Package ‘BoomSpikeSlab’

September 4, 2019

Date 2019-09-01
Title MCMC for Spike and Slab Regression
Author Steven L. Scott <steve.the.bayesian@gmail.com>
Maintainer Steven L. Scott <steve.the.bayesian@gmail.com>
Description Spike and slab regression with a variety of residual error
distributions corresponding to Gaussian, Student T, probit, logit, SVM, and a
few others. Spike and slab regression is Bayesian regression with prior
distributions containing a point mass at zero. The posterior updates the
amount of mass on this point, leading to a posterior distribution that is
actually sparse, in the sense that if you sample from it many coefficients are
actually zeros. Sampling from this posterior distribution is an elegant way
to handle Bayesian variable selection and model averaging. See
<DOI:10.1504/IJMMNO.2014.059942> for an explanation of the Gaussian case.

Depends Boom (>= 0.9.2) , R (>= 3.1.0)
LinkingTo Boom(>= 0.9.2)
Suggests MASS, testthat, mlbench
Imports igraph
Version 1.2.1
License LGPL-2.1 | file LICENSE
Encoding UTF-8
NeedsCompilation yes
Repository CRAN
Date/Publication 2019-09-04 05:10:25 UTC

R topics documented:

independent.spike.slab.prior ................................. 2
independent.student.spike.slab.prior .......................... 5
lm.spike .................................................. 7
logit.spike ............................................... 10
logit.zellner.prior ................................. 13
independent.spike.slab.prior

A spike and slab prior assuming a priori independence.

Description

A spike and slab prior on the regression coefficients. The prior distribution assumes coefficients to be independent.
Usage

IndependentSpikeSlabPrior(x = NULL,
y = NULL,
  expected.r2 = .5,
prior.df = .01,
  expected.model.size = 1,
prior.beta.sd = NULL,
  optional.coefficient.estimate = NULL,
  mean.y = mean(y, na.rm = TRUE),
  sdy = sd(as.numeric(y), na.rm = TRUE),
  sdx = apply(as.matrix(x), 2, sd, na.rm = TRUE),
  prior.inclusion.probabilities = NULL,
  number.of.observations = nrow(x),
  number.of.variables = ncol(x),
  scale.by.residual.variance = FALSE,
  sigma.upper.limit = Inf)

Arguments

  x  The design matrix for the regression problem. Missing data is not allowed.
  y  The vector of responses for the regression. Missing data is not allowed.
expected.r2  The expected R-square for the regression. The spike and slab prior requires an
  inverse gamma prior on the residual variance of the regression. The prior can
  be parameterized in terms of a guess at the residual variance, and a "degrees of
  freedom" representing the number of observations that the guess should weigh.
  The guess at $\sigma^2$ is set to $(1-\text{expected.r2}) \times \text{var(y)}$.
prior.df  A positive scalar representing the prior 'degrees of freedom' for estimating the
  residual variance. This can be thought of as the amount of weight (expressed as
  an observation count) given to the expected.r2 argument.
expected.model.size  A positive number less than ncol(x), representing a guess at the number of
  significant predictor p variables. Used to obtain the 'spike' portion of the spike
  and slab prior.
prior.beta.sd  A vector of positive numbers giving the prior standard deviation of each model
  coefficient, conditional on inclusion. If NULL it will be set to 10 * the ratio of
  sdy / sdx.
optional.coefficient.estimate  If desired, an estimate of the regression coefficients can be supplied. In most
  cases this will be a difficult parameter to specify. If omitted then a prior mean
  of zero will be used for all coordinates except the intercept, which will be set to
  mean(y).
mean.y  The mean of the response vector, for use in cases when specifying the response
  vector is undesirable.
sdy  The standard deviation of the response vector, for use in cases when specifying
  the response vector is undesirable.
sdx  The standard deviations to use when scaling the prior sd of each coefficient.
prior.inclusion.probabilities
   A vector giving the prior probability of inclusion for each variable.

number.of.observations
   The number of observations in the data to be modeled.

number.of.variables
   The number of potential predictor variables in the data to be modeled.

scale.by.residual.variance
   If TRUE the prior variance is $\sigma^2 \cdot V$, where $\sigma^2$ is the residual variance of the linear regression modeled by this prior. Otherwise the prior variance is $V$, unscaled.

sigma.upper.limit
   The largest acceptable value for the residual standard deviation. A non-positive number is interpreted as $\infty$.

Value
   A list with with the components necessary to run \texttt{lm.spike} with method "DA".

Author(s)
   Steven L. Scott

References

Examples

```r
x <- cbind(1, matrix(rnorm(900), ncol = 9))
beta <- rep(0, 10)
beta[1] <- 3
beta[5] <- -4
beta[8] <- 2
y <- rnorm(100, x %*% beta)
## x has 10 columns, including the intercept
prior <- IndependentSpikeSlabPrior(x, y,
expected.model.size = 3, # expect 3 nonzero predictors
prior.df = .01, # weaker prior than the default
optional.coefficientestimate = rep(0, 10) # shrink to zero
)
## now 'prior' can be fed to 'lm.spike'
x <- x[, -1]
model <- lm.spike(y ~ x, niter = 1000, prior = prior, model.options = OdaOptions())
```
Description

A spike and slab prior on the parameters of a regression model with Student T errors. The prior assumes independence among the regression coefficients.

Usage

StudentIndependentSpikeSlabPrior(
    predictor.matrix = NULL,
    response.vector = NULL,
    expected.r2 = .5,
    prior.df = .01,
    expected.model.size = 1,
    prior.beta.sd = NULL,
    optional.coefficient.estimate = NULL,
    mean.y = mean(response.vector, na.rm = TRUE),
    sdy = sd(as.numeric(response.vector), na.rm = TRUE),
    sdx = apply(as.matrix(predictor.matrix), 2, sd, na.rm = TRUE),
    prior.inclusion.probabilities = NULL,
    number.of.observations = nrow(predictor.matrix),
    number.of.variables = ncol(predictor.matrix),
    scale.by.residual.variance = FALSE,
    sigma.upper.limit = Inf,
    degrees.of.freedom.prior = UniformPrior(.1, 100)
)

Arguments

predictor.matrix
    The design matrix for the regression problem. Missing data is not allowed.

response.vector
    The vector of responses for the regression. Missing data is not allowed.

expected.r2
    The expected R-square for the regression. The spike and slab prior requires an inverse gamma prior on the residual variance of the regression. The prior can be parameterized in terms of a guess at the residual variance, and a "degrees of freedom" representing the number of observations that the guess should weigh. The guess at sigma^2 is set to (1-expected.r2) * var(y).

prior.df
    A positive scalar representing the prior 'degrees of freedom' for estimating the residual variance. This can be thought of as the amount of weight (expressed as an observation count) given to the expected.r2 argument.

expected.model.size
    A positive number less than ncol(x), representing a guess at the number of significant predictor p variables. Used to obtain the 'spike' portion of the spike and slab prior.
prior.beta.sd       A vector of positive numbers giving the prior standard deviation of each model
                    coefficient, conditionl on inclusion. If NULL it will be set to 10 * the ratio of
                    sdy / sdx.
optional.coefficient.estimate
                    If desired, an estimate of the regression coefficients can be supplied. In most
                    cases this will be a difficult parameter to specify. If omitted then a prior mean
                    of zero will be used for all coordinates except the intercept, which will be set to
                    mean(y).
mean.y             The mean of the response vector, for use in cases when specifying the response
                    vector is undesirable.
sdy                 The standard deviation of the response vector, for use in cases when specifying
                    the response vector is undesirable.
sdx                 The standard deviations to use when scaling the prior sd of each coefficient.
prior.inclusion.probabilities
                    A vector giving the prior probability of inclusion for each variable.
number.of.observations
                    The number of observations in the data to be modeled.
number.of.variables
                    The number of potential predictor variables in the data to be modeled.
scale.by.residual.variance
                    If TRUE the prior variance is sigma_sq * V, where sigma_sq is the residual vari-
                    ance of the linear regression modeled by this prior. Otherwise the prior variance
                    is V, unscaled.
sigma.upper.limit
                    The largest acceptable value for the residual standard deviation. A non-positive
                    number is interpreted as Inf.
degrees.of.freedom.prior
                    An object of class DoubleModel representing the prior distribution for the Stu-
                    dent T tail thickness (or "degrees of freedom") parameter.

Value

An IndependentSpikeSlabPrior with degrees.of.freedom.prior appended.

Author(s)

Steven L. Scott

References

Ghosh and Clyde (2011) "Rao-Blackwellization for Bayesian variable selection and model averag-
ing in linear and binary regression: A novel data augmentation approach", Journal of the Ameri-
clyde_2011_jasa.pdf
**lm.spike**

**Spike and slab regression**

**Description**

MCMC algorithm for linear regression models with a 'spike-and-slab' prior that places some amount of posterior probability at zero for a subset of the regression coefficients.

The model admits either Gaussian or student T errors; the latter are useful in the presence of outliers.

**Usage**

```r
lm.spike(formula, niter, data, subset, prior = NULL, error.distribution = c("gaussian", "student"), contrasts = NULL, drop.unused.levels = TRUE, model.options = SsvsOptions(), ping = niter / 10, seed = NULL, ...)
```

```r
SsvsOptions(adaptive.cutoff = 100, adaptive.step.size = .001, target.acceptance.rate = .345, correlation.swap.threshold = .8)
```

```r
OdaOptions(fallback.probability = 0.0, eigenvalue.fudge.factor = 0.01)
```

**Arguments**

- `formula` formula for the maximal model (with all variables included), this is parsed the same way as a call to `lm`.
- `niter` The number of MCMC iterations to run. Be sure to include enough so you can throw away a burn-in set.
- `data` An optional data frame, list or environment (or object coercible by `as.data.frame` to a data frame) containing the variables in the model. If not found in `data`, the variables are taken from `environment(formula)`, typically the environment from which `lm.spike` is called.
- `subset` An optional vector specifying a subset of observations to be used in the fitting process.
- `prior` An optional list returned by `SpikeSlabPrior`. If prior is missing then a default prior will be used. See `SpikeSlabPrior`.
error.distribution
   Specify either Gaussian or Student T errors. If the error distribution is student
   then the prior must be a StudentSpikeSlabPrior and the SSVS method must
   be used.

contrasts
   An optional list. See the contrasts.arg argument of model.matrix.default.

drop.unused.levels
   Logical indicating whether unobserved factor levels should be dropped from the
   model.

model.options
   A list containing the tuning parameters for the desired MCMC method.

ping
   The frequency with which to print status update messages to the screen. For ex-
   ample, if ping == 10 then an update will be printed every 10 MCMC iterations.

seed
   An integer to use as the random seed for the underlying C++ code. If NULL then
   the seed will be set using the clock.

... 
   Extra arguments to be passed to SpikeSlabPrior (if method == "SSVS") or
   IndependentSpikeSlabPrior (if method == "ODA").

fallback.probability
   When using the ODA method, each MCMC iteration will use SSVS instead of
   ODA with this probability. In cases where the latent data have high leverage,
   ODA mixing can suffer. Mixing in a few SSVS steps can help keep an errant
   algorithm on track.

eigenvalue.fudge.factor
   When using the ODA method, the latent X’s will be chosen so that the complete
   data X’X matrix (after scaling) is a constant diagonal matrix equal to the largest
   eigenvalue of the observed (scaled) X’X times (1 + eigenvalue.fudge.factor). This
   should be a small positive number.

adaptive.cutoff
   The traditional SSVS method (sample every predictor at every iteration) will be
   used when there are fewer than this many predictors. The adaptive method of
   Benson and Fried will be used if there are more.

adaptive.step.size
   The step size scaling factor to use in the adaptive SSVS algorithm.

target.acceptance.rate
   The target acceptance rate for the adaptive SSVS algorithm.

correlation.swap.threshold
   The minimal absolute correlation required for two variables to be considered for
   a swap move. Swap moves are currently only supported for less than adaptive.cutoff
   variables.

Details

There are two MCMC methods available. SSVS is the stochastic search variable selection algorithm
from George and McCulloch (1998). ODA is the orthogonal data augmentation method from Clyde
and Ghosh (2011). Both sampling methods ("ODA" and "SSVS") draw each variable inclusion
indicator given all others, in a Gibbs sampler. The ODA method includes an extra data augmentation
step that renders each indicator conditionally independent of the others given the latent data. There
is residual dependence between successive MCMC steps introduced by the latent data, but the paper
by Ghosh and Clyde suggested that on balance mixing should be improved.
SSVS offers a choice between two implementations. Classic SSVS attempts to flip each coefficient in or out of the model every iteration. The adaptive method attempts to learn which coefficients are likely to be included or excluded. It then biases its 'birth' and 'death' moves towards candidates that are likely to succeed.

Regarding the overall compute time, the DA method decomposes the (potentially very large) model matrix one time, at the start of the algorithm. But it then works with independent scalar updates. The SSVS algorithm does not have the upfront cost, but it works with many small matrix decompositions each MCMC iteration. The DA algorithm is very likely to be faster in terms of time per iteration.

Finally, note that the two algorithms require slightly different priors. The DA algorithm requires a priori independence, while the SSVS algorithm can work with arbitrary conjugate priors.

Value

Returns an object of class `lm.spike`, which is a list with the following elements

- **beta**: A `niter` by `ncol(x)` matrix of regression coefficients, many of which may be zero. Each row corresponds to an MCMC iteration.
- **sigma**: A vector of length `niter` containing the MCMC draws of the residual standard deviation parameter.
- **prior**: The prior used to fit the model. If a prior was supplied as an argument it will be returned. Otherwise this will be the automatically generated prior based on the other function arguments.

Author(s)

Steven L. Scott

References


See Also

*SpikeSlabPrior, plot.lm.spike, summary.lm.spike, predict.lm.spike.*

Examples

```r
n <- 100
p <- 10
ngood <- 3
niter <- 1000
sigma <- .8

x <- cbind(1, matrix(rnorm(n * (p-1)), nrow=n))
```
logit.spike

Spike and slab logistic regression

Description

MCMC algorithm for logistic regression models with a 'spike-and-slab' prior that places some amount of posterior probability at zero for a subset of the regression coefficients.

Usage

logit.spike(formula, niter, data, subset, prior = NULL, na.action = options("na.action"), contrasts = NULL,
logit.spike

drop.unused.levels = TRUE,
initial.value = NULL,
ping = niter / 10,
nthreads = 0,
clt.threshold = 2,
mh.chunk.size = 10,
proposal.df = 3,
sampler.weights = c("DA" = .333, "RWM" = .333, "TIM" = .333),
seed = NULL,
...)

**Arguments**

**formula**
- formula for the maximal model (with all variables included), this is parsed the same way as a call to `glm`, but no `family` argument is needed. Like `glm`, a two-column input format (success-count, failure-count). Otherwise, the response variable can be a logical or numeric vector. If numeric, then values >0 indicate a "success".

**niter**
- The number of MCMC iterations to run. Be sure to include enough so you can throw away a burn-in set.

**data**
- An optional data frame, list or environment (or object coercible by 'as.data.frame' to a data frame) containing the variables in the model. If not found in 'data', the variables are taken from 'environment(formula)', typically the environment from which logit.spike' is called.

