parameter.
start	The time of the first observation to be shown in the time series plot.

See Also

[MCMCpoissonChange](), [MCMCbinaryChange]()

PostProbMod   *Calculate Posterior Probability of Model*

Description

This function takes an object of class BayesFactor and calculates the posterior probability that each model under study is correct given that one of the models under study is correct.

Usage

```
PostProbMod(BF, prior.probs = 1)
```

Arguments

BF	An object of class BayesFactor.
prior.probs	The prior probabilities that each model is correct. Can be either a scalar or array. Must be positive. If the sum of the prior probabilities is not equal to 1 prior.probs will be normalized so that it does sum to unity.

Value

An array holding the posterior probabilities that each model under study is correct given that one of the models under study is correct.

See Also

MCMCregress

Examples

```
## Not run:
data(birthwt)

post1 <- MCMCregress(bwt~age+lwt+as.factor(race) + smoke + ht,
                     data=birthwt, b0=c(2700, 0, 0, -500, -500,
                                           -500, -500),
                     B0=c(1e-6, .01, .01, 1.6e-5, 1.6e-5, 1.6e-5,
                          1.6e-5), c0=10, d0=4500000,
                     marginal.likelihood="Chib95", mcmc=10000)

post2 <- MCMCregress(bwt~age+lwt+as.factor(race) + smoke,
                     data=birthwt, b0=c(2700, 0, 0, -500, -500,
                                           -500),
                     B0=c(1e-6, .01, .01, 1.6e-5, 1.6e-5, 1.6e-5),
                     c0=10, d0=4500000,
                     marginal.likelihood="Chib95", mcmc=10000)

post3 <- MCMCregress(bwt~as.factor(race) + smoke + ht,
                     data=birthwt, b0=c(2700, -500, -500,
                                           -500, -500),
                     B0=c(1e-6, 1.6e-5, 1.6e-5, 1.6e-5,
                          1.6e-5), c0=10, d0=4500000,
                     marginal.likelihood="Chib95", mcmc=10000)

BF <- BayesFactor(post1, post2, post3)
mod.probs <- PostProbMod(BF)
print(mod.probs)

## End(Not run)
```

---

procrustes   *Procrustes Transformation*

Description

This function performs a Procrustes transformation on a matrix X to minimize the squared distance between X and another matrix Xstar.

Usage

```
procrustes(X, Xstar, translation = FALSE, dilation = FALSE)
```

Arguments

X	The matrix to be transformed.
Xstar	The target matrix.
translation	logical value indicating whether X should be translated.
dilation	logical value indicating whether X should be dilated.

Details

R, tt, and s are chosen so that:

$$sXR + 1tt' \approx X^*$$

X.new is given by:

$$X_{new} = sXR + 1tt'$$

Value

A list containing: X.new the matrix that is the Procrustes transformed version of X, R the rotation matrix, tt the translation vector, and s the scale factor.

References

Borg and Groenen. 1997. *Modern Multidimensional Scaling*. New York: Springer. pp. 340-342.

See Also

MCMCirtKd

---

read.Scythe	*Read a Matrix from a File written by Scythe*

Description

This function reads a matrix from an ASCII file in the form produced by the Scythe Statistical Library. Scythe output files contain the number of rows and columns in the first row, followed by the data.

Usage

```
read.Scythe(infile = NA)
```

Arguments

infile   The file to be read. This can include path information.

Value

A matrix containing the data stored in the read file.

References

Daniel Pemstein, Kevin M. Quinn, and Andrew D. Martin. 2007. *Scythe Statistical Library 1.0.* <http://scythe.lsa.umich.edu>.

See Also

[write.Scythe](#)

Examples

```
   ## Not run:
   mymatrix <- read.Scythe("myfile.txt")

## End(Not run)
```

Rehnquist   *U.S. Supreme Court Vote Matrix, Rehnquist Court (1994-2004)*

Description

This dataframe contains a matrix of votes cast by U.S. Supreme Court justices by all cases in the 1994-2004 terms.

