Package ‘mombf’

November 3, 2022

Type Package

Version 3.2.0

Date 2022-11-03

Title Bayesian Model Selection and Averaging for Non-Local and Local Priors

Author David Rossell, John D. Cook, Donatello Telesca, P. Roebuck, Oriol Abril, Miquel Torrens

Maintainer David Rossell `<rosselldavid@gmail.com>`

Depends R (>= 2.14.0), methods, mvtnorm, ncvreg, mgcv

Imports Rcpp (>= 0.12.16), glmnet, mclust, sgd, survival, pracma

Suggests parallel, testthat, patrick

LinkingTo Rcpp, RcppArmadillo

Description Bayesian model selection and averaging for regression and mixtures for non-local and selected local priors.

License GPL (>= 2) | file LICENSE

URL `https://github.com/davidrusi/mombf`

BugReports `https://github.com/davidrusi/mombf/issues`

LazyLoad yes

Collate AllClasses.R AllGenerics.R alapl.R bms_ortho.R cil.R cox.R derivatives_nlps.R distribs.R dmom.R gam.R greedyGLM.R infocriteria.R initParameters.R msPriorSpec.R modelsearch.R modelSelection.R modelSelectionGLM.R mombf.R normaliwish.R normmix.R nlpMarginal.R postMode.R rmom.R RcppExports.R testfunction.R

NeedsCompilation yes

Repository CRAN

Date/Publication 2022-11-03 21:50:02 UTC

R topics documented:

`bbPrior` *Priors on model space for variable selection problems*

Description

`unifPrior` implements a uniform prior (equal a priori probability for all models). `binomPrior` implements a Binomial prior. `bbPrior` implements a Beta-Binomial prior.

Usage

```
unifPrior(sel, logscale=TRUE, groups=1:length(sel),
constraints=lapply(1:length(unique(groups)), function(z) integer(0)))

binomPrior(sel, prob=.5, logscale=TRUE, probconstr=prob, groups=1:length(sel),
constraints=lapply(1:length(unique(groups)), function(z) integer(0)))

bbPrior(sel, alpha=1, beta=1, logscale=TRUE, alphaconstr=alpha,
betaconstr=beta, groups=1:length(sel),
constraints=lapply(1:length(unique(groups)), function(z) integer(0)))
```

Arguments

`sel`	Logical vector indicating which variables are included in the model
`logscale`	Set to `TRUE` to return the log-prior probability.
`groups`	Group that each variable belongs to (e.g. dummy indicators for categorical variables with >2 categories). The idea is that all variables in a group are jointly added/removed from the model. By default all variables are assumed to be in separate groups
`constraints`	List with length equal to the number of groups (distinct elements in `groups`). Element j in the list should indicate any hierarchical constraints on the group, for instance constraints[[3]]==c(1,2) indicates that group 3 can only be included in the model if groups 1 and 2 are also in the model. This can be used to enforce that an interaction can only be in the model if the main effects are also in the model.
`prob`	Success probability for the Binomial prior
`probconstr`	Success probability for the Binomial prior for groups that are subject to constraints
`alpha`	First parameter of the Beta-Binomial prior, which is equivalent to specifying a Beta(alpha,beta) prior on `prob`.
`beta`	First parameter of the Beta-Binomial prior, which is equivalent to specifying a Beta(alpha,beta) prior on `prob`.
`alphaconstr`	Same as alpha for the groups that are subject to constraints
`betaconstr`	Same as beta for the groups that are subject to constraints

Value

Prior probability of the specified model

Author(s)

David Rossell

Examples

```
library(mombf)
sel <- c(TRUE,TRUE,FALSE,FALSE)
unifPrior(sel,logscale=FALSE)
binomPrior(sel,prob=.5,logscale=FALSE)
bbPrior(sel,alpha=1,beta=1,logscale=FALSE)
```

`bfnormmix` *Number of Normal mixture components under Normal-IW and Non-local priors*

Description

Posterior sampling and Bayesian model selection to choose the number of components k in multivariate Normal mixtures.

`bfnormmix` computes posterior probabilities under non-local MOM-IW-Dir(q) priors, and also for local Normal-IW-Dir(q.niw) priors. It also computes posterior probabilities on cluster occupancy and posterior samples on the model parameters for several k.

Usage

```
bfnormmix(x, k=1:2, mu0=rep(0,ncol(x)), g, nu0, S0, q=3, q.niw=1,
B=10^4, burnin= round(B/10), logscale=TRUE, returndraws=TRUE, verbose=TRUE)
```

Arguments

`x`	n x p input data matrix
`k`	Number of components
`mu0`	Prior on mu[j] is N(mu0,g Sigma[j])
`g`	Prior on mu[j] is N(mu0,g Sigma[j]). This is a critical MOM-IW prior parameter that specifies the separation between components deemed practically relevant. It defaults to assigning 0.95 prior probability to any pair of mu's giving a bimodal mixture, see details
`S0`	Prior on Sigma[j] is IW(Sigma_j; nu0, S0)
`nu0`	Prior on Sigma[j] is IW(Sigma_j; nu0, S0)
`q`	Prior parameter in MOM-IW-Dir(q) prior
`q.niw`	Prior parameter in Normal-IW-Dir(q.niw) prior
`B`	Number of MCMC iterations
`burnin`	Number of burn-in iterations
`logscale`	If set to TRUE then log-Bayes factors are returned
`returndraws`	If set to `TRUE` the MCMC posterior draws under the Normal-IW-Dir prior are returned for all `k`
`verbose`	Set to `TRUE` to print iteration progress

Details

The likelihood is

p(x[i,] | mu,Sigma,eta)= sum_j eta_j N(x[i,]; mu_j,Sigma_j)

The Normal-IW-Dir prior is

Dir(eta; q.niw) prod_j N(mu_j; mu0, g Sigma) IW(Sigma_j; nu0, S0)

The MOM-IW-Dir prior is

d(mu,Sigma) Dir(eta; q) prod_j N(mu_j; mu0, g Sigma) IW(Sigma_j; nu0, S0)

where

d(mu,Sigma)= [prod_j<l (mu_j-mu_l)' A (mu_j-mu_l)]

and A is the average of Sigma_1^-1,...,Sigma_k^-1. Note that one must have q>1 for the MOM-IW-Dir to define a non-local prior.

By default the prior parameter g is set such that

P((mu[j]-mu[l])' A (mu[j]-mu[l]) < 4)= 0.05.

The reasonale when Sigma[j]=Sigma[l] and eta[j]=eta[l] then (mu[j]-mu[l])' A (mu[j]-mu[l])>4 corresponds to a bimodal density. That is, the default g focuses 0.95 prior prob on a degree of separation between components giving rise to a bimodal mixture density.

`bfnormmix` computes posterior model probabilities under the MOM-IW-Dir and Normal-IW-Dir priors using MCMC output. As described in Fuquene, Steel and Rossell (2018) the estimate is based on the posterior probability that one cluster is empty under each possible k.

Value

A list with elements

`k`	Number of components
`pp.momiw`	Posterior probability of k components under a MOM-IW-Dir(q) prior
`pp.niw`	Posterior probability of k components under a Normal-IW-Dir(q.niw) prior
`probempty`	Posterior probability that any one cluster is empty under a MOM-IW-Dir(q.niw) prior
`bf.momiw`	Bayes factor comparing 1 vs k components under a MOM-IW-Dir(q) prior
`logpen`	log of the posterior mean of the MOM-IW-Dir(q) penalty term
`logbf.niw`	Bayes factor comparing 1 vs k components under a Normal-IW-Dir(q.niw) prior

Author(s)

David Rossell

References

Fuquene J., Steel M.F.J., Rossell D. On choosing mixture components via non-local priors. 2018. arXiv

Examples

```
x <- matrix(rnorm(100*2),ncol=2)

bfnormmix(x=x,k=1:3)
```

`cil` *Treatment effect estimation for linear models via Confounder Importance Learning using non-local priors.*

Description

Treatment effect estimation for linear models in the presence of multiple treatments and a potentially high-dimensional number of controls, i.e. $p \gg n$ can be handled.

Confounder Importance Learning (CIL) proposes an estimation framework where the importance of the relationship between treatments and controls is factored in into the establishment of prior inclusion probabilities for each of these controls on the response model. This is combined with the use of non-local priors to obtain BMA estimates and posterior model probabilities.

`cil` is built on `modelSelection` and produces objects of type `cilfit`. Use `coef` and `postProb` to obtain treatment effect point estimates and posterior model probabilities, respectively, on this object class.

Usage

```
cil(y, D, X, I = NULL, family = 'normal', familyD = 'normal',
  R = 1e4, Rinit = 500, th.search = 'EB', mod1 = 'lasso_bic',
  th.prior = 'unif', priorCoef, rho.min = NULL,
  th.range = NULL, max.mod = 2^20, lpen = 'lambda.1se',
  eps = 1e-10, bvs.fit0 = NULL, th.EP = NULL, center = TRUE, scale =
  TRUE, includevars, verbose = TRUE)
```

Arguments

`y`	one-column matrix containing the observed responses. The response must be continuous (currently the only type supported)
`D`	treatment matrix with numeric columns, continuous or discrete. Any finite number of treatments are supported. If only one treatment is provided, supply this object in the same format used for `y`
`X`	matrix of controls with numeric columns, continuous or discrete. If only one treatment is provided, supply this object in the same format used for `y`
`I`	matrix with the desired interaction terms between `D` and `X`. If not informed, i.e. supplied as the default `NULL`, this term will not be included into the response model
`family`	Distribution of the outcome, e.g. 'normal', 'binomial' or 'poisson'. See `help`(modelSelection) for a full list of options
`familyD`	Distribution of the treatment(s). Only 'normal' or 'binomial' currently allowed
`R`	Number of MCMC iterations to be run by `modelSelection` on each stage of CIL (see argument `niter` therein)
`Rinit`	MCMC iterations to estimate marginal posterior inclusion probabilities under a uniform model prior, needed for EP