**subset**
- An optional vector specifying a subset of observations to be used in the fitting process.

**prior**
- A n object inheriting from SpikeSlabGlmPrior. If prior is supplied it will be used. Otherwise a prior distribution will constructed by calling LogitZellnerPrior.

**na.action**
- A function which indicates what should happen when the data contain NAs. The default is set by the na.action setting of options, and is na.fail if that is unset. The factory-fresh default is na.omit. Another possible value is NULL, no action. Value na.exclude can be useful.

**contrasts**
- An optional list. See the contrasts.arg of model.matrix.default.

**drop.unused.levels**
- A logical value indicating whether factor levels that are unobserved should be dropped from the model.

**initial.value**
- Initial value for the MCMC algorithm. Can either be a numeric vector, a glm object (from which the coefficients will be used), or a logit.spike object. If a logit.spike object is supplied, it is assumed to be from a previous MCMC run for which niter additional draws are desired. If a glm object is supplied then its coefficients will be used as the initial values for the simulation.

**ping**
- If positive, then print a status update to the console every ping MCMC iterations.

**nthreads**
- The number of CPU-threads to use for data augmentation. There is some small overhead to stopping and starting threads. For small data sets, thread overhead will make it faster to run single threaded. For larger data sets multi-threading
can speed things up substantially. This is all machine dependent, so please experiment.

**clt.threshold** When the model is presented with binomial data (i.e. when the response is a two-column matrix) the data augmentation algorithm can be made more efficient by updating a single, asymptotically normal scalar quantity for each unique value of the predictors. The asymptotic result will be used whenever the number of successes or failures exceeds clt.threshold.

**mh.chunk.size** The maximum number of coefficients to draw in a single “chunk” of a Metropolis-Hastings update. See details.

**proposal.df** The degrees of freedom parameter to use in Metropolis-Hastings proposals. See details.

**sampler.weights** The proportion of MCMC iterations spent in each of the three algorithms described in the Details section. This must be a vector of length 3, with names "DA", "RWM" and "TIM", containing non-negative elements that sum to (within numerical error .999 or 1.001 are okay).

**seed** Seed to use for the C++ random number generator. It should be NULL or an int. If NULL the seed value will be taken from the global `.Random.seed` object.

... Extra arguments passed to `LogitZellnerPrior` in the case where prior is NULL.

### Details

Model parameters are updated using a composite of three Metropolis-Hastings updates. An auxiliary mixture sampling algorithm (Tuchler 2008) updates the entire parameter vector at once, but can mix slowly.

The second algorithm is a random walk Metropolis update based on a multivariate T proposal with `proposal.df` degrees of freedom. If `proposal.df` is nonpositive then a Gaussian proposal is used. The variance of the proposal distribution is based on the Fisher information matrix evaluated at the current draw of the coefficients.

The third algorithm is an independence Metropolis sampler centered on the posterior mode with variance determined by posterior information matrix (Fisher information plus prior information). If `proposal.df > 0` then the tails of the proposal are inflated so that a multivariate T proposal is used instead.

For either of the two MH updates, at most `mh.chunk.size` coefficients will be updated at a time. At each iteration, one of the three algorithms is chosen at random. The auxiliary mixture sampler is the only one that can change the dimension of the coefficient vector. The MH algorithms only update the coefficients that are currently nonzero.

### Value

Returns an object of class `logit.spike`, which inherits from `lm.spike`. The returned object is a list with the following elements

**beta** A `niter` by `ncol(x)` matrix of regression coefficients, many of which may be zero. Each row corresponds to an MCMC iteration.
The prior used to fit the model. If a prior was supplied as an argument it will be returned. Otherwise this will be the automatically generated prior based on the other function arguments.

Author(s)

Steven L. Scott

References


See Also

lm.spike, SpikeSlabPrior, plot.logit.spike, PlotLogitSpikeFitSummary, PlotLogitSpikeResiduals, summary.logit.spike, predict.logit.spike.

Examples

```r
if (requireNamespace("MASS")) {
  data(Pima.tr, package = "MASS")
  data(Pima.te, package = "MASS")
  pima <- rbind(Pima.tr, Pima.te)
  model <- logit.spike(type == "Yes" ~ ., data = pima, niter = 500)
  plot(model)
  plot(model, "fit")
  plot(model, "residuals")
  plot(model, "size")
  summary(model)
}
```

Description

A Zellner-style spike and slab prior for logistic regression models. See 'Details' for a definition.

Usage

```r
LogitZellnerPrior(
  predictors,
  successes = NULL,
  trials = NULL,
  prior.success.probability = NULL,
  expected.model.size = 1,
  prior.information.weight = .01,
  diagonal.shrinkage = .5,
```
optional.coefficient.estimate = NULL,
max.flips = -1,
prior.inclusion.probabilities = NULL)

Arguments

predictors The design matrix for the regression problem. No missing data is allowed.

successes The vector of responses, which can be 0/1, TRUE/FALSE, or 1/-1. This is only used to obtain the empirical overall success rate, so it can be left NULL if prior.success.probability is specified.

trials A vector of the same length as successes, giving the number of trials for each success count (trials cannot be less than successes). If successes is binary (or NULL) then this can be NULL as well, signifying that there was only one trial per experiment.

prior.success.probability The overall prior guess at the proportion of successes. This is used in two places. It is an input into the intercept term of the default optional.coefficient.estimate, and it is used as a weight for the prior information matrix. See 'Details'.

expected.model.size A positive number less than ncol(x), representing a guess at the number of significant predictor variables. Used to obtain the 'spike' portion of the spike and slab prior.

prior.information.weight A positive scalar. Number of observations worth of weight that should be given to the prior estimate of beta.

diagonal.shrinkage The conditionally Gaussian prior for beta (the "slab") starts with a precision matrix equal to the information in a single observation. However, this matrix might not be full rank. The matrix can be made full rank by averaging with its diagonal. diagonal.shrinkage is the weight given to the diagonal in this average. Setting this to zero gives Zellner's g-prior.

optional.coefficient.estimate If desired, an estimate of the regression coefficients can be supplied. In most cases this will be a difficult parameter to specify. If omitted then a prior mean of zero will be used for all coordinates except the intercept, which will be set to mean(y).

max.flips The maximum number of variable inclusion indicators the sampler will attempt to sample each iteration. If negative then all indicators will be sampled.

prior.inclusion.probabilities A vector giving the prior probability of inclusion for each variable. If NULL then a default set of probabilities is obtained by setting each element equal to min(1,expected.model.size / ncol(x)).

Details

A Zellner-style spike and slab prior for logistic regression. Denote the vector of coefficients by $\beta$, and the vector of inclusion indicators by $\gamma$. These are linked by the relationship $\beta_i \neq 0$ if $\gamma_i = 1$ and $\beta_i = 0$ if $\gamma_i = 0$. The prior is
\[ \beta | \gamma \sim N(b, V) \]
\[ \gamma \sim B(\pi) \]

where \( \pi \) is the vector of prior inclusion probabilities, and \( b \) is the optional coefficient estimate. Conditional on \( \gamma \), the prior information matrix is

\[ V^{-1} = \kappa ((1 - \alpha)x^Twx/n + \alpha \text{diag}(x^Twx/n)) \]

The matrix \( x^Twx \) is, for suitable choice of the weight vector \( w \), the total Fisher information available in the data. Dividing by \( n \) gives the average Fisher information in a single observation, multiplying by \( \kappa \) then results in \( \kappa \) units of "average" information. This matrix is averaged with its diagonal to ensure positive definiteness.

In the formula above, \( \kappa \) is prior information weight, \( \alpha \) is diagonal shrinkage, and \( w \) is a diagonal matrix with all elements set to prior success probability * (1 - prior success probability). The vector \( b \) and the matrix \( V^{-1} \) are both implicitly subscripted by \( \gamma \), meaning that elements, rows, or columns corresponding to gamma = 0 should be omitted.

**Value**

Returns an object of class LogitZellnerPrior, which is a list with data elements encoding the selected prior values. It inherits from LogitPrior, which implies that it contains an element prior success probability.

This object is intended for use with logit.spike.

**Author(s)**

Steven L. Scott

**References**


---

**mlm.spike**

*Spive and slab multinomial logistic regression*

**Description**

MCMC algorithm for multinomial logist models with a 'spike-and-slab' prior that places some amount of posterior probability at zero for a subset of the regression coefficients.
Usage

```r
mlm.spike(subject.formula,  
    choice.formula = NULL,  
    niter,  
    data,  
    choice.name.separator = ".",  
    contrasts = NULL,  
    subset,  
    prior = NULL,  
    ping = niter / 10,  
    proposal.df = 3,  
    rwm.scale.factor = 1,  
    nthreads = 1,  
    mh.chunk.size = 10,  
    proposal.weights = c("DA" = .5, "RWM" = .25, "TIM" = .25),  
    seed = NULL,  
    ...)  
```

Arguments

`subject.formula`  
A model formula describing the relationship between the response (which must be a factor) and the characteristics of the subjects associated with the decision process. If there are no subject-level predictors then `y ~ 1` will provide a model with a different intercept for each level of the response. If no intercepts are desired, use `y ~ 0`.

`choice.formula`  
A model formula describing the relationship between the response and the characteristics of the object being chosen. This can be left `NULL` if no choice-level characteristics are to be used in the model. The variables appearing on the right hand side must be stored in `data` with the name of response levels appended, and a character (choice.name.separator) used as a separator. For example, if "MPG" is one of the variables in the formula, and the response can assume values of "Toyota", "Honda", and "Chevy", then data must contain `MPG.Toyota`, `MPG.Honda`, and `MPG.Chevy`.

`niter`  
The number of MCMC iterations to run. Be sure to include enough so you can discard a burn-in set.

`data`  
A data frame containing the data referenced in `subject.formula` and `choice.formula` arguments. If `choice.formula` is `NULL` then this argument is optional, and variables will be pulled from the parent environment if it is omitted. If `choice.formula` is non-`NULL`, then data must be supplied. Each row in data represents a single observation containing the relevant data about both the subject making the choice, as well as about the items being chosen among. A variable measuring a choice characteristic must be present for each choice level in the response variable. The stems for the choice-variable names that measure the same concepts must be identical, and choice level must be appended as a suffix, separated by a "." character. Thus, if "HP" is a variable to be considered, and the response levels are "Toyota", "Honda", "Chevy", then the data must contain variables named "HP.Toyota", "HP.Honda", and "HP.Chevy".
choice.name.separator

The character used to separate the predictor names from the choice values for the choice-level predictor variables in 'data'.

contrasts

An optional list. See the contrasts.arg of model.matrix.default.

subset

An optional vector specifying a subset of observations to be used in the fitting process.

prior

An object of class IndependentSpikeSlabPrior. The portions of the prior distribution relating to the residual variance are not used.

A convenience function: MultinomialLogitSpikeSlabPrior is provided to help with the accounting headaches of vectorizing the subject.beta and choice.beta parameters.

ping

The frequency with which status updates are printed to the console. Measured in MCMC iterations. If ping < 0 then no status updates will be printed.

proposal.df

The "degrees of freedom" parameter that the Metropolis-Hastings algorithm should use for the multivariate T proposal distribution. If proposal.df <= 0 then a Gaussian proposal is used instead.

rwm.scale.factor

The scale factor to use for random walk Metropolis updates. See details.

nthreads

The number of CPU-threads to use for data augmentation.

mh.chunk.size

The maximum number of coefficients to draw in a single "chunk" of a Metropolis-Hastings update. See details.

proposal.weights

A vector of 3 probabilities (summing to 1) indicating the probability of each type of MH proposal during each iteration. The weights should be given names "DA", "RWM", and "TIM" for clarity.

seed

Seed to use for the C++ random number generator. It should be NULL or an int. If NULL the seed value will be taken from the global .Random.seed object.

...

Extra arguments to be passed to MultinomialLogitSpikeSlabPrior.

Details

Model Details:

A multinomial logit model has two sets of predictors: one measuring characteristics of the subject making the choice, and the other measuring characteristics of the items being chosen. The model can be written

\[ Pr(y[i] = m) \propto \exp(b_{subject}[i, m] * x_{subject}[i,] + b_{choice} * x_{choice}[i, m]) \]

The coefficients in this model are beta.subject and beta.choice. beta.choice is a subject.xdim by (‘nchoices’ - 1) matrix. Each row multiplies the design matrix produced by subject.formula for a particular choice level, where the first choice level is omitted (logically set to zero) for identifiability. beta.choice is a vector multiplying the design matrix produced by choice.formula, and there are ‘nchoices’ of such matrices.

The coefficient vector 'beta' is the concatenation c(bet(subject, beta.choice), where beta.subject is vectorized by stacking its columns (in the usual R fashion). This means that the first contiguous region of beta contains the subject-level coefficients for choice level 2.
MCMC Details: The MCMC algorithm randomly moves between three types of updates: data augmentation, random walk Metropolis (RWM), and tailored independence Metropolis (TIM).

- DA: Each observation in the model is associated with a set of latent variables that renders the complete data posterior distribution conditionally Gaussian. The augmentation scheme is described in Tuchler (2008). The data augmentation algorithm conditions on the latent data, and integrates out the coefficients, to sample the inclusion vector (i.e., the vector of indicators showing which coefficients are nonzero) using Gibbs sampling. Then the coefficients are sampled given complete data conditional on inclusion. This is the only move that attempts a dimension change.

- RWM: A chunk of the coefficient vector (up to mh.chunk.size) is selected. The proposal distribution is either multivariate normal or multivariate T (depending on 'proposal.df') centered on current values of this chunk. The precision parameter of the normal (or T) is the negative Hessian of the un-normalized log posterior, evaluated at the current value. The precision is divided by rwm.scale.factor. Only coefficients currently included in the model at the time of the proposal will be modified.

- TIM: A chunk of the coefficient vector (up to mh.chunk.size) is selected. The proposal distribution is constructed by locating the posterior mode (using the current value as a starting point). The proposal is a Gaussian (or multivariate T) centered on the posterior mode, with precision equal to the negative Hessian evaluated at the mode. This is an expensive, but effective step. If the posterior mode finding fails (for numerical reasons) then a RWM proposal will be attempted instead.

Value

Returns an object of class mlm.spike, which inherits from logit.spike and lm.spike. The returned object is a list with the following elements

- beta: A niter by ncol(x) matrix of regression coefficients, many of which may be zero. Each row corresponds to an MCMC iteration.
- prior: The prior used to fit the model. If a prior was supplied as an argument it will be returned. Otherwise this will be the automatically generated prior based on the other function arguments.
- MH.accounting: A summary of the amount of time spent in each type of MCMC move, and the acceptance rate for each move type.

