Package: modelSelection (via r-universe)

November 2, 2026

Type Package

Version 1.0.7

Date 2026-05-16

Title High-Dimensional Model Selection

Maintainer David Rossell <rosselldavid@gmail.com>

Depends R (>= 2.14.0), methods

Suggests parallel, testthat, patrick

Imports Rcpp (>= 0.12.16), dplyr, glmnet, intervals, L0Learn, Matrix, mclust, mgcv, mvtnorm, ncvreg, pracma, sparseMatrixStats, survival

LinkingTo Rcpp, RcppArmadillo, RcppEigen

Description Model selection and averaging for regression, generalized linear models, generalized additive models, graphical models and mixtures, focusing on Bayesian model selection and information criteria (Bayesian information criterion etc.). See Rossell (2025) <doi:10.5281/zenodo.17119597> (see the URL field below for its URL) for a hands-on book describing the methods, examples and suggested citations if you use the package.

License GPL (>= 2)

URL https://github.com/davidrusi/modelSelection, https://github.com/davidrusi/modelSelection-book

BugReports https://github.com/davidrusi/modelSelection/issues

LazyLoad yes

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

RoxygenNote 7.3.2

NeedsCompilation yes

Author David Rossell [aut, cre], John D. Cook [ctb], Donatello Telesca [aut], P. Roebuck [ctb], Oriol Abril [aut], Miquel Torrens [aut], Peter Mueller [ctb], William Hallahan [ctb]

Repository https://cran.r-universe.dev

Date/Publication 2026-05-16 15:31:17 UTC

RemoteUrl https://github.com/cran/modelSelection

RemoteRef HEAD

RemoteSha eb3fa2fe2390815c3196c237d1c160c16f47ff5f

Contents

bbPrior . . . 3
bestBIC . . . 4
bfnormmix . . . 6
cil . . . 8
dalapl . . . 12
ddir . . . 13
diwish . . . 14
dmom . . . 15
dpostNIW . . . 17
eprod . . . 18
icfit-class . . . 19
icov . . . 20
localnulltest . . . 21
marginalLikelihood . . . 24
marginalNIW . . . 27
mixturebf-class . . . 28
modelSelection . . . 29
modelSelectionGGM . . . 35
msfit-class . . . 39
msfit_ggm-class . . . 41
msPriorSpec-class . . . 42
plotprior . . . 45
postProb . . . 46
postSamples . . . 47
priorp2g . . . 48
rnlp . . . 49

Index **52**

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

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

`bestBIC` *Model with best AIC, BIC, EBIC or other general information criteria (getIC)*

Description

Search for the regression model attaining the best value of the specified information criterion

Usage

```
bestAIC(...)

bestAIC_fast(..., fastmethod="all")

bestBIC(...)

bestBIC_fast(..., fastmethod="all")

bestEBIC(...)

bestEBIC_fast(..., fastmethod="all")

bestIC(..., penalty)

bestIC_fast(..., penalty, fastmethod="all")
```

Arguments

`...`	Arguments passed on to `modelSelection`. The first and main argument is a model formula, see the examples
`penalty`	General information penalty. For example, since the AIC penalty is 2, bestIC(...,penalty=2) is the same as bestAIC(...)

`fastmethod` Method used for fast model search. Set "L0Learn" to use the L0Learn package, "L1" to use LASSO (glmnet package), "adaptiveL1" for adaptive LASSO, and "CDA" for coordinate descent. Not all these options may be available for some GLM families (e.g. L0Learn only supports `family="normal"` and `"binomial"`). The default "all" uses all available methods

Details

`bestAIC`, `bestBIC`, `bestEBIC`, `bestIC` perform full model enumeration when possible and otherwise resort to MCMC to explore the models, as discussed in function `modelSelection`.

`bestAIC_fast`, `bestBIC_fast`, `bestEBIC_fast`, `bestIC_fast` use a faster algorithm. It first identifies a subset of promising models, and then computes the specified criterion for each of them to find the best one within the subset. For Gaussian and binary outcomes it uses function `L0Learn.fit` from package `L0Learn` (Hazimeh et al, 2023), which combines coordinate descent with local combinatorial search to find good models of each size. L1 returns all the models found in the LASSO regularization path. CDA returns a single model found by coordinate descent, i.e. adding/dropping one covariate at a time to improve the specified criterion (BIC, AIC, ...).

`bestBIC` and the other functions documented here take similar arguments to those of `modelSelection`, but here no priors on models or parameters are needed.

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)

the AIC is

- 2 L_k + p_k 2

the EBIC is

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

and a general information criterion with a given model size penalty

- 2 L_k + p_k penalty

The MCMC model search is based on assigning a probability to each model, and then using MCMC to sample models from this distribution. The probability of model k is

exp(- IC_k / 2) / sum_l exp(- IC_l / 2)

where IC_k is the value of the information criterion (BIC, EBIC...)

Hence the model with best (lowest) IC_k has highest probability, which means that it is likely to be sampled by the MCMC algorithm.