`th.search`	method to estimate theta values in the marginal prior inclusion probabilities of the CIL model. Options are: `EB` (Empirical Bayes, based on maximum marginal likelihood) and `EP` (Expectation propagation approximation)
`mod1`	method to estimate the feature parameters corresponding to the influence of the controls on the treatments. Supported values for this argument are 'ginv' (generalised pseudo-inverse), `lasso` (see argument `lpen`), `lasso_bic` (default), and `ridge`)
`th.prior`	prior associated to the thetas for the Empirical Bayes estimation. Currently only `unif` (Uniform prior) is supported, effectively making the EB approach the maximisation of the marginal likelihood
`priorCoef`	Prior on the response model parameters, see `modelSelection`
`rho.min`	value of ρ in (0, 1/2) employed in the prior probability model of CIL. If left uninformed, i.e. supplied as the default `NULL`, it will be set to $1/p^2$, where p is the dimension of the response model.
`th.range`	sequence of values to be considered in the grid when searching for points to initialise the search for the optimal theta parameters. If left uninformed, the function will determine a computationally suitable grid depending on the number of parameters to be estimated
`max.mod`	Maximum number of models considered when computing the marginal likelihood required by empirical Bayes. If set to `Inf` all visited models by the enumeration/MCMC are considered, but it might be computationally desirable to restrict this number when the dimension of `D` and/or `X` is large
`lpen`	penalty type supplied to `glmnet` if `mod1` is set to `lasso`. Default is `lambda.1se` (see documentation corresponding to `glmnet` for options on how to set this parameter)
`eps`	small scalar used to avoid round-offs to absolute zeroes or ones in marginal prior inclusion probabilities.
`bvs.fit0`	object returned by `modelSelection` under $\theta = 0$, used as a model exploration tool to compute EB approximation on the thetas. This argument is only supposed to be used in case of a second computation the model on the same data where `th.search` has ben changed to `EB`, in order to avoid repeating the computation of the initial `modelSelection` fit. To use this argument, supply the object residing in the slot `init.msfit` of a `cilfit`-class object.
`th.EP`	Optimal theta values under the EP approximation, obtained in a previous CIL run. This argument is only supposed to be used in case of a second computation the model on the same data where `th.search` has ben changed to `EB`, in order to save the cost of the EP search to initialise the optimisation algorithm. To use this argument, supply the object residing int the slot `th.hat` of a `cilfit`-class object.
`center`	If `TRUE`, `y` and `x` are centered to have zero mean. Dummy variables corresponding to factors are NOT centered
`scale`	If `TRUE`, `y` and columns in `x` are scaled to have variance=1. Dummy variables corresponding to factors are NOT scaled
`includevars`	Logical vector of length ncol(x) indicating variables that should always be included in the model, i.e. variable selection is not performed for these variables
`verbose`	Set `verbose==TRUE` to print iteration progress

Details

We estimate treatment effects for the features present in the treatment matrix `D`. Features in `X`, which may or may not be causal factors of the treatments of interest, only act as controls and, therefore, are not used as inferential subjects.

Confounder importance learning is a flexible treatment effect estimation framework that essentially determines how the role of the influence of `X` on `D` should affect their relationship with the response, through establishing prior inclusion probabilities on the response model for `y` according to said role. This is regulated through a hyper- parameter theta that is set according to the method supplied to `th.search`. While the `EB` option obtains a more precise estimate a priori, the `EP` alternative achieves a reasonable approximation at a fraction of the computational cost.

See references for further details on implementation and computation.

Value

Object of class `cilfit`, which extends a list with elements

`cil.teff`	BMA estimates, 0.95 intervals and posterior inclusion probabilities for treatment effects in `D`
`coef`	BMA inference for treatment effects and all other covariates
`model.postprobs`	`matrix` returning the posterior model probabilities computed in the CIL model
`margpp`	`numeric` vector containing the estimated marginal posterior inclusion probabilities of the featured treatments and controls
`margprior`	Marginal prior inclusion probabilities, as estimated by CIL
`margpp.unif`	Marginal posterior inclusion probabilities that would be obtained under a uniform model prior
`theta.hat`	Values used for the hyper-parameter theta, estimated according to the argument `th.search` specified
`treat.coefs`	Estimated weights of the effect of the control variables on each of the treatments, as estimated with the method specified in argument `mod1`
`msfit`	Object returned by `modelSelection` (of class `msfit`) of the final model estimated by CIL.
`theta.EP`	Estimated values of theta using the EP algorithm. It coincides with `theta.hat` if the argument `th.search` is set to `EB`
`init.msfit`	Initial `msfit` object used to estimate the inital model where all elements in theta are set to zero (used in the optimisation process of this hyper-parameter)

Author(s)

Miquel Torrens

References

Torrens i Dinares M., Papaspiliopoulos O., Rossell D. Confounder importance learning for treatment effect inference. https://arxiv.org/abs/2110.00314, 2021, 1–48.

See Also

`postProb` to obtain posterior model probabilities.

`coef` for inference on the treatment parameters.

Examples

```
# Simulate data
set.seed(1)
X <- matrix(rnorm(100 * 50), nrow = 100, ncol = 50)
beta_y <- matrix(c(rep(1, 6), rep(0, 44)), ncol = 1)
beta_d <- matrix(c(rep(1, 6), rep(0, 44)), ncol = 1)
alpha <- 1
d <- X %*% beta_d + rnorm(100)
y <- d * alpha + X %*% beta_y + rnorm(100)

# Confounder Importance Learning
fit1 <- cil(y = y, D = d, X = X, th.search = 'EP')

# BMA for treatment effects
coef(fit1)

# BMA for all covariates
head(fit1$coef)

# Estimated prior inclusion prob
# vs. treatment regression coefficients
plotprior(fit1)
```

`dalapl` *Density and random draws from the asymmetric Laplace distribution*

Description

`dalapl` evaluates the probability density function, `palapl` the cumulative probability function and `ralapl` generates random draws.

Usage

```
dalapl(x, th=0, scale=1, alpha=0, logscale=FALSE)

palapl(x, th=0, scale=1, alpha=0)

ralapl(n, th=0, scale=1, alpha=0)
```

Arguments

`x`	Vector of values at which to evaluate the pdf/cdf
`n`	Number of random draws
`th`	Location parameter (mode)
`scale`	Scale parameter (proportional to variance)
`alpha`	Asymmetry parameter, must be between -1 and 1
`logscale`	If TRUE the log-pdf is returned

Details

For x<=th the asymmetric Laplace pdf is

0.5*exp(-abs(th-x)/(sqrt(scale)*(1+alpha)))/sqrt(scale)

and for x>th it is

0.5*exp(-abs(th-x)/(sqrt(scale)*(1-alpha)))/sqrt(scale)

Value

`dalapl` returns the density function, `palapl` the cumulative probability, `ralapl` random draws.

Author(s)

David Rossell

Examples

```
library(mombf)
e <- ralapl(n=10^4, th=1, scale=2, alpha=0.5)
thseq <- seq(min(e),max(e),length=1000)
hist(e, main='', breaks=30, prob=TRUE)
lines(thseq, dalapl(thseq, th=1, scale=2, alpha=0.5), col=2)
```

`ddir` *Dirichlet density*

Description

Evaluate the density of a Dirichlet distribution

Usage

```
ddir(x, q, logscale=TRUE)
```

Arguments

`x`	Vector or matrix containing the value at which to evaluate the density. If a matrix, the density is evaluated for each row. Rows are renormalized to ensure they add up to 1
`q`	Dirichlet parameters. Must have the same length as `ncol(x)`, or length 1 (in which case a symmetric Dirichlet density is valuated)
`logscale`	For `logscale==TRUE`, `dimom` returns the natural log of the prior density

Value

Density of a Dirichlet(q) distribution evaluated at each row of `x`

Author(s)

David Rossell

Examples

```
library(mombf)
x= matrix(c(1/3,2/3,.5,.5),nrow=2,byrow=TRUE)
ddir(x,q=2)
```

`diwish` *Density for Inverse Wishart distribution*

Description

`diwish` returns the density for the inverse Wishart(nu,S) evaluated at Sigma.

Usage

```
diwish(Sigma, nu, S, logscale=FALSE)
```

Arguments

`Sigma`	Positive-definite matrix
`nu`	Degrees of freedom of the inverse Wishart
`S`	Scale matrix of the inverse Wishart
`logscale`	If `logscale==TRUE` the log-density is returned

Value

Inverse Wishart(nu,S) density evaluated at Sigma

Author(s)

David Rossell

See Also

`dpostNIW` for the Normal-IW posterior density

Examples

```
library(mombf)
Sigma= matrix(c(2,1,1,2),nrow=2)
diwish(Sigma,nu=4,S=diag(2))
```

`dmom` *Non-local prior density, cdf and quantile functions.*

Description

`dmom`, `dimom` and `demom` return the density for the moment, inverse moment and exponential moment priors. `pmom`, `pimom` and `pemom` return the distribution function for the univariate moment, inverse moment and exponential moment priors (respectively). `qmom` and `qimom` return the quantiles for the univariate moment and inverse moment priors. `dmomigmarg` returns the marginal density implied by a MOM(x;tau*phi)*Invgamma(phi;a/2,b/2), `pmomigmarg` its cdf. Analogously `demomigmarg` and `demomigmarg` for eMOM(x;tau*phi)*Invgamma(phi;a/2,b/2)

Usage

```
dmom(x, tau, a.tau, b.tau, phi=1, r=1, V1, baseDensity='normal', nu=3,
logscale=FALSE, penalty='product')
dimom(x, tau=1, phi=1, V1, logscale=FALSE, penalty='product')
demom(x, tau, a.tau, b.tau, phi=1, logscale=FALSE)

pmom(q, V1 = 1, tau = 1)
pimom(q, V1 = 1, tau = 1, nu = 1)
pemom(q, tau, a.tau, b.tau)

qmom(p, V1 = 1, tau = 1)
qimom(p, V1 = 1, tau = 1, nu = 1)

dmomigmarg(x,tau,a,b,logscale=FALSE)
pmomigmarg(x,tau,a,b)

demomigmarg(x,tau,a,b,logscale=FALSE)
pemomigmarg(x,tau,a,b)
```

Arguments

`x`	In the univariate setting, `x` is a vector with the values at which to evaluate the density. In the multivariate setting it is a matrix with an observation in each row.
`q`	Vector of quantiles.
`p`	Vector of probabilities.