Author(s)

Steven L. Scott

References


See Also

lm.spike, SpikeSlabPrior, plot.lm.spike, summary.lm.spike, predict.lm.spike.
Examples

```r
rmulti <- function (prob) {
  ## Sample from heterogeneous multinomial distributions.
  if (is.vector(prob)) {
    S <- length(prob)
    return(sample(1:S, size = 1, prob = prob))
  }
  nc <- apply(prob, 1, sum)
  n <- nrow(prob)
  S <- ncol(prob)
  u <- runif(n, 0, nc)
  alive <- rep(TRUE, n)
  z <- numeric(n)
  p <- rep(0, n)
  for (s in 1:S) {
    p <- p + prob[, s]
    indx <- alive & (u < p)
    alive[indx] <- FALSE
    z[indx] <- s
    if (!any(alive))
      break
  }
  return(z)
}

## Define sizes for the problem
subject.predictor.dimension <- 3
choice.predictor.dimension <- 4
nchoices <- 5
nobs <- 1000

## The response can be "a", "b", "c", ...
choice.levels <- letters[1:nchoices]

## Create "subject level characteristics".
subject.x <- matrix(rnorm(nobs * (subject.predictor.dimension - 1)),
                     nrow = nobs)
subject.beta <- cbind(0, matrix(rnorm(subject.predictor.dimension * (nchoices - 1)),
                               ncol = nchoices - 1))
colnames(subject.x) <- state.name[1:ncol(subject.x)]

## Create "choice level characteristics".
choice.x <- matrix(rnorm(nchoices * choice.predictor.dimension * nobs),
                    nrow = nobs)
choice.characteristics <- c("foo", "bar", "baz", "qux")
choice.names <- as.character(outer(choice.characteristics, choice.levels, FUN = paste, sep = ":"))
colnames(choice.x) <- choice.names
choice.beta <- rnorm(choice.predictor.dimension)

## Combine an intercept term, subject data, and choice data.
```

mlm.spike.slab.prior

Create a spike and slab prior for use with mlm.spike.

Description

Creates a spike and slab prior for use with mlm.spike.

Usage

MultinomialLogitSpikeSlabPrior(
  response,
  subject.x,
  expected.subject.model.size = 1,
  choice.x = NULL,
  expected.choice.model.size = 1,
  max.flips = -1,
  nchoices = length(levels(response)),
  subject.dim = ifelse(is.null(subject.x), 0, ncol(subject.x)),
  choice.dim = ifelse(is.null(choice.x), 0, ncol(choice.x)))
Arguments

response The response variable in the multinomial logistic regression. The response variable is optional if nchoices is supplied. If ‘response’ is provided then the prior means for the subject level intercepts will be chosen to match the empirical values of the response.

subject.x The design matrix for subject-level predictors. This can be NULL or of length 0 if no subject-level predictors are present.

expected.subject.model.size The expected number of non-zero coefficients – per choice level – in the subject specific portion of the model. All coefficients can be forced into the model by setting this to a negative number, or by setting it to be larger than the dimension of the subject-level predictors.

choice.x The design matrix for choice-level predictors. Each row of this matrix represents the characteristics of a choice in a choice occasion, so it takes 'nchoices' rows to encode one observation. This can be NULL or of length 0 if no choice-level predictors are present.

expected.choice.model.size The expected number of non-zero coefficients in the choice-specific portion of the model. All choice coefficients can be forced into the model by setting this to a negative number, or by setting it to be larger than the dimension of the choice-level predictors (for a single response level).

max.flips The maximum number of variable inclusion indicators the sampler will attempt to sample each iteration. If max.flips <= 0 then all indicators will be sampled.

nchoices The number of potential response levels.

subject.dim The number of potential predictors in the subject-specific portion of the model.

choice.dim The number of potential predictors in the choice-specific portion of the model.

Value

An object of class IndependentSpikeSlabPrior, with elements arranged as expected by mlm.spike.

Author(s)

Steven L. Scott

References

model.matrix  

GetPredictorMatrix

Description

Extract the matrix of predictors.

Usage

GetPredictorMatrix(object, newdata, na.action = na.omit, ...)

Arguments

object  
An object of class glm.spike. The object must be a list with the following elements

• beta: a matrix of MCMC draws, with rows representing draws, and columns representing coefficients.
• xlevels: the levels of any contrasts present in the original training data.
• contrasts: the "contrasts" attribute of the original design matrix used to train the model.
• terms: the terms of the formula used to fit the original model.

newdata  
A data frame, matrix, or vector containing the predictors needed to make a prediction. If newdata is a matrix it must have the same number of columns as length(object$beta), unless it is off by one and the model contains an intercept, in which case an intercept term will be added. If length(object$beta) == 1 (or 2, with one element containing an intercept) then newdata can be a numeric vector.

na.action  
A function specifying what to do with NA's.

...  
Extra arguments passed to model.matrix, in the event that newdata is a data frame.

Value

A matrix of predictor variables suitable for multiplication by object$beta.

Author(s)

Steven L. Scott

See Also

lm.spike SpikeSlabPrior plot.lm.spike predict.lm.spike
Construct Design Matrices

Description

Creates a matrix of predictors appropriate for glm.spike models.

Usage

```r
## S3 method for class 'glm.spike'
model.matrix(object, data = NULL, ...)
```

Arguments

- `object` An object of class `glm.spike`.
- `data` Either a data frame to use when building the model matrix, or `NULL`. If `NULL` then the training data from `object` will be used.
- `...` Extra arguments passed to `model.matrix.default`.

Details

glm.spike objects do not store the predictors used to fit the model. If the training data is modified between when `object` is fit and when this function is called, the modifications will be reflected in the returned value.

Value

The matrix of predictors used at training time, so long as the original data used to fit the model is available in the frame where this function is called.

Author(s)

Steven L. Scott

See Also

`lm.spike`
nested.regression

Description
Fits a Bayesian hierarchical regression model to data nested within groups. The model is

\[ y_{ig} \sim N(x_i \beta_g, \sigma^2) / \sigma^2 \sim \text{Gamma}(df/2, ss/2) \beta_g \sim N(b, V) \]

Optional hyperprior distributions can be supplied to the prior parameters.

\[ b \sim N(prior\textunderscore mean, prior\textunderscore variance) V \sim \text{InverseWishart}(df, variance\textunderscore guess). \]

Either hyperprior can be omitted, in which case the corresponding prior parameter is assumed fixed at the user-supplied value.

Usage
```
NestedRegression(response, predictors, group.id, 
    residual.precison.prior = NULL, 
    coefficient.prior = NULL, 
    coefficient.mean.hyperprior = NULL, 
    coefficient.variance.hyperprior = NULL, 
    suf = NULL, 
    niter, 
    ping = niter / 10, 
    sampling.method = c("ASIS", "DA"), 
    seed = NULL)
```

Arguments
- `response`: A numeric vector. The response variable to be modeled.
- `predictors`: A numeric matrix of predictor variables, including an intercept term if one is desired. The number of rows must match length(response).
- `group.id`: A factor (or object that can be converted using `as.factor`) naming the group to which each entry in `response` belongs.
- `residual.precison.prior`: An object of type `SdPrior` describing the prior distribution of the residual standard deviation.
- `coefficient.prior`: An object of class `MvnPrior`, or `NULL`. If non-`NULL` this gives the initial values of the prior distribution of the regression coefficients in the nested regression model. This argument must be non-`NULL` if either `coefficient.mean.hyperprior` or `coefficient.variance.hyperprior` is `NULL`.


*nested.regression*

coefficient.mean.hyperprior

An object of class `MvnPrior`, specifying the hyperprior distribution for the mean of `coefficient.prior`. This argument can also be `NULL` or `FALSE`. If `NULL` then a default prior will be used when learning the mean of the prior distribution. If `FALSE` then the mean of the prior distribution will not be learned; the mean of the `coefficient.prior` distribution will be assumed instead.

coefficient.variance.hyperprior

An object of class `InverseWishartPrior`, specifying the hyperprior distribution for the variance of `coefficient.prior`. This argument can also be `NULL`, or `FALSE`. If `NULL` then a default prior will be used when learning the variance of the prior distribution. If `FALSE` then the variance of the prior distribution will not be learned; the variance of the `coefficient.prior` distribution will be assumed instead.

suf

A list, where each entry is of type `RegressionSuf`, giving the sufficient statistics for each group, or `NULL`. If `NULL` then `suf` will be computed from `response`, `predictors`, and `group.id`. If non-`NULL` then these arguments will not be accessed, in which case they can be left unspecified. In 'big data' problems this can be a significant computational savings.

niter

The desired number of MCMC iterations.

ping

The frequency with which to print status updates.

sampling.method

The MCMC sampling scheme that should be used. If either hyperprior is set to `FALSE` then the "DA" method will be used.

seed

The integer-valued seed (or `NULL`) to use for the C++ random number generator.

Details

Note: ASIS (Yu and Meng, 2011) has slightly better MCMC convergence, but is slightly slower than the classic DA (data augmentation) method, which alternates between sampling group-level regression coefficients and prior parameters. Both methods are pretty fast.

Value

A list containing MCMC draws from the posterior distribution of model parameters. Each of the following is a vector, matrix, or array, with first index corresponding to MCMC draws, and later indices to distinct parameters.

- coefficients: regression coefficients.
- residual.sd: the residual standard deviation from the regression model.
- prior.mean: The posterior distribution of the coefficient means across groups.
- prior.variance: The posterior distribution of the variance matrix describing the distribution of regression coefficients across groups.
- priors: A list of the prior distributions used to fit the model.

Author(s)

Steven L. Scott
Examples

```r
SimulateNestedRegressionData <- function() {
  beta.hyperprior.mean <- c(8, 6, 7, 5)
  xdim <- length(beta.hyperprior.mean)
  beta.hyperprior.variance <-
    rWishart(2 * xdim, diag(rep(1, xdim)), inverse = TRUE)

  number.of.groups <- 27
  nobs.per.group = 23
  beta <- rmvn(number.of.groups,
               beta.hyperprior.mean,
               beta.hyperprior.variance)

  residual.sd <- 2.4
  X <- cbind(1, matrix(rnorm(number.of.groups * (xdim - 1) * nobs.per.group),
                       ncol = xdim - 1))
  group.id <- rep(1:number.of.groups, len = nrow(X))
  y.hat <- numeric(nrow(X))
  for (i in 1:nrow(X)) {
    y.hat[i] = sum(X[i, ] * beta[group.id[i], ])
  }
  y <- rnorm(length(y.hat), y.hat, residual.sd)
  suf <- BoomSpikeSlab:::.RegressionSufList(X, y, group.id)

  return(list(beta.hyperprior.mean = beta.hyperprior.mean,
               beta.hyperprior.variance = beta.hyperprior.variance,
               beta = beta,
               residual.sd = residual.sd,
               X = X,
               y = y,
               group.id = group.id,
               suf = suf))
}

d <- SimulateNestedRegressionData()
model <- NestedRegression(suf = d$suf, niter = 500)
```

nnet

Bayesian Feed Forward Neural Networks

Description

Fit a feed forward neural network using MCMC.

Usage

```r
BayesNnet(formula, hidden.layers,)
```
nnet

niter,
data,
subset,
prior = NULL,
expected.model.size = Inf,
drop.unused.levels = TRUE,
contrasts = NULL,
ping = niter / 10,
seed = NULL)

HiddenLayer(number.of.nodes, prior = NULL, expected.model.size = Inf)

Arguments

formula A formula describing the model to be fit. The formula should be additive. The network will figure out any interactions or nonlinearities.

hidden.layers A list of objects created by HiddenLayer defining the network structure. The input layer is determined by the formula argument. The terminal layer is a linear regression on the outputs of the final hidden layer.

niter The number of MCMC iterations to run. Be sure to include enough so you can throw away a burn-in set.

data An optional data frame, list or environment (or object coercible by 'as.data.frame' to a data frame) containing the variables in the model. If not found in 'data', the variables are taken from 'environment(formula)', typically the environment from which BayesNnet is called.

subset an optional vector specifying a subset of observations to be used in the fitting process.

prior When passed to BayesNnet this is the prior distribution for the terminal layer, which must be an object of class SpikeSlabPrior, SpikeSlabPriorDirect, or NULL. If NULL then a default prior will be used.

When passed to HiddenLayer this is the prior distribution for the coefficients to that layer. The prior is specified for a single output node, and the same prior is used for all nodes. You can think of each hidden layer output node as a logistic regression model where the predictors are the outputs of the previous layer. This must be an object of class MvnPrior, SpikeSlabGlmPrior, or SpikeSlabGlmPriorDirect.

expected.model.size When prior is not specified a default spike-and-slab prior will be used. The expected.model.size argument to BayesNnet is passed to SpikeSlabPriorDirect. In HiddenLayer the argument is passed to SpikeSlabGlmPriorDirect.

The parameter is used to set the prior inclusion probabilities for the coefficients. If p coefficients are available then the prior inclusion probabilities are each set to expected.model.size / p. If this ratio exceeds 1 then model selection is turned off and all coefficients are included.

drop.unused.levels Logical indicating whether unobserved factor levels should be dropped when forming the model matrix.
contrasts  An optional list. See the contrasts.arg argument of model.matrix.default.

ping    The frequency with which to print status update messages to the screen. For example, if ping == 10 then an update will be printed every 10 MCMC iterations.

seed    An integer to use as the random seed for the underlying C++ code. If NULL then the seed will be set using the clock.

number.of.nodes  The number of nodes in this hidden layer. This must be a positive scalar integer.

Details

The model is a feedforward neural network regression. The model is fit using an MCMC algorithm based on data augmentation. Each hidden node is randomly assigned a 0/1 value from its full conditional distribution. Then conditional on the imputed data an MCMC draw is done on each latent logistic regression and on the regression model defining the terminal node.

Value

The returned object is a list with class BayesNnet. It contains the following objects

-\* residual.sd\* The standard deviation of the residuals from the model.
-\* hidden.layer.coefficients\* A list, with one element per hidden layer, giving the posterior draws of the hidden layer coefficients for that layer. Each list element is a 3-way array with dimensions corresponding to
   1. MCMC iteration
   2. Input node. For the first hidden layer each 'input node' is a predictor variable.
   3. Output node.

You can think of hidden.layer.coefficients[[ii]][, , j] as the posterior distribution of the logistic regression model defining node 'j' in hidden layer 'i'.

-\* terminal.layer.coefficients\* A matrix containing the MCMC draws of the model coefficients for the terminal layer.
- Other list elements needed to implement various methods (predict, plot, etc.).

Author(s)

Steven L. Scott

References

??

See Also

Examples

```r
if (require(mlbench)) {
  data(BostonHousing)
  hidden.layers <- list(
    HiddenLayer(10, expected.model.size = Inf))

  ## In real life you'd want more 50 MCMC draws.
  model <- BayesNnet(medv ~ .,
    hidden.layers = hidden.layers,
    niter = 50,
    data = BostonHousing)

  par(mfrow = c(1, 2))
  plot(model) # plots predicted vs actual.
  plot(model, "residual") # plots
  par(mfrow = c(1,1))
  plot(model, "structure")
  ## Examine all partial dependence plots.
  plot(model, "partial", pch = ".")

  ## Examine a single partial dependence plot.
  par(mfrow = c(1,1))
  plot(model, "lstat", pch = ".")

  ## Check out the mixing performance.
  PlotManyTs(model$terminal.layer.coefficients)
  PlotMacf(model$terminal.layer.coefficients)

  ## Get the posterior distribution of the function values for the
  ## training data.
  pred <- predict(model)

  ## Get predictions for data at new points (though in this example I'm
  ## reusing old points.
  pred2 <- predict(model, newdata = BostonHousing[1:12, ])
} else {
  cat("The Boston housing data from 'mlbench' is needed for this example.")
}
```

partial.dependence.plot

Plot a Bayesian Neural Network

Description

Plot the relationship between Y and a single X variable, averaging over the values of the other X's.
Usage

PartialDependencePlot(model,
    which.variable,
    burn = SuggestBurn(model),
    data.fraction = .2,
    gridsize = 50,
    mean.only = FALSE,
    show.points = TRUE,
    xlab = NULL,
    ylab = NULL,
    ylim = NULL,
    report.time = FALSE,
    ...
)

Arguments

model An object of class BayesNnet.

which.variable Either an integer denoting the position of the X variable in the data frame used to fit the model, or a character string naming that variable.

burn The number of MCMC iterations to discard as burn-in.

data.fraction The fraction of observations in the predictor matrix to use when constructing the partial dependence plot. A random sub-sample of this fraction will be taken (without replacement) for the purposes of marginalizing over the remaining predictors.

gridsize The number of grid points to use on the X axis.

mean.only Logical. If TRUE then only the mean is plotted at each point. If FALSE then the posterior of the function value is plotted.

show.points If TRUE then the scatterplot of x vs y is added to the graph. Otherwise the points are left off. Note that the estimated function might not match the pattern in the scatterplot, because the points in the scatterplot are not adjusted for the values of the other X variables.

xlab Label for the X axis. NULL produces a default label. Use "" for no label.

ylab Label for the Y axis. NULL produces a default label. Use "" for no label.

ylim Limits on the vertical axis. If NULL then the plot will default to its natural vertical limits.

report.time Print the time required to produce the plot.

