Package ‘bWGR’

October 30, 2024

Type Package

Title Bayesian Whole-Genome Regression

Version 2.2.11

Date 2024-10-30

Maintainer Alencar Xavier <alenxav@gmail.com>

Description Whole-genome regression methods on Bayesian framework fitted via EM or Gibbs sampling, single step (<doi:10.1534/g3.119.400728>), univariate and multivariate (<doi:10.1186/s12711-022-00730-w>), with optional kernel term and sampling techniques (<doi:10.1186/s12859-017-1582-3>).

License GPL-3

Imports Matrix, Rcpp

LinkingTo Rcpp, RcppEigen

Depends R (>= 4.0)

NeedsCompilation yes

Repository CRAN

Author Alencar Xavier [aut, cre] (<https://orcid.org/0000-0001-5034-9954>), William Muir [aut], David Habier [aut], Kyle Kocak [aut], Shizhong Xu [aut], Katy Rainey [aut]

Date/Publication 2024-10-30 16:50:02 UTC

Contents

`bWGR-package` *Bayesian Whole-Genome Regression*

Description

Whole-genome regression methods on Bayesian framework fitted via EM or Gibbs sampling, single step (<doi:10.1534/g3.119.400728>), univariate and multivariate (<doi:10.1186/s12711-022-00730-w>), with optional kernel term and sampling techniques (<doi:10.1186/s12859-017-1582-3>).

Details

Package:	bWGR
Type:	Package
Version:	2.2.11
Date:	2024-10-30
License:	GPL-3

Author(s)

Alencar Xavier, William Muir, David Habier, Kyle Kocak, Shizhong Xu, Katy Rainey. Maintainer: Alencar Xavier <alenxav@gmail.com>

Examples

```
data(tpod)
Fit = wgr(y,gen)
cor(y,Fit$hat)
```

`Dataset` *Tetra-seed Pods*

Description

Two biparental crosses phenotyped for the percentage of pods containing four seeds

Usage

```
data(tpod)
```

Details

Soybean nested association panel with 2 families (fam) containing 196 indiviuals. Genotypic matrix (gen) have 376 SNP across 20 chromosome (chr). Phenotypic information (y) regards the proportion of tetra-seed pods. Data provided by Rainey Lab for Soybean Breeding and Genetics, Purdue University.

Author(s)

Alencar Xavier and Katy Rainey

`WGR1 (MC)` *MCMC Whole-genome Regression*

Description

Univariate model to find breeding values through regression with optional resampling techniques (Xavier et al. 2017) and polygenic term (Kernel). See "Details" for additional standalone functions written in C++.

Usage

```
wgr(y,X,it=1500,bi=500,th=1,bag=1,rp=FALSE,iv=FALSE,de=FALSE,
    pi=0,df=5,R2=0.5,eigK=NULL,VarK=0.95,verb=FALSE)
```

Arguments

`y`	Numeric vector of observations (n) describing the trait to be analyzed. `NA` is allowed.
`X`	Numeric matrix containing the genotypic data. A matrix with n rows of observations and (m) columns of molecular markers.
`it`	Integer. Number of iterations or samples to be generated.
`bi`	Integer. Burn-in, the number of iterations or samples to be discarted.
`th`	Integer. Thinning parameter, used to save memory by storing only one every 'th' samples.
`bag`	If different than one, it indicates the proportion of data to be subsampled in each Markov chain. For datasets with moderate number of observations, values of bag from 0.30 to 0.60 may speed up computation without losses in predicion properties. This argument enable users to enhance MCMC through subsampling (Xavier et al. 2017).
`rp`	Logical. Use replacement for bootstrap samples when bag is different than one.
`iv`	Logical. Assign markers independent variance, a T prior from a mixture of normals. If true, turns the default model BLUP into BayesA.
`de`	Logical. Assign markers independent variance through double-exponential prior. If true, turns the default model BLUP into Bayesian LASSO. This argument overides iv.

`pi`	Value between 0 and 1. If greater than zero it activates variable selection, where markers have expected probability pi of having null effect.
`df`	Prior degrees of freedom of variance components.
`R2`	Expected R2, used to calculate the prior shape.
`eigK`	Output of function `eigen`. Spectral decomposition of the kernel used as a second random effect (eg. pedigree matrix).
`VarK`	Numeric between 0 and 1. For reduction of dimensionality. Indicates the proportion of variance explained by Eigenpairs used to fit second random effect.
`verb`	Logical. If verbose is TRUE, function displays MCMC progress bar.

