Package 'SimpleMating'

December 2, 2024

Title A R package to breeding crosses optimization

Version 0.0.9001

Description SimpleMating is a two core package for cross optimization. First, it estimates the performance for a set of crosses, such as Mid-Parental value, cross total genetic value, and/or usefulness (using additive and/or non-additive effects), allowing for multi-trait scenarios. Furthermore, it uses a mate allocation algorithm to create a mating plan aiming to maximize a target criterion (mean parental average, total genetic value, or usefulness) and constrains the next generation inbreeding levels.

Depends R (>= 3.6.0)

Imports stats (>= 4.2.3),
ggplot2 (>= 3.3.6),
AGHmatrix (>= 2.1.0)

License GPL-3

Encoding UTF-8

URL https://github.com/Resende-Lab/SimpleMating

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

VignetteBuilder knitr

Suggests knitr, rmarkdown

LazyData true

R topics documented:

`contrib2Cross` *Generates a mating plan based on parental contributions.*

Description

Generates a mating plan based on contribution values predetermined for a set of parents. In optimal contribution selection, a set of parameters are used, such as inbreeding level, estimated breeding value, etc., into optimization algorithms to come up with a set of contributions of each parent to the next generation. This function creates the mating plan based on the contribution of each parent to the next generation.

Usage

```
contrib2Cross(
  nContribution = NULL,
  Selfing = FALSE,
  nCrosses = NULL,
  nProgeny = 1
)
```

Arguments

`nContribution`	data frame with two columns: i. parent id and (ii) contribution of the parent.
`Selfing`	Boolean. Is self allowed in the crosses? Default is FALSE.
`nCrosses`	total number of crosses from the parents sets.
`nProgeny`	number of progeny for each cross generated.

Value

A mating plan based on the individuals and contributions of each one.

Author(s)

Marco Antonio Peixoto, <marco.peixotom@gmail.com>

Examples

```
## Not run:
# 1. Loading the data
Contrib = data.frame(id = paste0('Par_',sample(1:20, 20)),
                     Contrib = sample(c(0:5), 20, replace = T))

# 2. Contributions
MatingPlan <- contrib2Cross(nContribution=Contrib,
                            Selfing = FALSE,
                            nCrosses = 20,
                            nProgeny = 1)

# 3. Output
head(MatingPlan, 10)

## End(Not run)
```

`generic_GenMap` *generic_GenMap*

Description

A genetic map with the chromosome, position in the chromosome (centimorgan) and marker identification for the markers presented in the dataset. 12 chromosomes were simulated.

Usage

```
data(generic_GenMap)
```

Format

A data frame with 1440 rows and 3 variables.

`chr` Chromosome containing the locus

`pos` Genetic map position

`mkr` Unique identifier for locus

Details

This genetic map is used in the prediction of a recombination map for the parental population.

generic_Geno *generic_Geno*

Description

Dataset containing the markers coded as 0,1,2 (aa, Aa, AA) for a population of 100 individuals genotyped for 1440 codominante SNPs markers for 12 chromosome pairs.

Usage

```
data(generic_Geno)
```

Format

A data frame with 100 rows and 1440 variables.

Details

Data was simulated.

generic_IndBLUP *generic_IndBLUP*

Description

Best-linear unbiased prediction (BLUP) for 100 genotypes and two traits.

Usage

```
data(generic_IndBLUP)
```

Format

A data frame with 100 rows and 3 variables.

`Genotypes` Gentoypes identification

`Trait1` BLUP for trait number 1

`Trait2` BLUP for trait number 2

Details

Data was simulated.

generic_MrkEffects *generic_MrkEffects*

Description

Additive and dominance effects for 1440 markers referring to two traits of interest. The first two columns represents the additive effects and the later two columns represents the dominance effects. Each line represents one marker.

Usage

```
data(generic_MrkEffects)
```

Format

A data frame with 1440 rows and 4 variables:

`add_trait1` Additive effects for trait one

`add_trait2` Additive effects for trait two

`dom_trait1` Dominance effects for trait one

`dom_trait2` Dominance effects for trait two

Details

The SNP effects came from a Bayesian model implemented in a multi-trait framework, using both (additive and dominance) together. 12 chromosome pair were simulated.

generic_Pedigree *generic_Pedigree*

Description

A pedigree information for 100 individuals. The three columns represents the individuals, mother, and father identification, respectively.

Usage

```
data(generic_Pedigree)
```

Format

A data frame with 120 rows and 3 variables.

`id` Individuals' id

`mother` mother' id

`father` father' id

generic_Phasedgeno *generic_Phasedgeno*

Description

Dataset with the phased state of a 1440 SNPs coded as 0 (reference allele) and 1 (alternate allele) across 12 chromosome pairs. The haplotypes are available for 100 individuals.

