Package ‘BGLR’

October 8, 2024

Version 1.1.3

Title Bayesian Generalized Linear Regression

Depends R (>= 3.5.0)

Imports truncnorm, MASS, methods

Description Bayesian Generalized Linear Regression.

License GPL-3

NeedsCompilation yes

Author Gustavo de los Campos [aut],
Paulino Perez Rodriguez [aut, cre]

Maintainer Paulino Perez Rodriguez <perpdgo@colpos.mx>

Repository CRAN

Date/Publication 2024-10-08 03:00:02 UTC

Contents

`BFDR` *BFDR*

Description

Convert probabilities to Bayesian false discovery rates.

Usage

```
BFDR(prob)
```

Arguments

`prob`	(numeric), the vector of probabilities.

Value

A numeric vector of Bayesian false discovery rates.

BGLR *Bayesian Generalized Linear Regression*

Description

The BGLR ('Bayesian Generalized Linear Regression') function fits various types of parametric and semi-parametric Bayesian regressions to continuos (censored or not), binary and ordinal outcomes.

Usage

```
BGLR(y, response_type = "gaussian", a=NULL, b=NULL,ETA = NULL, nIter = 1500,
     burnIn = 500, thin = 5, saveAt = "", S0 = NULL,
     df0 =5, R2 = 0.5, weights = NULL,
     verbose = TRUE, rmExistingFiles = TRUE, groups=NULL)
```

Arguments

`y`	(numeric, n) the data-vector (NAs allowed).
`response_type`	(string) admits values `"gaussian"` or `"ordinal"`. The Gaussian outcome may be censored or not (see below). If `response_type="gaussian"`, `y` should be coercible to numeric. If `response_type="ordinal"`, `y` should be coercible to character, and the order of the outcomes is determined based on the alphanumeric order (`0<1<2..<a<b..`). For ordinal traits the probit link is used.
`a, b`	(numeric, n) only requiered for censored outcomes, `a` and `b` are vectors specifying lower and upper bounds for censored observations, respectively. The default value, for non-censored and ordinal outcomes, is `NULL` (see details).
`ETA`	(list) This is a two-level list used to specify the regression function (or linear predictor). By default the linear predictor (the conditional expectation function in case of Gaussian outcomes) includes only an intercept. Regression on covariates and other types of random effects are specified in this two-level list. For instance:

```
ETA=list(list(X=W, model="FIXED"),
              list(X=Z,model="BL"),
              list(K=G,model="RKHS")),
```

specifies that the linear predictor should include: an intercept (included by default) plus a linear regression on W with regression coefficients treated as fixed effects (i.e., flat prior), plus regression on Z, with regression coefficients modeled as in the Bayesian Lasso of Park and Casella (2008) plus and a random effect with co-variance structure G.

For linear regressions the following options are implemented: FIXED (Flat prior), BRR (Gaussian prior), BayesA (scaled-t prior), BL (Double-Exponential prior), BayesB (two component mixture prior with a point of mass at zero and a scaled-t slab), BayesC (two component mixture prior with a point of mass at zero and a Gaussian slab). In linear regressions X can be the incidence matrix for effects or a formula (e.g. `~factor(sex) + age`) in which case the incidence matrix is created internally using the `model.matrix` function of R. For Gaussian processes (RKHS) a co-variance matrix (K) must be provided. Further details about the models implemented in BGLR see the vignettes in the package.

`weights`	(numeric, n) a vector of weights, may be `NULL`. If weights is not `NULL`, the residual variance of each data-point is set to be proportional to the inverse of the squared-weight. Only used with Gaussian outcomes.
`nIter, burnIn, thin`	(integer) the number of iterations, burn-in and thinning.
`saveAt`	(string) this may include a path and a pre-fix that will be added to the name of the files that are saved as the program runs.
`S0, df0`	(numeric) The scale parameter for the scaled inverse-chi squared prior assigned to the residual variance, only used with Gaussian outcomes. In the parameterization of the scaled-inverse chi square in BGLR the expected values is `S0/(df0-2)`. The default value for the df parameter is 5. If the scale is not specified a value is calculated so that the prior mode of the residual variance equals `var(y)*R2` (see below). For further details see the vignettes in the package.
`R2`	(numeric, `0<R2<1`) The proportion of variance that one expects, a priori, to be explained by the regression. Only used if the hyper-parameters are not specified; if that is the case, internaly, hyper-paramters are set so that the prior modes are consistent with the variance partition specified by R2 and the prior distribution is relatively flat at the mode. For further details see the vignettes in the package.
`verbose`	(logical) if TRUE the iteration history is printed, default TRUE.
`rmExistingFiles`	(logical) if TRUE removes existing output files from previous runs, default TRUE.
`groups`	(factor) a vector of the same length of y that associates observations with groups, each group will have an associated variance component for the error term.

Details

BGLR implements a Gibbs sampler for a Bayesian regresion model. The linear predictor (or regression function) includes an intercept (introduced by default) plus a number of user-specified regression components (X) and random effects (u), that is:

$$\eta = 1\mu + X_1\beta_1 + ... + X_p\beta_p + u_1 + ... + u_q$$

The components of the linear predictor are specified in the argument ETA (see above). The user can specify as many linear terms as desired, and for each component the user can choose the prior density to be assigned. The distribution of the response is modeled as a function of the linear predictor.

For Gaussian outcomes, the linear predictor is the conditional expectation, and censoring is allowed. For censored data points the actual response value (y_i) is missing, and the entries of the vectors a and b (see above) give the lower an upper vound for y_i. The following table shows the configuration of the triplet (y, a, b) for un-censored, right-censored, left-censored and interval censored.

	a	y	b
Un-censored	NULL	y_i	NULL
Right censored	a_i	NA	∞
Left censored	$-\infty$	NA	b_i
Interval censored	a_i	NA	b_i

Internally, censoring is dealt with as a missing data problem.

Ordinal outcomes are modelled using the probit link, implemented via data augmentation. In this case the linear predictor becomes the mean of the underlying liability variable which is normal with mean equal to the linear predictor and variance equal to one. In case of only two classes (binary outcome) the threshold is set equal to zero, for more than two classess thresholds are estimated from the data. Further details about this approach can be found in Albert and Chib (1993).

Value

A list with estimated posterior means, estimated posterior standard deviations, and the parameters used to fit the model. See the vignettes in the package for further details.

Author(s)

Gustavo de los Campos, Paulino Perez Rodriguez,

References

Albert J,. S. Chib. 1993. Bayesian Analysis of Binary and Polychotomus Response Data. *JASA*, **88**: 669-679.

de los Campos G., H. Naya, D. Gianola, J. Crossa, A. Legarra, E. Manfredi, K. Weigel and J. Cotes. 2009. Predicting Quantitative Traits with Regression Models for Dense Molecular Markers and Pedigree. *Genetics* **182**: 375-385.

de los Campos, G., D. Gianola, G. J. M., Rosa, K. A., Weigel, and J. Crossa. 2010. Semi-parametric genomic-enabled prediction of genetic values using reproducing kernel Hilbert spaces methods. *Genetics Research*, **92**:295-308.

Park T. and G. Casella. 2008. The Bayesian LASSO. *Journal of the American Statistical Association* **103**: 681-686.

Spiegelhalter, D.J., N.G. Best, B.P. Carlin and A. van der Linde. 2002. Bayesian measures of model complexity and fit (with discussion). *Journal of the Royal Statistical Society*, Series B (Statistical Methodology) **64** (4): 583-639.

Examples

```
## Not run:
#Demos
library(BGLR)

#BayesA
demo(BA)

#BayesB
```

```
demo(BB)

#Bayesian LASSO
demo(BL)

#Bayesian Ridge Regression
demo(BRR)

#BayesCpi
demo(BayesCpi)

#RKHS
demo(RKHS)

#Binary traits
demo(Bernoulli)

#Ordinal traits
demo(ordinal)

#Censored traits
demo(censored)


## End(Not run)
```

BLR *Bayesian Linear Regression*

Description

The BLR ('Bayesian Linear Regression') function was designed to fit parametric regression models using different types of shrinkage methods. An earlier version of this program was presented in de los Campos *et al.* (2009).

Usage

```
BLR(y, XF, XR, XL, GF, prior, nIter, burnIn, thin,thin2,saveAt,
    minAbsBeta,weights)
```

Arguments

y	(numeric, n) the data-vector (NAs allowed).
XF	(numeric, $n \times pF$) incidence matrix for $\boldsymbol{\beta}_F$, may be NULL.
XR	(numeric, $n \times pR$) incidence matrix for $\boldsymbol{\beta}_R$, may be NULL.
XL	(numeric, $n \times pL$) incidence matrix for $\boldsymbol{\beta}_L$, may be NULL.

