Package 'IPLGP'

September 8, 2021

Type Package

Title Identification of Parental Lines via Genomic Prediction

Version 1.3.0

Description Combining genomic prediction with Monte Carlo simulation, three different strategies are implemented to select parental lines for multiple traits in plant breeding. The selection strategies include (i) GEBV-O considers only genomic estimated breeding values (GEBVs) of the candidate individuals; (ii) GD-O considers only genomic diversity (GD) of the candidate individuals; and (iii) GEBV-GD considers both GEBV and GD. The above method can be seen in Chung PY, Liao CT (2020) <doi:10.1371/journal.pone.0243159>. Multi-trait genomic best linear unbiased prediction (MT-GBLUP) model is used to simultaneously estimate GEBVs of the target traits, and then a selection index is adopted to evaluate the composite performance of an individual.

Imports ggplot2, sommer, grDevices, stats

License GPL-2

Encoding UTF-8

RoxygenNote 7.1.1

NeedsCompilation no

Author Ping-Yuan Chung [cre],
Chen-Tuo Liao [aut]

Maintainer Ping-Yuan Chung <r06621204@ntu.edu.tw>

Repository CRAN

Date/Publication 2021-09-08 10:40:02 UTC

R topics documented:

`GA.Dscore` *Search For A Subset With The Highest D-score*

Description

Search for an optimal subset of the candidate individuals such that it achieves the highest D-score by genetic algorithm (GA).

Usage

```
GA.Dscore(
  K,
  size,
  keep = c(),
  n0 = size,
  mut = 3,
  cri = 10000,
  console = FALSE
)
```

Arguments

`K`	matrix. An n*n matrix denotes the genomic relationship matrix of the n candidate individuals, where n > 4.
`size`	integer. An integer denotes the size of the subset, note that 3 < size < n.
`keep`	vector. A vector indicates those candidate individuals which will be retained in the subset before the search. The length of keep must be less than size.
`n0`	integer. An integer indicates the number of chromosomes (solutions) in the genetic algorithm, note that n0 > 3.
`mut`	integer. An integer indicates the number of mutations in the genetic algorithm, note that mut < size.
`cri`	integer. An integer indicates the stopping criterion, note that cri < 1e+06. The genetic algorithm will stop if the number of iterations reaches cri.
`console`	logical. A logical variable, if console is set to be TRUE, the searching process will be shown in the R console.

Value

`subset`	The optimal subset with the highest D-score.
`D.score`	The D.score of the optimal subset.
`time`	The number of iterations.

References

Chung PY, Liao CT. 2020. Identification of superior parental lines for biparental crossing via genomic prediction. PLoS ONE 15(12):e0243159.

Ou JH, Liao CT. 2019. Training set determination for genomic selection. Theor Appl Genet. 132:2781-2792.

Examples

```
# generate simulated data
geno.test <- matrix(sample(c(1, -1), 600, replace = TRUE), 20, 30)
K.test <- geno.test%*%t(geno.test)/ncol(geno.test)

# run with no specified individual
result1 <- GA.Dscore(K.test, 6, cri = 1000, console = TRUE)
result1

# run with some specified individuals
result2 <- GA.Dscore(K.test, 6, keep = c(1, 5, 10), cri = 1000, console = TRUE)
result2
```

GBLUP.fit *Muti-trait GBLUP Model*

Description

Built the muti-trait GBLUP model using the phenotypic and genotypic data of a training population by 'mmer' from R package 'sommer'. Then, output the fitted values of the training population.

Usage

```
GBLUP.fit(t1, t2, t3, t4, t5, geno = NULL, K = NULL)
```

Arguments

t1	vector. The phenotype of trait1. The missing value must be coded as NA. The length of all triat must be the same.
t2	vector. The phenotype of trait2. The missing value must be coded as NA. The length of all triat must be the same.
t3	vector. The phenotype of trait3. The missing value must be coded as NA. The length of all triat must be the same.
t4	vector. The phenotype of trait4. The missing value must be coded as NA. The length of all triat must be the same.
t5	vector. The phenotype of trait5. The missing value must be coded as NA. The length of all triat must be the same.
geno	matrix. An n*p matrix with n individuals and p markers of the training population. The markers must be coded as 1, 0, or -1 for alleles AA, Aa, or aa. The missing value must have been already imputed.

