Package 'AlphaSimR'

March 15, 2019

Type Package

Title Breeding Program Simulations

Version 0.10.0

Date 2019-3-15

Description The successor to the 'AlphaSim' software for breeding program simulation [Faux et al. (2016) <doi:10.3835/plantgenome2016.02.0013>]. Used for stochastic simulations of breeding programs to the level of DNA sequence for every individual. Contained is a wide range of functions for modeling common tasks in a breeding program, such as selection and crossing. These functions allow for constructing simulations of highly complex plant and animal breeding programs via scripting in the R software environment. Such simulations can be used to evaluate overall breeding program performance and conduct research into breeding program design, such as implementation of genomic selection. Included is the 'Markovian Coalescent Simulator' ('MaCS') for fast simulation of biallelic sequences according to a population demographic history [Chen et al. (2009) <doi:10.1101/gr.083634.108>].

License MIT + file LICENSE

URL https://alphagenes.roslin.ed.ac.uk/wp/software/alphasimr/, https://bitbucket.org/hickeyjohnteam/alphasimr

Encoding UTF-8

LazyData true

Depends R (>= 3.3.0), methods, R6

Imports Rcpp (>= 0.12.7)

LinkingTo Rcpp, RcppArmadillo (>= 0.7.500.0.0), BH

RoxygenNote 6.1.0

Suggests knitr, rmarkdown, testthat

VignetteBuilder knitr

NeedsCompilation yes

Author Chris Gaynor [aut, cre] (<https://orcid.org/0000-0003-0558-6656>), Gregor Gorjanc [ctb] (<https://orcid.org/0000-0001-8008-2787>),

David Wilson [ctb],
Daniel Money [ctb] (<https://orcid.org/0000-0001-5151-3648>),
John Hickey [ctb] (<https://orcid.org/0000-0001-5675-3974>)

Maintainer Chris Gaynor <gaynor.robert@hotmail.com>

Repository CRAN

Date/Publication 2019-03-15 15:33:25 UTC

R topics documented:

---

aa *Additive-by-additive epistatic deviations*

Description

Returns additive-by-additive epistatic deviations for all traits

Usage

```
aa(pop, simParam = NULL)
```

Arguments

pop	an object of Pop-class
simParam	an object of SimParam

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=10, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)
SP$addTraitAD(10, meanDD=0.5)
SP$setVarE(h2=0.5)

#Create population
pop = newPop(founderPop, simParam=SP)
aa(pop, simParam=SP)
```

AlphaSimR *AlphaSimR: Breeding Program Simulations*

Description

The successor to the 'AlphaSim' software for breeding program simulation [Faux et al. (2016) <doi:10.3835/plantgenome2016.02.0013>]. Used for stochastic simulations of breeding programs to the level of DNA sequence for every individual. Contained is a wide range of functions for modeling common tasks in a breeding program, such as selection and crossing. These functions allow for constructing simulations of highly complex plant and animal breeding programs via scripting in the R software environment. Such simulations can be used to evaluate overall breeding program performance and conduct research into breeding program design, such as implementation of genomic selection. Included is the 'Markovian Coalescent Simulator' ('MaCS') for fast simulation of biallelic sequences according to a population demographic history [Chen et al. (2009) <doi:10.1101/gr.083634.108>].

Please see the introductionary vignette for instructions for using this package. The vignette can be viewed using the following command: `vignette("intro",package="AlphaSimR")`

bv *Breeding value*

Description

Returns breeding values for all traits

Usage

```
bv(pop, simParam = NULL)
```

Arguments

pop	an object of Pop-class
simParam	an object of SimParam

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=10, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)
SP$addTraitAD(10, meanDD=0.5)
SP$setVarE(h2=0.5)

#Create population
pop = newPop(founderPop, simParam=SP)
bv(pop, simParam=SP)
```

calcGCA *Calculate GCA*

Description

Calculate general combining ability of test crosses. Intended for output from hybridCross using the "testcross" option, but will work for any population.

Usage

```
calcGCA(pop, use = "pheno")
```

Arguments

pop	an object of Pop-class or HybridPop-class
use	tabulate either genetic values "gv", estimated breeding values "ebv", or phenotypes "pheno"

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=10, nChr=1, segSites=10, inbred=TRUE)

#Set simulation parameters
SP = SimParam$new(founderPop)
SP$addTraitA(10)

#Create population
pop = newPop(founderPop, simParam=SP)

#Make crosses for full diallele
pop2 = hybridCross(pop, pop, simParam=SP)
GCA = calcGCA(pop2, use="gv")
```

---

dd *Dominance deviations*

Description

Returns dominance deviations for all traits

Usage

```
dd(pop, simParam = NULL)
```

Arguments

pop	an object of Pop-class
simParam	an object of SimParam

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=10, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)
SP$addTraitAD(10, meanDD=0.5)
SP$setVarE(h2=0.5)

#Create population
pop = newPop(founderPop, simParam=SP)
dd(pop, simParam=SP)
```

ebv *Estimated breeding value*

Description

A wrapper for accessing the ebv slot

Usage

```
ebv(pop)
```

Arguments

pop	a Pop-class or similar object

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=10, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)
SP$addTraitAD(10, meanDD=0.5)
SP$setVarE(h2=0.5)

#Create population
pop = newPop(founderPop, simParam=SP)
pop@ebv = matrix(rnorm(pop@nInd), nrow=pop@nInd, ncol=1)
ebv(pop)
```

editGenome *Edit genome*

Description

Edits selected loci of selected individuals to a homozygous state for either the 1 or 0 allele. The gv slot is recalculated to reflect the any changes due to editing, but other slots remain the same.

Usage

```
editGenome(pop, ind, chr, segSites, allele, simParam = NULL)
```

Arguments

pop	an object of Pop-class
ind	a vector of individuals to edit
chr	a vector of chromosomes to edit. Length must match length of segSites.
segSites	a vector of segregating sites to edit. Length must match length of chr.
allele	either 0 or 1 for desired allele
simParam	an object of SimParam

Value

Returns an object of Pop-class

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=2, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)
SP$addTraitA(10)

#Create population
pop = newPop(founderPop, simParam=SP)

#Change individual 1 to homozygous for the 1 allele
#at locus 1, chromosome 1
pop2 = editGenome(pop, ind=1, chr=1, segSites=1,
                  allele=1, simParam=SP)
```

---

editGenomeTopQtl *Edit genome - the top QTL*

Description

Edits the top QTL (with the largest additive effect) to a homozygous state for the allele increasing. Only nonfixed QTL are edited The gv slot is recalculated to reflect the any changes due to editing, but other slots remain the same.

Usage

```
editGenomeTopQtl(pop, ind, nQtl, trait = 1, increase = TRUE,
  simParam = NULL)
```

Arguments

pop	an object of Pop-class
ind	a vector of individuals to edit
nQtl	number of QTL to edit
trait	which trait effects should guide selection of the top QTL
increase	should the trait value be increased or decreased
simParam	an object of SimParam

Value

Returns an object of Pop-class

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=2, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)
SP$addTraitA(10)

#Create population
pop = newPop(founderPop, simParam=SP)

#Change up to 10 loci for individual 1
pop2 = editGenomeTopQtl(pop, ind=1, nQtl=10, simParam=SP)
```

---

fastRRBLUP *Fast RR-BLUP*

Description

Solves an RR-BLUP model for genomic predictions given known variance components. This implementation is meant as a fast and low memory alternative to RRBLUP or RRBLUP2. Unlike the those functions, the fastRRBLUP does not fit fixed effects (other than the intercept) or account for unequal replication.

Usage

```
fastRRBLUP(pop, traits = 1, use = "pheno", snpChip = 1,
  useQtl = FALSE, maxIter = 1000, Vu = NULL, Ve = NULL,
  simParam = NULL, ...)
```

Arguments

pop	a Pop-class to serve as the training population
traits	an integer indicating the trait to model or a function of the traits returning a single value. Only univariate models are supported.
use	train model using phenotypes "pheno", genetic values "gv", estimated breeding values "ebv", breeding values "bv", or randomly "rand"
snpChip	an integer indicating which SNP chip genotype to use
useQtl	should QTL genotypes be used instead of a SNP chip. If TRUE, snpChip specifies which trait's QTL to use, and thus these QTL may not match the QTL underlying the phenotype supplied in traits.
maxIter	maximum number of iterations.
Vu	marker effect variance. If value is NULL, a reasonable value is chosen automatically.
Ve	error variance. If value is NULL, a reasonable value is chosen automatically.
simParam	an object of SimParam
...	additional arguments if using a function for traits

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=10, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)
SP$addTraitA(10)
SP$setVarE(h2=0.5)
SP$addSnpChip(10)

#Create population
pop = newPop(founderPop, simParam=SP)

#Run GS model and set EBV
ans = fastRRBLUP(pop, simParam=SP)
pop = setEBV(pop, ans, simParam=SP)

#Evaluate accuracy
cor(gv(pop), ebv(pop))
```

GCAsol-class *RR-BLUP GCA Solution*

Description

Extends LociMap-class to contain estimated effects from RRBLUP_GCA

Slots

`femaleEff` marker GCA for "female" pool

`maleEff` marker GCA for "male" pool

`fixEff` Estimates for fixed effects

`Vu` Estimated marker variances in order: female effects, male effects and SCA effects (`SCAsol-class` only)

`Ve` Estimated error variance

`LL` Log-likelihood

`iter` Number of iterations for convergence

`genicVarA` *Additive genic variance*

Description

Returns additive genic variance for all traits

Usage

```
genicVarA(pop, simParam = NULL)
```

Arguments

`pop`	an object of `Pop-class`
`simParam`	an object of `SimParam`

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=10, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)
SP$addTraitAD(10, meanDD=0.5)
SP$setVarE(h2=0.5)

#Create population
pop = newPop(founderPop, simParam=SP)
genicVarA(pop, simParam=SP)
```

genicVarAA *Additive-by-additive genic variance*

Description

Returns additive-by-additive epistatic genic variance for all traits

Usage

```
genicVarAA(pop, simParam = NULL)
```

Arguments

pop	an object of Pop-class
simParam	an object of SimParam

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=10, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)
SP$addTraitAD(10, meanDD=0.5)
SP$setVarE(h2=0.5)

#Create population
pop = newPop(founderPop, simParam=SP)
genicVarAA(pop, simParam=SP)
```

genicVarD *Dominance genic variance*

Description

Returns dominance genic variance for all traits

Usage

```
genicVarD(pop, simParam = NULL)
```

Arguments

pop	an object of Pop-class
simParam	an object of SimParam

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=10, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)
SP$addTraitAD(10, meanDD=0.5)
SP$setVarE(h2=0.5)

#Create population
pop = newPop(founderPop, simParam=SP)
genicVarD(pop, simParam=SP)
```

---

genicVarG *Total genic variance*

Description

Returns total genic variance for all traits

Usage

```
genicVarG(pop, simParam = NULL)
```

Arguments

pop	an object of Pop-class
simParam	an object of SimParam

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=10, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)
SP$addTraitAD(10, meanDD=0.5)
SP$setVarE(h2=0.5)

#Create population
pop = newPop(founderPop, simParam=SP)
genicVarG(pop, simParam=SP)
```

genParam *Sumarize genetic parameters*

Description

Calculates genetic and genic additive and dominance variances for an object of Pop-class

Usage

```
genParam(pop, simParam = NULL)
```

Arguments

pop	an object of Pop-class
simParam	an object of SimParam

Value

varA an nTrait by nTrait matrix of additive genetic variances

varD an nTrait by nTrait matrix of dominance genetic variances

varAA an nTrait by nTrait matrix of additive-by-additive genetic variances

varG an nTrait by nTrait matrix of total genetic variances

genicVarA an nTrait vector of additive genic variances

genicVarD an nTrait vector of dominance genic variances

genicVarAA an nTrait vector of additive-by-additive genic variances

genicVarG an nTrait vector of total genic variances

randGenicVarA an nTrait vector of additive genic variances assuming random mating

randGenicVarD an nTrait vector of dominance genic variances assuming random mating

randGenicVarAA an nTrait vector of additive-by-additive genic variances assuming random mating

randGenicVarG an nTrait vector of total genic variances assuming random mating

mu an nTrait vector of trait means

mu_HWE an nTrait vector of expected trait means under Hardy-Weinberg equilibrium

gv a matrix of genetic values with dimensions nInd by nTraits

bv a matrix of breeding values with dimensions nInd by nTraits

dd a matrix of dominance deviations with dimensions nInd by nTraits

aa a matrix of additive-by-additive deviations with dimensions nInd by nTraits

gv_mu an nTrait vector of trait means for genotype with all zeros

gv_a a matrix of additive genetic values with dimensions nInd by nTraits

gv_d a matrix of dominance genetic values with dimensions nInd by nTraits

gv_aa a matrix of additive-by-additive genetic values with dimensions nInd by nTraits

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=10, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)
SP$addTraitAD(10, meanDD=0.5)
SP$setVarE(h2=0.5)

#Create population
pop = newPop(founderPop, simParam=SP)
ans = genParam(pop, simParam=SP)
```

---

getQtlMap *Get QTL genetic map*

Description

Retrieves the genetic map for the QTL of a given trait.

Usage

```
getQtlMap(trait = 1, gender = "A", simParam = NULL)
```

Arguments

trait	an integer for the
gender	determines which gender specific map is returned. Options are "A" for average map, "F" for female map, and "M" for male map. All options are equivalent if not using gender specific maps.
simParam	an object of SimParam

Value

Returns a data.frame for the SNP map.

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=10, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)
SP$addTraitA(5)

#Pull SNP map
getQtlMap(trait=1, simParam=SP)
```

getSnpMap *Get SNP genetic map*

Description

Retrieves the genetic map for a given SNP chip.

