Package ‘varbvs’

March 7, 2019

Encoding UTF-8

Type Package

Version 2.5-16

Date 2019-03-07

Title Large-Scale Bayesian Variable Selection Using Variational Methods

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, Matrix, stats, graphics, lattice, latticeExtra, Rcpp, nor1mix

Suggests curl, glmnet, qtl, knitr, rmarkdown, testthat

License GPL (>= 3)

NeedsCompilation yes

LazyData true

URL http://github.com/pcarbo/varbvs

BugReports http://github.com/pcarbo/varbvs/issues

LinkingTo Rcpp

VignetteBuilder knitr

Author Peter Carbonetto [aut, cre],
Matthew Stephens [aut],
David Gerard [ctb]

Repository CRAN

Date/Publication 2019-03-07 21:10:03 UTC

R topics documented:

`varbvs-package` *Large-scale Bayesian variable selection using variational methods*

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 genealized linear models.

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

Author(s)

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

References

P. Carbonetto and M. Stephens (2012). Scalable variational inference for Bayesian variable selection in regression, and its accuracy in genetic association studies. *Bayesian Analysis* **7**, 73–108.

`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

P. Carbonetto and M. Stephens (2012). Scalable variational inference for Bayesian variable selection in regression, and its accuracy in genetic association studies. *Bayesian Analysis* **7**, 73–108.

Examples

```
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

```
data(cytokine)
```

Format

`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

P. Carbonetto and M. Stephens (2013). Integrated enrichment analysis of variants and pathways in genome-wide association studies indicates central role for IL-2 signaling genes in type 1 diabetes, and cytokine signaling genes in Crohn's disease. *PLoS Genetics* **9**, e1003770.

Examples

```
# 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 lymphobastic leukemia (ALL) or acute myeloid leukemia (AML).

Usage

```
data(leukemia)
```

Format

Data are represented as a 72 x 3,571 matrix `x` 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

M. Dettling (2004). BagBoosting for tumor classification with gene expression data. *Bioinformatics* **20**, 3583–3593.

J. Friedman, T. Hastie and R. Tibshirani (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software* **33**, 1–22.

T. R. Golub, et al. (1999). Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science* **286**, 531–537.

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 probability is 1.

Value

Normalized probabilities such that the sum is equal to 1.

Author(s)

Peter Carbonetto <peter.carbonetto@gmail.com>

References

P. Carbonetto and M. Stephens (2012). Scalable variational inference for Bayesian variable selection in regression, and its accuracy in genetic association studies. *Bayesian Analysis* **7**, 73–108.

Examples

```
logw <- rnorm(6)
w    <- normalizelogweights(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

```
  ## S3 method for class 'varbvs'
plot(x, score, groups, vars = NULL, var.labels,
  draw.threshold = NA, gap = 0,col = "midnightblue", pch = 20,
  scales = NULL, xlab = "variable", ylab = "posterior probability",
  main = "fitted varbvs model: variable selection results",
  abline.args = list(lty = "dotted",col = "orangered"),
  vars.xyplot.args = list(pch = 20,col = "magenta"),
  vars.ltext.args = list(col = "black",pos = 4,cex = 0.5),
  par.settings = list(par.main.text = list(font = 1,cex = 0.8),
                          layout.heights = list(axis.top = 0, axis.bottom = 0)),
     ...)
```

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 `varbvsindep`.
`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.
`vars`	Indices (type `integer`) or names (type `character`) 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.
`scales`	Argument passed to `xyplot` specifying how x- and y-axes are drawn.
`xlab`	Argument passed to `xyplot` specifying horizontal axis title.
`ylab`	Argument passed to `xyplot` specifying vertical axis title.
`main`	Argument passed to `xyplot` specifying main plot 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.
`par.settings`	Argument passed to `xyplot` specifying additional adjustments to the plot settings.
`...`	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

P. Carbonetto and M. Stephens (2012). Scalable variational inference for Bayesian variable selection in regression, and its accuracy in genetic association studies. *Bayesian Analysis* **7**, 73–108.

See Also

xyplot, ltext, panel.abline, 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

