Supplementary material

Alencar Xavier and David Habier

June 17, 2022

1 Deterministic calculations to study estimators of variance components using *Tilde-Hat* or *Pseudo-Expectation*

A deterministic tool was coded in R to demonstrate that variances and covariances estimated by equations (5) and (6) of the main manuscript are unbiased for different heritabilities and genetic correlations. More specifically, the objective is to derive expected values of the estimators for variances and covariances and compare them to the true, given parameters.

The calculations are based on the bivariate model

$$\begin{aligned}\begin{bmatrix}\boldsymbol{y}_k\\ \boldsymbol{y}_{k'}\end{bmatrix} &= \begin{bmatrix}\mathbf{X}_k & 0\\ 0 & \mathbf{X}_{k'}\end{bmatrix}\begin{bmatrix}\boldsymbol{b}_k\\ \boldsymbol{b}_{k'}\end{bmatrix} + \begin{bmatrix}\mathbf{Z}_k & 0\\ 0 & \mathbf{Z}_{k'}\end{bmatrix}\begin{bmatrix}\boldsymbol{\beta}_k\\ \boldsymbol{\beta}_{k'}\end{bmatrix} + \begin{bmatrix}\boldsymbol{e}_k\\ \boldsymbol{e}_{k'}\end{bmatrix}\\ \boldsymbol{y} &= \mathbf{X}\boldsymbol{b} + \mathbf{Z}\boldsymbol{\beta} + \boldsymbol{e},\end{aligned}$$

where $\boldsymbol{y}_k$ and $\boldsymbol{y}_{k'}$ are vectors of phenotypes, $\mathbf{X}_k$ and $\mathbf{X}_{k'}$ are incidence matrices for fixed effects, $\mathbf{Z}_k$ and $\mathbf{Z}_{k'}$ are matrices of centered marker scores, $\boldsymbol{\beta}_k$ and $\boldsymbol{\beta}_{k'}$ are vectors of random marker effects, and $\boldsymbol{e}_k$ and $\boldsymbol{e}_{k'}$ are vectors of random residual effects, all for the two environments k and k', respectively. The expected value of $\boldsymbol{y}$ is $\mathbf{X}\boldsymbol{b}$, the variances of $\boldsymbol{\beta}$ and $\boldsymbol{e}$ are $Var(\boldsymbol{\beta}) = \mathbf{G} = \boldsymbol{\Sigma}_\beta \otimes \mathbf{I}_m$ and $Var(\boldsymbol{e}) = \mathbf{R} = \mathbf{I}_k\sigma^2_k \oplus \mathbf{I}_{k'}\sigma^2_{k'}$, where $\boldsymbol{\Sigma}_\beta$ is the variance-covariance matrix of marker effects, $\mathbf{I}_m$, $\mathbf{I}_k$ and $\mathbf{I}_{k'}$ are identity matrices, $\sigma^2_{e_k}$ and $\sigma^2_{e_{k'}}$ are residual variances for environments k and k', and $\oplus$ is the direct matrix sum. The variance of $\boldsymbol{y}$ is $\mathbf{V} = \mathbf{Z}\mathbf{G}\mathbf{Z}' + \mathbf{R}$. The elements of $\boldsymbol{\Sigma}_\beta$ are calculated as

$$\begin{aligned}\boldsymbol{\Sigma}_\beta &= \begin{bmatrix}\sigma^2_{\beta_k} & \sigma_{\beta_{kk'}}\\ \sigma_{\beta_{kk'}} & \sigma^2_{\beta_{k'}}\end{bmatrix}\\ &= \frac{1}{m}\begin{bmatrix}\sigma^2_{g_k} & \sigma_{g_{kk'}}\\ \sigma_{g_{kk'}} & \sigma^2_{g_{k'}}\end{bmatrix},\end{aligned}$$

where m is the number of markers, $\sigma^2_{g_k}$ and $\sigma^2_{g_{k'}}$ are the additive-genetic variances of environments k and k', and $\sigma_{g_{kk'}}$ is the additive-genetic covariance between the two environments. These additive-genetic parameters are provided as input to the deterministic calculations and they are compared to the expected values of their estimators to show that these are unbiased.

