Package 'GenEval'

August 6, 2021

Type Package

Title A package to simulate and evaluate genomic data

Version 1.0.0

Author Beatriz Cuyabano

Maintainer Beatriz Cuyabano <beatriz.castro-dias-cuyabano@inrae.fr>

Description GenEval is a package to simulate and evaluate genomic data.
Genotypes can be simulated either in complete linkage equilibrium (LE) or in linkage disequilibrium (LD). Phenotypes can be simulated as both continuous or discrete.

Depends R (>= 3.1.0),
base,
MASS,
stats,
Matrix,
matrixStats,
kinship2,
MM4LMM,
boot,
simpleboot,
car,
rootSolve

Imports base,
MASS,
stats,
Matrix,
matrixStats,
kinship2,
MM4LMM,
boot,
simpleboot,
car,
rootSolve

License None

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

R topics documented:

BLUPsolve . . . 2
ConfIntMeanBoot . . . 4
ConfIntMeanNorm . . . 4
dMP . . . 5
Gkappa . . . 5
LimPredAcc . . . 7
MatrixInv . . . 8
mkGRM . . . 9
mkPED . . . 10
plotMP . . . 10
pMP . . . 11
popBreed . . . 11
popCor . . . 12
REMLsolve . . . 12
simGeno . . . 13
simPheno . . . 14
simPopInfo . . . 16

Index **17**

`BLUPsolve` *BLUP*

Description

This function performs BLUP in a mixed model. If not provided, the function will estimate the variance components using REML.

Usage

```
BLUPsolve(
  y,
  X = NULL,
  FixedEff = NULL,
  Z = NULL,
  VarMat = NULL,
  R = NULL,
  VarComp = NULL,
  TestGroup = NULL,
  verbose = TRUE
)
```

Arguments

`y`	The response variable.
`X`	A matrix of fixed effects.
`FixedEff`	The effects of the fixed effects, if previously estimated or provided.
`Z`	The design matrix (or list of matrices) of the random effects. If all design matrices are an identity, this argument does not need to be provided.

VarMat	The variance-covariance matrix (or list of matrices) of the random effects.
R	A variance-covariance matrix for the residuals, if they are not independent and identically distributed (i.e. if R != In).
VarComp	The variance components of each random effect, including the residuals. If VarComp=NULL, the function will estimate these variances using REML.
TestGroup	The index of observations from a test population. This will make the function perform prediction over these observations.
verbose	Logical: should steps of the co,putation be printed on screen?

Value

FixedEff The estimated fixed effects.

VarComp The estimated variance components for all random effects, and for the residual.

RandomEff The predicted random effects and their reliabilities.

modelFit A summary of the fit from the linear mixed model.

modelPred A summary of the prediction from the linear mixed model (when TestGroup is provided).

See Also

REMLsolve

Examples

```
## SINGLE COMPONENT WITH FIXED EFFECTS ##
# simulate genotypes
M <- simGeno(n=500,AlleleFreq=runif(100,0.05,0.5))
# simulate phenotypes
pheno <- simPheno(M,0.6,Vy=10)
# add fixed effects
X <- cbind(rep(1,nrow(M)),runif(nrow(M)),rbinom(nrow(M),1,0.5))
b <- c(5,-1,2)
y <- pheno$y + as.numeric(X%*%b)
# REML
G <- mkGRM(M)
tmp <- REMLsolve(y=y,X=X,VarMat=G)
tmp
# BLUP
aux <- BLUPsolve(y=y,X=X,VarMat=G,VarComp=tmp$VarComp)
str(aux)
# BLUP estimating variance components
aux <- BLUPsolve(y=y,X=X,VarMat=G)
str(aux)

## TWO VARIANCE COMPONENTS ##
# simulate genotypes
M <- simGeno(n=500,AlleleFreq=runif(100,0.05,0.5))
# simulate phenotypes
pheno <- simPheno(list(M[,1:70],M[,71:100]),c(0.4,0.2),Vy=10)
# BLUP using GRMs
G <- list(mkGRM(M[,1:70]),mkGRM(M[,71:100]))
aux <- BLUPsolve(y=pheno$y,VarMat=G)
str(aux)
```

```
# prediction
aux <- BLUPsolve(y=pheno$y,VarMat=G,TestGroup=401:500)
str(aux)
# BLUP using genotype matrices
aux <- BLUPsolve(y=pheno$y,VarMat=list(diag(1,70),diag(1,30)),Z=list(M[,1:70],M[,71:100]))
str(aux)
# prediction
Vlist <- list(diag(1,70),diag(1,30))
Zlist <- list(M[,1:70],M[,71:100])
aux <- BLUPsolve(y=pheno$y,VarMat=Vlist,Z=Zlist,TestGroup=401:500)
str(aux)
```

`ConfIntMeanBoot` *Bootstrap confidence interval*

Description

This function calculates the bootstrap confidence interval for the mean of a set of data.

