Package ‘varbvs’

May 29, 2016

Version 2.0.0
Date 2016-05-26
Title Large-Scale Bayesian Variable Selection Using Variational Methods
Author Peter Carbonetto, Matthew Stephens
Maintainer Peter Carbonetto <peter.carbonetto@gmail.com>
Description Fast algorithms for fitting Bayesian variable selection models and computing Bayes factors, in which the outcome (or response variable) is modeled using a linear regression or a logistic regression. The algorithms are based on the variational approximations described in “Scalable variational inference for Bayesian variable selection in regression, and its accuracy in genetic association studies” (P. Carbonetto & M. Stephens, 2012, <DOI:10.1214/12-BA703>). This software has been applied to large data sets with over a million variables and thousands of samples.

Depends R (>= 3.1.0)
Imports methods, stats, graphics, lattice, latticeExtra
Suggests glmnet, qtl
License GPL (>= 3)
NeedsCompilation yes
URL http://github.com/pcarbo/varbvs
Repository CRAN
Date/Publication 2016-05-28 17:52:03

R topics documented:

varbvs-package ....................................................... 2
bayesfactor ......................................................... 3
cred ................................................................. 4
cytokine ............................................................. 5
leukemia .............................................................. 5
normalizelogweights ................................................ 6
plot.varbvs ......................................................... 7
Description

Fast algorithms for fitting Bayesian variable selection models and computing Bayes factors, in which the outcome (or response variable) is modeled using a linear regression or a logistic regression. The algorithms are based on the variational approximations described in "Scalable variational inference for Bayesian variable selection in regression, and its accuracy in genetic association studies" (P. Carbonetto and M. Stephens, Bayesian Analysis 7, 2012, pages 73-108). This software has been applied to large data sets with over a million variables and thousands of samples.

Details

The main functionality of this package is implemented in function `varbvs`. This function selects the most appropriate algorithm for the data set and selected model (linear or logistic regression). See `help(varbvs)` for details. The varbvs interface is intended to resemble interface for `glmnet`, the popular package for fitting generalized linear models.

For more details about this package, including the license and a list of available functions, see `help(package=varbvs)`.

Author(s)

Peter Carbonetto <peter.carbonetto@gmail.com>

References

bayesfactor

Compute numerical estimate of Bayes factor.

Description

Computes numerical estimate of

\[ BF = \frac{Pr(data|H1)}{Pr(data|H0)}, \]

the probability of the data given the "alternative" hypothesis (H1) over the probability of the data given the "null" hypothesis (H0). This is also known as a Bayes factor (see Kass & Raftery, 1995). Here we assume that although these probabilities cannot be computed analytically because they involve intractable integrals, we can obtain reasonable estimates of these probabilities with a simple numerical approximation over some latent variable assuming the prior over this latent variable is uniform. The inputs are the log-probabilities

\[ Pr(data, Z0|H0) = Pr(data|Z0, H0) \times Pr(Z0|H0), \]
\[ Pr(data, Z1|H1) = Pr(data|Z1, H1) \times Pr(Z1|H1), \]

where \( Pr(Z0 \mid H0) \) and \( Pr(Z1 \mid H1) \) are uniform over all \( Z0 \) and \( Z1 \).

Alternatively, this function can be viewed as computing an importance sampling estimate of the Bayes factor; see, for example, R. M. Neal, "Annealed importance sampling", Statistics and Computing, 2001. This formulation described above is a special case of importance sampling when the settings of the latent variable \( Z0 \) and \( Z1 \) are drawn from the same (uniform) distribution as the prior, \( Pr(Z0 \mid H0) \) and \( Pr(Z1 \mid H1) \), respectively.

Usage

bayesfactor (logw0, logw1)

Arguments

logw0 log-probabilities or log-importance weights under H0.

logw1 log-probabilities or log-importance weights under H1.

Value

The estimated Bayes factor.

Author(s)

Peter Carbonetto <peter.carbonetto@gmail.com>

References


See Also

varbvs, normalizelogweights

cred

Estimate credible interval.