`V1`	Scale matrix (ignored if `penalty=='product'`). Defaults to 1 in univariate setting and the identity matrix in the multivariate setting.
`tau`	Prior dispersion parameter is `tau*phi`. See details.
`a.tau`	If `tau` is left missing, an Inverse Gamma(a.tau/2,b.tau/2) is placed on `tau`. In this case `dmom` and `demom` return the density marginalized with respect to `tau`.
`b.tau`	See `a.tau`.
`phi`	Prior dispersion parameter is `tau*phi`. See details.
`r`	Prior power parameter for MOM prior is `2*r`
`baseDensity`	For `baseDensity=='normal'` a Normal MOM prior is used, for `baseDensity=='laplace'` a Laplace MOM prior, for `baseDensity=='t'` a T MOM prior with `nu` degrees of freedom is used.
`nu`	Prior parameter indicating the degrees of freedom for the quadratic T MOM and iMOM prior densities. The tails of the inverse moment prior are proportional to the tails of a multivariate T with `nu` degrees of freedom.
`penalty`	`penalty=='product'` indicates that product MOM/iMOM should be used. `penalty=='quadratic'` indicates quadratic iMOM. See Details.
`logscale`	For `logscale==TRUE`, `dimom` returns the natural log of the prior density.
`a`	The marginal prior on phi is IG(a/2,b/2)
`b`	The marginal prior on phi is IG(a/2,b/2)

Details

For `type=='quadratic'` the density is as follows. Define the quadratic form q(theta)= (theta-theta0)' * solve(V1) * (theta-theta0) / (tau*phi). The normal moment prior density is proportional to q(theta)*dmvnorm(theta,theta0,tau*phi*V1). The T moment prior is proportional to q(theta)*dmvt(theta,theta0,tau*phi*V1,c
The inverse moment prior density is proportional to `q(theta)^(-(nu+d)/2) * exp(-1/q(theta))`.

pmom, pimom and qimom use closed-form expressions, while qmom uses nlminb to find quantiles numerically. Only the univariate version is implemented. In this case the product MOM is equivalent to the quadratic MOM. The same happens for the iMOM.

`dmomigmarg` returns the marginal density

p(x)= int MOM(x;0,tau*phi) IG(phi;a/2,b/2) dphi

Value

Prior density, cumulative distribution function or quantile.

Author(s)

David Rossell

References

Johnson V.E., Rossell D. Non-Local Prior Densities for Default Bayesian Hypothesis Tests. Journal of the Royal Statistical Society B, 2010, 72, 143-170.

Johnson V.E., Rossell D. Bayesian model selection in high-dimensional settings. Journal of the American Statistical Assocation, 2012, 107, 649-660

See http://rosselldavid.googlepages.com for technical reports.

Examples

```
#evaluate and plot the moment and inverse moment priors
library(mombf)
tau <- 1
thseq <- seq(-3,3,length=1000)
plot(thseq,dmom(thseq,tau=tau),type='l',ylab='Prior density')
lines(thseq,dimom(thseq,tau=tau),lty=2,col=2)
```

`dpostNIW` *Posterior Normal-IWishart density*

Description

dpostNIW evalutes the posterior Normal-IWishart density at (mu,Sigma). rpostNIW draws independent samples. This posterior corresponds to a Normal model for the data

x[i,] ~ N(mu,Sigma) iid i=1,...,n

under conjugate priors

mu | Sigma ~ N(mu0, g Sigma) Sigma ~ IW(nu0, S0)

Usage

```
dpostNIW(mu, Sigma, x, g=1, mu0=rep(0,length(mu)), nu0=nrow(Sigma)+1, S0,
  logscale=FALSE)

rpostNIW(n, x, g=1, mu0=0, nu0, S0, precision=FALSE)
```

Arguments

`mu`	Vector of length p
`Sigma`	p x p positive-definite covariance matrix
`x`	n x p data matrix (individuals in rows, variables in columns)
`g`	Prior dispersion parameter for mu
`mu0`	Prior mean for mu
`nu0`	Prior degrees of freedom for Sigma
`S0`	Prior scale matrix for Sigma, by default set to I/nu0
`logscale`	set to TRUE to get the log-posterior density

`n`	Number of samples to draw
`precision`	If set to `TRUE`, samples from the precision matrix (inverse of Sigma) are returned instead

Value

`dpostNIW` returns the Normal-IW posterior density evaluated at (mu,Sigma).

`rpostNIW` returns a list with two elements. The first element are posterior draws for the mean. The second element are posterior draws for the covariance (or its inverse if `precision==TRUE`). Only lower-diagonal elements are returned (`Sigma[lower.tri(Sigma,diag=TRUE)]`).

Author(s)

David Rossell

See Also

`diwish` for the inverse Wishart prior density, `marginalNIW` for the integrated likelihood under a Normal-IW prior

Examples

```
#Simulate data
x= matrix(rnorm(100),ncol=2)
#Evaluate posterior at data-generating truth
mu= c(0,0)
Sigma= diag(2)
dpostNIW(mu,Sigma,x=x,g=1,nu0=4,log=FALSE)
```

`eprod` *Expectation of a product of powers of Normal or T random variables*

Description

Compute the mean of prod(x)^power when x follows T_dof(mu,sigma) distribution (dof= -1 for multivariate Normal).

Usage

```
eprod(m, S, power = 1, dof = -1)
```

Arguments

`m`	Location parameter
`S`	Scale matrix. For multivariate T with dof>2 the covariance is S*dof/(dof-2). For the multivariate Normal the covariance is S.
`power`	Power that the product is raised to
`dof`	Degrees of freedom of the multivariate T. Set to -1 for the multivariate Normal.

Details

The calculation is based on the computationally efficient approach by Kan (2008).

Value

Expectation of the above-mentioned product

Author(s)

John Cook

References

Kan R. From moments of sum to moments of product. Journal of Multivariate Analysis 99 (2008), 542-554.

Examples

```
#Check easy independence case
m <- c(0,3); S <- matrix(c(2,0,0,1),ncol=2)

eprod(m, S, power=2)

(m[1]^2+S[1][1])*(m[2]^2+S[2][2])
```

`getBIC` *Obtain BIC and EBIC*

Description

Obtain Bayesian Information Criterion and Extended Bayesian Information Criterion

Usage

```
getBIC(object)

getEBIC(object)
```

Arguments

`object`	Object of class msfit returned by `modelSelection`

Details

Let p be the total number of parameters and n the sample size. The BIC of a model k with p_k parameters is

- 2 L_k + p_k log(n)

and the EBIC is

- 2 L_k + p_k log(n) + 2 log(p choose p_k)

Function modelSelection returns the log posterior probability of a model, postProb = log(m_k) + log(prior k), where m_k is the marginal likelihood of the model and prior k its prior probability.

When running function modelSelection with priorCoef=bicprior() and priorDelta=modelunifprior(), the BIC approximation is used for m_k, that is

log(m_k) = L_k - 0.5 * p_k log(n)

and all models are equally likely a priori, log(prior k)= p log(1/2). Then the BIC can be easily recovered

BIC_k= -2 * [postProb + p log(2)]

When using priorCoef=bicprior() and priorDelta=modelbbprior(), log(prior k)= - log(p+1) - log(p choose p_k), hence

EBIC_k= -2 * [postProb + log(p+1)].

Value

BIC or EBIC values for all models enumerated / visited by modelSelection

Author(s)

David Rossell

See Also

`modelSelection` to perform model selection

Examples

```
x <- matrix(rnorm(100*3),nrow=100,ncol=3)
theta <- matrix(c(1,1,0),ncol=1)
y <- x %*% theta + rnorm(100)
ybin <- y>0

#Obtain BIC
ms= modelSelection(ybin, x=x, priorCoef=bicprior(),
priorDelta=modelunifprior(), family='binomial')
getBIC(ms)

#Obtain EBIC
ms2= modelSelection(ybin, x=x, priorCoef=bicprior(),
priorDelta=modelbbprior(), family='binomial')
getEBIC(ms2)
```

`hald` *Hald Data*

Description

Montgomery and Peck (1982) illustrated variable selection techniques on the Hald cement data and gave several references to other analysis. The response variable *y* is the *heat evolved* in a cement mix. The four explanatory variables are ingredients of the mix, i.e., x1: *tricalcium aluminate*, x2: *tricalcium silicate*, x3: *tetracalcium alumino ferrite*, x4: *dicalcium silicate*. An important feature of these data is that the variables x1 and x3 are highly correlated (corr(x1,x3)=-0.824), as well as the variables x2 and x4 (with corr(x2,x4)=-0.975). Thus we should expect any subset of (x1,x2,x3,x4) that includes one variable from highly correlated pair to do as any subset that also includes the other member.

Usage

```
data(hald)
```

Format

`hald` is a matrix with 13 observations (rows) and 5 variables (columns), the first column is the dependent variable. `y.hald` and `x.hald` are also availables.

Source

Montgomery, D.C., Peck, E.A. (1982) *Introduction to linear regression analysis,* John Wiley, New York.

`marginalNIW` *Marginal likelihood under a multivariate Normal likelihood and a conjugate Normal-inverse Wishart prior.*

Description

The argument `z` can be used to specify cluster allocations. If left missing then the usual marginal likelihood is computed, else it is computed conditional on the clusters (this is equivalent to the product of marginal likelihoods across clusters)

Usage

```
marginalNIW(x, xbar, samplecov, n, z, g,  mu0=rep(0,ncol(x)),
nu0=ncol(x)+4, S0, logscale=TRUE)
```

Arguments

`x`	Data matrix (individuals in rows, variables in columns). Alternatively you can leave missing and specify `xbar`, `samplecov` and `n` instead
`xbar`	Either a vector with column means of `x` or a list where each element corresponds to the column means for each cluster
`samplecov`	Either the sample covariance matrix `cov(x)` or a list where each element contains the covariance for each clsuter
`n`	Either an integer indicating the sample size `nrow(x)` or a vector indicating the cluster counts `table(z)`
`z`	Optional argument specifying cluster allocations
`g`	Prior dispersion parameter for mu
`mu0`	Prior mean for mu
`nu0`	Prior degrees of freedom for Sigma
`S0`	Prior scale matrix for Sigma, by default set to I/nu0
`logscale`	set to TRUE to get the log-posterior density

Details

The function computes

p(x)= int p(x | mu,Sigma) p(mu,Sigma) dmu dSigma

where p(x[i])= N(x[i]; mu,Sigma) iid i=1,...,n

p(mu | Sigma)= N(mu; mu0, g Sigma) p(Sigma)= IW(Sigma; nu0, S0)

Value

If `z` is missing the integrated likelihood under a Normal-IW prior. If `z` was specified then the product of integrated likelihoods across clusters

Author(s)

David Rossell

See Also

`dpostNIW` for the posterior Normal-IW density.

