# Package ‘GenomicMating’

July 2, 2018

<table>
<thead>
<tr>
<th>Type</th>
<th>Package</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>Efficient Breeding by Genomic Mating</td>
</tr>
<tr>
<td>Version</td>
<td>2.0</td>
</tr>
<tr>
<td>Date</td>
<td>2018-07-01</td>
</tr>
<tr>
<td>Author</td>
<td>Deniz Akdemir, Julio Isidro Sanchéz, Hanna Haikka, Itaraju Baracuhy Brum</td>
</tr>
<tr>
<td>Maintainer</td>
<td>Deniz Akdemir <a href="mailto:deniz.akdemir.work@gmail.com">deniz.akdemir.work@gmail.com</a></td>
</tr>
</tbody>
</table>

**Description**


**License**

GPL-2

**Imports**

Rcpp, parallel, stats, emoa,
scatterplot3d, qtl, SOMbrero, kohonen,
plotly, graphics, dplyr, magrittr, LowRankQP

**LinkingTo**

Rcpp, RcppArmadillo

**Suggests**

knitr, rmarkdown

**NeedsCompilation**

yes

**Repository**

CRAN

**Date/Publication**

2018-07-01 23:00:03 UTC

## R topics documented:

- GenomicMating-package ............................................. 2
- Amat.pieces ....................................................... 2
- getGaSolutions .................................................... 3
- getGaSolutionsFrontier ........................................ 8
- getGaSolutionsFrontierMultiTrait ............................ 11
- getGaSolutionsFrontierMultiTraitSimcross ................ 14
- getOptParentalProportions ................................... 16
- plotGM ............................................................ 18

**Index**

21
GenomicMating-package  Efficient Breeding by Genomic Mating

Description

Implements the mate selection approach in: Akdemir and Sanchez. "Efficient Breeding by Genomic Mating." Frontiers in Genetics (2016). Importance parameters were rescaled to the range \([0,1]\). A function to find points on the frontier was added in V1. V2.0 includes additional methods of calculation of mating statistics such as described in Lehermeier (2018) and using simulation of progeny in addition to optimization over multiple traits.

Author(s)

Maintainer: Deniz Akdemir <deniz.akdemir.work@gmail.com> Contributers:

References

Lehermeier at al. "Genetic gain increases by applying the usefulness criterion with improved variance prediction in selection of crosses" Genetics (2017).

Description

This calculates the genomic relationship matrix using the formula in VanRaden (2008)

Usage

amatNpieces(M, pieces=10, mc.cores=1)

Arguments

<table>
<thead>
<tr>
<th>M</th>
<th>The matrix of markers rows corresponding to individuals and columns for markers, the markers scores are coded as -1,0,1 (corresponding to allele counts 0,1,2).</th>
</tr>
</thead>
<tbody>
<tr>
<td>pieces</td>
<td>number of chunks to split the markers</td>
</tr>
<tr>
<td>mc.cores</td>
<td>number of cores to use</td>
</tr>
</tbody>
</table>

Value

a genomic relationship matrix.
getGaSolutions

Author(s)
Deniz Akdemir, Julio Isidro Sanchez, Hanna Haikka, Itaraju Baracuhy Brum

References

Examples

```r
library(genomicmating)
N=50
nmarkers=500
Markers<-c()
for (i in 1:N){
    Markers<-rbind(Markers, sample(-1:1,nmarkers, replace=TRUE))
}
markereffects<-rep(0,nmarkers)
markereffects[sample(1:nmarkers,nmarkers/2)]<-rnorm(nmarkers/2)
Markers[1:5,1:5]

K=Amat.pieces(Markers, pieces=5)
K[1:5,1:5]
```

Description
Genomic mating is similar to genomic selection in terms of estimating marker effects, but in genomic mating the genetic information and the estimated marker effects are used to decide which genotypes should be crossed to obtain the next breeding population. This program uses genetic algorithm to obtain a solution that minimizes inbreeding and maximize gain and usefulness for a given set of importance weights.