Usage

```
ConfIntMeanBoot(x, levels = NULL, conf = 0.95, n.rep = 10^4)
```

Arguments

`x`	A vector containing values to calculate confidence interval.
`levels`	A vector containing the levels (factor) from the data, to calculate separate confidence intervals.
`n.rep`	The number of bootstrap replicates.
`p`	The confidence level of the interval.

Value

A data frame with the confidence interals.

`ConfIntMeanNorm` *Confidence interval for normal distributed data*

Description

This function calculates the confidence interval for the mean of a set of normal distributed data.

Usage

```
ConfIntMeanNorm(x, levels = NULL, p = 0.95)
```

Arguments

x	A vector containing values to calculate confidence interval.
levels	A vector containing the levels (factor) from the data, to calculate separate confidence intervals.
p	The confidence level of the interval.

Value

A data frame with the confidence interals.

---

dMP *The Marchenko-Pastur density*

Description

Density function of the Marchenko-Pastur distribution with rate parameter equal to a and variance equal to s2.

Usage

```
dMP(x, a, s2)
```

Arguments

x	A vectors of quantiles.
a	Rate parameter (n/m = #observations/#variables).
s2	Variance parameter.

Value

The density.

Gkappa *Kappa statistics*

Description

This function calculates the kappa statistics, to compare two variance-covariance matrices (G1 and G2) defined for the same group of individuals. Kappa is computed using the eigen-decomposition of the matrices, G=ULU'} in which U and L are the eigen-vectors and eigen-values,respectively. \code{Kmat = U2'(G1)U2, and kappa=diag(Kmat). Finally, to compare G1 and G2, it is enough to plot lambda against kappa. The more linear their relationship, the more equivalent matrices G1 and G2 are, in terms of variance structure.

Usage

```
Gkappa(G1, G2 = NULL, U2 = NULL, L2 = NULL, Kmat = FALSE)
```

Arguments

G1	The Reference matrix.
G2	The matrix which we want to compare with G1.
U2	The eigen-vectors of G1. If not provided, the function will obtain them internally.
L2	The eigen-values of G2. If not provided, the function will obtain them internally.
Kmat	Should the entire Kappa matrix be returned? Default is Kmat=FALSE.

Value

lambda The eigen-values of G2.

kappa The kappa statistics that compare G2 to G1.

Kmat The whole Kappa matrix.