Marker effects are estimated by Best Linear Unbiased Prediction [1] as

$$\hat{\boldsymbol{\beta}} = Cov(\boldsymbol{\beta}, \boldsymbol{y}')\mathbf{P}\boldsymbol{y},$$

where $\mathbf{P} = \mathbf{V}^{-1}[\mathbf{I} - \mathbf{X}(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X})^{-1}\mathbf{X}'\mathbf{V}^{-1}]$ and

$$\begin{aligned} Cov(\boldsymbol{\beta}, \boldsymbol{y}') &= \begin{bmatrix} \sigma^2_{\beta_k}\mathbf{Z}'_k & \sigma_{\beta_{kk'}}\mathbf{Z}'_{k'} \\ \sigma_{\beta_{kk'}}\mathbf{Z}'_k & \sigma^2_{\beta_{k'}}\mathbf{Z}'_{k'} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{V}_{\beta_k y} \\ \mathbf{V}_{\beta_{k'} y} \end{bmatrix}. \end{aligned}$$

The expected value of $\hat{\boldsymbol{\beta}} = \mathbf{0}$ as $\mathbf{PX} = \mathbf{0}$. For estimating variances and covariances of marker effects $\tilde{\boldsymbol{\beta}}_k = \mathbf{D}_k^{-1}\mathbf{Z}'_k\mathbf{M}_k\boldsymbol{y}_k$ and $\tilde{\boldsymbol{\beta}}_{k'} = \mathbf{D}_{k'}^{-1}\mathbf{Z}'_{k'}\mathbf{M}_{k'}\boldsymbol{y}_{k'}$ are used in

$$\hat{\sigma}^2_{\beta_k} = \frac{\tilde{\boldsymbol{\beta}}'_k\hat{\boldsymbol{\beta}}_k}{tr(\mathbf{D}_k^{-1}\mathbf{Z}'_k\mathbf{M}_k\mathbf{Z}_k)},$$

and

$$\hat{\sigma}_{\beta_{kk'}} = \frac{\tilde{\boldsymbol{\beta}}'_k\hat{\boldsymbol{\beta}}_{k'} + \tilde{\boldsymbol{\beta}}'_{k'}\hat{\boldsymbol{\beta}}_k}{tr(\mathbf{D}_k^{-1}\mathbf{Z}'_k\mathbf{M}_k\mathbf{Z}_k) + tr(\mathbf{D}_{k'}^{-1}\mathbf{Z}'_{k'}\mathbf{M}_{k'}\mathbf{Z}_{k'})},$$

respectively. The expected value of $\tilde{\boldsymbol{\beta}}'_k\hat{\boldsymbol{\beta}}_k$ [2] is

$$\begin{aligned} E(\tilde{\boldsymbol{\beta}}'_k\hat{\boldsymbol{\beta}}_k) &= tr(Cov(\tilde{\boldsymbol{\beta}}_k, \hat{\boldsymbol{\beta}}'_k)) + E(\tilde{\boldsymbol{\beta}}_k)'E(\hat{\boldsymbol{\beta}}_k) \\ &= tr(\mathbf{D}_k^{-1}\mathbf{Z}'_k\mathbf{M}_k Cov(\boldsymbol{y}_k, \boldsymbol{y}')\mathbf{P}\mathbf{V}'_{\beta_k y}) \end{aligned}$$

and the expected values of $\tilde{\boldsymbol{\beta}}'_k\hat{\boldsymbol{\beta}}_{k'}$ and $\tilde{\boldsymbol{\beta}}'_{k'}\hat{\boldsymbol{\beta}}_k$ are

$$E(\tilde{\boldsymbol{\beta}}'_k\hat{\boldsymbol{\beta}}_{k'}) = tr(\mathbf{D}_k^{-1}\mathbf{Z}'_k\mathbf{M}_k Cov(\boldsymbol{y}_k, \boldsymbol{y}')\mathbf{P}\mathbf{V}'_{\beta_{k'} y})$$