Usage

```
getSnpMap(snpChip = 1, gender = "A", simParam = NULL)
```

Arguments

snpChip	an integer. Indicates which SNP chip's map to retrieve.
gender	determines which gender specific map is returned. Options are "A" for average map, "F" for female map, and "M" for male map. All options are equivalent if not using gender specific maps.
simParam	an object of SimParam

Value

Returns a data.frame for the SNP map.

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=10, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)
SP$addSnpChip(5)

#Pull SNP map
getSnpMap(snpChip=1, simParam=SP)
```

gv *Genetic value*

Description

A wrapper for accessing the gv slot

Usage

```
gv(pop)
```

Arguments

pop	a Pop-class or similar object

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=10, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)
SP$addTraitAD(10, meanDD=0.5)
SP$setVarE(h2=0.5)

#Create population
pop = newPop(founderPop, simParam=SP)
gv(pop)
```

---

hybridCross *Hybrid crossing*

Description

A convience function for hybrid plant breeding simulations. Allows for easy specification of a test cross scheme and/or creation of an object of HybridPop-class. Note that the HybridPop-class should only be used if the parents were created using the makeDH function or newPop using inbred founders. The id for new individuals is [mother_id]_[father_id]

Usage

```
hybridCross(females, males, crossPlan = "testcross",
  returnHybridPop = FALSE, simParam = NULL)
```

Arguments

females	female population, an object of Pop-class
males	male population, an object of Pop-class
crossPlan	either "testcross" for all possible combinantions or a matrix with two columns for designed crosses
returnHybridPop	should results be returned as HybridPop-class. If false returns results as Pop-class. Population must be fully inbred if TRUE.
simParam	an object of SimParam

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=2, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)

#Create population
pop = newPop(founderPop, simParam=SP)

#Make crosses for full diallele
pop2 = hybridCross(pop, pop, simParam=SP)
```

HybridPop-class *Hybrid population*

Description

A lightweight version of Pop-class for hybrid lines. Memory is saved by not storing genotypic data.

Usage

```
## S4 method for signature 'HybridPop'
x[i]

## S4 method for signature 'HybridPop'
c(x, ...)
```

Arguments

x	a 'HybridPop'
i	index of individuals
...	additional 'HybridPop' objects

Methods (by generic)

- [: Extract HybridPop using index or id
- c: Combine multiple HybridPops

Slots

nInd number of individuals

id an individual's identifier

mother the identifier of the individual's mother

`father` the identifier of the individual's father

`nTraits` number of traits

`gv` matrix of genetic values. When using GxE traits, gv reflects gv when w=0. Dimensions are nInd by nTraits.

`pheno` matrix of phenotypic values. Dimensions are nInd by nTraits.

`gxe` list containing GxE slopes for GxE traits

LociMap-class *Loci metadata*

Description

used for both SNPs and QTLs

Slots

`nLoci` total number of loci

`lociPerChr` number of loci per chromosome

`lociLoc` physical position of loci

---

makeCross *Make designed crosses*

Description

Makes crosses within a population using a user supplied crossing plan.

Usage

```
makeCross(pop, crossPlan, simParam = NULL)
```

Arguments

`pop`	an object of `Pop-class`
`crossPlan`	a matrix with two column representing female and male parents. Either integers for the position in population or character strings for the IDs.
`simParam`	an object of `SimParam`

Value

Returns an object of `Pop-class`

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=10, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)

#Create population
pop = newPop(founderPop, simParam=SP)

#Cross individual 1 with individual 10
crossPlan = matrix(c(1,10), nrow=1, ncol=2)
pop2 = makeCross(pop, crossPlan, simParam=SP)
```

`makeCross2` *Make designed crosses*

Description

Makes crosses between two populations using a user supplied crossing plan.

Usage

```
makeCross2(females, males, crossPlan, simParam = NULL)
```

Arguments

`females`	an object of `Pop-class` for female parents.
`males`	an object of `Pop-class` for male parents.
`crossPlan`	a matrix with two column representing female and male parents. Either integers for the position in population or character strings for the IDs.
`simParam`	an object of `SimParam`

Value

Returns an object of `Pop-class`

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=10, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)

#Create population
pop = newPop(founderPop, simParam=SP)
```

```
#Cross individual 1 with individual 10
crossPlan = matrix(c(1,10), nrow=1, ncol=2)
pop2 = makeCross2(pop, pop, crossPlan, simParam=SP)
```

---

makeDH *Generates DH lines*

Description

Creates DH lines from each individual in a population. Only works when gender is "no".

Usage

```
makeDH(pop, nDH = 1, useFemale = TRUE, simParam = NULL)
```

Arguments

pop	an object of 'Pop' superclass
nDH	total number of DH lines per individual
useFemale	should female recombination rates be used. This parameter has no effect if, recombRatio=1.
simParam	an object of 'SimParam' class

Value

Returns an object of Pop-class

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=2, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)

#Create population
pop = newPop(founderPop, simParam=SP)

#Create 1 DH for each individual
pop2 = makeDH(pop, simParam=SP)
```

MapPop-class *Raw population with genetic map*

Description

Extends RawPop-class to add a genetic map. This is the first object created in a simulation. It is used for creating initial populations and setting traits in the SimParam.

Usage

```
## S4 method for signature 'MapPop'
x[i]

## S4 method for signature 'MapPop'
c(x, ...)
```

Arguments

x	a 'MapPop'
i	index of chromosomes
...	aditional 'MapPop' objects

Methods (by generic)

- [: Extract MapPop by index
- c: Combine MapPop chromosomes

Slots

genMap "matrix" of chromsome genetic maps

centromere vector of centromere positions

meanG *Mean genetic values*

Description

Returns the mean genetic values for all traits

Usage

```
meanG(pop)
```

Arguments

pop	an object of Pop-class or HybridPop-class

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=10, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)
SP$addTraitA(10)
SP$setVarE(h2=0.5)

#Create population
pop = newPop(founderPop, simParam=SP)
meanG(pop)
```

---

meanP *Mean phenotypic values*

Description

Returns the mean phenotypic values for all traits

Usage

```
meanP(pop)
```

Arguments

pop	an object of Pop-class or HybridPop-class

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=10, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)
SP$addTraitA(10)
SP$setVarE(h2=0.5)

#Create population
pop = newPop(founderPop, simParam=SP)
meanP(pop)
```

mergePops *Merge list of populations*

Description

Rapidly merges a list of populations into a single population

Usage

```
mergePops(popList)
```

Arguments

popList	a list containing Pop-class elements

Value

Returns a Pop-class

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=10, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)

#Create a list of populations and merge list
pop = newPop(founderPop, simParam=SP)
popList = list(pop, pop)
pop2 = mergePops(popList)
```

newMapPop *New MapPop*

Description

Creates a new MapPop-class from user supplied genetic maps and haplotypes.

Usage

```
newMapPop(genMap, haplotypes, inbred = FALSE, ploidy = 2L)
```

Arguments

genMap	a list of genetic maps
haplotypes	a list of matrices or data.frames that can be coerced to matrices. See details.
inbred	are individuals fully inbred
ploidy	ploidy level of organism

Details

Each item of genMap must be a vector of ordered genetic lengths in Morgans. The first value must be zero. The length of the vector determines the number of segregating sites on the chromosome.

Each item of haplotypes must be coercible to a matrix. The columns of this matrix correspond to segregating sites and their number must match

Value

an object of MapPop-class

Examples

```
# Create genetic map for two chromosomes, each 1 Morgan long
# Each chromosome contains 11 equally spaced segregating sites
genMap = list(seq(0,1,length.out=11),
              seq(0,1,length.out=11))

# Create haplotypes for 10 outbred individuals
chr1 = sample(x=0:1,size=20*11,replace=TRUE)
chr1 = matrix(chr1,nrow=20,ncol=11)
chr2 = sample(x=0:1,size=20*11,replace=TRUE)
chr2 = matrix(chr2,nrow=20,ncol=11)
haplotypes = list(chr1,chr2)

founderPop = newMapPop(genMap=genMap,haplotypes=haplotypes)
```

---

newPop *Create new Population*

Description

Creates a new Pop-class from an object of MapPop-class or RawPop-class. The function is intended for creating initial populations from 'FOUNDERPOP' created by runMacs.

Usage

```
newPop(rawPop, mother = NULL, father = NULL, origM = NULL,
  origF = NULL, isDH = FALSE, simParam = NULL)
```

Arguments

rawPop	an object of MapPop-class or RawPop-class
mother	optional id for mothers. Must match id in pedigree if using track pedigree.
father	optional id for fathers. Must match id in pedigree if using track pedigree.
origM	optional alternative id for mothers
origF	optional alternative id for fathers
isDH	optional value indicating if the individuals are doubled haploids and/or inbred founders
simParam	an object of SimParam

Value

Returns an object of Pop-class

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=2, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)
SP$addTraitA(10)

#Create population
pop = newPop(founderPop, simParam=SP)
```

---

nInd *Number of individuals*

Description

A wrapper for accessing the nInd slot

Usage

```
nInd(pop)
```

Arguments

pop	a Pop-class or similar object

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=10, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)
SP$addTraitAD(10, meanDD=0.5)
SP$setVarE(h2=0.5)

#Create population
pop = newPop(founderPop, simParam=SP)
nInd(pop)
```

`pedigreeCross` *Pedigree cross*

Description

Creates a `Pop-class` from a generic pedigree and a set of founder individuals.

Usage

```
pedigreeCross(founderPop, id, mother, father, maxCycle = 100,
  DH = NULL, useFemale = TRUE, simParam = NULL)
```

Arguments

`founderPop`	a `Pop-class`
`id`	a vector of unique identifiers for individuals in the pedigree. The values of these ids are seperate from the ids in the founderPop.
`mother`	a vector of identifiers for the mothers of individuals in the pedigree. Must match one of the elements in the id vector or they will be treated as unknown.
`father`	a vector of identifiers for the fathers of individuals in the pedigree. Must match one of the elements in the id vector or they will be treated as unknown.
`maxCycle`	the maximum number of loops to make over the pedigree to sort it.
`DH`	an optional vector indicating if an individual should be made a doubled haploid.
`useFemale`	If creating DH lines, should female recombination rates be used. This parameter has no effect if, recombRatio=1.
`simParam`	an object of 'SimParam' class

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=2, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)

#Create population
pop = newPop(founderPop, simParam=SP)

#Pedigree for a biparental cross with 7 generations of selfing
id = 1:10
mother = c(0,0,1,3:9)
father = c(0,0,2,3:9)
pop2 = pedigreeCross(pop, id, mother, father, simParam=SP)
```