Details

The model for the whole-genome regression is as follows:

$$y = mu + Xb + u + e$$

where y is the response variable, mu is the intercept, X is the genotypic matrix, b is the regression coefficient or effect of an allele substitution, with d probability of being included into the model, u is the polygenic term if a kernel is used, and e is the residual term.

Users can obtain four WGR methods out of this function: BRR (pi=0,iv=F), BayesA (pi=0,iv=T), BayesB (pi=0.95,iv=T), BayesC (pi=0.95,iv=F) and Bayesian LASSO or BayesL (pi=0,de=T). Theoretical basis of each model is described by de los Campos et al. (2013).

Gibbs sampler that updates regression coefficients is adapted from GSRU algorithm (Legarra and Misztal 2008). The variable selection of functions wgr, $BayesB$ and $BayesC$ works through the unconditional prior algorithm proposed by Kuo and Mallick (1998), whereas $BayesCpi$ and $BayesDpi$ are based on Metropolis-Hastings. Prior shape estimates are computed as Sb = R2*df*var(y)/MSx and Se = (1-R2)*df*var(y), with an exception for $BayesC$ and $BayesCpi$ where the prior shape is Sb = R2*df*var(y)/MSx/(1-pi). The polygenic term is solved by Bayesian algorithm of reproducing kernel Hilbert Spaces proposed by de los Campos et al. (2010).

In addition to `wgr`, standalone C++ functions available include:

01) `BayesA(y,X,it=1500,bi=500,df=5,R2=0.5)`

02) `BayesB(y,X,it=1500,bi=500,pi=0.95,df=5,R2=0.5)`

03) `BayesC(y,X,it=1500,bi=500,pi=0.95,df=5,R2=0.5)`

04) `BayesCpi(y,X,it=1500,bi=500,df=5,R2=0.5)`

05) `BayesDpi(y,X,it=1500,bi=500,df=5,R2=0.5)`

06) `BayesL(y,X,it=1500,bi=500,df=5,R2=0.5)`

07) `BayesRR(y,X,it=1500,bi=500,df=5,R2=0.5)`

The implementations that support two random effects include:

08) `BayesA2(y,X1,X2,it=1500,bi=500,df=5,R2=0.5)`

09) `BayesB2(y,X1,X2,it=1500,bi=500,pi=0.95,df=5,R2=0.5)`

10) `BayesRR2(y,X1,X2,it=1500,bi=500,df=5,R2=0.5)`

And the cross-validation for the C++ implementations, with arguments analogous to `emCV`.

```
mcmcCV(y,gen,k=5,n=5,it=1500,bi=500,pi=0.95,df=5,R2=0.5,avg=T,llo=NULL,tbv=NULL,ReturnGebv=FALSE)
```

Value

The function wgr returns a list with expected value from the marker effect (b), probability of marker being in the model (d), regression coefficient (g), variance of each marker (Vb), the intercept (mu), the polygene (u) and polygenic variance (Vk), residual variance (Ve) and the fitted value (hat).

Author(s)

Alencar Xavier

References

de los Campos, G., Hickey, J. M., Pong-Wong, R., Daetwyler, H. D., and Calus, M. P. (2013). Whole-genome regression and prediction methods applied to plant and animal breeding. Genetics, 193(2), 327-345.

de los Campos, G., Gianola, D., Rosa, G. J., Weigel, K. A., & Crossa, J. (2010). Semi-parametric genomic-enabled prediction of genetic values using reproducing kernel Hilbert spaces methods. Genetics Research, 92(04), 295-308.

Kuo, L., & Mallick, B. (1998). Variable selection for regression models. Sankhya: The Indian Journal of Statistics, Series B, 65-81.

Legarra, A., & Misztal, I. (2008). Technical note: Computing strategies in genome-wide selection. Journal of dairy science, 91(1), 360-366.

Xavier, A., Xu, S., Muir, W., & Rainey, K. M. (2017). Genomic prediction using subsampling. BMC bioinformatics, 18(1), 191.