```
  ## S3 method for class 'varbvs'
predict(object, X, Z = NULL,
                           type = c("link","response","class"),
                           averaged = TRUE, ...)
```

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.
type	Type of prediction to output. The default, "link", gives the linear predictors for family = "binomial", and gives the fitted values for family = "gaussian". For logistic regression (family = "binomial"), there are two alternative predictions: "response" givees the fitted probabilities, and "class" produces the maximum-probability outcome (0 or 1).

`averaged`	When `averaged = TRUE`, the predictions are computed by averaging over the hyperparameter settings, treating `object$logw` as (unnormalized) log-marginal probabilities. (See `varbvs` for more details about averaging.) When `averaged = FALSE`, the predictions are returned as a matrix when one row for each data sample, and one column for each hyperparameter setting.
`...`	Other arguments to generic predict function. These extra arguments are not used here.

Details

Note that the classification probabilities $Pr(Y = 1|X, Z, \theta)$ are not guaranteed to be calibrated under the variational approximation.

Value

When `averaged = TRUE`, the output is a vector containing the predicted outcomes for all samples. For `family = "binomial"`, all vector entries are 0 or 1.

When `averaged = FALSE`, the return value is a matrix with one row for each sample, and one column for each hyperparameter setting.

Author(s)

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

References

P. Carbonetto and M. Stephens (2012). Scalable variational inference for Bayesian variable selection in regression, and its accuracy in genetic association studies. *Bayesian Analysis* **7**, 73–108.

See Also

`varbvs`, `summary.varbvs`

Examples

```
# See help(varbvs) for examples.
```

`rand,randn` *Return matrices of pseudorandom values.*

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

Value

An m x n numeric matrix.

Author(s)

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

References

P. Carbonetto and M. Stephens (2012). Scalable variational inference for Bayesian variable selection in regression, and its accuracy in genetic association studies. *Bayesian Analysis* **7**, 73–108.

Examples

```
x <- rand(10,5)
y <- randn(10,5)
```

`subset.varbvs` *Select hyperparameter settings from varbvs analysis.*

Description

Select a subset of the candidate hyperparameter settings, and return a new varbvs analysis object with these hyperparameter settings only.

Usage

```
  ## S3 method for class 'varbvs'
subset(x, subset, ...)
```

Arguments

`x`	Output of function `varbvs`.
`subset`	Expression indicating hyperparameter settings to select. This expression should include one or more of `logodds`, `sigma` and `sa`.
`...`	Other arguments to generic subset function. These extra arguments are not used here.

Value

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

Author(s)

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

References

P. Carbonetto and M. Stephens (2012). Scalable variational inference for Bayesian variable selection in regression, and its accuracy in genetic association studies. *Bayesian Analysis* **7**, 73–108.

See Also

`varbvs`

Examples

```
# First run one of the examples in help(varbvs), then try running
# this.
#
#   fit.new <- subset(fit,logodds < (-2))
#
```

`summary.varbvs,print.summary.varbvs`
Summarize a fitted variable selection model.

Description

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

Usage