Value

Object of class `icfit`. Use (coef, summary, confint, predict) to get inference for the top model, and `help(icfit-class)` for more details on the returned object.

Author(s)

David Rossell

References

H. Hazimeh, R. Mazumder, T. Nonet. L0learn: A scalable package for sparse learning using l0 regularization. Journal of Machine Learning Research 24.205 (2023): 1-8.

See Also

`modelSelection` to perform Bayesian model selection, and for a description of the arguments that should be passed in ...

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
df <- data.frame(y, ybin, x)

#BIC for all models (the intercept is also selected in/out)
fit= bestBIC(y ~ X1 + X2, data=df)
fit

#Same, but setting the BIC's log(n) penalty manually
#change the penalty for other General Info Criteria
#n= nrow(x)
#fit= bestIC(y ~ X1 + X2, data=df, penalty=log(n))

summary(fit) #usual GLM summary

coef(fit) #MLE under top model

#confint(fit) #conf int under top model (requires MASS package)


#Binary outcome
fit2= bestBIC(ybin ~ X1 + X2, data=df, family='binomial')
fit2
```

`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

$$d(\mu, A) Dir(\eta; q) \prod_j N(\mu_j; \mu 0, g\Sigma_j) IW(\Sigma_j; \nu_0, S0)$$

where

$$d(\mu, A) = [\prod_{j<l} (\mu_j - \mu_l)' A (\mu_j - \mu_l)]$$

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 = momprior(taustd=1),
  rho.min = NULL, rho.max = 0.95,
  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	Lower bound on the covariate's prior inclusion probability. By default, it is set to $1/p$, where p is the number of covariates
rho.max	Upper bound on the covariate's prior inclusion probability
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 learns from data the amount of confounding in the data, based on a so-called confounding coefficient, and uses this to set covariate prior inclusion probabilities. Roughly, the coefficient measures to what extend covariates that are truly related to the outcome are also truly related (high confounding) or not related (no confounding) to the treatment(s).

In high confounding, covariates in X that appear to be related to D are assigned high inclusion probability in the regression for y. In low confounding, they're assigned low prior inclusion probability. See function plotprior to visualize these prior probabilities. Prior probabilities are 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, particularly when there are many covariates in X, the EP is much faster computationally and typically gives very similar results.

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.

plotprior to plot the estimated prior probabilities, as a function of

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

```
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

```
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

```
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 moment and inverse moment prior densities
tau <- 1
thseq <- seq(-3, 3, length=500)
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])
```

---

icfit-class *Class "icfit"*

Description

Stores the output of the search for the model with best information criterion value, e.g. produced by `bestBIC`, `bestBIC`, `bestAIC` or `bestIC`. The class extends a list, so all usual methods for lists also work for `icfit` objects, e.g. accessing elements, retrieving names etc.

Methods are provided to extract coefficients, predictions, confidence intervals and summary information about the best model.

Objects from the Class

icfit objects are automatically created by a call to `bestBIC` or similar.

Slots

The class extends a list with elements:

topmodel names of the variables in the top model

topmodel.fit top model as fitted by glm

models data frame with the information criterion for all models (when enumeration is feasible) or those visited by an MCMC model search in modelSelection (when enumeration is not feasible)

varnames the names of all variables in the design matrix

msfit Output of modelSelection (used to search the top model)

Methods

coef glm fit for the top model

confint Confidence intervals under the top model

predict Predictions for the top model (predict.glm)

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

Author(s)

David Rossell

See Also

See also `bestBIC`.

Examples

```
showClass("icfit")
```

---

icov *Extract estimated inverse covariance*

Description

Extract the estimated inverse covariance from an `msfit_ggm` object.

Bayesian model averaging is used, optionally entries with posterior probability of being non-zero below a threshold are set to 0.

Usage

```
icov(fit, threshold)
```

Arguments

`fit`	Object of class `msfit_ggm`, returned by `modelSelectionGGM`
`threshold`	Entries with posterior probability of being non-zero below threshold are set to 0. If missing this argument is ignored and no entries are set to exact zeroes. When the goal is to identify zeroes, a sensible default is `threshold=0.95`

Details

The inverse covariance is obtained via Bayesian model averaging, using posterior samples of Omega. When `threshold` is specified, entries in the BMA estimate are set to zero, which may result in a non positive-definite matrix.

Value

Estimated inverse covariance matrix.

Author(s)

David Rossell

See Also

modelSelectionGGM, coef.msfit_ggm for Bayesian model averaging estimates and intervals.

Examples

```
## See modelSelectionGGM
```

---

localnulltest *Local variable selection*

Description

Learn whether covariate effects are zero at given coordinates using Bayesian model selection or information criteria.

Use coef to extract estimates and posterior probabilities for local effects.

Usage

```
localnulltest(y, x, z, x.adjust, localgridsize, localgrid,
nbaseknots=20, nlocalknots=c(5,10,15), localknots,
basedegree=3, cutdegree=0,
usecutbasis=TRUE, priorCoef=normalidprior(),
priorGroup=priorCoef, priorModel=modelbbprior(),
mc.cores=min(4,length(nlocalknots)), return.mcmc=FALSE, verbose=FALSE,
...)