`GF`	(list) providing an \$ID (integer, n) linking observations to groups (e.g., lines or sires) and a (co)variance structure (\$A, numeric, $pU \times pU$) between effects of the grouping factor (e.g., line or sire effects). Note: ID must be an integer taking values from 1 to pU; ID[i]=q indicates that the ith observation in $\boldsymbol{y}$ belongs to cluster q whose (co)variance function is in the qth row (column) of $\boldsymbol{A}$. GF may be NULL.
`weights`	(numeric, n) a vector of weights, may be NULL.
`nIter, burnIn, thin`	(integer) the number of iterations, burn-in and thinning.
`saveAt`	(string) this may include a path and a pre-fix that will be added to the name of the files that are saved as the program runs.
`prior`	(list) containing the following elements, • prior\$varE, prior\$varBR, prior\$varU: (list) each providing degree of freedom (\$df) and scale (\$S). These are the parameters of the scaled inverse-χ^2 distributions assigned to variance components, see Eq. (2) below. In the parameterization used by BLR() the prior expectation of variance parameters is $S/(df-2)$. • prior\$lambda: (list) providing \$value (initial value for λ); \$type ('random' or 'fixed') this argument specifies whether λ should be kept fixed at the value provided by \$value or updated with samples from the posterior distribution; and, either \$shape and \$rate (this when a Gamma prior is desired on λ^2) or \$shape1, \$shape2 and \$max, in this case $p(\lambda\vert\max,\alpha_1,\alpha_2) \propto Beta\left(\frac{\lambda}{\max}\vert\alpha_1,\alpha_2\right)$. For detailed description of these priors see de los Campos *et al.* (2009).
`thin2`	This value controls wether the running means are saved to disk or not. If thin2 is greater than nIter the running means are not saved (default, thin2=1×10^{10}).
`minAbsBeta`	The minimum absolute value of the components of $\boldsymbol{\beta}_L$ to avoid numeric problems when sampling from $\boldsymbol{\tau}^2$, default 1×10^{-9}

Details

The program runs a Gibbs sampler for the Bayesian regression model described below.

Likelihood. The equation for the data is:

$$\boldsymbol{y} = \boldsymbol{1}\mu + \boldsymbol{X}_F\boldsymbol{\beta}_F + \boldsymbol{X}_R\boldsymbol{\beta}_R + \boldsymbol{X}_L\boldsymbol{\beta}_L + \boldsymbol{Z}\boldsymbol{u} + \boldsymbol{\varepsilon} \quad (1)$$

where $\boldsymbol{y}$, the response is a $n \times 1$ vector (NAs allowed); μ is an intercept; $\boldsymbol{X}_F, \boldsymbol{X}_R, \boldsymbol{X}_L$ and $\boldsymbol{Z}$ are incidence matrices used to accommodate different types of effects (see below), and; $\boldsymbol{\varepsilon}$ is a vector of model residuals assumed to be distributed as $\boldsymbol{\varepsilon} \sim N(\boldsymbol{0}, Diag(\sigma_{\boldsymbol{\varepsilon}}^2/w_i^2))$, here $\sigma_{\boldsymbol{\varepsilon}}^2$ is an (unknown) variance parameter and w_i are (known) weights that allow for heterogeneous-residual variances.

Any of the elements in the right-hand side of the linear predictor, except μ and $\boldsymbol{\varepsilon}$, can be omitted; by default the program runs an intercept model.

Prior. The residual variance is assigned a scaled inverse-χ^2 prior with degree of freedom and scale parameter provided by the user, that is, $\sigma_{\boldsymbol{\varepsilon}}^2 \sim \chi^{-2}(\sigma_{\boldsymbol{\varepsilon}}^2 \vert df_{\boldsymbol{\varepsilon}}, S_{\boldsymbol{\varepsilon}})$. The regression coefficients $\{\mu, \boldsymbol{\beta}_F, \boldsymbol{\beta}_R, \boldsymbol{\beta}_L, \boldsymbol{u}\}$ are assigned priors that yield different type of shrinkage. The intercept and the vector of regression coefficients $\boldsymbol{\beta}_F$ are assigned flat priors (i.e., estimates are not shrunk). The vector of regression coefficients $\boldsymbol{\beta}_R$ is assigned a Gaussian prior with variance common to all effects, that is, $\beta_{R,j} \overset{iid}{\sim} N(0, \sigma^2_{\boldsymbol{\beta}_R})$. This prior is the Bayesian counterpart of Ridge Regression. The variance parameter $\sigma^2_{\boldsymbol{\beta}_R}$, is treated as unknown and it is assigned a scaled inverse-χ^2 prior, that is, $\sigma^2_{\boldsymbol{\beta}_R} \sim \chi^{-2}(\sigma^2_{\boldsymbol{\beta}_R}|df_{\boldsymbol{\beta}_R}, S_{\boldsymbol{\beta}_R})$ with degrees of freedom $df_{\boldsymbol{\beta}_R}$, and scale $S_{\boldsymbol{\beta}_R}$ provided by the user.

The vector of regression coefficients $\boldsymbol{\beta}_L$ is treated as in the Bayesian LASSO of Park and Casella (2008). Specifically,

$$p(\boldsymbol{\beta}_L, \boldsymbol{\tau}^2, \lambda|\sigma^2_{\boldsymbol{\varepsilon}}) = \left\{\prod_k N(\beta_{L,k}|0, \sigma^2_{\boldsymbol{\varepsilon}}\tau^2_k) Exp\left(\tau^2_k|\lambda^2\right)\right\} p(\lambda),$$

where, $Exp(\cdot|\cdot)$ is an exponential prior and $p(\lambda)$ can either be: (a) a mass-point at some value (i.e., fixed λ); (b) $p(\lambda^2) \sim Gamma(r, \delta)$ this is the prior suggested by Park and Casella (2008); or, (c) $p(\lambda|\max, \alpha_1, \alpha_2) \propto Beta\left(\frac{\lambda}{\max}|\alpha_1, \alpha_2\right)$, see de los Campos *et al.* (2009) for details. It can be shown that the marginal prior of regression coefficients $\beta_{L,k}$, $\int N(\beta_{L,k}|0, \sigma^2_{\boldsymbol{\varepsilon}}\tau^2_k) Exp\left(\tau^2_k|\lambda^2\right) \partial\tau^2_k$, is Double-Exponential. This prior has thicker tails and higher peak of mass at zero than the Gaussian prior used for $\boldsymbol{\beta}_R$, inducing a different type of shrinkage.

The vector $\boldsymbol{u}$ is used to model the so called 'infinitesimal effects', and is assigned a prior $\boldsymbol{u} \sim N(\boldsymbol{0}, \boldsymbol{A}\sigma^2_{\boldsymbol{u}})$, where, $\boldsymbol{A}$ is a positive-definite matrix (usually a relationship matrix computed from a pedigree) and $\sigma^2_{\boldsymbol{u}}$ is an unknow variance, whose prior is $\sigma^2_{\boldsymbol{u}} \sim \chi^{-2}(\sigma^2_{\boldsymbol{u}}|df_{\boldsymbol{u}}, S_{\boldsymbol{u}})$.

Collecting the above mentioned assumptions, the posterior distribution of model unknowns, $\boldsymbol{\theta} = \left\{\mu, \boldsymbol{\beta}_F, \boldsymbol{\beta}_R, \sigma^2_{\boldsymbol{\beta}_R}, \boldsymbol{\beta}_L, \boldsymbol{\tau}^2, \lambda, \boldsymbol{u}, \sigma^2_{\boldsymbol{u}}, \sigma^2_{\boldsymbol{\varepsilon}},\right\}$, is,

$$\begin{aligned} p(\boldsymbol{\theta}|\boldsymbol{y}) \quad \propto \quad & N\left(\boldsymbol{y}|\boldsymbol{1}\mu + \boldsymbol{X}_F\boldsymbol{\beta}_F + \boldsymbol{X}_R\boldsymbol{\beta}_R + \boldsymbol{X}_L\boldsymbol{\beta}_L + \boldsymbol{Z}\boldsymbol{u}; Diag\left\{\frac{\sigma^2_\varepsilon}{w^2_i}\right\}\right) \\ & \times\left\{\prod_j N\left(\beta_{R,j}|0, \sigma^2_{\boldsymbol{\beta}_R}\right)\right\}\chi^{-2}\left(\sigma^2_{\boldsymbol{\beta}_R}|df_{\boldsymbol{\beta}_R}, S_{\boldsymbol{\beta}_R}\right) \\ & \times\left\{\prod_k N\left(\beta_{L,k}|0, \sigma^2_{\boldsymbol{\varepsilon}}\tau^2_k\right) Exp\left(\tau^2_k|\lambda^2\right)\right\} p(\lambda) \qquad (2) \\ & \times N(\boldsymbol{u}|\boldsymbol{0}, \boldsymbol{A}\sigma^2_{\boldsymbol{u}})\chi^{-2}(\sigma^2_{\boldsymbol{u}}|df_{\boldsymbol{u}}, S_{\boldsymbol{u}})\chi^{-2}(\sigma^2_{\boldsymbol{\varepsilon}}|df_{\boldsymbol{\varepsilon}}, S_{\boldsymbol{\varepsilon}}) \end{aligned}$$

Value

A list with posterior means, posterior standard deviations, and the parameters used to fit the model:

`$yHat`	the posterior mean of $\boldsymbol{1}\mu + \boldsymbol{X}_F\boldsymbol{\beta}_F + \boldsymbol{X}_R\boldsymbol{\beta}_R + \boldsymbol{X}_L\boldsymbol{\beta}_L + \boldsymbol{Z}\boldsymbol{u} + \varepsilon$.
`$SD.yHat`	the corresponding posterior standard deviation.
`$mu`	the posterior mean of the intercept.
`$varE`	the posterior mean of $\sigma^2_{\boldsymbol{\varepsilon}}$.
`$bR`	the posterior mean of $\boldsymbol{\beta}_R$.
`$SD.bR`	the corresponding posterior standard deviation.
`$varBr`	the posterior mean of $\sigma^2_{\boldsymbol{\beta}_R}$.
`$bL`	the posterior mean of $\boldsymbol{\beta}_L$.
`$SD.bL`	the corresponding posterior standard deviation.