---

pheno *Phenotype*

Description

A wrapper for accessing the pheno slot

Usage

```
pheno(pop)
```

Arguments

pop	a Pop-class or similar object

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=10, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)
SP$addTraitAD(10, meanDD=0.5)
SP$setVarE(h2=0.5)

#Create population
pop = newPop(founderPop, simParam=SP)
pheno(pop)
```

Pop-class *Population*

Description

Extends RawPop-class to add gender, genetic values, phenotypes, and pedigrees.

Usage

```
## S4 method for signature 'Pop'
x[i]

## S4 method for signature 'Pop'
c(x, ...)
```

Arguments

x	a 'Pop'
i	index of individuals
...	additional 'Pop' objects

Methods (by generic)

- [: Extract Pop by index or id
- c: Combine multiple Pops

Slots

id an individual's identifier

mother the identifier of the individual's mother

father the identifier of the individual's father

gender gender of individuals

nTraits number of traits

gv matrix of genetic values. When using GxE traits, gv reflects gv when w=0. Dimensions are nInd by nTraits.

pheno matrix of phenotypic values. Dimensions are nInd by nTraits.

ebv matrix of estimated breeding values. Dimensions are nInd rows and a variable number of columns.

gxe list containing GxE slopes for GxE traits

fixEff a fixed effect relating to the phenotype. Used by genomic selection models but otherwise ignored.

reps the number of replications used to measure the phenotype. Used by genomic selection models, but otherwise ignored.

popVar *Population variance*

Description

Calculates the population variance matrix as opposed to the sample variance matrix calculated by var. i.e. divides by n instead of n-1

Usage

```
popVar(X)
```

Arguments

X	an n by m matrix

Value

an m by m variance-covariance matrix

pullIbdHaplo *Pull Identity By Descent (IBD) haplotypes*

Description

Retrieves Identity By Descent (IBD) haplotype data

Usage

```
pullIbdHaplo(pop = NULL, chr = NULL, snpChip = NULL,
  pedigree = NULL, simParam = NULL)
```

Arguments

pop	an object of Pop-class or RawPop-class. If NULL, haplotypes for the whole ancestral pedigree are retreived. Otherwise, haplotypes just for the pop individuals are retreived. In both cases the base population is controlled by pedigree.
chr	a vector of chromosomes to retrieve. If NULL, all chromosomes are retrieved.
snpChip	an integer. Indicates which SNP array loci are retrieved. If NULL, all sites are retrieved.
pedigree	a matrix with ancestral pedigree to set a base population. It should be of the same form as simParam$pedigree (see SimParam_setTrackPed), i.e., two columns (mother and father) and the same number of rows as simParam$pedigree. Base population can be set by setting parents as 0. If NULL, pedigree from SimParam is taken.
simParam	an object of SimParam

Value

Returns a matrix of haplotypes with Identity By Descent (IBD) coding of locus alleles. The matrix colnames reflect whether all segregagting loci (sites) are retreived or only SNP array loci.

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=10, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)
SP$addTraitA(10)
SP$addSnpChip(5)
SP$setTrackRec(TRUE)

#Create population
pop = newPop(founderPop, simParam=SP)
pullIbdHaplo(pop, simParam=SP)
```

---

pullMultipleSnpGeno *Pull SNP genotype for multiple snp chips*

Description

Retrieves SNP genotype data for multiple snp chips

Usage

```
pullMultipleSnpGeno(pop, chips, missing = 9, simParam = NULL)
```

Arguments

pop	an object of Pop-class
chips	a vector. For each animal indicates what snp chip to use
missing	What value to use for missing
simParam	an object of SimParam

Value

Returns a matrix of SNP genotypes.

pullMultipleSnpHaplo *Pull SNP haplotypes for multiple chips*

Description

Retrieves SNP haplotype data for multiple snp

Usage

```
pullMultipleSnpHaplo(pop, chips, haplo = "all", missing = 9,
  simParam = NULL)
```

Arguments

pop	an object of Pop-class
chips	a vector. For each animal indicates what snp chip to use
haplo	either "all" for all haplotypes or an integer for a single set of haplotypes. Use a value of 1 for female haplotyes and a value of 2 for male haplotypes.
missing	What value to use for missing
simParam	an object of SimParam

Value

Returns a matrix of SNP haplotypes.

pullQtlGeno *Pull QTL genotype*

Description

Retrieves QTL genotype data

Usage

```
pullQtlGeno(pop, trait = 1, chr = NULL, simParam = NULL)
```

Arguments

pop	an object of Pop-class
trait	an integer. Indicates which trait's QTL genotypes to retrieve.
chr	a vector of chromosomes to retrieve. If NULL, all chromosome are retrieved.
simParam	an object of SimParam

Value

Returns a matrix of QTL genotypes.

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=10, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)
SP$addTraitA(10)
SP$addSnpChip(5)

#Create population
pop = newPop(founderPop, simParam=SP)
pullQtlGeno(pop, simParam=SP)
```

`pullQtlHaplo` *Pull QTL haplotypes*

Description

Retrieves QTL haplotype data

Usage

```
pullQtlHaplo(pop, trait = 1, haplo = "all", chr = NULL,
  simParam = NULL)
```

Arguments

`pop`	an object of `Pop-class`
`trait`	an integer. Indicates which trait's QTL haplotypes to retrieve.
`haplo`	either "all" for all haplotypes or an integer for a single set of haplotypes. Use a value of 1 for female haplotyes and a value of 2 for male haplotypes.
`chr`	a vector of chromosomes to retrieve. If NULL, all chromosome are retrieved.
`simParam`	an object of `SimParam`

Value

Returns a matrix of QTL haplotypes.

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=10, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)
SP$addTraitA(10)
SP$addSnpChip(5)

#Create population
pop = newPop(founderPop, simParam=SP)
pullQtlHaplo(pop, simParam=SP)
```

---

pullSegSiteGeno *Pull seg site genotypes*

Description

Retrieves genotype data for all segregating sites

Usage

```
pullSegSiteGeno(pop, chr = NULL, simParam = NULL)
```

Arguments

pop	an object of Pop-class or RawPop-class
chr	a vector of chromosomes to retrieve. If NULL, all chromosome are retrieved.
simParam	an object of SimParam

Value

Returns a matrix of genotypes

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=10, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)
SP$addTraitA(10)
SP$addSnpChip(5)

#Create population
pop = newPop(founderPop, simParam=SP)
pullSegSiteGeno(pop, simParam=SP)
```

pullSegSiteHaplo *Pull seg site haplotypes*

Description

Retrieves haplotype data for all segregating sites

Usage

```
pullSegSiteHaplo(pop, haplo = "all", chr = NULL, simParam = NULL)
```

Arguments

pop	an object of Pop-class or RawPop-class
haplo	either "all" for all haplotypes or an integer for a single set of haplotypes. Use a value of 1 for female haplotyes and a value of 2 for male haplotypes.
chr	a vector of chromosomes to retrieve. If NULL, all chromosome are retrieved.
simParam	an object of SimParam

Value

Returns a matrix of haplotypes

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=10, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)
SP$addTraitA(10)
SP$addSnpChip(5)

#Create population
pop = newPop(founderPop, simParam=SP)
pullSegSiteHaplo(pop, simParam=SP)
```

pullSnpGeno *Pull SNP genotype*

Description

Retrieves SNP genotype data

Usage

```
pullSnpGeno(pop, snpChip = 1, chr = NULL, simParam = NULL)
```

Arguments

pop	an object of Pop-class
snpChip	an integer. Indicates which SNP chip's genotypes to retrieve.
chr	a vector of chromosomes to retrieve. If NULL, all chromosome are retrieved.
simParam	an object of SimParam

Value

Returns a matrix of SNP genotypes.

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=10, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)
SP$addTraitA(10)
SP$addSnpChip(5)

#Create population
pop = newPop(founderPop, simParam=SP)
pullSnpGeno(pop, simParam=SP)
```

pullSnpHaplo *Pull SNP haplotypes*

Description

Retrieves SNP haplotype data

Usage

```
pullSnpHaplo(pop, snpChip = 1, haplo = "all", chr = NULL,
  simParam = NULL)
```

Arguments

pop	an object of Pop-class
snpChip	an integer. Indicates which SNP chip's haplotypes to retrieve.
haplo	either "all" for all haplotypes or an integer for a single set of haplotypes. Use a value of 1 for female haplotyes and a value of 2 for male haplotypes.
chr	a vector of chromosomes to retrieve. If NULL, all chromosome are retrieved.
simParam	an object of SimParam

Value

Returns a matrix of SNP haplotypes.

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=10, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)
SP$addTraitA(10)
SP$addSnpChip(5)

#Create population
pop = newPop(founderPop, simParam=SP)
pullSnpHaplo(pop, simParam=SP)
```

---

quickHaplo *Quick founder haplotype simulation*

Description

Rapidly simulates founder haplotypes by randomly sampling 0s and 1s. This is equivalent to having all loci with allele frequency 0.5 and being in linkage equilibrium.

Usage

```
quickHaplo(nInd, nChr, segSites, genLen = 1, ploidy = 2L,
  inbred = FALSE)
```

Arguments

nInd	number of individuals to simulate
nChr	number of chromosomes to simulate
segSites	number of segregating sites per chromosome
genLen	genetic length of chromosomes
ploidy	ploidy level of organism
inbred	should founder individuals be inbred

Value

an object of MapPop-class

Examples

```
# Creates a populations of 10 outbred individuals
# Their genome consists of 1 chromosome and 100 segregating sites
founderPop = quickHaplo(nInd=10,nChr=1,segSites=100)
```

---

randCross *Make random crosses*

Description

A wrapper for makeCross that randomly selects parental combinations for all possible combinations.

Usage

```
randCross(pop, nCrosses, nProgeny = 1, balance = TRUE,
  parents = NULL, ignoreGender = FALSE, simParam = NULL)
```

Arguments

pop	an object of Pop-class
nCrosses	total number of crosses to make
nProgeny	number of progeny per cross
balance	if using gender, this option will balance the number of progeny per parent
parents	an optional vector of indices for allowable parents
ignoreGender	should gender be ignored
simParam	an object of SimParam

Value

Returns an object of Pop-class

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=10, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)

#Create population
pop = newPop(founderPop, simParam=SP)

#Make 10 crosses
pop2 = randCross(pop, 10, simParam=SP)
```

`randCross2` *Make random crosses*

Description

A wrapper for `makeCross2` that randomly selects parental combinations for all possible combinantions between two populations.

Usage

```
randCross2(females, males, nCrosses, nProgeny = 1, balance = TRUE,
  femaleParents = NULL, maleParents = NULL, ignoreGender = FALSE,
  simParam = NULL)
```

Arguments

`females`	an object of `Pop-class` for female parents.
`males`	an object of `Pop-class` for male parents.
`nCrosses`	total number of crosses to make
`nProgeny`	number of progeny per cross
`balance`	this option will balance the number of progeny per parent
`femaleParents`	an optional vector of indices for allowable female parents
`maleParents`	an optional vector of indices for allowable male parents
`ignoreGender`	should gender be ignored
`simParam`	an object of `SimParam`

Value

Returns an object of `Pop-class`

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=10, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)

#Create population
pop = newPop(founderPop, simParam=SP)

#Make 10 crosses
pop2 = randCross2(pop, pop, 10, simParam=SP)
```

RawPop-class *Raw Population*

Description

The raw population class contains only genotype data.

Usage

```
## S4 method for signature 'RawPop'
x[i]

## S4 method for signature 'RawPop'
c(x, ...)
```

Arguments

x	a 'RawPop'
i	index of individuals
...	additional 'RawPop' objects

Methods (by generic)

- [: Extract RawPop by index
- c: Combine multiple RawPops

Slots

nInd number of individuals

nChr number of chromosomes

ploidy level of ploidy

nLoci number of loci per chromosome

geno "matrix" containing chromosome genotypes. The "matrix" has dimensions nChr by 1 and each element is a three dimensional array of raw values. The array dimensions are nLoci by ploidy by nInd.

`resetPop` *Reset population*

Description

Recalculates a population's genetic values and resets phenotypes and EBVs.

