Package 'xbreed'

October 8, 2019

Title Genomic Simulation of Purebred and Crossbred Populations

Version 1.0.1.1

Description Simulation of purebred and crossbred genomic data as well as pedigree and phenotypes are possible by this package. 'xbreed' can be used for the simulation of populations with flexible genome structures and trait genetic architectures. It can also be used to evaluate breeding schemes and generate genetic data to test statistical tools.

Depends R (>= 3.3.2)

Imports stats, utils, BGLR

Suggests knitr, rmarkdown

VignetteBuilder knitr

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

NeedsCompilation yes

Author Hadi Esfandyari [aut, cre],
Anders Christian Sørensen [aut]

Maintainer Hadi Esfandyari <esfandyari.hadi@gmail.com>

Repository CRAN

Date/Publication 2019-10-08 05:45:48 UTC

R topics documented:

calc_LD 2
make_hp 4
make_rp 7
sample_hp 12
xbreed 16

Index **25**

`calc_LD` *Calculate linkage disequilibrium*

Description

Different measures of linkage disequilibrium (LD) such as D,r and r^2 are calculated for phased genotypes. LD measurements can be calculated both for adjacent and pairwise loci. Decay of LD between marker pairs can be assessed as well.

Usage

```
calc_LD(mat, MAF, method, LD_summary, saveAt, linkage_map, interval)
```

Arguments

`mat`	(`matrix`) Matrix with phased genotypes of individuals coded as 11,12,21,22 for genotypes AA, Aa, aA and AA respectively. Dimension is $n * p$ where n is number of individuals and p is twice the number of loci.
`MAF`	*Optional* Minor allele frequency threshold for LD calculation. Loci with minor allele frequency less than `MAF` will be excluded. Default: 0.05.
`method`	*Optional* (`character`) Method to be used for calculation of LD. Possible options are: • "adjacent" LD will be calculated for adjacent loci only. • "pairwise" LD will be calculated for all pairwise loci. Default: "adjacent"
`LD_summary`	*Optional* (`Logical`) Display LD calculation summary if is not `FALSE`. Default: `True`.
`saveAt`	*Optional* (`character`). Name to be used to save output files.
`linkage_map`	*Optional* (`data.frame`) or (`vector`) Linkage map of the loci for the measurement of LD decay. **Note:** both argument `linkage_map` and `interval` should be present in the function and also argument `method` should be set to "pairwise" for the calculation of LD decay.
`interval`	*Optional* Interval to be used for grouping loci by their pairwise distance. See `details`.

Details

The extent of LD is an important factor both in association studies and genomic selection. Commonly used measure to calculate LD between loci A and B is Pearson coefficient (r) of correlation as:

$$r = D/\sqrt{(p_1 p_2 q_1 q_2)}$$

where D is

$$D_{ij} = p(A_iB_j) - p(A_i)p(B_j)$$

However, squared coefficient of correlation r^2 is often used to remove the arbitrary sign introduced:

$$r^2_{ij} = D^2_{ij}/(p(A_i)(1-p(A_i))p(B_j)(1-p(B_j)))$$

To determine the decay of LD with increasing distance between loci (SNPs), the average r^2 can be expressed as a function of distance between SNPs. SNP pairs are grouped by their pairwise distance into intervals defined by the user in argument `interval`. The average r^2 for SNP pairs in each interval are estimated as the mean of all r^2 within that interval.

Value

`list` with data of LD calculations.

$Mean_r2 Mean r^2 for provided genotypes based on the method specified.

$ld_data Data frame with 12 columns including pair id, frequencies of alleles and haplotypes for each pair as well as measurements of LD.

$ld_decay Data for LD decay including the average r^2 for loci pairs in each interval.

Examples

```
# Calculate mean r2 and LD decay.

genome<-data.frame(matrix(NA, nrow=1, ncol=6))
names(genome)<-c("chr","len","nmrk","mpos","nqtl","qpos")
genome$chr<-c(1)
genome$len<-c(100)
genome$nmrk<-c(100)
genome$mpos<-c("rnd")
genome$nqtl<-c(50)
genome$qpos<-c("even")
genome

hp<-make_hp(hpsize=100,
ng=10,h2=0.3,phen_var=1 ,genome=genome,
mutr=2.5e-4)

# Mean r2

mat<-hp$hp_mrk[,-1]
rLD<-calc_LD(mat=mat,MAF=0.1,method='adjacent',LD_summary=TRUE)

# LD decay

 linkage_map<-hp$linkage_map_mrk[,3]
```

```
 rLD<-calc_LD(mat=mat,MAF=0.1,method='pairwise'
,LD_summary=TRUE,linkage_map=linkage_map,interval=5)

rLD$ld_decay
```

---

make_hp *Create historical population*

Description

Simulates historical generations to establish mutation-drift equilibrium and create linkage disequilibrium.