```
  ## S3 method for class 'varbvs'
summary(object, cred.int = 0.95, nv, pip.cutoff, ...)
  ## S3 method for class 'summary.varbvs'
print(x, digits = 3, ...)
  ## S3 method for class 'varbvs'
print(x, digits = 3, ...)
```

Arguments

`object`	Output of function `varbvs`.
`cred.int`	Size of credible interval, a number between 0 and 1.
`nv`	Show detailed statistics for top nv variables, ranked according to their posterior inclusion probabilities. Only one of `nv` and `pip.cutoff` may be specified. If neither are specified, the default is `nv = 5`.
`pip.cutoff`	Show detailed statistics for all variables in which the posterior inclusion probability (PIP) is at least `pip.cutoff`. Only one of `nv` and `pip.cutoff` may be specified.
`x`	Output of function `summary.varbvs`.
`digits`	Number of digits shown when printing posterior probabilities of top nv variables.
`...`	Additional summary or 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"`) when no additional covariates (`Z`) are included, it reports the estimated proportion of variance in the outcome explained by the model (PVE), and the credible interval of the PVE estimate brackets.

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 `objectlogw` 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).

Value

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

Author(s)

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

References

P. Carbonetto and M. Stephens (2012). Scalable variational inference for Bayesian variable selection in regression, and its accuracy in genetic association studies. *Bayesian Analysis* **7**, 73–108.

See Also

`varbvs`, varbvs.properties

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, weights, resid.vcov, alpha, mu, eta, update.sigma,
       update.sa, optimize.eta, initialize.params, update.order,
       nr = 100, sa0 = 1, n0 = 10, 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 = NULL`. The covariates are assigned an improper, uniform prior. Although improper priors are generally not advisable because they can result in improper posteriors and Bayes factors, this choice allows us to easily integrate out these covariates.
`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). Scaling the variance of the coefficients in this way is necessary to ensure that this prior is invariant to measurement scale (e.g., switching from grams to kilograms). 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 `n0 > 0` and `sa0 >     0`.
`logodds`	Hyperparameter logodds is the prior log-odds that a variable is included in the regression model; it is defined as $logodds = log10(q/(1-q))$, 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. (To obtain the prior probability from the log-odds, use the following formula: $q = 1/(1 + 10^(- logodds))$. The prior log-odds may also be specified separately for each variable, which is useful is 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.
`weights`	Optional vector of weights for weighted linear regression; larger weights correspond to smaller variances. It should be `NULL`, or a numeric vector of the same length as `y`. This is equivalent to the `weights` argument in `lm`. This option is only available for linear regression (`family = "gaussian"`), and cannot be combined with `resid.vcov`.
`resid.vcov`	Optional matrix specifying the covariance of the residual, scaled by sigma. This is useful to allow for observations with different variances (similar to `weights`), or for observations that are not independent. It should either be `NULL`, or a symmetric positive definite matrix in which the number of rows and columns is equal to the length of `y`. This option is only available for linear regression (`family = "gaussian"`), and cannot be combined with `weights`.
`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 FALSE, the initialization stage of the variational inference algorithm (see below) will be skipped, which saves computation time for large data sets.
`update.order`	Order of the co-ordinate ascent updates for fitting the variational approximation. The default is `update.order = 1:p`, where `p` is the number of variables (the number of columns of `X`).
`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. Large settings of `n0` provide greater stability of the parameter estimates for cases when the model is "sparse"; that is, when few variables are included in the model.
`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".
`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. These provide approximate values of the marginal log-likelihood for each hyperparameter setting.
`w`	Normalized weights (posterior probabilities) for each of the hyperparameter settings computed from `logw` using `normalizelogweights`.
`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.
`pip`	"Averaged" posterior inclusion probabilities computed as a weighted average of the individual PIPs (`alpha`), with weights given by `w`.
`beta`	"Averaged" posterior mean regression coefficients.
`mu.cov`	Posterior mean regression coefficients for covariates, including intercept, for each hyperparameter setting.
`beta.cov`	"Averaged" posterior mean regression coefficients for covariates, including intercept.
`eta`	Additional variational parameters for `family = "binomial"` only.
`optimize.eta`	If TRUE, eta was fit to data (`family = "binomial"` only).
`fitted.values`	Matrix containing the predicted (or "fitted") values of the outcome at each hyperparameter setting. For the logistic regression model (`family = "binomial"`), each matrix entry gives the probability that the binary outcome is equal to 1.
`residuals`	For linear regression, this is a matrix containing the model residuals at each hyperparameter setting. For `family = "binomial"`, this is a list with two matrices of the same size; `residuals$deviance` contains the deviance residuals based on the likelihood ratio chi-squared statistic; `residuals$response` contains the "response" residuals simply defined as the difference between the actual binary values (0 or 1) and the fitted binomial probabilities (*i.e.*, `fitted.values`).
`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 $u0 + Z * u + X * b$ and variance sigma, in which u and b are vectors of regresion coefficients, and u0 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 $sigmoid(u0 + Z * u + X * b)$. See `help(sigmoid)` for a description of the sigmoid function. Note that the regression always includes an intercept term (u0).

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 `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.

The variational estimates should be interpreted carefully, especially when variables are strongly correlated. For example, consider the simple scenario in which 2 candidate variables are closely correlated, and at least one of them explains the outcome with probability close to 1. Under the correct posterior distribution, we would expect that each variable is included with probability ~0.5. However, the variational approximation, due to the conditional independence assumption, will typically get this wrong, and concentrate most of the posterior weight on one variable (the actual variable that is chosen will depend on the starting conditions of the optimization). Although the individual PIPs are incorrect, a statistic summarizing the variable selection for both correlated variables (e.g., the total number of variables included in the model) should be reasonably accurate.

More generally, if variables can be reasonably grouped together based on their correlations, we recommend interpreting the variable selection results at a group level. For example, in genome-wide association studies (see the vignettes) ,a SNP with a high PIP indicates that this SNP is probably associated with the trait, and one or more nearby SNPs within a chromosomal region, or "locus," may be associated as well. Therefore, we interpreted the GWAS variable selection results at the level of loci, rather than at the level of individual SNPs.

Also 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 `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 sa 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, sa and 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 logw 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,sa,logodds) and the sampling density used to draw candidate settings of the hyperparameters, adjust the log-probabilities by setting `fit$logw <- fit$logw + logp/logq`, where logp is the log-density of the prior distribution, and logq 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 (sa) 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 preferrable to average over candidate settings of sa instead of fitting sa to the data. To choose a set of candidate settings for sa, we have advocated for setting sa 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 or the linear regression model (`family = "gaussian"`), but could potentially be used for the logistic 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 (sa) as follows:

```
sx <- sum(var1.cols(X))
sa <- PVE/(1-PVE)/(sigmoid(log(10)*logodds)*sx)
```

Note that this calculation will yield `sa = 0` when `PVE =   0`, and `sa = Inf` when `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 these calculations. The PVE would then represent the prior proportion of variance in the residuals of Y that are explained by the candidate variables. Alternatively, one could include the matrix Z in the calculations 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

P. Carbonetto and M. Stephens (2012). Scalable variational inference for Bayesian variable selection in regression, and its accuracy in genetic association studies. *Bayesian Analysis* **7**, 73–108.

Y. Guan and M. Stephens (2011). Bayesian variable selection regression for genome-wide association studies and other large-scale problems. *Annals of Applied Statistics* **5**, 1780–1815.

X. Zhou, P. Carbonetto and M. Stephens (2013). Polygenic modeling with Bayesian sparse linear mixed models. *PLoS Genetics* **9**, e1003264.

See Also

`summary.varbvs`, varbvs.properties, `varbvspve`, `varbvsnorm`, `varbvsbin`, `varbvsbinz`, `normalizelogweights`, varbvs-internal

Examples

```
# 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.
u    <- c(-1,2,1)
beta <- c(rnorm(20),rep(0,180))
beta <- 1/sd(c(X %*% beta)) * beta

# Generate the quantitative trait measurements.
y <- c(-2 + Z %*% u + X %*% beta + rnorm(400))

# Fit the variable selection model.
fit <- varbvs(X,Z,y,logodds = seq(-3,-1,0.1))
print(summary(fit))

# Compute the posterior mean estimate of hyperparameter sa.
sa <- with(fit,sum(sa * w))

# Compare estimated outcomes against observed outcomes.
y.fit <- predict(fit,X,Z)
print(cor(y,y.fit))
```

```
# LOGISTIC REGRESSION EXAMPLE
# ---------------------------
# Data are 100 uncorrelated ("unlinked") single nucleotide polymorphisms
# (SNPs) with simulated genotypes, in which the first 10 of them have an
# effect on the outcome. Also generate data for 2 covariates.
maf <- 0.05 + 0.45*runif(100)
X   <- (runif(750*100) < maf) + (runif(750*100) < maf)
X   <- matrix(as.double(X),750,100,byrow = TRUE)
Z   <- randn(750,2)

# Generate the ground-truth regression coefficients for the variables
# (X) and additional 2 covariates (Z).
u    <- c(-1,1)
beta <- c(0.5*rnorm(10),rep(0,90))