K matrix. An n*n matrix denotes the genomic relationship matrix of the training population if geno is set to be NULL.

Value

The fitted values of the training population.

Note

Due to restrictions on the use of the funtion 'mmer', if an unknown error occurs during use, please try to input the phenotype data as the format shown in the example.

References

Habier D, Fernando RL, Dekkers JCM. 2007. The impact of genetic relationship information on genome-assisted breeding values. Genetics 177:2389-2397.

VanRaden PM. 2008. Efficient methods to compute genomic predictions. J Dairy Sci. 91:4414-4423.

See Also

mmer

Examples

```
# generate simulated data
t1 <- rnorm(50,30,10)
t2 <- rnorm(50,10,5)
t3 <- rnorm(50,20,20)
t4 <- NULL
t5 <- NULL

# run with the marker score matrix
geno.test <- matrix(sample(c(1, -1), 5000, replace = TRUE), 50, 100)
result1 <- GBLUP.fit(t1, t2, t3, t4, t5, geno = geno.test)
result1

# run with the genomic relationship matrix
K.test <- geno.test%*%t(geno.test)/ncol(geno.test)
result2 <- GBLUP.fit(t1, t2, t3, t4, t5, K = K.test)
result2
```

---

output.best *Summary For The Best Individuals*

Description

Output the GEBV average curves and the summary statistics for the best individuals selected over generations.

Usage

```
output.best(result, save.pdf = FALSE)
```

Arguments

result	list. The data list of the output from simu.GEBVO, simu.GDO, or simu.GEBVGD.
save.pdf	logical. A logical variable, if save.pdf is set to be TRUE, the pdf file of plots will be saved in the working directory instead of being shown in the console.

Value

The GEBV averages of the best individuals among the repetitions over generations for each trait.

Note

The figure output contains the plots of GEBV averages of the best individuals selected over generations for each trait. If save.pdf is set to be TRUE, the pdf file of plots will be saved in the working directory instead of being shown in the console.

References

Chung PY, Liao CT. 2020. Identification of superior parental lines for biparental crossing via genomic prediction. PLoS ONE 15(12):e0243159.

See Also

simu.GEBVO simu.GDO simu.GEBVGD ggplot

Examples

```
# generate simulated data
set.seed(2000)
t1 <- rnorm(10,30,10)
t2 <- rnorm(10,10,5)
t3 <- NULL
t4 <- NULL
t5 <- NULL
geno.test <- matrix(sample(c(1, -1), 200, replace = TRUE), 10, 20)
marker.test <- cbind(rep(1:2, each=10), rep(seq(0, 90, 10), 2))
fit <- GBLUP.fit(t1, t2, t3, t4, t5, geno = geno.test)

geno.candidate <- matrix(sample(c(1,-1), 300, replace = TRUE), 15, 20)

# run
result <- simu.GEBVO(fit, geno.test, marker.test, geno.candidate,
nprog = 5, nsele = 10, ngen = 5, nrep = 5)

# summary for the best individuals
output <- output.best(result)
output
```

output.gain *Summary For Genetic Gain*

Description

Output the GEBV average of parental lines, the GEBV average of the last generation in simulation process, and the genetic gain average over repetitions for each target trait.

Usage

```
output.gain(result)
```

Arguments

result	list. The data list of the output from simu.GEBVO, simu.GDO, or simu.GEBVGD.

Value

The output contains the table of the GEBV average of parental lines, the GEBV average of the last generation in simulation process, and the genetic gain average over repetitions for each target trait.