Format

The dataframe has contains data for justices Rehnquist, Stevens, O'Connor, Scalia, Kennedy, Souter, Thomas, Ginsburg, and Breyer for the 1994-2004 terms of the U.S. Supreme Court. The dataframe also contains the term of the case, and a time variable that counts from term 1 to 11. The votes are coded liberal (1) and conservative (0) using the protocol of Spaeth (2003). The unit of analysis is the case citation (ANALU=0). We are concerned with formally decided cases issued with written opinions, after full oral argument and cases decided by an equally divided vote (DECTYPE=1,5,6,7).

Source

Harold J. Spaeth. 2005. *Original United States Supreme Court Database: 1953-2004 Terms.*

Senate   *106th U.S. Senate Roll Call Vote Matrix*

Description

This dataframe contains a matrix of votes cast by U.S. Senators in the 106th Congress.

Format

The dataframe contains roll call data for all Senators in the 106th Senate. The first column (id) is the ICPSR member ID number, the second column (statecode) is the ICPSR state code, the third column (party) is the member's state name, and the fourth column (member) is the member's name. This is followed by all roll call votes (including unanimous ones) in the 106th. Nay votes are coded 0, yea votes are coded 1, and NAs are missing votes.

Source

Keith Poole. 2005. *106th Roll Call Vote Data*.

SSVSquantreg   *Stochastic search variable selection for quantile regression*

Description

This function uses stochastic search to select promising regression models at a fixed quantile τ. Indicator variables γ are used to represent whether a predictor is included in the model or not. The user supplies the data and the prior distribution on the model size. A list is returned containing the posterior sample of γ and the associated regression parameters β.

Usage

```
SSVSquantreg(
  formula,
  data = NULL,
  tau = 0.5,
  include = NULL,
  burnin = 1000,
  mcmc = 10000,
  thin = 1,
  verbose = 0,
  seed = sample(1:1e+06, 1),
  pi0a0 = 1,
  pi0b0 = 1,
  ...
)
```

Arguments

`formula`	Model formula.
`data`	Data frame.
`tau`	The quantile of interest. Must be between 0 and 1. The default value of 0.5 corresponds to median regression model selection.
`include`	The predictor(s) that should definitely appear in the model. Can be specified by name, or their position in the formula (taking into account the intercept).
`burnin`	The number of burn-in iterations for the sampler.
`mcmc`	The number of MCMC iterations after burnin.
`thin`	The thinning interval used in the simulation. The number of MCMC iterations must be divisible by this value.
`verbose`	A switch which determines whether or not the progress of the sampler is printed to the screen. If `verbose` is greater than 0 the iteration number, the most recently sampled values of γ and the associated values of β are printed to the screen every `verbose`th iteration.
`seed`	The seed for the random number generator. If NA, the Mersenne Twister generator is used with default seed 12345; if an integer is passed it is used to seed the Mersenne twister. The default value for this argument is a random integer between 1 and 1,000,000. This default value ensures that if the function is used again with a different value of τ, it is extremely unlikely that the seed will be identical. The user can also pass a list of length two to use the L'Ecuyer random number generator, which is suitable for parallel computation. The first element of the list is the L'Ecuyer seed, which is a vector of length six or NA (if NA a default seed of `rep(12345,6)` is used). The second element of list is a positive substream number. See the MCMCpack specification for more details.
`pi0a0, pi0b0`	Hyperparameters of the beta prior on π_0, the prior probability of including a predictor. Default values of (1,1) are equivalent to a uniform distribution.
`...`	Further arguments

Details

`SSVSquantreg` implements stochastic search variable selection over a set of potential predictors to obtain promising models. The models considered take the following form:

$$Q_\tau(y_i|x_{i\gamma}) = x'_{i\gamma}\beta_\gamma,$$

where $Q_\tau(y_i|x_{i\gamma})$ denotes the conditional τth quantile of y_i given $x_{i\gamma}$, $x_{i\gamma}$ denotes x_i with those predictors x_{ij} for which $\gamma_j = 0$ removed and β_γ denotes the model specific regression parameters.

The likelihood is formed based on the assumption of independent asymmetric Laplace distributions on the y_i with skewness parameter τ and location parameters $x'_{i\gamma}\beta_\gamma$. This assumption ensures that the likelihood function is maximised by the τth conditional quantile of the response variable.