and

$$E(\tilde{\boldsymbol{\beta}}'_{k'}\hat{\boldsymbol{\beta}}_k) = tr(\mathbf{D}_{k'}^{-1}\mathbf{Z}'_{k'}\mathbf{M}_{k'} Cov(\boldsymbol{y}_{k'}, \boldsymbol{y}')\mathbf{P}\mathbf{V}'_{\beta_k y}),$$

respectively. The matrices $Cov(\boldsymbol{y}_k, \boldsymbol{y}')$ and $Cov(\boldsymbol{y}_{k'}, \boldsymbol{y}')$ are row-partitions of $\mathbf{V}$. The expected values of $\hat{\sigma}^2_{\beta_k}$ and $\hat{\sigma}_{\beta_{kk'}}$ are

$$E(\hat{\sigma}^2_{\beta_k}) = \frac{tr(\mathbf{D}_k^{-1}\mathbf{Z}'_k\mathbf{M}_k Cov(\boldsymbol{y}_k, \boldsymbol{y}')\mathbf{P}\mathbf{V}'_{\beta_k y})}{tr(\mathbf{D}_k^{-1}\mathbf{Z}'_k\mathbf{M}_k\mathbf{Z}_k)}, \tag{1}$$

and

$$E(\hat{\sigma}_{\beta_{kk'}}) = \frac{tr(\mathbf{D}_k^{-1}\mathbf{Z}'_k\mathbf{M}_k Cov(\boldsymbol{y}_k, \boldsymbol{y}')\mathbf{P}\mathbf{V}'_{\beta_{k'} y}) + tr(\mathbf{D}_{k'}^{-1}\mathbf{Z}'_{k'}\mathbf{M}_{k'} Cov(\boldsymbol{y}_{k'}, \boldsymbol{y}')\mathbf{P}\mathbf{V}'_{\beta_k y})}{tr(\mathbf{D}_k^{-1}\mathbf{Z}'_k\mathbf{M}_k\mathbf{Z}_k) + tr(\mathbf{D}_{k'}^{-1}\mathbf{Z}'_{k'}\mathbf{M}_{k'}\mathbf{Z}_{k'})}, \tag{2}$$

respectively. The expected additive-genetic variance and covariance are

$$\begin{aligned} E(\hat{\sigma}^2_{g_k}) &= m \cdot E(\hat{\sigma}^2_{\beta_k}) \\ E(\hat{\sigma}^2_{g_{k'}}) &= m \cdot E(\hat{\sigma}^2_{\beta_{k'}}) \\ E(\hat{\sigma}_{g_{kk'}}) &= m \cdot E(\hat{\sigma}_{\beta_{kk'}}). \end{aligned}$$

and biases are

$$\begin{aligned} \text{Bias}(\hat{\sigma}^2_{g_k}) &= E(\hat{\sigma}^2_{g_k}) - \sigma^2_{g_k} \\ \text{Bias}(\hat{\sigma}^2_{g_{k'}}) &= E(\hat{\sigma}^2_{g_{k'}}) - \sigma^2_{g_{k'}} \\ \text{Bias}(\hat{\sigma}_{g_{kk'}}) &= E(\hat{\sigma}_{g_{kk'}}) - \sigma_{g_{kk'}}. \end{aligned}$$

In the R code, shown in section 5, $\mathbf{D} = \mathbf{I}_m$. Biases are zero for different additive-genetic variances and covariances as well as residual variances. Thus, by inspection, $\mathbf{M}_k Cov(\boldsymbol{y}_k, \boldsymbol{y}')\mathbf{P} = [\mathbf{M}_k \vdots \mathbf{0}]$ and $\mathbf{M}_{k'} Cov(\boldsymbol{y}_{k'}, \boldsymbol{y}')\mathbf{P} = [\mathbf{0} \vdots \mathbf{M}_{k'}]$, so that equations (1) and (2) reduce to $E(\hat{\sigma}^2_{\beta_k}) = \sigma^2_{\beta_k}$ and $E(\hat{\sigma}_{\beta_{kk'}}) = \sigma_{\beta_{kk'}}$, respectively.