Examples

```
#Simulate data
x= matrix(rnorm(100),ncol=2)

#Integrated likelihood under correct model
marginalNIW(x,g=1,nu0=4,log=FALSE)

#Integrated likelihood under random cluster allocations
z= rep(1:2,each=25)
marginalNIW(x,z=z,g=1,nu0=4,log=FALSE)
```

`mixturebf-class` *Class "mixturebf"*

Description

Stores the output of Bayesian model selection for mixture models, e.g. as produced by function `bfnormmix`.

Methods are provided for retrieving the posterior probability of a given number of mixture components, posterior means and posterior samples of the mixture model parameters.

Objects from the Class

Typically objects are automatically created by a call to `bfnormmix`.

Slots

The class has the following slots:

postprob data.frame containing posterior probabilities for different numbers of components (k) and log-posterior probability of a component being empty (contain no individuals)

p Number of variables in the data to which the model was fit

n Number of observations in the data to which the model was fit

priorpars Prior parameters used when fitting the model

postpars Posterior parameters for a 1-component mixture, e.g. for a Normal mixture the posterior is N(mu1,Sigma/prec) IW(nu1,S1)

mcmc For each considered value of k, posterior samples for the parameters of the k-component model are stored

Methods

coef Computes posterior means for all parameters

show `signature(object = "mixturebf")`: Displays general information about the object.

postProb `signature(object = "mixturebf")`: Extracts posterior model probabilities, Bayes factors and posterior probability of a cluster being empty

postSamples `signature(object = "mixturebf")`: Extracts posterior samples

Author(s)

David Rossell

References

Fuquene J., Steel M.F.J., Rossell D. On choosing mixture components via non-local priors. 2018. arXiv

See Also

See also `bfnormmix`

Examples

```
showClass("mixturebf")
```

`modelSelection` *Bayesian variable selection for linear models via non-local priors.*

Description

Bayesian model selection for linear, asymmetric linear, median and quantile regression under non-local or Zellner priors. p»n can be handled.

modelSelection enumerates all models when feasible and uses a Gibbs scheme otherwise. See `coef` and `coefByModel` for estimates and posterior intervals of regression coefficients, and `rnlp` for posterior samples.

modelsearchBlockDiag seeks the highest posterior probability model using an iterative block search.

Usage

```
modelSelection(y, x, data, smoothterms, nknots=9,
groups=1:ncol(x), constraints, center=TRUE, scale=TRUE,
enumerate, includevars=rep(FALSE,ncol(x)), models,
maxvars, niter=5000, thinning=1,
burnin=round(niter/10), family='normal', priorCoef,
priorGroup, priorDelta=modelbbprior(1,1),
priorConstraints,
priorVar=igprior(.01,.01),
priorSkew=momprior(tau=0.348), phi, deltaini=rep(FALSE,ncol(x)),
initSearch='greedy', method='auto', adj.overdisp='intercept',
hess='asymp', optimMethod, optim_maxit, initpar='none', B=10^5,
XtXprecomp= ifelse(ncol(x)<10^4,TRUE,FALSE), verbose=TRUE)

modelsearchBlockDiag(y, x, priorCoef=momprior(tau=0.348),
priorDelta=modelbbprior(1,1), priorVar=igprior(0.01,0.01),
blocksize=10, maxiter=10, maxvars=100, maxlogmargdrop=20,
maxenum=10, verbose=TRUE)
```

Arguments

`y`	Either a formula with the regression equation or a vector with observed responses. The response can be either continuous or of class `Surv` (survival outcome). If `y` is a formula then `x`, `groups` and `constraints` are automatically created

`x`	Design matrix with linear covariates for which we want to assess if they have a linear effect on the response. Ignored if `y` is a formula
`data`	If `y` is a formula then `data` should be a data frame containing the variables in the model
`smoothterms`	Formula for non-linear covariates (cubic splines), modelSelection assesses if the variable has no effect, linear or non-linear effect. `smoothterms` can also be a design matrix or data.frame containing linear terms, for each column modelSelection creates a spline basis and tests no/linear/non-linear effects
`nknots`	Number of spline knots. For cubic splines the non-linear basis adds knots-4 coefficients for each linear term, we recommend setting `nknots` to a small/moderate value
`groups`	If variables in `x` such be added/dropped in groups, `groups` indicates the group that each variable corresponds to (by default each variable goes in a separate group)
`constraints`	Constraints on the model space. List with length equal to the number of groups; if group[[i]]=c(j,k) then group i can only be in the model if groups j and k are also in the model
`center`	If `TRUE`, `y` and `x` are centered to have zero mean. Dummy variables corresponding to factors are NOT centered
`scale`	If `TRUE`, `y` and columns in `x` are scaled to have variance=1. Dummy variables corresponding to factors are NOT scaled
`enumerate`	Default is `TRUE` if there's less than 15 variable groups. If `TRUE` all models with up to `maxvars` are enumerated, else Gibbs sampling is used to explore the model space
`includevars`	Logical vector of length ncol(x) indicating variables that should always be included in the model, i.e. variable selection is not performed for these variables
`models`	Optional logical matrix indicating the models to be enumerated with rows equal to the number of desired models and columns to the number of variables in `x`.
`maxvars`	When `enumerate==TRUE` only models with up to maxvars variables enumerated (defaults to all variables). In `modelsearchBlockDiag` a sequence of models is defined from 1 up to `maxvars`
`niter`	Number of Gibbs sampling iterations
`thinning`	MCMC thinning factor, i.e. only one out of each `thinning` iterations are reported. Defaults to thinning=1, i.e. no thinning
`burnin`	Number of burn-in MCMC iterations. Defaults to `.1*niter`. Set to 0 for no burn-in
`family`	Family of parametric distribution. Use 'normal' for Normal errors, 'binomial' for logistic regression, 'poisson' for Poisson regression. 'twopiecenormal' for two-piece Normal, 'laplace' for Laplace errors and 'twopiecelaplace' for double exponential. For 'auto' the errors are assumed continuous and their distribution is inferred from the data among 'normal', 'laplace', 'twopiecenormal' and 'twopiecelaplace'. 'laplace' corresponds to median regression and 'twopiecelaplace' to quantile regression. See argument `priorSkew`

`priorCoef`	Prior on coefficients, created by `momprior`, `imomprior`, `emomprior` or `zellnerprior`. Prior dispersion is on coefficients/sqrt(scale) for Normal and two-piece Normal, and on coefficients/sqrt(2*scale) for Laplace and two-piece Laplace.
`priorGroup`	Prior on grouped coefficients (e.g. categorical predictors with >2 categories, splines). Created by `groupmomprior`, `groupemomprior`, `groupimomprior` or `groupzellnerprior`
`priorDelta`	Prior on model space. Use `modelbbprior()` for Beta-Binomial prior, `modelbinomprior(p)` for Binomial prior with prior inclusion probability p, `modelcomplexprior` for Complexity prior, or `modelunifprior()` for Uniform prior
`priorConstraints`	Prior distribution on the number of terms subject to hierarchical constrains that are included in the model
`priorVar`	Inverse gamma prior on scale parameter. For Normal outcomes variance=scale, for Laplace outcomes variance=2*scale
`priorSkew`	Either a fixed value for tanh(alpha) where alpha is the asymmetry parameter or a prior on tanh(alpha). For `family=='twopiecelaplace'` setting alpha=a is equivalent to performing quantile regression for the quantile (1+a)/2. Ignored if `family` is 'normal' or 'laplace'.
`phi`	The error variance in Gaussian models, typically this is unknown and is left missing
`deltaini`	Logical vector of length `ncol(x)` indicating which coefficients should be initialized to be non-zero. Defaults to all variables being excluded from the model
`initSearch`	Algorithm to refine `deltaini`. `initSearch=='greedy'` uses a greedy Gibbs sampling search. `initSearch=='SCAD'` sets `deltaini` to the non-zero elements in a SCAD fit with cross-validated regularization parameter. `initSearch=='none'` leaves `deltaini` unmodified
`method`	Method to compute marginal likelihood. `method=='Laplace'` for Laplace approx, `method=='ALA'` for approximate Laplace approximation. `method=='MC'` for Importance Sampling, `method=='Hybrid'` for Hybrid Laplace-IS (only available for piMOM prior). See Details. `method=='auto'` attempts to use exact calculations when possible, otherwise ALA if available, otherwise Laplace approx.
`adj.overdisp`	Only used when method=='ALA'. Over-dispersion adjustment in models with fixed dispersion parameter, as in logistic and Poisson regression. adj.overdisp='none' for no adjustment (not recommended, particularly for Poisson models). adj.overdisp='intercept' to estimate over-dispersion from the intercept-only model, and adj.overdisp='residuals' from the Pearson residuals of each model
`hess`	Method to estimat the hessian in the Laplace approximation to the integrated likelihood under Laplace or asymmetric Laplace errors. When hess=='asymp' the asymptotic hessian is used, hess=='asympDiagAdj' a diagonal adjustment is applied (see Rossell and Rubio for details).
`optimMethod`	Algorithm to maximize objective function when method=='Laplace'. Leave unspecified or set optimMethod=='auto' for an automatic choice. optimMethod=='LMA' uses modified Newton-Raphson algorithm, 'CDA' coordinate descent algorithm
`optim_maxit`	Maximum number of iterations when method=='Laplace'

`initpar`	Initial regression parameter values when finding the posterior mode to approximate the integrated likelihood. 'none', 'MLE', 'MLE-aisgd', 'L1', 'L2-aisgd', or a numeric vector with initial values. If p<n/2 MLE is used, else L1 (regularization parameter set via BIC). 'auto': if n>10,000 or p>200, then MLE-aisgd or L2-aisgd are used. aisgd stands for averaged intrinsic stochastic gradient descent (see function sgd in package sgd)
`B`	Number of samples to use in Importance Sampling scheme. Ignored if `method=='Laplace'`
`XtXprecomp`	Set to `TRUE` to pre-compute the Gram matrix x'x upfront (saves time), to `FALSE` to compute and store elements only as needed (saves memory)
`verbose`	Set `verbose==TRUE` to print iteration progress
`blocksize`	Maximum number of variables in a block. Careful, the cost of the algorithm is of order `2^blocksize`
`maxiter`	Maximum number of iterations, each iteration includes a screening pass to add and subtract variables
`maxlogmargdrop`	Stop the sequence of models when the drop in log p(ylmodel) is greater than `maxlogmargdrop`. This option avoids spending unnecessary time exploring overly large models
`maxenum`	If the posterior mode found has less than `maxenum` variables then do a full enumeration of all its submodels

Details

Let delta be the vector indicating inclusion/exclusion of each column of `x` in the model. The Gibbs algorithm sequentially samples from the posterior of each element in delta conditional on all the remaining elements in delta and the data. To do this it is necessary to evaluate the marginal likelihood for any given model. These have closed-form expression for the MOM prior, but for models with >15 variables these are expensive to compute and Laplace approximations are used instead (for the residual variance a log change of variables is used, which improves the approximation). For other priors closed forms are not available, so by default Laplace approximations are used. For the iMOM prior we also implement a Hybrid Laplace-IS which uses a Laplace approximation to evaluate the integral wrt beta and integrates wrt phi (residual variance) numerically.