Usage

```
resetPop(pop, simParam = NULL)
```

Arguments

pop	an object of Pop-class
simParam	an object of SimParam

Value

an object of Pop-class

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=2, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)
SP$addTraitA(10)

#Create population
pop = newPop(founderPop, simParam=SP)

#Rescale to set mean to 1
SP$rescaleTraits(mean=1)
pop = resetPop(pop, simParam=SP)
```

RRBLUP *RR-BLUP Model*

Description

Fits an RR-BLUP model for genomic predictions.

Usage

```
RRBLUP(pop, traits = 1, use = "pheno", snpChip = 1, useQtl = FALSE,
  maxIter = 1000L, simParam = NULL, ...)
```

Arguments

pop	a Pop-class to serve as the training population
traits	an integer indicating the trait or traits to model, or a function of the traits returning a single value.
use	train model using phenotypes "pheno", genetic values "gv", estimated breeding values "ebv", breeding values "bv", or randomly "rand"
snpChip	an integer indicating which SNP chip genotype to use
useQtl	should QTL genotypes be used instead of a SNP chip. If TRUE, snpChip specifies which trait's QTL to use, and thus these QTL may not match the QTL underlying the phenotype supplied in traits.
maxIter	maximum number of iterations. Only used when number of traits is greater than 1.
simParam	an object of SimParam
...	additional arguments if using a function for traits

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=10, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)
SP$addTraitA(10)
SP$setVarE(h2=0.5)
SP$addSnpChip(10)

#Create population
pop = newPop(founderPop, simParam=SP)

#Run GS model and set EBV
ans = RRBLUP(pop, simParam=SP)
pop = setEBV(pop, ans, simParam=SP)
```

```
#Evaluate accuracy
cor(gv(pop), ebv(pop))
```

RRBLUP2 *RR-BLUP Model 2*

Description

Fits an RR-BLUP model for genomic predictions. This implementation is meant for situations where RRBLUP is too slow. Note that RRBLUP2 is only faster in certain situations, see details below. Most users should use RRBLUP.

Usage

```
RRBLUP2(pop, traits = 1, use = "pheno", snpChip = 1,
  useQtl = FALSE, maxIter = 10, Vu = NULL, Ve = NULL,
  useEM = TRUE, tol = 1e-06, simParam = NULL, ...)
```

Arguments

pop	a Pop-class to serve as the training population
traits	an integer indicating the trait to model or a function of the traits returning a single value. Unlike RRBLUP, only univariate models are supported.
use	train model using phenotypes "pheno", genetic values "gv", estimated breeding values "ebv", breeding values "bv", or randomly "rand"
snpChip	an integer indicating which SNP chip genotype to use
useQtl	should QTL genotypes be used instead of a SNP chip. If TRUE, snpChip specifies which trait's QTL to use, and thus these QTL may not match the QTL underlying the phenotype supplied in traits.
maxIter	maximum number of iterations.
Vu	marker effect variance. If value is NULL, a reasonable starting point is chosen automatically.
Ve	error variance. If value is NULL, a reasonable starting point is chosen automatically.
useEM	use EM to solve variance components. If false, the initial values are considered true.
tol	tolerance for EM algorithm convergence
simParam	an object of SimParam
...	additional arguments if using a function for traits

Details

The RRBLUP2 function works best when the number of markers is not too large. This is because it solves the RR-BLUP problem by setting up and solving Henderson's mixed model equations. Solving these equations involves a square matrix with dimensions equal to the number of fixed effects plus the number of random effects (markers). Whereas the RRBLUP function solves the RR-BLUP problem using the EMMA approach. This approach involves a square matrix with dimensions equal to the number of phenotypic records. This means that the RRBLUP2 function uses less memory than RRBLUP when the number of markers is approximately equal to or smaller than the number of phenotypic records.

The RRBLUP2 function is not recommend for cases where the variance components are unknown. This is uses the EM algorithm to solve for unknown variance components, which is generally considerably slower than the EMMA approach of RRBLUP. The number of iterations for the EM algorith is set by maxIter. The default value is typically too small for convergence. When the algorithm fails to converage a warning is displayed, but results are given for the last iteration. These results may be "good enough". However we make no claim to this effect, because we can not generalize to all possible use cases.

The RRBLUP2 function can quickly solve the mixed model equations without estimating variance components. The variance components are set by defining Vu and Ve. Estimation of components is suppressed by setting useEM to false. This may be useful if the model is being retrained multiple times during the simulation. You could run RRBLUP function the first time the model is trained, and then use the variance components from this output for all future runs with the RRBLUP2 functions. Again, we can make no claim to the general robustness of this approach.

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=10, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)
SP$addTraitA(10)
SP$setVarE(h2=0.5)
SP$addSnpChip(10)

#Create population
pop = newPop(founderPop, simParam=SP)

#Run GS model and set EBV
ans = RRBLUP2(pop, simParam=SP)
pop = setEBV(pop, ans, simParam=SP)

#Evaluate accuracy
cor(gv(pop), ebv(pop))
```

RRBLUPMemUse *RRBLUP Memory Usage*

Description

Estimates the amount of RAM needed to run the RRBLUP and its related functions for a given training population size. Note that this functions may underestimate total usage.

Usage

```
RRBLUPMemUse(nInd, nMarker, model = "REG")
```

Arguments

nInd	the number of individuals in the training population
nMarker	the number of markers per individual
model	either "REG", "GCA", or "SCA" for RRBLUP RRBLUP_GCA and RRBLUP_SCA respectively.

Value

Returns an estimate for the required gigabytes of RAM

Examples

```
RRBLUPMemUse(nInd=1000, nMarker=5000)
```

RRBLUP_D *RR-BLUP Model with Dominance*

Description

Fits an RR-BLUP model for genomic predictions that includes dominance effects.

Usage

```
RRBLUP_D(pop, traits = 1, use = "pheno", snpChip = 1,
  useQtl = FALSE, maxIter = 40L, simParam = NULL, ...)
```

Arguments

pop	a Pop-class to serve as the training population
traits	an integer indicating the trait to model, or a function of the traits returning a single value.
use	train model using phenotypes "pheno", genetic values "gv", estimated breeding values "ebv", breeding values "bv", or randomly "rand"
snpChip	an integer indicating which SNP chip genotype to use

useQtl	should QTL genotypes be used instead of a SNP chip. If TRUE, snpChip specifies which trait's QTL to use, and thus these QTL may not match the QTL underlying the phenotype supplied in traits.
maxIter	maximum number of iterations. Only used when number of traits is greater than 1.
simParam	an object of SimParam
...	additional arguments if using a function for traits

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=10, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)
SP$addTraitAD(10, meanDD=0.5)
SP$setVarE(h2=0.5)
SP$addSnpChip(10)

#Create population
pop = newPop(founderPop, simParam=SP)

#Run GS model and set EBV
ans = RRBLUP_D(pop, simParam=SP)
pop = setEBV(pop, ans, simParam=SP)

#Evaluate accuracy
cor(gv(pop), ebv(pop))
```

RRBLUP_D2 *RR-BLUP with Dominance Model 2*

Description

Fits an RR-BLUP model for genomic predictions that includes dominance effects. This implementation is meant for situations where RRBLUP_D is too slow. Note that RRBLUP_D2 is only faster in certain situations. Most users should use RRBLUP_D.

Usage

```
RRBLUP_D2(pop, traits = 1, use = "pheno", snpChip = 1,
  useQtl = FALSE, maxIter = 10, Va = NULL, Vd = NULL, Ve = NULL,
  useEM = TRUE, tol = 1e-06, simParam = NULL, ...)
```

Arguments

pop	a Pop-class to serve as the training population
traits	an integer indicating the trait to model, or a function of the traits returning a single value.
use	train model using phenotypes "pheno", genetic values "gv", estimated breeding values "ebv", breeding values "bv", or randomly "rand"
snpChip	an integer indicating which SNP chip genotype to use
useQtl	should QTL genotypes be used instead of a SNP chip. If TRUE, snpChip specifies which trait's QTL to use, and thus these QTL may not match the QTL underlying the phenotype supplied in traits.
maxIter	maximum number of iterations. Only used when number of traits is greater than 1.
Va	marker effect variance for additive effects. If value is NULL, a reasonable starting point is chosen automatically.
Vd	marker effect variance for dominance effects. If value is NULL, a reasonable starting point is chosen automatically.
Ve	error variance. If value is NULL, a reasonable starting point is chosen automatically.
useEM	use EM to solve variance components. If false, the initial values are considered true.
tol	tolerance for EM algorithm convergence
simParam	an object of SimParam
...	additional arguments if using a function for traits

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=10, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)
SP$addTraitAD(10, meanDD=0.5)
SP$setVarE(h2=0.5)
SP$addSnpChip(10)

#Create population
pop = newPop(founderPop, simParam=SP)

#Run GS model and set EBV
ans = RRBLUP_D2(pop, simParam=SP)
pop = setEBV(pop, ans, simParam=SP)

#Evaluate accuracy
cor(gv(pop), ebv(pop))
```

RRBLUP_GCA *RR-BLUP GCA Model*

Description

Fits an RR-BLUP model that estimates seperate marker effects for females and males. Useful for predicting GCA of parents in single cross hybrids. Can also predict performance of specific single cross hybrids.

Usage

```
RRBLUP_GCA(pop, traits = 1, use = "pheno", snpChip = 1,
  useQtl = FALSE, maxIter = 40L, simParam = NULL, ...)
```

Arguments

pop	a Pop-class to serve as the training population
traits	an integer indicating the trait to model, or a function of the traits returning a single value.
use	train model using phenotypes "pheno", genetic values "gv", estimated breeding values "ebv", breeding values "bv", or randomly "rand"
snpChip	an integer indicating which SNP chip genotype to use
useQtl	should QTL genotypes be used instead of a SNP chip. If TRUE, snpChip specifies which trait's QTL to use, and thus these QTL may not match the QTL underlying the phenotype supplied in traits.
maxIter	maximum number of iterations for convergence.
simParam	an object of SimParam
...	additional arguments if using a function for traits

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=10, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)
SP$addTraitA(10)
SP$setVarE(h2=0.5)
SP$addSnpChip(10)

#Create population
pop = newPop(founderPop, simParam=SP)

#Run GS model and set EBV
ans = RRBLUP_GCA(pop, simParam=SP)
pop = setEBV(pop, ans, simParam=SP)
```

```
#Evaluate accuracy
cor(gv(pop), ebv(pop))
```

RRBLUP_GCA2 *RR-BLUP GCA Model 2*

Description

Fits an RR-BLUP model that estimates seperate marker effects for females and males. This implementation is meant for situations where RRBLUP_GCA is too slow. Note that RRBLUP_GCA2 is only faster in certain situations. Most users should use RRBLUP_GCA.