Usage

getGaSolutions(Markers, Markers2=NULL, K, markereffects, markermap=NULL, mmates=NULL, minparents=10, impinbreedstepsize=.02, impvar=.1, impforinbreed=.7, keepbest=TRUE, npopGA=100, nitGA=100, plotiters=TRUE, nelite=10, mutprob=1, mc.cores=1, miniters=100, minitbefstop=80, tolparconv=1e-6, noself=F, method=1L, type=0, generation=1, plotMates=TRUE)
Arguments

Markers: The matrix of markers rows corresponding to individuals and columns for markers, the markers scores are coded as -1,0,1. (For Method=3 the markers are coded as probabilities between 0 and 1.)

Markers2: The matrix of markers rows corresponding to individuals and columns for markers, the markers scores are coded as -1,0,1. (For Method=3 the markers are coded as probabilities between 0 and 1.)

K: symmetric genomic relationship matrix, the order of the row and columns of this matrix should follow the order of genotypes in the rows of rbind(Markers, Markers2).

markermap: A map for markers. two columns, first column is named chr second named pos for the chromosome and position of the markers specified above.

markereffects: effects of markers for a trait.

nmates: number of mates to select, default value is NULL (number of mates is equal to number of mates).

minparents: minimum number of parents in the solution (importance parameter for inbreeding is increased till minimum number of parents are included in the mating solution), minimum is 1.

impinbreedstepsize: stepsize for importance parameter for inbreeding to be increased till minimum number of parents are included in the mating solution.

impvar: importance parameter of the cross variance term.

impforinbreed: importance parameter for inbreeding.

keepbest: logical. Default value is TRUE. GA parameter, whether to keep the best solution in each iteration.

npopGA: genetic algorithm parameter: number of solutions generated at each cycle of the GA.

nitGA: genetic algorithm parameter: number of GA cycles before algorithm stops.

plotiters: genetic algorithm parameter: if TRUE the value of the objective function over iterations will be plotted.

nelite: genetic algorithm parameter: number of elite solutions selected at each cycle of the GA.


mc.cores: genetic algorithm parameter: number of cores to use.

miniters: genetic algorithm parameter: minimum number of GA cycles before algorithm stops.

minitbefstop: genetic algorithm parameter: minimum number of GA cycles before algorithm continues when the tolerance is reached (no change in the criterion value).

tolparconv: genetic algorithm parameter: the maximum change in criterion value accepted for convergence.

noself: Is selfing allowed? (TRUE or FALSE).

method: Which method to use? (1,2,3) See Details.

type: Only for method=2. Type of crosses (1 (DH), 2 (RISELFF)).

generation: Only for method=2. Generation at which the cross variances are calculated.

plotMates: Plot a final 3D plot of solutions. (TRUE or FALSE).
Details

The efficient mating problem can be stated as an optimization problem as follows: minimize w.r.t. $P_{32}$

$$r(\lambda_1, \lambda_2, P_{32}) = -(1 - \lambda_1 - \lambda_2) \times \text{Gain}(P_{32}) - \lambda_1 \times \text{Usefulness}(P_{32}) + \lambda_2 \times \text{Inbreeding}(P_{32})$$

where $0 \leq \lambda_1, \lambda_2 \leq 1$, $0 \leq \lambda_2 \leq 1$, $\lambda_1 + \lambda_2 \leq 1$ and the minimization is over the space of the mating matrices $P_{32}$ construction of which is described in detail in the cited article.

Gain($P$) for a mating design $P$ is calculated as $P g$ where $g$ is the vector of genomically estimated breeding values for the parents. Inbreeding($P$) for a mating design $P$ is calculated as $\text{trace}(P K P' + D(P))$ where $K$ is the matrix of genomically estimated relationship matrix for the parents and $D(P)$ is a diagonal matrix for adjustment of the parent relationship matrix to progeny relationship matrix.

Usefulness($P$) measures the variance of a mate pair. The average or the sum of usefulnesses for all pairs in a mating plan can be used to measure the usefulness of a mating plan. There are three options for the calculation of usefulness. Method 1 uses the calculations in "Efficient Breeding by Genomic Mating", Method 2 uses the calculations in "Genetic gain increases by applying the usefulness criterion with improved variance prediction in selection of crosses" without the estimation variance terms. Method 2 comes with two types (DH (type=0) or riself (type=1)) and each of these types can be applied for progeny at a specified “generation”. Method 3 is for polyploid organisms, where the marker data is recorded as proportions of alleles at genomewide loci.