Examples

```
# simulate uncorrelated genotypes
M <- simGeno(n=100,AlleleFreq=rep(0.5,200))
# three GRMs from the simulated genotypes
# the first with 100 SNPs
G1 <- mkGRM(M[,1:100])
# the second with 100 SNPs that don't overlap those from the first
G2 <- mkGRM(M[,101:200])
# the third with all SNPs (100 SNPs overlap with those from the first matrix)
G3 <- mkGRM(M)
#----------------------------------------------------------------------#
# compare GRMs using kappa
# verify how that is different from directly comparing the values
# verify how that is also different from comparing the eigen-values
#----------------------------------------------------------------------#
############
# G1 x G2 #
############
par(mar=c(4.5,4.5,1,1))
# plot G1 x G2
plot(c(diag(G1),G1[upper.tri(G1)]),c(diag(G2),G2[upper.tri(G2)]),xlab="G1",ylab="G2")
abline(0,1,lty=2,col=2)
# plot eigen-values(G1) x eigen-values(G2)
plot(eigen(G1)$values,eigen(G2)$values,xlab="eigen-values(G1)",ylab="eigen-values(G2)")
abline(0,1,lty=2,col=2)
# plot eigen-values(G2)xkappa
tmp <- Gkappa(G1,G2)
plot(tmp$lambda,tmp$kappa,xlim=range(tmp),ylim=range(tmp),xlab="lambda",ylab="kappa")
abline(0,1,lty=2,col=2)
############
# G1 x G3 #
############
par(mar=c(4.5,4.5,1,1))
# plot G1xG3
plot(c(diag(G1),G1[upper.tri(G1)]),c(diag(G3),G3[upper.tri(G3)]),xlab="G1",ylab="G3")
abline(0,1,lty=2,col=2)
# plot eigen-values(G1)xeigen-values(G3)
plot(eigen(G1)$values,eigen(G3)$values,xlab="eigen-values(G1)",ylab="eigen-values(G3)")
abline(0,1,lty=2,col=2)
```

```
# plot eigen-values(G3)xkappa
tmp <- Gkappa(G1,G3)
plot(tmp$lambda,tmp$kappa,xlim=range(tmp),ylim=range(tmp),xlab="lambda",ylab="kappa")
abline(0,1,lty=2,col=2)

# simulate correlated genotypes
M <- simGeno(n=100,AlleleFreq=rep(0.5,1000),LD=TRUE,phased=FALSE)
# three GRMs from the simulated genotypes
# the first with 500 SNPs
G1 <- mkGRM(M[,1:500])
# the second with 500 SNPs that don't overlap those from the first
G2 <- mkGRM(M[,501:1000])
# the third with all SNPs (500 SNPs overlap with those from the first matrix)
G3 <- mkGRM(M)
#--------------------------------------------------------------------#
# compare GRMs using kappa
# verify how that is different from directly comparing the values
# verify how that is also different from comparing the eigen-values
#--------------------------------------------------------------------#
############
# G1 x G2 #
############
par(mfrow=c(2,2),mar=c(4.5,4.5,1,1))
# plot G1xG2
plot(c(diag(G1),G1[upper.tri(G1)]),c(diag(G2),G2[upper.tri(G2)]),xlab="G1",ylab="G2")
abline(0,1,lty=2,col=2)
# plot eigen-values(G1)xeigen-values(G2)
plot(eigen(G1)$values,eigen(G2)$values,xlab="lambda(G1)",ylab="lambda(G2)")
abline(0,1,lty=2,col=2)
# plot eigen-values(G2)xkappa
tmp <- Gkappa(G1,G2)
plot(tmp$lambda,tmp$kappa,xlim=range(tmp),ylim=range(tmp),xlab="lambda",ylab="kappa")
abline(0,1,lty=2,col=2)
plot(NA,xlim=c(0,1),ylim=c(0,1),xlab="",ylab="",axes=FALSE)
############
# G1 x G3 #
############
par(mfrow=c(2,2),mar=c(4.5,4.5,1,1))
# plot G1xG3
plot(c(diag(G1),G1[upper.tri(G1)]),c(diag(G3),G3[upper.tri(G3)]),xlab="G1",ylab="G3")
abline(0,1,lty=2,col=2)
# plot eigen-values(G1)xeigen-values(G3)
plot(eigen(G1)$values,eigen(G3)$values,xlab="lambda(G1)",ylab="lambda(G3)")
abline(0,1,lty=2,col=2)
# plot eigen-values(G3)xkappa
tmp <- Gkappa(G1,G3)
plot(tmp$lambda,tmp$kappa,xlim=range(tmp),ylim=range(tmp),xlab="lambda",ylab="kappa")
abline(0,1,lty=2,col=2)
plot(NA,xlim=c(0,1),ylim=c(0,1),xlab="",ylab="",axes=FALSE)
```

LimPredAcc *Calculates the expected limit of prediction accuracy*

Description

This function calculates the expected limit of prediction accuracy, using the relationship between two populations based on the singular-value decomposition of their genotype matrices. This limit is obtained as in Cuyabano and Gondro, 2020 (in prep), and the two populations must have the SNP-genotypes on the same set of SNPs.

Usage

```
LimPredAcc(a, b, n, h2, p = 0.95)
```

Arguments

a	The intercept obtained with popCor.
b	The slope obtained with popCor.
n	The size of the test population (the population for which predictions are to be made).
h2	The heritability of the trait.
p	The confidence level of the interval for the expected limit of prediction accuracy.

Value

A data frame with the mean expected limit of prediction accuracy, as well as the boundaries of its confidence interval.