References

Searle, S.R., Casella, G., McCulloch, C.E.: Mixed model prediction (blup). In: Variance Components, pp. 269–277. John Wiley and Sons, Inc., New York (1992). doi:10.1002/9780470316856.ch7

Searl, S.R.: Linear Models, p. 65. John Wiley Sons, Inc., New York (1971)

2 Scenario 1 with more environments

Figure 1 of the main manuscript showed that PEGS and THGS were not better than UV-THGS in scenario 1 when both heritability and genetic correlations were low. With increasing number of environments, and thereby phenotypes per individual, PEGS and THGS had a higher accuracy than UV-THGS, and their accuracy approaches that of REML (Figure 1).

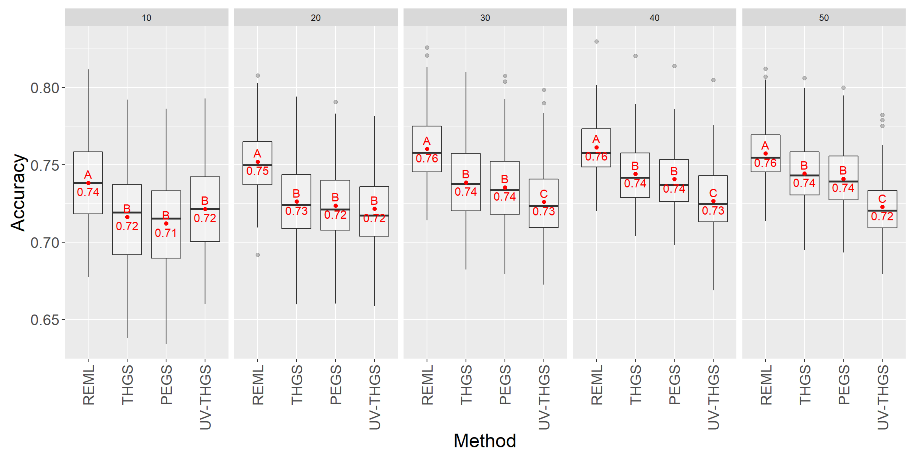

Figure 1: Accuracy of GEBVs in scenario 1 with an increasing number of environments (10, 20, 30, 40 and 50), heritability $h^2 = 0.2$ and low genetic correlations $(0.2 - 0.4)$.

3 Estimated covariances for different degrees of balanced data

REML results were not provided for scenario 2 because ASREML and AIREMLF90 were not suited for the estimation of covariance components when there was no overlap of individuals and environments. In Figure 2, the data from scenario 2 were adapted to demonstrate that an overlapping of individuals and environments is necessary for REML with ASREML and AIREMLF90 to adequately estimate covariance components. This pitfall was not a problem for PEGS and THGS.