Value

Returns a list with three elements: the first element in this list is the list of mates in the best solution, the second element in the list is the criterion values for the best solutions through the iterations. The last item in the list is a list itself that contains the values of the statistics Gain, Usefulness and Inbreeding.

Author(s)

Deniz Akdemir, Julio Isidro Sanch\'ez, Hanna Haikka, Itaraju Baracuhy Brum

References

Lehermeier at al. "Genetic gain increases by applying the usefulness criterion with improved variance prediction in selection of crosses" Genetics (2017).

Examples

```r
## Not run:
library( GenomicMating )

### Create 100 markers for two sets of populations of size 20.
N=20
rmmarkers=100
Markers<-c()
```
for (i in 1:N){
    Markers<-cbind(Markers,rbinom(nmarkers, 2,.1)-1)
}

Markers2<-c()
for (i in 1:N){
    Markers2<-cbind(Markers2,rbinom(nmarkers, 2,.1)-1)
}

######Marker effects for a trait.
markereffects<-rep(0,nmarkers)
markereffects[sample(1:nmarkers,nmarkers/2)]<-rnorm(nmarkers/2)
Markers[1:5,1:5]

####Relationship matrices (K only for the first population.
#K2 for both populations together.)
#library(parallel)
K<-Amat.pieces(rbind(Markers), pieces=5)
K2<-Amat.pieces(rbind(Markers,Markers2), pieces=5)
K[1:5,1:5]

###putting names
rownames(Markers)<-paste("1", 1:nrow(Markers),sep="_")
rownames(Markers2)<-paste("1", (nrow(Markers)+1):(nrow(Markers)+
nrow(Markers2)),sep="_")
rownames(K2)<-colnames(K2)<-c(rownames(Markers),rownames(Markers2))
rownames(K)<-colnames(K)<-c(rownames(Markers))

###Best genotype in pop 1
which.max(Markers%<%markereffects)
markereffects=markereffects,markermapp=markermapp,nmates=10,
minparents=3, impinbreedstepsize=.02, impvar=.01,
impforinbreed=.01,npopGA=50, nitGA=10, miniters=10,minitbefstop=20,
plotiters=TRUE,mc.cores=1,nelite=20, mutprob=0.8, noself=TRUE,
method=1, type=0L, generation=0L)
gasols

###Mating between pop1 and pop2. Method 1.
getGaSolutions

```r
getGaSolutions <- function(Markers, Markers2, K = K2, markdown, map, nmates = 10,
  minparents = 3,
  impinbreedsteps = 0.02, impvar = 0.02,
  impforinbreed = 0.07,
  npopGA = 50, nitGA = 10, miniters = 10, minitbefstop = 20,
  plotiters = TRUE,
  mc.cores = 2, nelite = 20, mutprob = 0.8, noself = F, method = 1,
  type = 0L, generation = 0L)


gasols2 <- getGaSolutions(Markers = Markers, Markers2 = Markers2, K = K2,
  markdown, map, nmates = 10,
  minparents = 3,
  impinbreedsteps = 0.02, impvar = 0.02,
  impforinbreed = 0.07,
  npopGA = 50, nitGA = 10, miniters = 10, minitbefstop = 20,
  plotiters = TRUE,
  mc.cores = 2, nelite = 20, mutprob = 0.8, noself = F, method = 2,
  type = 0L, generation = 0L)

#### For method 3 polyploid. Markers need to be coded between 0 and 1.
N = 20
nmarkers = 100
Markers <- c()
for (i in 1:N){
  Markers <- rbind(Markers, runif(nmarkers))
}
Markers2 <- c()
for (i in 1:N){
  Markers2 <- rbind(Markers2, runif(nmarkers))
}

markdown <- rep(0, nmarkers)
markdown[sample(1:nmarkers, nmarkers / 2)] <- rnorm(nmarkers / 2)
Markers[1:5, 1:5] = library(parallel)
K = Amat.pieces(rbind(Markers) * 2 - 1, pieces = 5)
K2 = Amat.pieces(rbind(Markers, Markers2) * 2 - 1, pieces = 5)
K[1:5, 1:5]
rownames(Markers) <- paste("1", 1:nrow(Markers), sep = "_")
rownames(Markers2) <- paste("1", 1:nrow(Markers2), sep = "_")
rownames(K2) <- colnames(K2) <- c(rownames(Markers), rownames(Markers2))
rownames(K) <- c(rownames(Markers), rownames(Markers2))

which.max(Markers %*% markdown)
markermap = as.matrix(data.frame(chr = rep(1, nmarkers), pos = seq(0, 1, length = nmarkers)))
```
colnames(Markers)<-1:nmarkers