Description

Estimate credible interval from weighted samples.

Usage

cred(x, x0, w = NULL, cred.int = 0.95)

Arguments

x Vector of random samples of variable.

x0 Mean of median of variable.

w Weight > 0 assigned to each sample. If w = NULL, all weights are the same.

cred.int Credible interval must contain probability mass of at least this amount. A number between 0 and 1.

Details

Credible interval [a,b] is defined as smallest interval containing x0 that contains cred.int of the probability mass. Note that the credible interval is not necessarily symmetric about x0. (Other definitions of the credible interval are possible.)

The algorithm is quadratic in the length of x (and w), so should not be used for large numbers of samples.

Value

list(a = a,b = b).

Author(s)

Peter Carbonetto <peter.carbonetto@gmail.com>

References


Examples

```r
x <- rnorm(100)
out <- cred(x, mean(x), cred.int = 0.68)
```
**cytokine**

*Cytokine signaling genes SNP annotation.*

**Description**

This gene set was selected in Carbonetto and Stephens (2013) from an interrogation of 3,158 derived from 8 publicly available pathway databases.

**Usage**

```r
data(cytokine)
```

**Format**

```r
cytokine[i] = 1 if SNP i lies within 100 kb of a gene in the "Cytokine signaling in immune system" gene set, and cytokine[i] = 0 otherwise.
```

**Source**

Pathway id 75790 from the Reactome database, or pathway id 366171 from the BioSystems database.

**References**


**Examples**

```r
# See demo.cytokine.R vignette.
```

---

**leukemia**

*Expression levels recorded in leukemia patients.*

**Description**

Expression levels recorded for 3,571 genes in 72 patients with leukemia (Golub et al, 1999). The binary outcome encodes the disease subtype: acute lymphoblastic leukemia (ALL) or acute myeloid leukemia (AML).

**Usage**

```r
data(leukemia)
```
Format

Data are represented as a 72 x 3,571 matrix of gene expression values, and a vector y of 72 binary
disease outcomes.

Source

These are the preprocessed data of Dettling (2004) retrieved from the supplementary materials
accompanying Friedman et al (2010).

References


J. Friedman, T. Hastie and R. Tibshirani (2010). Regularization paths for generalized linear models


Examples

# See demo.leukemia.R vignette.

```

normalizeLogWeights
```

Compute normalized probabilities.

Description

Compute normalized probabilities from unnormalized log-probabilities.

Usage

```
normalizeLogWeights(logw)
```

Arguments

- `logw` Vector of unnormalized log-probabilities.

Details

Guards against underflow or overflow by adjusting the log-probabilities so that the largest probabil-
ity is 1.

Value

Normalized probabilities such that the sum is equal to 1.
plot.varbvs

Author(s)
Peter Carbonetto <peter.carbonetto@gmail.com>

References

Examples
```r
logw <- rnorm(6)
w <- normalize_logweights(logw)
```

plot.varbvs  
**Summarize variable selection results in a single plot.**

Description
Generate a single plot that summarizes the results of fitting the Bayesian variable selection model to the data. When the variables are genetic markers, the groups are chromosomes, and the posterior probabilities are plotted on the vertical axis (typically on the logarithmic scale), the figure resembles a "Manhattan plot" typically used to summarize the results of a genome-wide association study or quantitative trait locus (QTL) mapping study.

Usage
```r
# S3 method for class 'varbvs'
plot(x, score, groups, vars = NULL, var.labels,
     draw.threshold = NA, gap = 0, col = "midnightblue", pch = 20,
     xlab = "", ylab = "",
     abline.args = list(lty = "dotted", col = "orangered"),
     vars.xplot.args = list(pch = 20, col = "magenta"),
     vars.ltext.args = list(col = "black", pos = 4, cex = 0.5), ...)
```