References

Chung PY, Liao CT. 2020. Identification of superior parental lines for biparental crossing via genomic prediction. PLoS ONE 15(12):e0243159.

See Also

simu.GEBVO simu.GDO simu.GEBVGD

Examples

```
# generate simulated data
set.seed(2000)
t1 <- rnorm(10,30,10)
t2 <- rnorm(10,10,5)
t3 <- NULL
t4 <- NULL
t5 <- NULL
geno.test <- matrix(sample(c(1, -1), 200, replace = TRUE), 10, 20)
marker.test <- cbind(rep(1:2, each=10), rep(seq(0, 90, 10), 2))
fit <- GBLUP.fit(t1, t2, t3, t4, t5, geno = geno.test)

geno.candidate <- matrix(sample(c(1,-1), 300, replace = TRUE), 15, 20)

# run
result <- simu.GEBVO(fit, geno.test, marker.test, geno.candidate,
nprog = 5, nsele = 10, ngen = 5, nrep = 5)
```

```
# summary for genetic gain
output <- output.gain(result)
output
```

---

phe.sd *Standardize Phenotypic Values*

Description

Standardize the phenotypic values of all the target traits from a training population. Then, output the standardized phenotypic values, the mean vector, and the standard deviation vector of the target traits.

Usage

```
phe.sd(phe)
```

Arguments

phe	matrix. An n*t matrix with n individuals and t traits, denotes the phenotypic values. The missing value must be coded as NA.

Value

standardize.phe	An n*t matrix contains the standardized phenotypic values.
mu	A vector with length t contains the averages of the phenotypic values of the t target traits.
sd	A vector with length t contains the standard deviations of the phenotypic values of the t target traits.

Examples

```
# generate simulated data
phe.test <- data.frame(trait1 = rnorm(50,30,10), trait2 = rnorm(50,10,5), trait3 = rnorm(50,20,20))

# run and output
result <- phe.sd(phe.test)
result
```

simu.gamete *Simulate The Genotypic Values Of A Gamete*

Description

Generate the genotypic values of a gamete from the genotypic data of its parents by Monte Carlo simulation. The recombination rate is caculate by Haldane's mapping function.

Usage

```
simu.gamete(marker)
```

Arguments

marker	data frame. A p*4 data frame whose first column indicates the chromosome number to which a marker belongs; second column indicates the position of the marker in centi-Morgan (cM); and 3rd and 4th columns indicates the genotype of the marker (numeric or character).

Value

The SNP sequence of gamete.

References

Haldane J.B.S. 1919. The combination of linkage values and the calculation of distance between the loci for linked factors. Genetics 8: 299–309.

Examples

```
# generate simulated data
marker.test <- data.frame(c(1,1,1,1,1,2,2,2,2,2),c(10,20,30,40,50,10,20,30,40,50),
c("A","T","C","G","A","A","G","A","T","A"),c("A","A","G","C","T","A","G","T","T","A"))

# run
simu.gamete(marker.test)
```

simu.GDO *GD-O Strategy*

Description

Identify parental lines based on GD-O strategy and simulate their offsprings.

Usage

```
simu.GDO(
  fitted.t,
  geno.t,
  marker,
  geno.c = NULL,
  npl = NULL,
  better.c = FALSE,
  weight = NULL,
  direction = NULL,
  nprog = 50,
  nsele = NULL,
  ngen = 10,
  nrep = 30,
  cri = 10000,
  console = TRUE
)
```