localnulltest_fda(y, x, z, x.adjust, function_id,
Sigma='AR/MA', localgridsize, localgrid, nbaseknots=20,
nlocalknots=c(5,10,15), localknots,
basedegree=3, cutdegree=0, usecutbasis=TRUE,
priorCoef=momprior(), priorGroup=groupmomprior(),
priorModel=modelbbprior(), mc.cores=min(4,length(nlocalknots)),
return.mcmc=FALSE, verbose=FALSE, ...)

localnulltest_givenknots(y, x, z, x.adjust, localgridsize,
localgrid, nbaseknots=20, nlocalknots=10, localknots,
basedegree=3, cutdegree=0,
usecutbasis=TRUE, priorCoef=normalidprior(),
priorGroup=priorCoef, priorModel=modelbbprior(),
```

```
verbose=FALSE, ...)

localnulltest_fda_givenknots(y, x, z, x.adjust, function_id,
Sigma='AR/MA', localgridsize, localgrid, nbaseknots=20,
nlocalknots=10, localknots,
basedegree=3, cutdegree=0, usecutbasis=TRUE,
priorCoef=momprior(), priorGroup=groupmomprior(),
priorModel=modelbbprior(), verbose=FALSE, ...)
```

Arguments

y	Vector with the outcome variable
x	Numerical matrix with covariate values
z	Matrix with d-dimensional coordinates (d>=1$ for each entry in y, and d columns)
x.adjust	Optionally, further adjustment covariates to be included in the model with no testing being performed
function_id	Function identifier. It is assumed that one observes multiple functions over z, this is the identifier of each individual function
Sigma	Error covariance. By default 'identity', other options are 'MA', 'AR' or 'AR/MA' (meaning that BIC is used to choose between MA and AR). Alternatively the user can supply a function such that `Sigma(z[i,],z[j,])` returns the within-function `cov(y[i,], y[j,])`
localgridsize	Local test probabilities will be returned for a grid of z values of size `localgridsize` for each dimension. If `ncol(z)==1` then `localgridsize` defaults to 100, else to 10
localgrid	Regions at which tests will be performed. Defaults to dividing each `[min(z[,i]), max(z[,i])]` into 10 equal intervals. If provided, `localgrid` must be a list with one entry for each `z[,i]`, containing a vector with the desired grid for that `z[,i]`
nbaseknots	Number of knots for the spline approximation to the baseline effect of x on y
nlocalknots	Number of knots for the basis capturing the local effects. Ignored if `localknots` is specified
localknots	Knots to be used for the local tests. The same knots are used for all columns in z, so make sure that all columns span the same range of values. For a multi-resolution analysis, `localknots` should be a list where each entry specifyes the knots for one resolution
basedegree	Degree of the spline approximation to the baseline
cutdegree	Degree of the cut spline basis used for testing
usecutbasis	If `FALSE`, then the basis is not cut and a standard spline basis is returned (not recommended unless you know what you're doing)
priorCoef	Prior on the coefficients, passed on to `modelSelection`
priorGroup	Prior on grouped coefficients, passed on to `modelSelection`
priorModel	Prior on the models, passed on to `modelSelection`

mc.cores	If package parallel is available on your system and nlocalknots has several entries defining several resolution levels, they will be run in parallel on mc.cores
return.mcmc	Set to TRUE to return the MCMC output from modelSelection
verbose	If TRUE some progress information is printed
...	Other arguments to be passed on to modelSelection, e.g. family='binomial' for logistic regression

Details

Local variable selection considers the model

$$y_i = \beta_0(z_i) + sum_{j=1}^{p}\beta_j(z_i, x_i) + e_i$$

$\beta_0(z_i)$ is the baseline mean

$\beta_j(z_i, x_i)$ is local effect of covariate j at coordinate z_i

e_i a Gaussian error term assumed either independent or with a covariance structure given by Sigma. If assuming independence it is possible to consider alternatives to Gaussianity, e.g. set family='binomial' for logistic regression or family='poisson' for Poisson regression

Note: a sum-to-zero type constraint is set on $\beta_1(z_i, x_i)$ so that it defines a deviation from the baseline mean $\beta_0(z_i)$

We model β_0 using B-splines of degree basedegree with nbaseknots knots. We model β_j using B-splines of degree cutdegree with nlocalknots. Using cutdegree=0 runs fastest is usually gives similar inference than higher degrees, and is hence recommended by default.

Value

Object of class localtest, which extends a list with elements

covareffects	Estimated local covariate effects at different z values, 0.95 posterior intervals and posterior probability for the existence of an effect
pplocalgrid	Posterior probabilities for the existence of an effect for regions of z values. Do not use these unless you know what you're doing
covareffects.mcmc	MCMC output used to build covareffects. Only returned if return.mcmc=TRUE
ms	Objects of class msfit returned by modelSelection
pp_localknots	Posterior probability for each resolution level (value of nlocalknots)
Sigma	Input parameter
nlocalknots	Input parameter
basedegree	Input parameter
cutdegree	Input parameter
knots	Input parameters
regionbounds	List with region bounds defined by the local testing knots at each resolution level

Author(s)

David Rossell

Examples