Arguments
- `x`  
  Output of function `varbvs`.
- `score`  
  Value to plot on vertical axis. Must be a numeric vector with one entry for each variable. If missing, the posterior inclusion probability for each variable is plotted in the vertical axis, in which this probability is averaged over hyperparameter settings, treating `x$logw` as (unnormalized) log-marginal probabilities. As alternative, set `score` to the posterior inclusion probabilities that ignore correlations, using `varbvs.indep`.
- `groups`  
  Group the variables in the plot according to this argument. This must be a vector with one entry for each variable. If missing, all variables are treated as a single group. This is useful for grouping the genetic markers by chromosome in a genome-wide association study.
**plot.varbvs**

vars | Indices of variables to highlight and label. By default, `vars = NULL`, meaning no variables are highlighted.

var.labels | Labels to accompany the highlighted variables only. If missing, labels are retrieved from `x`. If `var.labels = NULL`, no labels are plotted.

draw.threshold | Plot a horizontal line at this location on the vertical axis.

gap | Amount of space to leave between each group of variables in the plot.

col | Argument passed to `xyplot` specifying color of points.

pch | Argument passed to `xyplot` specifying symbol type.

xlab | Argument passed to `xyplot` specifying horizontal axis title.

ylab | Argument passed to `xyplot` specifying vertical axis title.

abline.args | Additional arguments passed to `panel.abline` specifying how to draw the horizontal line at the location specified by `draw.threshold`.

vars.xyplot.args | Additional arguments passed to `xyplot` for drawing the highlighted variables.

vars.ltext.args | Additional arguments passed to `ltext` for specifying how the labels of the highlighted variables are drawn in the plot.

... | Additional arguments passed to `xyplot` for drawing the un-highlighted variables.

**Details**

Note that `plot.varbvs` uses function `xyplot` from the `lattice` package, and `as.layer` from the `latticeExtra` package.

**Value**

An object of class "trellis" generated by functions `xyplot` and `as.layer`.

**Author(s)**

Peter Carbonetto <peter.carbonetto@gmail.com>

**References**


**See Also**

`varbvs`, `summary.varbvs`, `varbvsindep`
predict.varbvs

Make predictions from a model fitted by varbvs.

Description

This function predicts outcomes (Y) given the observed variables (X) and observed covariates (Z), and a model fitted using varbvs.

Usage

```r
## S3 method for class 'varbvs'
predict(object, X, Z = NULL, ...)
```

Arguments

- `object` Output of function varbvs.
- `X` n x p input matrix, in which p is the number of variables, and n is the number of samples for which predictions will be made using the fitted model. X cannot be sparse, and cannot have any missing values (NA).
- `Z` n x m covariate data matrix, where m is the number of covariates. Do not supply an intercept as a covariate (i.e., a column of ones), because an intercept is automatically included in the regression model. For no covariates, set Z = NULL.
- `...` Other arguments to generic predict function. These extra arguments are not used here.

Details

For the logistic regression model, we do not provide classification probabilities $Pr(Y = 1|X, Z)$ because these probabilities are not necessarily calibrated under the variational approximation.

The predictions are computed by averaging over the hyperparameter settings, treating `object$logw` as (unnormalized) log-marginal probabilities. See varbvs for more details about correctly using `object$logw` for approximate numerical integration over the hyperparameters, for example by treating these as importance weights.

Value

Vector containing the predicted outcomes for all samples. For `family = "binomial"`, all vector entries are 0 or 1.

Author(s)

Peter Carbonetto <peter.carbonetto@gmail.com>

References

Description

Generate matrices of pseudorandom values.

Usage

rand(m, n)
randn(m, n)

Arguments

m Number of matrix rows.
n Number of matrix columns.

Details

Function \texttt{rand} returns a matrix containing pseudorandom values drawn from the standard uniform distribution (using \texttt{runif}). Function \texttt{randn} returns a matrix containing pseudorandom values drawn from the standard normal distribution (using \texttt{rnorm}).

Value

An \( m \times n \) numeric matrix.