Usage

```
make_hp(hpsize, ng, genome, h2, d2, phen_var, mutr, laf, sel_seq_qtl,
  sel_seq_mrk, saveAt)
```

Arguments

hpsize	Size of historical population. Range: $0 <$ hpsize ≤ 10000.
ng	Number of generations. Range: $0 \leq$ ng ≤ 10000.
genome	data.frame specifying genome parameters with dimension $n * p$ where n is number of chromosomes and p=6 are columns defined as following: "chr" Chromosome id starting from 1 to the number of specified chromosomes (Max=100). "len" Chromosome length in cM in range $0 <$ len ≤ 10000. "nmrk" Number of markers in each chromosome in range $1 \leq$ nmrk ≤ 10000. "mpos" Marker position along chromosome with options: 'rnd' - samples marker positions from uniform distribution. 'even' - markers are evenly spaced. "nqtl" Number of qtl in each chromosome in range $1 \leq$ nqtl ≤ 2000. "qpos" QTL position along chromosome with options: 'rnd' and 'even' similar to "mpos".
h2	Narrow sense heritability. Range: $0 <$ h2 ≤ 1.
d2	Ratio of dominance variance to phenotypic variance. Range: $0 \leq$ d2 ≤ 1.
phen_var	Phenotypic variance. Range: $0 <$ x ≤ 10000.
mutr	Mutation rate. Range: $0 \leq$ mutr ≤ 0.01.
laf	*Optional* Loci (marker and qtl) allele frequencies in the first historical generation with options: • "rnd" where allele frequencies will be sampled from uniform distribution. • 0 < laf <1 where all loci allele frequencies will be equal to laf. Default: "rnd"

`sel_seq_qtl` *Optional* Select segregating qtl in the last historical generation. QTL with minor allele frequency larger than or equal to `sel_seq_qtl` will be selected. Range: $0 \leq \mathtt{x} \leq 0.4999$. Default: 0

`sel_seq_mrk` *Optional* Select segregating markers in the last historical generation. Markers with minor allele frequency larger than or equal to `sel_seq_mrk` will be selected. Range: $0 \leq \mathtt{x} \leq 0.4999$. Default: 0

`saveAt` *Optional* (`character`). Name to be used to save output files.

Details

Historical population

In order to create initial linkage disequilibrium (LD) and to establish mutation-drift equilibrium, a historical population is simulated by considering only two evolutionary forces: mutation and drift. Mutation constantly introduces new variation and genetic drift shifts the variation to fixation. Offspring are produced by random union of gametes, each from the male and female gametic pools. Population size is constant over discrete generations with equal number of males and females.

Genome

A wide range of parameters can be specified for simulating the genome, such as: number of chromosomes, markers and QTL, location of markers and QTL, mutation rate and initial allelic frequencies. This flexibility permits for a wide variety of genetic architectures to be considered. No allelic effects are simulated for markers, so they are treated as neutral. For QTL, additive allelic effects are sampled from gamma distribution with shape and scale parameters of 0.4 and 1.66, respectively. This provided an L-shaped distribution of QTL effects. To simulate dominance effects, first dominance degrees h_i are sampled from normal distribution ($N(0.5, 1)$) then absolute dominance effects are considered as $d_i = h_i.|a_i|$ where $|a_i|$ is the absolute value of the additive effect. Thus, additive and dominance effects are dependent. Next, additive and dominance effects are scaled such that user defined h^2 and d^2 are met. Trait phenotypes are simulated by adding a standard normal residual effect to the genotypic value of each individual.

One important aspect of genome simulation is to model the recombination appropriately to produce realistic level of LD, given the recent and past population structures. `make_hp` models crossover process, using a Poisson model. This is done by sampling the number of crossovers from a Poisson distribution and then the crossovers are located randomly across the chromosome. Because the input map is in centiMorgan it is straightforward to take into account the pattern of recombination hotspots and cold spots along the genome by adjusting the distances between markers. To establish mutation-drift equilibrium in the historical generations recurrent mutation model is used. The recurrent mutation model assumes that a mutation alters an allelic state to another and does not create a new allele. In the recurrent model, transition probabilities from one allelic state to another are assumed equal. Different mutation rates for simulated loci can be specified. The number of mutations is sampled from a Poisson distribution and it is assumed that mutation rates are equal for all loci.

In conclusion the main features for `make_hp` are as following:

- Multiple chromosomes with similar or different genome length in cM, each with different or similar density of markers and QTL maps, can be generated.
- Trait of interest can be controlled by additive or both additive and dominance effects.

Value

`list` with data of last generation of historical population.

$hploci Genotype (both marker (SNP) and QTL) of individuals. Coded as 11,12,21,22 where first allele is always paternal allele.

$hp_mrk Marker genotye of individuals.

$hp_qtl QTL genotye of individuals.

$freqQTL QTL allele frequency.

$freqMrk Marker allele frequency.

$linkage_map_qtl Linkage map for qtl.

$linkage_map_mrk Linkage map for marker.

$linkage_map_qtl_mrk Integrated linkage map for both marker and qtl.

$allele_effcts QTL allelic effects.

$hp_data Individuals data except their genotypes.

$trait Trait specifications.

$mut_data Mutation data.

Examples

```
# # # EXAMPLE 1 Simulation of a historical population
#for an additive trait (h2=0.3) for 10 generations.
# Two  chromosome with different parameters

genome<-data.frame(matrix(NA, nrow=2, ncol=6))
names(genome)<-c("chr","len","nmrk","mpos","nqtl","qpos")
genome$chr<-c(1,2)
genome$len<-c(100,200)
genome$nmrk<-c(100,100)
genome$mpos<-c("rnd","even")
genome$nqtl<-c(50,50)
genome$qpos<-c("even","rnd")
genome

hp<-make_hp(hpsize=100,
 ng=10,h2=0.3,phen_var=1 ,genome=genome,
 mutr=2.5e-4,saveAt="hp1")

head(hp$hp_data)
head(hp$freqQTL)
head(hp$linkage_map_qtl_mrk)


# # # EXAMPLE 2 Simulation of a historical population for a trait with both additive and
# dominance effects (h2=0.3, d2=0.1).
# All loci will have the same allele frequencies in the first generation.
# Segregating markers and qtl with MAF>0.1 will be selected in the last historical population.
```

```
genome<-data.frame(matrix(NA, nrow=3, ncol=6))
names(genome)<-c("chr","len","nmrk","mpos","nqtl","qpos")
genome$chr<-c(1,2,3)
genome$len<-c(12,8,11)
genome$nmrk<-c(140,80,73)
genome$mpos<-c("rnd","even","even")
genome$nqtl<-c(40,65,24)
genome$qpos<-rep("rnd",3)
genome

hp2<-make_hp(hpsize=100,
  ng=10,h2=0.3,d2=0.1,phen_var=1 ,genome=genome,
  mutr=2.5e-4,sel_seq_qtl=0.1,sel_seq_mrk=0.1,
  laf=0.1,saveAt="hp2")

head(hp2$hp_data)
head(hp2$freqQTL)
head(hp2$linkage_map_qtl_mrk)
```