The prior on each β_j is

$$(1-\gamma_j)\delta_0 + \gamma_j\text{Cauchy}(0,1),$$

where δ_0 denotes a degenerate distribution with all mass at 0. A standard Cauchy distribution is chosen conditional on $\gamma_j = 1$. This allows for a wider range of nonzero values of β_j than a standard Normal distribution, improving the robustness of the method. Each of the indicator variables γ_j is independently assigned a Bernoulli prior, with prior probability of inclusion π_0. This in turn is assigned a beta distribution, resulting in a beta-binomial prior on the model size. The user can supply the hyperparameters for the beta distribution. Starting values are randomly generated from the prior distribution.

It is recommended to standardise any non-binary predictors in order to compare these predictors on the same scale. This can be achieved using the `scale` function.

If it is certain that a predictor should be included, all predictors specified are brought to the first positions for computational convenience. The regression parameters associated with these predictors are given independent improper priors. Users may notice a small speed advantage if they specify the predictors that they feel certain should appear in the model, particularly for large models with a large number of observations.

Value

A list containing:

gamma	The posterior sample of γ. This has associated summary and plot methods.
beta	The posterior sample of the associated regression parameters β. This can be analysed with functions from the coda package.

Author(s)

Craig Reed

References

Craig Reed, David B. Dunson and Keming Yu. 2010. "Bayesian Variable Selection for Quantile Regression" Technical Report.

Daniel Pemstein, Kevin M. Quinn, and Andrew D. Martin. 2007. *Scythe Statistical Library 1.2.* http://scythe.lsa.umich.edu.

Keming Yu and Jin Zhang. 2005. "A Three Parameter Asymmetric Laplace Distribution and it's extensions." *Communications in Statistics - Theory and Methods*, 34, 1867-1879.

Martyn Plummer, Nicky Best, Kate Cowles, and Karen Vines. 2006. "Output Analysis and Diagnostics for MCMC (CODA)", *R News*. 6(1): 7-11. https://CRAN.R-project.org/doc/Rnews/Rnews_2006-1.pdf.

See Also

MCMCquantreg, summary.qrssvs, plot.qrssvs, mptable, topmodels, scale, rq

Examples

```
## Not run:
```

```
set.seed(1)
epsilon<-rnorm(100)
set.seed(2)
x<-matrix(rnorm(1000),100,10)
y<-x[,1]+x[,10]+epsilon
qrssvs<-SSVSquantreg(y~x)
model.50pc<-SSVSquantreg(y~x)
model.90pc<-SSVSquantreg(y~x,tau=0.9)
summary(model.50pc) ## Intercept not in median probability model
summary(model.90pc) ## Intercept appears in median probability model

## End(Not run)
```

---

summaryqrssvs	*Summarising the results of quantile regression stochastic search variable selection (QR-SSVS).*

Description

This function produces a table of predictors and their associated marginal posterior probability of inclusion. It also returns the median probability model (see the details section).

Usage

```
## S3 method for class 'qrssvs'
summary(object, ...)
```

Arguments

object   An object of class qrssvs. Typically this will be the gamma component of the
           list returned by SSVSquantreg.

...   Further arguments.

Details

The median probability model is defined to be the model that contains any predictor with marginal posterior probability greater than or equal to 0.5. If the goal is to select a single model e.g. for prediction, Barbieri and Berger (2004) recommend the median probability model. In some cases, this will coincide with the maximum probability model.

Author(s)

Craig Reed

References

Maria M. Barbieri, and James O. Berger (2004). "Optimal predictive model selection". *Annals of Statistics*, 32, 870-897.

See Also

SSVSquantreg, mptable, topmodels

Examples

```
## Not run:
set.seed(1)
epsilon<-rnorm(100)
set.seed(2)
x<-matrix(rnorm(1000),100,10)
y<-x[,1]+x[,10]+epsilon
qrssvs<-SSVSquantreg(y~x)
summary(qrssvs$gamma)

## End(Not run)
```

SupremeCourt   *U.S. Supreme Court Vote Matrix*

Description

This dataframe contains a matrix votes cast by U.S. Supreme Court justices in all cases in the 2000 term.