```
#Simulate outcome and 2 covariates
#Covariate 1 has local effect for z>0
#Covariate 2 has no effect for any z

truemean= function(x,z) {
    ans= double(nrow(x))
    group1= (x[,1]==1)
    ans[group1]= ifelse(z[group1] <=0, cos(z[group1]), 1)
    ans[!group1]= ifelse(z[!group1]<=0, cos(z[!group1]), 1/(z[!group1]+1)^2)
    return(ans)
}

n= 1000
x1= rep(0:1,c(n/2,n/2))
x2= x1 + rnorm(n)
x= cbind(x1,x2)
z= runif(n,-3,3)
m= truemean(x,z)
y= truemean(x,z) + rnorm(n, 0, .5)

#Run localnulltest with 10 knots
fit0= localnulltest(y, x=x, z=z, nlocalknots=10, niter=1000)

#Estimated covariate effects and posterior probabilities
b= coef(fit0)
b
```

`marginalLikelihood` *Marginal (or integrated) likelihood density of the observed data for an individual model handled by modelSelection (regression, GLM, GAM, accelerated failure time, regression with Normal, two-piece Normal, Laplace or two-piece Laplace residuals*

Description

The marginal density of the data, i.e. the likelihood integrated with respect to the prior distribution on the regression coefficients of the variables included in the model.

Usage

```
marginalLikelihood(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),
```

```
neighbours,
phi, method='auto', adj.overdisp='intercept', hess='asymp', optimMethod,
optim_maxit, initpar='none', 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	Parametric distribution family. Possible values include 'normal','laplace', 'binomial', 'poisson', see function modelSelection for the full list
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'.
neighbours	Only used if priorCoef is an icarplus prior. neighbours is a list with the same length as the design matrix. Its entry j should be a vector indicating the neighbours of j, and have 0 length if j has no neighbours.
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	If the disperson parameter (e.g. error variance) is known, it can be specified here. Leave blank unless you know what you're doing

Details

The marginal density of the data y under a given model is

p(y | model) int p(y | theta) d P(theta | model)

where P(theta | model) is the prior distribution on the parameters included by the model.

Value

Marginal (or integrated) likelihood of the data under the specified prior.

Author(s)

David Rossell

See Also

`modelSelection` to perform model selection based on product non-local priors.

Examples

```
x <- matrix(rnorm(100*2),ncol=2)
y <- x %*% matrix(c(.5,1),ncol=1) + rnorm(nrow(x))

#Marginal likelihood for 2 models under pMOM prior
marginalLikelihood(c(TRUE,FALSE), y=y, x=x, priorCoef=momprior())
marginalLikelihood(c(TRUE, TRUE), y=y, x=x, priorCoef=momprior())
```

```
#Same, under Normal prior with diagonal covariance
marginalLikelihood(c(TRUE,FALSE), y=y, x=x, priorCoef=normalidprior())
marginalLikelihood(c(TRUE, TRUE), y=y, x=x, priorCoef=normalidprior())
```

`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 generalized linear and generalized additive models.*

Description

Bayesian model selection for linear, asymmetric linear, median and quantile regression, for GLMs and GAMs 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.

modelSelection performs standard Bayesian model selection.

modelSelection_eBayes uses empirical Bayes to set prior inclusion probabilities pi. pi can be regressed on meta-covariates Z via log(pi/(1-pi))= Z w. If Z contains only an intercept, a global prior inclusion probability is learn from data.

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

Usage

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

modelSelection_eBayes(Z, wini, niter.mcmc= 5000, niter.mstep= 1000,
niter.eBayes= 20, priorvar.w, verbose=TRUE, ...)

modelsearchBlockDiag(y, x, priorCoef=momprior(tau=0.348),
priorModel=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 groups is 1:ncol(x), i.e. each variable goes into 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, i.e. variable selection is not performed for these variables. By default includevars is rep(FALSE,ncol(x))

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

priorModel — 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'.

`neighbours` Only used if priorCoef is an icarplus prior. neighbours is a list with the same length as the design matrix. Its entry j should be a vector indicating the neighbours of j, and have 0 length if j has no neighbours.

`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 being zero, except those indicated by `includevars`

`initSearch` Algorithm to refine `deltaini`. `initSearch=='CDA'` uses a coordinate descent algorithm. `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', 'L1', or a numeric vector with initial values. 'auto': if p<n/2 MLE is used, else L1 (regularization parameter set via BIC)

`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). By default `XtXprecomp <- ifelse(ncol(x)<10^4,TRUE,FALSE)`