`$tau2`	the posterior mean of $\boldsymbol{\tau}^2$.
`$lambda`	the posterior mean of λ.
`$u`	the posterior mean of $\boldsymbol{u}$.
`$SD.u`	the corresponding posterior standard deviation.
`$varU`	the posterior mean of $\sigma^2_{\boldsymbol{u}}$.
`$fit`	a list with evaluations of effective number of parameters and DIC (Spiegelhalter *et al.*, 2002).
`$whichNa`	a vector indicating which entries in $\boldsymbol{y}$ were missing.
`$prior`	a list containig the priors used during the analysis.
`$weights`	vector of weights.
`$fit`	list containing the following elements, • $logLikAtPostMean: log-likelihood evaluated at posterior mean. • $postMeanLogLik: the posterior mean of the Log-Likelihood. • $pD: estimated effective number of parameters, Spiegelhalter *et al.* (2002). • $DIC: the deviance information criterion, Spiegelhalter *et al.* (2002).
`$nIter`	the number of iterations made in the Gibbs sampler.
`$burnIn`	the nuber of iteratios used as burn-in.
`$thin`	the thin used.
`$y`	original data-vector.

The posterior means returned by BLR are calculated after burnIn is passed and at a thin as specified by the user.

Save. The routine will save samples of μ, variance components and λ and running means (rm*.dat). Running means are computed using the thinning specified by the user (see argument thin above); however these running means are saved at a thinning specified by argument thin2 (by default, thin2=1×10^{10} so that running means are computed as the sampler runs but not saved to the disc).

Author(s)

Gustavo de los Campos, Paulino Perez Rodriguez,

References

de los Campos G., H. Naya, D. Gianola, J. Crossa, A. Legarra, E. Manfredi, K. Weigel and J. Cotes. 2009. Predicting Quantitative Traits with Regression Models for Dense Molecular Markers and Pedigree. *Genetics* **182**: 375-385.

Park T. and G. Casella. 2008. The Bayesian LASSO. *Journal of the American Statistical Association* **103**: 681-686.

Spiegelhalter, D.J., N.G. Best, B.P. Carlin and A. van der Linde. 2002. Bayesian measures of model complexity and fit (with discussion). *Journal of the Royal Statistical Society*, Series B (Statistical Methodology) **64** (4): 583-639.

Examples

```
## Not run:
########################################################################
##Example 1:
########################################################################

rm(list=ls())
setwd(tempdir())
library(BGLR)
data(wheat)     #Loads the wheat dataset

y=wheat.Y[,1]
### Creates a testing set with 100 observations
whichNa<-sample(1:length(y),size=100,replace=FALSE)
yNa<-y
yNa[whichNa]<-NA

### Runs the Gibbs sampler
fm<-BLR(y=yNa,XL=wheat.X,GF=list(ID=1:nrow(wheat.A),A=wheat.A),
                           prior=list(varE=list(df=3,S=0.25),
                           varU=list(df=3,S=0.63),
                           lambda=list(shape=0.52,rate=1e-4,
                           type='random',value=30)),
                           nIter=5500,burnIn=500,thin=1)

MSE.tst<-mean((fm$yHat[whichNa]-y[whichNa])^2)
MSE.tst
MSE.trn<-mean((fm$yHat[-whichNa]-y[-whichNa])^2)
MSE.trn
COR.tst<-cor(fm$yHat[whichNa],y[whichNa])
COR.tst
COR.trn<-cor(fm$yHat[-whichNa],y[-whichNa])
COR.trn

plot(fm$yHat~y,xlab="Phenotype",
     ylab="Pred. Gen. Value" ,cex=.8)
points(x=y[whichNa],y=fm$yHat[whichNa],col=2,cex=.8,pch=19)

x11()
plot(scan('varE.dat'),type="o",
        ylab=expression(paste(sigma[epsilon]^2)))

########################################################################
#Example 2: Ten fold, Cross validation, environment 1,
########################################################################

rm(list=ls())
setwd(tempdir())
library(BGLR)
data(wheat)     #Loads the wheat dataset
nIter<-1500     #For real data sets more samples are needed
burnIn<-500
```

```
thin<-10
folds<-10
y<-wheat.Y[,1]
A<-wheat.A

priorBL<-list(
               varE=list(df=3,S=2.5),
               varU=list(df=3,S=0.63),
               lambda = list(shape=0.52,rate=1e-5,value=20,type='random')
             )

set.seed(123)  #Set seed for the random number generator
sets<-rep(1:10,60)[-1]
sets<-sets[order(runif(nrow(A)))]
COR.CV<-rep(NA,times=(folds+1))
names(COR.CV)<-c(paste('fold=',1:folds,sep=''),'Pooled')
w<-rep(1/nrow(A),folds) ## weights for pooled correlations and MSE
yHatCV<-numeric()

for(fold in 1:folds)
{
   yNa<-y
   whichNa<-which(sets==fold)
   yNa[whichNa]<-NA
   prefix<-paste('PM_BL','_fold_',fold,'_',sep='')
   fm<-BLR(y=yNa,XL=wheat.X,GF=list(ID=(1:nrow(wheat.A)),A=wheat.A),prior=priorBL,
               nIter=nIter,burnIn=burnIn,thin=thin)
   yHatCV[whichNa]<-fm$yHat[fm$whichNa]
   w[fold]<-w[fold]*length(fm$whichNa)
   COR.CV[fold]<-cor(fm$yHat[fm$whichNa],y[whichNa])
}

COR.CV[11]<-mean(COR.CV[1:10])
COR.CV

########################################################################

## End(Not run)
```

BLRCross *Bayesian Linear Regression*

Description

The BLR (‘Bayesian Linear Regression’) function fits various types of parametric Bayesian regressions to continuos outcomes.

Usage

```
BLRCross(y=NULL,my=NULL,vy=NULL,n=NULL,
          XX,Xy,nIter=1500,burnIn=500,
          thin=5,R2=0.5,
          S0=NULL,df0=5,
          priors=NULL,
          idPriors=NULL,
          verbose=TRUE,
          saveAt = "",
          rmExistingFiles=TRUE)
```

Arguments

`y`	A numeric vector of length n, NAs not allowed, if NULL you must provide my, vy and n.
`my`	numeric, sample mean of y. If NULL you must provide y.
`vy`	numeric, sample variance of y. If NULL you must provide y.
`n`	integer, sample size. If NULL you must provide y.
`XX`	A matrix, XX=crossprod(X), with X an incidence matrix of dimension n times p.
`Xy`	A numeric vector of length p, Xy=crossprod(X,y).
`nIter`, `burnIn`, `thin`	(integer) the number of iterations, burn-in and thinning.
`R2`	(numeric, `0<R2<1`) The proportion of variance that one expects, a priori, to be explained by the regression. Only used if the hyper-parameters are not specified; if that is the case, internally, hyper-parameters are set so that the prior modes are consistent with the variance partition specified by R2 and the prior distribution is relatively flat at the mode.
`S0`, `df0`	(numeric) The scale parameter for the scaled inverse-chi squared prior assigned to the residual variance. In the parameterization of the scaled-inverse chi square in BLRCross the expected values is `S0/(df0-2)`. The default value for the df parameter is 5. If the scale is not specified a value is calculated so that the prior mode of the residual variance equals `var(y)*R2`.
`priors`	(list) This is a two-level list used to specify the regression function (or linear predictor). Regression on covariates and other types of random effects are specified in this two-level list. For linear regressions the following options are implemented: FIXED (flat prior), BayesA, BayesB, BRR (Gaussian prior), BayesC, SSVS and RKHS.
`idPriors`	(numeric) an integer vector that allow us to specify the priors for the columns of matrix X.
`verbose`	(logical) if TRUE the iteration history is printed, default TRUE.
`saveAt`	(string) this may include a path and a pre-fix that will be added to the name of the files that are saved as the program runs.
`rmExistingFiles`	(logical) if TRUE removes existing output files from previous runs, default TRUE.

Author(s)

Gustavo de los Campos, Paulino Perez Rodriguez.

References

de los Campos G., H. Naya, D. Gianola, J. Crossa, A. Legarra, E. Manfredi, K. Weigel and J. Cotes. 2009. Predicting Quantitative Traits with Regression Models for Dense Molecular Markers and Pedigree. *Genetics* **182**: 375-385.

de los Campos, G., D. Gianola, G. J. M., Rosa, K. A., Weigel, and J. Crossa. 2010. Semi-parametric genomic-enabled prediction of genetic values using reproducing kernel Hilbert spaces methods. *Genetics Research*, **92**:295-308.