... Extra arguments are passed either to 'plot' (if mean.only is TRUE)’ or 'PlotDynamicDistribution’ (otherwise).

Details

A partial dependence plot shows the relationship between Y and a single X variable, averaging over the values of the other X’s in a possibly nonlinear regression model. Partial dependence plots are a generalization of the "added variable plot" idea from linear regression models.
A partial dependence plot is more expensive to produce than most other plots, because a set of predictions must be generated at each point on the X axis. This is done by taking a random subset of the training data, and evaluating the posterior predictive distribution with each observation’s target X value set to each value of X on the grid.

Author(s)

Steven L. Scott

See Also

plot.BayesNnet

Examples

# Please see the code in ?BayesNnet

plot.BayesNnet plot a Bayesian Neural Network

Description

The default plot is a barplot of the marginal inclusion probabilities for each variable, as obtained by PlotMarginalInclusionProbabilities. Other interesting plots can be obtained by supplying a string as the second argument.

Usage

## S3 method for class 'BayesNnet'
plot(x,
y = c("predicted", "residual", "structure", "partial", "help"),
...

PlotBayesNnetPredictions(model, burn = SuggestBurn(model), ...)

PlotBayesNnetResiduals(model, burn = SuggestBurn(model), ...)

PlotNetworkStructure(model, ...)

Arguments

model An object of class BayesNnet.
x An object of class BayesNnet. The name x is required to conform with the plot generic function signature.
The type of plot desired, or the name of the variable to plot against. The name y is required to conform with the plot generic function signature.

If y matches (or partially matches) one of the names in the function signature, then the corresponding plot function handles the plot request.

- "predicted" (the default) plot actual vs predicted values using `PlotBayesNnetPredictions`.
- "residual" plot residuals vs predicted values using `PlotBayesNnetResiduals`.
- "structure" plot network structure using `PlotNetworkStructure`.
- "partial" Draw the partial dependence plot for each predictor variable in the training data. This is an expensive plot. It might take a while to draw for large data sets or complex models.
- "help" show this help page in a browser

If y fails to match any of the above, but it (partially) the name of one of the variables in the training data, then a partial dependence plot vs that variable is produced.

The number of MCMC iterations to discard as burn-in.

Additional arguments passed to the specific functions that do the plotting. For residual and predicted plots that is the plot function. For network structure it is `plot.igraph`. For partial dependence plots it is `PartialDependencePlot`.

Details

Residual and predicted plots should be self explanatory. The network structure plot is fairly standard for neural network models. The width of a line linking two nodes is determined by the absolute value of the corresponding coefficient.

Author(s)

Steven L. Scott

See Also

BayesNnet PartialDependencePlot

Examples

```r
## See the examples in ?BayesNnet
```

Description

Produces boxplots showing the marginal distribution of the coefficients.
Usage

PlotLmSpikeCoefficients(
  beta,
  burn = 0,
  inclusion.threshold = 0,
  scale.factors = NULL,
  number.of.variables = NULL,
  ...
)

Arguments

beta  A matrix of model coefficients. Each row represents an MCMC draw. Each column represents a coefficient for a variable.
burn  The number of MCMC iterations in the object to be discarded as burn-in.
inclusion.threshold  Only plot coefficients with posterior inclusion probabilities exceeding this value.
scale.factors  If non-null then a vector of scale factors with which to scale the columns of beta. A NULL value is ignored.
number.of.variables  If non-NULL this specifies the maximum number of coefficients to plot. A NULL value is ignored.
...  Additional arguments to be passed to boxplot.

Value

Returns the value from the final call to boxplot.

Author(s)

Steven L. Scott

See Also

lm.spike SpikeSlabPrior summary.lm.spike predict.lm.spike

Examples

simulate.lm.spike <- function(n = 100, p = 10, ngood = 3, niter=1000, sigma = 1){
  x <- cbind(matrix(rnorm(n * (p-1)), nrow=n), rep(0, n - ngood))
  beta <- c(rnorm(ngood), rep(0, p - ngood))
  y <- rnorm(n, beta[1] + x %*% beta[-1], sigma)
  draws <- lm.spike(y ~ x, niter=niter)
  return(invisible(draws))
}
model <- simulate.lm.spike(n = 1000, p = 50, sigma = .3)
plot(model, "coef", inclusion.threshold = .01)
plot.lm.spike  
Plot the results of a spike and slab regression.

Description

The default plot is a barplot of the marginal inclusion probabilities for each variable, as obtained by 
PlotMarginalInclusionProbabilities. Other interesting plots can be obtained by supplying a string as the second argument.

Usage

## S3 method for class 'lm.spike'
plot(x,  
y = c("inclusion", "coefficients", "scaled.coefficients",  
"residuals", "size", "help"),  
burn = SuggestBurnLogLikelihood(x$log.likelihood),  
...)

Arguments

x  An object of class lm.spike.
y  The type of plot desired.
burn  The number of MCMC iterations to discard as burn-in.
...  Additional arguments passed to the specific functions that do the plotting.

Details

The actual plotting will be handled by PlotMarginalInclusionProbabilities, PlotLmSpikeCoefficients, 
PlotLmSpikeResiduals, or PlotModelSize. See the appropriate function for more options.

Author(s)

Steven L. Scott

See Also

PlotMarginalInclusionProbabilities,PlotLmSpikeCoefficients,PlotLmSpikeResiduals,PlotModelSize

Examples

simulate.lm.spike <- function(n = 100, p = 10, ngood = 3, niter=1000, sigma = 8){
x <- cbind(matrix(rnorm(n * (p-1)), nrow=n))
beta <- c(rnorm(ngood), rep(0, p - ngood))
y <- rnorm(n, beta[1] + x %*% beta[-1], sigma)
draws <- lm.spike(y ~ x, niter=niter)
return(invisible(draws))
model <- simulate.lm.spike(n = 1000, p = 50, sigma = .3)
plot(model, inclusion.threshold = .01)
plot(model, "size")

plot.lm.spike.residuals

*Residual plot for lm.spike*

**Description**

Plot residuals vs. fitted values in an lm.spike model.

**Usage**

```r
PlotLmSpikeResiduals(
  object,
  burn = SuggestBurnLogLikelihood(object$log.likelihood),
  ...
)
```

**Arguments**

- **object**: A model object inheriting from `lm.spike`.
- **burn**: The number of MCMC iterations to be discarded as burn-in before computing posterior means.
- **...**: Additional arguments passed to `plot`.

**Details**

This plot is normally called via the plot function for lm.spike objects. See the help entry for `lm.spike` for example usage.

**Author(s)**

Steven L. Scott

**See Also**

`lm.spike` `plot.lm.spike`
Description

Plot a \texttt{logit.spike} object. The default plot is a barplot of the marginal inclusion probabilities for each variable, as obtained by \texttt{PlotMarginalInclusionProbabilities}. See below for other types of plots.

Usage

```r
## S3 method for class 'logit.spike'
plot(x,
y = c("inclusion", "coefficients", "scaled.coefficients", "fit",
"residuals", "size", "help"),
burn = SuggestBurnLogLikelihood(x$log.likelihood),
...)
```

```r
## S3 method for class 'probit.spike'
plot(x,
y = c("inclusion", "coefficients", "scaled.coefficients", "fit",
"residuals", "size", "help"),
burn = SuggestBurnLogLikelihood(x$log.likelihood),
...)
```

Arguments

- \texttt{x} An object of class \texttt{logit.spike}.
- \texttt{y} The type of plot desired.
- \texttt{burn} The number of MCMC iterations to discard as burn-in.
- \texttt{...} Additional arguments passed to the specific functions that do the plotting.

Details

The default plot is a barplot showing the marginal inclusion probabilities of the coefficients, constructed using \texttt{PlotMarginalInclusionProbabilities}.

The plot of the fit summary is handled by \texttt{PlotLogitSpikeFitSummary}.

The plot of the residuals is handled by \texttt{PlotLogitSpikeResiduals}.

The plot of model size is handled by \texttt{PlotModelSize}.

Author(s)

Steven L. Scott
plot.logit.spike.fit.summary

See Also

PlotMarginalInclusionProbabilities PlotModelSize PlotLogitSpikeFitSummary PlotLogitSpikeResiduals

Examples

## See the examples in ?logit.spike

---

plot.logit.spike.fit.summary

*Plot Logit or Probit Fit Summary*

Description

Two plots can be accessed by this function. The first is a time series plot of the "deviance R-square" statistic, by MCMC iteration. The second is a Hosmer-Lemeshow plot in which the data is divided into 10 groups based on predicted probabilities, and the empirical success probabilities for that group are plotted against the expected probabilities from the model.

Usage

```r
PlotLogitSpikeFitSummary(
  model,
  burn = 0,
  which.summary = c("both", "r2", "bucket"),
  scale = c("logit", "probability"),
  cutpoint.basis = c("sample.size", "equal.range"),
  number.of.buckets = 10,
  ...
)
```

```r
PlotProbitSpikeFitSummary(
  model,
  burn = 0,
  which.summary = c("both", "r2", "bucket"),
  scale = c("probit", "probability"),
  cutpoint.basis = c("sample.size", "equal.range"),
  number.of.buckets = 10,
  ...
)
```

Arguments

- `model` A model object inheriting from `logit.spike` or `probit.spike`.
- `burn` The number of MCMC iterations in the object to be discarded as burn-in. Note that this only affects the deviance R-square plot. The fit summaries in the Hosmer-Lemeshow plot are constructed by `logit.spike` or `probit.spike` in order to keep permanent object sizes small.
- `which.summary` Which plot is desired?
scale

The scale to use for the predicted probabilities in the Hosmer-Lemeshow plot.

cutpoint.basis

How should cutpoints be determined for the Hosmer-Lemeshow plot? If "sample.size" then each bucket will have equal sample size. If "equal.range" then each bucket will occupy the same size on the chosen (logit/probit or probability) scale.

number.of.buckets

The number of buckets to use in the Hosmer-Lemeshow plot.

Additional arguments to be passed to barplot.

Author(s)

Steven L. Scott

See Also

lm.spike SpikeSlabPrior summary.lm.spike predict.lm.spike

Examples

simulate.logit.spike <- function(n = 100, p = 10, ngood = 3,
                                 niter=1000)
  x <- cbind(1, matrix(rnorm(n * (p-1)), nrow=n))
  beta <- c(rnorm(ngood), rep(0, p - ngood))
  prob <- plogis(x %*% beta)
  y <- runif(n) < prob
  x <- x[,1]
  draws <- logit.spike(y ~ x, niter=niter)
  plot.ts(draws$beta)
  return(invisible(draws))
}
model <- simulate.logit.spike()
plot(model, "fit")
plot(model, "fit", scale = "probability", number.of.buckets = 15)

plot.logit.spike.residuals

Residual plot for logit.spike objects.

Description

Plots the "deviance residuals" from a logit.spike model.

Usage

PlotLogitSpikeResiduals(model, ...)
PlotProbitSpikeResiduals(model, ...)
Arguments

model A model object inheriting from logit.spike or probit.spike.

... Additional arguments to be passed to plot.

Details

The "deviance residuals" are defined as the signed square root each observation’s contribution to log likelihood. The sign of the residual is positive if half or more of the trials associated with an observation are successes. The sign is negative otherwise.

The "contribution to log likelihood" is taken to be the posterior mean of an observations log likelihood contribution, averaged over the life of the MCMC chain.

The deviance residual is plotted against the fitted value, again averaged over the life of the MCMC chain.

The plot also shows the .95 and .99 bounds from the square root of a chi-square(1) random variable. As a rough approximation, about 5% and 1% of the data should lie outside these bounds.

Author(s)

Steven L. Scott

See Also

logit.spike plot.logit.spike

Examples

simulate.logit.spike <- function(n = 100, p = 10, ngood = 3,
                                 niter=1000)
  x <- cbind(1, matrix(rnorm(n * (p-1)), nrow=n))
  beta <- c(rnorm(ngood), rep(0, p - ngood))
  prob <- plogis(x @% beta)
  y <- runif(n) < prob
  x <- x[,,-1]
  draws <- logit.spike(y ~ x, niter=niter)
  plot.ts(draws$beta)
  return(invisible(draws))
}
model <- simulate.logit.spike()
plot(model, "fit")
plot(model, "fit", scale = "probability", number.of.buckets = 15)
**plot.marginal.inclusion.probabilities**

*Plot marginal inclusion probabilities.*

**Description**

Produces a barplot of the marginal inclusion probabilities for a set of model coefficients sampled under a spike and slab prior. The coefficients are sorted by the marginal inclusion probability, and shaded by the conditional probability that a coefficient is positive, given that it is nonzero.

**Usage**

```r
PlotMarginalInclusionProbabilities(
    beta,
    burn = 0,
    inclusion.threshold = 0,
    unit.scale = TRUE,
    number.of.variables = NULL,
    ...)
```

**Arguments**

- **beta**
  A matrix of model coefficients. Each row represents an MCMC draw. Each column represents a coefficient for a variable.

- **burn**
  The number of MCMC iterations in the object to be discarded as burn-in.

- **inclusion.threshold**
  Only plot coefficients with posterior inclusion probabilities exceeding this value.

- **unit.scale**
  A logical value indicating whether the scale of the plot should be from 0 to 1. Otherwise the scale is determined by the maximum inclusion probability.

- **number.of.variables**
  If non-NULL this specifies the number of coefficients to plot, taking precedence over inclusion.threshold.

- **...**
  Additional arguments to be passed to `barplot`.

**Value**

Invisibly returns a list with the following elements.

- **barplot**
  The midpoints of each bar, which is useful for adding to the plot.

- **inclusion.prob**
  The marginal inclusion probabilities of each variable, ordered smallest to largest (the same order as the plot).

- **positive.prob**
  The probability that each variable has a positive coefficient, in the same order as inclusion.prob.