Usage

```
RRBLUP_GCA2(pop, traits = 1, use = "pheno", snpChip = 1,
  useQtl = FALSE, maxIter = 10, VuF = NULL, VuM = NULL,
  Ve = NULL, useEM = TRUE, tol = 1e-06, simParam = NULL, ...)
```

Arguments

pop	a Pop-class to serve as the training population
traits	an integer indicating the trait to model, or a function of the traits returning a single value.
use	train model using phenotypes "pheno", genetic values "gv", estimated breeding values "ebv", breeding values "bv", or randomly "rand"
snpChip	an integer indicating which SNP chip genotype to use
useQtl	should QTL genotypes be used instead of a SNP chip. If TRUE, snpChip specifies which trait's QTL to use, and thus these QTL may not match the QTL underlying the phenotype supplied in traits.
maxIter	maximum number of iterations for convergence.
VuF	marker effect variance for females. If value is NULL, a reasonable starting point is chosen automatically.
VuM	marker effect variance for males. If value is NULL, a reasonable starting point is chosen automatically.
Ve	error variance. If value is NULL, a reasonable starting point is chosen automatically.
useEM	use EM to solve variance components. If false, the initial values are considered true.
tol	tolerance for EM algorithm convergence
simParam	an object of SimParam
...	additional arguments if using a function for traits

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=10, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)
SP$addTraitA(10)
SP$setVarE(h2=0.5)
SP$addSnpChip(10)

#Create population
pop = newPop(founderPop, simParam=SP)

#Run GS model and set EBV
ans = RRBLUP_GCA2(pop, simParam=SP)
pop = setEBV(pop, ans, simParam=SP)

#Evaluate accuracy
cor(gv(pop), ebv(pop))
```

RRBLUP_SCA *RR-BLUP SCA Model*

Description

An extention of RRBLUP_GCA that adds dominance effects. Note that we have not seen any consistent benefit of this model over RRBLUP_GCA.

Usage

```
RRBLUP_SCA(pop, traits = 1, use = "pheno", snpChip = 1,
  useQtl = FALSE, maxIter = 40L, simParam = NULL, ...)
```

Arguments

pop	a Pop-class to serve as the training population
traits	an integer indicating the trait to model, or a function of the traits returning a single value.
use	train model using phenotypes "pheno", genetic values "gv", estimated breeding values "ebv", breeding values "bv", or randomly "rand"
snpChip	an integer indicating which SNP chip genotype to use
useQtl	should QTL genotypes be used instead of a SNP chip. If TRUE, snpChip specifies which trait's QTL to use, and thus these QTL may not match the QTL underlying the phenotype supplied in traits.
maxIter	maximum number of iterations for convergence.
simParam	an object of SimParam
...	additional arguments if using a function for traits

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=2, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)
SP$addTraitA(10)
SP$setVarE(h2=0.5)
SP$addSnpChip(10)

#Create population
pop = newPop(founderPop, simParam=SP)

#Run GS model and set EBV
ans = RRBLUP_SCA(pop, simParam=SP)
pop = setEBV(pop, ans, simParam=SP)

#Evaluate accuracy
cor(gv(pop), ebv(pop))
```

RRDsol-class *RR-BLUP Solution with Dominance*

Description

Extends LociMap-class to contain estimated effects from RRBLUP

Slots

markerEff GEBVs for markers

addEff additive effects

domEff dominance effects

fixEff Estimates for fixed effects

Vu Estimated marker variance

Ve Estimated error variance

LL Log-likelihood

iter Number of iterations for convergence

RRsol-class *RR-BLUP Solution*

Description

Extends LociMap-class to contain estimated effects from RRBLUP

Slots

markerEff GEBVs for markers

fixEff Estimates for fixed effects

Vu Estimated marker variance

Ve Estimated error variance

LL Log-likelihood

iter Number of iterations for convergence

runMacs *Create founder haplotypes using MaCS*

Description

Uses the MaCS software to produce founder haplotypes.

Usage

```
runMacs(nInd, nChr = 1, segSites = NULL, inbred = FALSE,
  species = "GENERIC", split = NULL, ploidy = 2L,
  manualCommand = NULL, manualGenLen = NULL, nThreads = NULL)
```

Arguments

nInd	number of individuals to simulate
nChr	number of chromosomes to simulate
segSites	number of segregating sites to keep per chromosome. A value of NULL results in all sites being retained.
inbred	should founder individuals be inbred
species	species history to simulate. See details.
split	an optional historic population split in terms of generations ago.
ploidy	ploidy level of organism
manualCommand	user provided MaCS options. For advanced users only.

manualGenLen — user provided genetic length. This must be supplied if using manualCommand. If not using manualCommand, this value will replace the predefined genetic length for the species. However, this the genetic length is only used by AlphaSimR and is not passed to MaCS, so MaCS still uses the predefined genetic length. For advanced users only.

nThreads — if OpenMP is available, this will allow for simulating chromosomes in parallel. If the value is NULL, the number of threads is automatically detected.

Details

The current species histories are included: GENERIC, CATTLE, WHEAT, MAIZE, and EUROPEAN.

Value

an object of MapPop-class

Examples

```
# Creates a populations of 10 outbred individuals
# Their genome consists of 1 chromosome and 100 segregating sites
founderPop = runMacs(nInd=10,nChr=1,segSites=100)
```

---

runMacs2 *Alternative wrapper for MaCS*

Description

A wrapper function for runMacs. This wrapper is designed to be easier to use than supply custom comands to manualCommand in runMacs. It effectively automates the creation of an appropriate manualCommand using user supplied variables, but only deals with a subset of the possibilities. The defaults were chosen to match species="GENERIC" in runMacs.

Usage

```
runMacs2(nInd, nChr = 1, segSites = NULL, Ne = 100, bp = 1e+08,
  genLen = 1, mutRate = 2.5e-08, histNe = c(500, 1500, 6000, 12000,
  1e+05), histGen = c(100, 1000, 10000, 1e+05, 1e+06), inbred = FALSE,
  split = NULL, ploidy = 2L, returnCommand = FALSE,
  nThreads = NULL)
```

Arguments

nInd	number of individuals to simulate
nChr	number of chromosomes to simulate
segSites	number of segregating sites to keep per chromosome
Ne	effective population size
bp	base pair length of chromosome
genLen	genetic length of chromosome in Morgans
mutRate	per base pair mutation rate
histNe	effective population size in previous generations
histGen	number of generations ago for effective population sizes given in histNe
inbred	should founder individuals be inbred
split	an optional historic population split in terms of generations ago
ploidy	ploidy level of organism
returnCommand	should the command passed to manualCommand in runMacs be returned. If TRUE, MaCS will not be called and the command is returned instead.
nThreads	if OpenMP is available, this will allow for simulating chromosomes in parallel. If the value is NULL, the number of threads is automatically detected.

Value

an object of MapPop-class or if returnCommand is true a string giving the MaCS command passed to the manualCommand argument of runMacs.

Examples

```
# Creates a populations of 10 outbred individuals
# Their genome consists of 1 chromosome and 100 segregating sites
# The command is equivalent to using species="GENERIC" in runMacs
founderPop = runMacs2(nInd=10,nChr=1,segSites=100)
```

---

sampleHaplo *Sample haplotypes from a MapPop*

Description

Creates a new MapPop-class from an existing MapPop-class by randomly sampling haplotypes.

Usage

```
sampleHaplo(mapPop, nInd, inbred = FALSE, ploidy = NULL,
  replace = TRUE)
```

Arguments

mapPop	the MapPop-class used to sample haplotypes
nInd	the number of individuals to create
inbred	should new individuals be fully inbred
ploidy	new ploidy level for organism. If NULL, the ploidy level of the mapPop is used.
replace	should haplotypes be sampled with replacement

Value

an object of MapPop-class

Examples

```
founderPop = quickHaplo(nInd=2,nChr=1,segSites=11,inbred=TRUE)
founderPop = sampleHaplo(nInd=20,mapPop=founderPop)
```

SCAsol-class *RR-BLUP SCA Solution*

Description

Extends GCAsol-class to contain estimated effects from RRBLUP_SCA

Slots

d dominance effect

---

selectCross *Select and randomly cross*

Description

This is a wrapper that combines the functionalities of randCross and selectInd. The purpose of this wrapper is to combine both selection and crossing in one function call that minimized the amount of intermediate populations created. This reduces RAM usage and simplifies code writing. Note that this wrapper does not provide the full functionality of either function.

Usage

```
selectCross(pop, nInd = NULL, nFemale = NULL, nMale = NULL, nCrosses,
  nProgeny = 1, trait = 1, use = "pheno", selectTop = TRUE,
  simParam = NULL, ..., balance = TRUE)
```

Arguments

pop	an object of Pop-class
nInd	the number of individuals to select. These individuals are selected without regards to gender and it supercedes values for nFemale and nMale. Thus if the simulation uses gender, it is likely better to leave this value as NULL and use nFemale and nMale instead.
nFemale	the number of females to select. This value is ignored if nInd is set.
nMale	the number of males to select. This value is ignored if nInd is set.
nCrosses	total number of crosses to make
nProgeny	number of progeny per cross
trait	the trait for selection. Either a number indicating a single trait or a function returning a vector of length nInd.
use	select on genetic values "gv", estimated breeding values "ebv", breeding values "bv", phenotypes "pheno", or randomly "rand"
selectTop	selects highest values if true. Selects lowest values if false.
simParam	an object of SimParam
...	additional arguments if using a function for trait
balance	if using gender, this option will balance the number of progeny per parent. This argument occurs after ..., so the argument name must be matched exactly.

Value

Returns an object of Pop-class

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=10, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)
SP$addTraitA(10)
SP$setVarE(h2=0.5)

#Create population
pop = newPop(founderPop, simParam=SP)

#Select 4 individuals and make 8 crosses
pop2 = selectCross(pop, nInd=4, nCrosses=8, simParam=SP)
```

`selectFam` *Select families*

Description

Selects a subset of full-sib families from a population.

Usage

```
selectFam(pop, nFam, trait = 1, use = "pheno", gender = "B",
  famType = "B", selectTop = TRUE, returnPop = TRUE,
  candidates = NULL, simParam = NULL, ...)
```

Arguments

`pop`	and object of `Pop-class` or `HybridPop-class`
`nFam`	the number of families to select
`trait`	the trait for selection. Either a number indicating a single trait or a function returning a vector of length nInd.
`use`	select on genetic values "gv", estimated breeding values "ebv", breeding values "bv", phenotypes "pheno", or randomly "rand"
`gender`	which gender to select. Use "B" for both, "F" for females and "M" for males. If the simulation is not using gender, the argument is ignored.
`famType`	which type of family to select. Use "B" for full-sib families, "F" for half-sib families on female side and "M" for half-sib families on the male side.
`selectTop`	selects highest values if true. Selects lowest values if false.
`returnPop`	should results be returned as a `Pop-class`. If FALSE, only the index of selected individuals is returned.
`candidates`	an optional vector of eligible selection candidates.
`simParam`	an object of `SimParam`
`...`	additional arguments if using a function for trait

Value

Returns an object of `Pop-class` or `HybridPop-class`

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=10, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)
SP$addTraitA(10)
SP$setVarE(h2=0.5)
```

```
#Create population
pop = newPop(founderPop, simParam=SP)

#Create 3 biparental families with 10 progeny
pop2 = randCross(pop, nCrosses=3, nProgeny=10, simParam=SP)

#Select best 2 families
pop3 = selectFam(pop2, 2, simParam=SP)
```

`selectInd` *Select individuals*

Description

Selects a subset of nInd individuals from a population.

Usage

```
selectInd(pop, nInd, trait = 1, use = "pheno", gender = "B",
  selectTop = TRUE, returnPop = TRUE, candidates = NULL,
  simParam = NULL, ...)
```

Arguments

`pop`	and object of `Pop-class` or `HybridPop-class`
`nInd`	the number of individuals to select
`trait`	the trait for selection. Either a number indicating a single trait or a function returning a vector of length nInd.
`use`	select on genetic values "gv", estimated breeding values "ebv", breeding values "bv", phenotypes "pheno", or randomly "rand"
`gender`	which gender to select. Use "B" for both, "F" for females and "M" for males. If the simulation is not using gender, the argument is ignored.
`selectTop`	selects highest values if true. Selects lowest values if false.
`returnPop`	should results be returned as a `Pop-class`. If FALSE, only the index of selected individuals is returned.
`candidates`	an optional vector of eligible selection candidates.
`simParam`	an object of `SimParam`
`...`	additional arguments if using a function for trait

Value

Returns an object of `Pop-class` or `HybridPop-class`

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=10, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)
SP$addTraitA(10)
SP$setVarE(h2=0.5)

#Create population
pop = newPop(founderPop, simParam=SP)

#Select best 5
pop2 = selectInd(pop, 5, simParam=SP)
```

---

selectOP *Select open pollinating plants*

Description

This function models selection in an open pollinating plant population. It allows for varying the percentage of selfing. The function also provides an option for modeling selection as occuring before or after pollination.

Usage

```
selectOP(pop, nInd, nSeeds, probSelf = 0, pollenControl = FALSE,
  trait = 1, use = "pheno", selectTop = TRUE, candidates = NULL,
  simParam = NULL, ...)
```

Arguments

pop	an object of Pop-class
nInd	the number of plants to select
nSeeds	number of seeds per plant
probSelf	percentage of seeds expected from selfing. Value ranges from 0 to 1.
pollenControl	are plants selected before pollination
trait	the trait for selection. Either a number indicating a single trait or a function returning a vector of length nInd.
use	select on genetic values "gv", estimated breeding values "ebv", breeding values "bv", phenotypes "pheno", or randomly "rand"
selectTop	selects highest values if true. Selects lowest values if false.
candidates	an optional vector of eligible selection candidates.
simParam	an object of SimParam
...	additional arguments if using a function for trait

Value

Returns an object of Pop-class

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=10, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)
SP$addTraitA(10)
SP$setVarE(h2=0.5)

#Create population
pop = newPop(founderPop, simParam=SP)

#Create new population by selecting the best 3 plant
#Assuming 50% selfing in plants and 10 seeds per plant
pop2 = selectOP(pop, nInd=3, nSeeds=10, probSelf=0.5, simParam=SP)
```

---

selectWithinFam *Select individuals within families*

Description

Selects a subset of nInd individuals from each full-sib family within a population. Will return all individuals from a full-sib family if it has less than or equal to nInd individuals.