# Simulate the binary trait (case-control status) as a coin toss with
# success rates given by the logistic regression.
sigmoid <- function (x)
  1/(1 + exp(-x))
y <- as.double(runif(750) < sigmoid(-1 + Z %*% u + X %*% beta))

# Fit the variable selection model.
fit <- varbvs(X,Z,y,"binomial",logodds = seq(-2,-0.5,0.5))
print(summary(fit))
```

---

varbvs.properties *Accessing Properties of Fitted varbvs Models*

Description

All these functions are methods for class "varbvs" objects.

Usage

```
## S3 method for class 'varbvs'
nobs(object, ...)
## S3 method for class 'varbvs'
case.names(object, ...)
## S3 method for class 'varbvs'
variable.names(object, full = FALSE,
                                    include.threshold = 0.01, ...)
## S3 method for class 'varbvs'
labels(object, ...)
## S3 method for class 'varbvs'
coef(object, ...)
## S3 method for class 'varbvs'
confint(object, parm, level = 0.95, ...)
## S3 method for class 'varbvs'
```

```
fitted(object, ...)
## S3 method for class 'varbvs'
resid(object, type = c("deviance","response"), ...)
## S3 method for class 'varbvs'
residuals(object, type = c("deviance","response"), ...)
## S3 method for class 'varbvs'
deviance(object, ...)
```

Arguments

`object`	An object inheriting from class `varbvs`, usually the result of calling function `varbvs`.
`full`	logical; if `TRUE`, names of all variables (columns of `X`) are returned, including variables that have zero probability of being included in the regression model.
`include.threshold`	When `full = FALSE`, names of all variables (columns of `X`) with "averaged" posterior inclusion probability greater than `include.threshold` are returned.
`parm`	Confidence intervals are computed for these selected variables. These may either be specified as numbers (column indices of `varbvs` input matrix `X`) or names (column names of `X`). If not specified, confidence intervals will be computed for the top 5 variables by posterior inclusion probability. Confidence intervals are not provided for covariates (columns of `Z`); see below for details.
`level`	Size of confidence level.
`type`	Type of residuals to be returned. This argument is only relevant for logistic regression models (`family = "binomial"`). See `varbvs` for more details about the two available types of residuals for logistic regression.
`...`	Further arguments passed to or from other methods.

Details

The generic accessor functions `nobs`, `case.names`, `variable.names` and `labels` can be used to extract various useful properties of the fitted `varbvs` model. Method `labels`, in particular, returns the names of the candidate variables (columns of `X`) which may be used, for example, to plot posterior inclusion probabilities or effect estimates.

`coef` returns a matrix containing the posterior estimates of the regression coefficients at each hyperparameter setting, as well as an additional column containing "averaged" coefficient estimates.

`confint` returns confidence intervals (also, equivalently in this case, "credible intervals") for all selected variables `parm`. These are *conditional* confidence intervals; that is, conditioned on each variable being included in the regression model.

The `confint` return value is different from the usual confidence interval (e.g., for an `lm` result) because a confidence interval is provided for each hyperparameter setting, as well as an additional "averaged" confidence interval. The confidence intervals are returned a list, with one list element per selected variable, and each list element is a matrix with columns giving lower and upper confidence limits for each hyperparameter setting, as well as the averaged limits.

Note that confidence intervals cannot currently be requested for covariates (columns of `Z`).

`fitted` returns a matrix containing the predicted (or "fitted") values of the outcome at each hyperparameter setting. For the logistic regression model (`family = "binomial"`), each matrix entry gives the probability that the binary outcome is equal to 1.

Likewise, `resid` and `residuals` each return a matrix containing the model residuals at each hyperparameter setting.

`deviance` returns the deviance for the fitted model at each hyperparameter setting.

See Also

`varbvs`, `nobs`, `case.names`, `variable.names`, `labels`, `coef`, `coef`, `fitted`, `residuals`, `deviance`

`varbvsbf` *Compute numerical estimate of Bayes factor.*

Description

The Bayes factor is the ratio of the marginal likelihoods under two different models (see Kass & Raftery, 1995). Function `varbvsbf` provides a convenient interface for computing the Bayes factor comparing the fit of two different `varbvs` models.

Usage