`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(y|model) 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
Z	p x q matrix containing the q meta-covariates for the p covariates. An intercept is automatically added (including when Z is missing)
wini	Optional. Initial value for the q-dimensional hyper-parameter w
niter.mcmc	Number of iterations in the final MCMC, run once after hyper-parameter estimates have been obtained
niter.mstep	Number of MCMC iterations in each M-step required to update hyper-parameter estimates
niter.eBayes	Max number of iterations in the empirical Bayes optimization algorithm. The algorithm also stop when the objective function didn't improve in >=2 iterations
priorvar.w	Hyper-parameters w follow a prior w ~ N(0, priorvar.w (Z^T Z/p)^(-1)) where priorvar.w is the prior variance. By default is set such that all prior inclusion probabilities lie in (0.001,0.999) with 0.95 prior probability
...	Further parameters passed onto modelSelection

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. `plot.msfit` for an MCMC diagnostic plot showing estimated marginal posterior inclusion probabilities vs. iteration number. `rnlp` to obtain posterior samples for the coefficients. `marginalLikelihood` 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)
```

```
df <- data.frame(y, x)

#Specify prior parameters
priorCoef <- momprior()
priorModel <- modelunifprior()

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

#Alternative: Beta-Binomial prior for model space
priorModel <- modelbbprior()

#Model selection
fit1 <- modelSelection(y ~ ., data=df,
priorCoef=priorCoef, priorModel=priorModel)

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

postProb(fit1) #posterior model probabilities
```

`modelSelectionGGM` *Bayesian variable selection for Gaussian graphical models*

Description

Bayesian model selection and averaging for Gaussian graphical models.

modelSelectionGGM performs standard BMS/BMA under the specified priors

modelSelectionGGM_eBayes uses empirical Bayes to set edge prior inclusion probabilities pi. By default a common prior inclusion probability is set on all edges. If Z is specified, the edge inclusion probabilities can be regressed on meta-covariates Z via log(pi/(1-pi))= Z w.

Usage

```
modelSelectionGGM(y, priorCoef=normalidprior(tau=1),
priorModel=modelbinomprior(1/ncol(y)),
priorDiag=exponentialprior(lambda=1), center=TRUE, scale=TRUE,
global_proposal= 'none', prob_global=0.5,
tempering= 0.5, truncratio= 100,
save_proposal= FALSE, niter=10^3, burnin= round(niter/10),
updates_per_iter= ceiling(sqrt(ncol(y))), updates_per_column= 10,
sampler='Gibbs', pbirth=0.75, pdeath=0.5*(1-pbirth),
bounds_LIT, Omegaini='glasso-ebic', verbose=TRUE)