- **permutation**
  The permutation of beta that puts the coefficients in the same order as positive.prob and inclusion.prob. That is: `beta[,permutation]` will have the most significant coefficients in the right hand columns.
Author(s)
Steven L. Scott

See Also
lm.spike SpikeSlabPrior summary.lm.spike predict.lm.spike

Examples

```r
simulate.lm.spike <- function(n = 100, p = 10, ngood = 3, niter=1000, sigma = 8){
  x <- cbind(matrix(rnorm(n * (p-1)), nrow=n, ncol=p-1), nrow=n)
  beta <- c(rnorm(ngood), rep(0, p - ngood))
  y <- rnorm(n, beta[1] + x %*% beta[-1], sigma)
  draws <- lm.spike(y ~ x, niter=niter)
  return(invisible(draws))
}
model <- simulate.lm.spike(n = 1000, p = 50, sigma = .3)
plot(model, inclusion.threshold = .01)
```

Description

Plot a `poisson.spike` object. The default plot is a barplot of the marginal inclusion probabilities for each variable, as obtained by `PlotMarginalInclusionProbabilities`. See below for other types of plots.

Usage

```r
## S3 method for class 'poisson.spike'
plot(x, y = c("inclusion", "coefficients", "scaled.coefficients", "size", "help"),
    burn = SuggestBurnLogLikelihood(x$log.likelihood),
    ...)```

Arguments

- `x`: An object of class `poisson.spike`.
- `y`: The type of plot desired.
- `burn`: The number of MCMC iterations to discard as burn-in.
- `...`: Additional arguments passed to the specific functions that do the plotting.

Details

The default plot is a barplot showing the marginal inclusion probabilities of the coefficients, constructed using `PlotMarginalInclusionProbabilities`. The plot of model size is handled by `PlotModelSize`. 
plot.qreg.spike

Author(s)

Steven L. Scott

See Also

PlotMarginalInclusionProbabilities, PlotModelSize

Examples

## See the examples in ?poisson.spike

```
plot.qreg.spike
```

Plot the results of a spike and slab regression.

Description

The default plot is a barplot of the marginal inclusion probabilities for each variable, as obtained by `PlotMarginalInclusionProbabilities`. Other interesting plots can be obtained by supplying a string as the second argument.

Usage

```
## S3 method for class 'qreg.spike'
plot(x,
y = c("inclusion", "coefficients", "scaled.coefficients", "size", "help"),
burn = SuggestBurnLogLikelihood(x$log.likelihood),
...)
```

Arguments

- `x`: An object of class `qreg.spike`.
- `y`: The type of plot desired.
- `burn`: The number of MCMC iterations to discard as burn-in.
- `...`: Additional arguments passed to the specific functions that do the plotting.

Details

The actual plotting will be handled by `PlotMarginalInclusionProbabilities`, `PlotLmSpikeCoefficients`, or `PlotModelSize`. See the appropriate function for more options.

Author(s)

Steven L. Scott
See Also

PlotMarginalInclusionProbabilities PlotLmSpikeCoefficients PlotModelSize qreg.spike SpikeSlabPrior predict.qreg.spike

Examples

```r
n <- 50
x <- rnorm(n)
y <- rnorm(n, 4 * x)
model <- qreg.spike(y ~ x,
                   quantile = .8,
                   niter = 1000,
                   expected.model.size = 100)
plot(model)
plot(model, "coef")
plot(model, "coefficients")
plot(model, "scaled.coefficients")
plot(model, "scal")
plot(model, "size")
plot(model, "help")
```

---

**PlotModelSize**

Plot a distribution of model size

**Description**

Produces a histogram of number of nonzero coefficients in a spike-and-slab regression.

**Usage**

```r
PlotModelSize(beta, burn = 0, xlab = "Number of nonzero coefficients", ...)
```

**Arguments**

- `beta` A matrix of model coefficients. Each row represents an MCMC draw. Each column represents a coefficient for a variable.
- `burn` The number of MCMC iterations to be discarded as burn-in.
- `xlab` Label for the horizontal axis.
- `...` Additional arguments to be passed to `hist`

**Value**

Invisibly returns the vector of MCMC draws of model sizes.

**Author(s)**

Steven L. Scott
poisson.spike

Spike and slab Poisson regression

Description

MCMC algorithm for Poisson regression models with a 'spike-and-slab' prior that places some amount of posterior probability at zero for a subset of the coefficients.

Usage

poisson.spike(formula, exposure = 1, niter, data, subset, prior = NULL, na.action = options("na.action"), contrasts = NULL, drop.unused.levels = TRUE, initial.value = NULL, ping = niter / 10, nthreads = 4, seed = NULL, ...)

Examples

simulate.lm.spike <- function(n = 100, p = 10, ngood = 3, niter=1000, sigma = 8){
  x <- cbind(matrix(rnorm(n * (p-1)), nrow=n))
  beta <- c(rnorm(ngood), rep(0, p - ngood))
  y <- rnorm(n, beta[1] + x %*% beta[-1], sigma)
  draws <- lm.spike(y ~ x, niter=niter)
  return(invisible(draws))
}
model <- simulate.lm.spike(n = 1000, p = 50, sigma = .3)

# To get the plot of model size directly.
PlotModelSize(model$beta, burn = 10)

# Another way to get the same plot.
plot(model, "size", burn = 10)
Arguments

- **formula**: A model formula, as would be passed to `glm`, specifying the maximal model (i.e. the model with all predictors included).
- **exposure**: A vector of exposure durations matching the length of the response vector. If exposure is of length 1 it will be recycled.
- **niter**: The number of MCMC iterations to run.
- **data**: An optional data frame, list or environment (or object coercible by `as.data.frame` to a data frame) containing the variables in the model. If not found in data, the variables are taken from `environment(formula)`, typically the environment from which `poisson.spike` is called.
- **subset**: An optional vector specifying a subset of observations to be used in the fitting process.
- **prior**: A list such as that returned by `SpikeSlabPrior`. If prior is supplied it will be used. Otherwise a prior distribution will be built using the remaining arguments. See `SpikeSlabPrior`.
- **na.action**: A function which indicates what should happen when the data contain NAs. The default is set by the `na.action` setting of `options`, and is `na.fail` if that is unset. The factory-fresh default is `na.omit`. Another possible value is `NULL`, no action. Value `na.exclude` can be useful.
- **contrasts**: An optional list. See the `contrasts.arg` of `model.matrix.default`.
- **drop.unused.levels**: A logical value indicating whether factor levels that are unobserved should be dropped from the model.
- **initial.value**: Initial value for the MCMC algorithm. Can either be a numeric vector, a `glm` object (from which the coefficients will be used), or a `poisson.spike` object. If a `poisson.spike` object is supplied, it is assumed to be from a previous MCMC run for which `niter` additional draws are desired. If a `glm` object is supplied then its coefficients will be used as the initial values for the simulation.
- **ping**: If positive, then print a status update to the console every `ping` MCMC iterations.
- **nthreads**: The number of CPU-threads to use for data augmentation.
- **seed**: Seed to use for the C++ random number generator. It should be `NULL` or an int. If `NULL` the seed value will be taken from the global `.Random.seed` object.
- **...**: Extra arguments to be passed to `SpikeSlabPrior`.

Details

The MCMC algorithm used here is based on the auxiliary mixture sampling algorithm published by Fruhwirth-Schnatter, Fruhwirth, Held, and Rue (2009).

Value

Returns an object of class `poisson.spike`. The returned object is a list with the following elements.
poisson.zellner.prior

beta A niter by ncol(x) matrix of regression coefficients, many of which may be zero. Each row corresponds to an MCMC iteration.
prior The prior used to fit the model. If a prior was supplied as an argument it will be returned. Otherwise this will be the automatically generated prior based on the other function arguments.

Author(s)
Steven L. Scott

References

See Also
lm.spike SpikeSlabPrior, plot.lm.spike, summary.lm.spike, predict.lm.spike.

Examples
simulate.poisson.spike <- function(n = 100, p = 10, ngood = 3, niter=1000){
x <- cbind(1, matrix(rnorm(n * (p-1)), nrow=n))
beta <- c(rnorm(ngood), rep(0, p - ngood))
lambda <- exp(x %*% beta)
y <- rpois(n, lambda)
x <- x[,,-1]
model <- poisson.spike(y ~ x, niter=niter)
return(invisible(model))
}
model <- simulate.poisson.spike()
plot(model)
summary(model)

Zellner Prior for Poisson Regression

Description
A Zellner-style spike and slab prior for Poisson regression models. See 'Details' for a definition.

Usage
PoissonZellnerPrior(
predictors, 
counts = NULL, 
exposure = NULL, 
prior.event.rate = NULL, 
)
expected.model.size = 1,
prior.information.weight = .01,
diagonal.shrinkage = .5,
onoptional.coefficient.estimate = NULL,
max.flips = -1,
prior.inclusion.probabilities = NULL)

Arguments

predictors The design matrix for the regression problem. No missing data is allowed.
counts The vector of responses. This is only used to obtain the empirical overall event rate, so it can be left NULL if prior.event.rate is specified.
exposure A vector of the same length as counts, giving the "exposure time" for each observation. This can also be NULL, signifying that exposure = 1.0 for each observation.
prior.event.rate An a priori guess at the overall event rate. Used in two places: to set the prior mean of the intercept (if optional.coefficient.estimate is NULL) and to weight the information matrix in the "slab" portion of the prior.
expected.model.size A positive number less than ncol(x), representing a guess at the number of significant predictor variables. Used to obtain the 'spike' portion of the spike and slab prior.
prior.information.weight A positive scalar. Number of observations worth of weight that should be given to the prior estimate of beta.
diagonal.shrinkage The conditionally Gaussian prior for beta (the "slab") starts with a precision matrix equal to the information in a single observation. However, this matrix might not be full rank. The matrix can be made full rank by averaging with its diagonal. diagonal.shrinkage is the weight given to the diagonal in this average. Setting this to zero gives Zellner's g-prior.
onoptional.coefficient.estimate If desired, an estimate of the regression coefficients can be supplied. In most cases this will be a difficult parameter to specify. If omitted then a prior mean of zero will be used for all coordinates except the intercept, which will be set to mean(y).
max.flips The maximum number of variable inclusion indicators the sampler will attempt to sample each iteration. If negative then all indicators will be sampled.
prior.inclusion.probabilities A vector giving the prior probability of inclusion for each variable. If NULL then a default set of probabilities is obtained by setting each element equal to min(1,expected.model.size / ncol(x)).

Details

A Zellner-style spike and slab prior for Poisson regression. Denote the vector of coefficients by $\beta$, and the vector of inclusion indicators by $\gamma$. These are linked by the relationship $\beta_i \neq 0$ if $\gamma_i = 1$
and $\beta_i = 0$ if $\gamma_i = 0$. The prior is

$$
\beta | \gamma \sim N(b, V) \\
\gamma \sim B(\pi)
$$

where $\pi$ is the vector of prior.inclusion.probabilities, and $b$ is the optional.coefficient.estimate.

Conditional on $\gamma$, the prior information matrix is

$$
V^{-1} = \kappa((1 - \alpha)x^T wx/n + \alpha \text{diag}(x^T wx/n))
$$

The matrix $x^T wx$ is, for suitable choice of the weight vector $w$, the total Fisher information available in the data. Dividing by $n$ gives the average Fisher information in a single observation, multiplying by $\kappa$ then results in $\kappa$ units of "average" information. This matrix is averaged with its diagonal to ensure positive definiteness.

In the formula above, $\kappa$ is prior.information.weight, $\alpha$ is diagonal.shrinkage, and $w$ is a diagonal matrix with all elements set to prior.success.probability * (1 - prior.success.probability).

The vector $b$ and the matrix $V^{-1}$ are both implicitly subscripted by $\gamma$, meaning that elements, rows, or columns corresponding to gamma = 0 should be omitted.

Value

Returns an object of class PoissonZellnerPrior, which is a list with data elements encoding the selected prior values. It inherits from PoissonPrior and from SpikeSlabGlmPrior, which implies that it contains an element prior.success.probability.

This object is intended for use with poisson.spike.

Author(s)

Steven L. Scott

References

predict.lm.spike

Usage

## S3 method for class 'lm.spike'
predict(object, newdata = NULL, burn = 0,
    na.action = na.pass, mean.only = FALSE, ...)

## S3 method for class 'logit.spike'
predict(object, newdata, burn = 0,
    type = c("prob", "logit", "link", "response"),
    na.action = na.pass, ...)

## S3 method for class 'poisson.spike'
predict(object, newdata = NULL,
    exposure = NULL, burn = 0,
    type = c("mean", "log", "link", "response"),
    na.action = na.pass, ...)

## S3 method for class 'probit.spike'
predict(object, newdata, burn = 0,
    type = c("prob", "probit", "link", "response"),
    na.action = na.pass, ...)

## S3 method for class 'qreg.spike'
predict(object, newdata, burn = 0,
    na.action = na.pass, ...)

## S3 method for class 'BayesNnet'
predict(object, newdata = NULL, burn = 0,
    na.action = na.pass, mean.only = FALSE, seed = NULL, ...)

Arguments

object
    A model object of class lm.spike, logit.spike, etc.

newdata
    Either NULL, or else a data frame, matrix, or vector containing the predictors needed to make the prediction.
    If newdata is NULL then the predictors are taken from the training data used to create the model object. Note that object does not store its training data, so the data objects used to fit the model must be present for the training data to be recreated.
    If newdata is a data.frame it must contain variables with the same names as the data frame used to fit object. If it is a matrix, it must have the same number of columns as object$beta. An intercept term will be implicitly added if the number of columns is too small by one. If the dimension of object$beta is 1 or 2, then newdata can be a vector.

exposure
    A vector of positive real numbers the same size as newdata, or NULL. If both newdata and exposure are NULL then exposure is taken to be the exposure from the training data. If newdata is supplied and exposure is NULL then exposure is taken to be 1 for all observations.
The number of MCMC iterations in the object to be discarded as burn-in.

A function which indicates what should happen when the data contain NA's. The default is set by the na.action setting of options, and is na.fail if that is unset. The "factory-fresh" default is na.omit. Another possible value is NULL, no action. Value na.exclude can be useful.

The type of prediction desired.
For logit.spike, prob means the prediction is returned on the probability scale, while logit returns the scale of the linear predictor. Probits work similarly to logits.
For poisson.spike, mean means the prediction is returned on the scale of the data, while log means it is on the scale of the linear predictor.
Both cases also accept link and response for compatibility with predict.glm.
Logical. If TRUE then return the posterior mean of the predictive distribution. If FALSE then return the entire distribution.
Random seed for the C++ random number generator. This is only needed for models that require C++ to implement their predict method.
Unused, but present for compatibility with generic predict.

Returns a matrix of predictions, with each row corresponding to a row in newdata, and each column to an MCMC iteration.

Steven L. Scott

lm.spike SpikeSlabPrior summary.lm.spike plot.lm.spike

Examples

niter <- 1000
n <- 100
p <- 10
ngood <- 3
x <- cbind(1, matrix(rnorm(n * (p-1)), nrow=n))
beta <- rep(0, p)
good <- sample(1:p, ngood)
beta[good] <- rnorm(ngood)
sigma <- 1

y <- rnorm(n, x %*% beta, sigma)
model <- lm.spike(y ~ x - 1, niter=niter)
plot(model)
plot.ts(model$beta)
hist(model$sigma)  ## should be near true value
new.x <- cbind(1, matrix(rnorm(100 * (p-1)), ncol = (p-1)))
pred <- predict(model, newdata = new.x, burn = 100)

print.summary.lm.spike

Print method for spikeslab objects.

Description
Print a spikeslab object.

Usage
## S3 method for class 'summary.lm.spike'
print(x, ...)
## S3 method for class 'summary.logit.spike'
print(x, ...)

Arguments
x                  An object of class summary.lm.spike.
...                Additional arguments passed to print.default.

Value
This function is called for its side effect, which is to print the spikeslab object to the screen.

Author(s)
Steven L. Scott

See Also
lm.spike summary.lm.spike

Examples
n <- 100
p <- 10
ngood <- 3
niter <- 1000
sigma <- 2
x <- cbind(1, matrix(rnorm(n * (p-1)), nrow=n))
beta <- c(rnorm(ngood), rep(0, p - ngood))
y <- rnorm(n, x %*% beta, sigma)
x <- x[, -1]
model <- lm.spike(y ~ x, niter=niter)
summary(model)
Spike and slab probit regression

Description

MCMC algorithm for logistic regression models with a 'spike-and-slab' prior that places some amount of posterior probability at zero for a subset of the regression coefficients.

Usage