Usage

```
data(generic_Phasedgeno)
```

Format

A data frame with 200 rows and 1440 variables.

Details

Dataset was simulated.

getIndex *Estimates a selection index based on the genotypes values and weights*

Description

The function calculates the index-based trait using as input BLUPs, estimated breeding value, or genetic value for a set of individuals. Weights should be given.

Usage

```
getIndex(Criterion, Weights = NULL, Scale = TRUE)
```

Arguments

Criterion	matrix containing the BLUPs, estimated breeding value, or genetic value for all individuals and several traits.
Weights	row vector containing the weights for each trait.
Scale	logical. Default value is TRUE. If FALSE, it will not scale the traits.

Value

A index based on the traits and weights.

Author(s)

Marco Antonio Peixoto, <marco.peixotom@gmail.com>

Examples

```
## Not run:
# 1. Loading the data
data(generic_IndBLUP)

Crit <- as.matrix(generic_IndBLUP)

# 2. Index
trait_index <- getIndex(Criterion = Crit,
                        Weights = c(0.5, 0.5),
                        Scale = TRUE)

# 3. Output

head(trait_index, 10)

## End(Not run)
```

`getMPV` *Estimates the expected mid-parental value for a set of crosses.*

Description

The expected mid-parental value (MPV) is estimated between each pair of individuals presented in the MatePlan. It uses the Criterion argument input for the estimation. Then, the Criterion should be BLUPs for the individuals presented in the MatePlan. For more than one trait, the MPV is estimated by an index, a linear combination among the traits. Weights should be informed for each trait in this case. The relationship matrix (K) here presented as an argument is for creating the output in the form of the input for the optimization function (selectCrosses).

Usage

```
getMPV(MatePlan, Criterion, K = NULL, Weights = NULL, Scale = TRUE)
```

Arguments

`MatePlan`	vector of possible crosses to make.
`Criterion`	data frame with the parents id and the BLUPs of the individuals.
`K`	relationship matrix between all individuals.
`Weights`	vector with the weights for each trait. Only used when more than one trait is given.
`Scale`	Boolean. If TRUE, the traits values will be scaled. Default is TRUE. Only used when more than one trait is given.

Value

A data frame with all possible crosses from the MatePlan, their mid-parental value, and covariance from the relationship matrix.

Author(s)

Rodrigo R Amadeu, <rramadeu@gmail.com> & Marco A. Peixoto, <marco.peixotom@gmail.com>

Examples

```
## Not run:
# 1.Loading the data
data(lines_Geno)
data(lines_IndBLUP)

# 2.Criterion
Crit <- data.frame(Id = lines_IndBLUP[, 1],
                   Criterion = lines_IndBLUP[, 2])

# 3. Creating relationship matrix
relMat <- (lines_Geno %*% t(lines_Geno)) / ncol(lines_Geno)

# 4.Mating Plan
CrossPlan <- planCross(TargetPop = lines_IndBLUP[, 1])

# 5.Single trait mid-parental value
ST_mpv <- getMPV(MatePlan = CrossPlan,
                 Criterion = Crit,
                 K = relMat)

head(ST_mpv, 20)

# 6. Criterion for MTM
CritMT <- data.frame(Id = lines_IndBLUP[, 1],
                     Criterion = lines_IndBLUP[, 2:3])


# 7. Multi trait mean parental average
MT_mpv <- getMPV(MatePlan = CrossPlan,
                 Criterion = CritMT,
                 K = relMat,
                 Scale = TRUE,
                 Weights = c(0.3, 0.7))

head(MT_mpv, 20)

## End(Not run)
```

---

getTGV *Estimates the total genetic value for a set of crosses*

Description

The total genetic value for a set of crosses is estimated following the formulae proposed by Falconer and Mackay (1996). It uses the markers and the markers effects (additive and dominance effects). For more than one trait, the total genetic value is estimated by an index, a linear combination among the traits. Weights should be informed for each trait in this later case. The relationship matrix (K) here presented as an argument for creating the output as the input for the optimization function (selectCrosses).

Usage

```
getTGV(
  MatePlan,
  Markers,
  addEff,
  domEff,
  K,
  ploidy = 2,
  Weights = NULL,
  Scale = TRUE
)
```

Arguments

MatePlan	data frame with two columns indicating the crosses.
Markers	matrix with markers information for all candidate parents, coded as 0,1,2.
addEff	column vector (for one trait) or a matrix (for more than one trait) with additive marker effects.
domEff	column vector (for one trait) or a matrix (for more than one trait) with dominance markers effects.
K	relationship matrix between all candidates to parents.
ploidy	data ploidy (generally an even number). Default=2.
Weights	vector with the weights for each trait. Only used when more than one trait is given.
Scale	Boolean. If TRUE, the trait values will be scaled. The default is TRUE. It is only used when more than one trait is given.