Arguments

fitted.t	matrix. An n*t matrix denotes the fitted values of each traits of the training population. The missing value must have been already imputed.
geno.t	matrix. An n*p matrix denotes the marker score matrix of the training population. The markers must be coded as 1, 0, or -1 for alleles AA, Aa, or aa. The missing value must have been already imputed.
marker	matrix. A p*2 matrix whose first column indicates the chromosome number to which a marker belongs; and second column indicates the position of the marker in centi-Morgan (cM).
geno.c	matrix. An nc*p matrix denotes the marker score matrix of the candidate population with nc individuals and p markers. The markers must be coded as 1, 0, or -1 for alleles AA, Aa, or aa. The missing value must have been already imputed. If geno.c is set to be NULL, the candidate population is exactly the training population.
npl	integer. An integer indicates the number of individuals who will be chosen as the parental lines. If npl = NULL, it will be 4 times the number of traits.
better.c	logical. A logical variable, if better.c is set to be TRUE, the candidate individuals with GEBVs better than average for all the target traits will comprise the candidate set. Otherwise, all the candidate individuals will comprise the candidate set.
weight	vector. A vector with length t indicates the weights of target traits in selection index. If weight is set to be NULL, the equal weight will be assigned to all the target traits.
direction	vector. A vector with length t indicates the selecting directions for target traits. The elements of direction are 1, or 0 representing the rule that the larger the better; or the smaller the better. If direction is set to be NULL, the selecting direction will be the same as weight.

nprog	integer. An integer indicates the number of progenies which will be produced for each of the best individuals at every generation.
nsele	integer. An integer indicates the number of the best individuals which will be selected at each generation. If nsele is set to be NULL, the number will be the same as the number of F1 individuals.
ngen	integer. An integer indicates the number of generations in the simulation process.
nrep	integer. An integer indicates the number of repetitions in the simulation process.
cri	integer. An integer indicates the stopping criterion, note that cri < 1e+06. The genetic algorithm will stop if the number of iterations reaches cri.
console	logical. A logical variable, if console is set to be TRUE, the simulation process will be shown in the R console.

Value

method	The GD-O strategy.
weight	The weights of target traits in selection index.
direction	The selecting directions of target traits in selection index.
mu	The mean vector of target traits.
sd	The standard deviation vector of target traits.
GEBV.value	The GEBVs of target traits in each generation and each repetition.
parental.lines	The IDs and D-score of parental lines selected in each repetition.
suggested.subset	The most frequently selected parental lines by this strategy.

Note

The function output.best and output.gain can be used to summarize the result.

The fitted value data in the input data can be obtained by the function GBLUP.fit and mmer, that can be seen in the Examples shown below.

References

Chung PY, Liao CT. 2020. Identification of superior parental lines for biparental crossing via genomic prediction. PLoS ONE 15(12):e0243159.

See Also

mmer GBLUP.fit GA.Dscore simu.gamete simu.GEBVO simu.GEBVGD output.best output.gain

Examples

```
# generate simulated data
set.seed(2000)
t1 <- rnorm(10,30,10)
t2 <- rnorm(10,10,5)
t3 <- NULL
t4 <- NULL
t5 <- NULL
geno.test <- matrix(sample(c(1, -1), 200, replace = TRUE), 10, 20)
marker.test <- cbind(rep(1:2, each=10), rep(seq(0, 90, 10), 2))
fit <- GBLUP.fit(t1, t2, t3, t4, t5, geno = geno.test)

geno.candidate <- matrix(sample(c(1,-1), 300, replace = TRUE), 15, 20)

# run and output
result <- simu.GDO(fit, geno.test, marker.test, geno.candidate,
nprog = 5, nsele = 10, ngen = 5, nrep = 5, cri = 250)
result$suggested.subset



# other method: use mmer to obtain the fitted value
## Not run:
set.seed(2000)
t1 <- rnorm(10,30,10)
t2 <- rnorm(10,10,5)
phe <- cbind(t1, t2)
nt <- ncol(phe)
geno.test <- matrix(sample(c(1, -1), 200, replace = TRUE), 10, 20)
marker.test <- cbind(rep(1:2, each=10), rep(seq(0, 90, 10), 2))
rownames(geno.test) <- 1:nrow(geno.test)
id <- rownames(geno.test)
K0 <- geno.test%*%t(geno.test)/ncol(geno.test)
diag(K0) <- 1

dat <- data.frame(id, phe)
fit0 <- sommer::mmer(cbind(t1, t2)~1,
      random = ~sommer::vs(id, Gu = K0, Gtc = sommer::unsm(nt)),
      rcov = ~sommer::vs(units, Gtc = sommer::unsm(nt)),
      data = dat,
      tolparinv = 0.01)

u0 <- fit0$U$`u:id`
fit <- matrix(unlist(u0), ncol = nt)
colnames(fit) <- names(u0)

fit <- fit+matrix(fit0$fitted[1,], nrow(fit), nt, byrow = TRUE)
fit <- fit[order(as.numeric(names((u0[[1]])))),]

## End(Not run)
```