Format

The dataframe has contains data for justices Rehnquist, Stevens, O'Connor, Scalia, Kennedy, Souter, Thomas, Ginsburg, and Breyer for the 2000 term of the U.S. Supreme Court. It contains data from 43 non-unanimous cases. The votes are coded liberal (1) and conservative (0) using the protocol of Spaeth (2003). The unit of analysis is the case citation (ANALU=0). We are concerned with formally decided cases issued with written opinions, after full oral argument and cases decided by an equally divided vote (DECTYPE=1,5,6,7).

Source

Harold J. Spaeth. 2005. *Original United States Supreme Court Database: 1953-2004 Terms.* http://supremecourtdatabase.org.

---

testpanelGroupBreak   *A Test for the Group-level Break using a Multivariate Linear Regression Model with Breaks*

Description

testpanelGroupBreak fits a multivariate linear regression model with parametric breaks using panel residuals to test the existence of group-level breaks in panel residuals. The details are discussed in Park (2011).

Usage

```
testpanelGroupBreak(
  subject.id,
  time.id,
  resid,
  m = 1,
  mcmc = 1000,
  burnin = 1000,
  thin = 1,
  verbose = 0,
  b0,
  B0,
  c0,
  d0,
  a = NULL,
  b = NULL,
  seed = NA,
  marginal.likelihood = c("none", "Chib95"),
  ...
)
```

Arguments

subject.id	A numeric vector indicating the group number. It should start from 1.
time.id	A numeric vector indicating the time unit. It should start from 1.
resid	A vector of panel residuals
m	The number of changepoints.
mcmc	The number of MCMC iterations after burn-in.
burnin	The number of burn-in iterations for the sampler.
thin	The thinning interval used in the simulation. The number of MCMC iterations must be divisible by this value.
verbose	A switch which determines whether or not the progress of the sampler is printed to the screen. If verbose is greater than 0, the iteration number and the posterior density samples are printed to the screen every verboseth iteration.

b0	The prior mean of the residual mean.
B0	The prior precision of the residual variance
c0	$c_0/2$ is the shape parameter for the inverse Gamma prior on σ^2. The amount of information in the inverse Gamma prior is something like that from c_0 pseudo-observations.
d0	$d_0/2$ is the scale parameter for the inverse Gamma prior on σ^2.
a	a is the shape1 beta prior for transition probabilities. By default, the expected duration is computed and corresponding a and b values are assigned. The expected duration is the sample period divided by the number of states.
b	b is the shape2 beta prior for transition probabilities. By default, the expected duration is computed and corresponding a and b values are assigned. The expected duration is the sample period divided by the number of states.
seed	The seed for the random number generator. If NA, current R system seed is used.
marginal.likelihood	How should the marginal likelihood be calculated? Options are: none in which case the marginal likelihood will not be calculated and Chib95 in which case the method of Chib (1995) is used.
...	further arguments to be passed

Details

testpanelGroupBreak fits a multivariate linear regression model with parametric breaks using panel residuals to detect the existence of system-level breaks in unobserved factors as discussed in Park (2011).

The model takes the following form:

$$e_i \sim \mathcal{N}(\beta_m, \sigma_m^2 I) \;\; m = 1, \ldots, M$$

We assume standard, semi-conjugate priors:

$$\beta \sim \mathcal{N}(b0, B0)$$

And:

$$\sigma^{-2} \sim \mathcal{G}amma(c_0/2, d_0/2)$$

Where β and σ^{-2} are assumed *a priori* independent.

And:

$$p_{mm} \sim \mathcal{B}eta(a, b), \;\; m = 1, \ldots, M$$

Where M is the number of states.

Value

An mcmc object that contains the posterior sample. This object can be summarized by functions provided by the coda package. The object contains an attribute prob.state storage matrix that contains the probability of $state_i$ for each period, and the log-marginal likelihood of the model (logmarglike).