```
probit.spike(formula, niter, data, subset, prior = NULL, na.action = options("na.action"), contrasts = NULL, drop.unused.levels = TRUE, initial.value = NULL, ping = niter / 10, clt.threshold = 5, proposal.df = 3, sampler.weights = c(.5,.5), seed = NULL, ...)
```

Arguments

- **formula**: Formula for the maximal model (with all variables included). This is parsed the same way as a call to `glm`, but no family argument is needed. Like `glm`, a two-column input format (success-count, failure-count) can be used for the response. Otherwise, the response variable can be a logical or numeric vector. If a single-column response is numeric, then a positive value indicates a "success".
- **niter**: The number of MCMC iterations to run. Be sure to include enough so you can throw away a burn-in set.
- **data**: An optional data frame, list or environment (or object coercible by 'as.data.frame' to a data frame) containing the variables in the model. If not found in 'data', the variables are taken from 'environment(formula)', typically the environment from which `probit.spike` is called.
- **subset**: An optional vector specifying a subset of observations to be used in the fitting process.
- **prior**: An object inheriting from `LogitPrior` and `SpikeSlabPriorBase`. If prior is supplied it will be used. Otherwise a prior distribution will constructed by calling `LogitZellnerPrior` with the remaining arguments. Despite the name, LogitPrior objects are appropriate for Probit models.
A function which indicates what should happen when the data contain NAs. The default is set by the `na.action` setting of options, and is `na.fail` if that is unset. The factory-fresh default is `na.omit`. Another possible value is `NULL`, no action. Value `na.exclude` can be useful.

An optional list. See the `contrasts.arg` of `model.matrix.default`.

A logical value indicating whether factor levels that are unobserved should be dropped from the model.

Initial value for the MCMC algorithm. Can either be a numeric vector, a `glm` object (from which the coefficients will be used), or a `probit.spike` object. If a `probit.spike` object is supplied, it is assumed to be from a previous MCMC run for which `niter` additional draws are desired. If a `glm` object is supplied then its coefficients will be used as the initial values for the simulation.

If positive, then print a status update to the console every `ping` MCMC iterations.

When the model is presented with binomial data (i.e. when the response is a two-column matrix) the data augmentation algorithm can be made more efficient by updating a single, asymptotically normal scalar quantity for each unique value of the predictors. The asymptotic result will be used whenever the number of successes or failures exceeds `clt.threshold`.

The degrees of freedom parameter to use in Metropolis-Hastings proposals. See details.

A two-element vector giving the probabilities of drawing from the two base sampling algorithm. The first element refers to the spike and slab algorithm. The second refers to the tailored independence Metropolis sampler. TIM is usually faster mixing, but cannot change model dimension.

Seed to use for the C++ random number generator. It should be `NULL` or an int. If `NULL` the seed value will be taken from the global `.Random.seed` object.

Extra arguments to be passed to `LogitZellnerPrior`.

Model parameters are updated using a composite of two Metropolis-Hastings updates. A data augmentation algorithm (Albert and Chib 1993) updates the entire parameter vector at once, but can mix slowly.

The second algorithm is an independence Metropolis sampler centered on the posterior mode with variance determined by posterior information matrix (Fisher information plus prior information). If `proposal.df > 0` then the tails of the proposal are inflated so that a multivariate T proposal is used instead.

At each iteration, one of the three algorithms is chosen at random. The auxiliary mixture sampler is the only one that can change the dimension of the coefficient vector. The MH algorithm only updates the coefficients that are currently nonzero.
qreg.spike

Description

MCMC algorithm for quasi-Bayesian quantile models with a 'spike-and-slab' prior that places some amount of posterior probability at zero for a subset of the regression coefficients.

Usage

qreg.spike(formula, quantile, niter, ping = niter / 10, nthreads = 0,

Value

Returns an object of class probit.spike, which inherits from lm.spike. The returned object is a list with the following elements:

- **beta**: A \( n_{iter} \times \text{ncol}(x) \) matrix of regression coefficients, many of which may be zero. Each row corresponds to an MCMC iteration.
- **prior**: The prior used to fit the model. If a prior was supplied as an argument it will be returned. Otherwise this will be the automatically generated prior based on the other function arguments.

Author(s)

Steven L. Scott

See Also

lm.spike, SpikeSlabPrior, plot.probit.spike, PlotProbitSpikeFitSummary, PlotProbitSpikeResiduals, summary.logit.spike, predict.logit.spike.

Examples

```r
if (requireNamespace("MASS")) {
  data(Pima.tr, package = "MASS")
  data(Pima.te, package = "MASS")
  pima <- rbind(Pima.tr, Pima.te)
  model <- probit.spike(type == "Yes" ~ ., data = pima, niter = 500)
  plot(model)
  plot(model, "fit")
  plot(model, "residuals")
  plot(model, "size")
  summary(model)
}
```
Arguments

- **formula**: Formula for the maximal model (with all variables included).
- **quantile**: A scalar value between 0 and 1 indicating the quantile of the conditional distribution being modeled.
- **niter**: The number of MCMC iterations to run. Be sure to include enough so you can throw away a burn-in set.
- **ping**: If positive, then print a status update to the console every ping MCMC iterations.
- **nthreads**: The number of CPU-threads to use for data augmentation. There is some small overhead to stopping and starting threads. For small data sets, thread overhead will make it faster to run single threaded. For larger data sets multi-threading can speed things up substantially. This is all machine dependent, so please experiment.
- **data**: An optional data frame, list or environment (or object coercible by `as.data.frame`) containing the variables in the model. If not found in data, the variables are taken from `environment(formula)`, typically the environment from which `qreg.spike` is called.
- **subset**: An optional vector specifying a subset of observations to be used in the fitting process.
- **prior**: An optional list such as that returned from `SpikeSlabPrior`. If missing, `SpikeSlabPrior` will be called using the extra arguments passed via `...`.
- **na.action**: A function which indicates what should happen when the data contain NAs. The default is set by the `na.action` setting of `options`, and is `na.fail` if that is unset. The factory-fresh default is `na.omit`. Another possible value is `NULL`, no action. Value `na.exclude` can be useful.
- **contrasts**: An optional list. See the `contrasts.arg` of `model.matrix.default`.
- **drop.unused.levels**: A logical value indicating whether factor levels that are unobserved should be dropped from the model.
- **initial.value**: Initial value for the MCMC algorithm. Can either be a numeric vector, a `glm` object (from which the coefficients will be used), or a `qreg.spike` object. If a `qreg.spike` object is supplied, it is assumed to be from a previous MCMC run for which `niter` additional draws are desired. If a `glm` object is supplied then its coefficients will be used as the initial values for the simulation.
seed          Seed to use for the C++ random number generator. It should be NULL or an int. If NULL the seed value will be taken from the global .Random.seed object.
...
Extra arguments to be passed to SpikeSlabPrior.

Details
Just like ordinary regression models the mean of a distribution as a linear function of X, quantile regression models a specific quantile (e.g. the 90th percentile) as a function of X.

Median regression is a special case of quantile regression. Median regression is sometimes cast in terms of minimizing \( |y - X \cdot \beta| \), because the median is the optimal action under L1 loss. Similarly, selecting quantile tau is optimal under the asymmetric loss function

\[
\rho_\tau(u) = \tau u I(u > 0) + (1 - \tau)u I(u < 0)
\]

Thus quantile regression (for a specific quantile tau) minimizes

\[
Q(\beta) = \sum_i \rho_\tau(y_i - \beta^T x_i)
\]

Bayesian quantile regression treats

\[
\exp(-2Q(\beta))
\]

as a likelihood function to which a prior distribution \( p(\beta) \) is applied. For posterior sampling, a data augmentation scheme is used where each observation is associated with a latent variable \( \lambda_i \), which has a marginal distribution of

\[
\text{Exp}(2\tau(1 - \tau)).
\]

The conditional distribution given the residual \( r = y - x\beta \) is

\[
\frac{1}{\lambda} |r| \sim \text{InvGaus}(1/|r|, 1.0)
\]

The conditional distribution of beta given complete data (lambda and y) is a weighted least squares regression, where observation i has precision \( \lambda_i \) and where observation i is offset by \( 2(\tau - 1)\lambda_i \).

Value
Returns an object of class qreg.spine, which inherits from lm.spine. The returned object is a list with the following elements

beta          A niter by ncol(x) matrix of regression coefficients, many of which may be zero. Each row corresponds to an MCMC iteration.
prior         The prior used to fit the model. If a prior was supplied as an argument it will be returned. Otherwise this will be the automatically generated prior based on the other function arguments.

Author(s)
Steven L. Scott
residuals.lm.spike

References
Parzen and Polson (2011, unpublished)

See Also
lm.spike SpikeSlabPrior, plot.qreg.spike, predict.qreg.spike.

Examples
n <- 50
x <- rnorm(n)
y <- rnorm(n, 4 * x)
model <- qreg.spike(y ~ x, quantile = .8,
   niter = 1000,
   expected.model.size = 100)

## Should get a slope near 4 and an intercept near qnorm(.8).
PlotManyTs(model$beta[-(1:100),],
   same.scale = TRUE,
   truth = c(qnorm(.8), 4))

---

residuals.lm.spike Extract lm.spike Residuals

Description
Get residuals from an lm.spike object.

Usage
## S3 method for class 'lm.spike'
residuals(
   object,
   burn = SuggestBurnLogLikelihood(object$log.likelihood),
   mean.only = FALSE,
   ...)

Arguments

object An object of class lm.spike.
burn The number of MCMC iterations in the object to be discarded as burn-in.
mean.only Logical. If TRUE then the posterior mean of each residual is returned. If FALSE then the full posterior distribution of residuals is returned.
... Unused, but present for compatibility with generic residuals function.
shrinkage.regression

Value

The posterior distribution (or posterior mean) of residuals from the model object. If mean.only is TRUE then the return value is the vector of residuals, otherwise the return value is a matrix, with rows corresponding to MCMC iterations, and columns to individual observations.

Author(s)

Steven L. Scott

See Also

lm.spike SpikeSlabPrior summary.lm.spike plot.lm.spike

Examples

```r
niter <- 1000	n <- 100	p <- 10
ngood <- 3

x <- cbind(1, matrix(rnorm(n * (p-1)), nrow=n))
beta <- rep(0, p)
good <- sample(1:p, ngood)
beta[good] <- rnorm(ngood)
sigma <- 1