Value

A data frame with all possible crosses from the MatePlan (Parent1 and Parent2), their total genetic value (Y), and covariance from the relationship matrix (K).

Author(s)

Marco Antonio Peixoto, <marco.peixotom@gmail.com>

References

Falconer, DS & Mackay TFC (1996). Introduction to quantitative genetics. Pearson Education India.

Examples

```
## Not run:
# 1. Loading the data
data(generic_MrkEffects) # Additive effects
data(generic_Geno) # Markers

# 2. Parents
Parents <- rownames(generic_Geno)

# 3. Creating the mating plan
```

```
CrossPlan <- planCross(TargetPop = Parents,
                       MateDesign = "half")

# 4. Creating relationship matrix
relMat <- (generic_Geno %*% t(generic_Geno)) / ncol(generic_Geno)

# 4. Single trait
ST_tgv <- getTGV(MatePlan = CrossPlan,
                 Markers = generic_Geno,
                 addEff = generic_MrkEffects[, 1],
                 domEff = generic_MrkEffects[, 3],
                 K = relMat)

head(ST_tgv, 20)

# 5. Multi trait
MT_tgv <- getTGV(MatePlan = CrossPlan,
                 Markers = generic_Geno,
                 addEff = generic_MrkEffects[, 1:2],
                 domEff = generic_MrkEffects[, 3:4],
                 K = relMat,
                 Weights = c(0.8, 0.2))

head(MT_tgv, 20)

## End(Not run)
```

`getUsefA` *Prediction of usefulness for a set of crosses (Single additive trait)*

Description

Predicts usefulness component for a set of crosses. It accounts for only one trait controlled by additive effects. The variances were implemented according to Lehermeier et al. (2017). It accommodates Doubled-haploids (DH) and Recombinant inbred lines (RILs) types. The genetic map is used to built a recombination map for the population (we implemented the Haldane map function internally).

Usage

```
getUsefA(
  MatePlan,
  Markers,
  addEff,
  K,
  Map.In,
  linkDes = NULL,
  propSel = 0.05,
  Type = "DH",
  Generation = 1
)
```

Arguments

MatePlan	data frame with the two columns indicating the crosses to estimates usefulness.
Markers	matrix with markers information for all candidate parents, coded as 0,2. Missing values should be coded as NA.
addEff	column vector with additive marker effects.
K	relationship matrix between all genotypes.
Map.In	data frame with the genetic map information, i.e., Chromosome containing the locus, genetic map position, and unique identifier for locus.
linkDes	Linkage disequilibrium matrix with the size of the total number of SNPs. This is optional, and it should be used only if the information on the genetic map is not available.
propSel	Value representing the proportion of the selected individuals. Default is 0.05.
Type	which kind of system of mating: "DH": doubled-haploids lines or "RIL": Recombinant inbred lines.
Generation	integer. Indicates the generation where the DH lines are generate or the RILs are extracted. According to Lehermeier et al. (2017) DH derived from F1 generation is '1' and RILs from F2 generation is '1'. Also, for infinite generation a value superior to 10 should be used.

Value

A data frame with means, variances, and usefulness for each pair of crosses presented in the MatePlan.

Author(s)

Marco Antonio Peixoto, <marco.peixotom@gmail.com>

References

Lehermeier, C., de los Campos, TeyssC(dre, S., & SchC6n, C. C. (2017). Genetic gain increases by applying the usefulness criterion with improved variance prediction in selection of crosses. Genetics, 207(4), 1651-1661.

Bonk, S., Reichelt, M., Teuscher, F., Segelke, D., & Reinsch, N. (2016). Mendelian sampling covariability of marker effects and genetic values. Genetics Selection Evolution, 48(1), 1-11.

Examples

```
## Not run:
# 1. Loading the dataset
data(lines_GenMap) # Genetic Map

data(lines_addEffects) # Additive effects

data(lines_Geno) # Markers

# 2. Parents
Parents <- rownames(lines_Geno)

# 3. Creating the mating plan
plan <- planCross(TargetPop = Parents,
```

```
                  MateDesign = "half")

# 4. Creating relationship matrix based on markers
relMat <- (lines_Geno %*% t(lines_Geno)) / ncol(lines_Geno)

# 5. Calculating the usefulness of trait number 1
usef_add <- getUsefA(MatePlan = plan,
                     Markers = lines_Geno,
                     addEff = lines_addEffects[, 1],
                     Map.In = lines_GenMap,
                     K = relMat,
                     propSel = 0.05,
                     Type = "DH",
                     Generation = 1)

head(usef_add[[1]], 10)

head(usef_add[[2]], 10)

## End(Not run)
```

`getUsefAD` *Prediction of usefulness for a set of crosses (Single additive/dominance trait)*

Description

Predicts usefulness component for a set of crosses. It accounts for only one trait controlled by additive effects. The variances were implemented according to Lehermeier et al. (2017), Bonk et al. (2016), and Wolfe et al. (2021). The genetic map is used to built a recombination map for the population (we implemented the Haldane map function). Two methods are implemented, one that uses phased haplotypes and another that uses non phased diplotypes.