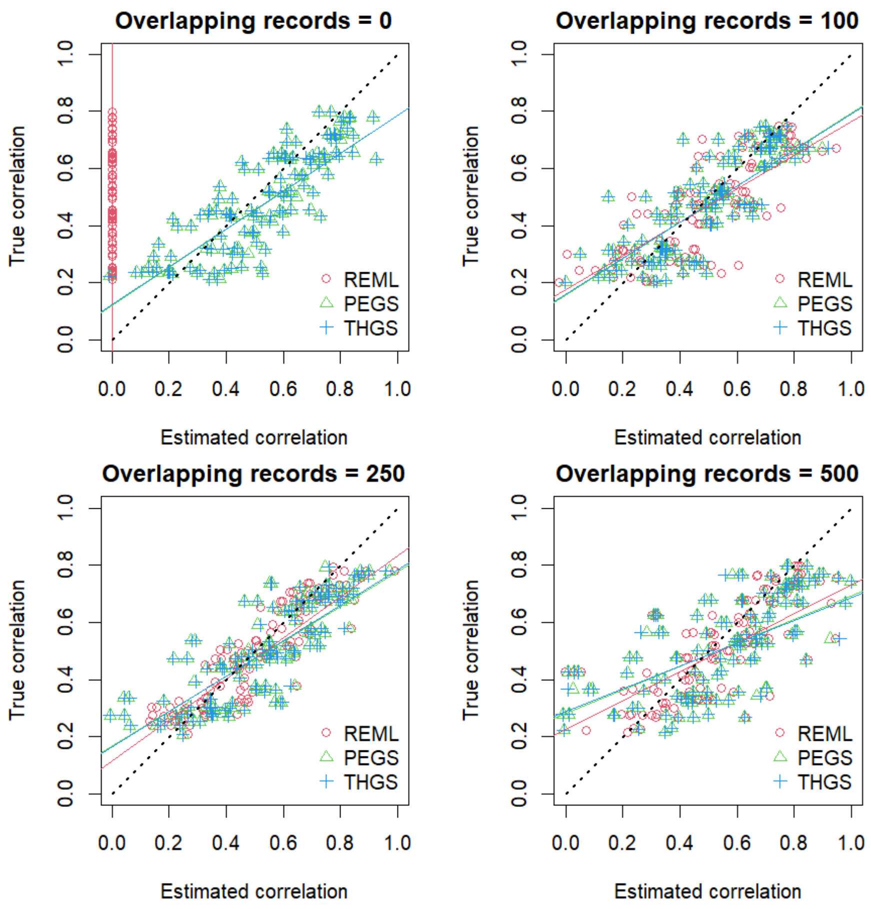

Figure 2: Scatter plot between true and estimated genetic correlations using the soybean dataset with varying number of overlapping individuals across environments.

4 Summary of scenario 2 and balanced case

Table 1: Accuracy of GEBVs, regression of TBV on GEBV (Slope) and bias of estimated heritabilities ($\hat{h}^2$) and genetic correlations (GC) using the soybean dataset for unbalanced (scenario 2) and balanced datasets (all response variables observed in all individuals) with varying number of observations per environment (Obs/Env). Based on 100 replicates of the simulation.

Scenario	Obs/Env	Method	Accuracy	Slope	Bias $\hat{h}^2$	Bias of GC
Unbalanced	514	PEGS	0.88 (0.02)	1.00 (0.03)	-0.01 (0.03)	0.00 (0.06)
Unbalanced	514	THGS	0.88 (0.02)	1.00 (0.03)	-0.01 (0.03)	-0.01 (0.06)
Unbalanced	514	UV-THGS	0.86 (0.03)	1.03 (0.05)	-0.01 (0.03)	-
Balanced	5,142	REML	0.97 (0.00)	1.00 (0.01)	0.00 (0.01)	0.00 (0.03)
Balanced	5,142	PEGS	0.97 (0.00)	1.00 (0.01)	-0.01 (0.02)	0.00 (0.05)
Balanced	5,142	THGS	0.97 (0.00)	1.00 (0.01)	0.00 (0.02)	0.00 (0.03)
Balanced	5,142	UV-THGS	0.96 (0.01)	1.00 (0.01)	0.00 (0.02)	-