y <- rnorm(n, x %*% beta, sigma)
model <- lm.spike(y ~ x - 1, niter=niter)
plot(model)
residuals(model)
residuals(model, mean.only = TRUE)
```

shrinkage.regression  Shrinking Regression Coefficients

Description

Fits a Bayesian regression model with a shrinkage prior on the coefficient. The model is

\[
y_i \sim N(x_i \beta, \sigma^2)1/\sigma^2 \sim Gamma(df/2,ss/2)g_1(\beta) \sim N(b_1,v_1)g_2(\beta) \sim N(b_2,v_2)\ldots
\]

In this notation, \(g_k(\beta) \sim N(b_k,v_k)\) indicates that the subset of coefficients in group \(k\) are a priori independent draws from the specified normal distribution. In addition, each subset-level prior may include a hyperprior, in which case the subset-level prior parameters will be updated as part of the MCMC. The hyperprior has the form of independent priors on the mean and precision parameters:

\[
b_i \sim N(prior\text{-}mean,prior\text{-}variance)1/v_i Chisq(df,\text{guess.at.sd}).
\]
Usage

ShrinkageRegression(response, predictors, coefficient.groups,
residual.precision.prior = NULL,
suf = NULL, niter, ping = niter / 10,
seed = NULL)

CoefficientGroup(indices, mean.hyperprior = NULL, sd.hyperprior = NULL,
prior = NULL)

Arguments

response The numeric vector of responses.
predictors The matrix of predictors, including an intercept term, if desired.
coefficient.groups A list of objects of type CoefficientGroup, defining the pattern in which the
coefficients should be shrunk together. Each coefficient must belong to exactly
one CoefficientGroup.
residual.precision.prior An object of type SdPrior describing the prior distribution of the residual stan-
dard deviation.
suf An object of class RegressionSuf containing the sufficient statistics for the
regression model. If this is NULL then it will be computed from response and
predictors. If it is supplied then response and predictors are not used and
can be left missing.
niter The desired number of MCMC iterations.
ping The frequency with which to print status updates.
seed The integer-valued seed (or NULL) to use for the C++ random number generator.
indices A vector of integers giving the positions of the regression coefficients that should
be viewed as exchangeable.
mean.hyperprior A NormalPrior object describing the hyperprior distribution for the average
coefficient.
sd.hyperprior An SdPrior object describing the hyperprior distribution for the standard devi-
ation of the coefficients.
prior An object of type NormalPrior giving the initial value of the distribution de-
scribing the collection of coefficients in this group. If either hyperprior is NULL
then the corresponding prior parameter will not be updated. If both hyperpriors
are non-NULL then this parameter can be left unspecified.

Value

ShrinkageRegression returns a list containing MCMC draws from the posterior distribution of
model parameters. Each of the following is a matrix, with rows corresponding to MCMC draws,
and columns to distinct parameters.
• coefficients: regression coefficients.
• residual.sd: the residual standard deviation from the regression model.
• group.means: The posterior distribution of the mean of each coefficient group. If no mean hyperprior was assigned to a particular group, then the value here will be a constant (the values supplied by the prior argument to CoefficientGroup for that group).
• group.sds: The posterior distribution of the standard deviation of each coefficient group. If no sd.hyperprior was assigned to a particular group, then the value here will be a constant (the values supplied by the prior argument to CoefficientGroup for that group).

CoefficientGroup is a configuration utility used to define which coefficients should be shrunk together. It returns an object (list) formatted in the manner expected by ShrinkageRegression.

Author(s)
Steven L. Scott

Examples

b0 <- -1
b1 <- rnorm(20, 3, .2)
b2 <- rnorm(30, -4, 7)
nobs <- 10000
beta <- c(b0, b1, b2)

X <- cbind(1, matrix(rnorm(nobs * (length(beta) - 1)), nrow = nobs, ncol = length(beta) - 1))
y.hat <- X %*% beta
y <- rnorm(nobs, y.hat, .5)

groups <- list(intercept = CoefficientGroup(1, prior = NormalPrior(0, 100)),
                first = CoefficientGroup(2:21,
                                          mean.hyperprior = NormalPrior(0, 100),
                                          sd.hyperprior = SdPrior(.2, 1)),
                second = CoefficientGroup(22:51,
                                          mean.hyperprior = NormalPrior(0, 100),
                                          sd.hyperprior = SdPrior(7, 1)))

model <- ShrinkageRegression(y, X, groups,
                              residual.precision.prior = SdPrior(.5, 1),
                              niter = 1000)

Description

A Zellner-style spike and slab prior for generalized linear models. It is intended as a base class for LogitZellnerPrior, PoissonZellnerPrior, and potential future extensions.
spike.slub.glm.prior

Usage

SpikeSlabGlmPrior(
  predictors,
  weight,
  mean.on.natural.scale,
  expected.model.size,
  prior.information.weight,
  diagonal.shrinkage,
  optional.coefficient.estimate,
  max.flips,
  prior.inclusion.probabilities)

SpikeSlabGlmPriorDirect(
  coefficient.mean,
  coefficient.precision,
  prior.inclusion.probabilities = NULL,
  expected.model.size = NULL,
  max.flips = -1)

Arguments

predictors The design matrix for the regression problem. No missing data is allowed.
weight A vector of length nrow(predictors) giving the prior weight assigned to each
observation in predictors. This should ideally match the weights from the
Fisher information (e.g. \( p \times (1-p) \)) for logistic regression, or lambda for Poisson
regression, but that depends on the model, so a typical thing to do is to set
all the weights the same.
mean.on.natural.scale Used to set the prior mean for the intercept. The mean of the response, expressed
on the natural scale. This is logit(p-hat) for logits and log(ybar) for Poissons.
expected.model.size A positive number less than ncol(x), representing a guess at the number of
significant predictor variables. Used to obtain the 'spike' portion of the spike
and slab prior.
prior.information.weight A positive scalar. Number of observations worth of weight that should be given
to the prior estimate of beta.
diagonal.shrinkage The conditionally Gaussian prior for beta (the "slab") starts with a precision
matrix equal to the information in a single observation. However, this matrix
might not be full rank. The matrix can be made full rank by averaging with
its diagonal. diagonal.shrinkage is the weight given to the diagonal in this
average. Setting this to zero gives Zellner’s g-prior.
optional.coefficient.estimate If desired, an estimate of the regression coefficients can be supplied. In most
cases this will be a difficult parameter to specify. If omitted then a prior mean
of zero will be used for all coordinates except the intercept, which will be set to mean(y).

max.flips The maximum number of variable inclusion indicators the sampler will attempt to sample each iteration. If negative then all indicators will be sampled.

prior.inclusion.probabilities A vector giving the prior probability of inclusion for each variable. If NULL then a default set of probabilities is obtained by setting each element equal to min(1,expected.model.size / ncol(x)).

coefficient.mean The prior mean of the coefficients in the maximal model (with all coefficients included).

coefficient.precision The prior precision (inverse variance) of the coefficients in the maximal model (with all coefficients included).

Details

A Zellner-style spike and slab prior for generalized linear models. Denote the vector of coefficients by $\beta$, and the vector of inclusion indicators by $\gamma$. These are linked by the relationship $\beta_i \neq 0$ if $\gamma_i = 1$ and $\beta_i = 0$ if $\gamma_i = 0$. The prior is

$$
\beta | \gamma \sim N(b, V) \\
\gamma \sim B(\pi)
$$

where $\pi$ is the vector of prior.inclusion.probabilities, and $b$ is the optional.coefficient.estimate. Conditional on $\gamma$, the prior information matrix is

$$
V^{-1} = \kappa((1 - \alpha)x^Twx/n + \alpha \text{diag}(x^Twx/n))
$$

The matrix $x^Twx$ is, for suitable choice of the weight vector $w$, the total Fisher information available in the data. Dividing by $n$ gives the average Fisher information in a single observation, multiplying by $\kappa$ then results in $\kappa$ units of "average" information. This matrix is averaged with its diagonal to ensure positive definiteness.

In the formula above, $\kappa$ is prior.information.weight, $\alpha$ is diagonal.shrinkage, and $w$ is a diagonal matrix with all elements set to prior.success.probability * (1 -prior.success.probability). The vector $b$ and the matrix $V^{-1}$ are both implicitly subscripted by $\gamma$, meaning that elements, rows, or columns corresponding to gamma = 0 should be omitted.

The "Direct" version is intended for situations where the predictors are unavailable, or if the user wants more control over the prior precision matrix.

Value

Returns an object of class SpikeSlabGlmPrior, which is a list with data elements encoding the selected prior values.

This object is intended for use as a base class for LogitZellnerPrior and PoissonZellnerPrior.
spike.slab.prior

Author(s)
Steven L. Scott

References

spike.slab.prior Create a spike and slab prior for use with lm.spike.

Description
Creates a spike and slab prior for use with lm.spike.

Usage
SpikeSlabPrior(x,
y = NULL,
expected.r2 = .5,
prior.df = .01,
expected.model.size = 1,
prior.information.weight = .01,
diagonal.shrinkage = .5,
optional.coefficient.estimate = NULL,
max.flips = -1,
mean.y = mean(y, na.rm = TRUE),
sdy = sd(as.numeric(y), na.rm = TRUE),
prior.inclusion.probabilities = NULL,
sigma.upper.limit = Inf)

SpikeSlabPriorDirect(coefficient.mean,
coefficient.precision,
prior.inclusion.probabilities,
prior.df,
sigma.guess,
max.flips = -1,
sigma.upper.limit = Inf)

ConditionalZellnerPrior(xdim,
optional.coefficient.estimate = NULL,
expected.model.size = 1,
prior.information.weight = .01,
diagonal.shrinkage = .5,
max.flips = -1,
spike.slab.prior

prior.inclusion.probabilities = NULL)

Arguments

x
The design matrix for the regression problem. Missing data is not allowed.

y
The vector of responses for the regression. Missing data is not allowed. If y is not available, you can pass y = NULL, and specify mean.y and sdy instead.

expected.r2
The expected R-square for the regression. The spike and slab prior requires an inverse gamma prior on the residual variance of the regression. The prior can be parameterized in terms of a guess at the residual variance, and a "degrees of freedom" representing the number of observations that the guess should weigh. The guess at sigma^2 is set to (1-expected.r2) * var(y).

prior.df
A positive scalar representing the prior 'degrees of freedom' for estimating the residual variance. This can be thought of as the amount of weight (expressed as an observation count) given to the expected.r2 argument.

expected.model.size
A positive number less than ncol(x), representing a guess at the number of significant predictor variables. Used to obtain the 'spike' portion of the spike and slab prior.

prior.information.weight
A positive scalar. Number of observations worth of weight that should be given to the prior estimate of beta.

diagonal.shrinkage
The conditionally Gaussian prior for beta (the "slab") starts with a precision matrix equal to the information in a single observation. However, this matrix might not be full rank. The matrix can be made full rank by averaging with its diagonal. diagonal.shrinkage is the weight given to the diagonal in this average. Setting this to zero gives Zellner's g-prior.

optional.coefficient.estimate
If desired, an estimate of the regression coefficients can be supplied. In most cases this will be a difficult parameter to specify. If omitted then a prior mean of zero will be used for all coordinates except the intercept, which will be set to mean(y).

max.flips
The maximum number of variable inclusion indicators the sampler will attempt to sample each iteration. If max.flips <= 0 then all indicators will be sampled.

mean.y
The mean of the response vector, for use in cases when specifying the response vector is undesirable.

xdim
The dimension of the predictor matrix.

sdy
The standard deviation of the response vector, for use in cases when specifying the response vector is undesirable.

prior.inclusion.probabilities
A vector giving the prior probability of inclusion for each variable.

sigma.upper.limit
The largest acceptable value for the residual standard deviation. A non-positive number is interpreted as Inf.
coefficient.mean
   The prior mean of the coefficients in the maximal model (with all variables
   included).

coefficient.precision
   The prior precision (inverse variance) of the coefficients in the maximal model
   (with all variables included).

sigma.guess
   Prior estimate of the residual standard deviation.

Value
   A list with with the components necessary to run \texttt{lm.spike}.

\texttt{SpikeSlabPrior} is intended for use in traditional regression problems, when the matrix of predic-
tors and the vector of responses are available to the modeler.

\texttt{ConditionalZellnerPrior} is intended for cases where the predictor variables are potentially un-
known, because they depend on model parameters or latent variables, for example. For models that
support \texttt{ConditionalZellnerPrior}, the underlying C++ code must know where to find the relevant
predictors on which to condition the prior.

Author(s)
   Steven L. Scott

References
   George and McCulloch (1997), "Approaches to Bayesian Variable Selection", \textit{Statistica Sinica}, 7,
   339 – 373.

   \url{http://www3.stat.sinica.edu.tw/statistica/oldpdf/A7n26.pdf}

Examples
   
   \begin{verbatim}
   x <- cbind(1, matrix(rnorm(900), ncol = 9))
   beta <- rep(0, 10)
   beta[1] <- 3
   beta[5] <- -4
   beta[8] <- 2
   y <- rnorm(100, x %*% beta)
   ## x has 10 columns, including the intercept
   prior <- SpikeSlabPrior(x, y,
      expected.model.size = 3,  # expect 3 nonzero predictors
      prior.df = .01,            # weaker prior than the default
      prior.information.weight = .01,
      diagonal.shrinkage = 0,   # use Zellner's prior
      optional.coefficient.estimate = rep(0, 10)  # shrink to zero
   )
   ## now 'prior' can be fed to 'lm.spike'
   model <- lm.spike(y ~ x - 1, niter = 1000, prior = prior)
   \end{verbatim}
spike.slab.prior.base  
**Base class for spike and slab priors**

**Description**
A base class for SpikeSlabPrior and SpikeSlabPriorBase to ensure that elements common to both classes are handled consistently. Users will not normally interact with this function.

**Usage**

```r
SpikeSlabPriorBase(number.of.variables,
    expected.r2 = .5,
    prior.df = .01,
    expected.model.size = 1,
    optional.coefficient.estimate = NULL,
    mean.y,
    sdy,
    prior.inclusion.probabilities = NULL,
    sigma.upper.limit = Inf)
```

**Arguments**

- `number.of.variables`
  The number of columns in `x`.

- `expected.r2`
  The expected R-square for the regression. The spike and slab prior requires an inverse gamma prior on the residual variance of the regression. The prior can be parameterized in terms of a guess at the residual variance, and a "degrees of freedom" representing the number of observations that the guess should weigh. The guess at $\sigma^2$ is set to \((1-\text{expected.r2}) \times \text{var}(y)\).

- `prior.df`
  A positive scalar representing the prior 'degrees of freedom' for estimating the residual variance. This can be thought of as the amount of weight (expressed as an observation count) given to the expected.r2 argument.

- `expected.model.size`
  A positive number less than `ncol(x)`, representing a guess at the number of significant predictor p variables. Used to compute a default value of `prior.inclusion.probabilities` if the latter is NULL.

- `optional.coefficient.estimate`
  If desired, an estimate of the regression coefficients can be supplied. In most cases this will be a difficult parameter to specify. If omitted then a prior mean of zero will be used for all coordinates except the intercept, which will be set to `mean.y`.

- `mean.y`
  The mean of the response vector. Used to create a default value of `optional.coefficient.estimate` when the latter is NULL.

- `sdy`
  The standard deviation of the response vector. Used along with `expected.r2` to create a prior estimate of the residual variance.
prior.inclusion.probabilities
A vector giving the prior probability of inclusion for each coefficient.

sigma.upper.limit
The largest acceptable value for the residual standard deviation. A non-positive number is interpreted as Inf.

Value
Returns an object of class SpikeSlabPriorBase, which is a list with the following elements.

- prior.inclusion.probabilities: A vector giving the prior probability of inclusion for each coefficient.
- mu: A vector giving the prior mean of each coefficient conditional on inclusion.
- sigma.guess: A prior estimate of the residual standard deviation.
- prior.df: The number of observations worth of weight to be given to sigma.guess.

Author(s)
Steven L. Scott

References

Spline Basis Expansions

Description
Spline basis expansions of a continuous variable.

Usage
BsplineBasis(x, knots = NULL, numknots = 3)
MsplineBasis(x, knots = NULL, numknots = 3)
IsplineBasis(x, knots = NULL, numknots = 3)

## S3 method for class 'SplineBasis'
knots(Fn, ...)

spliunes
Spline Basis Expansions
Arguments

- **x**  
  A numeric vector to be expanded.

- **knots**  
  A numeric vector of knots defining the expansion. The smallest and largest elements in knots defines the range of the expansion. These knots are (notionally) replicated infinitely many times.

- **numknots**  
  If the knot vector is NULL then create a vector of length numknots that partitions \( x \) into \( \text{numknots} + 1 \) equiprobable segments.

- **Fn**  
  A spline basis matrix.

- **...**  
  Unused, but required to match the signature of the knots generic function in the stats package.

Details

B-splines are the basis most commonly used for additive regression models.

M-splines are an alternative to B-splines, but are rarely used.

I-splines are integrated M-splines. These are monotonic functions, which is useful in monotonic regression problems. If all regression coefficients are positive then the resulting function is nondecreasing.

Value

- `XsplineBasis` returns a matrix formed by the spline basis expansion of \( x \).

- `knots(Fn)` returns the knots attribute of \( Fn \), which might be useful in a second call to the basis expansion function.

Author(s)

Steven L. Scott

References

Bsplines are described in deBoor (2001), "A Practical Guide to Splines". Springer.

Msplines and Isplines are reviewed by Ramsay (1988), Statistical Science pp. 425-461.

Examples

```r
# Plot the B-spline basis for x with knots determined by 3 quantiles.
x <- sort(rnorm(1000))
basis <- BsplineBasis(x, numknots=3)
par(mfrow=c(2,3))
for(i in 1:5) plot(x, basis[, i], type="l")

# Plot the I-spline basis for x with the same knots.
basis <- IsplineBasis(x, numknots=3)
par(mfrow=c(2,3))
for(i in 1:5) plot(x, basis[, i], type="l")
```
# Bring you own knots...
basis <- BsplineBasis(x, knots = quantile(x, c(.2, .5, .8, .9)))
par(mfrow=c(2,3))
for(i in 1:6) plot(x, basis[, i], type="l")

knots(basis)

---

**student.spike.slab.prior**

*Spike and Slab Prior for Student-T Regression*

---

**Description**

A Zellner-style spike and slab prior for regression models with Student-t errors.

**Usage**