Author(s)

Peter Carbonetto \(<\text{peter.carbonetto@gmail.com}>\)

References


Examples

\begin{verbatim}
x <- rand(10, 5) y <- randn(10, 5)
\end{verbatim}
Summary

Generate a summary of the Bayesian variable selection model fitted using variational approximation methods.

Usage

```r
summarize_affine_object(type = "class", 
                         cred.int = 0.95, nv = 5, nr = 1000, ...) 
print_summarize_affine_object(x, digits = 3, ...) 
```

Arguments

- `object`: Output of function `varbvs`
- `cred.int`: Size of credible interval (number between 0 and 1).
- `nv`: Show detailed statistics for top `nv` variables, ranked according to their posterior inclusion probabilities.
- `nr`: Number of Monte Carlo samples to draw to estimate credible intervals for coefficients of selected variables.
- `x`: Output of function `summary.varbvs`.
- `digits`: Number of digits shown when printing posterior probabilities of top `nv` variables.
- `...`: Additional print arguments.

Details

The printed summary is divided into three parts. The first part summarizes the data and optimization settings. It also reports the hyperparameter setting that yields the largest marginal likelihood—more precisely, the approximate marginal likelihood computed using the variational method. For the linear regression only (`family = "gaussian"`), it reports the estimated proportion of variance in the outcome explained by the model (PVE), and the credible interval of the PVE estimate brackets. Note that this is the PVE in the outcome after removing variance in the outcome due to linear effects of the covariates.

The second part summarizes the approximate posterior distribution of the hyperparameters (sigma, sa, logodds). The "estimate" column is the value averaged over hyperparameter settings, treating `object_logw` as (unnormalized) log-marginal probabilities. The next column, labeled "Pr>x", where `x = cred.int` gives the credible interval based on these weights (computed using function `cred`).

The third part summarizes the variable selection results. This includes the total number of variables included in the model at different posterior probability thresholds, and a more detailed summary of
The variables included in the model with highest posterior probability. For family = "gaussian", the "PVE" column gives the estimated proportion of variance in the outcome explained by the variable (conditioned on being included in the model). Again, this is the PVE after removing variance in the outcome due to linear effects of the covariates. Finally, note that the credible intervals reported in the right-most column are Monte Carlo estimates, so the interval will be slightly different each time summary is called; a more accurate estimate can be obtained by setting input nr to a larger number, at the cost of increased computation time.

Value

An object of class summary.varbvs, to be printed by print.summary.varbvs.

Author(s)

Peter Carbonetto <peter.carbonetto@gmail.com>

References


See Also

varbvs, varbvscoefcred

Examples

# See help(varbvs) for examples.

varbvs

Fit variable selection model using variational approximation methods.

Description

Compute fully-factorized variational approximation for Bayesian variable selection in linear (family = gaussian) or logistic regression (family = binomial). More precisely, find the "best" fully-factorized approximation to the posterior distribution of the coefficients, with spike-and-slab priors on the coefficients. By "best", we mean the approximating distribution that locally minimizes the Kullback-Leibler divergence between the approximating distribution and the exact posterior.

Usage

varbvs (X, Z, y, family = c("gaussian","binomial"), sigma, sa, logodds, alpha, mu, eta, update.sigma, update.sa, optimize.eta, initialize.params, nr = 100, sa0 = 0, n0 = 0, tol = 1e-4, maxiter = 1e4, verbose = TRUE)
**Arguments**

**X**

n x p input matrix, where n is the number of samples, and p is the number of variables. X cannot be sparse, and cannot have any missing values (NA).

**Z**

n x m covariate data matrix, where m is the number of covariates. Do not supply an intercept as a covariate (i.e., a column of ones), because an intercept is automatically included in the regression model. For no covariates, set $Z = \text{NULL}$.

**y**

Vector of length n containing observations of binary (family = "binomial") or continuous (family = "gaussian") outcome. For a binary outcome, all entries of y must be 0 or 1.

**family**

"gaussian" for linear regression model, or "binomial" for logistic regression model.