simu.GEBVGD *GEBV-GD Strategy*

Description

Identify parental lines based on GEBV-GD strategy and simulate their offsprings.

Usage

```
simu.GEBVGD(
  fitted.t,
  geno.t,
  marker,
  geno.c = NULL,
  npl = NULL,
  better.c = FALSE,
  npl.best = NULL,
  weight = NULL,
  direction = NULL,
  nprog = 50,
  nsele = NULL,
  ngen = 10,
  nrep = 30,
  cri = 10000,
  console = TRUE
)
```

Arguments

fitted.t	matrix. An n*t matrix denotes the fitted values of each traits of the training population. The missing value must have been already imputed.
geno.t	matrix. An n*p matrix denotes the marker score matrix of the training population. The markers must be coded as 1, 0, or -1 for alleles AA, Aa, or aa. The missing value must have been already imputed.
marker	matrix. A p*2 matrix whose first column indicates the chromosome number to which a marker belongs; and second column indicates the position of the marker in centi-Morgan (cM).
geno.c	matrix. An nc*p matrix denotes the marker score matrix of the candidate population with nc individuals and p markers. The markers must be coded as 1, 0, or -1 for alleles AA, Aa, or aa. The missing value must have been already imputed. If geno.c is set to be NULL, the candidate population is exactly the training population.
npl	integer. An integer indicates the number of individuals who will be chosen as the parental lines. If npl = NULL, it will be 4 times the number of traits.

better.c	logical. A logical variable, if better.c is set to be TRUE, the candidate individuals with GEBVs better than average for all the target traits will comprise the candidate set. Otherwise, all the candidate individuals will comprise the candidate set.
npl.best	integer. A integer indicates the numbers of the candidate individuals with the top GEBV index will be retained. If npl.best is set to be NULL, it will be 2 times the number of traits.
weight	vector. A vector with length t indicates the weights of target traits in selection index. If weight is set to be NULL, the equal weight will be assigned to all the target traits.
direction	vector. A vector with length t indicates the selecting directions for target traits. The elements of direction are 1, or 0 representing the rule that the larger the better; or the smaller the better. If direction is set to be NULL, the selecting direction will be the same as weight.
nprog	integer. An integer indicates the number of progenies which will be produced for each of the best individuals at every generation.
nsele	integer. An integer indicates the number of the best individuals which will be selected at each generation. If nsele is set to be NULL, the number will be the same as the number of F1 individuals.
ngen	integer. An integer indicates the number of generations in the simulation process.
nrep	integer. An integer indicates the number of repetitions in the simulation process.
cri	integer. An integer indicates the stopping criterion, note that cri < 1e+06. The genetic algorithm will stop if the number of iterations reaches cri.
console	logical. A logical variable, if console is set to be TRUE, the simulation process will be shown in the R console.

Value

method	The GEBV-GD strategy.
weight	The weights of target traits in selection index.
direction	The selecting directions of target traits in selection index.
mu	The mean vector of target traits.
sd	The standard deviation vector of target traits.
GEBV.value	The GEBVs of target traits in each generation and each repetition.
parental.lines	The IDs and D-score of parental lines selected in each repetition.
suggested.subset	The most frequently selected parental lines by this strategy.