```r
StudentSpikeSlabPrior(predictor.matrix, 
response.vector = NULL, 
expected.r2 = .5, 
prior.df = .01, 
expected.model.size = 1, 
prior.information.weight = .01, 
diagonal.shrinkage = .5, 
optional.coefficient.estimate = NULL, 
max.flips = -1, 
mean.y = mean(response.vector, na.rm = TRUE), 
sdy = sd(as.numeric(response.vector), na.rm = TRUE), 
prior.inclusion.probabilities = NULL, 
sigma.upper.limit = Inf, 
degrees.of.freedom.prior = UniformPrior(.1, 100))
```

**Arguments**

- **predictor.matrix**
  - The design matrix for the regression problem. Missing data is not allowed.

- **response.vector**
  - The vector of responses for the regression. Missing data is not allowed. If `response.vector` is not available, you can pass `response.vector = NULL`, and specify `mean.y` and `sdy` instead.

- **expected.r2**
  - The expected R-square for the regression. The spike and slab prior requires an inverse gamma prior on the residual variance of the regression. The prior can be parameterized in terms of a guess at the residual variance, and a "degrees of freedom" representing the number of observations that the guess should weigh. The guess at sigma^2 is set to \((1\text{-}expected.r2) \times \text{var}(y)\).
prior.df
A positive scalar representing the prior 'degrees of freedom' for estimating the residual variance. This can be thought of as the amount of weight (expressed as an observation count) given to the expected \( r^2 \) argument.

expected.model.size
A positive number less than \( \text{nvec}(x) \), representing a guess at the number of significant predictor variables. Used to obtain the 'spike' portion of the spike and slab prior.

prior.information.weight
A positive scalar. Number of observations worth of weight that should be given to the prior estimate of beta.

diagonal.shrinkage
The conditionally Gaussian prior for beta (the "slab") starts with a precision matrix equal to the information in a single observation. However, this matrix might not be full rank. The matrix can be made full rank by averaging with its diagonal. diagonal.shrinkage is the weight given to the diagonal in this average. Setting this to zero gives Zellner's g-prior.

optional.coefficient.estimate
If desired, an estimate of the regression coefficients can be supplied. In most cases this will be a difficult parameter to specify. If omitted then a prior mean of zero will be used for all coordinates except the intercept, which will be set to mean(y).

max.flips
The maximum number of variable inclusion indicators the sampler will attempt to sample each iteration. If max.flips <= 0 then all indicators will be sampled.

mean.y
The mean of the response vector, for use in cases when specifying the response vector is undesirable.

sdy
The standard deviation of the response vector, for use in cases when specifying the response vector is undesirable.

prior.inclusion.probabilities
A vector giving the prior probability of inclusion for each variable.

sigma.upper.limit
The largest acceptable value for the residual standard deviation. A non-positive number is interpreted as \( \text{Inf} \).

degrees.of.freedom.prior
An object of class DoubleModel representing the prior distribution for the Student T tail thickness (or "degrees of freedom") parameter.

Value
A SpikeSlabPrior with degrees.of.freedom.prior appended.

Author(s)
Steven L. Scott
suggest.burn

References


---

suggest.burn  Suggest Burn-in

Description

Suggest a burn-in period for a Bayesian neural network model.

Usage

SuggestBurn(model)

Arguments

model  An object inheriting from class BayesNnet.

Details

See SuggestBurnLogLikelihood for details of the on how the burn-in period is suggested. In this case the negative the residual standard deviation is used as a proxy for log likelihood.

Value

A non-negative integer less than the number of MCMC draws.

Author(s)

Steven L. Scott

See Also

SuggestBurnLogLikelihood
SummarizeSpikeSlabCoefficients

Numerical summaries of coefficients from a spike and slab regression.

Description

Produces a summary of the marginal distribution of model coefficients from a spike and slab regression.

Usage

SummarizeSpikeSlabCoefficients(beta, burn = 0, order = TRUE)

Arguments

beta A matrix containing MCMC draws of regression coefficients. Each row is an MCMC draw. Each column is a coefficient.

burn The number of MCMC iterations in the object to be discarded as burn-in.

order Logical. If TRUE then the coefficients are presented in order of their posterior inclusion probabilities. Otherwise the order of the coefficients is the same as in object.

Value

A five-column matrix with rows representing model coefficients. The first two columns are the posterior mean and standard deviation of each coefficient, including the point mass at zero. The next two columns are the posterior mean and standard deviations conditional on the coefficient being nonzero. The last column is the probability of a nonzero coefficient.

Author(s)

Steven L. Scott

See Also

lm.spike summary.lm.spike

Examples

```
n <- 100
p <- 10
ngood <- 3
niter <- 1000
sigma <- 2

x <- cbind(1, matrix(rnorm(n * (p-1)), nrow=n))
beta <- c(rnorm(ngood), rep(0, p - ngood))
y <- rnorm(n, x %*% beta, sigma)
```
\texttt{x <- x[,,-1]}
\texttt{model <- lm.spike(y ~ x, niter=niter)}
\texttt{plot(model)}
\texttt{plot.ts(model$beta)}
\texttt{hist(model$sigma) \# should be near 8}
\texttt{summary(model)}
\texttt{SummarizeSpikeSlabCoefficients(model$beta, burn = 100)}

---

**summary.lm.spike**

Numerical summaries of the results from a spike and slab regression.

**Description**

Produces a summary of the marginal distribution of model coefficients from a spike and slab regression.

**Usage**

\texttt{## S3 method for class \texttt{\textquotesingle lm.spike\textquotesingle}}
\texttt{summary(object, burn = 0, order = TRUE, \ldots)}

**Arguments**

- \texttt{object} An object of class \texttt{lm.spike}.
- \texttt{burn} The number of MCMC iterations in the object to be discarded as burn-in.
- \texttt{order} Logical. If \texttt{TRUE} then the coefficients are presented in order of their posterior inclusion probabilities. Otherwise the order of the coefficients is the same as in \texttt{object}.
- \ldots Unused. Present for compatibility with generic \texttt{summary}().

**Value**

Returns a list with the following elements:

- \texttt{coefficients} A five-column matrix with rows representing model coefficients. The first two columns are the posterior mean and standard deviation of each coefficient, including the point mass at zero. The next two columns are the posterior mean and standard deviations conditional on the coefficient being nonzero. The last column is the probability of a nonzero coefficient.
- \texttt{residual.sd} A summary of the posterior distribution of the residual standard deviation parameter.
- \texttt{rsquare} A summary of the posterior distribution of the R^2 statistic: 1 - residual.sd^2 / var(y)

**Author(s)**

Steven L. Scott
See Also

lm.spike SpikeSlabPrior plot.lm.spike predict.lm.spike

Examples

```r
n <- 100
p <- 10
ngood <- 3
niter <- 1000
sigma <- 2
x <- cbind(1, matrix(rnorm(n * (p-1)), nrow=n))
beta <- c(rnorm(ngood), rep(0, p - ngood))
y <- rnorm(n, x %*% beta, sigma)
x <- x[, -1]
model <- lm.spike(y ~ x, niter=niter)
plot(model)
plot.ts(model$beta)
hist(model$sigma) # should be near 8
summary(model)
```

summary.logit.spike

*Numerical summaries of the results from a spike and slab logistic regression.*

Description

Produces a summary of the marginal distribution of model coefficients from a spike and slab logistic regression.

Usage

```r
## S3 method for class 'logit.spike'
summary(object, 
  burn = 0, 
  order = TRUE, 
  cutpoint.scale = c("probability", "logit"), 
  cutpoint.basis = c("sample.size", "equal.range"), 
  number.of.buckets = 10, 
  coefficients = TRUE, 
  ...)

## S3 method for class 'probit.spike'
summary(object, 
  burn = 0, 
  order = TRUE, 
  cutpoint.scale = c("probability", "probit"),
```


cutpoint.basis = c("sample.size", "equal.range"),
number.of.buckets = 10,
coefficients = TRUE,
...)

Arguments

object An object of class logit.spike or probit.spike.
burn The number of MCMC iterations in the object to be discarded as burn-in.
order Logical. If TRUE then the coefficients are presented in order of their posterior inclusion probabilities. Otherwise the order of the coefficients is the same as in object.
cutpoint.scale The scale that should be used to determine the buckets for the comparison of predicted and actual probabilities.
cutpoint.basis How should the buckets be determined in the comparison of predicted to actual probabilities? If "sample.sample", then each bucket contains the same fraction of data. If "equal.range" then the buckets are formed by partitioning the range of the predicted probabilities, and each bucket occupies the same amount of space on the real line.
number.of.buckets The number of buckets to use in the comparison of predicted to actual probabilities.
coefficients Logical value indicating whether the coefficient summary should be included in the output. It can be useful to suppress the coefficients if there are many of them.
...
Unused. Present for compatibility with generic summary().

Value

Returns a list with the following elements

- coefficients: A five-column matrix summarizing the model coefficients, produced by SummarizeSpikeSlabCoefficients.
- null.log.likelihood: The log likelihood of the null binomial model evaluated at the MLE.
- mean.log.likelihood: The average value of log likelihood visited by the sampler.
- max.log.likelihood: The largest log likelihood value visited by the sampler.
- deviance.r2: The deviance R-square obtained by taking (null.likelihood - mean.log.likelihood) / null.log.likelihood
- deviance.r2.distribution: The value of the deviance R-square statistic at each point visited by the MCMC chain. This is not printed by the print method.
- predicted.vs.actualA table obtained by partitioning the data into buckets, and comparing the average predicted probability with the empirical success rate in each bucket.

Author(s)

Steven L. Scott
See Also

logit.spike probit.spike SpikeSlabPrior

Examples

```r
n <- 100
p <- 10
ngood <- 3
niter <- 1000

x <- cbind(1, matrix(rnorm(n * (p-1)), nrow=n))
beta <- c(rnorm(ngood), rep(0, p - ngood))
prob <- plogis(x %*% beta)
y <- runif(n) < prob
x <- x[, -1]
model <- logit.spike(y ~ x, niter=niter)
summary(model)
```
Index

*Topic models
lm.spike, 7
logit.spike, 10
mlm.spike, 15
mlm.spike.slab.prior, 20
model.matrix.glm.spike, 23
nnet, 26
plot.BayesNnet, 31
plot.coefficients, 32
plot.lm.spike, 34
plot.lm.spike.residuals, 35
plot.logit.spike, 36
plot.logit.spike.fit.summary, 37
plot.logit.spike.residuals, 38
plot.marginal.inclusion.probabilities, 40
plot.poisson.spike, 41
plot.qreg.spike, 42
PlotModelSize, 43
poisson.spike, 44
probit.spike, 52
qreg.spike, 54
spike.slab.prior, 63
spliunes, 67
student.spike.slab.prior, 69
Random.seed, 12, 17, 45, 53, 56
as.data.frame, 45
as.factor, 24
barplot, 38, 40
BayesNnet, 27, 32
BayesNnet (nnet), 26
boxplot, 33
BsplineBasis(spliunes), 67
CoefficientGroup, 59
CoefficientGroup
(shrinkage.regression), 58
ConditionalZellnerPrior
(spike.slab.prior), 63
DoubleModel, 6, 70
formula, 16
GetPredictorMatrix (model.matrix), 22
glm, 11, 45, 52, 53, 55
HiddenLayer, 27
HiddenLayer (nnet), 26
hist, 43
independent.spike.slab.prior, 2
independent.student.spike.slab.prior, 5

*Topic regression
lm.spike, 7
logit.spike, 10
mlm.spike, 15
mlm.spike.slab.prior, 20
model.matrix.glm.spike, 23
plot.BayesNnet, 31
plot.coefficients, 32
plot.lm.spike, 34
plot.lm.spike.residuals, 35
plot.logit.spike, 36
plot.logit.spike.fit.summary, 37
plot.logit.spike.residuals, 38
plot.marginal.inclusion.probabilities, 40
plot.poisson.spike, 41
plot.qreg.spike, 42
PlotModelSize, 43
poisson.spike, 44
probit.spike, 52
qreg.spike, 54
spike.slab.prior, 63
spliunes, 67
student.spike.slab.prior, 69
as.data.frame, 45
as.factor, 24
barplot, 38, 40
BayesNnet, 27, 32
BayesNnet (nnet), 26
boxplot, 33
BsplineBasis(spliunes), 67
CoefficientGroup, 59
CoefficientGroup
(shrinkage.regression), 58
ConditionalZellnerPrior
(spike.slab.prior), 63
DoubleModel, 6, 70
formula, 16
GetPredictorMatrix (model.matrix), 22
glm, 11, 45, 52, 53, 55
HiddenLayer, 27
HiddenLayer (nnet), 26
hist, 43
independent.spike.slab.prior, 2
independent.student.spike.slab.prior, 5
INDEX

IndependentSpikeSlabPrior, 6, 8, 17, 21
IndependentSpikeSlabPrior  
(independent.spike.slab.prior), 2
InverseWishartPrior, 25
IsplineBasis(spliunes), 67
knots(spliunes), 67
lm.spike, 7, 13, 18, 22, 23, 33–35, 38, 41, 44, 46, 50, 51, 54, 57, 58, 72, 74
logit.spike, 10, 11, 15, 36–39, 76
logit.zellner.prior, 13
LogitPrior, 52
LogitPrior (logit.zellner.prior), 13
LogitZellnerPrior, 11, 12, 52, 53, 62
LogitZellnerPrior  
(logit.zellner.prior), 13
mlm.spike, 15, 21
mlm.spike.slab.prior, 20
model.matrix, 22, 22
model.matrix.default, 8, 11, 17, 23, 28, 45, 53, 55
model.matrix.glm.spike, 23
M splineBasis(spliunes), 67
MultinomialLogitSpikeSlabPrior, 17
MultinomialLogitSpikeSlabPrior  
(mlm.spike.slab.prior), 20
MvnPrior, 25, 27
na.excl ude, 50
na.fail, 50
na.omit, 50
nested.regression, 24
NestedRegression(nested.regression), 24
nnet, 26
NormalPrior, 59
OdaOptions(lm.spike), 7
options, 50
partial.dependence.plot, 29
PartialDependencePlot, 32
PartialDependencePlot  
(partial.dependence.plot), 29
plot, 32, 35, 39
plot.BayesNnet, 28, 31, 31
plot.coefficients, 32
plot.igraph, 32
plot.lm.spike, 9, 18, 22, 34, 35, 44, 46, 50, 58, 74
plot.lm.spike.residuals, 35
plot.logit.spike, 13, 36, 39
plot.logit.spike.fit.summary, 37
plot.logit.spike.residuals, 38
plot.marginal.inclusion.probabilities, 40
plot.poisson.spike, 41
plot.probit.spike, 54
plot.probit.spike(plot.logit.spike), 36
plot.qreg.spike, 42, 57
PlotBayesNnetPredictions  
(plot.BayesNnet), 31
PlotBayesNnetResiduals  
(plot.BayesNnet), 31
PlotLmSpikeCoefficients, 34, 42, 43
PlotLmSpikeCoefficients  
(plot.coefficients), 32
PlotLmSpikeResiduals, 34
PlotLmSpikeResiduals  
(plot.lm.spike.residuals), 35
PlotLogitSpikeFitSummary, 13, 36, 37
PlotLogitSpikeFitSummary  
(plot.logit.spike.fit.summary), 37
PlotLogitSpikeResiduals, 13, 36, 37
PlotLogitSpikeResiduals  
(plot.logit.spike.residuals), 38
PlotMarginalInclusionProbabilities, 31, 34, 36, 37, 41–43
PlotMarginalInclusionProbabilities  
(plot.marginal.inclusion.probabilities), 40
PlotModelSize, 34, 36, 37, 41–43, 43
PlotNetworkStructure(plot.BayesNnet), 31
PlotProbitSpikeFitSummary, 54
PlotProbitSpikeFitSummary  
(plot.logit.spike.fit.summary), 37
PlotProbitSpikeResiduals, 54
PlotProbitSpikeResiduals  
(plot.logit.spike.residuals), 38
poisson.spike, 41, 44, 45, 48
poisson.zellner.prior, 46