Usage

```
getUsefAD(
  MatePlan,
  Markers,
  addEff,
  domEff,
  K,
  Map.In,
  linkDes = NULL,
  propSel = 0.05,
  Method = "Phased"
)
```

Arguments

`MatePlan`	data frame with the two columns indicating the crosses.

Markers	matrix with markers information for all candidate parents, coded as 0,1,2. If Method is equal 'Phased', phased haplotypes should be given and the markers is coded as 0 and 1. Missing values should be coded as NA.
addEff	column vector with additive marker effects for the trait.
domEff	column vector with dominance markers effects for the trait.
K	relationship matrix.
Map.In	data frame with the mapping information, i.e., Chromosome containing the locus, genetic map position, and unique identifier for locus.
linkDes	Linkage disequilibrium matrix with the size of the total number of SNPs. This is optional, and it should be used only if the information on the genetic map is not available.
propSel	Value representing the proportion of the selected individuals. Default is 0.05.
Method	Which method should be used to calculates the progeny variances. The implemented methods are Phased and NonPhased.

Value

A data frame with means, variances, and usefulness for each pair of crosses presented in the MatePlan.

Author(s)

Marco Antonio Peixoto, <marco.peixotom@gmail.com>

References

Lehermeier, C., de los Campos, TeyssC(dre, S., & SchC6n, C. C. (2017). Genetic gain increases by applying the usefulness criterion with improved variance prediction in selection of crosses. Genetics, 207(4), 1651-1661.

Bonk, S., Reichelt, M., Teuscher, F., Segelke, D., & Reinsch, N. (2016). Mendelian sampling covariability of marker effects and genetic values. Genetics Selection Evolution, 48(1), 1-11.

Wolfe, M. D., Chan, A. W., Kulakow, P., Rabbi, I., & Jannink, J. L. (2021). Genomic mating in outbred species: predicting cross usefulness with additive and total genetic covariance matrices. Genetics, 219(3), iyab122.

Peixoto, Amadeu, Bhering, Ferrao, Munoz, & Resende Jr. (2024). SimpleMating: R-package for prediction and optimization of breeding crosses using genomic selection

Examples

```
## Not run:
# 1.Loading the dataset.
data(generic_GenMap) # Genetic Map

data(generic_MrkEffects) # Additive effects

data(generic_Geno) # Markers

data(generic_Phasedgeno) # Phased haplotypes

# 2. Parents
Parents <- rownames(generic_Geno)
```

```
# 3.Creating the mating plan
plan <- planCross(TargetPop = Parents,
                  MateDesign = "half")

# 4. Creating relationship matrix based on markers
relMat <- (generic_Geno %*% t(generic_Geno)) / ncol(generic_Geno)

# 5.Calculating the usefulness using Phased method
usefPhased <- getUsefAD(MatePlan = plan,
                        Markers = generic_Phasedgeno,
                        addEff = generic_MrkEffects[, 1],
                        domEff = generic_MrkEffects[, 3],
                        Map.In = generic_GenMap,
                        K = relMat,
                        propSel = 0.05,
                        Method = "Phased")

head(usefPhased[[1]], 10)

head(usefPhased[[2]], 10)

# 6.Calculating the usefulness using 'NonPhased' method
usefNonPhased <- getUsefAD(MatePlan = plan,
                           Markers = generic_Geno,
                           addEff = generic_MrkEffects[, 1],
                           domEff = generic_MrkEffects[, 3],
                           Map.In = generic_GenMap,
                           K = relMat,
                           propSel = 0.05,
                           Method = "NonPhased" )


head(usefNonPhased[[1]], 10)

head(usefNonPhased[[2]], 10)


## End(Not run)
```

`getUsefAD_mt` *Prediction of usefulness for a set of crosses (Multi additive/dominance traits)*

Description

Predicts usefulness component for a set of crosses. It accounts for only one trait controlled by additive effects. The variances were implemented according to Lehermeier et al. (2017), Bonk et al. (2016), Wolfe et al. (2021), and Peixoto et al. (2024). The genetic map is used to built a recombination map for the population (we implemented the Haldane map function). Two methods are implemented, one that uses phased haplotypes and another that uses non phased diplotypes. Weights should be given.