Note

The function output.best and output.gain can be used to summarize the result.

The fitted value data in the input data can be obtained by the function GBLUP.fit and mmer, that can be seen in the Examples shown below.

References

Chung PY, Liao CT. 2020. Identification of superior parental lines for biparental crossing via genomic prediction. PLoS ONE 15(12):e0243159.

See Also

mmer GBLUP.fit GA.Dscore simu.gamete simu.GEBVO simu.GDO output.best output.gain

Examples

```
# generate simulated data
set.seed(2000)
t1 <- rnorm(10,30,10)
t2 <- rnorm(10,10,5)
t3 <- NULL
t4 <- NULL
t5 <- NULL
geno.test <- matrix(sample(c(1, -1), 200, replace = TRUE), 10, 20)
marker.test <- cbind(rep(1:2, each=10), rep(seq(0, 90, 10), 2))
fit <- GBLUP.fit(t1, t2, t3, t4, t5, geno = geno.test)

geno.candidate <- matrix(sample(c(1,-1), 300, replace = TRUE), 15, 20)

# run and output
result <- simu.GEBVGD(fit, geno.test, marker.test, geno.candidate,
nprog = 5, nsele = 10, ngen = 5, nrep = 5, cri = 250)
result$suggested.subset



# other method: use mmer to obtain the fitted value
## Not run:
set.seed(2000)
t1 <- rnorm(10,30,10)
t2 <- rnorm(10,10,5)
phe <- cbind(t1, t2)
nt <- ncol(phe)
geno.test <- matrix(sample(c(1, -1), 200, replace = TRUE), 10, 20)
marker.test <- cbind(rep(1:2, each=10), rep(seq(0, 90, 10), 2))
rownames(geno.test) <- 1:nrow(geno.test)
id <- rownames(geno.test)
K0 <- geno.test%*%t(geno.test)/ncol(geno.test)
diag(K0) <- 1

dat <- data.frame(id, phe)
fit0 <- sommer::mmer(cbind(t1, t2)~1,
      random = ~sommer::vs(id, Gu = K0, Gtc = sommer::unsm(nt)),
      rcov = ~sommer::vs(units, Gtc = sommer::unsm(nt)),
      data = dat,
      tolparinv = 0.01)

u0 <- fit0$U$`u:id`
```

```
fit <- matrix(unlist(u0), ncol = nt)
colnames(fit) <- names(u0)

fit <- fit+matrix(fit0$fitted[1,], nrow(fit), nt, byrow = TRUE)
fit <- fit[order(as.numeric(names((u0[[1]])))),]

## End(Not run)
```

---

simu.GEBVO *GEBV-O Strategy*

Description

Identify parental lines based on GEBV-O strategy and simulate their offsprings.

Usage

```
simu.GEBVO(
  fitted.t,
  geno.t,
  marker,
  geno.c = NULL,
  npl = NULL,
  weight = NULL,
  direction = NULL,
  nprog = 50,
  nsele = NULL,
  ngen = 10,
  nrep = 30,
  console = TRUE
)
```

Arguments

fitted.t	matrix. An n*t matrix denotes the fitted values of each traits of the training population. The missing value must have been already imputed.
geno.t	matrix. An n*p matrix denotes the marker score matrix of the training population. The markers must be coded as 1, 0, or -1 for alleles AA, Aa, or aa. The missing value must have been already imputed.
marker	matrix. A p*2 matrix whose first column indicates the chromosome number to which a marker belongs; and second column indicates the position of the marker in centi-Morgan (cM).
geno.c	matrix. An nc*p matrix denotes the marker score matrix of the candidate population with nc individuals and p markers. The markers must be coded as 1, 0, or -1 for alleles AA, Aa, or aa. The missing value must have been already imputed. If geno.c is set to be NULL, the candidate population is exactly the training population.