Examples

```
## Not run:

library(BGLR)

p=1000
n=1500

data(mice)
X=scale(mice.X[1:n,1:p],center=TRUE)
QTL=seq(from=50,to=p-50,by=80)

b=rep(0,p)
b[QTL]=1
signal=as.vector(X%*%b)

error=rnorm(sd=sd(signal),n=n)
y=error+signal
y=y-mean(y)

XX=crossprod(X)
Xy=as.vector(crossprod(X,y))

#Example 1

## BayesA ################################################################################
priors=list(list(model="BayesA"))
idPriors=rep(1,p)
fm1=BLRCross(y=y,XX=XX,Xy=Xy,nIter=10000,priors=priors,idPriors=idPriors)
plot(as.vector(X%*%fm1$ETA[[1]]$b),y)

#summary statistics
fm1s=BLRCross(my=mean(y),vy=var(y),n=length(y),
              XX=XX,Xy=Xy,nIter=10000,priors=priors,idPriors=idPriors)
plot(as.vector(X%*%fm1s$ETA[[1]]$b),y)

## BayesB ################################################################################
priors=list(list(model="BayesB"))
```

```
idPriors=idPriors=rep(1,p)
fm1=BLRCross(y=y,XX=XX,Xy=Xy,nIter=10000,burnIn=5000,priors=priors,idPriors=idPriors)
plot(as.vector(X%*%fm1$ETA[[1]]$b),y)
plot(fm1$ETA[[1]]$b)
points(QTL,b[QTL],col="red")

#summary statistics
fm1s=BLRCross(my=mean(y),vy=var(y),n=length(y),
              XX=XX,Xy=Xy,nIter=10000,priors=priors,idPriors=idPriors)
plot(as.vector(X%*%fm1s$ETA[[1]]$b),y)
plot(fm1s$ETA[[1]]$b)
points(QTL,b[QTL],col="red")

## BayesC ##################################################################################
priors=list(list(model="BayesC",R2=NULL,df0=NULL,S0=NULL,probIn=1/100,counts=1E6))
idPriors=rep(1,p)

fm1=BLRCross(y=y,XX=XX,Xy=Xy,nIter=10000,priors=priors,idPriors=idPriors)
plot(as.vector(X%*%fm1$ETA[[1]]$b),y)
plot(fm1$ETA[[1]]$b)
points(QTL,b[QTL],col="red")

plot(fm1$ETA[[1]]$d)
points(QTL,b[QTL],col="red")

#summary statistics
fm1s=BLRCross(my=mean(y),vy=var(y),n=length(y),
              XX=XX,Xy=Xy,nIter=10000,priors=priors,idPriors=idPriors)
plot(as.vector(X%*%fm1s$ETA[[1]]$b),y)
plot(fm1s$ETA[[1]]$b)
points(QTL,b[QTL],col="red")


## SSVS (Absolutely Continuous Spike Slab) ################################################
priors=list(list(model="SSVS",R2=NULL,df0=NULL,S0=NULL,probIn=NULL,counts=NULL))
idPriors=rep(1,p)

fm1=BLRCross(y=y,XX=XX,Xy=Xy,nIter=10000,priors=priors,idPriors=idPriors)
plot(as.vector(X%*%fm1$ETA[[1]]$b),y)
plot(fm1$ETA[[1]]$b)

#summary statistics
fm1s=BLRCross(my=mean(y),vy=var(y),n=length(y),
              XX=XX,Xy=Xy,nIter=10000,priors=priors,idPriors=idPriors)
plot(as.vector(X%*%fm1s$ETA[[1]]$b),y)
plot(fm1s$ETA[[1]]$b)

priors=list(list(model="SSVS",R2=NULL,df0=NULL,S0=NULL,probIn=NULL,
                      counts=NULL,cprobIn=0.5,ccounts=2))
idPriors=rep(1,p)

fm1=BLRCross(y=y,XX=XX,Xy=Xy,nIter=10000,priors=priors,idPriors=idPriors)
plot(as.vector(X%*%fm1$ETA[[1]]$b),y)
plot(fm1$ETA[[1]]$b)

#summary statistics
fm1s=BLRCross(my=mean(y),vy=var(y),n=length(y),
              XX=XX,Xy=Xy,nIter=10000,priors=priors,idPriors=idPriors)

plot(as.vector(X%*%fm1s$ETA[[1]]$b),y)
plot(fm1s$ETA[[1]]$b)

## Ridge Regression ########################################################################
priors=list(list(model="BRR",R2=NULL,df0=NULL,S0=NULL))
idPriors=rep(1,p)

fm1=BLRCross(y=y,XX=XX,Xy=Xy,nIter=10000,priors=priors,idPriors=idPriors)
plot(as.vector(X%*%fm1$ETA[[1]]$b),y)

#summary statistics
fm1s=BLRCross(my=mean(y),vy=var(y),n=length(y),
              XX=XX,Xy=Xy,nIter=10000,priors=priors,idPriors=idPriors)
plot(as.vector(X%*%fm1s$ETA[[1]]$b),y)

#Example 2
priors=list(list(model="BayesC",R2=NULL,df0=NULL,S0=NULL,probIn=NULL,counts=NULL),
            list(model="BayesC",R2=NULL,df0=NULL,S0=NULL,probIn=NULL,counts=NULL))

idPriors=c(rep(1,p/2),rep(2,p/2))

fm2=BLRCross(y=y,XX=XX,Xy=Xy,nIter=10000,priors=priors,idPriors=idPriors)
bHat=c(fm2$ETA[[1]]$b,fm2$ETA[[2]]$b)
plot(as.vector(X%*%bHat),y)
plot(bHat)

#summary statistics
fm2s=BLRCross(my=mean(y),vy=var(y),n=length(y),
              XX=XX,Xy=Xy,nIter=10000,priors=priors,idPriors=idPriors)
bHat=c(fm2s$ETA[[1]]$b,fm2s$ETA[[2]]$b)
plot(as.vector(X%*%bHat),y)
plot(bHat)

#Example 3
priors=list(list(model="BayesC",R2=NULL,df0=NULL,S0=NULL,probIn=NULL,counts=NULL),
            list(model="BRR",R2=NULL,df0=NULL,S0=NULL))
idPriors=c(rep(1,p/2),rep(2,p/2))

fm3=BLRCross(y=y,XX=XX,Xy=Xy,nIter=10000,priors=priors,idPriors=idPriors)
bHat=c(fm3$ETA[[1]]$b,fm3$ETA[[2]]$b)
plot(as.vector(X%*%bHat),y)
plot(bHat)

#summary statistics
fm3s=BLRCross(my=mean(y),vy=var(y),n=length(y),
              XX=XX,Xy=Xy,nIter=10000,priors=priors,idPriors=idPriors)
bHat=c(fm3s$ETA[[1]]$b,fm3s$ETA[[2]]$b)
```

```
plot(as.vector(X%*%bHat),y)
plot(bHat)

#Example 4 Fixed effects

y=y+3

X=cbind(1,X)

XX=crossprod(X)
Xy=as.vector(crossprod(X,y))

priors=list(list(model="FIXED"),
            list(model="BayesC",R2=NULL,df0=NULL,S0=NULL,probIn=1/100,counts=1e6),
            list(model="BRR",R2=NULL,df0=NULL,S0=NULL))
idPriors=c(1,rep(2,p/2),rep(3,p/2))

fm0=BLRCross(y=y,XX=XX,Xy=Xy,nIter=10000,burnIn=5000,priors=priors,idPriors=idPriors)
bHat=c(fm0$ETA[[1]]$b,fm0$ETA[[2]]$b,fm0$ETA[[3]]$b)
plot(y,X%*%bHat)
plot(bHat)

#summary statistics
fm0s=BLRCross(my=mean(y),vy=var(y),n=length(y),
              XX=XX,Xy=Xy,nIter=10000,burnIn=5000,priors=priors,idPriors=idPriors)
bHat=c(fm0s$ETA[[1]]$b,fm0s$ETA[[2]]$b,fm0s$ETA[[3]]$b)
plot(y,X%*%bHat)
plot(bHat)

priors=list(list(model="FIXED"),
            list(model="BayesC",R2=NULL,df0=NULL,S0=NULL,probIn=1/100,counts=1e6),
            list(model="BRR",R2=NULL,df0=NULL,S0=NULL),
            list(model="BayesA"))
idPriors=c(1,rep(2,333),rep(3,333),rep(4,334))

fm0=BLRCross(y=y,XX=XX,Xy=Xy,nIter=10000,burnIn=5000,priors=priors,idPriors=idPriors)
bHat=c(fm0$ETA[[1]]$b,fm0$ETA[[2]]$b,fm0$ETA[[3]]$b,fm0$ETA[[4]]$b)
plot(y,X%*%bHat)
plot(bHat)

#summary statistics
fm0s=BLRCross(my=mean(y),vy=var(y),n=length(y),
              XX=XX,Xy=Xy,nIter=10000,burnIn=5000,priors=priors,idPriors=idPriors)

bHat=c(fm0s$ETA[[1]]$b,fm0s$ETA[[2]]$b,fm0s$ETA[[3]]$b,fm0$ETA[[4]]$b)
plot(y,X%*%bHat)
plot(bHat)


priors=list(list(model="FIXED"),
            list(model="BayesC",R2=NULL,df0=NULL,S0=NULL,probIn=1/100,counts=1e6),
            list(model="BRR",R2=NULL,df0=NULL,S0=NULL),
            list(model="BayesA"),
            list(model="BayesB"))
idPriors=c(1,rep(2,250),rep(3,250),rep(4,250),rep(5,250))

fm0=BLRCross(y=y,XX=XX,Xy=Xy,nIter=10000,burnIn=5000,priors=priors,idPriors=idPriors)
bHat=c(fm0$ETA[[1]]$b,fm0$ETA[[2]]$b,fm0$ETA[[3]]$b,fm0$ETA[[4]]$b,fm0$ETA[[5]]$b)
plot(y,X%*%bHat)
plot(bHat)

#summary statistics
fm0s=BLRCross(my=mean(y),vy=var(y),n=length(y),
              XX=XX,Xy=Xy,nIter=10000,burnIn=5000,priors=priors,idPriors=idPriors)
bHat=c(fm0s$ETA[[1]]$b,fm0s$ETA[[2]]$b,fm0s$ETA[[3]]$b,fm0s$ETA[[4]]$b,fm0s$ETA[[5]]$b)
plot(y,X%*%bHat)
plot(bHat)

priors=list(list(model="FIXED"),
            list(model="BayesC",R2=NULL,df0=NULL,S0=NULL,probIn=1/100,counts=1e6),
            list(model="BRR",R2=NULL,df0=NULL,S0=NULL),
            list(model="BayesA"),
            list(model="BayesB"),
            list(model="SSVS"))
idPriors=c(1,rep(2,200),rep(3,200),rep(4,200),rep(5,200),rep(6,200))

fm0=BLRCross(y=y,XX=XX,Xy=Xy,nIter=10000,burnIn=5000,priors=priors,idPriors=idPriors)
bHat=c(fm0$ETA[[1]]$b,fm0$ETA[[2]]$b,fm0$ETA[[3]]$b,fm0$ETA[[4]]$b,fm0$ETA[[5]]$b,fm0$ETA[[6]]$b)
plot(y,X%*%bHat)
plot(bHat)

#summary statistics
fm0s=BLRCross(my=mean(y),vy=var(y),n=length(y),XX=XX,Xy=Xy,nIter=10000,burnIn=5000,
              priors=priors,idPriors=idPriors)
bHat=c(fm0s$ETA[[1]]$b,fm0s$ETA[[2]]$b,fm0s$ETA[[3]]$b,fm0s$ETA[[4]]$b,
       fm0s$ETA[[5]]$b,fm0s$ETA[[6]]$b)
plot(y,X%*%bHat)
plot(bHat)


## End(Not run)
```