5 R code demonstrating unbiasedness of PEGS

The **input** variables are the marker scores, the additive-genetic variance of environment k, the additive-genetic variance of environment k', the additive-genetic correlation between the two environments, the residual variance of environment k, the residual variance of environment k', the number of individuals in environment k, and the number of individuals in environment k'. The marker scores are obtained from the R package SoyNAM, and matrices $\mathbf{Z}_k$ and $\mathbf{Z}_{k'}$ are populated with random samples of individuals from that dataset. The **output** is a vector that contains the heritabilities of the two environments, true additive-genetic variances and covariances, and the expected values and biases of their estimates.

```
# Function to estimate PEGS bias
#install.packages("SoyNAM")
require(Matrix)
bias = function(x,                # Genotypic information matrix
                sigma2_gk = 1,    # Additive genetic variance for environment 1
                sigma2_gkp = 2,   # Additive genetic variance for environment 2
                corr_gkkp = 0.7,  # Additive genetic correlation
                sigma2_ek = 5,    # Residual variance for environment 1
                sigma2_ekp = 7, # Residual variance for environment 2
                nk=100,           # Number of individuals in environment 1
                nkp=110){         # Number of individuals in environment 2
  sigma_gkkp = corr_gkkp*sqrt(sigma2_gk*sigma2_gkp)
  n   = nk+nkp # Total number of individuals
  s = sample(nk+nkp,replace = F) # Sample random individuals
  Zk = x[s[1:nk],] # Marker scores of environment k
  Zkp = x[s[(nk+1):n],] # Marker scores of environment kp
  m = ncol(Zk) # Number of markers
  Zk = apply(Zk,2,function(x) x-mean(x)) # Centering of marker scores
  Zkp = apply(Zkp,2,function(x) x-mean(x)) # Centering of marker scores
  sigma2_betak  = sigma2_gk/m  # Marker variance for environment k
  sigma2_betakp = sigma2_gkp/m # Marker variance for environment kp
  sigma_betakkp = sigma_gkkp/m # Marker covariance between environments k and kp
  ZGZ = tcrossprod(bdiag(list(sqrt(sigma2_betak)*Zk,sqrt(sigma2_betakp)*Zkp)))
  ZGZ = as.matrix(ZGZ); ZGZ[1:nk,(nk+1):n] = sigma_betakkp*Zk%*%t(Zkp);