`make_rp` *Make recent population*

Description

Creates recent population similar to function `sample_hp` with some modifications.

Usage

```
make_rp(sh_out, Male_founders, Female_founders, ng, litter_size, Selection,
  Training, saveAt, rp_output, Display)
```

Arguments

`sh_out`	(`list`) Output of function `sample_hp` or `make_rp`.
`Male_founders`	(`data.frame`) Data frame with 1 row and 3 colomns as following: Colomn 1) "number" is the number of male individuals to be selected. Colomn 2) "generation" is the generation number from which males will be selected. Colomn 3) "select" indicates the type of selection with options: • "rnd" Select individuals randomly. • "phen" Select individuals based on their phenotypes. • "tbv" Select individuals based on their true breeding value (tbv). • "gebv" Select individuals based on their genomic estimated breeding value (gebv). **Note:** If "gebv" was not selection criteria for the population from which founders are being selected, then "gebv" as selection criteria will be ignored.

Colomn 4) "value" Indicates to select high: "h" or low: "l" values. Note: This colomn is ignored if indviduals are selected randomly.

`Female_founders`

(`data.frame`) Data frame with 1 row and 3 columns containing information to select female founders. Details are similar to argument `Male_founders`.

`ng` Number of generations. Range: $1 \leq \texttt{ng} \leq 500$.

`litter_size` Litter size or the number of progeny per dam. Range: $1 \leq \texttt{x} \leq 200$.

`Selection` (`data.frame`) Data frame with 2 rows and 3 columns. First row is for the selection design of males and second row is for the selection design of females. The columns are as following:
Column 1) "size" is the number of individuals to be selected as sires/dams.
Column 2) "type" indicates the type of selection with options:

- "rnd" Select individuals randomly.
- "phen" Select individuals based on their phenotypes.
- "tbv" Select individuals based on their true breeding value (tbv).
- "gebv" Select individuals based on their genomic estimated breeding value (gebv).

Column 3) "value" Indicates to select "h" or "l" values. Note: This colomn is ignored if indviduals are selected randomly.

`Training` *Optional* (`data.frame`) Data frame with 1 row and 8 columns. The columns are as following:
Column 1) "size" is the number of individuals to be selected for training.
Column 2) "sel" *Optional* (`character`) Indicates the type of the selection of individuals for training. The possible options are:

- "rnd" Select individuals for training randomly.
- "min_rel_mrk" Select individuals for training, where genomic relationship among individuals based on marker information is minimum.
- "max_rel_mrk" Select individuals for training, where genomic relationship among individuals based on marker information is maximum.
- "min_rel_qtl" Select individuals for training, where genomic relationship among individuals based on qtl information is minimum.
- "max_rel_qtl" Select individuals for training, where genomic relationship among individuals based on qtl information is maximum.

Default: "rnd"
Column 3) "method" *Optional* (`character`) Method used for the estimation of marker effects. The possible options are:

- "BRR" Gaussian prior.
- "BayesA" scaled-t prior.
- "BL" Double-Exponential prior.
- "BayesB" two component mixture prior with a point of mass at zero and a sclaed-t slab.
- "BayesC" two component mixture prior with a point of mass at zero and a Gaussian slab.

Default: "BRR"
Column 4) "nIter" *Optional* The number of iterations. Default: 1500
Column 5) "burnIn" *Optional* The number of burn-in. Default: 500
Column 6) "thin" *Optional* The number of thinning. Default: 5
Column 7) "save" *Optional* This may include a path and a pre-fix that will be added to the name of the files that are saved as the program runs. Default:"Out_BGLR"
Column 8) "show" *Optional* (`Logical`) if TRUE the iteration history is printed. Default: `TRUE`.
Note: This argument is compulsory if `"type"` in argument `Selection` is "gebv". More details about the argument can be found in package **BGLR**.

`saveAt` *Optional* (`character`). Name to be used to save output files.

`rp_output` *Optional* (`data.frame`). Data frame to specify generations indexs and type of data to be written to output files. User can define which type of data and which generation to be written to output files. The possible options are:

"data" Individuals data except their genotypes.
"qtl" QTL genotye of individuals coded as 11,12,21,22.
"marker" Marker genotye of individuals.
"seq" Genotype (both marker (SNP) and QTL) of individuals.
"freq_qtl" QTL allele frequency.
"freq_mrk" Marker allele frequency.
Note: Both arguments `rp_output` and `saveAt` should present in the function in order to write the output files.

`Display` *Optional* (`Logical`) Display summary of the simulated generations if is not `FALSE`. Default: `TRUE`.