`BLRXy` *Bayesian Linear Regression*

Description

The BLRXy('Bayesian Linear Regression') function fits various types of parametric Bayesian regressions to continuos outcomes. This is a wrapper for function BLRCross.

Usage

```
BLRXy(y, intercept=TRUE, ETA,
             nIter = 1500, burnIn = 500, thin = 5,
             S0 = NULL, df0 = 5, R2 = 0.5,
             verbose = TRUE, saveAt="",
             rmExistingFiles = TRUE)
```

Arguments

`y`	(numeric, n) the data-vector (NAs allowed).
`intercept`	(logical) indicates if an intercept is included.
`ETA`	(list) This is a two-level list used to specify the regression function (or linear predictor). Regression on covariates and other types of random effects are specified in this two-level list. For instance: `ETA=list(list(X=W, model="FIXED"), list(X=Z,model="BRR")),` specifies that the linear predictor should include: an intercept (included by default), a linear regression on W with regression coefficients treated as fixed effects (i.e., flat prior), plus regression on Z, with regression coefficients modeled as in the Bayesian Ridge Regression. The following options are implemented for linear regressions: FIXED (flat prior), BayesA, BayesB, BRR (Gaussian prior), BayesC, SSVS and RKHS.
`nIter`, `burnIn`, `thin`	(integer) the number of iterations, burn-in and thinning.
`saveAt`	(string) this may include a path and a pre-fix that will be added to the name of the files that are saved as the program runs.
`S0`, `df0`	(numeric) The scale parameter for the scaled inverse-chi squared prior assigned to the residual variance, only used with Gaussian outcomes. In the parameterization of the scaled-inverse chi square in BGLR the expected values is `S0/(df0-2)`. The default value for the df parameter is 5. If the scale is not specified a value is calculated so that the prior mode of the residual variance equals `var(y)*R2` (see below). For further details see the vignettes in the package.
`R2`	(numeric, `0<R2<1`) The proportion of variance that one expects, a priori, to be explained by the regression. Only used if the hyper-parameters are not specified; if that is the case, internaly, hyper-paramters are set so that the prior modes are consistent with the variance partition specified by R2 and the prior distribution is relatively flat at the mode. For further details see the vignettes in the package.
`verbose`	(logical) if TRUE the iteration history is printed, default TRUE.
`rmExistingFiles`	(logical) if TRUE removes existing output files from previous runs, default TRUE.

Author(s)

Gustavo de los Campos, Paulino Perez Rodriguez.

References

de los Campos G., H. Naya, D. Gianola, J. Crossa, A. Legarra, E. Manfredi, K. Weigel and J. Cotes. 2009. Predicting Quantitative Traits with Regression Models for Dense Molecular Markers and Pedigree. *Genetics* **182**: 375-385.

de los Campos, G., D. Gianola, G. J. M., Rosa, K. A., Weigel, and J. Crossa. 2010. Semi-parametric genomic-enabled prediction of genetic values using reproducing kernel Hilbert spaces methods. *Genetics Research*, **92**:295-308.

Examples

```
## Not run:

library(BGLR)

p=1000
n=1500

data(mice)
X=scale(mice.X[1:n,1:p],center=TRUE)

A=mice.A
A=A[1:n,1:n]

QTL=seq(from=50,to=p-50,by=80)

b=rep(0,p)
b[QTL]=1
signal=as.vector(X%*%b)

error=rnorm(sd=sd(signal),n=n)
y=error+signal
y=2+y

#Example 1

#BayesA
ETA=list(list(X=X,model="BayesA"))

fm1=BLRXy(y=y,ETA=ETA)
plot(fm1$yHat,y)

#Example 2, missing values
yNA<-y
whichNA<-sample(1:length(y),size=100,replace=FALSE)
yNA[whichNA]<-NA

fm2<-BLRXy(y=yNA,ETA=ETA)
plot(fm2$yHat,y)
points(fm2$yHat[whichNA],y[whichNA],col="red",pch=19)

#Example 3, RKHS with no-missing values
```

```
ETA<-list(list(K=A,model="RKHS"))
fm3<-BLRXy(y=y,ETA=ETA)
plot(fm3$yHat,y)

#Example 4, RKHS with missing values
fm4<-BLRXy(y=yNA,ETA=ETA)
plot(fm4$yHat,y)
points(fm4$yHat[whichNA],y[whichNA],col="red",pch=19)



## End(Not run)
```

getGCovar *getGCovar*

Description

Genetic co-variance matrix using MCMC samples.

Usage

```
getGCovar(X,B)
```

Arguments

X	(numeric) matrix of covariates.
B	(numeric) matrix that contains samples for regression coefficients, 3D array, with dim=c(nRow,p,traits), where nRow number of MCMC samples saved, p is the number of predictors and traits is the number of traits.

Value

Genetic co-variance matrix.

Author(s)

Gustavo de los Campos.

References

Lehermeier, C., G. de los Campos, V. Wimmer and C.-C. Schon. Genomic Variance Estimates: With or without Disequilibrium Covariances?. *J Anim Breed Genet.*, **134** (3): 232-241.

`getVariances` *getVariances*

Description

Computes the sample-variance (var()) for sets of markers as well as the total variance.

Usage

```
getVariances(X, B, sets, verbose=TRUE)
```

Arguments

`X`	(numeric, $n \times p$) incidence matrix for $\boldsymbol{\beta}$.
`B`	(numeric), object returned by the function readBinMat().
`sets`	(numeric).
`verbose`	(logical), if TRUE it shows progress information in the console.

Value

A matrix with variances for markers as well as the total.

Author(s)

Gustavo de los Campos.