Examples

```
    ## Not run:

# Load data
data(tpod)

# BLUP
fit_BRR = wgr(y,gen,iv=FALSE,pi=0)
cor(y,fit_BRR$hat)

# BayesA
fit_BayesA = wgr(y,gen,iv=TRUE,pi=0)
cor(y,fit_BayesA$hat)

# BayesB
fit_BayesB = wgr(y,gen,iv=TRUE,pi=.95)
cor(y,fit_BayesB$hat)

# BayesC
fit_BayesC = wgr(y,gen,iv=FALSE,pi=.95)
cor(y,fit_BayesC$hat)

# BayesCpi
fit_BayesCpi = BayesCpi(y,gen)
```

```
cor(y,fit_BayesCpi$hat)

# BayesDpi
fit_BayesDpi = BayesDpi(y,gen)
cor(y,fit_BayesDpi$hat)

# BayesL
fit_BayesL = wgr(y,gen,de=TRUE)
cor(y,fit_BayesL$hat)

# Bagging BLUP
fit_Bag = wgr(y,gen,bag=0.5)
cor(y,fit_Bag$hat)


## End(Not run)
```

WGR2 (EM) *Expectation-Maximization WGR*

Description

Univariate models to find breeding values through regression fitted via expectation-maximization implemented in C++.

Usage

```
emRR(y, gen, df = 10, R2 = 0.5)
emBA(y, gen, df = 10, R2 = 0.5)
emBB(y, gen, df = 10, R2 = 0.5, Pi = 0.75)
emBC(y, gen, df = 10, R2 = 0.5, Pi = 0.75)
emBCpi(y, gen, df = 10, R2 = 0.5, Pi = 0.75)
emBL(y, gen, R2 = 0.5, alpha = 0.02)
emEN(y, gen, R2 = 0.5, alpha = 0.02)
emDE(y, gen, R2 = 0.5)
emML(y, gen, D = NULL)
lasso(y, gen)

emCV(y, gen, k = 5, n = 5, Pi = 0.75, alpha = 0.02,
     df = 10, R2 = 0.5, avg=TRUE, llo=NULL, tbv=NULL, ReturnGebv = FALSE)
```

Arguments

y	Numeric vector of response variable (n). NA is not allowed.
gen	Numeric matrix containing the genotypic data. A matrix with n rows of observations and m columns of molecular markers.
df	Hyperprior degrees of freedom of variance components.
R2	Expected R2, used to calculate the prior shape (de los Campos et al. 2013).

Pi	Value between 0 and 1. Expected probability pi of having null effect (or 1-Pi if Pi>0.5).
alpha	Value between 0 and 1. Intensity of L1 variable selection.
D	NULL or numeric vector with length p. Vector of weights for markers.
k	Integer. Folding of a k-fold cross-validation.
n	Integer. Number of cross-validation to perform.
avg	Logical. Return average across CV, or correlations within CV.
llo	NULL or a vector (numeric or factor) with the same length as y. If provided, the cross-validations are performed as Leave a Level Out (LLO). This argument allows the user to predefine the splits. This argument overrides k and n.
tbv	NULL or numeric vector of 'true breeding values' (n) to use to compare cross-validations to. If NULL, the cross-validations will have the phenotypes as prediction target.
ReturnGebv	Logical. If TRUE, it returns a list with the average marker values and fitted values across all cross-validations, in addition to the regular output.

Details

The model for the whole-genome regression is as follows:

$$y = mu + Xb + e$$

where y is the response variable, mu is the intercept, X is the genotypic matrix, b is the effect of an allele substitution (or regression coefficient) and e is the residual term. A k-fold cross-validation for model evaluation is provided by $emCV$.

Value

The EM functions returns a list with the intercept (mu), the regression coefficient (b), the fitted value (hat), and the estimated intraclass-correlation ($h2$).

The function emCV returns the predictive ability of each model, that is, the correlation between the predicted and observed values from k-fold cross-validations repeated n times.

Author(s)

Alencar Xavier

Examples

```
    ## Not run:

data(tpod)
emCV(y,gen,3,3)


## End(Not run)
```

WGR3 (MV) *Multivariate Regression*

Description

Multivariate model to find breeding values.

Usage

```
mkr(Y,K,...)
mrr(Y,X,...)
mrr_float(Y,X,...)
```

Arguments

`Y`	Numeric matrix of observations x trait. `NA` is allowed.
`K`	Numeric matrix containing the relationship matrix.
`X`	Numeric matrix containing the genotyping matrix.
`...`	Arguments to pass to MRR3/MRR3F. See `args(MRR3)`.

Details

Algorithm is described in Xavier and Habier (2022). The model for the ridge regression (mrr) is as follows:

$$Y = Mu + XB + E$$

where Y is a matrix of response variables, Mu represents the intercepts, X is the matrix of genotypic information, B is the matrix of marker effects, and E is the residual matrix.