Examples

```
# simulate genotypes
M <- simGeno(n=500,AlleleFreq=rep(0.5,100))
# separate two populations of individuals
# note that the populations do not need to have the same size
M1 <- M[1:300,]
M2 <- M[301:500,]
tmp <- popCor(M1,M2)
LimPredAcc(a=tmp$a,b=tmp$b,n=200,h2=0.5)
LimPredAcc(a=tmp$a,b=tmp$b,n=200,h2=seq(0.1,0.9,0.1))
# compare the expected limits of prediction accuracy to the classic sqrt(h2)
par(mar=c(4.5,4.5,1,1))
curve(sqrt(x),xlim=c(0,1),ylim=c(0,1),xlab=expression(h^2),ylab="Limit of prediction accuracy")
aux <- LimPredAcc(a=tmp$a,b=tmp$b,n=200,h2=seq(0.01,0.99,0.01))
lines(seq(0.01,0.99,0.01),aux$mean,lty=2)
lines(seq(0.01,0.99,0.01),aux$CI_lower,lty=3,col=2)
lines(seq(0.01,0.99,0.01),aux$CI_upper,lty=3,col=2)
legend("topleft",lty=c(1,2,3),col=c(1,1,2),legend=c("sqrt(h2)","R_expected","R_CI"))
```

MatrixInv *Inverse of a matrix*

Description

This function inverts any matrix, using the best method for that specific matrix.

Usage

```
MatrixInv(A, use.eigen = FALSE, U = NULL, L = NULL)
```

Arguments

A	A matrix.
use.eigen	A logical indicating if the eigen-decomposition should be used to obtain the inverse. Valid only for symmetric matrices.
U	A matrix of the eigenvectors (optional).
L	A vector with the eigenvalues (optional).

Value

The inverse of the matrix.

---

mkGRM *Genomic relationship matrix*

Description

This function builds the genomic relationship matrix based on SNP-genotypes.

Usage

```
mkGRM(M, method = "VanRaden")
```

Arguments

M	A SNP-genotype matrix (unphased).
method	The method to use for standardizing the SNP-genotypes. Deafult is `method="VanRaden"`, and alternatively can be set `method="Individual"`, which will standardize each SNP individually.

Value

The genomic relationship matrix.

Examples

```
M <- simGeno(5,rep(0.5,1000))
M <- M[,apply(M,2,var) != 0]
mkGRM(M)
mkGRM(M,method="Individual")
```

mkPED *Pedigree relationship matrix*

Description

This function builds the pedigree relationship matrix based on a population info in the same format as the generated by function `simPopInfo`. If you are generating the population info using `simPopInfo`, the pedigree relationship matrix can be created directly from `simPopInfo` by setting the parameter `pedigree=TRUE`.

Usage

```
mkPED(pop.info)
```

Arguments

`pop.info`	The population info, in the same format generated by `simPopInfo`.

Value

The pedigree relationship matrix.

Examples

```
mkPED(simPopInfo(10,0))
```

plotMP *Plot eigen-values comparatively with The Marchenko-Pastur distribution*

Description

Plot eigen-values comparatively with The Marchenko-Pastur distribution

Usage

```
plotMP(L, a, s2, f = "cumulative")
```

Arguments

`L`	A vectors of eigen-values.
`a`	Rate parameter (n/m = #observations/#variables).
`s2`	Variance parameter.
`f`	The type of function to be plotted, `"density"` or `"cumulative"` (default).

pMP *The Marchenko-Pastur cumulative distribution*

Description

Cumulative distribution function of the Marchenko-Pastur distribution with rate parameter equal to `a` and variance equal to `s2`.

Usage

```
pMP(x, a, s2)
```

Arguments

`x`	A vectors of quantiles.
`a`	Rate parameter (n/m = #observations/#variables).
`s2`	Variance parameter.

Value

The cumulative probability distribution.

popBreed *Simulates genotypes for generations of individuals*

Description

This function outputs SNP-genotypes for generations of individuals, based on the given SNPs for generation zero (founder population) and on the population info in the same format generated by `simPopInfo`.

Usage

```
popBreed(M, pop.info)
```

Arguments

`M`	A phased SNP matrix for generation zero.
`pop.info`	The population info, in the same format generated by `simPopInfo`.

Value

A matrix with the SNPs for all individuals.

See Also

`simGeno` `simPopInfo`

Examples

```
M <- simGeno(10,rep(0.5,3),phased=TRUE)
pop.info <- simPopInfo(c(10,15),1)
popBreed(M,pop.info)
```

---

popCor *Calculates the relationship between two populations*

Description

This function calculates the relationship between two populations based on the singular-value decomposition of their genotype matrices. The relationship is obtained as in Cuyabano and Gondro, 2020 (in prep.), and the two populations must have the SNP-genotypes on the same set of SNPs.