Examples

```
## Not run:
#Demos

library(BGLR)
data(wheat)
y=wheat.Y[,1] ; X=scale(wheat.X)
dir.create('test_saveEffects')
setwd('test_saveEffects')
fm=BGLR(y=y,ETA=list(list(X=X,model='BayesB',saveEffects=TRUE)),nIter=12000,thin=2,burnIn=2000)
B=readBinMat('ETA_1_b.bin')
plot(B[,1],type='o',col=4)
VAR=getVariances(B=B,X=X,sets=sample(1:20,size=1279,replace=T))
head(VAR)
plot(VAR[,"total"])


## End(Not run)
```

`mice` *mice dataset*

Description

The mice data comes from an experiment carried out to detect and locate QTLs for complex traits in a mice population (Valdar et al. 2006a; 2006b). This data has already been analyzed for comparing genome-assisted genetic evaluation methods (Legarra et al. 2008). The data file consists of 1814 individuals, each genotyped for 10,346 polymorphic markers. The information is related to obesity and biochemistry and other covariates related to the traits.

Usage

```
data(mice)
```

Format

Matrix mice.A contains the pedigree. The matrix mice.X contains the markes information and the map data.frame contains information for the genetic map. The data.frame mice.pheno contains phenotypical information.

References

Legarra A., Robert-Granie, E. Manfredi, and J. M. Elsen, 2008 Performance of genomic selection in mice. Genetics 180:611-618.

Valdar, W., L. C. Solberg, D. Gauguier, S. Burnett, P. Klenerman et al., 2006a Genome-wide genetic association of complex traits in heterogeneous stock mice. Nat. Genet. 38:879-887.

Valdar, W., L. C. Solberg, D. Gauguier, W. O. Cookson, J. N. P. Rawlis et al., 2006b Genetic and environmental effects on complex traits in mice. Genetics, 174:959-984.

`mice.A` *Pedigree info for the mice dataset*

Description

Is a numerator relationship matrix (1814 x 1814) computed from a pedigree that traced back many generations.

References

de los Campos G., H. Naya, D. Gianola, J. Crossa, A. Legarra, E. Manfredi, K. Weigel and J. Cotes. 2009. Predicting Quantitative Traits with Regression Models for Dense Molecular Markers and Pedigree. *Genetics* **182**: 375-385.

`mice.map` *Genetic map info for the mice dataset*

Description

Is a data.frame with 10436 rows and 4 columns: chr, snp_id, mbp and alleles.

The data can be downloaded from http://mtweb.cs.ucl.ac.uk/mus/www/GSCAN/HS_GENOTYPES/.

`mice.pheno` *Phenotypical data for the mice dataset*

Description

A data frame with pheotypical information related to obesity and biochemistry. The data frame has several columns: SUBJECT.NAME, Obesity.BMI, Obesity.BodyLength, Obesity.Date.Month, Obesity.Date.Year, Obesity.Date.Season, Obesity.Date.StudyStartSeconds, Obesity.Date.Hour, Obesity.Date.StudyDay, GENDER, Obesity.EndNormalBW, CoatColour, CageDensity, Litter, cage, Biochem.Albumin, Biochem.ALP, Biochem.ALT, Biochem.AST, Biochem.Calcium, Biochem.Chloride, Biochem.Creatinine, Biochem.Glucose, Biochem.HDL, Biochem.LDL, Biochem.Phosphorous, Biochem.Potassium, Biochem.Sodium, Biochem.Tot.Cholesterol, Biochem.Tot.Protein, Biochem.Triglycerides, Biochem.Urea, Biochem.EndNormalBW, Biochem.Date.StudyDay, Biochem.Date.Season, Biochem.Date.Month, Biochem.Date.Year and Biochem.Age.

The data can be downloaded from http://mtweb.cs.ucl.ac.uk/mus/www/GSCAN/HS_PHENOTYPES/.

`mice.X` *Molecular markers*

Description

Is a matrix (1814 x 10346) with SNP markers.

The data can be downloaded from http://mtweb.cs.ucl.ac.uk/mus/www/GSCAN/HS_GENOTYPES/.

`Multitrait` *Multi trait models*

Description

The Multitrait function fits Bayesian multitrait models with arbitrary number of random effects ussing a Gibbs sampler. The data equation is as follows:

$$y_j = 1\mu_j + X_{Fj}\beta_{Fj} + X_1\beta_{j1} + ... + X_k\beta_{jk} + u_{j1} + ... + u_{jq} + e_j,$$

where:

- y_j is a n-dimensional response vector of phenotypes (NAs allowed) with y_{ij} representing the phenotypic record of the i-th subject for the j-th trait.
- μ_j is an intercept.
- X_{Fj} is a matrix of fixed effects.
- β_{Fj} is a vector of fixed effects.
- X_s is an incidence matrix for predictors that are common for all the individuals (e.g. markers), s=1,...,k.
- β_{js} is a vector of regression coefficients, s=1,...,k. Different priors can be assigned to these regression coefficients (spike-slab and Gaussian) and regression coefficients are correlated across traits.
- u_{jr} is a n-dimensional vector of random effects.
- e_j is a n-dimensional vector of residuals.

Usage

```
Multitrait(y, ETA, intercept=TRUE,resCov = list(df0=5,S0=NULL,type="UN"),
  R2=0.5, nIter=1000,burnIn=500,thin=10, saveAt="",verbose=TRUE)
```

Arguments

y	a matrix of dimension $n \times t$, where n is the number of individials in each trait, t is the number of traits, NAs are allowed.
ETA	(list) This is a two-level list used to specify the regression function (or linear predictor). Regression on covariates and other types of random effects are specified in this two-level list. For instance:

```
ETA=list(list(X=W, model="FIXED"),
      list(X=Z,model="BRR"),
      list(K=G,model="RKHS"))
```

specifies that the linear predictor should include: a linear regression on W with regression coefficients treated as fixed effects (i.e., flat prior), plus regression on Z, with regression coefficients modeled as in the Ridge Regression plus and a random effect with co-variance structure G.

`intercept`	logical, if TRUE (default) an intercept is included.
`nIter, burnIn, thin`	(integer) the number of iterations, burn-in and thinning.
`resCov`	A list used to define the co-variance matrix for model residuals (R). Four co-variance strucures are supported: i) Unstructured (`"UN"`), ii)Diagonal (`"DIAG"`), iii)Factor Analytic (`"FA"`) and iv)Recursive (`"REC"`), for example: `resCov=list(type="UN", df0=4, S0=V)` specifies an UN-structured covariance matrix, with an Inverse Whishart prior with degree of freedom df0 (scalar) and scale matrix (t x t) V.
`saveAt`	(string) this may include a path and a pre-fix that will be added to the name of the files that are saved as the program runs.
`R2`	(numeric, `0<R2<1`) The proportion of co-variance that one expects, a priori, to be explained by the regression. Only used if the hyper-parameters are not specified; if that is the case, internaly, hyper-paramters are set so that the prior modes are consistent with the co-variance partition specified by R2 and the prior distribution is relatively flat at the mode.
`verbose`	(logical) if TRUE the iteration history is printed, default TRUE.

Details

Conditional distribution of the data

Model residuals are assumed to follow a multivariate normal distribution, with null mean and co-variance matrix $Cov((e_1', ..., e_n')') = R_0 \otimes I$ where R_0 is a t x t (within-subject) covariance matrix of model residuals and n-dimensional identity matrix. Therefore:

$$p(y_{i1}, ..., y_{it}|\theta) = MN(\eta_i, R_0)$$

where MN(.,.), denotes the multivariate normal distribution with mean η_i and covariance matrix R_0; here η_i is a t-dimensional vector whose entries are the expected values of the response variable for the i-th individual.

Prior distribution

The prior distribution is structured hierarchically. The first level of the prior specifies the distribution of the fixed and random effects given the codispersion parameters (the covariance matrices of the random effects, see below). The priors for the codispersion parameters are specified in a deeper level of the hierarchy.

The intercepts and vectors of fixed effects are assigned flat prior (each unknown is assigned a Gaussian prior with null mean and very large variance).

The vectors of random effects u_r are assigned independent multivariate normal priors with null mean and covariance matrices $Cov(u_r) = G_r \otimes K_r$, u_r represent the vector of effects for the r-th random effects (sorted by subject first and trait within subject), G_r is an t x t (within-subject) covariance matrix of the r-th random effect and K_r is a user defined (between subjects) covariance matrix for the r-th random effect, for instance, may be a pedigree or marker-based relationship matrix. The covariance matrix of random effects are assigned Inverse Wishart priors (for the case of unstructured options) or priors that are structured according to some model (diagonal, factor analytic or recursive).

The vector or regression coefficients β_s are assigned Gaussian and Spike Slab priors whose covariance matrixes depend on a Ω_s covariance matrix of dimmensions t x t (within subject). The covariance matrix of random effects are assigned Inverse Wishart priors (for the case of unstructured options) or priors that are structured according to some model (diagonal, factor analytic or recursive).

Algorithm

Internally, samples from all the model unknowns are drawn using a Gibbs sampler (i.e., based on fully conditional distributions).

Value

List containing estimated posterior means and estimated posterior standard deviations.

Author(s)

Gustavo de los Campos, Paulino Perez-Rodriguez

References

Burgueno, J., G. de los Campos, K. Weigel and J. Crossa. 2012. Genomic Prediction of Breeding Values when Modelling Genotype x Environment Interaction using Pedigree and Dense Molecular Markers. *Crop Sci.*, **52(2)**:707-719.

de los Campos, G. and D. Gianola. 2007. Factor analysis models for structuring covariance matrices of additive genetic effects: a Bayesian implementation. *Genet. Sel. Evol.*, **39**:481-494.