Details

Function `make_rp` is used to create recent population(s) similar to function `sample_hp`. The difference between these two functions is that, for function `sample_hp`, male and female founders are always from the last generation of historical population. However, in function `make_rp`, male and female founders can be selected from any generation of population created by function `sample_hp` or in the second usage, founders can be selected from any generation of population created by the function `make_rp` itself. So, basically function `make_rp` can be used multiple times to sample individuals from the desired generation of population created by function `sample_hp` or function `make_rp`. See details for function `sample_hp` and the package vignette for more clarification.

Value

`list` with all data of simulated generations.

$output (`list`) Two-level list (`$output[[x]][[y]]`) containing information about simulated generations. First index (x) indicates generation number. It should be noted that as data for base generation (0) is also stored by the function, to retrive data for a specific generation, index should be equal to generation number plus one. As an example to observe data for generation

2 index should be 3 i.e, `$output[[3]]$data`. Second index (y) that ranges from 1 to 6 contain the information as following:

- `$output[[x]]$data` Individuals data except their genotypes. Here x is the generation index
- `$output[[x]]$qtl` QTL genotye of individuals..
- `$output[[x]]$mrk` Marker genotye of individuals.
- `$output[[x]]$sequ` Genotype (both marker (SNP) and QTL) of individuals.
- `$output[[x]]$freqQTL` QTL allele frequency.
- `$output[[x]]$freqMRK` Marker allele frequency.

$summary_data Data frame with summary of simulated generations.

$linkage_map_qtl Linkage map for qtl.

$linkage_map_mrk Linkage map for marker.

$linkage_map_qtl_mrk Integrated linkage map for both marker and qtl.

$allele_effcts QTL allele effects.

$trait Trait specifications.

$genome Genome specifications.

See Also

sample_hp

Examples

```
# # # Simulation of a population where founders are from a population created by function sample_hp.

# CREATE HISTORICAL POPULATION

genome<-data.frame(matrix(NA, nrow=2, ncol=6))
names(genome)<-c("chr","len","nmrk","mpos","nqtl","qpos")
genome$chr<-c(1,2)
genome$len<-c(12,8)
genome$nmrk<-c(140,80)
genome$mpos<-c("rnd","even")
genome$nqtl<-c(25,25)
genome$qpos<-rep("rnd",2)
genome

hp<-make_hp(hpsize=100
,ng=10,h2=0.3,phen_var=1
,genome=genome,mutr=5*10**-4,sel_seq_qtl=0.05,sel_seq_mrk=0.05,laf=0.5)

# # MAKE FIRST RECENT POPULATION USING FUNCTION sample_hp

Male_founders<-data.frame(number=50,select='rnd')
Female_founders<-data.frame(number=50,select='rnd')

# Selection scheme in each generation of recent population
Selection<-data.frame(matrix(NA, nrow=2, ncol=2))
```

```
names(Selection)<-c('Number','type')
Selection$Number[1:2]<-c(50,50)
Selection$type[1:2]<-c('rnd','rnd')
Selection

RP_1<-sample_hp(hp_out=hp,Male_founders=
Male_founders,Female_founders=Female_founders,
ng=4,Selection=Selection,Training=Training,
litter_size=3,Display=TRUE)

# # MAKE SECOND RP (RP2) USING FUNCTION make_hp
  # Select founders
  # Select 30 males based on 'tbv' from generation  2 of RP1.
  # Select 40 females based on 'phen' from generation 4 of RP1.

Males<-data.frame(number=30,generation=2,select='tbv',value='h')
Females<-data.frame(number=40,generation=4,select='phen',value='l')

# Selection scheme for RP2

# Selection of 20 sires and 50 dam
# Selection criteria is "tbv" for sires and "phen" for dams

Selection<-data.frame(matrix(NA, nrow=2, ncol=3))
names(Selection)<-c('Number','type','Value')
Selection$Number[1:2]<-c(20,50)
Selection$type[1:2]<-c('tbv','phen')
Selection$Value[1:2]<-c('h','h')
Selection

# Save "data" and "qtl" for first and last generation of RP1

rp2_output<-data.frame(matrix(NA, nrow=2, ncol=2))
names(rp2_output)<-c("data","qtl")
rp2_output[,1]<-c(1,4) # Save data for generations 1 and 4
rp2_output[,2]<-c(1,4) # Save qtl genotype for generations 1 and 4
rp2_output

RP_2<-make_rp(sh_out=RP_1,Male_founders=Males,
Female_founders=Females,Selection=Selection,
ng=4,litter_size=4,saveAt='RP2',
rp_output=rp2_output)

# Some output display

RP_2$summary_data
RP_2$output[[1]]$data # Data for base Generation
RP_2$output[[2]]$freqQTL # qtl frequencies for 1st Generation
RP_2$output[[4]]$freqMRK # Marker frequencies for 3rd Generation
RP_2$linkage_map_qtl
RP_2$allele_effcts
```

`sample_hp` *Sample from historical population*

Description

Samples individuals from historical population as founders and simulates subsequent generations for a recent population based on user defined selection parameters.

Usage

```
sample_hp(hp_out, Male_founders, Female_founders, ng, litter_size, Selection,
  Training, saveAt, sh_output, Display)
```

Arguments

`hp_out` (`list`) Output of function `make_hp`.

`Male_founders` (`data.frame`) Data frame with 1 row and 3 columns as following:
Column 1) "number" is the number of male individuals to be selected from the last generation of historical population.
Column 2) "select" indicates the type of selection with options:

- "rnd" Select individuals randomly.
- "phen" Select individuals based on their phenotypes.
- "tbv" Select individuals based on their true breeding value (tbv).

Column 3) "value" Indicates to select hight: "h" or low: "l" values. Note: This Column is ignored if individuals are selected randomly.