**sigma**

Candidate settings for the residual variance parameter. Must be of the same length as inputs sa and logodds (or have length equal to the number of columns of logodds). Only used for linear regression, and will generate an error if family = "binomial". If missing, residual variance parameter is automatically fitted to data by computing approximate maximum-likelihood (ML) estimate.

**sa**

Hyperparameter sa is the prior variance of regression coefficients for variables that are included in the model. This prior variance is always scaled by sigma (for logistic regression, we take sigma = 1). This input specifies the candidate settings for sa, of the same length as inputs sigma and logodds (or have length equal to the number of columns of logodds). If missing, prior variance is automatically fitted to data by compute approximate maximum (ML) estimates, or maximum a posteriori estimates when $nP > P$ and $saP > P$.

**logodds**

Hyperparameter logodds is the prior log-odds that a variable is included in the regression model; it is defined as $\text{logodds} = \log_{10}(q/(1 - q))$, where q is the prior probability that a variable is included in the regression model. Note that we use the base-10 logarithm instead of the natural logarithm because it is usually more natural to specify prior log-odds settings in this way. The prior log-odds may also be specified separately for each variable, which is useful if there is prior information about which variables are most relevant to the outcome. This is accomplished by setting logodds to a p x ns matrix, where p is the number of variables, and ns is the number of hyperparameter settings. Note it is not possible to fit the logodds parameter; if logodds input is not provided as input, then it is set to the default value when sa and sigma are missing, and otherwise an error is generated.

**alpha**

Good initial estimate for the variational parameter alpha for each hyperparameter setting. Either missing, or a p x ns matrix, where p is the number of variables, and ns is the number of hyperparameter settings.

**mu**

Good initial estimate for the variational parameter mu for each hyperparameter setting. Either missing, or a p x ns matrix, where p is the number of variables, and ns is the number of hyperparameter settings.

**eta**

Good initial estimate of the additional free parameters specifying the variational approximation to the logistic regression factors. Either missing, or an n x ns matrix, where n is the number of samples, and ns is the number of hyperparameter settings.
update.sigma  Setting this to TRUE ensures that sigma is always fitted to data, in which case input vector sigma is used to provide initial estimates.

update.sa    Setting this to TRUE ensures that sa is always fitted to data, in which case input vector sa is used to provide initial estimates.

optimize.eta When optimize.eta = TRUE, eta is fitted to the data during the inner loop coordinate ascent updates, even when good estimates of eta are provided as input.

initialize.params
If TRUE, the initialization stage of the variational inference algorithm (see below) will be skipped, which saves computation time for large data sets.

nr  Number of samples of "model PVE" to draw from posterior.

sa0  Scale parameter for a scaled inverse chi-square prior on hyperparameter sa. Must be >= 0.

n0  Number of degrees of freedom for a scaled inverse chi-square prior on hyperparameter sa. Must be >= 0.

tol  Convergence tolerance for inner loop.

maxiter  Maximum number of inner loop iterations.

verbose  If verbose = TRUE, print progress of algorithm to console.

Value
An object with S3 class c("varbvs", "list").

family  Either "gaussian" or "binomial".

n  Number of data samples used to fit model.

ncov  Number of covariates (columns of Z).

sigma  Settings for sigma (family = "gaussian" only).

sa  Settings for prior variance parameter.

logodds  Prior log-odds settings.

prior.same  TRUE if prior is identical for all variables. When logodds is a p x ns matrix, prior.same = FALSE.

sa0  Scale parameter for prior on hyperparameter sa.

n0  Degrees of freedom for prior on hyperparameter sa.

update.sigma  If TRUE, sigma was fit to data for each setting of prior logodds (family = "gaussian" only).

update.sa  If TRUE, sa was fit to data for each setting of prior logodds.

logw  An array with ns elements, in which logw[i] is the variational lower bound on the marginal log-likelihood for setting i of the hyperparameters.

alpha  Variational estimates of posterior inclusion probabilities for each hyperparameter setting.

mu  Variational estimates of posterior mean coefficients for each hyperparameter setting.