Usage

```
getUsefAD_mt(
  MatePlan,
  Markers,
  addEff,
  domEff,
  K,
  Map.In,
  linkDes = NULL,
  propSel = 0.05,
  Weights = NULL,
  Method = "Phased"
)
```

Arguments

MatePlan	data frame with the two columns indicating the crosses.
Markers	matrix with markers information for all candidate parents, coded as 0,1,2. If Method is equal 'Phased', phased haplotypes should be given and the markers is coded as 0 and 1. Missing values should be coded as NA.
addEff	matrix with additive marker effects for each trait (mxn, where 'm' is the number of individuals and 'n' represents the number of traits).
domEff	matrix with dominance markers effects for each trait (mxn, where 'm' is the number of individuals and 'n' represents the number of traits).
K	relationship matrix.
Map.In	data frame with the mapping information, i.e., Chromosome containing the locus, genetic map position, and unique identifier for locus.
linkDes	Linkage disequilibrium matrix with the size of the total number of SNPs. This is optional, and it should be used only if the information on the genetic map is not available.
propSel	Value representing the proportion of the selected individuals.
Weights	Row vector containing the weights for each trait
Method	Which method should be used to calculates the progeny variances. The implemented methods are Phased and NonPhased.

Value

A data frame with means, variances, and usefulness for each pair of crosses presented in the MatePlan.

Author(s)

Marco Antonio Peixoto, <marco.peixotom@gmail.com>

References

Lehermeier, C., de los Campos, TeyssC(dre, S., & SchC6n, C. C. (2017). Genetic gain increases by applying the usefulness criterion with improved variance prediction in selection of crosses. Genetics, 207(4), 1651-1661.

Bonk, S., Reichelt, M., Teuscher, F., Segelke, D., & Reinsch, N. (2016). Mendelian sampling covariability of marker effects and genetic values. Genetics Selection Evolution, 48(1), 1-11.

Wolfe, M. D., Chan, A. W., Kulakow, P., Rabbi, I., & Jannink, J. L. (2021). Genomic mating in outbred species: predicting cross usefulness with additive and total genetic covariance matrices. Genetics, 219(3), iyab122.

Peixoto, Amadeu, Bhering, Ferrao, Munoz, & Resende Jr. (2024). SimpleMating: R-package for prediction and optimization of breeding crosses using genomic selection

Examples

```
## Not run:
# 1. Loading the dataset
data(generic_GenMap) # Genetic Map

data(generic_MrkEffects) # Additive effects

data(generic_Geno) # Markers

data(generic_Phasedgeno) # Phased haplotypes


# 2. Parents
Parents <- rownames(generic_Geno)

# 3.Creating the mating plan
plan <- planCross(TargetPop = Parents,
                  MateDesign = "half")

# 4. Creating relationship matrix based on markers
relMat <- (generic_Geno %*% t(generic_Geno)) / ncol(generic_Geno)

# 5.Calculating the usefulness using 'Phased' method
usefPhased <- getUsefAD_mt(MatePlan = plan,
                           Markers = generic_Phasedgeno,
                           addEff = generic_MrkEffects[, 1:2],
                           domEff = generic_MrkEffects[, 3:4],
                           Map.In = generic_GenMap,
                           K = relMat,
                           propSel = 0.05,
                           Weights = c(0.2, 0.3),
                           Method = "Phased")

head(usefPhased[[1]] ,10)

head(usefPhased[[2]] ,10)

# 6.Calculating the usefulness using 'NonPhased' method
usefNonPhased <- getUsefAD_mt(MatePlan = plan,
                              Markers = generic_Geno,
                              addEff = generic_MrkEffects[, 1:2],
                              domEff = generic_MrkEffects[, 3:4],
                              Map.In = generic_GenMap,
                              K = relMat,
                              propSel = 0.05,
                              Weights = c(0.2, 0.3),
                              Method = "NonPhased")
```

```
head(usefNonPhased[[1]], 10)

head(usefNonPhased[[2]], 10)

## End(Not run)
```

`getUsefA_mt` *Prediction of usefulness for a set of crosses (Multi additive traits)*

Description

Predicts usefulness component for a set of crosses. It accounts for more than one trait controlled by additive effects. The variances were implemented according to Lehermeier et al. (2017) and Bonk et al. (2016). It accommodates Doubled-haploids (DH) and Recombinant inbred lines (RILs) types. The genetic map is used to built a recombination map for the population (we implemented the Haldane map function internally). Weights should be given.