`Female_founders` (`data.frame`) Data frame with 1 row and 3 columns as following:
Column 1) "number" is the number of female individuals to be selected from the last generation of historical population.
Column 2) "select" indicates the type of selection with options:

- "rnd" Select individuals randomly.
- "phen" Select individuals based on their phenotypes.
- "tbv" Select individuals based on their true breeding value (tbv).

Column 3) "value" Indicates to select "h" or "l" values. Note: This column is ignored if individuals are selected randomly.

`ng` Number of generations. Range: $1 \leq \texttt{ng} \leq 500$.

`litter_size` Litter size or the number of progeny per dam. Range: $1 \leq \texttt{x} \leq 200$.

`Selection` (`data.frame`) Data frame with 2 rows and 3 colomns. First row is for the selection design of males and second row is for the selection design of females. The colomns are as following:
Column 1) "size" is the number of individuals to be selected as sires/dams.
Column 2) "type" indicates the type of selection with options:

- "rnd" Select individuals randomly.
- "phen" Select individuals based on their phenotypes.
- "tbv" Select individuals based on their true breeding value (tbv).
- "gebv" Select individuals based on their genomic estimated breeding value (gebv).

Column 3) "value" Indicates to select "h" or "l" values. Note: This column is ignored if individuals are selected randomly.

`Training` *Optional* (`data.frame`) Data frame with 1 row and 8 columns. The columns are as following:
Column 1) "size" is the number of individuals to be selected for training.
Column 2) "sel" *Optional* (`character`) Indicates the type of the selection of individuals for training. The possible options are:

- "rnd" Select individuals for training randomly.
- "min_rel_mrk" Select individuals for training, where genomic relationship among individuals based on marker information is minimum.
- "max_rel_mrk" Select individuals for training, where genomic relationship among individuals based on marker information is maximum.
- "min_rel_qtl" Select individuals for training, where genomic relationship among individuals based on qtl information is minimum.
- "max_rel_qtl" Select individuals for training, where genomic relationship among individuals based on qtl information is maximum.

Default: "rnd"
Column 3) "method" *Optional* (`character`) Method used for the estimation of marker effects. The possible options are:

- "BRR" Gaussian prior.
- "BayesA" scaled-t prior.
- "BL" Double-Exponential prior.
- "BayesB" two component mixture prior with a point of mass at zero and a sclaed-t slab.
- "BayesC" two component mixture prior with a point of mass at zero and a Gaussian slab.

Default: "BRR"
Column 4) "nIter" *Optional* The number of iterations. Default:1500
Column 5) "burnIn" *Optional* The number of burn-in. Default:500
Column 6) "thin" *Optional* The number of thinning. Default:5
Column 7) "save" *Optional* This may include a path and a pre-fix that will be added to the name of the files that are saved as the program runs. Default:"Out_BGLR"
Column 8) "show" *Optional* (`Logical`) if TRUE the iteration history is printed. Default: `TRUE`.
Note: This argument is compulsory if `"type"` in argument `Selection` is "gebv". More details about the argument can be found in package **BGLR**.

`saveAt` *Optional* (`character`). Name to be used to save output files.

sh_output — *Optional* (data.frame). Data frame to specify generations indexs and type of data to be written to output files. User can define which type of data and which generation to be written to output files. The possible options are:

"data" Individuals data except their genotypes.
"qtl" QTL genotye of individuals coded as 11,12,21,22.
"marker" Marker genotye of individuals.
"seq" Genotype (both marker (SNP) and QTL) of individuals.
"freq_qtl" QTL allele frequency.
"freq_mrk" Marker allele frequency.
Note: Both arguments sh_output and saveAt should present in the function in order to write the output files.

Display — *Optional* (Logical) Display summary of the simulated generations if is not FALSE. Default: TRUE.

Details

Function sample_hp is used to create recent population(s). This function can be used multiple times to sample individuals from the historical population created by function make_hp. For the start up of the recent population, male and female founders come from the last generation of historical population and can be selected based on one of the options described in argument Male_founders or Female_founders. For the subsequent generations individuals can be selected based on genomic estimated breeding value "gebv". To do so, argument Training should present in the model to estimate the marker effects. Selected individuals for training are always from a generation preceding the target generation. As an example, for the calculation of GEBV for the individuals in generation 4, selected individuals from generation 3 are used for training. In order to select individuals for training, user can control type of selection by argument Training. For the options "min_rel_mrk" and "max_rel_mrk", genomic relationship matrix is constructed as following:

$$G = ZZ'/2\sum_{j=1}^{m} p_j(1-p_j)$$

where $Z = M - P$. Here M is an allele-sharing matrix with m columns (m = number of markers) and n rows (n = number of genotyped individuals), and P is a matrix containing the frequency of the second allele (p_j), expressed as $2p_j$. M_{ij} is 0 if the genotype of individual i for SNP j is homozygous 11, is 1 if heterozygous, or 2 if the genotype is homozygous 22. Frequencies are the observed allele frequency of each SNP. After constructing genomic relationship matrix, individuals are sorted based on their genomic relationship. User can define whether to select individuals with low relationship ("min_rel_mrk") or high relationship ("max_rel_mrk") among each other for training. As an example if option "min_rel_mrk" is considered, then selected individuals for training have the lowest relationship with each other compared to the whole population they belong.

Genomic relationship matrix for the options "min_rel_qtl" and "max_rel_qtl" are constracted as the same procedure described above except that qtl genotype of individual rather than markers are used to calculate genomic relationships among individuals.

The main features for sample_hp are as following:

- Selection criteria can differ between males and females.

- Different models can be used for the estimation of marker effects.
- Multiple options for constructing the reference population for training.
- Dynamic control of output files to be saved.