Usage

```
popCor(M1, M2)
```

Arguments

M1	The SNP-genotypes of population one.
M2	The SNP-genotypes of population two.

Value

The parameters of the linear regression (intercept and slope) on the components built from the SVDs.

Examples

```
# simulate genotypes
M <- simGeno(n=500,AlleleFreq=rep(0.5,100),LD=TRUE)
# separate two populations of individuals
# note that the populations do not need to have the same size
M1 <- M[1:300,]
M2 <- M[301:500,]
popCor(M1,M2)
```

REMLsolve *REML estimates of variance components*

Description

This function estimates the variance components and the fixed effects in a mixed model, using the restricted maximum likelihood (REML).

Usage

```
REMLsolve(y, X = NULL, Z = NULL, VarMat = NULL, R = NULL)
```

Arguments

y	The response variable.
X	A matrix of fixed effects.
Z	The design matrix (or list of matrices) of the random effects. If all design matrices are an identity, this argument does not need to be provided.
VarMat	The variance-covariance matrix (or list of matrices) of the random effects.
R	A variance-covariance matrix for the residuals, if they are not independent and identically distributed (i.e. R != In)

Value

FixedEff The estimated fixed effects.

VarComp The estimated variance components for all random effects, and for the residual.

Examples

```
## SINGLE COMPONENT WITH FIXED EFFECTS ##
# simulate genotypes
M <- simGeno(n=500,AlleleFreq=runif(100,0.05,0.5))
# simulate phenotypes
pheno <- simPheno(M,0.6,Vy=10)
# add fixed effects
X <- cbind(rep(1,nrow(M)),runif(nrow(M)),rbinom(nrow(M),1,0.5))
b <- c(5,-1,2)
y <- pheno$y + as.numeric(X%*%b)
# REML
G <- mkGRM(M)
REMLsolve(y=y,X=X,VarMat=G)

## TWO VARIANCE COMPONENTS ##
# simulate genotypes
M <- simGeno(n=500,AlleleFreq=runif(100,0.05,0.5))
# simulate phenotypes
pheno <- simPheno(list(M[,1:70],M[,71:100]),c(0.4,0.2),Vy=10)
# add fixed effects
X <- cbind(rep(1,nrow(M)),runif(nrow(M)),rbinom(nrow(M),1,0.5))
b <- c(5,-1,2)
y <- pheno$y + as.numeric(X%*%b)
# REML
G <- list(mkGRM(M[,1:70]),mkGRM(M[,71:100]))
REMLsolve(y=y,X=X,VarMat=G)
REMLsolve(y=y,X=X,VarMat=list(diag(1,70),diag(1,30)),Z=list(M[,1:70],M[,71:100]))
tmp <- REMLsolve(y=y,X=X,VarMat=list(diag(1,70),diag(1,30)),Z=list(M[,1:70],M[,71:100]))$VarComp
c(sum(apply(M[,1:70],2,var)),sum(apply(M[,71:100],2,var)),1)*tmp
```

---

simGeno *Simulate genotypes*

Description

This function simulates numeric SNP-genotypes.

Usage

```
simGeno(n, AlleleFreq, LD = FALSE, R = NULL, phased = FALSE)
```

Arguments

n	The number of individuals to simulate.
AlleleFreq	A vector with the required allele frequencies for each SNP, with length(AlleleFreq) being the number of SNPs.
LD	Logical indicating if SNP-genotypes should be simulated in linkage disequilibrium.
R	The correlation matrix for the linkage disequilibrium between SNPs. If R=NULL, a default matrix with 0.95^(lag between SNPs) will be generated.
phased	Lagical indicating if phased genotypes should be returned.

Value

The simulated SNPs, as a single matrix with allele counts (phased=FALSE), or as a list of matrices with t0,1 at each allele (phased=TRUE).