gasols3<-getGaSolutionsFrontier(Markers=Markers,Markers2=Markers2, K=K2,
markereffects,markermap=markermap,nmates=10,
mniparents=1,
impinbreedstepsize=.02, impvar=.02,
impforinbreed=.07,
npopGA=50, nitGA=10, miniters=10,minitbefstop=20,plotiters=TRUE,
mc.cores=1,nelite=20, mutprob=0.8, noself=F, method=3,
type=0L, generation=0L)
gasols3

## End(Not run)

---

**Description**

Genomic mating is similar to genomic selection in terms of estimating marker effects, but in genomic mating the genetic information and the estimated marker effects are used to decide which genotypes should be crossed to obtain the next breeding population. This program uses genetic algorithm to obtain the frontier solutions that minimize inbreeding and maximize gain and usefulness. The solutions in the optimized frontier are nondominated.

**Usage**

```r
getGaSolutionsFrontier(Markers, Markers2=NULL,K,
markereffects,markermap=NULL,nmates=NULL,npopGA=100,
nitGA=100, mutprob=1, mc.cores=1, noself=F,method=1L,
type=0L, generation=1L,plotiters=F)
```

**Arguments**

- **Markers**
  
  The matrix of markers rows corresponding to individuals and columns for markers, the markers scores are coded as -1,0,1. (For Method=3 the markers are coded as probabilities between 0 and 1.)

- **Markers2**
  
  The matrix of markers rows corresponding to individuals and columns for markers, the markers scores are coded as -1,0,1. (For Method=3 the markers are coded as probabilities between 0 and 1.)

- **K**
  
  symmetric genomic relationship matrix, the order of the row and columns of this matrix should follow the order of genotypes in the rows of `rbind(Markers, Markers2).`
### getGaSolutionsFrontier

- **markermap**: A map for markers. Two columns, first column is named chr, second named pos for the chromosome and position of the markers specified above.
- **markereffects**: Effects of markers for a trait.
- **nmates**: Number of mates to select, default value is NULL (number of mates is equal to number of mates).
- **npopGA**: Genetic algorithm parameter: number of solutions generated at each cycle of the GA.
- **nitGA**: Genetic algorithm parameter: number of GA cycles before algorithm stops.
- **mutprob**: Genetic algorithm parameter: mutation probability.
- **mc.cores**: Genetic algorithm parameter: number of cores to use.
- **plotiters**: Genetic algorithm parameter: if TRUE the value of the objective function over iterations will be plotted.
- **noself**: Is selfing allowed? (TRUE or FALSE).
- **method**: Which method to use? (1, 2, 3) See Details.
- **type**: Only for method=2. Type of crosses (1 (DH), 2 (RISELF)).
- **generation**: Only for method=2. Generation at which the cross variances are calculated.

### Details

This program uses genetic algorithm to produce a number of solutions on the frontier curve simultaneously for the multi-objective optimization problem which is defined by minimization of $-\text{Gain}(P_{32})$, $-\text{Usefulness}(P_{32})$ and $\text{Inbreeding}(P_{32})$ with respect to $P_{32}$.

### Value

Returns a list with two elements: the first element in this list is a list of solutions found on the frontier, the second element is the matrix of criterion values (Gain, Usefulness, and Inbreeding) corresponding to these solutions.

### Author(s)

Deniz Akdemir, Julio Isidro Sanch\'ez, Hanna Haikka, Itaraju Baracuhy Brum

### References


Lehermeier et al. "Genetic gain increases by applying the usefulness criterion with improved variance prediction in selection of crosses" Genetics (2017).