s  Variational estimates of posterior variances for each hyperparameter setting.
mu.cov  Posterior mean regression coefficients for covariates, including intercept, for each hyperparameter setting.

eta     Additional variational parameters for family = "binomial" only.

optimize.eta  If TRUE, eta was fit to data (family = "binomial" only).

pve     For each hyperparameter setting, and for each variable, mean estimate of the proportion of variance in outcome explained conditioned on variable being included in the model (family = "gaussian" only).

model.pve  Samples drawn from posterior distribution giving estimates of proportion of variance in outcome (y) explained by fitted variable selection model. This is for family = "gaussian" only. To obtain the posterior mean estimate of the proportion of variance explained (PVE), for example, simply type mean(fit$model.pve).

Regression models

Two types of outcomes (y) are modeled: (1) a continuous outcome, also a "quantitative trait" in the genetics literature; or (2) a binary outcome with possible values 0 and 1. For the former, set family = "gaussian", in which case, the outcome is i.i.d. normal with mean $u_0 + Z * u + X * b$ and variance sigma, in which u and b are vectors of regression coefficients, and $u_0$ is the intercept. In the second case, we use logistic regression to model the outcome, in which the probability that $y = 1$ is equal to $sigmoid(u_0 + Z * u + X * b)$. See help(sigmoid) for a description of the sigmoid function. Note that the regression always includes an intercept term ($u_0$).

Co-ordinate ascent optimization procedure

For both regression models, the fitting procedure consists of an inner loop and an outer loop. The outer loop iterates over each of the hyperparameter settings (sa, sigma and logodds). Given a setting of the hyperparameters, the inner loop cycles through coordinate ascent updates to tighten the lower bound on the marginal likelihood, $Pr(Y|X, sigma, sa, logodds)$. The inner loop coordinate ascent updates terminate when either (1) the maximum number of inner loop iterations is reached, as specified by options.maxiter, or (2) the maximum difference between the estimated posterior inclusion probabilities is less than tol.

To provide a more accurate variational approximation of the posterior distribution, by default the fitting procedure has two stages. In the first stage, the entire fitting procedure is run to completion, and the variational parameters (alpha, mu, s, eta) corresponding to the maximum lower bound are then used to initialize the coordinate ascent updates in a second stage. Although this has the effect of doubling the computation time (in the worst case), the final posterior estimates tend to be more accurate with this two-stage fitting procedure.

Variational approximation

Outputs alpha, mu and s specify the approximate posterior distribution of the regression coefficients. Each of these outputs is a p x ns matrix. For the ith hyperparameter setting, alpha[,i] is the variational estimate of the posterior inclusion probability (PIP) for each variable; mu[,i] is the variational estimate of the posterior mean coefficient given that it is included in the model; and s[,i] is the estimated posterior variance of the coefficient given that it is included in the model. These are also the quantities that are optimized as part of the inner loop coordinate ascent updates. An additional free parameter, eta, is needed for fast computation with the logistic regression model (family = "binomial"). The fitted value of eta is returned as an n x ns matrix.
Note that special care is required for interpreting the results of the variational approximation with the logistic regression model. In particular, interpretation of the individual estimates of the regression coefficients (e.g., the posterior mean estimates \( \text{fit}_\mu \)) is not straightforward due to the additional approximation introduced on the individual nonlinear factors in the likelihood. As a general guideline, only the relative magnitudes of the coefficients are meaningful.

**Averaging over hyperparameter settings**

In many settings, it is good practice to account for uncertainty in the hyperparameters when reporting final posterior quantities. For example, hyperparameter \( \sigma_a \) is often estimated with a high degree of uncertainty when only a few variables are included in the model. Provided that (1) the hyperparameter settings \( \sigma, \sigma_a \) and \( \text{logodds} \) adequately represent the space of possible hyperparameter settings with high posterior mass, (2) the hyperparameter settings are drawn from the same distribution as the prior, and (3) the fully-factorized variational approximation closely approximates the true posterior distribution, then final posterior quantities can be calculated by using \( \log w \) as (unnormalized) log-marginal probabilities.