The model for the kernel regression (mkr) is as follows:

$$Y = Mu + UB + E$$

where Y is a matrix of response variables, Mu represents the intercepts, U is the matrix of Eigenvector of K, b is a vector of regression coefficients and E is the residual matrix.

Algorithm: Residuals are assumed to be independent among traits. Regression coefficients are solved via a multivaraite adaptation of Gauss-Seidel Residual Update. Since version 2.0, the solver of `mrr` is based on the Randomized Gauss-Seidel algorithm. Variance and covariance components are solved with an EM-REML like approach proposed by Schaeffer called Pseudo-Expectation.

Other related implementations:

01) `mkr2X(Y,K1,K2)`: Solves multi-trait kernel regressions with two random effects.

02) `mrr2X(Y,X1,X2)`: Solves multi-trait ridge regressions with two random effects.

03) `MRR3(Y,X,...)`: Extension of mrr with additional parameters.

04) `MRR3F(Y,X,...)`: MRR3 running on float.

05) `mrr_svd(Y,W)`: Solves mrr through the principal components of parameters.

06) `MLM(Y,X,Z,maxit=500,logtol=-8,cores=1)`: Multivariate model with fixed effects.

07) `SEM(Y,Z,...)`: Fits a MegaSEM with both shared- and trait-specific terms.

08) `MEGA(Y,X,npc=-1)`: Toy implementation of MegaLMM, imputing missing with GEBVs.

09) `GSEM(Y,X,npc=-1)`: Toy C++ implementaiton of MegaSEM, jointly fits FA and XB.

10) `ZSEMF(Y,X,npc=0)`: Full-rank MegaSEM, float precision.

11) `YSEMF(Y,X,npc=-1)`: Reduced-rank MegaSEM, float, two-steps approach.

12) `XSEMF(Y,X,npc=0)`: Full-rank MegaSEM, h2 fixed at 0.5, float precision.

In GSEM, XSEMF and MEGA, 'npc' means number of latent spaces if input is above zero, otherwise, 0 means all and -1 means `2*sqrt(ncol(Y))`.

Value

Returns a list with the random effect covariances (`Vb`), residual variances (`Ve`), genetic correlations (`GC`), matrix with marker effects (`b`) or eigenvector effects (if `mkr`), intercepts (`mu`), heritabilities (`h2`), and a matrix with fitted values (`hat`).

NOTE: Numeric stability is a serious concern with multivariate models with large number of response variables, as the covariance matrix is often not invesible. If output is filled with NAs, try using `MRR3` and play with some parameters. For example, one may try adding priors to stabilize variances, e.g., `fit=MRR3(Y,X,df0=20)`.

Author(s)

Alencar Xavier, David Habier

References

Xavier, A and Habier, D. (2022). A new approach fits multivariate genomic prediction models efficiently. GSE, DOI: 10.1186/s12711-022-00730-w

Examples

```
# Load genomic data

data(tpod)
X = CNT(gen)

# Simulate phenotyp

sim = SimY(X)
Y = sim$Y
TBV = sim$tbv

# Fit regression model

test = mrr(Y,X)

# Genetic correlation
```

```
test$GC

# Heritabilies

test$h2

# Accuracy

diag(cor(TBV,test$hat))

# try: demo(multivariates)
```

XTRA 1 *Mixed model solver*

Description

Function to solve univariate mixed models with or without the usage of omic information. This function allows single-step modeling of replicated observations with marker information available through the usage of a linkage function to connect to a whole-genome regression method. Genomic estimated values can be optionally deregressed (no shrinkage) while fitting the model.

Usage

```
mixed(y,random=NULL,fixed=NULL,data=NULL,X=list(),
      alg=emML,maxit=10,Deregress=FALSE,...)
```

Arguments

`y`	Response variable from the data frame containg the dataset.
`random`	Formula. Right-hand side formula of random effects.
`fixed`	Formula. Right-hand side formula of fixed effects.
`data`	Data frame containing the response variable, random and fixed terms.
`X`	List of omic incidence matrix. Row names of these matrices connect the omic information to the levels of the indicated random terms (eg. `X=list("ID"=gen)`).
`alg`	Function. Whole-genome regression algorithm utilized to solve link functions. These include MCMC (`wgr`, `BayesB`, etc) and EM (`emEN`, `emDE`, etc) algorithms. By default, it runs maximum likelihood `emML`.
`maxit`	Integer. Maximum number of iterations.
`Deregress`	Logical. Deregress (unshrink) coefficients while fitting the model?
`...`	Additional arguments to be passed to the whole-genome regression algorithms especified on `alg`.