Usage

```
getUsefA_mt(
  MatePlan,
  Markers,
  addEff,
  K,
  Map.In,
  linkDes = NULL,
  propSel = 0.05,
  Type = "DH",
  Generation = 1,
  Weights = NULL
)
```

Arguments

`MatePlan`	data frame with the two columns indicating the crosses to predict.
`Markers`	matrix with markers information for all candidate parents, coded as 0,2. Missing values should be coded as NA.
`addEff`	matrix with additive marker effects.
`K`	relationship matrix.
`Map.In`	data frame with the genetic map information, i.e., Chromosome containing the locus, genetic map position, and unique identifier for locus.
`linkDes`	Linkage disequilibrium matrix with the size of the total number of SNPs. This is optional, and it should be used only if the information on the genetic map is not available.
`propSel`	Value representing the proportion of the selected individuals. Default is 0.05.
`Type`	which kind of system of mating: "DH": doubled haploids lines or "RIL": Recombinant inbred lines.

Generation	integer. Indicates the generation where the DH lines are generate or the RILs are extracted. According to Lehermeier et al. (2017) DH derived from F1 generation is '1' and RILs from F2 generation is '1'. Also, for infinite generation a value superior to 10 should be used.
Weights	row vector containing the weights for each trait.

Value

A data frame with means, variances, and usefulness for each pair of crosses presented in the MatePlan.

Author(s)

Marco Antonio Peixoto, <marco.peixotom@gmail.com>

References

Lehermeier, C., de los Campos, G., TeyssC(dre, S., & SchC6n, C. C. (2017). Genetic gain increases by applying the usefulness criterion with improved variance prediction in selection of crosses. Genetics, 207(4), 1651-1661.

Bonk, S., Reichelt, M., Teuscher, F., Segelke, D., & Reinsch, N. (2016). Mendelian sampling covariability of marker effects and genetic values. Genetics Selection Evolution, 48(1), 1-11.

Examples

```
## Not run:
# 1. Loading the dataset
data(lines_GenMap) # Genetic Map

data(lines_addEffects) # Additive effects

data(lines_Geno) # Markers

# 2. Parents
Parents <- rownames(lines_Geno)

# 3. Creating the mating plan
plan <- planCross(TargetPop = Parents,
                  MateDesign = "half")

# 4. Creating relationship matrix based on markers
relMat <- (lines_Geno %*% t(lines_Geno)) / ncol(lines_Geno)

# 5.  Usefulness for both traits (DH case)
usef_add <- getUsefA_mt(MatePlan = plan,
                        Markers = lines_Geno,
                        addEff = lines_addEffects,
                        Map.In = lines_GenMap,
                        K = relMat,
                        propSel = 0.05,
                        Type = "DH",
                        Generation = 1,
                        Weights = c(0.4, 0.6))
```

```
head(usef_add[[1]], 10)

head(usef_add[[2]], 10)

## End(Not run)
```

`lines_addEffects` *lines_addEffects*

Description

Additive effects for 1230 markers and for two traits of interest. Each line represents one marker.

Usage

```
data(lines_addEffects)
```

Format

A data frame with 1230 rows and 2 variables.

`add_trait1` Additive effects for trait one

`add_trait2` Additive effects for trait two

Details

The SNP effects came from a Bayesian model implemented using both traits together. A total of 10 chromosomes were simulated.

`lines_GenMap` *"lines_GenMap"*

Description

A genetic map with the chromosome, position in the chromosome (centimorgan) and marker identification for the markers presented in the dataset.

Usage

```
data(lines_GenMap)
```

Format

A data frame with 1230 rows and 3 variables.

`chr` Chromosome containing the locus

`pos` Genetic map position

`mkr` Unique identifier for locus

Details

This genetic map is used in the prediction of a recombination map for the parental population. A total of 10 chromosomes were simulated.

`lines_Geno` *lines_Geno*

Description

Dataset containing the markers coded as 0,1,2 (aa, Aa, AA) for a population of doubled-haploid individuals genotyped for 1230 SNPs markers and 10 chromosomes.

Usage

```
data(lines_Geno)
```

Format

A data frame with 100 rows and 1230 variables.

Details

Data was simulated.

`lines_IndBLUP` *lines_IndBLUP*

Description

Best-linear unbiased prediction (BLUP) for 100 genotypes and two traits.

Usage

```
data(lines_IndBLUP)
```

Format

A data frame with 100 rows and 3 variables:

`Genotypes` Gentoypes identification

`Trait1` BLUP for trait number 1

`Trait2` BLUP for trait number 2

Details

Data was simulated.

planCross *Create a mating plan from a group of parents*

Description

A mating plan with all possible crosses for a given set of parents is generated. When two target populations are used (TargetPop and TargetPop2) the North Carolina 2 mating design (Comstock and Robinson, 1952) is implemented.