Cheng, H., K. Kadir, J., Zeng, D. Garrick and R. Fernando. 2018. Genomic Prediction from Multiple-Trait Bayesian Regression Methods Using Mixture Priors. *Genetics*, **209(1)**: 89-103.

Sorensen, D. and D. Gianola. 2002. Likelihood, Bayesian, and MCMC methods in quantitative genetics. *Springer-Verlag, New York.*

Examples

```
## Not run:

library(BGLR)
data(wheat)
X<-wheat.X
K<-wheat.A
y<-wheat.Y

#Example 1, Spike Slab regression
ETA1<-list(list(X=X,model="SpikeSlab"))

fm1<-Multitrait(y=y,ETA=ETA1,nIter=1000,burnIn=500)

#Example 2, Ridge Regression
ETA2<-list(list(X=X,model="BRR"))
fm2<-Multitrait(y=y,ETA=ETA2,nIter=1000,burnIn=500)

#Example 3, Random effects with user defined covariance structure
```

```
#for individuals derived from pedigree
ETA3<-list(list(K=K,model="RKHS"))
fm3<-Multitrait(y=y,ETA=ETA3,nIter=1000,burnIn=500)

#Example 4, Markers and pedigree
ETA4<-list(list(X=X,model="BRR"), list(K=K,model="RKHS"))

fm4<-Multitrait(y=y,ETA=ETA4,nIter=1000,burnIn=500)

#Example 5, recursive structures for within subject covariance matrix
M1 <- matrix(nrow = 4, ncol = 4, FALSE)
M1[3, 2] <- M1[4, 2] <- TRUE # Adding recursion from trait 2 onto traits 3 and 4
M1[4, 3] <- TRUE # Adding recursion from trait 3 on trait 4

ETA5<-list(list(K=K,model="RKHS",Cov=list(type="REC",M=M1)))
fm5<-Multitrait(y=y,ETA=ETA5,nIter=1000,burnIn=500)

#Example 6, diagonal residual covariance matrix with the predictor
#used in example 5
residual1<-list(type="DIAG")
fm6<-Multitrait(y=y,ETA=ETA5,resCov=residual1,nIter=1000,burnIn=500)


## End(Not run)
```

`plot.BGLR` *Plots for BGLR Analysis*

Description

Plots observed vs predicted values for objects of class BGLR.

Usage

```
## S3 method for class 'BGLR'
plot(x, ...)
```

Arguments

`x`	An object of class `BGLR`.
`...`	Further arguments passed to or from other methods.

Author(s)

Gustavo de los Campos, Paulino Perez Rodriguez,

See Also

`BGLR`.

Examples

```
## Not run:

setwd(tempdir())
library(BGLR)
data(wheat)
out=BLR(y=wheat.Y[,1],XL=wheat.X)
plot(out)


## End(Not run)
```

---

predict.BGLR *Model Predictions*

Description

extracts predictions from the results of BGLR function.

Usage

```
## S3 method for class 'BGLR'
predict(object, newdata, ...)
```

Arguments

`object`	An object of class `BGLR`.
`newdata`	Currently not supported, for new data you should assing missing value indicator (NAs) to the corresponding entries in the response vector (y).
`...`	Further arguments passed to or from other methods.

Author(s)

Gustavo de los Campos, Paulino Perez Rodriguez,

See Also

`BGLR`.

Examples

```
## Not run:

setwd(tempdir())
library(BGLR)
data(wheat)
out=BLR(y=wheat.Y[,1],XL=wheat.X)
predict(out)


## End(Not run)
```

`readBinMat` *readBinMat*

Description

Function to read effects saved by BGLR when ETA[[j]]$saveEffects=TRUE.

Usage

```
readBinMat(filename,byrow=TRUE,storageMode="double")
```

Arguments

`filename`	(string), the name of the file to be read.
`byrow`	(logical), if TRUE the matrix is created by filling its corresponding elements by rows.
`storageMode`	(character), the storage mode used to save effects via ETA[[j]]$storageMode: 'double' (default) or 'single'.

Value

A matrix with samples of regression coefficients.

Author(s)

Gustavo de los Campos.

Examples

```
## Not run:
#Demos

library(BGLR)
data(wheat)
```

```
y=wheat.Y[,1] ; X=scale(wheat.X)
dir.create('test_saveEffects')
setwd('test_saveEffects')
fm=BGLR(y=y,ETA=list(list(X=X,model='BayesB',saveEffects=TRUE)),nIter=12000,thin=2,burnIn=2000)
B=readBinMat('ETA_1_b.bin')


## End(Not run)
```

`readBinMatMultitrait` *readBinMatMultitrait*

Description

Function to read effects saved by Multitrait when ETA[[j]]$saveEffects=TRUE.

Usage

```
readBinMatMultitrait(filename,storageMode="double")
```

Arguments

`filename`	(string), the name of the file to be read.
`storageMode`	(character), the storage mode used to save effects via ETA[[j]]$storageMode: 'double' (default) or 'single'.

Value

A 3D array, with dim=c(nRow,p,traits), where nRow number of MCMC samples saved, p is the number of predictors and traits is the number of traits.

Author(s)

Gustavo de los Campos, Paulino Perez-Rodriguez.

Examples

```
## Not run:
library(BGLR)

data(wheat)
y<-wheat.Y
X<-wheat.X

fm<-Multitrait(y=y,ETA=list(list(X=X,model='BRR',saveEffects=TRUE)),
                      nIter=1000,thin=10,burnIn=500)
```

```
 B<-readBinMatMultitrait('ETA_1_beta.bin')


## End(Not run)
```

`read_bed` *read_bed*

Description

This function reads genotype information stored in binary PED (BED) files used in plink. These files save space and time. The pedigree/phenotype information is stored in a separate file (*.fam) and the map information is stored in an extededed MAP file (*.bim) that contains information about the allele names, which would otherwise be lost in the BED file. More details `http://zzz.bwh.harvard.edu/plink/binary.shtml`.

Usage

```
read_bed(bed_file,bim_file,fam_file,na.strings,verbose)
```

Arguments

`bed_file`	binary file with genotype information.
`bim_file`	text file with pedigree/phenotype information.
`fam_file`	text file with extended map information.
`na.strings`	missing value indicators, default=c("0","-9").
`verbose`	logical, if true print hex dump of bed file.

Value

The routine will return a vector of dimension n*p (n=number of individuals, p=number of snps), with the snps(individuals) stacked, depending whether the BED file is in SNP-major or individual-major mode.

The vector contains integer codes:

Integer code	Genotype
0	00 Homozygote "1"/"1"
1	01 Heterozygote
2	10 Missing genotype
3	11 Homozygote "2"/"2"

Author(s)

Gustavo de los Campos, Paulino Perez Rodriguez,

Examples

```
## Not run:

library(BGLR)
demo(read_bed)


## End(Not run)
```

`read_ped` *read_ped*

Description

This function reads genotype information stored in PED format used in plink.

Usage

```
read_ped(ped_file)
```

Arguments

`ped_file`	ASCII file with genotype information.

Details

The PED file is a white-space (space or tab) delimited file: the first six columns are mandatory:

Family ID Individual ID Paternal ID Maternal ID Sex (1=male; 2=female; other=unknown) Phenotype

The IDs are alphanumeric: the combination of family and individual ID should uniquely identify a person. A PED file must have 1 and only 1 phenotype in the sixth column. The phenotype can be either a quantitative trait or an affection status column.

Value

The routine will return a vector of dimension n*p (n=number of individuals, p=number of snps), with the snps stacked.

The vector contains integer codes:

Integer code	Genotype
0	00 Homozygote "1"/"1"
1	01 Heterozygote
2	10 Missing genotype
3	11 Homozygote "2"/"2"

Author(s)

Gustavo de los Campos, Paulino Perez Rodriguez,

Examples

```
## Not run:

library(BGLR)
demo(read_ped)


## End(Not run)
```

`residuals.BGLR` *Extracts models residuals*

Description

extracts model residuals from objects returned by BGLR function.

Usage

```
## S3 method for class 'BGLR'
residuals(object, ...)
```

Arguments

`object`	An object of class `BGLR`.
`...`	Further arguments passed to or from other methods.

Author(s)

Gustavo de los Campos, Paulino Perez Rodriguez,

See Also

`BGLR`.