It should be noted that Laplace approximations tend to under-estimate the marginal densities when the MLE for some parameter is very close to 0. That is, it tends to be conservative in the sense of excluding more variables from the model than an exact calculation would.

Finally, method=='plugin' provides a BIC-type approximation that is faster than exact or Laplace methods, at the expense of some accuracy. In non-sparse situations where models with many variables have large posterior probability method=='plugin' can be substantially faster.

For more details on the methods used to compute marginal densities see Johnson & Rossell (2012).

`modelsearchBlockDiag` uses the block search method described in Papaspiliopoulos & Rossell. Briefly, spectral clustering is run on X'X to cluster variables into blocks of `blocksize` and subsequently the Coolblock algorithm is used to define a sequence of models of increasing size. The exact integrated likelihood is evaluated for all models in this path, the best model chosen, and the scheme iteratively repeated to add and drop variables until convergence.

Value

Object of class `msfit`, which extends a list with elements

`postSample`	`matrix` with posterior samples for the model indicator. `postSample[i,j]==1` indicates that variable j was included in the model in the MCMC iteration i
`postOther`	`postOther` returns posterior samples for parameters other than the model indicator, i.e. basically hyper-parameters. If hyper-parameters were fixed in the model specification, `postOther` will be empty.
`margpp`	Marginal posterior probability for inclusion of each covariate. This is computed by averaging marginal post prob for inclusion in each Gibbs iteration, which is much more accurate than simply taking `colMeans(postSample)`

.

`postMode`	Model with highest posterior probability amongst all those visited
`postModeProb`	Unnormalized posterior prob of posterior mode (log scale)
`postProb`	Unnormalized posterior prob of each visited model (log scale)
`priors`	List with priors specified when calling `modelSelection`

Author(s)

David Rossell

References

Johnson V.E., Rossell D. Non-Local Prior Densities for Default Bayesian Hypothesis Tests. Journal of the Royal Statistical Society B, 2010, 72, 143-170.

Johnson V.E., Rossell D. Bayesian model selection in high-dimensional settings. Journal of the American Statistical Association, 2012, 107, 649-660.

Papaspiliopoulos O., Rossell, D. Scalable Bayesian variable selection and model averaging under block orthogonal design. 2016

Rossell D., Rubio F.J. Tractable Bayesian variable selection: beyond normality. 2016

See Also

`msfit-class` for details on the output. `postProb` to obtain posterior model probabilities. `coef.msfit` for Bayesian model averaging estimates and intervals. `predict.msfit` for BMA estimates and intervals for user-supplied covariate values. `rnlp` to obtain posterior samples for the coefficients. `nlpMarginal` to compute marginal densities for a given model.

Examples

```
#Simulate data
x <- matrix(rnorm(100*3),nrow=100,ncol=3)
theta <- matrix(c(1,1,0),ncol=1)
y <- x %*% theta + rnorm(100)

#Specify prior parameters
```

```
priorCoef <- momprior(tau=0.348)
priorDelta <- modelunifprior()

#Alternative model space prior: 0.5 prior prob for including any covariate
priorDelta <- modelbinomprior(p=0.5)

#Alternative: Beta-Binomial prior for model space
priorDelta <- modelbbprior(alpha.p=1,beta.p=1)

#Model selection
fit1 <- modelSelection(y=y, x=x, center=FALSE, scale=FALSE,
priorCoef=priorCoef, priorDelta=priorDelta)
postProb(fit1) #posterior model probabilities

fit1$margpp #posterior marginal inclusion prob

coef(fit1) #BMA estimates, 95% intervals, marginal post prob
```

`mombf` *Moment and inverse moment Bayes factors for linear models.*

Description

`mombf` computes moment Bayes factors to test whether a subset of regression coefficients are equal to some user-specified value. `imombf` computes inverse moment Bayes factors.

Usage

```
mombf(lm1, coef, g, prior.mode, baseDensity='normal', nu=3, theta0,
logbf=FALSE, B=10^5)
imombf(lm1, coef, g, prior.mode, nu = 1, theta0 , method='adapt',
nquant=100, B = 10^5)
```

Arguments

`lm1`	Linear model fit, as returned by `lm1`.
`coef`	Vector with indexes of coefficients to be tested. e.g. `coef==c(2,3)` and `theta0==c(0,0)` tests `coef(lm1)[2]=coef(lm1)[3]=0`.
`g`	Vector with prior parameter values. See `dmom` and `dimom` for details.
`prior.mode`	If specified, `g` is set such that the prior mode is `prior.mode`
`baseDensity`	Density upon which the Mom prior is based. `baseDensity=='normal'` results in the normal Mom prior, `baseDensity=='t'` in the t Mom prior with `nu` degrees of freedom.
`nu`	For `mombf`, `nu` specifies the degrees of freedom of the t Mom prior. It is ignored unless `baseDensity=='t'`. `nu` defaults to 3. For `imombf`, `nu` specifies the degrees of freedom for the inverse moment prior (see `dimom` for details). Defaults to `nu=1`, which Cauchy-like tails.

`theta0`	Null value for the regression coefficients. Defaults to 0.
`logbf`	If `logbf==TRUE` the natural logarithm of the Bayes factor is returned.
`method`	Numerical integration method to compute the bivariate integral (only used by `imombf`). For `method=='adapt'`, the inner integral is evaluated (via `integrate`) at a series of `nquant` quantiles of the residual variance posterior distribution, and then averaged as described in Johnson (1992). Set `method=='MC'` to use Monte Carlo integration.
`nquant`	Number of quantiles at which to evaluate the integral for known `sigma`. Only used if `method=='adapt'`.
`B`	Number of Monte Carlo samples to estimate the T Mom and the inverse moment Bayes factor. Only used in `mombf` if `baseDensity=='t'`. Only used in `imombf` if `method=='MC'`.

Details

These functions actually call `momunknown` and `imomunknown`, but they have a simpler interface. See `dmom` and `dimom` for details on the moment and inverse moment priors.

Value

`mombf` returns the moment Bayes factor to compare the model where `theta!=theta0` with the null model where `theta==theta0`. Large values favor the alternative model; small values favor the null. `imombf` returns inverse moment Bayes factors.

Author(s)

David Rossell

References

See http://rosselldavid.googlepages.com for technical reports. For details on the quantile integration, see Johnson, V.E. A Technique for Estimating Marginal Posterior Densities in Hierarchical Models Using Mixtures of Conditional Densities. Journal of the American Statistical Association, Vol. 87, No. 419. (Sep., 1992), pp. 852-860.

See Also

`nlpMarginal` for a better interface to integrated likelihoods and `modelSelection` to also search over the model space

Examples

```
##compute Bayes factor for Hald's data
data(hald)
lm1 <- lm(hald[,1] ~ hald[,2] + hald[,3] + hald[,4] + hald[,5])

# Set g so that interval (-0.2,0.2) has 5% prior probability
# (in standardized effect size scale)
priorp <- .05; q <- .2
```

```
gmom <- priorp2g(priorp=priorp,q=q,prior='normalMom')
gimom <- priorp2g(priorp=priorp,q=q,prior='iMom')

mombf(lm1,coef=2,g=gmom) #moment BF
imombf(lm1,coef=2,g=gimom,B=10^5) #inverse moment BF
```

`momknown` *Bayes factors for moment and inverse moment priors*

Description

`momknown` and `momunknown` compute moment Bayes factors for linear models when `sigma^2` is known and unknown, respectively. The functions can also be used to compute approximate Bayes factors for generalized linear models and other settings. `imomknown`, `imomunknown` compute inverse moment Bayes factors.

Usage

```
momknown(theta1hat, V1, n, g = 1, theta0, sigma, logbf = FALSE)
momunknown(theta1hat, V1, n, nuisance.theta, g = 1, theta0, ssr, logbf =
FALSE)
imomknown(theta1hat, V1, n, nuisance.theta, g = 1, nu = 1, theta0,
sigma, method='adapt', B=10^5)
imomunknown(theta1hat, V1, n, nuisance.theta, g = 1, nu = 1, theta0,
ssr, method='adapt', nquant = 100, B = 10^5)
```

Arguments

`theta1hat`	Vector with regression coefficients estimates.
`V1`	Matrix proportional to the covariance of `theta1hat`. For linear models, the covariance is `sigma^2*V1`.
`n`	Sample size.
`nuisance.theta`	Number of nuisance regression coefficients, i.e. coefficients that we do not wish to test for.
`ssr`	Sum of squared residuals from a linear model call.
`g`	Prior parameter. See `dmom` and `dimom` for details.
`theta0`	Null value for the regression coefficients. Defaults to 0.
`sigma`	Dispersion parameter is `sigma^2`.
`logbf`	If `logbf==TRUE` the natural logarithm of the Bayes factor is returned.
`nu`	Prior parameter for the inverse moment prior. See `dimom` for details. Defaults to `nu=1`, which Cauchy-like tails.

`method`	Numerical integration method (only used by `imomknown` and `imomunknown`). Set `method=='adapt'` in `imomknown` to integrate using adaptive quadrature of functions as implemented in the function `integrate`. In `imomunknown` the integral is evaluated as in `imomknown` at a series of `nquant` quantiles of the posterior for `sigma`, and then averaged as described in Johnson (1992). Set `method=='MC'` to use Monte Carlo integration.
`nquant`	Number of quantiles at which to evaluate the integral for known `sigma`.
`B`	Number of Monte Carlo samples to estimate the inverse moment Bayes factor. Ignored if `method!='MC'`.