```
varbvsbf (fit0, fit1)

bayesfactor (logw0, logw1)
```

Arguments

`fit0`	An output returned from `varbvs`.
`fit1`	Another output returned from `varbvs`.
`logw0`	log-probabilities or log-importance weights under H0.
`logw1`	log-probabilities or log-importance weights under H1.

Details

Computes numerical estimate of

$$BF = Pr(data|H1)/Pr(data|H0),$$

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

$Pr(data, Z0|H0) = Pr(data|Z0, H0)xPr(Z0|H0), Pr(data, Z1|H1) = Pr(data|Z1, H1)xPr(Z1|H1),$

where Pr(Z0 | H0) and Pr(Z1 | 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 A1 are drawn from the same (uniform) distribution as the prior, Pr(Z0 | H0) and Pr(Z1 | H1), respectively.

Value

The estimated Bayes factor.

Author(s)

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

References

P. Carbonetto and M. Stephens (2012). Scalable variational inference for Bayesian variable selection in regression, and its accuracy in genetic association studies. *Bayesian Analysis* **7**, 73–108.

R. E. Kass and A. E. Raftery (1995). Bayes Factors. *Journal of the American Statistical Association* **90**, 773–795.

R. M. Neal (2001). Annealed importance sampling. *Statistics and Computing* **11**, 125–139.

See Also

`varbvs`, `normalizelogweights`

`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

P. Carbonetto and M. Stephens (2012). Scalable variational inference for Bayesian variable selection in regression, and its accuracy in genetic association studies. *Bayesian Analysis* **7**, 73–108.

See Also

`varbvs`

`varbvsmix`	*Fit linear regression with mixture-of-normals priors using variational approximation methods.*

Description

Find the "best" fully-factorized approximation to the posterior distribution of the coefficients, with linear regression likelihood and mixture-of-normals 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. In the original formulation (see `varbvs`), each regression coefficient was drawn identically from a spike-and-slab prior. Here, we instead formulate the "slab" as a mixture of normals.

Usage

```
varbvsmix(X, Z, y, sa, sigma, w, alpha, mu, update.sigma, update.sa,
          update.w, w.penalty, drop.threshold = 1e-8, 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 = NULL`.
`y`	Vector of length n containing values of the continuous outcome.
`sa`	Vector specifying the prior variance of the regression coefficients (scaled by `sigma`) for each mixture component. The variance of the first mixture component is the "spike", and therefore should be exactly zero.
`sigma`	Residual variance parameter. If missing, it is automatically fitted to the data by computing an approximate maximum-likelihood estimate.
`w`	If missing, it is automatically fitted to the data by computing an approximate maximum-likelihood estimate.
`alpha`	Initial estimates of the approximate posterior mixture assignment probabilities. These should be specified as a p x K matrix, where K is the number of mixture components. Each row must add up to 1.
`mu`	Initial estimates of the approximate regression coefficients conditioned on being drawn from each of the K mixture components. These estimates should be provided as a p x K matrix, where K is the number of mixture components.
`update.sigma`	If `TRUE`, sigma is fitted to data using an approximate EM algorithm, in which case argument `sigma`, if provided, is the initial estimate.
`update.sa`	Currently, estimate of mixture component variances is not implemented, so this must be set to `TRUE`, otherwise an error will be generated.

`update.w`	If `TRUE`, mixture weights are fitted using an approximate EM algorithm, in which case argument `w`, if provided, is the initial estimate.
`w.penalty`	Penalty term for the mixture weights. It is useful for "regularizing" the estimate of `w` when we do not have a lot of information. It should be a vector with one positive entry for each mixture component. Larger values place more weight on the corresponding mixture components. It is based on the Dirichlet distribution with parameters `w.penalty`. The default is a vector of ones, which reduces to a uniform prior on `w`.
`drop.threshold`	Posterior probability threshold for dropping mixture components. Should be a positive number close to zero. If, at any point during the optimization, all posterior mixture assignment probabilities for a given mixture component `k` are less than `drop.threshold`, the mixture weight for component `k` is automatically set to zero. Set `drop.threshold` to zero to disable this behaviour. Setting larger values for `drop.threshold` may improve computation speed at a small cost to numerical accuracy of the final results.
`tol`	Convergence tolerance for co-ordinate ascent updates.
`maxiter`	Maximum number of co-ordinate ascent iterations.
`verbose`	If `verbose = TRUE`, print progress of algorithm to console.