Examples

```
simGeno(n=10,AlleleFreq=rep(0.5,5))
simGeno(n=10,AlleleFreq=rep(0.5,5),phased=TRUE)
simGeno(n=10,AlleleFreq=rep(0.5,5),LD=TRUE)
R_LD <- diag(1,5)
R_LD[abs(col(R_LD)-row(R_LD)) == 1] <- 0.95
R_LD[abs(col(R_LD)-row(R_LD)) == 2] <- 0.85
R_LD[abs(col(R_LD)-row(R_LD)) == 3] <- 0.75
R_LD[abs(col(R_LD)-row(R_LD)) == 4] <- 0.65
simGeno(n=10,AlleleFreq=rep(0.5,5),LD=TRUE,R=R_LD)
geno <- simGeno(n=1000,AlleleFreq=rep(0.5,5),LD=TRUE,R=R_LD,phased=TRUE)
str(geno)
cor(rbind(geno[[1]],geno[[2]]))
```

---

simPheno *Simulate phenotypes*

Description

This function simulates phenotypes (continuous or binary) based on the quantitative trait loci (QTL), only with the genetic values and residuals (y=g+e).

Usage

```
simPheno(M, h2, Vy = 1, binary = FALSE, p = NULL, sim.error = 1e-04)
```

Arguments

M	A matrix or a list of matrices of the (unphased) SNP-genotypes set to be the QTL.
h2	The heritability.
Vy	The phenotypic variance (for continuous traits). Default is Vy=1.
binary	Logical indicating whether simulated phenotypes should be binary. Default is binary=FALSE.
p	If binary=TRUE, the probability of observing 1's in the phenotypes, with default p=0.5.
sim.error	An error threshold to be considered on the orthogonalization of genetic values and residuals.

Value

y The simulated phenotypes.

a The simulated QTL effects.

g The simulated genetic values.

e The simulated residuals.

h2 The simulated heritability.

See Also

simGeno

Examples

```
## general examples ##
AF <- runif(500,0.05,1)
geno <- simGeno(n=1000,AlleleFreq=AF,LD=TRUE)
pheno <- simPheno(M=geno,h2=0.6,Vy=10)
str(pheno)
pheno <- simPheno(M=geno,h2=0.6,binary=TRUE)
str(pheno)
mean(pheno$y) # this is the realized probability of observing y = 1
pheno <- simPheno(M=geno,h2=0.6,binary=TRUE,p=0.2)
str(pheno)
mean(pheno$y) # this is the realized probability of observing y = 1

## example with multiple components with different heritabilities ##
AF <- runif(500,0.05,1)
geno <- simGeno(n=1000,AlleleFreq=AF,LD=TRUE)
pheno <- simPheno(M=list(geno[,1:350],geno[,351:500]),h2=c(0.2,0.4),Vy=10)
str(pheno)
pheno <- simPheno(M=list(geno[,1:350],geno[,351:500]),h2=c(0.2,0.4),Vy=10,binary=TRUE,p=0.3)
str(pheno)
mean(pheno$y) # this is the realized probability of observing y = 1
```

`simPopInfo` *Simulate a population info*

Description

This function simulates a population info, with individuals' ID, sex, generation and parent information.

Usage

```
simPopInfo(
  n,
  n.gen = 0,
  p.male = 0.5,
  pop.info = NULL,
  pedigree = FALSE,
  pre.id = ""
)
```

Arguments

`n`	The size of each generation simulated. If a single number is provided, all generations will be of the same size.
`n.gen`	The number of generations to simulate after generation zero. The generation counter starts at zero, being that the founder population. By default, `n.gen=0`, and if `n.gen` is set to any positive number, e.g. `n.gen=1`, that means that the info on the founder generation and on one generation after that will be generated.
`p.male`	The proportion of male individuals in the population.
`pop.info`	The information on a base population on which the code should build new generations.
`pedigree`	Should the pedigree be generated? Default is `pedigree=FALSE`.
`pre.id`	An ID label to paste at the begining of the default ID generated by the function.

Value

A data frame with the population info, and the pedigree when required.

info The population info.

ped The pedigree matrix.

Examples

```
simPopInfo(10,0)
simPopInfo(10,2)
simPopInfo(3,1,pedigree=TRUE)
simPopInfo(10,2,0.25)
simPopInfo(c(5,10,15),2)
```

Index

BLUPsolve, 2

ConfIntMeanBoot, 4
ConfIntMeanNorm, 4

dMP, 5

Gkappa, 5

LimPredAcc, 7

MatrixInv, 8
mkGRM, 9
mkPED, 10

plotMP, 10
pMP, 11
popBreed, 11
popCor, 12

REMLsolve, *3*, 12

simGeno, *11*, 13, *15*
simPheno, 14
simPopInfo, *11*, 16