Details

See `dmom` and `dimom` for details on the moment and inverse moment priors. The Zellner-Siow g-prior is given by dmvnorm(theta,theta0,n*g*V1).

Value

`momknown` and `momunknown` return the moment Bayes factor to compare the model where `theta!=theta0` with the null model where `theta==theta0`. Large values favor the alternative model; small values favor the null. `imomknown` and `imomunknown` return inverse moment Bayes factors.

Author(s)

David Rossell

References

See http://rosselldavid.googlepages.com for technical reports.

For details on the quantile integration, see Johnson, V.E. A Technique for Estimating Marginal Posterior Densities in Hierarchical Models Using Mixtures of Conditional Densities. Journal of the American Statistical Association, Vol. 87, No. 419. (Sep., 1992), pp. 852-860.

See Also

`mombf` and `imombf` for a simpler interface to compute Bayes factors in linear regression

Examples

```
#simulate data from probit regression
set.seed(4*2*2008)
n <- 50; theta <- c(log(2),0)
x <- matrix(NA,nrow=n,ncol=2)
x[,1] <- rnorm(n,0,1); x[,2] <- rnorm(n,.5*x[,1],1)
p <- pnorm(x[,1]*theta[1]+x[,2]+theta[2])
y <- rbinom(n,1,p)

#fit model
glm1 <- glm(y~x[,1]+x[,2],family=binomial(link = ”probit”))
thetahat <- coef(glm1)
V <- summary(glm1)$cov.scaled
```

```
#compute Bayes factors to test whether x[,1] can be dropped from the model
g <- .5
bfmom.1 <- momknown(thetahat[2],V[2,2],n=n,g=g,sigma=1)
bfimom.1 <- imomknown(thetahat[2],V[2,2],n=n,nuisance.theta=2,g=g,sigma=1)
bfmom.1
bfimom.1
```

`msfit-class` *Class "msfit"*

Description

Stores the output of Bayesian variable selection, as produced by function `modelSelection`. The class extends a list, so all usual methods for lists also work for `msfit` objects, e.g. accessing elements, retrieving names etc.

Methods are provided to compute posterior probabilities, obtaining regression coefficient estimates and posterior intervals (both via Bayesian model averaging and for individual models), and sampling from their posterior distribution, as indicated below.

Objects from the Class

Typically objects are automatically created by a call to `modelSelection`. Alternatively, objects can be created by calls of the form `new("msfit",x)` where `x` is a list with the adequate elements (see slots).

Slots

The class extends a list with elements:

postSample `matrix` with posterior samples for the model indicator. `postSample[i,j]==1` indicates that variable j was included in the model in the MCMC iteration i

postOther `postOther` returns posterior samples for parameters other than the model indicator, i.e. basically hyper-parameters. If hyper-parameters were fixed in the model specification, `postOther` will be empty.

margpp Marginal posterior probability for inclusion of each covariate. This is computed by averaging marginal post prob for inclusion in each Gibbs iteration, which is much more accurate than simply taking `colMeans(postSample)`.

postMode Model with highest posterior probability amongst all those visited

postModeProb Unnormalized posterior prob of posterior mode (log scale)

postProb Unnormalized posterior prob of each visited model (log scale)

family Residual distribution, i.e. argument `family` when calling `modelSelection`

p Number of variables

priors Priors specified when calling `modelSelection`

ystd For internal use. Stores the response variable, standardized if `center` or `scale` were set to `TRUE`

xstd For internal use. Stores the covariates, standardized if `center` or `scale` were set to `TRUE`

stdconstants For internal use. If `center` or `scale` were set to `TRUE`, stores the sample mean and standard deviation of the outcome and covariates

call Stores info about the call, the formula used (if any), splines used etc

Methods

coef Obtains posterior means and intervals via Bayesian model averaging

coefByModel Obtains posterior means and intervals for individual models

predict Obtains posterior means and intervals for given covariate values. These are posterior intervals for the mean, not posterior predictive intervals for the outcome

show `signature(object = "msfit")`: Displays general information about the object.

postProb `signature(object = "msfit")`: Extracts posterior model probabilities.

rnlp `signature(object = "msfit")`: Obtain posterior samples for regression coefficients.

Author(s)

David Rossell

References

Johnson VE, Rossell D. Non-Local Prior Densities for Default Bayesian Hypothesis Tests. Journal of the Royal Statistical Society B, 2010, 72, 143-170

Johnson VE, Rossell D. Bayesian model selection in high-dimensional settings. Journal of the American Statistical Association, 107, 498:649-660.

See Also

See also `modelSelection` and `rnlp`.

Examples

```
showClass("msfit")
```

`msPriorSpec-class` *Class "msPriorSpec"*

Description

Stores the prior distributions to be used for Bayesian variable selection in normal regression models. This class can be used to specify the prior on non-zero regression coefficients, the model indicator or the nuisance parameters.

Usage

```
bicprior()

momprior(taustd=1, tau, tau.adj=10^6, r=1)
imomprior(tau, tau.adj=10^6)
emomprior(tau, tau.adj=10^6)
zellnerprior(taustd=1, tau, tau.adj=10^6)
normalidprior(taustd=1, tau, tau.adj=10^6)

groupmomprior(taustd=1, tau, tau.adj=10^6)
groupimomprior(tau, tau.adj=10^6)
groupemomprior(tau, tau.adj=10^6)
groupzellnerprior(taustd=1, tau, tau.adj=10^6)

modelunifprior()
modelbinomprior(p=0.5)
modelbbprior(alpha.p=1, beta.p=1)
modelcomplexprior(c=1)

igprior(alpha=.01, lambda=.01)
```

Arguments

`tau`	Prior dispersion parameter for covariates undergoing selection
`taustd`	Prior dispersion parameter for covariates undergoing selection. It is calibrated so that 'taustd=1' equals the unit information prior.
`tau.adj`	Prior variance in Normal prior for covariates not undergoing selection
`r`	MOM prior parameter is `2*r`
`p`	Prior inclusion probability for binomial prior on model space
`alpha.p`	Beta-binomial prior on model space has parameters alpha.p, beta.p
`beta.p`	Beta-binomial prior on model space has parameters alpha.p, beta.p
`c`	Under the Complexity prior the prior probability of having `k` variables in the model is proportional to `1/p^(ck)`
`alpha`	Inverse gamma prior has parameters alpha/2, lambda/2
`lambda`	Inverse gamma prior has parameters alpha/2, lambda/2

Details

DISCUSSION OF PRIOR ON PARAMETERS

Let beta=(beta_1,...,beta_p) be the regression coefficients for individual variables and delta=(delta_1,...,delta_q) those for grouped variables (e.g. factors or smooth terms in modelSelection).

momprior, emomprior, imomprior, zellnerprior and normalid can be priors on both beta or delta. For further information see the vignette.

groupzellnerprior is the prior density on delta

p_z(delta; tau)= prod_j N(delta_j; 0, (tau/p_j)) (X_j'X_j)^-1)

where X_j are the design matrix columns associated to delta_j and p_j=ncol(X_j) is th number of covariates in the group (for groupmomprior, the term in the denominator is (p_j +2) instead of p_j). A default tau=n=nrow(X_j) mimics the unit information prior and implies that the ratio of variance explained by X_j / residual variance is expected to be 1 a priori. To set the dispersion in terms of unit information prior, taustd is also available.

groupmomprior adds a quadratic MOM penalty

p_m(delta; tau)= p_z(delta; tau * n) prod_j delta_j'X_j'X_jdelta_j ncol(X_j)/(tau * n * p_j / (p_j + 2))

and analogously for eMOM and iMOM. Note that unlike groupzellnerprior, the nrow(X_j) factor is already included in the code. This is done to give user introduced tau values a roughly similar meaning between momprior and groupmomprior.

DISCUSSION OF PRIOR ON MODELS

Under the uniform prior, the prior probability of any model is 1 / number of models.

Under the Binomial, Beta-Binomial and Complexity priors a model with k out of K active variables has prior probability P(Z=k) / (K choose k), where where Z ~ Binomial(K,p), Z ~ BetaBinomial(K,alpha.p,beta.p) or for the Complexity prior P(Z=k) proportional to `1/K^(c*k)`.

Objects from the Class

Objects can be created by calls of the form `new("msPriorSpec", ...)`, but it is easier to use creator functions.

For priors on regression coefficients use `momprior`, `imomprior` or `emomprior`. For prior on model space `modelunifprior`, `modelbinomprior` `modelbbprior`, or `modelcomplexprior`. For prior on residual variance use `igprior`.

Slots

`priorType`**:** Object of class `"character"`. `"coefficients"` indicates that the prior is for the non-zero regression coefficients. `"modelIndicator"` that it is for the model indicator, and `"nuisancePars"` that it is for the nuisance parameteres. Several prior distributions are available for each choice of `priorType`, and these can be speicified in the slot `priorDist`.

`priorDistr`**:** Object of class `"character"`. If `priorType=="coefficients"`, `priorDistr` can be equal to "pMOM", "piMOM", "peMOM", "zellner", "normalid", "groupMOM" or "groupzellner" (product moment, product inverse moment, product exponential moment, Zellner prior, normal prior with $\Sigma = \mathbf{I}$, respectively). If `priorType=="modelIndicator"`, `priorDistr` can be equal to "uniform" or "binomial" to specify a uniform prior (all models equaly likely a priori) or a binomial prior, or to "complexity" for the Complexity prior of Castillo et al 2015. For a binomial prior, the prior inclusion probability for any single variable must be specified in slot `priorPars['p']`. For a beta-binomial prior, the Beta hyper-prior parameters must be in `priorPars['alpha.p']` and `priorPars['beta.p']`. For the Complexity prior, the prior parameter must be in the slot `priorPars['c']`. If `priorType=="nuisancePars"`, `priorDistr` must be equal to "invgamma". This corresponds to an inverse gamma distribution for the residual variance, with parameters specified in the slot `priorPars`.