Examples

```r
## Not run:
library(GenomicMating)

####
### For method 3 polyploid. Markers need to be coded between 0 and 1.
N=20
nmarkers=100
Markers<-c()
for (i in 1:N){
  Markers<-rbind(Markers,runif(nmarkers))
}

Markers2<-c()
for (i in 1:N){
  Markers2<-rbind(Markers2,runif(nmarkers))
}

markereffects<-rep(0,nmarkers)
markereffects[sample(1:nmarkers,nmarkers/2)]<-rnorm(nmarkers/2)
Markers[1:5,1:5]
#library(parallel)
K=Amat.pieces(rbind(Markers)*2-1, pieces=5)
K2=Amat.pieces(rbind(Markers,Markers2)*2-1, pieces=5)
K[1:5,1:5]
rownames(Markers)<-paste("1", 1:nrow(Markers),sep="_")
rownames(Markers2)<-paste("1", (nrow(Markers)+1):(nrow(Markers)+ nrow(Markers2)), sep="_")
rownames(K2)<-colnames(K2)<-c(rownames(Markers),rownames(Markers2))
rownames(K)<-colnames(K)<-c(rownames(Markers))

which.max(Markers%*%markereffects)
markermap=as.matrix(data.frame(chr=rep(1,nmarkers),
pos=seq(0,1,length=nmarkers)))

colnames(Markers)<-1:nmarkers

gasols<-getGasolutionsFrontier(Markers=Markers,Markers2=Markers2, K=K2, markereffects,markermap=markermap,nmates=10,npopGA=100, nitGA=100, mc.cores=1, mutprob=0.999, nolself= TRUE, method=3,
type=2L, generation=1L, plotiters= TRUE)

##plot results
pairs(gasols4[[1]])

##Use plotGM.
```
getGaSolutionsFrontierMultiTrait

plotGM(GMsols=gasols4, type="3D", traitnum=1)
plotGM(GMsols=gasols4, type="SOM", traitnum=1)

## End(Not run)

getGaSolutionsFrontierMultiTrait

Description

Generalizes the approach in getGaSolutions to multiple traits specified by a list of marker effects.

Usage

getGaSolutionsFrontierMultiTrait(Markers, Markers2=NULL, K,
markereffectslist, markermapi, nmates=NULL, npopGA, nitGA, mutprob,
mc. cores, noself=F, method=1, type=0L, generation=0L, plotiters=F)

Arguments

Markers The matrix of markers rows corresponding to individuals and columns for markers, the markers scores are coded as -1,0,1. (For Method=3 the markers are coded as probabilities between 0 and 1.)
Markers2 The matrix of markers rows corresponding to individuals and columns for markers, the markers scores are coded as -1,0,1. (For Method=3 the markers are coded as probabilities between 0 and 1.)
K symmetric genomic relationship matrix, the order of the row and columns of this matrix should follow the order of genotypes in the rows of rbind(Markers, Markers2).
markermapi a map for markers. two columns, firts column is named chr second named pos for the chromosome and position of the markers specified above
markereffectslist effects of markers for several traits given as a list
nmates number of mates to select, default value is NULL (number of mates is equal to number of genotypes)
npopGA genetic algorithm parameter: number of solutions generated at each cycle of the GA
nitGA genetic algorithm parameter: number of GA cycles before algorithm stops
mutprob genetic algorithm parameter: mutation probability
mc. cores genetic algorithm parameter: number of cores to use
noself Is selfing allowed? (TRUE or FALSE)
method Which method to use? (1,2,3) See Details.
getGaSolutionsFrontierMultiTrait

type Only for method=2. Type of crosses (0 (DH), 1 (RISELF)).
generation Only for method=2. Generation at which the cross variances are calculated.
plotiters genetic algorithm parameter: if TRUE the value of the objective function over iterations will be plotted

Details

This program uses genetic algorithm to produce a number of solutions on the frontier curve simultaneously for the multi-objective optimization problem which is defined by minimization of 

\[-Gain_j(P_{32}), -Usefulness_j(P_{32})\text{ for } j = 1, 2, ..., ntraits \text{ and } Inbreeding(P_{32})\text{ with respect to } P_{32}.