  ZGZ[(nk+1):n,1:nk] = sigma_betakkp*Zkp%*%t(Zk)
  R = bdiag(list(Diagonal(nk,sigma2_ek),Diagonal(nkp,sigma2_ekp)))
  V = ZGZ + as.matrix(R); invV = solve(V)
  X = as.matrix(Matrix::bdiag(list(matrix(1,nk),matrix(1,nkp))))
  P = invV%*%(diag(n)-X%*%solve(t(X)%*%invV%*%X)%*%t(X)%*%invV)
  Mk  = diag(nk)-matrix(1/nk,nk,nk)
  Mkp = diag(nkp)-matrix(1/nkp,nkp,nkp)
  Vky  = cbind(sigma2_betak*t(Zk),sigma_betakkp*t(Zkp))
  Vkpy = cbind(sigma_betakkp*t(Zk),sigma2_betakp*t(Zkp))
```

```r
  trZkMkZk = sum(diag(t(Zk)%*%Mk%*%Zk)) #tr(Zk*Mk*Zk)
  trZkpMkpZkp = sum(diag(t(Zkp)%*%Mkp%*%Zkp)) #tr(Zkp*Mkp*Zkp)
  ExpVal_tildeBetak_hatBeta_k  = sum(diag(t(Zk)%*%Mk%*%V[1:nk,1:n]%*%P%*%t(Vky)))
  ExpVal_tildeBetak_hatBeta_kp = sum(diag(t(Zk)%*%Mk%*%V[1:nk,1:n]%*%P%*%t(Vkpy)))
  ExpVal_tildeBetakp_hatBeta_k = sum(diag(t(Zkp)%*%Mkp%*%V[(nk+1):n,1:n]%*%P%*%t(Vky)))
  ExpVal_tildeBetakp_hatBeta_kp = sum(diag(t(Zkp)%*%Mkp%*%V[(nk+1):n,1:n]%*%P%*%t(Vkpy)))
  ExpVal_sigma2_gk_hat  = m*(ExpVal_tildeBetak_hatBeta_k)/trZkMkZk
  ExpVal_sigma2_gkp_hat  = m*(ExpVal_tildeBetakp_hatBeta_kp)/trZkpMkpZkp
  covSS = m*(ExpVal_tildeBetak_hatBeta_kp+ExpVal_tildeBetakp_hatBeta_k)
  ExpVal_sigma_gkkp_hat = covSS/(trZkMkZk+trZkpMkpZkp)
  bias_sigma2_gk_hat  = ExpVal_sigma2_gk_hat - sigma2_gk
  bias_sigma2_gkp_hat  = ExpVal_sigma2_gkp_hat - sigma2_gkp
  bias_sigma_gkkp_hat = ExpVal_sigma_gkkp_hat - sigma_gkkp
  h2_k = sigma2_gk/(sigma2_gk+sigma2_ek)
  h2_kp = sigma2_gkp/(sigma2_gkp+sigma2_ekp)
  ExpVal_Corr = ExpVal_sigma_gkkp_hat / sqrt(ExpVal_sigma2_gk_hat*ExpVal_sigma2_gkp_hat)
  bias_corr = ExpVal_Corr - corr_gkkp
  out = c(H2_k = h2_k, H2_kp = h2_kp,
          Sigma2_gk = sigma2_gk, ExpVal_sigma2_gk_hat = ExpVal_sigma2_gk_hat,
          Bias_sigma2_gk_hat = bias_sigma2_gk_hat,
          Sigma2_gkp = sigma2_gkp, ExpVal_sigma2_gkp_hat = ExpVal_sigma2_gkp_hat,
          Bias_sigma2_gkp_hat = bias_sigma2_gkp_hat,
          Corr_gkkp = corr_gkkp, ExpVal_Corr_gkkp = ExpVal_Corr,
          Bias_Corr_gkkp = bias_corr,
          Sigma_gkkp = sigma_gkkp, ExpVal_sigma_gkkp_hat = ExpVal_sigma_gkkp_hat,
          Bias_sigma_gkkp_hat= bias_sigma_gkkp_hat)
  return(round(out,4))}

# Get soybean data
x = SoyNAM::ENV()$gen

# Run example
bias(x)
```