`priorPars`: Object of class `"vector"`, where each element must be named. For `priorDistr=='pMOM'`, there must be an element "r" (MOM power is 2r). For any `priorDistr` there must be either an element "tau" indicating the prior dispersion or elements "a.tau" and "b.tau" specifying an inverse gamma hyper-prior for "tau". Optionally, there may be an element "tau.adj" indicating the prior dispersion for the adjustment variables (i.e. not undergoing variable selection). If not defined, "tau.adj" is set to 0.001 by default. For `priorDistr=='binomial'`, there must be either an element "p" specifying the prior inclusion probability for any single covariate, or a vector with elements "alpha.p" and "beta.p" specifying a Beta(alpha.p,beta.p) hyper-prior on p. For `priorDistr=='invgamma'` there must be elements "alpha" and "lambda". The prior for the residual variance is an inverse gamma with parameteres `.5*alpha` and `.5*lambda`.

Methods

No methods defined with class "msPriorSpec" in the signature.

Note

When new instances of the class are created a series of check are performed to ensure that a valid prior specification is produced.

Author(s)

David Rossell

References

Johnson VE, Rossell D. Non-Local Prior Densities for Default Bayesian Hypothesis Tests. Journal of the Royal Statistical Society B, 2010, 72, 143-170

Johnson VE, Rossell D. Bayesian model selection in high-dimensional settings. Journal of the American Statistical Association, 107, 498:649-660.

See Also

See also `modelSelection` for an example of defining an instance of the class and perform Bayesian model selection.

Examples

```
showClass("msPriorSpec")
```

`nlpmarginals` *Marginal density of the observed data for linear regression with Normal, two-piece Normal, Laplace or two-piece Laplace residuals under non-local and Zellner priors*

Description

The marginal density of the data, i.e. the likelihood integrated with respect to the given prior distribution on the regression coefficients of the variables included in the model and an inverse gamma prior on the residual variance.

`nlpMarginal` is the general function, the remaining ones correspond to particular cases and are kept for backwards compatibility with old code, and will be deprecated in the future.

Usage

```
nlpMarginal(sel, y, x, data, smoothterms, nknots=9, groups=1:ncol(x),
family="normal", priorCoef, priorGroup,
priorVar=igprior(alpha=0.01,lambda=0.01), priorSkew=momprior(tau=0.348),
phi, method='auto', adj.overdisp='intercept', hess='asymp', optimMethod,
optim_maxit, initpar='none', B=10^5, logscale=TRUE, XtX, ytX)

pimomMarginalK(sel, y, x, phi, tau=1, method='Laplace', B=10^5, logscale=TRUE, XtX, ytX)
pimomMarginalU(sel, y, x, alpha=0.001, lambda=0.001, tau=1,
method='Laplace', B=10^5, logscale=TRUE, XtX, ytX)
pmomMarginalK(sel, y, x, phi, tau, r=1, method='auto', B=10^5,
logscale=TRUE, XtX, ytX)
pmomMarginalU(sel, y, x, alpha=0.001, lambda=0.001, tau=1,
r=1, method='auto', B=10^5, logscale=TRUE, XtX, ytX)
```

Arguments

`sel`	Vector with indexes of columns in x to be included in the model. Ignored if `y` is a formula
`y`	Either a formula with the regression equation or a vector with observed responses. The response can be either continuous or of class `Surv` (survival outcome). If `y` is a formula then `x`, `groups` and `constraints` are automatically created
`x`	Design matrix with linear covariates for which we want to assess if they have a linear effect on the response. Ignored if `y` is a formula
`data`	If `y` is a formula then `data` should be a data frame containing the variables in the model
`smoothterms`	Formula for non-linear covariates (cubic splines), modelSelection assesses if the variable has no effect, linear or non-linear effect. `smoothterms` can also be a design matrix or data.frame containing linear terms, for each column modelSelection creates a spline basis and tests no/linear/non-linear effects

`nknots`	Number of spline knots. For cubic splines the non-linear basis adds knots-4 coefficients for each linear term, we recommend setting `nknots` to a small/moderate value
`groups`	If variables in `x` such be added/dropped in groups, `groups` indicates the group that each variable corresponds to (by default each variable goes in a separate group)
`family`	Residual distribution. Possible values are 'normal','twopiecenormal','laplace', 'twopiecelaplace'
`priorCoef`	Prior on coefficients, created by `momprior`, `imomprior`, `emomprior` or `zellnerprior`. Prior dispersion is on coefficients/sqrt(scale) for Normal and two-piece Normal, and on coefficients/sqrt(2*scale) for Laplace and two-piece Laplace.
`priorGroup`	Prior on grouped coefficients (e.g. categorical predictors with >2 categories, splines). Created by `groupmomprior`, `groupemomprior`, `groupimomprior` or `groupzellnerprior`
`priorVar`	Inverse gamma prior on scale parameter, created by `igprior(). For Normal variance=scale, for Laplace variance=2*scale.`
`priorSkew`	Either a number fixing tanh(alpha) where alpha is the asymmetry parameter or a prior on residual skewness parameter, assumed to be of the same family as priorCoef. Ignored if `family` is 'normal' or 'laplace'.
`method`	Method to approximate the integral. See `help(modelSelection)`.
`adj.overdisp`	Only used for method=='ALA'. Over-dispersion adjustment for models with fixed dispersion parameter such as logistic and Poisson regression
`hess`	Method to estimat the hessian in the Laplace approximation to the integrated likelihood under Laplace or asymmetric Laplace errors. When hess=='asymp' the asymptotic hessian is used, hess=='asympDiagAdj' a diagonal adjustment is applied (see Rossell and Rubio for details).
`optimMethod`	Algorithm to maximize objective function when method=='Laplace'. Leave unspecified or set optimMethod=='auto' for an automatic choice. optimMethod=='LMA' uses modified Newton-Raphson algorithm, 'CDA' coordinate descent algorithm
`optim_maxit`	Maximum number of iterations when method=='Laplace'
`initpar`	Initial regression parameter values when finding the posterior mode to approximate the integrated likelihood. See help(modelSelection)
`B`	Number of Monte Carlo samples to use (ignored unless `method=='MC'`)
`logscale`	If `logscale==TRUE` the log marginal density is returned.
`XtX`	Optionally, specify the matrix X'X. Useful when the function must be called a large number of times.
`ytX`	Optionally, specify the vector y'X. Useful when the function must be called a large number of times.
`phi`	Disperson parameter. See `help(modelSelection)`
`alpha`	Prior for phi is inverse gamma `alpha/2`, `lambda/2`
`lambda`	Prior for phi is inverse gamma `alpha/2`, `lambda/2`
`tau`	Prior dispersion parameter for MOM and iMOM priors (see details)
`r`	Prior power parameter for MOM prior is `2*r`

Details

The marginal density of the data is equal to the integral of N(y;x[,sel]*theta,phi*I) * pi(theta|phi,tau) * IG(phi;alpha/2,lambda/2) with respect to theta, where pi(theta|phi,tau) is a non-local prior and IG denotes the density of an inverse gamma.

`pmomMarginalK` and `pimomMarginalK` assume that the residual variance is known and therefore the inverse-gamma term in the integrand can be ommitted.

The product MOM and iMOM densities can be evaluated using the functions `dmom` and `dimom`.

Value

Marginal density of the observed data under the specified prior.

Author(s)

David Rossell

References

Johnson V.E., Rossell D. Non-Local Prior Densities for Default Bayesian Hypothesis Tests. Journal of the Royal Statistical Society B, 2010, 72, 143-170. See http://rosselldavid.googlepages.com for technical reports.

See Also

`modelSelection` to perform model selection based on product non-local priors. `momunknown`, `imomunknown`, `momknown`, `imomknown` to compute Bayes factors for additive MOM and iMOM priors

Examples

```
x <- matrix(rnorm(100*2),ncol=2)
y <- x %*% matrix(c(.5,1),ncol=1) + rnorm(nrow(x))
pmomMarginalK(sel=1, y=y, x=x, phi=1, tau=1, method='Laplace')
pmomMarginalK(sel=1:2, y=y, x=x, phi=1, tau=1, method='Laplace')
```

`plotprior` *Plot estimated marginal prior inclusion probabilities*

Description

Plot marginal prior inclusion probabilities as estimated by cil versus regression coefficients for the treatment(s) equation(s)

Usage

```
plotprior(object, xlab, ylab, ylim=c(0,1), ...)
```

Arguments

`object`	Object of class cilfit returned by `cil`
`xlab`	x-axis label
`ylab`	y-axis label
`ylim`	y-axis limits
`...`	Other arguments passed on to `plot`

Value

A plot of prior inclusion probabilities vs treatment regression coefficients (dots). The line shows the (empirical Bayes) fit

Author(s)

David Rossell

See Also

`cil`

Examples

```
#See help(cil)
```

`postModeOrtho` *Bayesian model selection and averaging under block-diagonal X'X for linear models.*

Description

postModeOrtho is for diagonal X'X, postModeBlockDiag for the more general block-diagonal X'X, where X is the matrix with predictors.

Both functions return the model of highest posterior probability of any given size using an efficient search algorithm. This sequence of models includes the highest posterior probability model (HPM). Posterior model probabilities, marginal variable inclusion probabilities and Bayesian model averaging estimates are also provided. The unknown residual variance is integrated out using an exact deterministic algorithm of low computational cost (see details in reference).

Usage

```
postModeOrtho(y, x, priorCoef=momprior(tau=0.348), priorDelta=modelbbprior(1,1),
priorVar=igprior(0.01,0.01), bma=FALSE, includeModels, maxvars=100)

postModeBlockDiag(y, x, blocks, priorCoef=zellnerprior(tau=nrow(x)),
priorDelta=modelbinomprior(p=1/ncol(x)),priorVar=igprior(0.01,0.01), bma=FALSE,
maxvars=100, momcoef)
```

Arguments

`y`	Outcome
`x`	Matrix with predictors. If an intercept is desired x should include a column of 1's.
`blocks`	Factor or integer vector of length ncol(x) indicating the block that each column in x belongs to.
`priorCoef`	Prior distribution for the coefficients. Object created with `momprior`, `imomprior`, `emomprior` or `zellnerprior`.
`priorDelta`	Prior on model space. Use `modelbbprior()` for Beta-Binomial prior, `modelbinomprior(p)` for Binomial prior with prior inclusion probability p, `modelcomplexprior` for Complexity prior, or `modelunifprior()` for Uniform prior
`priorVar`	Inverse gamma prior on residual variance, created with `igprior()`
`bma`	Set to TRUE to obtain marginal inclusion probabilities and Bayesian model averaging parameter estimates for each column of x.
`includeModels`	Models that should always be included when computing posterior model probabilities. It must be a list, each element in the list corresponds to a model and must be a logical or numeric vector indicating the variables in that model
`maxvars`	The search for the HPM is restricted to models with up to maxvars variables (note: posterior model probabilities and BMA are valid regardless of maxvars)
`momcoef`	optional argument containing pre-computed coefficients needed to obtain the marginal likelihood under the pMOM prior. A first call to postModeBlockDiag returns these coefficients, thus this argument is useful to speed up successive calls.