Usage

```
selectWithinFam(pop, nInd, trait = 1, use = "pheno", gender = "B",
  famType = "B", selectTop = TRUE, returnPop = TRUE,
  candidates = NULL, simParam = NULL, ...)
```

Arguments

pop	and object of Pop-class or HybridPop-class
nInd	the number of individuals to select within a family
trait	the trait for selection. Either a number indicating a single trait or a function returning a vector of length nInd.
use	select on genetic values "gv", estimated breeding values "ebv", breeding values "bv", phenotypes "pheno", or randomly "rand"
gender	which gender to select. Use "B" for both, "F" for females and "M" for males. If the simulation is not using gender, the argument is ignored.
famType	which type of family to select. Use "B" for full-sib families, "F" for half-sib families on female side and "M" for half-sib families on the male side.

selectTop	selects highest values if true. Selects lowest values if false.
returnPop	should results be returned as a Pop-class. If FALSE, only the index of selected individuals is returned.
candidates	an optional vector of eligible selection candidates.
simParam	an object of SimParam
...	additional arguments if using a function for trait

Value

Returns an object of Pop-class or HybridPop-class

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=10, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)
SP$addTraitA(10)
SP$setVarE(h2=0.5)

#Create population
pop = newPop(founderPop, simParam=SP)

#Create 3 biparental families with 10 progeny
pop2 = randCross(pop, nCrosses=3, nProgeny=10, simParam=SP)

#Select best individual per family
pop3 = selectWithinFam(pop2, 1, simParam=SP)
```

---

self *Self individuals*

Description

Creates selfed progeny from each individual in a population. Only works when gender is "no".

Usage

```
self(pop, nProgeny = 1, parents = NULL, simParam = NULL)
```

Arguments

pop	an object of Pop-class
nProgeny	total number of selfed progeny per individual
parents	an optional vector of indices for allowable parents
simParam	an object of SimParam

Value

Returns an object of Pop-class

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=2, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)

#Create population
pop = newPop(founderPop, simParam=SP)

#Self pollinate each individual
pop2 = self(pop, simParam=SP)
```

---

selIndex *Selection index*

Description

Calculates values of a selection index given trait values and weights. This function is intended to be used in combination with selection functions working on populations such as selectInd.

Usage

```
selIndex(Y, b, scale = FALSE)
```

Arguments

Y	a matrix of trait values
b	a vector of weights
scale	should Y be scaled and centered

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=10, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)
#Model two genetically correlated traits
G = 1.5*diag(2)-0.5 #Genetic correlation matrix
SP$addTraitA(10, mean=c(0,0), var=c(1,1), corA=G)
SP$setVarE(h2=c(0.5,0.5))
```

```
#Create population
pop = newPop(founderPop, simParam=SP)

#Calculate Smith-Hazel weights
econWt = c(1, 1)
b = smithHazel(econWt, varG(pop), varP(pop))

#Selection 2 best individuals using Smith-Hazel index
#selIndex is used as a trait
pop2 = selectInd(pop, nInd=2, trait=selIndex,
                 simParam=SP, b=b)
```

---

selInt *Selection intensity*

Description

Calculates the standardized selection intensity

Usage

```
selInt(p)
```

Arguments

p	the proportion of individuals selected

Examples

```
selInt(0.1)
```

---

setEBV *Set EBV*

Description

Sets a population's EBV with genomic estimated values from RRBLUP, RRBLUP_GCA, or RRBLUP_SCA.

Usage

```
setEBV(pop, solution, gender = NULL, useGV = FALSE, append = FALSE,
  simParam = NULL)
```

Arguments

pop	an object of Pop-class
solution	an object of RRsol-class, SCAsol-class, or GCAsol-class
gender	either NULL, "male" or "female". If solution is GCAsol-class or SCAsol-class the EBV is the GCA if used in the corresponding pool
useGV	if model is RRDsol-class, setting this parameter to TRUE will give use estimated genetic values. Otherwise, you get estimated breeding values that depend on the population's allele frequency.
append	should EBVs be appended to existing EBVs
simParam	an object of SimParam

Value

Returns an object of Pop-class

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=10, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)
SP$addTraitA(10)
SP$setVarE(h2=0.5)
SP$addSnpChip(10)

#Create population
pop = newPop(founderPop, simParam=SP)

#Run GS model and set EBV
ans = RRBLUP(pop, simParam=SP)
pop = setEBV(pop, ans, simParam=SP)

#Evaluate accuracy
cor(gv(pop), ebv(pop))
```

---

setPheno *Set phenotypes*

Description

Sets phenotypes for all traits by adding random error from a multivariate normal distribution.

Usage

```
setPheno(pop, varE = NULL, reps = 1, fixEff = 1L, p = 0.5,
  onlyPheno = FALSE, simParam = NULL)
```

Arguments

pop	an object of Pop-class or HybridPop-class
varE	error variances for phenotype. A vector of length nTraits for independent error or a square matrix of dimensions nTraits for correlated errors. If NULL, value in simParam is used.
reps	number of replications for phenotype. See details.
fixEff	fixed effect to assign to the population. Used by genomic selection models only.
p	the p-value for the environmental covariate used by GxE traits.
onlyPheno	should only the phenotype be returned, see return
simParam	an object of SimParam

Details

The reps parameter is for convient representation of replicated data. It is intended to represent replicated yield trials in plant breeding programs. In this case, varE is set to the plot error and reps is set to the number of plots per entry. The resulting phenotype represents entry means.

Value

Returns an object of Pop-class or HybridPop-class if onlyPheno=FALSE, if onlyPheno=TRUE a matrix is returned

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=10, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)
SP$addTraitA(10)

#Create population
pop = newPop(founderPop, simParam=SP)

#Add phenotype with error variance of 1
pop = setPheno(pop, varE=1)
```

---

setPhenoGCA *Set GCA as phenotype*

Description

Calculates general combining ability from a set of testers and returns these values as phenotypes for a population.

Usage

```
setPhenoGCA(pop, testers, use = "pheno", varE = NULL, reps = 1,
  fixEff = 1L, p = 0.5, inbred = FALSE, onlyPheno = FALSE,
  simParam = NULL)
```

Arguments

pop	an object of Pop-class
testers	an object of Pop-class
use	true genetic value (gv) or phenotypes (pheno, default)
varE	error variances for phenotype if use="pheno". A vector of length nTraits for independent error or a square matrix of dimensions nTraits for correlated errors.
reps	number of replications for phenotype. See details.
fixEff	fixed effect to assign to the population. Used by genomic selection models only.
p	the p-value for the environmental covariate
inbred	are both pop and testers fully inbred. They are only fully inbred if created by newPop using inbred founders or by the makeDH function
onlyPheno	should only the phenotype be returned, see return
simParam	an object of SimParam

Details

The reps parameter is for convient representation of replicated data. It was intended for representation of replicated yield trials in plant breeding programs. In this case, varE is set to the plot error and reps is set to the number plots per entry. The resulting phenotype would reflect the mean of all replications.

Value

Returns an object of Pop-class or a matrix if onlyPheno=TRUE

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=10, nChr=1, segSites=10, inbred=TRUE)

#Set simulation parameters
SP = SimParam$new(founderPop)
SP$addTraitA(10)

#Create population
pop = newPop(founderPop, simParam=SP)

#Set phenotype to average per
pop2 = setPhenoGCA(pop, pop, use="gv", inbred=TRUE, simParam=SP)
```

SimParam *Simulation parameters*

Description

Container for global simulation parameters. Saving this object as SP will allow it to be accessed by function defaults.

Usage

```
SimParam
```

Format

An object of class `R6ClassGenerator` of length 24.

Fields

`nChr` number of chromosomes

`nTraits` number of traits

`nSnpChips` number of SNP chips

`segSites` segregating sites per chromosome

`gender` is gender used for mating

`genMap` "matrix" of chromosome genetic maps

`femaleMap` "matrix" of chromosome genetic maps for females

`maleMap` "matrix" of chromosome genetic maps for males

`sepMap` are there seperate genetic maps for males and females

`femaleCentromere` position of centromere on female genetic map

`maleCentromere` position of centromere on male genetic map

`recombRatio` ratio of genetic recombination in females relative to male

`traits` list of trait

`snpChips` list of SNP chips

`potQtl` list of potential QTL segregating sites

`potSnp` list of potential SNP segregating sites

`lastId` last ID number assigned

`isTrackPed` is pedigree being tracked

`pedigree` pedigree matrix for all individuals

`isTrackRec` is recombination being tracked

`recHist` list of historic recombination events

`varA` additive genetic variance in founderPop

`varG` total genetic variance in founderPop

`varE` default error variance

`founderPop` the founder population used for scaling traits

`quadProb` the probability of quadrivalent formation

`nThreads` number of threads used on platforms with OpenMP support

`version` the version of AlphaSimR used to generate this object

`SimParam_addSnpChip` *Add SNP chip*

Description

Randomly assigns eligble SNPs to a SNP chip

Arguments

`nSnpPerChr`	number of SNPs per chromosome. Can be a single value or nChr values.
`force`	should the check for a running simulation be ignored. Only set to TRUE if you know what you are doing.

Usage

SP$addSnpChip(nSnpPerChr, force = FALSE)

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=10, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)
SP$addSnpChip(10)
```

SimParam_addStructuredSnpChips *Add Structured SNP chips*

Description

Randomly selects the number of snps in structure and then assigns them to chips based on structure

Arguments

nSnpPerChr	number of SNPs per chromosome. Can be a single value or nChr values.
structure	a matrix. Rows are snp chips, columns are chips. If value is true then that snp is on that chip.
force	should the check for a running simulation be ignored. Only set to TRUE if you know what you are doing.

Usage

SP$addStructuredSnpChip(nSnpPerChr, structure, force = FALSE)

SimParam_addTraitA *Add additive traits*

Description

Randomly assigns eligble QTLs for one or more additive traits. If simulating more than one trait, all traits will be pleiotrophic with correlated additive effects.

Arguments

nQtlPerChr	number of QTLs per chromosome. Can be a single value or nChr values.
mean	a vector of desired mean genetic values for one or more traits
var	a vector of desired genetic variances for one or more traits
corA	a matrix of correlations between additive effects
gamma	should a gamma distribution be used instead of normal
shape	the shape parameter for the gamma distribution
force	should the check for a running simulation be ignored. Only set to TRUE if you know what you are doing.

Usage

SP$addTraitA(nQtlPerChr, mean = 0, var = 1, corA = NULL, gamma = FALSE, shape = 1, force = FALSE)

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=10, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)
SP$addTraitA(10)
```

SimParam_addTraitAD *Add additive and dominance traits*

Description

Randomly assigns eligble QTLs for one or more traits with dominance. If simulating more than one trait, all traits will be pleiotrophic with correlated effects.

Arguments

nQtlPerChr	number of QTLs per chromosome. Can be a single value or nChr values.
mean	a vector of desired mean genetic values for one or more traits
var	a vector of desired genetic variances for one or more traits
meanDD	mean dominance degree
varDD	variance of dominance degree
corA	a matrix of correlations between additive effects
corDD	a matrix of correlations between dominance degrees
useVarA	tune according to additive genetic variance if true. If FALSE, tuning is performed according to total genetic variance.
gamma	should a gamma distribution be used instead of normal
shape	the shape parameter for the gamma distribution
force	should the check for a running simulation be ignored. Only set to TRUE if you know what you are doing.