modelSelectionGGM_eBayes(Z, wini, niter.mcmc= 5000,
niter.mstep= 1000, niter.eBayes= 20, priorvar.w,
verbose=TRUE, ...)
```

Arguments

y	Data matrix
priorCoef	Prior on off-diagonal entries of the precision matrix, conditional on their not being zero (slab). See details
priorModel	Prior probabilities on having non-zero diagonal entries. Possible priors are modelbinomprior() and modelbbprior(). See details. See also modelSelectionGGM_eBayes for an empirical Bayes model prior
priorDiag	Prior on diagonal entries of the precision matrix. See details.
center	If TRUE, the columns of y will be centered to zero mean
scale	If TRUE, the columns of y will be scaled to unit sample variance
global_proposal	Either 'none', 'regression' or 'in-sample'. If 'none', serial MCMC is used as specified by sampler. If 'regression', MCMC uses Metropolis-Hastings where models are proposed for each column using regression posterior model probabilities for that column. If 'in-sample', each column's proposal use posterior model probabilities given a precision matrix estimate for the other columns
prob_global	Probability of proposing a sample from a global proposal, see details. This argument is ignored if global_proposal == "none".
tempering	If global_proposal != 'none', the posterior model probabilities of the proposal distribution are raised to the power indicated by tempering (set to 1 for no tempering)
truncratio	In the global proposal, any model's proposal probability is >= prob(top model) / truncratio. This ensures bounded weight ratios in the MH step, to improve poor mixing when the current state has low proposal probability, often at the cost of decreasing the acceptance rate. If truncratio <= 0, no truncation is done
save_proposal	If TRUE, the global proposals are saved in proposal (a list with p entries, one per column) and the corresponding proposal densities in proposaldensity. Neither are typically needed, as they were already used to produce the posterior samples in postSample
sampler	Posterior sampler used when global=="none", and also to run the parallel proposals when global!="none". Options are "Gibbs" for Gibbs sampling,"birthdeath" for birth-death-swap, and "LIT" for the locally-informed truncated algorithm of Zhou et al (2022)
niter	Number of posterior samples to be obtained. Each iteration consists of selecting a column of the precision matrix at random and making updates_per_column updates to its entries
burnin	The first burnin samples will be discarded
updates_per_iter	An iteration consists of selecting updates_per_iter columns at random, and proposing updates_per_column edge updates within each column
updates_per_column	See updates_per_iter

pbirth	Probability of a birth move. The probability of a swap move is 1-pbirth-pdeath. Ignored unless sampler=="birthdeath"
pdeath	Probability of a death move. Ignored unless sampler=="birthdeath"
bounds_LIT	log-proposal density bound for the locally-informed LIT algorithm of Zhou et al. These bound the proposal density for a death move and for a birth move. By default, bounds_LIT is log(c("lbound_death"=1/p, "ubound_death"= 1, "lbound_birth"=1/p, "ubound_birth"=p))
Omegaini	Initial value of the precision matrix Omega. "null" sets all off-diagonal entries to 0. "glasso-bic" and "glasso-ebic" use GLASSO with regularization parameter set via BIC/EBIC, respectively. Alternatively, Omegaini can be a matrix
verbose	Set verbose==TRUE to print iteration progress
Z	p x q matrix containing the q meta-covariates for the p covariates. An intercept is automatically added (including when Z is missing)
wini	Optional. Initial value for the q-dimensional hyper-parameter w
niter.mcmc	Number of iterations in the final MCMC, run once after hyper-parameter estimates have been obtained
niter.mstep	Number of MCMC iterations in each M-step required to update hyper-parameter estimates
niter.eBayes	Max number of iterations in the empirical Bayes optimization algorithm. The algorithm also stop when the objective function didn't improve in >=2 iterations
priorvar.w	Hyper-parameters w follow a prior w ~ N(0, priorvar.w (Z^T Z/p)^(-1)) where priorvar.w is the prior variance. By default is set such that all prior inclusion probabilities lie in (0.001,0.999) with 0.95 prior probability
...	Further parameters passed onto modelSelection

Details

Let Omega be the inverse covariance matrix. A spike-and-slab prior is used. Specifically, independent priors are set on all Omega[j,k], and then a positive-definiteness truncation is added.

The prior on diagonal entries Omega[j,j] is given by priorDiag. Specificaly, if priorDiag = exponentialprior(lambda) then the prior is Omega[j,j] ~ Exp(lambda).

Off-diagonal Omega[j,k] are equal to zero with probability given by modelPrior and otherwise arises from the distribution indicated in priorCoef (slab). If priorCoef == normalidprior(tau) then Omega[j,k] ~ N(0, var=tau).

If modelPrior == modelbinomprior(p) then Omega[j,k] = 0 with probability p independently across j < k. If modelPrior == modelbbprior(a,b) then p ~ Beta(a,b).

Inference is based on MCMC posterior sampling. All sampling algorithms proceed by updating Omega[,k] given y and Omega[,-k] (of course, Omega[k,] is also set to Omega[,k]). Omega[,k] is updated by first updating the set of non-zero entries (i.e. edges in the graphical model) using either Gibbs sampling or a proposal distribution (see below), and then the non-zero entries of Omega[,k] are updated from their exact posterior given the current set of edges.

An MCMC iteration consists of iterating over updates_per_iter columns chosen at random and, for each column, do updates_per_column proposals.

If `global_proposal == "none"`, a local MCMC proposal is used to update what entries in Omega[,k] are zero. Local means that the proposed model is a neighbour of the current model, e.g. only one edge is added / killed. Available local kernels are Gibbs, birth-death-swap and LIT (Zhou et al 2022).

If `global_proposal == "regression"`, a global proposal is used. A new model (set of non-zero entries in Omega[,k]) is proposed based on the posterior distribution of a linear regression of y[,k] on the other covariates. `prob_global` indicates the probability of using the global proposal, otherwise a local proposal is used. Proposal probabilities are tempered by raising them to the power `tempering`. Further, any model with probability below prob(top model) / truncratio is assigned proposal probability prob(top model) / truncratio.

Value

Posterior inference on the inverse covariance of `y`. Object of class `msfit_ggm`, which extends a list with elements

`postSample`	Posterior samples for the upper-diagonal entries of the precision matrix. Stored as a sparse matrix, see package Matrix to utilities to work with such matrices
`prop_accept`	If `almost_parallel` is `TRUE`, a vector with the proportion of accepted edge proposals. Note that Omega[,k] is always updated from its exact conditional posterior, regardless of the edge proposal being accepted or rejected.
`proposal`	If `almost_parallel` and `save_proposal` are `TRUE`, this is a list with one entry per column of Omega, containing the proposed values of each column
`proposaldensity`	log-proposal density for the samples in `proposal`. Entry (i,j) stores the log-proposal density for proposed sample i of column j
`margpp`	Rao-Blackwellized estimates of posterior marginal inclusion probabilities. Only valid when using the Gibbs algorithm
`priors`	List storing the priors specified when calling `modelSelectionGGM`
`p`	Number of columns in `y`
`indexes`	Indicates what row/column of Omega is stored in each column of `postSample`
`samplerPars`	MCMC sampling parameters
`almost_parallel`	Stores the input argument `almost_parallel`

Author(s)

David Rossell

References

Zhou, Quan, et al. Dimension-free mixing for high-dimensional Bayesian variable selection. JRSS-B 2022, 84, 1751-1784

See Also

`msfit_ggm-class` for further details on the output. `icov` for the estimated precision (inverse covariance) matrix. `coef.msfit_ggm` for Bayesian model averaging estimates and intervals.

Examples