Details

The first step is to list a sequence of models with 0,...,maxvars variables which, under fairly general conditions listed in Papaspiliopoulos & Rossell (2016), is guaranteed to include the HPM. Then posterior model probabilities are computed for all these models to determine the HPM, evaluate the marginal posterior of the residual variance on a grid, and subsequently compute the marginal density p(y) via adaptive quadrature. Finally this adaptive grid is used to compute marginal inclusion probabilities and Bayesian model averaging estimates. For more details see Papaspiliopoulos & Rossell (2016).

Value

List with elemants

`models`	data.frame indicating the variables included in the sequence of models found during the search of the HPM, and their posterior probabilities. The model with highest posterior probability in this list is guaranteed to be the HPM.
`phi`	data.frame containing an adaptive grid of phi (residual variance) values and their marginal posterior density p(phily).
`logpy`	log-marginal density p(y), i.e. normalization constant of p(phily).

`bma`	Marginal posterior inclusion probabilities and Bayesian model averaging estimates for each column in x.
`postmean.model`	Coefficient estimates conditional on each of the models in `models`
`momcoef`	If a MOM prior was specified in priorCoef, momcoef stores some coefficients needed to compute its marginal likelihood

Author(s)

David Rossell

References

Papaspiliopoulos O., Rossell D. Scalable Bayesian variable selection and model averaging under block-orthogonal design. 2016

Examples

```
#Simulate data
set.seed(1)
p <- 400; n <- 410
x <- scale(matrix(rnorm(n*p),nrow=n,ncol=p),center=TRUE,scale=TRUE)
S <- cov(x)
e <- eigen(cov(x))
x <- t(t(x %*% e$vectors)/sqrt(e$values))
th <- c(rep(0,p-3),c(.5,.75,1)); phi <- 1
y <- x %*% matrix(th,ncol=1) + rnorm(n,sd=sqrt(phi))

#Fit
priorCoef=zellnerprior(tau=n); priorDelta=modelbinomprior(p=1/p); priorVar=igprior(0.01,0.01)
pm.zell <- postModeOrtho(y,x=x,priorCoef=priorCoef,priorDelta=priorDelta,priorVar=priorVar,
bma=TRUE)

#Best models
head(pm.zell$models)

#Posterior probabilities for sequence of models
nvars <- sapply(strsplit(as.character(pm.zell$models$modelid),split=','),length)
plot(nvars,pm.zell$models$pp,ylab='post prob',xlab='number of vars',ylim=0:1,xlim=c(0,50))

#Marginal posterior of phi
plot(pm.zell$phi,type='l',xlab='phi',ylab='p(phi|y)')

#Marginal inclusion prob & BMA estimates
plot(pm.zell$bma$margpp,ylab='Marginal inclusion prob')
plot(pm.zell$bma$coef,ylab='BMA estimate')
```

`postProb` *Obtain posterior model probabilities*

Description

Obtain posterior model probabilities after running Bayesian model selection

Usage

```
postProb(object, nmax, method='norm')
```

Arguments

`object`	Object of class msfit returned by `modelSelection` or class mixturebf, e.g. returned by `bfnormmix`
`nmax`	Maximum number of models to report (defaults to no max)
`method`	Only when `class(object)` is msfit. For 'norm' probabilities are obtained by renormalizing the stored integrated likelihoods, for 'exact' they are given by the proportion of MCMC visits to each model. 'norm' has less variability but can be biased if the chain has not converged.

Value

A `data.frame` with posterior model probabilities in column pp. Column modelid indicates the indexes of the selected covariates (empty for the null model with no covariates).

Author(s)

David Rossell

See Also

`modelSelection` to perform model selection

Examples

```
#See help(modelSelection)
```

`postSamples` *Extract posterior samples from an object*

Description

Obtain posterior model probabilities after running Bayesian model selection

Usage

```
postSamples(object)
```

Arguments

`object`	Object containing posterior samples, e.g. of class mixture bf as returned by `bfnormmix`

Value

For objects of class `mixturebf`, a list with one element for each considered number of mixture components.

Each element in the list contains posterior samples on the mixture weights (eta) and other component-specific parameters such as means (mu) and Cholesky decomposition of the inverse covariance matrix (cholSigmainv)

Author(s)

David Rossell

Examples

```
#See help(bfnormmix)
```

`priorp2g` *Moment and inverse moment prior elicitation*

Description

`priorp2g` finds the `g` value giving `priorp` prior probability to the interval (`-q`,`q`).

Usage

```
priorp2g(priorp, q, nu=1, prior=c("iMom", "normalMom", "tMom"))
```

Arguments

`prior`	`prior=='normalMom'` does computations for the normal moment prior, `prior=='tMom'` for the T moment prior, `prior=='iMom'` does computations for the inverse moment prior. Currently `prior=='tMom'` is not implemented in `priorp2g`.
`q`	`priorp2g` returns g giving `priorp` prior probability to the interval `(-q,q)`.
`nu`	Prior degrees of freedom for the T moment prior or the iMom prior (ignored if `prior=='normalMom'`).
`priorp`	`priorp2g` returns g giving `priorp` prior probability to the interval `(-q,q)`

Details

See `pmom` and `pimom` for the MOM/iMOM cumulative distribution functions.

Value

`priorp2g` returns g giving `priorp` prior probability to the interval `(-q,q)`.

Author(s)

David Rossell <`rosselldavid@gmail.com`>

References

See http://rosselldavid.googlepages.com for technical reports.

See Also

`pmom`, `pimom`

Examples

```
data(hald)
lm1 <- lm(hald[, 1] ~ hald[, 2] + hald[, 3] + hald[, 4] + hald[, 5])

#find g value giving 0.05 probability to interval (-.2,.2)
priorp <- .05; q <- .2
gmom <- priorp2g(priorp=priorp, q=q, prior='normalMom')
gimom <- priorp2g(priorp=priorp, q=q, prior='iMom')
gmom
gimom
```

`rnlp` *Posterior sampling for regression parameters*

Description

Gibbs sampler for linear, generalized linear and survival models under product non-local priors, Zellner's prior and a Normal approximation to the posterior. Both sampling conditional on a model and Bayesian model averaging are implemented (see Details).

If `x` and `y` not specified samples from non-local priors/posteriors with density proportional to d(theta) N(theta; m, V) are produced, where d(theta) is the non-local penalty term.

Usage

```
rnlp(y, x, m, V, msfit, outcometype, family, priorCoef, priorGroup,
priorVar, isgroup, niter=10^3, burnin=round(niter/10), thinning=1, pp='norm')
```

Arguments

`y`	Vector with observed responses. When `class(y)=='Surv'` sampling is based on the Cox partial likelihood, else a linear model is assumed.
`x`	Design matrix with all potential predictors
`m`	Mean for the Normal kernel
`V`	Covariance for the Normal kernel
`msfit`	Object of class `msfit` returned by `modelSelection`. If specified Bayesian model averaging posterior samples are returned, according to posterior model probabilities in `msfit`, and then arguments `y`, `x`, `m`, `V` etc. If `msfit` is missing then posterior samples under the full model `y ~ x` are returned
`outcometype`	Type of outcome. Possible values are "Continuous", "glm" or "Survival"
`family`	Assumed family for the family. Some possible values are "normal", "binomial logit" and "Cox"
`priorCoef`	Prior distribution for the coefficients. Ignored if `msfit` is supplied. Must be object of class `msPriorSpec`, e.g. created by `momprior`, `emomprior`, `imomprior`, `zellnerprior`
`priorGroup`	Prior on grouped coefficients (e.g. categorical predictors with >2 categories, splines), as passed to `modelSelection`
`priorVar`	Prior on residual variance. Ignored if `msfit` supplied. Must be object of class `msPriorSpec`, e.g. created with `igprior`
`isgroup`	Logical vector where `TRUE` indicates that the variable is part of a group, e.g. one of several dummy indicators for a discrete covariate
`niter`	Number of MCMC iterations
`burnin`	Number of burn-in MCMC iterations. Defaults to `.1*niter`. Set to 0 for no burn-in

`thinning`	MCMC thinning factor, i.e. only one out of each `thinning` iterations are reported. Defaults to no thinning
`pp`	When `msfit` is provided this is the method to compute posterior model probabilities, which determine the sampled models. Can be 'norm' or 'exact', see `postProb` for details.

Details

The algorithm is implemented for product MOM (pMOM), product iMOM (piMOM) and product eMOM (peMOM) priors. The algorithm combines an orthogonalization that provides low serial correlation with a latent truncation representation that allows fast sampling.

When `y` and `x` are specified sampling is for the linear regression posterior. When argument `msfit` is left missing, posterior sampling is for the full model regressing `y` on all covariates in `x`. When `msfit` is specified each model is drawn with probability given by `postProb(msfit)`. In this case, a Bayesian Model Averaging estimate of the regression coefficients can be obtained by applying `colMeans` to the `rnlp` ouput matrix.

When `y` and `x` are left missing, sampling is from a density proportional to d(theta) N(theta; m,V), where d(theta) is the non-local penalty (e.g. d(theta)=prod(theta^(2r)) for the pMOM prior).

Value

Matrix with posterior samples

Author(s)

David Rossell

References

D. Rossell and D. Telesca. Non-local priors for high-dimensional estimation, 2014. http://arxiv.org/pdf/1402.5107v2.pdf

See Also

`modelSelection` to perform model selection and compute posterior model probabilities. For more details on prior specification see `msPriorSpec-class`.

Examples

```
#Simulate data
x <- matrix(rnorm(100*3),nrow=100,ncol=3)
theta <- matrix(c(1,1,0),ncol=1)
y <- x %*% theta + rnorm(100)
fit1 <- modelSelection(y=y, x=x, center=FALSE, scale=FALSE)

th <- rnlp(msfit=fit1, niter=100)
colMeans(th)
```

Index