Usage

SP$addTraitAD(nQtlPerChr, mean = 0, var = 1, meanDD = 0, varDD = 0, corA = NULL, corDD = NULL, useVarA = TRUE, gamma = FALSE, shape = 1, force = FALSE)

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=10, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)
SP$addTraitAD(10, meanDD=0.5)
```

SimParam_addTraitADE *Add additive, dominance and epistasis traits*

Description

Randomly assigns eligble QTLs for one or more traits with dominance and epistasis. If simulating more than one trait, all traits will be pleiotrophic with correlated effects.

Arguments

nQtlPerChr	number of QTLs per chromosome. Can be a single value or nChr values.
mean	a vector of desired mean genetic values for one or more traits
var	a vector of desired genetic variances for one or more traits
meanDD	mean dominance degree
varDD	variance of dominance degree
relAA	the relative variance of additive-by-additive effects compared to the additive effects
corA	a matrix of correlations between additive effects
corDD	a matrix of correlations between dominance degrees
corAA	a matrix of correlations between additive-by-additive effects
useVarA	tune according to additive genetic variance if true. If FALSE, tuning is performed according to total genetic variance.
gamma	should a gamma distribution be used instead of normal
shape	the shape parameter for the gamma distribution
force	should the check for a running simulation be ignored. Only set to TRUE if you know what you are doing.

Usage

SP$addTraitADE(nQtlPerChr, mean = 0, var = 1, meanDD = 0, varDD = 0, relAA = 0, corA = NULL, corDD = NULL, corAA = NULL, useVarA = TRUE, gamma = FALSE, shape = 1, force = FALSE)

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=10, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)
SP$addTraitADE(10)
```

SimParam_addTraitADEG *Add an additive, dominance, and epistasis GxE trait*

Description

Randomly assigns eligble QTLs for a trait with dominance, epistasis and GxE.

Arguments

nQtlPerChr	number of QTLs per chromosome. Can be a single value or nChr values.
mean	a vector of desired mean genetic values for one or more traits
var	a vector of desired genetic variances for one or more traits
varGxE	a vector of total genotype-by-environment variances for the traits
varEnv	a vector of environmental variances for one or more traits
meanDD	mean dominance degree
varDD	variance of dominance degree
relAA	the relative variance of additive-by-additive effects compared to the additive effects
corA	a matrix of correlations between additive effects
corDD	a matrix of correlations between dominance degrees
corAA	a matrix of correlations between additive-by-additive effects
corGxE	a matrix of correlations between GxE effects
useVarA	tune according to additive genetic variance if true
gamma	should a gamma distribution be used instead of normal
shape	the shape parameter for the gamma distribution
force	should the check for a running simulation be ignored. Only set to TRUE if you know what you are doing.

Usage

SP$addTraitADEG(nQtlPerChr, mean = 0, var = 1, varGxE = 1e-6, varEnv = 0, meanDD = 0, varDD = 0, relAA = 0, corA = NULL, corDD = NULL, corAA = NULL, corGxE = NULL, useVarA = TRUE, gamma = FALSE, shape = 1, force = FALSE)

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=10, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)
SP$addTraitADEG(10, meanDD=0.5, varGxE=2)
```

SimParam_addTraitADG *Add an additive and dominance GxE trait*

Description

Randomly assigns eligble QTLs for a trait with dominance and GxE.

Arguments

nQtlPerChr	number of QTLs per chromosome. Can be a single value or nChr values.
mean	a vector of desired mean genetic values for one or more traits
var	a vector of desired genetic variances for one or more traits
varGxE	a vector of total genotype-by-environment variances for the traits
varEnv	a vector of environmental variances for one or more traits
meanDD	mean dominance degree
varDD	variance of dominance degree
corA	a matrix of correlations between additive effects
corDD	a matrix of correlations between dominance degrees
corGxE	a matrix of correlations between GxE effects
useVarA	tune according to additive genetic variance if true
gamma	should a gamma distribution be used instead of normal
shape	the shape parameter for the gamma distribution
force	should the check for a running simulation be ignored. Only set to TRUE if you know what you are doing.

Usage

SP$addTraitADG(nQtlPerChr, mean = 0, var = 1, varGxE = 1e-6, varEnv = 0, meanDD = 0, varDD = 0, corA = NULL, corDD = NULL, corGxE = NULL, useVarA = TRUE, gamma = FALSE, shape = 1, force = FALSE)

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=10, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)
SP$addTraitADG(10, meanDD=0.5, varGxE=2)
```

SimParam_addTraitAE *Add additive and epistasis traits*

Description

Randomly assigns eligble QTLs for one or more additive and epistasis traits. If simulating more than one trait, all traits will be pleiotrophic with correlated additive effects.

Arguments

nQtlPerChr	number of QTLs per chromosome. Can be a single value or nChr values.
mean	a vector of desired mean genetic values for one or more traits
var	a vector of desired genetic variances for one or more traits
relAA	the relative variance of additive-by-additive effects compared to the additive effects
corA	a matrix of correlations between additive effects
corAA	a matrix of correlations between additive-by-additive effects
useVarA	tune according to additive genetic variance if true. If FALSE, tuning is performed according to total genetic variance.
gamma	should a gamma distribution be used instead of normal
shape	the shape parameter for the gamma distribution
force	should the check for a running simulation be ignored. Only set to TRUE if you know what you are doing.

Usage

SP$addTraitAA(nQtlPerChr, mean = 0, var = 1, relAA = 0, corA = NULL, corAA = NULL, gamma = FALSE, shape = 1, force = FALSE)

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=10, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)
SP$addTraitAE(10)
```

SimParam_addTraitAEG *Add additive and epistasis GxE traits*

Description

Randomly assigns eligble QTLs for one or more additive and epistasis GxE traits. If simulating more than one trait, all traits will be pleiotrophic with correlated effects.

Arguments

nQtlPerChr	number of QTLs per chromosome. Can be a single value or nChr values.
mean	a vector of desired mean genetic values for one or more traits
var	a vector of desired genetic variances for one or more traits
relAA	the relative variance of additive-by-additive effects compared to the additive effects
varGxE	a vector of total genotype-by-environment variances for the traits
varEnv	a vector of environmental variances for one or more traits
corA	a matrix of correlations between additive effects
corAA	a matrix of correlations between additive-by-additive effects
corGxE	a matrix of correlations between GxE effects
useVarA	tune according to additive genetic variance if true. If FALSE, tuning is performed according to total genetic variance.
gamma	should a gamma distribution be used instead of normal
shape	the shape parameter for the gamma distribution
force	should the check for a running simulation be ignored. Only set to TRUE if you know what you are doing.

Usage

SP$addTraitAG(nQtlPerChr, mean = 0, var = 1, relAA = 0, varGxE = 1e-6, varEnv = 0, corA = NULL, corAA = NULL, corGxE = NULL, useVarA = TRUE, gamma = FALSE, shape = 1)

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=10, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)
SP$addTraitAEG(10, varGxE=2)
```

SimParam_addTraitAG *Add additive GxE traits*

Description

Randomly assigns eligble QTLs for one ore more additive GxE traits. If simulating more than one trait, all traits will be pleiotrophic with correlated effects.

Arguments

nQtlPerChr	number of QTLs per chromosome. Can be a single value or nChr values.
mean	a vector of desired mean genetic values for one or more traits
var	a vector of desired genetic variances for one or more traits
varGxE	a vector of total genotype-by-environment variances for the traits
varEnv	a vector of environmental variances for one or more traits
corA	a matrix of correlations between additive effects
corGxE	a matrix of correlations between GxE effects
gamma	should a gamma distribution be used instead of normal
shape	the shape parameter for the gamma distribution
force	should the check for a running simulation be ignored. Only set to TRUE if you know what you are doing.

Usage

SP$addTraitAG(nQtlPerChr, mean = 0, var = 1, varGxE = 1e-6, varEnv = 0, corA = NULL, corGxE = NULL, gamma = FALSE, shape = 1)

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=10, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)
SP$addTraitAG(10, varGxE=2)
```

SimParam_manAddSnpChip *Manually add SNP chip*

Description

Adds a new LociMap-class for a SNP chip.

Arguments

lociMap	a new LociMap-class
force	should the check for a running simulation be ignored. Only set to TRUE if you know what you are doing.

Usage

SP$manAddSnpChips(lociMap, force = FALSE)

SimParam_manAddTrait *Manually add trait*

Description

Add a new trait to the simulation.

Arguments

lociMap	a new object descended from LociMap-class
varA	the value for varA in the base population, optional
varG	the value for varG in the base population, optional
varE	default error variance for phenotype, optional
force	should the check for a running simulation be ignored. Only set to TRUE if you know what you are doing

Usage

SP$manAddTrait(lociMap, varA = NA_real_, varG = NA_real_, varE = NA_real_, force = FALSE)

SimParam_new *Create new simulation*

Description

Starts the process of building a new simulation by creating a new SimParam object and assigning a founder population to the class. It is recommended that you save the object with the name "SP", because subsequent functions will check your global enviroment for an object of this name if their simParam arguments are NULL. This allows you to call these functions without explicitly supplying a simParam argument with every call.

Arguments

founderPop	an object of MapPop-class

Usage

SimParam$new(founderPop)

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=10, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)
```

SimParam_removeFounderPop *Remove founder population*

Description

Removes the founder population from the founderPop field. This can be ran after all traits have been added to reduce the size of the SimParam object.

Usage

SP$removeFounderPop()

SimParam_removeSnpChip *Remove SNP chip*

Description

Removes designated SNP chip(s).

Arguments

chips	a vector of SNP chips to remove
force	should the check for a running simulation be ignored. Only set to TRUE if you know what you are doing.

Usage

SP$removeSnpChip(chips, force = FALSE)

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=10, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)
SP$addSnpChip(10)
SP$nSnpChips
SP$removeSnpChip(1)
SP$nSnpChips
```

SimParam_removeTrait *Remove trait*

Description

Removes designated trait(s).

Arguments

traits	a vector of traits to remove
force	should the check for a running simulation be ignored. Only set to TRUE if you know what you are doing.

Usage

SP$removeTrait(traits, force = FALSE)

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=10, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)
SP$addTraitA(10)
SP$nTraits
SP$removeTrait(1)
SP$nTraits
```

SimParam_rescaleTraits

Rescale traits

Description

Linearly scales all traits to achieve desired values of means and variances in the founder population.

Arguments

mean	a vector of new trait means
var	a vector of new trait variances
varEnv	a vector of new environmental variances
varGxE	a vector of new GxE variances
useVarA	tune according to additive genetic variance if true

Usage

SP$rescaleTraits(mean = 0, var = 1, relAA = 1e-6, varEnv = 0, varGxE = 1e-6, useVarA = TRUE)

Note

By default the founder population is the population used to initalize the SimParam object. This population can be changed using the switchFounderPop function in the SimParam object (see SimParam_switchFounderPop). You must run resetPop on existing populations to obtain the new trait values.

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=10, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)
```

```
SP$addTraitA(10)

#Create population
pop = newPop(founderPop, simParam=SP)
meanG(pop)

#Change mean to 1
SP$rescaleTraits(mean=1)
#Run resetPop for change to take effect
pop = resetPop(pop, simParam=SP)
meanG(pop)
```

SimParam_resetPed *Reset pedigree*

Description

Resets the internal lastId, the pedigree and recombination tracking, if it is being used, to the supplied lastId. Be careful using this function because it may introduce bug if you supsequently use individuals that come from a portion the pedigree that is being reset.

Arguments

lastId	last ID to include in pedigree

Usage

SP$resetPed(lastId = 0L)

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=10, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)

#Create population
pop = newPop(founderPop, simParam=SP)
pop@id # 1:10

#Create another population after reseting pedigree
SP$resetPed()
pop2 = newPop(founderPop, simParam=SP)
pop2@id # 1:10
```

SimParam_restrSegSites *Restrict segregating sites*

Description

Sets restrictions on which segregating sites can serve as SNP and/or QTL.

Arguments

maxQtl	the maximum number of segSites for QTLs. Can be a single value or a vector values for each chromosome.
maxSnp	the maximum number of segSites for SNPs. Can be a single value or a vector values for each chromosome.
overlap	should SNP and QTL sites be allowed to overlap.
minSnpFreq	minimum allowable frequency for SNP loci. No minimum SNP frequency is used if value is NULL.
force	should the check for a running simulation be ignored. Only set to TRUE if you know what you are doing.