Details

The model for the whole-genome regression is as follows:

$$y = Xb + Zu + Wa + e$$

where y is the response variable, Xb corresponds to the fixed effect term, Zu corresponds to one or more random effect terms, W is the incidence matrix of terms with omic information and a is omic values by $a = Mg$, where M is the genotypic matrix and g are marker effects. Here, e is the residual term. An example is provided using the data from the NAM package with: `demo(mixedmodel)`.

Alterinative (and updated) implementations have similar syntax:

01) `mm(y,random=NULL,fixed=NULL,data=NULL, M=NULL,bin=FALSE,AM=NULL,it=10,verb=TRUE, FLM=TRUE,wgtM=TRUE,cntM=TRUE,nPc=3)`

02) `mtmixed = function(resp, random=NULL, fixed=NULL, data, X=list(), maxit=10, init=10, regVC=FALSE)`

Value

The function wgr returns a list with Fitness values (`Fitness`) containing observation `obs`, fitted values `hat`, residuals `res`, and fitted values by model term `fits`; Estimated variance components (`VarComp`) containing the variance components per se (`VarComponents`) and variance explained by each model term (`VarExplained`), regression coefficients by model term (`Coefficients`), and the effects of structured terms (`Structure`) containing the marker effects of each model term where markers were provided.

Author(s)

Alencar Xavier

References

Xavier, A. (2019). Efficient Estimation of Marker Effects in Plant Breeding. G3: Genes, Genomes, Genetics, DOI: 10.1534/g3.119.400728

Examples

```
## Not run:
demo(mixedmodel)

## End(Not run)
```

XTRA 2 *Additional tools*

Description

Complementary functions that may help with handling parameters and routine operations.

Details

`emGWA(y,gen)` # Simple MLM for association analysis

`markov(gen,chr=NULL)` # Markovian imputation of genotypes coded as 012

`IMP(X)` # Imputes genotypes with SNP expectation (column average)

`CNT(X)` # Recodes SNPs by centralizing columns in a matrix

`GAU(X)` # Creates Gaussian kernel as exp(-Dist2/mean(Dist2))

`GRM(X,Code012=FALSE)` # Creates additive kinship matrix VanRaden 2008

`SPC(y,blk,row,col,rN=3,cN=1)` # Spatial covariate

`SPM(blk,row,col,rN=3,cN=1)` # Spatial design matrix

`SibZ(id,p1,p2)` # Pedigree design matrix compatible to regression methods

`Hmat(ped,gen=NULL)` # Kinship combining pedigree and genomics

`EigenGRM(X, centralizeZ = TRUE, cores = 1)` # GRM using Eigen library

`EigenARC(X, centralizeZ = TRUE, cores = 1)` # ArcCosine kernel

`EigenGAU(X, phi = 1.0, cores = 1)` # Gaussian kernel using Eigen library

`EigenCNT(X, cores = 1)` # Center SNPs without missing Eigen library

`EigenEVD(A, cores = 1)` # Eigendecomposition from Eigen library

`EigenBDCSVD(X, cores = 1)` # BDC single value composition from Eigen

`EigenJacobiSVD(X, cores = 1)` # Jacobi single value composition from Eigen

`EigenAcc(X1, X2, h2 = 0.5, cores = 1)` # Deterministic accuracy X1 -> X2 via V

`AccByC(X1, X2, h2 = 0.5, cores = 1)` # Deterministic accuracy X1 -> X2 via C

`EigenArcZ(Zfndr, Zsamp, cores = 1)` # Reduced rank ArcCos kernel PCs with founder rotation

`EigenGauZ(Zfndr, Zsamp, phi=1, cores = 1)` # Reduced rank Gaussian kernel PCs with founder rotation

`K2X(K, MinEV = 1e-8, cores = 1)` # Reparametrize kernel to PCs to run regression models

`SimY(Z,k=5,h2=0.5,GC=0.5,seed=123,unbalanced=FALSE,PercMiss=0,BlkMiss=FALSE)` # Simulate phenotypes

`SimZ(ind=500,snp=500,chr=2,F2=TRUE,rec=0.01)` # Simulate genome

`SimGC(k=50,...)` # Simulate genetic correlation matrix

`MvSimY(Ufndr,Zfndr,Zsamp,GxY,GxL,H2plot,nLoc=20,Seed=123)` # Simulate phenotypes given founders

Author(s)

Alencar Xavier

Index