```
#Simulate data with p=3
Th= diag(3); Th[1,2]= Th[2,1]= 0.5
sigma= solve(Th)

z= matrix(rnorm(1000*3), ncol=3)
y= z

#Obtain posterior samples
#y has few columns, initialize the MCMC at the sample precision matrix
#Else, leave the argument Omegaini in modelSelectionGGM empty
Omegaini= solve(cov(y))
fit= modelSelectionGGM(y, scale=FALSE, Omegaini=Omegaini)

#Parameter estimates, intervals, prob of non-zero
coef(fit)

#Estimated inverse covariance
icov(fit)

#Estimated inverse covariance, entries set to 0
icov(fit, threshold=0.95)

#Shows first posterior samples
head(fit$postSample)
```

`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

plot Shows estimated posterior inclusion probability for each parameter vs. number of MCMC iterations

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")
```

`msfit_ggm-class` *Class "msfit_ggm"*

Description

Stores the output of Bayesian Gaussian graphical model selection and averaging, as produced by function `modelSelectionGGM`. The class extends a list, so all usual methods for lists also work for `msfit_ggm` objects, e.g. accessing elements, retrieving names etc.

Methods are provided to obtain parameter estimates, posterior intervals (Bayesian model averaging), and posterior probabilities of parameters being non-zero

Objects from the Class

Objects are created by a call to `modelSelectionGGM`.

Slots

The class extends a list with elements:

postSample Sparse matrix (`dgCMatrix`) with posterior samples for the Gaussian precision (inverse covariance) parameters. Each row is a posterior sample. Within each row, only the upper-diagonal of the precision matrix is stored in a flat manner. The row and column indexes are stored in indexes

indexes For each column in postSample, it indicates the row and column of the precision matrix

p Number of variables

priors Priors specified when calling `modelSelection`

Methods

coef Obtain BMA posterior means, intervals and posterior probability of non-zeroes

plot Shows estimated posterior inclusion probability for each parameter vs. number of MCMC iterations. Only up to the first 5000 parameters are shown

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

Author(s)

David Rossell

See Also

modelSelectionGGM

Examples

```
showClass("msfit_ggm")
```

---

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

```
aic()
bic()
bicprior()
ic(penalty)

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)
icarplusprior(a=0.5, taustd=1, tau.adj=10^6)

exponentialprior(lambda = 1)

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

penalty	Penalty on model dimension, i.e. for the AIC penalty=2
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
a	The icarplus prior precision matrix is a P + (1-a) tau I, where P is an ICAR precision matrix and tau I a normalidprior precision matrix
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	igprior defines an inverse gamma prior with parameters alpha/2, lambda/2. exponentialprior defines an exponential prior with rate parameter lambda

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, normalid and icarplus 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 the 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.

Rossell D, Abril O, Bhattacharya A. Approximate Laplace approximations for scalable model selection (2021). Journal of the Royal Statistical Society B, 83, 4, 853-879

See Also

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

Examples

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

---

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)
```

---

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, class mixturebf returned by bfnormmix, class cilfit returned by cil or class localtest returned by localnulltest
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)

For msfit_ggm objects, a list with posterior probabilities (element "pp") and the corresponding model (element "modelid") indicating what edges are non-zero

For mixturebf objects, posterior probabilities for the specified number of components

For localtest objects, posterior probabilities of a local covariate effect at various regions