Usage

SP$restrSegSites(maxQtl = 0, maxSnp = 0, snpQtlOverlap = FALSE, minSnpFreq = NULL, force = FALSE)

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=10, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)
SP$restrSegSites(maxQtl=5, maxSnp=5)
```

SimParam_setCorE *Set correlated error variance*

Description

Defines a correlation structure for default error variances. You must call SimParam_setVarE first to define the default error variances.

Arguments

corE	a correlation matrix for the error variances

Usage

SP$setCorE(corE)

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=10, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)
SP$addTraitA(10, mean=c(0,0), var=c(1,1), corA=diag(2))
SP$setVarE(varE=c(1,1))
E = 0.5*diag(2)+0.5 #Positively correlated error
SP$setCorE(E)
```

SimParam_setGender *Set gender in simulation*

Description

Changes how gender is used in the simulation. The default gender of a simulation is "no". To add gender to the simulation, run this function with "yes_sys" or "yes_rand". The value "yes_sys" will systematically assign gender to newly created individuals as first male, then female. Thus, odd numbers of individuals will have one more male than female. The value "yes_rand" will randomly assign gender to individuals.

Arguments

gender	acceptable value are "no", "yes_sys", or "yes_rand"
force	should the check for a running simulation be ignored. Only set to TRUE if you know what you are doing.

Usage

SP$setGender(gender, force = FALSE)

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=10, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)
SP$setGender("yes_sys")
```

SimParam_setRecRatio *Set gender specific recombination*

Description

Defines a gender specific recombination ratio. The ratio is defined as the amount of recombination in females relative to male. Thus, a value of 1 (default) specifies equal recombination rates in both males and females. A value of 2 specifies twice as much recombination in females and a value of 0.5 specifies half as much recombination in females.

Arguments

ratio	any value greater than 0
force	should the check for a running simulation be ignored. Only set to TRUE if you know what you are doing.

Usage

SP$setRecRatio(ratio, force = FALSE)

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=10, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)
SP$setRecRatio(2) #Twice as much recombination in females
```

SimParam_setTrackPed *Set pedigree tracking*

Description

Sets pedigree tracking for the simulation. By default pedigree tracking is turned off. When turned on, the pedigree of all individuals created will be tracked, except those created by hybridCross. Turning off pedigree tracking will turn off recombination tracking if it is turned on.

Arguments

isTrackPed	should pedigree tracking be on.
force	should the check for a running simulation be ignored. Only set to TRUE if you know what you are doing.

Usage

SP$setTrackPed(isTrackPed, force = FALSE)

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=10, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)
SP$setTrackPed(TRUE)
```

SimParam_setTrackRec *Set recombination tracking*

Description

Sets recombination tracking for the simulation. By default recombination tracking is turned off. When turned on recombination tracking will also turn on pedigree tracking. Recombination tracking keeps records of all individuals created, except those created by `hybridCross`, because their pedigree is not tracked.

Arguments

`isTrackRec`	should recombination tracking be on.
`force`	should the check for a running simulation be ignored. Only set to TRUE if you know what you are doing.

Usage

SimParam$setTrackRec(isTrackRec, force = FALSE)

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=10, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)
SP$setTrackRec(TRUE)
```

SimParam_setVarE *Set simulation error variance*

Description

Defines a default value for error variances in the simulation.

Arguments

h2	a vector of desired narrow-sense heritabilities
H2	a vector of desired broad-sense heritabilities
varE	a vector of error variances

Usage

SP$setVarE(h2 = NULL, H2 = NULL, varE = NULL)

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=10, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)
SP$addTraitA(10)
SP$setVarE(h2=0.5)
```

SimParam_switchFemaleMap *Switch female genetic map*

Description

Replaces existing female genetic map.

Arguments

genMap	a list of length nChr containing numeric vectors for the position of each segregating site on a chromosome.
centromere	a numeric vector of centromere positions. If NULL, the centromere are assumed to be metacentric.

Usage

SP$switchFemaleMap(genMap)

SimParam_switchFounderPop *Switch founder population*

Description

Switches the founder population in the founderPop field. This may be desirable if traits are to be tuned to a population derived from the original founderPop. Note that no checking is performed to verify that the genetic map and/or number of segregating sites hasn't changed. The new founderPop can be MapPop-class or RawPop-class

Usage

SP$switchFounderPop(founderPop)

SimParam_switchGenMap *Switch genetic map*

Description

Replaces existing genetic map.

Arguments

genMap	a list of length nChr containing numeric vectors for the position of each segregating site on a chromosome.
centromere	a numeric vector of centromere positions. If NULL, the centromere are assumed to be metacentric.

Usage

SP$switchGenMap(genMap)

SimParam_switchMaleMap

Switch male genetic map

Description

Replaces existing male genetic map.

Arguments

genMap	a list of length nChr containing numeric vectors for the position of each segregating site on a chromosome.
centromere	a numeric vector of centromere positions. If NULL, the centromere are assumed to be metacentric.

Usage

SP$switchMaleMap(genMap)

SimParam_switchSnpChip

Switch SNP chip

Description

Replaces the LociMap-class for a SNP chip.

Arguments

lociMap	a new LociMap-class
chip	an integer indicating which chip to replace
force	should the check for a running simulation be ignored. Only set to TRUE if you know what you are doing.

Usage

SP$switchSnpChips(lociMap, chip, force = FALSE)

SimParam_switchTrait *Switch trait*

Description

Replaces an existing trait.

Arguments

lociMap	a new object descended from LociMap-class
trait	an integer indicating which trait to replace
varA	a new value for varA in the base population. If NULL, the existing value is retained.
varG	a new value for varG in the base population. If NULL, the existing value is retained.
force	should the check for a running simulation be ignored. Only set to TRUE if you know what you are doing.

Usage

SP$switchTrait(lociMap, trait, varA = NULL, varG = NULL, force = FALSE)

smithHazel *Calculate Smith-Hazel weights*

Description

Calculates weights for Smith-Hazel index given economice weights and phenotypic and genotypic variance-covariance matrices.

Usage

```
smithHazel(econWt, varG, varP)
```

Arguments

econWt	vector of economic weights
varG	the genetic variance-covariance matrix
varP	the phenotypic variance-covariance matrix

Value

a vector of weight for calculating index values

Examples

```
G = 1.5*diag(2)-0.5
E = diag(2)
P = G+E
wt = c(1,1)
smithHazel(wt, G, P)
```

`TraitA-class` *Additive trait*

Description

Extends `LociMap-class` to model additive traits

Slots

`addEff` additive effects

`intercept` adjustment factor for gc

`TraitAD-class` *Additive and dominance trait*

Description

Extends `TraitA-class` to add dominance

Slots

`domEff` dominance effects

`TraitADE-class` *Additive, dominance, and epistatic trait*

Description

Extends `TraitAD-class` to add epistasis

Slots

`epiEff` epistatic effects

TraitADEG-class *Additive, dominance, epistasis, and GxE trait*

Description

Extends TraitADE-class to add GxE effects

Slots

gxeEff GxE effects

gxeInt GxE intercept

envVar Environmental variance

TraitADG-class *Additive, dominance and GxE trait*

Description

Extends TraitAD-class to add GxE effects

Slots

gxeEff GxE effects

gxeInt GxE intercept

envVar Environmental variance

TraitAE-class *Additive and epistatic trait*

Description

Extends TraitA-class to add epistasis

Slots

epiEff epistatic effects

TraitAEG-class *Additive, epistasis and GxE trait*

Description

Extends TraitAE-class to add GxE effects

Slots

gxeEff GxE effects

gxeInt GxE intercept

envVar Environmental variance

TraitAG-class *Additive and GxE trait*

Description

Extends TraitA-class to add GxE effects

Slots

gxeEff GxE effects

gxeInt GxE intercept

envVar Environmental variance

usefulness *Usefulness criterion*

Description

Calculates the usefulness criterion

Usage

```
usefulness(pop, trait = 1, use = "gv", p = 0.1, selectTop = TRUE,
  simParam = NULL, ...)
```

Arguments

pop	and object of Pop-class or HybridPop-class
trait	the trait for selection. Either a number indicating a single trait or a function returning a vector of length nInd.
use	select on genetic values (gv, default), estimated breeding values (ebv), breeding values (bv), or phenotypes (pheno)
p	the proportion of individuals selected
selectTop	selects highest values if true. Selects lowest values if false.
simParam	an object of SimParam
...	additional arguments if using a function for trait

Value

Returns a numeric value

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=2, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)
SP$addTraitA(10)

#Create population
pop = newPop(founderPop, simParam=SP)

#Determine usefulness of population
usefulness(pop, simParam=SP)

#Should be equivalent to GV of best individual
max(gv(pop))
```

---

varA *Additive variance*

Description

Returns additive variance for all traits

Usage

```
varA(pop, simParam = NULL)
```

Arguments

pop	an object of Pop-class
simParam	an object of SimParam

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=10, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)
SP$addTraitAD(10, meanDD=0.5)
SP$setVarE(h2=0.5)

#Create population
pop = newPop(founderPop, simParam=SP)
varA(pop, simParam=SP)
```

---

varAA *Additive-by-additive epistatic variance*

Description

Returns additive-by-additive epistatic variance for all traits

Usage

```
varAA(pop, simParam = NULL)
```

Arguments

pop	an object of Pop-class
simParam	an object of SimParam

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=10, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)
SP$addTraitAD(10, meanDD=0.5)
SP$setVarE(h2=0.5)

#Create population
pop = newPop(founderPop, simParam=SP)
varAA(pop, simParam=SP)
```

varD *Dominance variance*

Description

Returns dominance variance for all traits

Usage

```
varD(pop, simParam = NULL)
```

Arguments

pop	an object of Pop-class
simParam	an object of SimParam

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=10, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)
SP$addTraitAD(10, meanDD=0.5)
SP$setVarE(h2=0.5)

#Create population
pop = newPop(founderPop, simParam=SP)
varD(pop, simParam=SP)
```

varG *Total genetic variance*

Description

Returns total genetic variance for all traits

Usage

```
varG(pop)
```

Arguments

pop	an object of Pop-class or HybridPop-class

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=10, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)
SP$addTraitA(10)
SP$setVarE(h2=0.5)

#Create population
pop = newPop(founderPop, simParam=SP)
varG(pop)
```

---

varP *Phenotypic variance*

Description

Returns phenotypic variance for all traits

Usage

```
varP(pop)
```

Arguments

pop	an object of Pop-class or HybridPop-class

Examples

```
#Create founder haplotypes
founderPop = quickHaplo(nInd=10, nChr=1, segSites=10)

#Set simulation parameters
SP = SimParam$new(founderPop)
SP$addTraitA(10)
SP$setVarE(h2=0.5)

#Create population
pop = newPop(founderPop, simParam=SP)
varP(pop)
```

writePlink *Writes a Pop-class as PLINK files*

Description

Writes a Pop-class as PLINK PED and MAP files

Usage

```
writePlink(pop, baseName, trait = 1L, snpChip = 1L, simParam = NULL,
  chromLength = 10L^8)
```

Arguments

pop	an object of Pop-class
baseName	a character. Basename of PED and MAP files.
trait	an integer. Which phenotype trait should be used.
snpChip	an integer. Which SNP array should be used.
simParam	an object of SimParam
chromLength	an integer. The size of chromosomes in base pairs; assuming all chromosomes are of the same size.

writeRecords *Write data records*

Description

Saves a population's phenotypic and marker data to a directory.

Usage

```
writeRecords(pop, dir, snpChip = 1, useQtl = FALSE,
  includeHaplo = FALSE, append = TRUE, simParam = NULL)
```

Arguments

pop	an object of Pop-class
dir	path to a directory for saving output
snpChip	which SNP chip genotype to save. If useQtl=TRUE, this value will indicate which trait's QTL genotype to save. A value of 0 will skip writing a snpChip.
useQtl	should QTL genotype be written instead of SNP chip genotypes.
includeHaplo	should markers be seperated by female and male haplotypes.

append	if true, new records are added to any existing records. If false, any existing records are deleted before writing new records. Note that this will delete all files in the 'dir' directory.
simParam	an object of SimParam

Index