Usage

```
planCross(TargetPop, MateDesign = "half", TargetPop2 = NULL, Indiv2keep = NULL)
```

Arguments

TargetPop	Individuals that should be mated.
MateDesign	indicates which type of mating design should be build up. 'full_p': this option creates a crossing plan where all parents are crossed with all parents, considering self of parents and reciprocal crosses. 'full': this option creates a crossing plan where all parents are crossed with all parents, considering reciprocal crosses. 'half_p': this option creates a crossing plan where all parents are crossed with all parents, considering self of parents but not reciprocal crosses. 'half': this option creates a crossing plan where all parents are crossed with all parents, with neither, self of parents or reciprocal crosses. 'maxAvoid': this option creates a maximum avoidance of inbreeding crossing plan (Kimura & Crow, 1963). 'circularPlan': this option creates a circular crossing plan (Kimura & Crow, 1963).
TargetPop2	Optional. Individuals that should be mated against the first list of individuals.
Indiv2keep	Optional. Character vector with a list of candidate individuals. It will be used to filter just those individuals to keep for built the mate plan.

Value

A data frame with all possible crosses.

Author(s)

Marco Antonio Peixoto, <marco.peixotom@gmail.com>

Examples

```
## Not run:
# 1.Loading the dataset.
Parents <- rep(paste0("UF", rep(1:20)))

# 2.Creating the mating plan
plan1 <- planCross(TargetPop = Parents,
                   MateDesign = "half")


head(plan1, 10)
```

```
# 3.Using two set of parents
Parents1 <- Parents[1:10]
Parents2 <- Parents[11:20]

# 4.Creating the mating plan
plan2 <- planCross(TargetPop = Parents1,
                   TargetPop2 = Parents2,
                   MateDesign = "half")


head(plan2, 10)

## End(Not run)
```

`relateThinning` *Return a vector of individuals that maximizes criteria per cluster in the K matrix*

Description

The function filter the individuals and cut off those with high relationship with each other. It does that based on the relationship matrix, the criteria used to rank the individuals, a threshold for group formation and a maximum number of individuals per cluster.

Usage

```
relateThinning(K, Criterion, threshold = 0.5, max.per.cluster = 2)
```

Arguments

`K`	relationship matrix. It could be a SNP-based or pedigree-based relationship matrix.
`Criterion`	data frame with variable to maximize, i.e., estimated breeding value, BLUP, genetic value, etc., for each candidate parent.
`threshold`	K threshold to cluster.
`max.per.cluster`	maximum number of entries per cluster to keep.

Value

A vector with genotype names

Author(s)

Rodrigo R Amadeu, <rramadeu@gmail.com>

Examples

```
## Not run:
# 1.Loading the data
data(generic_IndBLUP) # BLUPs/Criteria
data(generic_Geno) # Markers

# 2.Criterion
Crit <- data.frame(Id = generic_IndBLUP[, 1],
                   Criterion = generic_IndBLUP[, 2])

# 3. Creating relationship matrix
ScaleMarkers <- scale(generic_Geno, scale = FALSE)

relMat <- (ScaleMarkers %*% t(ScaleMarkers)) / ncol(ScaleMarkers)


# 4.Thinning
parents2keep <- relateThinning(K = relMat,
                               Criterion = Crit,
                               threshold = 0.08, # Relationship matrix value
                               max.per.cluster = 5)


# 5.List with the parents to keep
parents2keep


## End(Not run)
```

`selectCrosses` *Select a mating plan that maximizes the criteria given some parameters*

Description

The is the core function for optimization. The input (argument data) is a data frame with four columns: Parent1, Parent2, Y, and K. The Parent1 and Parent2 represent the cross itself. The column Y is a criterion to be maximized. It could be any estimates between the pair of parents, like mid-parental value, cross total genetic value, and usefulness. The last column (K) is the covariance between the pair of individuals that is presented in the cross (i.e., Parent1 and Parent2). This estimation is used to do a cut off in the mating plan, based on the value given in culling.pairwise.k. All crosses with relatedness values higher than this threshold will be discarded. This is done in order to improve the value of the given criterion (Y) under a constrain in relatedness.

Usage

```
selectCrosses(
  data,
  n.cross = 200,
  max.cross = 4,
  min.cross = 2,
  max.cross.to.search = 1e+05,
```

```
  culling.pairwise.k = NULL
)
```

Arguments

data	data frame with possible crosses to filter out.
n.cross	number of crosses in the mating plan.
max.cross	maximum number of crosses per individual in the plan.
min.cross	minimum number of crosses per individual in the plan (this is the target, some parents might be used just one time).
max.cross.to.search	maximum number of rows in the input to search for the solution.
culling.pairwise.k	don't allow crosses with more than this threshold of relatedness. Default is NULL, don't use it.

Value

A data frame with all possible crosses.