Examples

```
## Not run:

setwd(tempdir())
library(BGLR)
data(wheat)
out=BLR(y=wheat.Y[,1],XL=wheat.X)
residuals(out)
```

```
## End(Not run)
```

`simulated3t` *simulated data for 3 traits*

Description

Simulated dataset for three traits. Markers, QTL and phenotypes are simulated for three traits. Here we extend the simulation scheme described by Cheng et al. (2018) for the case of three traits. So we simulated 100 evenly spaced loci on each of 4 chromosomes of length 10 cM. We selected 10 loci on each chromosome as QTL. We sampled states from Bernoulli distribution with p=0.5. After that we simulated 500 generations to obtain linkage disequilibrium using 500 males and 500 females that were mated at random. Random mating was continued for 5 more generations and population size was increased to 4000 males and 4000 females. The QTL on chromosome 1 has effect on trait 1, wherehas chomosomes 1 and 2 had effects on traits 2 and 3 respectively. The QTL on chromosome 4 had effects on the 3 traits. The effects of QTL on chromosome 4 were simulated from a multivariate normal distribution with null mean and variance covariance matrix:

```
1.00  0.75  0.50
0.75  1.00  0.75
0.50  0.75  1.00
```

The genetic values were scaled to have variance 1.0. The phenotypes for these traits were obtained by adding residuals to genetic values, residuals were simulated from a multivariate normal distribution with null mean and covariance matrix:

```
6.0  6.0  1.0
6.0  8.0  2.0
1.0  2.0  1.0
```

In total, 8000 individuals were simulated and the genetic covariance matrix is:

```
1.00  0.34  0.07
0.34  1.00  0.21
0.07  0.21  1.00
```

Usage

```
data(simulated3t)
```

Format

Matrix simulated3t.X contains the marker information. Matrix simulated3t.pheno contains the phenotypical information.

References

Cheng, H., K. Kadir, J., Zeng, D. Garrick and R. Fernando. 2018. Genomic Prediction from Multiple-Trait Bayesian Regression Methods Using Mixture Priors. *Genetics*, **209(1)**: 89-103.

`simulated3t.pheno` *Phenotypical data for simulated dataset with 3 traits*

Description

A matrix with 8000 rows and 6 columns. Columns 1 to 3 corresponds to simulated phenotypes for 3 traits, columns 4-6 corresponds to true simulated genetic values for 3 traits.

`simulated3t.X` *Molecular markers*

Description

Is a matrix (8000 x 327) with recoded SNP markers for additive effects as 0, 1, 2.

`summary.BGLR` *summary for BGLR fitted models*

Description

Gives a summary for a fitted model using BGLR function.

Usage

```
## S3 method for class 'BGLR'
summary(object, ...)
```

Arguments

`object`	An object of class `BGLR`.
`...`	Further arguments passed to or from other methods.

Author(s)

Gustavo de los Campos, Paulino Perez Rodriguez,

See Also

`BGLR`.

Examples

```
## Not run:

setwd(tempdir())
library(BGLR)
data(wheat)
out=BLR(y=wheat.Y[,1],XL=wheat.X)
summary(out)


## End(Not run)
```

---

vech *Extract Lower Triangular Elements from a Symmetric Matrix*

Description

This function takes a symmetric matrix and extracts a list of all lower triangular elements.

Usage

```
vech(x)
```

Arguments

x	A symmetric matrix.

Details

This function checks to make sure the matrix is square, but it does not check for symmetry (it just pulls the lower triangular elements). The elements are stored in column major order. The original matrix can be restored using the xpnd command.

Value

A list of the lower triangular elements.

See Also

xpnd

Examples

```
   symmat <- matrix(c(1,2,3,4,2,4,5,6,3,5,7,8,4,6,8,9),4,4)
   vech(symmat)
```

`wheat` *wheat dataset*

Description

Information from a collection of 599 historical CIMMYT wheat lines. The wheat data set is from CIMMYT's Global Wheat Program. Historically, this program has conducted numerous international trials across a wide variety of wheat-producing environments. The environments represented in these trials were grouped into four basic target sets of environments comprising four main agroclimatic regions previously defined and widely used by CIMMYT's Global Wheat Breeding Program. The phenotypic trait considered here was the average grain yield (GY) of the 599 wheat lines evaluated in each of these four mega-environments.

A pedigree tracing back many generations was available, and the Browse application of the International Crop Information System (ICIS), as described in McLaren *et al.* (2000, 2005) was used for deriving the relationship matrix A among the 599 lines; it accounts for selection and inbreeding.

Wheat lines were recently genotyped using 1447 Diversity Array Technology (DArT) generated by Triticarte Pty. Ltd. (Canberra, Australia; https://www.diversityarrays.com). The DArT markers may take on two values, denoted by their presence or absence. Markers with a minor allele frequency lower than 0.05 were removed, and missing genotypes were imputed with samples from the marginal distribution of marker genotypes, that is, $x_{ij} = Bernoulli(\hat{p}_j)$, where $\hat{p}_j$ is the estimated allele frequency computed from the non-missing genotypes. The number of DArT MMs after edition was 1279.

Usage

```
data(wheat)
```

Format

Matrix Y contains the average grain yield, column 1: Grain yield for environment 1 and so on. The matrix A contains additive relationship computed from the pedigree and matrix X contains the markers information.

Source

International Maize and Wheat Improvement Center (CIMMYT), Mexico.

References

McLaren, C. G., L. Ramos, C. Lopez, and W. Eusebio. 2000. "Applications of the geneaology manegment system." In *International Crop Information System. Technical Development Manual, version VI*, edited by McLaren, C. G., J.W. White and P.N. Fox. pp. 5.8-5.13. CIMMyT, Mexico: CIMMyT and IRRI.

McLaren, C. G., R. Bruskiewich, A.M. Portugal, and A.B. Cosico. 2005. The International Rice Information System. A platform for meta-analysis of rice crop data. *Plant Physiology* **139**: 637-642.

`wheat.A` *Pedigree info for the wheat dataset*

Description

Is a numerator relationship matrix (599 x 599) computed from a pedigree that traced back many generations. This relationship matrix was derived using the Browse application of the International Crop Information System (ICIS), as described in McLaren *et al.* (2000, 2005).

Source

International Maize and Wheat Improvement Center (CIMMYT), Mexico.

References

McLaren, C. G., L. Ramos, C. Lopez, and W. Eusebio. 2000. "Applications of the geneaology manegment system." In *International Crop Information System. Technical Development Manual, version VI*, edited by McLaren, C. G., J.W. White and P.N. Fox. pp. 5.8-5.13. CIMMyT, Mexico: CIMMyT and IRRI.

McLaren, C. G., R. Bruskiewich, A.M. Portugal, and A.B. Cosico. 2005. The International Rice Information System. A platform for meta-analysis of rice crop data. *Plant Physiology* **139**: 637-642.

`wheat.sets` *Sets for cross validation (CV)*

Description

Is a vector (599 x 1) that assigns observations to 10 disjoint sets; the assignment was generated at random. This is used later to conduct a 10-fold CV.

Source

International Maize and Wheat Improvement Center (CIMMYT), Mexico.

`wheat.X` *Molecular markers*

Description

Is a matrix (599 x 1279) with DArT genotypes; data are from pure lines and genotypes were coded as 0/1 denoting the absence/presence of the DArT. Markers with a minor allele frequency lower than 0.05 were removed, and missing genotypes were imputed with samples from the marginal distribution of marker genotypes, that is, $x_{ij} = Bernoulli(\hat{p}_j)$, where $\hat{p}_j$ is the estimated allele frequency computed from the non-missing genotypes. The number of DArT MMs after edition was 1279.

Source

International Maize and Wheat Improvement Center (CIMMYT), Mexico.

`wheat.Y` *Grain yield*

Description

A matrix (599 x 4) containing the 2-yr average grain yield of each of these lines in each of the four environments (phenotypes were standardized to a unit variance within each environment).

Source

International Maize and Wheat Improvement Center (CIMMYT), Mexico.

`write_bed` *write_bed*

Description

This function writes genotype information into a binary PED (BED) filed used in plink. For more details about this format see `http://zzz.bwh.harvard.edu/plink/binary.shtml`.

Usage

```
write_bed(x,n,p,bed_file)
```

Arguments

`n`	integer, number of individuals.
`p`	integer, number of SNPs.
`x`	integer vector that contains the genotypic information coded as 0,1,2 and 3 (see details below). The information must be in snp major order. The vector should be of dimension n*p with the snps stacked.
`bed_file`	output binary file with genotype information.

Details

The vector contains integer codes:

Integer code	Genotype
0	00 Homozygote "1"/"1"
1	01 Heterozygote
2	10 Missing genotype
3	11 Homozygote "2"/"2"

Author(s)

Gustavo de los Campos, Paulino Perez Rodriguez,

Examples

```
## Not run:

library(BGLR)
demo(write_bed)


## End(Not run)
```

`xpnd` *Expand a Vector into a Symmetric Matrix*

Description

This function takes a vector of appropriate length (typically created using `vech`) and creates a symmetric matrix.

Usage

```
xpnd(x, nrow = NULL)
```

Arguments

`x`	A list of elements to expand into symmetric matrix.
`nrow`	The number of rows (and columns) in the returned matrix. Look into the details.

Details

This function is particularly useful when dealing with variance covariance matrices. Note that R stores matrices in column major order, and that the items in `x` will be recycled to fill the matrix if need be.

The number of rows can be specified or automatically computed from the number of elements in a given object via $(-1 + \sqrt{(1 + 8 * length(x))})/2$.

Value

An $(nrows \times nrows)$ symmetric matrix.

See Also

`vech`

Examples

```
xpnd(c(1,2,3,4,4,5,6,7,8,9),4)
xpnd(c(1,2,3,4,4,5,6,7,8,9))
```

Index