Value

`list` with all data of simulated generations.

$output (`list`) Two-level list (`$output[[]][[]]`) containing information about simulated generations. First index (x) indicates generation number. It should be noted that as data for base generation (0) is also stored by the function, to retrive data for a specific generation, index should be equal to generation number plus one. As an example to observe data for generation 2 index should be 3 i.e, `$output[[3]]$data`. Second index (y) that ranges from 1 to 6 contain the information as following:

- `$output[[x]]$data` Individuals data except their genotypes. Here x is the generation index.
- `$output[[x]]$qtl` QTL genotye of individuals..
- `$output[[x]]$mrk` Marker genotye of individuals.
- `$output[[x]]$sequ` Genotype (both marker (SNP) and QTL) of individuals.
- `$output[[x]]$freqQTL` QTL allele frequency.
- `$output[[x]]$freqMRK` Marker allele frequency.

$summary_data Data frame with summary of simulated generations.

$linkage_map_qtl Linkage map for qtl.

$linkage_map_mrk Linkage map for marker.

$linkage_map_qtl_mrk Integrated linkage map for both marker and qtl.

$allele_effcts QTL allele effects.

$trait Trait specifications.

$genome Genome specifications.

See Also

make_hp

Examples

```
# # # Simulation of a recent population following a historical population.

# CREATE HISTORICAL POPULATION

genome<-data.frame(matrix(NA, nrow=2, ncol=6))
names(genome)<-c("chr","len","nmrk","mpos","nqtl","qpos")
genome$chr<-c(1,2)
genome$len<-c(50,60)
genome$nmrk<-c(130,75)
genome$mpos<-c("rnd","rnd")
genome$nqtl<-c(30,30)
```

```
genome$qpos<-rep("even",2)
genome

hp<-make_hp(hpsize=100
,ng=10,h2=0.3,d2=0.15,phen_var=1
,genome=genome,mutr=5*10**-4,sel_seq_qtl=0.05,sel_seq_mrk=0.05,laf=0.5)

# # MAKE FIRST RECENT POPULATION USING FUNCTION sample_hp

Male_founders<-data.frame(number=50,select="rnd")
Female_founders<-data.frame(number=50,select="rnd")

# Selection scheme in each generation of recent population

Selection<-data.frame(matrix(NA, nrow=2, ncol=2))
names(Selection)<-c("Number","type")
Selection$Number[1:2]<-c(50,50)
Selection$type[1:2]<-c("rnd","rnd")
Selection

# Save "data" and "freq_mrk" for first and last generation of RP

my_files<-data.frame(matrix(NA, nrow=2, ncol=2))
names(my_files)<-c("data","marker")
my_files[,1]<-c(1,4) # Save data for generations 1 and 4
my_files[,2]<-c(1,4) # Save freq_mrk for generations 1 and 4
my_files

RP<-sample_hp(hp_out=hp,Male_founders=
Male_founders,Female_founders=Female_founders,
ng=4,Selection=Selection,litter_size=3,saveAt="my_RP",sh_output=my_files,Display=TRUE)

# Some results

RP$summary_data
RP$output[[2]]$data      # Data for 1st Generation
RP$output[[4]]$freqMRK   # Marker frequencies for 3rd Generation
RP$linkage_map_qtl
RP$allele_effcts
```

---

xbreed *Create crossbred population*

Description

This function can be used for crossing between populations. Different crossbreeding schemes such as two-way, three-way and four-way crossbreeding schemes can be simulated.

Usage

```
xbreed(pop1, pop2, founder_pop1, founder_pop2, founder_cross, Selection_pop1,
  Selection_pop2, Cross_design, ng, litter_size, train_type, train_pop1,
  train_pop2, train_cross, saveAt, output_pop1, output_pop2, output_cross,
  Display)
```

Arguments

pop1	(list) Output of function make_rp, sample_hp or xbreed.
pop2	(list) Output of function make_rp, sample_hp or xbreed.
founder_pop1	(data.frame) Data frame with 2 rows and 3 columns. First row is for the selection of male founders and second row is for the selection of female founders from pop1. The columns are as following: Colomn 1) "size" is the number of individuals to be selected. Column 2) "generation" is the generation number from which males/females will be selected. Column 3) "select" indicates the type of selection with options: • "rnd" Select individuals randomly. • "phen" Select individuals based on their phenotypes. • "tbv" Select individuals based on their true breeding value (tbv). • "gebv" Select individuals based on their genomic estimated breeding value (gebv). Column 4) "value" Indicates to select high: "h" or low: "l" values. Note: This colomn is ignored if individuals are selected randomly.
founder_pop2	(data.frame) Data frame with 2 rows and 3 columns. First row is for the selection of male founders and second row is for the selection of female founders from pop2. Details are similar to argument founder_pop1.
founder_cross	(data.frame) Data frame with 2 rows and 4 columns that contains information about creating base generation of crossbreds. First row is for the selection of male founders and second row is for the selection of female founders in order to create crossbred population. The columns are as following: Column 1) "pop" Indication of the population name from which males/females for crossbreeding will be selected. The two possible options are: • "pop1" Select individuals are from pop1. • "pop2" Select individuals are from pop2. Column 2) "size" is the number of individuals to be selected. Column 3) "select" indicates the type of selection with options: • "rnd" Select individuals randomly. • "phen" Select individuals based on their phenotypes. • "tbv" Select individuals based on their true breeding value (tbv). • "gebv" Select individuals based on their genomic estimated breeding value (gebv).