Author(s)

Rodrigo R Amadeu, <rramadeu@gmail.com> Marco Antonio Peixoto, <marco.peixotom@gmail.com>

Examples

```
## Not run:
# 1. Loading the data
data(lines_Geno)
data(lines_addEffects)
data(lines_GenMap)

# 2. Crossing plan
CrossPlan <- planCross(TargetPop = rownames(lines_Geno))

# 3 Creating relationship matrix
relMat <- (lines_Geno %*% t(lines_Geno)) / ncol(lines_Geno)

# 4. Based on usefulness

# 4.1 Calculating the usefulness of trait number 1
usef_add <- getUsefA(MatePlan = CrossPlan,
                     Markers = lines_Geno,
                     addEff = lines_addEffects[, 1],
                     Map.In = lines_GenMap,
                  K = relMat,
                     propSel = 0.05,
                     K = relMat,
                     Type = "RIL")


# 4.2 Crosses selected
maxGainPlan <- selectCrosses(data = usef_add[[2]],
                             n.cross = 25,
```

```
                              max.cross = 3,
                              min.cross = 1,
                              culling.pairwise.k = 1)

# Crosses parameters
maxGainPlan[[1]]

# Mating Plan
maxGainPlan[[2]]

# Graph
maxGainPlan[[3]]

# 5. Based on mid-parental value

# 5.1 Criterion
Crit <- data.frame(Id = lines_IndBLUP[, 1],
                   Criterion = lines_IndBLUP[, 2])

# 5.2 Single trait mean parental average
ST_mpV <- getMPV(MatePlan = CrossPlan,
                 Criterion = Crit,
                 K = relMat)

# 5.3 Crosses selected
maxGainPlan2 <- selectCrosses(data = ST_mpV,
                              n.cross = 25,
                              max.cross = 3,
                              min.cross = 1,
                              culling.pairwise.k = 1)


# Crosses parameters
maxGainPlan[[1]]

# Mating Plan
maxGainPlan[[2]]

# Graph
maxGainPlan[[3]]

## End(Not run)
```

`setCrosses` *Creates a mating plan that maximizes the mid-parental value of a set of crosses*

Description

For a set of individuals (parents), this function creates a mating plan with the mid-parental value of the individuals. The best option here is to use BLUPs from each candidate to be parent. Multiple trait can also be used, and weights should be given. No constrains to inbreeding is used. If there is information regarding relationship among individuals, please, use the function selectCrosses.

Usage

```
setCrosses(
  Criterion,
  MateDesign = "half",
  n.cross = 200,
  max.cross = 4,
  min.cross = 2,
  max.cross.to.search = 1e+05,
  Weights = NULL,
  Scale = TRUE
)
```

Arguments

Criterion	data frame with the parents and the criterion used for each parent.
MateDesign	indicates which type of mating design should be build up. 'full_p' all parents crossed with all parents, considering self of parents and reciprocal crosses. 'full' all parents crossed with all parents, considering reciprocal crosses. 'half_p' all parents crossed with all parents, considering self of parents but not reciprocal crosses. 'half' all parents crossed with all parents, with neither, self of parents or reciprocal crosses.
n.cross	number of crosses in the mating plan.
max.cross	maximum number of crosses per individual in the plan.
min.cross	minimum number of crosses per individual in the plan (this is the target, some parents might be used just one time).
max.cross.to.search	maximum number of rows in the input to search for the solution.
Weights	vector. Weights for each trait for building the selection index to be used.
Scale	logical. If 'TRUE' it will scale the traits. Otherwise, will not scale the traits beforehand. The default is TRUE.

Value

A data frame with all possible crosses.

Author(s)

Rodrigo R Amadeu, <rramadeu@gmail.com> Marco Antonio Peixoto, <marco.peixotom@gmail.com>

Examples

```
## Not run:
# 1.Loading the data
data(lines_IndBLUP) # BLUPs/Criteria

# 2. Criterion for one trait
Crit <- data.frame(Id = lines_IndBLUP[, 1],
                   Crit = lines_IndBLUP[, 2] )


# 3. Cross
Matplan <- setCrosses(Criterion = Crit,
```

```
                      MateDesign = "half",
                      n.cross = 100,
                      max.cross = 10,
                      min.cross = 1)


# 4. Stats
Matplan[[1]]

# 5.Mating plan
Matplan[[2]]


# 6. Criterion for two traits
CritMT <- data.frame(Id = lines_IndBLUP[, 1],
                     Crit = lines_IndBLUP[, 2:3])


# 7. Cross
MatplanMT <- setCrosses(Criterion = CritMT,
                        MateDesign = "half",
                        n.cross = 100,
                        max.cross = 10,
                        min.cross = 1,
                        Weights = c(0.5, 0.4))


# 8. Stats
MatplanMT[[1]]

# 9.Mating plan
MatplanMT[[2]]

## End(Not run)
```

Index