References

Jong Hee Park, 2012. "Unified Method for Dynamic and Cross-Sectional Heterogeneity: Introducing Hidden Markov Panel Models." *American Journal of Political Science*.56: 1040-1054. <doi: 10.1111/j.1540-5907.2012.00590.x>

Siddhartha Chib. 1998. "Estimation and comparison of multiple change-point models." *Journal of Econometrics*. 86: 221-241. <doi: 10.1080/01621459.1995.10476635>

Examples

```
## Not run:
  ## data generating
 set.seed(1977)
 Q <- 3
 true.beta1   <-  c(1, 1, 1) ; true.beta2   <-  c(1, -1, -1)
 true.sigma2 <-  c(1, 3); true.D1 <- diag(.5, Q); true.D2 <- diag(2.5, Q)
 N=20; T=100;
 NT <- N*T
 x1 <- rnorm(NT)
 x2 <- runif(NT, 5, 10)
 X <- cbind(1, x1, x2);   W <- X;   y <- rep(NA, NT)

 ## true break numbers are one and at the center
 break.point = rep(T/2, N); break.sigma=c(rep(1, N));
 break.list <- rep(1, N)
 id  <-  rep(1:N, each=NT/N)
 K <-  ncol(X);
 ruler <- c(1:T)

 ## compute the weight for the break
 W.mat <- matrix(NA, T, N)
 for (i in 1:N){
   W.mat[, i] <- pnorm((ruler-break.point[i])/break.sigma[i])
 }
 Weight <- as.vector(W.mat)

 ## data generating by weighting two means and variances
 j = 1
 for (i in 1:N){
   Xi <- X[j:(j+T-1), ]
   Wi <- W[j:(j+T-1), ]
   true.V1 <- true.sigma2[1]*diag(T) + Wi%*%true.D1%*%t(Wi)
   true.V2 <- true.sigma2[2]*diag(T) + Wi%*%true.D2%*%t(Wi)
   true.mean1 <- Xi%*%true.beta1
```

```
    true.mean2 <- Xi%*%true.beta2
    weight <- Weight[j:(j+T-1)]
    y[j:(j+T-1)] <- (1-weight)*true.mean1 + (1-weight)*chol(true.V1)%*%rnorm(T) +
      weight*true.mean2 + weight*chol(true.V2)%*%rnorm(T)
    j <- j + T
  }
  ## model fitting
  subject.id <- c(rep(1:N, each=T))
  time.id <- c(rep(1:T, N))


  resid <- rstandard(lm(y ~X-1 + as.factor(subject.id)))
  G <- 100
  out0 <- testpanelGroupBreak(subject.id, time.id, resid, m=0,
          mcmc=G, burnin=G, thin=1, verbose=G,
          b0=0, B0=1/100, c0=2, d0=2, marginal.likelihood = "Chib95")
  out1 <- testpanelGroupBreak(subject.id, time.id, resid, m=1,
          mcmc=G, burnin=G, thin=1, verbose=G,
          b0=0, B0=1/100, c0=2, d0=2, marginal.likelihood = "Chib95")
  out2 <- testpanelGroupBreak(subject.id, time.id, resid, m=2,
          mcmc=G, burnin=G, thin=1, verbose=G,
          b0=0, B0=1/100, c0=2, d0=2, marginal.likelihood = "Chib95")
  out3 <- testpanelGroupBreak(subject.id, time.id, resid, m=3,
          mcmc=G, burnin=G, thin=1, verbose=G,
          b0=0, B0=1/100, c0=2, d0=2, marginal.likelihood = "Chib95")

  ## Note that the code is for a hypothesis test of no break in panel residuals.
 ## When breaks exist, the estimated number of break in the mean and variance of panel residuals
  ## tends to be larger than the number of break in the data generating process.
  ## This is due to the difference in parameter space, not an error of the code.
  BayesFactor(out0, out1, out2, out3)

  ## In order to identify the number of breaks in panel parameters,
  ## use HMMpanelRE() instead.


 ## End(Not run)
```

---

testpanelSubjectBreak *A Test for the Subject-level Break using a Unitivariate Linear Regression Model with Breaks*

Description

testpanelSubjectBreak fits a unitivariate linear regression model with parametric breaks using panel residuals to test the existence of subject-level breaks in panel residuals. The details are discussed in Park (2011).