Even when conditions (1), (2) and/or (3) are not satisfied, this can approach can still often yield reasonable estimates of averaged posterior quantities. The examples below demonstrate how final posterior quantities are reported by function `summary.varbvs` (see `help(summary.varbvs)` for more details). To account for discrepancies between the prior on \( (\sigma, \sigma_a, \text{logodds}) \) and the sampling density used to draw candidate settings of the hyperparameters, adjust the log-probabilities by setting \( \log \text{p} \Leftrightarrow \log \text{p} + \log \text{p} / \log \text{q} \), where \( \log \text{p} \) is the log-density of the prior distribution, and \( \log \text{q} \) is the log-density of the sampling distribution. (This is importance sampling; see, for example, R. M. Neal, Annealed importance sampling, *Statistics and Computing*, 2001.)

**Prior on proportion of variance explained**

Specifying the prior variance of the regression coefficients \( \sigma_a \) can be difficult, which is why we have included the option of fitting this hyperparameter to the data (see input argument `update\_sa` above). However, in many settings, especially when a small number of variables are included in the regression model, it is preferable to average over candidate settings of \( \sigma_a \) instead of fitting \( \sigma_a \) to the data. To choose a set of candidate settings for \( \sigma_a \), we have advocated for setting \( \sigma_a \) indirectly through a prior estimate of the proportion of variance in the outcome explained by the variables (abbreviated as PVE), since it is often more natural to specify the PVE rather than the prior variance (see references below). This is technically only suitable for the linear regression model (family = "gaussian"), but could potentially be used for the linear regression model in an approximate way.

For example, one could approximate a uniform prior on the PVE by drawing the PVE uniformly between 0 and 1, additionally specifying candidate settings for the prior log-odds, then computing the prior variance \( \sigma_a \) as follows:

```r
sx <- sum(var1.cols(X))
\sigma_a <- \text{PVE}/(1-\text{PVE})/\text{sigmoid}(\log(10)*\text{logodds})*sx)
```

Note that this calculation will yield \( \sigma_a = 0 \) when \( \text{PVE} = 0 \), and \( \sigma_a = \text{Inf} \) when \( \text{PVE} = 1 \).

Also, bear in mind that if there are additional covariates \( (Z) \) included in the linear regression model that explain variance in \( Y \), then it will usually make more sense to first remove the linear effects of these covariates before performing this calculation. The PVE would then represent the prior
The proportion of variance in the residuals of Y that are explained by the candidate variables. For an example of how to do this, see the code for function `varbvs`, in file `varbvs.R`, under "preprocessing steps". Alternatively, one could include the matrix Z in the calculation above, taking care to ensure that the covariates are included in the model with probability 1.

**Memory requirements**

Finally, we point out that the optimization procedures were carefully designed so that they can be applied to very large data sets; to date, this code has been tested on data sets with >500,000 variables and >10,000 samples. An important limiting factor is the ability to store the data matrix X in memory. To reduce memory requirements, in the MATLAB interface we require that X be single precision, but this option is not available in R. Additionally, we mostly avoid generating intermediate products that are of the same size as X. Only one such intermediate product is generated when `family = "gaussian"`, and none for `family = "binomial"`.

**Author(s)**

Peter Carbonetto <peter.carbonetto@gmail.com>

**References**


**See Also**

`summary.varbvs`, `varbvscoefcred`, `varbvspve`, `varbvsnorm`, `varbvsbin`, `varbvsbinz`, `normalizeLogweights`, `varbvs-internal`