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

```
#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, priorprec, 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
priorprec	Prior precision matrix (inverse covariance). Currently, this is only needed for the ICAR+ prior
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

∗ **cil**
cil, 8
∗ **classes**
icfit-class, 19
mixturebf-class, 28
msfit-class, 39
msfit_ggm-class, 41
msPriorSpec-class, 42
∗ **distribution**
bbPrior, 3
bestBIC, 4
dalapl, 12
ddir, 13
diwish, 14
dmom, 15
dpostNIW, 17
postProb, 46
postSamples, 47
priorp2g, 48
rnlp, 49
∗ **distrib**
eprod, 18
∗ **hplot**
plotprior, 45
∗ **htest**
bfnormmix, 6
dmom, 15
icov, 20
localnulltest, 21
marginalLikelihood, 24
marginalNIW, 27
modelSelection, 29
modelSelectionGGM, 35
priorp2g, 48
∗ **models**
bestBIC, 4
bfnormmix, 6
cil, 8
eprod, 18
icov, 20
localnulltest, 21
marginalLikelihood, 24
marginalNIW, 27
modelSelection, 29
modelSelectionGGM, 35
plotprior, 45
postProb, 46
postSamples, 47
rnlp, 49

aic *(msPriorSpec-class)*, 42

bbPrior, 3
bestAIC *(bestBIC)*, 4
bestAIC_fast *(bestBIC)*, 4
bestBIC, 4, *20*
bestBIC_fast *(bestBIC)*, 4
bestEBIC *(bestBIC)*, 4
bestEBIC_fast *(bestBIC)*, 4
bestIC *(bestBIC)*, 4
bestIC_fast *(bestBIC)*, 4
bfnormmix, 6, *29*
bic *(msPriorSpec-class)*, 42
bicprior *(msPriorSpec-class)*, 42
binomPrior *(bbPrior)*, 3

cil, 8, *46*
coef.mixturebf *(mixturebf-class)*, 28
coefByModel *(msfit-class)*, 39
coefByModel,msfit-method *(msfit-class)*, 39
coefByModel-methods *(msfit-class)*, 39

dalapl, 12
ddir, 13
demom *(dmom)*, 15
demom,data.frame-method *(dmom)*, 15
demom,matrix-method *(dmom)*, 15
demom,vector-method *(dmom)*, 15

demom-methods (dmom), 15
demomigmarg (dmom), 15
dimom (dmom), 15
diwish, 14, *18*
dmom, 15
dmomigmarg (dmom), 15
dpostNIW, *15*, 17, *28*

emomprior (msPriorSpec-class), 42
eprod, 18
exponentialprior (msPriorSpec-class), 42

groupemomprior (msPriorSpec-class), 42
groupimomprior (msPriorSpec-class), 42
groupmomprior (msPriorSpec-class), 42
groupzellnerprior (msPriorSpec-class), 42

ic (msPriorSpec-class), 42
icarplusprior (msPriorSpec-class), 42
icfit (icfit-class), 19
icfit-class, 19
icfit.coef (icfit-class), 19
icfit.predict (icfit-class), 19
icfit.summary (icfit-class), 19
icov, 20
igprior (msPriorSpec-class), 42
imomprior (msPriorSpec-class), 42

localnulltest, 21
localnulltest_fda (localnulltest), 21
localnulltest_fda_givenknots (localnulltest), 21
localnulltest_givenknots (localnulltest), 21

marginalLikelihood, 24, *34*
marginalNIW, *18*, 27
marginalNIW,matrix,missing,missing,missing,missing-method (marginalNIW), 27
marginalNIW,matrix,missing,missing,missing,vector-method (marginalNIW), 27
marginalNIW,missing,ANY,matrix,numeric,missing-method (marginalNIW), 27
marginalNIW,missing,list,list,numeric,missing-method (marginalNIW), 27
marginalNIW-methods (marginalNIW), 27
mixturebf (mixturebf-class), 28
mixturebf-class, 28

modelbbprior (msPriorSpec-class), 42
modelbinomprior (msPriorSpec-class), 42
modelcomplexprior (msPriorSpec-class), 42
modelsearchBlockDiag (modelSelection), 29
modelSelection, *6*, *26*, 29, *41*, *45*, *47*, *50*
modelSelection_eBayes (modelSelection), 29
modelSelectionGGM, *21*, 35, *42*
modelSelectionGGM_eBayes (modelSelectionGGM), 35
modelunifprior (msPriorSpec-class), 42
momprior (msPriorSpec-class), 42
msfit (msfit-class), 39
msfit-class, 39
msfit.coef (msfit-class), 39
msfit.plot (msfit-class), 39
msfit.predict (msfit-class), 39
msfit_ggm (msfit_ggm-class), 41
msfit_ggm-class, 41
msfit_ggm.coef (msfit_ggm-class), 41
msPriorSpec (msPriorSpec-class), 42
msPriorSpec-class, 42

normalidprior (msPriorSpec-class), 42

palapl (dalapl), 12
pemom (dmom), 15
pemomigmarg (dmom), 15
pimom, *48*
pimom (dmom), 15
plotprior, 45
plotprior,cilfit-method (plotprior), 45
plotprior-methods (plotprior), 45
pmom, *48*
pmom (dmom), 15
pmomigmarg (dmom), 15
postGrid, *12*, *34*, 46
postProb,cilfit-method (postProb), 46
postProb,localtest-method (postProb), 46
postProb,mixturebf-method (postProb), 46
postProb,msfit-method (postProb), 46
postProb,msfit_ggm-method (postProb), 46
postProb-methods (postProb), 46
postSamples, 47
postSamples,mixturebf-method (postSamples), 47
postSamples-methods (postSamples), 47

priorp2g, 48

qimom *(*dmom*)*, 15
qmom *(*dmom*)*, 15

ralapl *(*dalapl*)*, 12
rnlp, *34*, *41*, 49
rnlp,ANY,matrix,missing,missing,missing,character,character-method
*(*rnlp*)*, 49
rnlp,ANY,matrix,missing,missing,missing,missing,missing-method
*(*rnlp*)*, 49
rnlp,ANY,matrix,missing,missing,msfit,missing,missing-method
*(*rnlp*)*, 49
rnlp,missing,missing,missing,missing,msfit,missing,missing-method
*(*rnlp*)*, 49
rnlp,missing,missing,numeric,matrix,missing,missing,missing-method
*(*rnlp*)*, 49
rnlp-methods *(*rnlp*)*, 49
rpostNIW *(*dpostNIW*)*, 17

show,icfit-method *(*icfit-class*)*, 19
show,mixturebf-method
*(*mixturebf-class*)*, 28
show,msfit-method *(*msfit-class*)*, 39
show,msfit_ggm-method
*(*msfit_ggm-class*)*, 41

unifPrior *(*bbPrior*)*, 3

zellnerprior *(*msPriorSpec-class*)*, 42