Usage

```
testpanelSubjectBreak(
  subject.id,
  time.id,
  resid,
  max.break = 2,
  minimum = 10,
  mcmc = 1000,
  burnin = 1000,
  thin = 1,
  verbose = 0,
  b0,
  B0,
  c0,
  d0,
  a = NULL,
  b = NULL,
  seed = NA,
  Time = NULL,
  ps.out = FALSE,
  ...
)
```

Arguments

subject.id	A numeric vector indicating the group number. It should start from 1.
time.id	A numeric vector indicating the time unit. It should start from 1.
resid	A vector of panel residuals.
max.break	An upper bound of break numbers for the test.
minimum	A minimum length of time series for the test. The test will skip a subject with a time series shorter than this.
mcmc	The number of MCMC iterations after burn-in.
burnin	The number of burn-in iterations for the sampler.
thin	The thinning interval used in the simulation. The number of MCMC iterations must be divisible by this value.
verbose	A switch which determines whether or not the progress of the sampler is printed to the screen. If verbose is greater than 0, the iteration number and the posterior density samples are printed to the screen every verboseth iteration.
b0	The prior mean of the residual mean.
B0	The prior precision of the residual variance
c0	$c_0/2$ is the shape parameter for the inverse Gamma prior on σ^2. The amount of information in the inverse Gamma prior is something like that from c_0 pseudo-observations.
d0	$d_0/2$ is the scale parameter for the inverse Gamma prior on σ^2.

a	a is the shape1 beta prior for transition probabilities. By default, the expected duration is computed and corresponding a and b values are assigned. The expected duration is the sample period divided by the number of states.
b	b is the shape2 beta prior for transition probabilities. By default, the expected duration is computed and corresponding a and b values are assigned. The expected duration is the sample period divided by the number of states.
seed	The seed for the random number generator. If NA, current R system seed is used.
Time	Times of the observations. This will be used to find the time of the first observations in panel residuals.
ps.out	If ps.out == TRUE, state probabilities are exported. If the number of panel subjects is huge, users can turn it off to save memory.
...	further arguments to be passed

Details

`testpanelSubjectBreak` fits a univariate linear regression model for subject-level residuals from a panel model. The details are discussed in Park (2011).

The model takes the following form:

$$e_{it} = \alpha_{im} + \varepsilon_{it} \ \ m = 1, \ldots, M$$

The errors are assumed to be time-varying at the subject level:

$$\varepsilon_{it} \sim \mathcal{N}(0, \sigma_{im}^2)$$

We assume standard, semi-conjugate priors:

$$\beta \sim \mathcal{N}(b_0, B_0^{-1})$$

And:

$$\sigma^{-2} \sim \mathcal{G}amma(c_0/2, d_0/2)$$

Where β and σ^{-2} are assumed *a priori* independent.

And:

$$p_{mm} \sim \mathcal{B}eta(a, b), \ \ m = 1, \ldots, M$$

Where M is the number of states.

OLS estimates are used for starting values.

Value

The returned object is a matrix containing log marginal likelihoods for all HMMs. The dimension of the returned object is the number of panel subjects by max.break + 1. If psout == TRUE, the returned object has an array attribute `psout` containing state probabilities for all HMMs.

References

Jong Hee Park, 2012. "Unified Method for Dynamic and Cross-Sectional Heterogeneity: Introducing Hidden Markov Panel Models." *American Journal of Political Science*.56: 1040-1054. <doi: 10.1111/j.1540-5907.2012.00590.x>

Siddhartha Chib. 1998. "Estimation and comparison of multiple change-point models." *Journal of Econometrics*. 86: 221-241. <doi: 10.1080/01621459.1995.10476635>