Gain(P) for a mating design P is calculated as \(Pg\) where g is the vector of genomically estimated breeding values for the parents.

Inbreeding(P) for a mating design P is calculated as \(trace(KP^TP' + D(P))\) where K is the matrix of genomically estimated relationship matrix for the parents and D(P) is a diagonal matrix for adjustment of the parent relationship matrix to progeny relationship matrix.

Usefulness(P) measures the variance of a mate pair. The average or the sum of usefulnesses for all pairs in a mating plan can be used to measure the usefulness of a mating plan. There are three options for the calculation of usefulness. Method 1 uses the calculations in "Efficient Breeding by Genomic Mating". Method 2 uses the calculations in "Genetic gain increases by applying the usefulness criterion with improved variance prediction in selection of crosses" without the estimation variance terms. Method 2 comes with two types (DH (type=0) or riself (type=1)) and each of these types can be applied for progeny at a specified "generation". Method 3 is for polyploid organisms, where the marker data is recorded as proportions of alleles at genomewide loci.

Value

Returns a list with two elements: the first element in this list is a list of solutions found on the frontier, the second element is the matrix of criterion values (Gain, Usefulness, and Inbreeding) corresponding to these solutions.

Author(s)

Deniz Akdemir, Julio Isidro Sanchínez, Hanna Haikka, Itaraju Baracuhý Brum

References

Lehermeier at al. "Genetic gain increases by applying the usefulness criterion with improved variance prediction in selection of crosses" Genetics (2017).
Examples

```r
## Not run:
library("GenomicMating")

N=10

nmarkers=200
Markers<-c()
for (i in 1:N){
  Markers<-rbind(Markers,rbinom(nmarkers, 2,.1)-1)
}

Markers2<-c()
for (i in 1:N){
  Markers2<-rbind(Markers2,rbinom(nmarkers, 2,.1)-1)
}

markereffects<-rep(0,nmarkers)
markereffects[sample(1:nmarkers,nmarkers/2)]<-rnorm(nmarkers/2)

Markers[1:5,1:5]

K=Amat.pieces(rbind(Markers), pieces=5)
K2=Amat.pieces(rbind(Markers,Markers2), pieces=5)

rownames(Markers)<-paste("l", 1:nrow(Markers), sep="_")
rownames(Markers2)<-paste("l", (nrow(Markers)+1):((nrow(Markers)+nrow(Markers2))), sep="_")

rownames(K2)<-colnames(K2)<-c(rownames(Markers),rownames(Markers2))
rownames(K)<-colnames(K)<-c(rownames(Markers))

# Two sets of marker effects
markereffects2<-rep(0,nmarkers)
markereffects2[sample(1:nmarkers,nmarkers/2)]<-rnorm(nmarkers/2)
markereffects2<-rep(0,nmarkers)
markereffects2[sample(1:nmarkers,nmarkers/2)]<-rnorm(nmarkers/2)

gasols4<-getGaSolutionsFrontierMultiTrait(Markers=Markers,
Markers2=Markers2,K=K2,
markereffectslist=list(markereffects,markereffects2),
markermmap=markermmap,nmates=20, npopGA=100, nitGA=10,
mc.cores=1, mutprob=0.99,method=2,
type=0, generation=3, plotiters= TRUE)

str(gasols4)
gasols4[[1]][1:5,]

## End(Not run)
```
Description

A generalization of the single trait mating approach in getGaSolutionsFrontier. The usefulness statistic is calculated using the simcross function in the qtl package.

Usage

```r
getGasolutionsFrontierMultiTraitSimcross(markersL, markersR=NULL, K, map, markereffectslist, nmates=NULL, nSim = 5, npopGA, nitGA, mutprob, mc.cores, noself = FALSE, simtype = "riself", plotiters = FALSE)
```

Arguments

- **Markers**
  - The matrix of markers rows corresponding to individuals and columns for markers, the markers scores are coded as -1, 0, 1. (For Method=3 the markers are coded as probabilities between 0 and 1.)