**Examples**

```r
# LINEAR REGRESSION EXAMPLE
# ---------------------------
# Data are 200 uncorrelated ("unlinked") single nucleotide polymorphisms
# (SNPs) with simulated genotypes, in which the first 20 of them have an
# effect on the outcome. Also generate data for 3 covariates.
maf <- 0.05 + 0.45*runif(200)
X <- (runif(400*200) < maf) + (runif(400*200) < maf)
X <- matrix(as.double(X),400,200,byrow = TRUE)
Z <- randn(400,3)

# Generate the ground-truth regression coefficients for the variables
# (X) and additional 3 covariates (Z). Adjust the QTL effects so that
# the variables (SNPs) explain 50 percent of the variance in the
# outcome.
```
Compute credible intervals for regression coefficients.
varbvscoefcred

Description

Compute Monte Carlo estimates of credible intervals for coefficients in the fitted variable selection model. This function is used by summary.varbvs to generate credible intervals for coefficients of top-ranked variables.

Usage

varbvscoefcred (fit, vars, cred.int = 0.95, nr = 1000)

Arguments

- **fit**: Output of function `varbvs`.
- **vars**: Vector of indices or names of variables. If not specified, credible intervals are computed for all variables.
- **cred.int**: Size of credible interval (number between 0 and 1).
- **nr**: Number of Monte Carlo samples to draw to estimate credible intervals. A more accurate estimate of the credible interval can be obtained by setting `nr` to a larger number, at the cost of increased computation time.

Details

Here, the credible interval \([a, b]\) is simply defined as \(a = \text{quantile}(x, 0.5 - \text{cred.int}/2)\) and \(b = \text{quantile}(x, 0.5 + \text{cred.int}/2)\), in which \(x\) is a vector of samples drawn from the posterior distribution.

Value

- **a**: Credible interval lower bounds.
- **b**: Credible interval upper bounds.

Author(s)

Peter Carbonetto <peter.carbonetto@gmail.com>

References


See Also

- `varbvs`, `summary.varbvs`
varbvsindep  Compute posterior statistics, ignoring correlations.

Description

Compute the mean and variance of the coefficients, and the posterior inclusion probabilities (PIPs), ignoring correlations between variables. This is useful for inspecting or visualizing groups of correlated variables (e.g., genetic markers in linkage disequilibrium).

Usage

`varbvsindep (fit, X, Z, y)`

Arguments

- `fit`  Output of function `varbvs`.
- `X`  n x p input matrix, where n is the number of samples, and p is the number of variables. X cannot be sparse, and cannot have any missing values (NA).
- `Z`  n x m covariate data matrix, where m is the number of covariates. Do not supply an intercept as a covariate (i.e., a column of ones), because an intercept is automatically included in the regression model. For no covariates, set `Z = NULL`.
- `y`  Vector of length n containing observations of binary (`family = "binomial"`) or continuous (`family = "gaussian"`) outcome. For a binary outcome, all entries of y must be 0 or 1.

Details

For the ith hyperparameter setting, $\alpha[i]$ is the variational estimate of the posterior inclusion probability (PIP) for each variable; $\mu[i]$ is the variational estimate of the posterior mean coefficient given that it is included in the model; and $s[i]$ is the estimated posterior variance of the coefficient given that it is included in the model.

Value

- `alpha`  Variational estimates of posterior inclusion probabilities for each hyperparameter setting.
- `mu`  Variational estimates of posterior mean coefficients for each hyperparameter setting.
- `s`  Variational estimates of posterior variances for each hyperparameter setting.

Author(s)

Peter Carbonetto <peter.carbonetto@gmail.com>
References


See Also

varbvs
Index

*Topic **datasets**
  cytokine, 5
  leukemia, 5
bayesfactor, 3
cred, 4, 11
cytokine, 5
leukemia, 5
normalizelogweights, 6
plot.varbvs, 7
predict.varbvs, 9
print.summary.varbvs
  (summary.varbvs, print.summary.varbvs), 11
rand(rand, randn), 10
rand, randn, 10
randn(rand, randn), 10
summary.varbvs, 16
summary.varbvs
  (summary.varbvs, print.summary.varbvs), 11
summary.varbvs, print.summary.varbvs, 11
varbvs, 2, 7, 9, 11, 12, 19
varbvs-package, 2
varbvscoeffcred, 18
varbvsindep, 7, 20