Column 4) "value" Indicates to select high: "h" or low: "l" values. This column is ignored if individuals are selected randomly.
Note: After selecting founders for pop1 and pop2, by arguments `founder_pop1` and `founder_pop2`, user can select individuals from these founders as parents of crossbreds for the base generation of crossbreds.

`Selection_pop1` (`data.frame`) Selection design for pop1 (Breed 1). Data frame with 2 rows and 3 columns. First row is for the selection design of males and second row is for the selection design of females. The columns are as following:
Column 1) "Number" is the number of individuals to be selected as sires/dams.
Column 2) "type" indicates the type of selection with options:

- "rnd" Select individuals randomly.
- "phen" Select individuals based on their phenotypes.
- "tbv" Select individuals based on their true breeding value (tbv).
- "tbvc" Select individuals based on true breeding value for crossbred performance(tbvc).
- "gebv" Select individuals based on their genomic estimated breeding value (gebv).
- "gebvc" Select individuals based on genomic estimated breeding value for crossbred performance (gebvc).

Column 3) "value" Indicates to select high: "h" or low: "l" values. Note: This colomn is ignored if individuals are selected randomly.

`Selection_pop2` (`data.frame`) Selection design for pop2 (Breed 2). Details are similar to argument `Selection_pop1`.

`Cross_design` (`data.frame`) Data frame containing information on how to select individuals from pop1 and pop2 as parents of crossbreds over generations. This argument is a data frame with 2 rows and 4 columns. First row is for the selection of males as sires of crossbreds and second row is for the selection of females as dams of crossbreds. The columns are as following:
Column 1) "pop" Indication of the population name from which males/females for crossing will be selected. The two possible options are:

- "pop1" Selected individuals are from pop1.
- "pop2" Selected individuals are from pop2.

As an example if row 1 and column 1 of argument `Cross_design` is equal to "pop2", this means that sires of crossbred are from pop2.
Column 2) "size" is the number of individuals to be selected as sires/dams.
Column 3) "select" indicates the type of selection with options:

- "rnd" Select individuals randomly.
- "phen" Select individuals based on their phenotypes.
- "tbv" Select individuals based on their true breeding value (tbv).
- "tbvc" Select individuals based on true breeding value for crossbred performance(tbvc).
- "gebv" Select individuals based on their genomic estimated breeding value (gebv).

- "gebvc" Select individuals based on genomic estimated breeding value for crossbred performance (gebvc).

Column 4) "value" Indicates to select high: "h" or low: "l" values. Note: This column is ignored if individuals are selected randomly.

`ng` Number of generations. Range: $1 \leq$ `ng` ≤ 200.

`litter_size` Litter size or the number of progeny per dam. Range: $1 \leq$ `x` ≤ 200.

`train_type` *Optional* (`character`) Type of training for the estimation of marker effects. The two possible options are:

- "purebred" Training will be on purebreds. This means that training for each population is done separately. So, there will be two reference population; one for pop1 (Breed 1) and one for pop2 (Breed 2).
- "crossbred" Training will be on crossbreds. So, estimated marker effects will be the same for both pop1 and pop2.

Note: If selection criteria for any population is defined as gebv/gebvc, argument `train_type` should be defined.

`train_pop1` *Optional* (`data.frame`) Data frame with 1 row and 8 columns. The columns are as following:
Column 1) "size" is the number of individuals to be selected for training.
Column 2) "sel" *Optional* (`character`) Indicates the type of the selection of individuals for training. The possible options are:

- "rnd" Select individuals for training randomly.
- "min_rel_mrk" Select individuals for training, where genomic relationship among individuals based on marker information is minimum.
- "max_rel_mrk" Select individuals for training, where genomic relationship among individuals based on marker information is maximum.
- "min_rel_qtl" Select individuals for training, where genomic relationship among individuals based on qtl information is minimum.
- "max_rel_qtl" Select individuals for training, where genomic relationship among individuals based on qtl information is maximum.

Default: "rnd"
Column 3) "method" *Optional* (`character`) Method used for the estimation of marker effects. The possible options are:

- "BRR" Gaussian prior.
- "BayesA" scaled-t prior.
- "BL" Double-Exponential prior.
- "BayesB" two component mixture prior with a point of mass at zero and a scaled-t slab.
- "BayesC" two component mixture prior with a point of mass at zero and a Gaussian slab.

Default: "BRR"
Column 4) "nIter" *Optional* The number of iterations. Default: 1500
Column 5) "burnIn" *Optional* The number of burn-in. Default: 500

Column 6) "thin" *Optional* The number of thinning. Default: 5
Column 7) "save" *Optional* This may include a path and a pre-fix that will be added to the name of the files that are saved as the program runs. Default:"Out_BGLR"
Column 8) "show" *Optional* (`Logical`) if TRUE the iteration history is printed. Default: `TRUE`.
Note: This argument is compulsory if argument `train_type` is "purbred".
More details about the argument can be found in package **BGLR**.

`train_pop2` *Optional* (`data.frame`) Data frame with 1 row and 8 columns similar to argument `train_pop1`.
Note: This argument is compulsory if argument `train_type` is "purbred".

`train_cross` *Optional* (`data.frame`) Data frame with 1 row and 8 columns similar to argument `train_pop1`.
Note: This argument is compulsory if argument `train_type` is "crossbred".

`saveAt` *Optional* (`character`). Name to be used to save output files.