- **Markers2**
  - The matrix of markers rows corresponding to individuals and columns for markers, the markers scores are coded as -1, 0, 1. (For Method=3 the markers are coded as probabilities between 0 and 1.)

- **K**
  - symmetric genomic relationship matrix, the order of the row and columns of this matrix should follow the order of genotypes in the rows of `rbind(markersL, markersR)`.

- **map**
  - a map for markers. two columns, first column is named chr second named pos for the chromosome and position of the markers specified above.

- **markereffectslist**
  - effects of markers for several traits given as a list.

- **nmates**
  - number of mates to select, default value is NULL (number of mates is equal to number of genotypes).

- **nSim**
  - number of progeny simulated for each pair.

- **npopGA**
  - genetic algorithm parameter: number of solutions generated at each cycle of the GA.

- **nitGA**
  - genetic algorithm parameter: number of GA cycles before algorithm stops.

- **mutprob**
  - genetic algorithm parameter: mutation probability.

- **mc.cores**
  - genetic algorithm parameter: number of cores to use.

- **noself**
  - Default is FALSE. Specifies if selfing is allowed.

- **simtype**
  - Default is "riself". Argument passed to simcross, not all types work with the GenomicMating package.

- **plotiters**
  - Logical. Default is FALSE. Iterations are plotted if TRUE.
getGaSolutionsFrontierMultiTraitSimcross

Details

This program uses genetic algorithm to produce a number of solutions on the frontier curve simultaneously for the multi-objective optimization problem which is defined by minimization of $-\text{Gain}(P_{32})$, $-\text{Usefulness}(P_{32})$ and $\text{Inbreeding}(P_{32})$ with respect to $P_{32}$.

Value

Returns a list with two elements: the first element in this list is a list of solutions found on the frontier, the second element is the matrix of criterion values (Gain, Usefulness, and Inbreeding) corresponding to these solutions.

Author(s)

Deniz Akdemir, Julio Isidro Sanch\'ez, Hanna Haikka, Itaraju Baracuhy Brum

References

Lehermeier at al. "Genetic gain increases by applying the usefulness criterion with improved variance prediction in selection of crosses" Genetics (2017).

Examples

```r
## Not run:
library(GenomicMating)
N=10
nmarkers=200
Markers<-c()
for (i in 1:N){
  Markers<-rbind(Markers,rbinom(nmarkers, 2,.1)-1)
}
Markers2<-c()
for (i in 1:N){
  Markers2<-rbind(Markers2,rbinom(nmarkers, 2,.1)-1)
}
markereffects<-rep(0,nmarkers)
markereffects[sample(1:nmarkers,nmarkers/2)]<-rnorm(nmarkers/2)
Markers[1:5,1:5]
library(parallel)
K=Amat.pieces(rbind(Markers), pieces=5)
K2=Amat.pieces(rbind(Markers,Markers2), pieces=5)
K[1:5,1:5]
rownames(Markers)<-paste("1", 1:nrow(Markers),sep="_")
rownames(Markers2)<-paste("1", (nrow(Markers)+1):(nrow(Markers)+nrow(Markers2)),sep="_")
```
getOptParentalProportions

**Description**
Parental proportions to balance gains and inbreeding

**Usage**
getOptParentalProportions(amat, gebvs, lambda, ul)

**Arguments**
- `amat` Additive genomic relationship matrix for a set of individuals
- `gebvs` Estimated breeding values in a vector, listed in the same order as they were in `amat`
getOptParentalProportions

lambda relative importance of inbreeding. 0 <= lambda <= 1.
ul maximum proportion assigned to a single genotype. 0 <= ul <= 1.

Value
A data frame with parental proportions and values of "lambda", "Gain", "Inbreeding", "G/I ratio".