Details

See https://www.overleaf.com/8954189vvpqnwpxhvhq.

Value

An object with S3 class `c("varbvsmix","list")`.

`n`	Number of data samples used to fit model.
`mu.cov`	Posterior mean regression coefficients for covariates, including intercept.
`update.sigma`	If `TRUE`, residual variance parameter `sigma` was fit to data.
`update.sa`	If `TRUE`, mixture variances were fit to data.
`update.w`	If `TRUE`, mixture weights were fit to data.
`w.penalty`	Penalty used for updating mixture weights.
`drop.threshold`	Posterior probabiltiy threshold used in the optimization procedure for setting mixture weights to zero.
`sigma`	Fitted or user-specified residual variance parameter.
`sa`	User-specified mixture variances.
`w`	Fitted or user-specified mixture weights.
`alpha`	Variational estimates of posterior mixture assignent probabilities.
`mu`	Variational estimates of posterior mean coefficients.
`s`	Variational estimates of posterior variances.
`lfsr`	Local false sign rate (LFSR) for each variable computed from variational estimates of posterior assignment probabilities and posterior means and variances. See Stephens (2017) for a definition of the LFSR.

`logZ`	Variational lower bound to marginal log-likelihood at each iteration of the co-ordinate ascent algorithm.
`err`	Maximum difference in the variational posterior probabilities at each iteration of the co-ordinate ascent algorithm.
`nzw`	Number of nonzero mixture components (including the "spike") at each iteration of the co-ordinate ascent algorithm.

Author(s)

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

References

M. Stephens (2017). False discovery rates: a new deal. *Biostatistics* **18**, 275–294.

See Also

`varbvs`

Examples

```
# Generate the data set.
set.seed(1)
n    <- 200
p    <- 500
X    <- randn(n,p)
sd   <- c(0,0.2,0.5)
w    <- c(0.9,0.05,0.05)
k    <- sample(length(w),p,replace = TRUE,prob = w)
beta <- sd[k] * rnorm(p)
y    <- c(X %*% beta + rnorm(n))

# Fit the model to the data.
fit <- varbvsmix(X,NULL,y,sd^2)

## Not run:
library(lattice)
print(xyplot(beta.est ~ beta.true,
             data.frame(beta.true = beta,
                        beta.fitted = rowSums(fit$alpha * fit$mu)),
             pch = 20,col = "royalblue",cex = 1))

## End(Not run)
```

varbvsproxybf — *Compute Bayes factors measuring improvement-in-fit along 1 dimension.*

Description

For each candidate variable j, this function returns a Bayes factor measuring the improvement in fit when variable j is included in the model instead of variable i; that is, a larger Bayes factor indicates a better model fit by swapping variables i and j. From an optimization perspective, this could be viewed as addressing the following question: if you had to update the variational parameters for one variable so as to improve the "fit" of the variational approximation after setting the posterior inclusion probability for variable i to zero, which variable would you choose?

Usage

```
varbvsproxybf(X, Z, y, fit, i, vars)
```

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 = NULL`.
`y`	Vector of length n containing values of the continuous outcome.
`fit`	An object inheriting from class `varbvs`, usually the result of calling function `varbvs`. Currently, this is only implemented for linear regression (`family = "gaussian"`); any other choice will produce an error.
`i`	Variable against will. Typically, will be a variable included in the regression model with high probability, but not always.
`vars`	Set of candidate "proxy" variables. This set may include `i`, but not does not have to.

Value

`varbvsproxybf` returns a list with the following components:

`BF`	Matrix containing Bayes factors for each candidate proxy variable and for each hyperparameter setting.
`mu`	Matrix containing estimated posterior means for each candidate proxy variable and for each hyperparameter setting.
`s`	Matrix containing estimated posterior variances for each candidate proxy variance for each hyperparameter setting.

Author(s)

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

See Also

`varbvs`

Index