Examples

```
## Not run:
  set.seed(1974)
  N <- 30
  T <- 80
  NT <- N*T

  ## true parameter values
  true.beta <- c(1, 1)
  true.sigma <- 3
  x1 <- rnorm(NT)
  x2 <- runif(NT, 2, 4)

  ## group-specific breaks
  break.point = rep(T/2, N); break.sigma=c(rep(1, N));
  break.list <- rep(1, N)

  X <- as.matrix(cbind(x1, x2), NT, );
  y <- rep(NA, NT)
  id  <-  rep(1:N, each=NT/N)
  K <-  ncol(X);
  true.beta <- as.matrix(true.beta, K, 1)

  ## compute the break probability
  ruler <- c(1:T)
  W.mat <- matrix(NA, T, N)
  for (i in 1:N){
    W.mat[, i] <- pnorm((ruler-break.point[i])/break.sigma[i])
  }
  Weight <- as.vector(W.mat)

  ## draw time-varying individual effects and sample y
  j = 1
  true.sigma.alpha <- 30
  true.alpha1 <- true.alpha2 <- rep(NA, N)
  for (i in 1:N){
    Xi <- X[j:(j+T-1), ]
    true.mean <- Xi  %*% true.beta
    weight <- Weight[j:(j+T-1)]
    true.alpha1[i] <- rnorm(1, 0, true.sigma.alpha)
    true.alpha2[i] <- -1*true.alpha1[i]
    y[j:(j+T-1)] <- ((1-weight)*true.mean + (1-weight)*rnorm(T, 0, true.sigma) +
```

```
        (1-weight)*true.alpha1[i]) +
        (weight*true.mean + weight*rnorm(T, 0, true.sigma) + weight*true.alpha2[i])
   j <- j + T
 }

 ## extract the standardized residuals from the OLS with fixed-effects
 FEols <- lm(y ~ X + as.factor(id) -1 )
 resid.all <- rstandard(FEols)
 time.id <- rep(1:80, N)

 ## model fitting
 G <- 1000
 BF <- testpanelSubjectBreak(subject.id=id, time.id=time.id,
       resid= resid.all, max.break=3, minimum = 10,
       mcmc=G, burnin = G, thin=1, verbose=G,
       b0=0, B0=1/100, c0=2, d0=2, Time = time.id)

 ## estimated break numbers
 ## thresho
 estimated.breaks <- make.breaklist(BF, threshold=3)

 ## print all posterior model probabilities
 print(attr(BF, "model.prob"))

## End(Not run)
```

---

  tomogplot   *Tomography Plot*

Description

tomogplot is used to produce a tomography plot (see King, 1997) for a series of partially observed
2 x 2 contingency tables.

Usage

```
tomogplot(
  r0,
  r1,
  c0,
  c1,
  xlab = "fraction of r0 in c0 (p0)",
  ylab = "fraction of r1 in c0 (p1)",
  bgcol = "white",
  ...
)
```

Arguments

r0	An $(ntables \times 1)$ vector of row sums from row 0.
r1	An $(ntables \times 1)$ vector of row sums from row 1.
c0	An $(ntables \times 1)$ vector of column sums from column 0.
c1	An $(ntables \times 1)$ vector of column sums from column 1.
xlab	The x axis label for the plot.
ylab	The y axis label for the plot.
bgcol	The background color for the plot.
...	further arguments to be passed

Details

Consider the following partially observed 2 by 2 contingency table:

	$\mid Y = 0$	$\mid Y = 1$	$\mid$
$X = 0$	$\mid Y_0$	$\mid$	$\mid r_0$
$X = 1$	$\mid Y_1$	$\mid$	$\mid r_1$
	$\mid c_0$	$\mid c_1$	$\mid N$

where r_0, r_1, c_0, c_1, and N are non-negative integers that are observed. The interior cell entries are not observed. It is assumed that $Y_0 | r_0 \sim \mathcal{B}inomial(r_0, p_0)$ and $Y_1 | r_1 \sim \mathcal{B}inomial(r_1, p_1)$.

This function plots the bounds on the maximum likelihood estimatess for (p0, p1).

References

Gary King, 1997. *A Solution to the Ecological Inference Problem*. Princeton: Princeton University Press.

Jonathan C. Wakefield. 2004. "Ecological Inference for 2 x 2 Tables." *Journal of the Royal Statistical Society, Series A*. 167(3): 385445.

See Also

MCMChierEI, MCMCdynamicEI, dtomogplot

Examples

```
r0 <- rpois(100, 500)
r1 <- rpois(100, 200)
c0 <- rpois(100, 100)
c1 <- (r0 + r1) - c0
tomogplot(r0, r1, c0, c1)
```

topmodels	*Shows an ordered list of the most frequently visited models sampled during quantile regression stochastic search variable selection (QR-SSVS).*

Description

Given output from quantile regression stochastic search variable selection, this function returns a table of the 'best' models together with their associated empirical posterior probability.