npl	integer. An integer indicates how many parental lines with the top GEBV index will be chosen from each trait. If npl is set to be NULL, there will be be 4 times the number of traits.
weight	vector. A vector with length t indicates the weights of target traits in selection index. If weight is set to be NULL, the equal weight will be assigned to all the target traits.
direction	vector. A vector with length t indicates the selecting directions for target traits. The elements of direction are 1, or 0 representing the rule that the larger the better; or the smaller the better. If direction is set to be NULL, the selecting direction will be the same as weight.
nprog	integer. An integer indicates the number of progenies which will be produced for each of the best individuals at every generation.
nsele	integer. An integer indicates the number of the best individuals which will be selected at each generation. If nsele is set to be NULL, the number will be the same as the number of F1 individuals.
ngen	integer. An integer indicates the number of generations in the simulation process.
nrep	integer. An integer indicates the number of repetitions in the simulation process.
console	logical. A logical variable, if console is set to be TRUE, the simulation process will be shown in the R console.

Value

method	The GEBV-O strategy.
weight	The weights of target traits in selection index.
direction	The selecting directions of target traits in selection index.
mu	The mean vector of target traits.
sd	The standard deviation vector of target traits.
GEBV.value	The GEBVs of target traits in each generation and each repetition.
parental.lines	The IDs and D-score of parental lines selected in each repetition.
suggested.subset	The most frequently selected parental lines by this strategy.

Note

The function output.best and output.gain can be used to summarize the result.

The fitted value data in the input data can be obtained by the function GBLUP.fit and mmer, that can be seen in the Examples shown below.

References

Chung PY, Liao CT. 2020. Identification of superior parental lines for biparental crossing via genomic prediction. PLoS ONE 15(12):e0243159.

See Also

mmer GBLUP.fit GA.Dscore simu.gamete simu.GDO simu.GEBVGD output.best output.gain

Examples

```
# generate simulated data
set.seed(2000)
t1 <- rnorm(10,30,10)
t2 <- rnorm(10,10,5)
t3 <- NULL
t4 <- NULL
t5 <- NULL
geno.test <- matrix(sample(c(1, -1), 200, replace = TRUE), 10, 20)
marker.test <- cbind(rep(1:2, each=10), rep(seq(0, 90, 10), 2))
fit <- GBLUP.fit(t1, t2, t3, t4, t5, geno = geno.test)

geno.candidate <- matrix(sample(c(1,-1), 300, replace = TRUE), 15, 20)

# run and output
result <- simu.GEBVO(fit, geno.t = geno.test, marker = marker.test,
geno.c = geno.candidate, nprog = 5, nsele = 10, ngen = 5, nrep = 5)
result$suggested.subset



# other method: use mmer to obtain the fitted value
## Not run:
set.seed(2000)
t1 <- rnorm(10,30,10)
t2 <- rnorm(10,10,5)
phe <- cbind(t1, t2)
nt <- ncol(phe)
geno.test <- matrix(sample(c(1, -1), 200, replace = TRUE), 10, 20)
marker.test <- cbind(rep(1:2, each=10), rep(seq(0, 90, 10), 2))
rownames(geno.test) <- 1:nrow(geno.test)
id <- rownames(geno.test)
K0 <- geno.test%*%t(geno.test)/ncol(geno.test)
diag(K0) <- 1

dat <- data.frame(id, phe)
fit0 <- sommer::mmer(cbind(t1, t2)~1,
      random = ~sommer::vs(id, Gu = K0, Gtc = sommer::unsm(nt)),
      rcov = ~sommer::vs(units, Gtc = sommer::unsm(nt)),
      data = dat,
      tolparinv = 0.01)

u0 <- fit0$U$`u:id`
fit <- matrix(unlist(u0), ncol = nt)
colnames(fit) <- names(u0)

fit <- fit+matrix(fit0$fitted[1,], nrow(fit), nt, byrow = TRUE)
fit <- fit[order(as.numeric(names((u0[[1]])))),]
```

```
## End(Not run)
```

Index