6 An R implementation of PEGS

The code below can be pasted and loaded in using the Rcpp package in R. It is a function that takes two inputs: Y and X. The input Y is numeric matrix of phenoypes where columns corresponds to traits and rows are individuals, where missing values are allowed. The input X is numeric matrix of genomic information where columns corresponds to markers and rows are individuals, where missing values are not allowed.

```
// [[Rcpp::depends(RcppEigen)]]
#include <RcppEigen.h>
#include <iostream>
#include <random>

// [[Rcpp::export]]
SEXP PEGS(Eigen::MatrixXd Y, Eigen::MatrixXd X){
 int J, maxit = 1000, k = Y.cols(), n0 = Y.rows(), p = X.cols();
 // Incidence matrix Z
 Eigen::MatrixXd Z(n0,k);
 for(int i=0; i<n0; i++){ for(int j=0; j<k; j++){
  if(std::isnan(Y(i,j))){Z(i,j) = 0.0; Y(i,j) = 0.0;}else{ Z(i,j) = 1.0;}}}
 // Count observations per trait
 Eigen::VectorXd n = Z.colwise().sum();
 Eigen::VectorXd iN = n.array().inverse();
 // Centralize y
 Eigen::VectorXd mu=Y.colwise().sum(); mu=mu.array()*iN.array(); Eigen::MatrixXd y(n0,k);
 for(int i=0; i<k; i++){ y.col(i) = (Y.col(i).array()-mu(i)).array()*Z.col(i).array();}
 // Sum of squares of X abd Tr(XSX)
 Eigen::MatrixXd XX(p,k), XSX(p,k);
 for(int i=0; i<p; i++){ XX.row(i) = X.col(i).array().square().matrix().transpose()*Z;}
 for(int i=0; i<p; i++){ XSX.row(i) = XX.row(i).transpose().array()*iN.array() -
  ((X.col(i).transpose()*Z).transpose().array()*iN.array()).square();}
 Eigen::VectorXd MSx = XSX.colwise().sum();
 Eigen::VectorXd TrXSX = n.array()*MSx.array();
 // Variances
 iN = (n.array()-1).inverse();
 Eigen::VectorXd h2(k), vy = y.colwise().squaredNorm(); vy = vy.array()*iN.array();
 Eigen::VectorXd ve = vy*0.5; Eigen::VectorXd iVe = ve.array().inverse();
 Eigen::MatrixXd vb(k,k), iG(k,k), TildeHat(k,k), GC(k,k);
 vb = (ve.array()/MSx.array()).matrix().asDiagonal(); iG = vb.inverse();
 // Initialize coefficient matrices;
 Eigen::MatrixXd tilde = X.transpose()*y;
 Eigen::MatrixXd LHS(k,k), b=Eigen::MatrixXd::Zero(p,k), e(n0,k); e=y*1.0;
 Eigen::VectorXd RHS(k), b0(k), b1(k);
 // RGS
 std::vector<int> RGSvec(p);  for(int j=0; j<p; j++){RGSvec[j]=j;}
 std::random_device rd;  std::mt19937 g(rd());
 // Convergence control
 double deflate = 1.0, deflateMax = 0.75, cnv = 10.0, logtol = -10.0;
 Eigen::MatrixXd beta0(p,k), A(k,k); int numit = 0;
 // Loop
 while(numit<maxit){
  beta0 = b*1.0; // Store current values
  std::shuffle(RGSvec.begin(), RGSvec.end(), g); // Randomization
  for(int j=0; j<p; j++){ // Gauss-Seidel loop
   J = RGSvec[j]; b0 = b.row(J)*1.0;
   LHS = iG;  LHS.diagonal() += (XX.row(J).transpose().array() * iVe.array()).matrix();
   RHS = (X.col(J).transpose()*e).array() + XX.row(J).array()*b0.transpose().array();
   RHS = RHS.array() *iVe.array();
   b1 = LHS.llt().solve(RHS); b.row(J) = b1; // Update coefficient
   // Update residuals
   e = (e-(X.col(J)*(b1-b0).transpose()).cwiseProduct(Z)).matrix();}
  // Estimate residual variances
  ve = (e.cwiseProduct(y)).colwise().sum();
  ve = ve.array()*iN.array(); iVe = ve.array().inverse();
  // Genetic variance
  TildeHat = b.transpose()*tilde;
  for(int i=0; i<k; i++){for(int j=0; j<k; j++){
   if(i==j){ vb(i,i) = TildeHat(i,i)/TrXSX(i); }else{
   vb(i,j) = (TildeHat(i,j)+TildeHat(j,i))/(TrXSX(i)+TrXSX(j));}}}
  // Bending
  A = vb*1.0;
  for(int i=0; i<k; i++){for(int j=0; j<k; j++){ if(i!=j){A(i,j) *= deflate;}}}
  while(A.llt().info()==Eigen::NumericalIssue && deflate>deflateMax){
    Rcpp::Rcout << "Bend at "<< numit << "\n";  deflate = deflate - 0.01;
    for(int i=0;i<k;i++){for(int j=0; j<k; j++){if(i!=j){A(i,j) = vb(i,j)*deflate;}}}}
  iG = A.inverse();
  // Print status
  cnv = log10((beta0.array()-b.array()).square().sum());  ++numit;
  if(numit%100==0){Rcpp::Rcout<<"Iter: "<< numit << " |Conv: "<< cnv << "\n";}
  if(cnv<logtol){break;}}
 // Fitting the model, genetic correlations
 h2 = 1 - ve.array()/vy.array(); Eigen::MatrixXd hat = X * b;
 for(int i=0; i<k; i++){hat.col(i) = hat.col(i).array() + mu(i);}
 for(int i=0;i<k;i++){for(int j=0;j<k;j++){GC(i,j)=vb(i,j)/(sqrt(vb(i,i)*vb(j,j)));}}
 // Output
 return Rcpp::List::create(Rcpp::Named("mu")=mu, Rcpp::Named("b")=b,
                           Rcpp::Named("hat")=hat, Rcpp::Named("h2")=h2,
                           Rcpp::Named("GC")=GC, Rcpp::Named("bend")=deflate,
                           Rcpp::Named("cnv")=cnv);}
```