Usage

```
topmodels(qrssvs, nmodels = 5, abbreviate = FALSE, minlength = 3)
```

Arguments

qrssvs	An object of class qrssvs. Typically this will be the gamma component of the list returned by SSVSquantreg.
nmodels	The number of models to tabulate.
abbreviate	Logical: should the names of the predictors be abbreviated?
minlength	If abbreviate is set to TRUE, the minimum length of the abbreviations.

Value

A table with the models and their associated posterior probability. The models are arranged in descending order of probability.

Author(s)

Craig Reed

See Also

SSVSquantreg

Examples

```
## Not run:
set.seed(1)
epsilon<-rnorm(100)
set.seed(2)
x<-matrix(rnorm(1000),100,10)
y<-x[,1]+x[,10]+epsilon
qrssvs<-SSVSquantreg(y~x)
topmodels(qrssvs$gamma)

## End(Not run)
```

---

vech   *Extract Lower Triangular Elements from a Symmetric Matrix*

Description

This function takes a symmetric matrix and extracts a list of all lower triangular elements.

Usage

```
vech(x)
```

Arguments

x   A symmetric matrix.

Details

This function checks to make sure the matrix is square, but it does not check for symmetry (it just pulls the lower triangular elements). The elements are stored in column major order. The original matrix can be restored using the xpnd command.

Value

A list of the lower triangular elements.

See Also

xpnd

Examples

```
symmat <- matrix(c(1,2,3,4,2,4,5,6,3,5,7,8,4,6,8,9),4,4)
vech(symmat)
```

Wishart   *The Wishart Distribution*

Description

Density function and random generation from the Wishart distribution.

Usage

```
rwish(v, S)

dwish(W, v, S)
```

Arguments

v	Degrees of freedom (scalar).
S	Inverse scale matrix $(p \times p)$.
W	Positive definite matrix W $(p \times p)$.

Details

The mean of a Wishart random variable with v degrees of freedom and inverse scale matrix S is vS.

Value

dwish evaluates the density at positive definite matrix W. rwish generates one random draw from the distribution.

Examples

```
density <- dwish(matrix(c(2,-.3,-.3,4),2,2), 3, matrix(c(1,.3,.3,1),2,2))
draw <- rwish(3, matrix(c(1,.3,.3,1),2,2))
```

---

write.Scythe	*Write a Matrix to a File to be Read by Scythe*

Description

This function writes a matrix to an ASCII file that can be read by the Sycthe Statistical Library. Scythe requires that input files contain the number of rows and columns in the first row, followed by the data.

Usage

```
write.Scythe(outmatrix, outfile = NA, overwrite = FALSE)
```

Arguments

outmatrix	The matrix to be written to a file.
outfile	The file to be written. This can include path information.
overwrite	A logical that determines whether an existing file should be over-written. By default, it protects the user from over-writing existing files.

Value

A zero if the file is properly written.

References

Daniel Pemstein, Kevin M. Quinn, and Andrew D. Martin. 2007. *Scythe Statistical Library 1.0.*
http://scythe.lsa.umich.edu.

See Also

write.Scythe

Examples

```
## Not run:
write.Scythe(mymatrix, file.path(tempdir(), "myfile.txt"))

## End(Not run)
```

---

xpnd   *Expand a Vector into a Symmetric Matrix*

Description

This function takes a vector of appropriate length (typically created using vech) and creates a symmetric matrix.

Usage

```
xpnd(x, nrow = NULL)
```

Arguments

x   A list of elements to expand into symmetric matrix.

nrow   The number of rows (and columns) in the returned matrix. Look into the details.

Details

This function is particularly useful when dealing with variance covariance matrices. Note that R stores matrices in column major order, and that the items in x will be recycled to fill the matrix if need be.

The number of rows can be specified or automatically computed from the number of elements in a given object via $(-1 + \sqrt{(1 + 8 * length(x))})/2$.

Value

An $(nrows \times nrows)$ symmetric matrix.

See Also

vech

Examples

```
xpnd(c(1,2,3,4,4,5,6,7,8,9),4)
xpnd(c(1,2,3,4,4,5,6,7,8,9))
```

Index

207