Author(s)
Deniz Akdemir, Julio Isidro Sanchez, Hanna Haikka, Itaraju Baracuhy Brum

References

Examples
library(genomicmating)
set.seed(12345)
N=20
nmmarkers=100
Markers<-c()
for (i in 1:N){
    Markers<-rbind(Markers, rbinom(nmarkers, 2,.1)-1)
}

markereffects<-rep(0,nmarkers)
markereffects[sample(1:nmarkers,nmarkers/2)]<-rnorm(nmarkers/2)
Markers[1:5,1:5]
#library(parallel)
K=Amat.pieces(rbind(Markers), pieces=5)
rownames(Markers)<-paste("1", 1:nrow(Markers), sep="_")
rownames(K)<-colnames(K)<-c(rownames(Markers))
which.max(Markers%%markereffects)
colnames(Markers)<-1:nmarkers

oprop<-getOptParentalProportions(Amat=K,
gebv=Markers%%markereffects, lambda=.8, ul=1)
pout<-plotOPFrontier(Amat=K,
gebv=Markers%%markereffects, ul=1, identify=FALSE)
round(oprop,3)
what<-Markers%%markereffects
plotGM

Description
For plotting GA results. See examples.

Usage
plotGM(GMsols, type="3D", plotly = FALSE, idealsol=NULL, traitnum=1)

Arguments
- type: Options are "3D", "SOM", "SOM2".
- plotly: Logical, default is FALSE. Uses plotly for 3D plot if plotly is installed.
- idealsol: For coloring the plot. Defaults is NULL. Otherwise, a vector of the same length as the GMsols statistics.
- traitnum: which trait (order of trait in the markereffectslist).

Details
See examples

Value
NULL

Author(s)
Deniz Akdemir, Julio Isidro Sanchez, Hanna Haikka, Itaraju Baracuhy Brum
References


Examples

```r
## Not run:

library(genomicmating)

### for method 3 polyploid. Markers need to be coded between 0 and 1.
N=20
nmarkers=100
Markers<-c()
for (i in 1:N){
  Markers<-rbind(Markers,runif(nmarkers))
}

Markers2<-c()
for (i in 1:N){
  Markers2<-rbind(Markers2,runif(nmarkers))
}

markereffects<-rep(0,nmarkers)
markereffects[sample(1:nmarkers,nmarkers/2)]<-rnorm(nmarkers/2)
Markers[1:5,1:5]

#library(parallel)
K=Amat.pieces(rbind(Markers)*2-1, pieces=5)

K2=Amat.pieces(rbind(Markers,Markers2)*2-1, pieces=5)
K[1:5,1:5]

rownames(Markers)<-paste("1", 1:nrow(Markers), sep="_")
rownames(Markers2)<-paste("1", (nrow(Markers)+1):nrow(Markers+markers2)), sep="_")
rownames(K2)<-colnames(K2)<-c(rownames(Markers),rownames(Markers2))
rownames(K)<-colnames(K)<-c(rownames(Markers))

which.max(Markers%*%markereffects)
markermmap=as.matrix(data.frame(chr=rep(1,nmarkers),pos=seq(0,1,length=nmarkers)))

columns(Markers)<-1:nmarkers

gasol4<-getGasolutionsFrontier(Markers=Markers,Markers2=Markers2, K=K2,
markereffects,markermmap=markermmap,nnames=10,npopGA=100, nitGA=100,
mc.cores=1, mutprob=0.999, noself= TRUE, method=3,
type=2L, generation=1L, plotiters= TRUE)

###plot results
```
pairs(gasols4[[1]])

### Use plotGM.

plotGM(GMsols=gasols4, type="3D", traitnum=1)
plotGM(GMsols=gasols4, type="SOM", traitnum=1)

## End(Not run)
Index

Amat.pieces, 2

calculatecrossvalue (getGaSolutions), 3

GenomicMating (GenomicMating-package), 2
GenomicMating-package, 2
getGaSolutions, 3
getGaSolutionsFrontier, 8
getGaSolutionsFrontierMultiTrait, 11
getGaSolutionsFrontierMultiTraitSimcross, 14
getOptParentalProportions, 16
getstatsfromsim (getGaSolutions), 3
getstatsM1 (getGaSolutions), 3
getstatsM2 (getGaSolutions), 3
getstatsM3 (getGaSolutions), 3
Kmatfunc (getGaSolutions), 3
mapfunct (getGaSolutions), 3
pairs3d (getGaSolutions), 3
par.name (getGaSolutions), 3
par.position (getGaSolutions), 3
plotGM, 18
plotOPFrontier
  (getOptParentalProportions), 16
tails (getGaSolutions), 3