`output_pop1` *Optional* (`data.frame`). Data frame to specify generation indexs and type of data to be written to output files for "pop1". User can define which type of data and which generations to be written to output files. The possible options are:

"data" Individuals data except their genotypes.
"qtl" QTL genotye of individuals coded as 11,12,21,22.
"marker" Marker genotye of individuals.
"seq" Genotype (both marker (SNP) and QTL) of individuals.
"freq_qtl" QTL allele frequency.
"freq_mrk" Marker allele frequency.
Note: Both arguments `output_pop1` and `saveAt` should present in the function in order to write the output files for "pop1".

`output_pop2` *Optional* (`data.frame`). Data frame to specify generations indexs and type of data to be written to output files for pop2. Details are similar to argument `output_pop1`.
Note: Both arguments `output_pop2` and `saveAt` should present in the function in order to write the output files for "pop2".

`output_cross` *Optional* (`data.frame`). Data frame to specify generations indexs and type of data to be written to output files for crossbred population. Details are similar to argument `output_pop1`.
Note: Both arguments `output_cross` and `saveAt` should present in the function in order to write the output files for crossbred population.

`Display` *Optional* (`Logical`) Display summary of the simulated generations if is not `FALSE`. Default: `TRUE`.

Details

Function `xbreed` is used for crossing between populations. These populations can be the ones created by functions `sample_hp` and `make_rp`. Also, if user would like to have multi-way crossbreeding schemes such as three-way crossbreeding, then function `xbreed` can be used to get the output of itself as input data in order to create the multi-way crossbreed populations. Simulations of two-way and multi-way crossbreeding schemes are presented in the package vignette.

Value

`list` with all data of simulated populations.

$pop1 (`list`) Two-level list (`$pop1[[]][[]]`) containing information about simulated generations. First index (x) indicates generation number. It should be noted that as data for base generation (0) is also stored by the function, to retrive data for a specific generation, index should be equal to generation number plus one. As an example to observe data for generation 2 index should be 3 i.e, `$pop1[[3]]$data`. Second index (y) that ranges from 1 to 6 contain the information as following:

- `$pop1[[x]]$data` Individuals data except their genotypes. Here x is the generation index.
- `$pop1[[x]]$qtl` QTL genotye of individuals.
- `$pop1[[x]]$mrk` Marker genotye of individuals.
- `$pop1[[x]]$sequ` Genotype (both marker (SNP) and QTL) of individuals.
- `$pop1[[x]]$freqQTL` QTL allele frequency.
- `$pop1[[x]]$freqMRK` Marker allele frequency.

$pop2 Similar to $pop1

$cross (`list`) Two-level list (`$cross$output[[x]][[y]]`) containing information about simulated crossbred generations. Details are similar to `$pop1` such as:

- `$cross$output[[x]]$data` Crossbred individuals data except their genotypes. Here x is the generation index.
- `$cross$output[[x]]$qtl` QTL genotye of crossbred individuals.
- `$cross$output[[x]]$mrk` Marker genotye of individuals.
- `$cross$output[[x]]$sequ` Genotype (both marker (SNP) and QTL) of individuals.
- `$cross$output[[x]]$freqQTL` QTL allele frequency.
- `$cross$output[[x]]$freqMRK` Marker allele frequency.

$cross$summary_data_cross Data frame with summary of simulated crossbreds.

$summary_data_pop1 Data frame with summary of simulated generations for pop1 (Breed 1).

$summary_data_pop2 Data frame with summary of simulated generations for pop2 (Breed 2).

$linkage_map_qtl Linkage map for qtl.

$linkage_map_mrk Linkage map for marker.

$linkage_map_qtl_mrk Integrated linkage map for both marker and qtl.

$allele_effcts QTL allele effects.

$trait Trait specifications.

$genome Genome specifications.

References

Esfandyari H., A.C Sorensen and P. Bijma. 2015. A crossbred reference population can improve the response to genomic selection for crossbred performance. *Genetics Selection Evolution* **47**: 76

Esfandyari H., A.C Sorensen and P. Bijma. 2015. Maximizing crossbred performance through purebred genomic selection. *Genetics Selection Evolution* **47**: 16

See Also

make_rp, sample_hp

Examples

```
## Not run:
# # Simulation of a two-way crossbreeding program.
# The crossbreeding scheme in this example involves three steps:
#Step 1: Historical population is created.
#Step 2: Two recent populations named as Breed A and B
# are created by sampling individuals from historical population.
#Step 3: Breed A and B are crossed.

#--------------------------------------
# # STEP 1: CREATE HISTORICAL POPULATION

# Genome consisted of 3 chromosomes
genome<-data.frame(matrix(NA, nrow=3, ncol=6))
names(genome)<-c("chr","len","nmrk","mpos","nqtl","qpos")
genome$chr<-c(1:3)
genome$len<-rep(100,3)
genome$nmrk<-rep(100,3)
genome$mpos<-rep('rnd',3)
genome$nqtl<-rep(25,3)
genome$qpos<-rep('rnd',3)
genome

historical<-make_hp(hpsize=300
,ng=10,h2=0.25,d2=0.10,phen_var=1
,genome=genome,mutr=5*10**-4,laf=0.5)

# # STEP 2: MAKE BREED A AND B
# BREED A
Breed_A_Male_fndrs<-data.frame(number=50,select='rnd')
Breed_A_Female_fndrs<-data.frame(number=50,select='rnd')

# Selection and matings in Breed A
# Selection of 50 sires and 50 dam
# Selection criteria is "rnd" for both sires and dams
 Selection<-data.frame(matrix(NA, nrow=2, ncol=2))
 names(Selection)<-c("Number","type")
 Selection$Number[1:2]<-c(50,50)
 Selection$type[1:2]<-c("rnd","rnd")
 Selection
```

```
Breed_A<-sample_hp(hp_out=historical,Male_founders=
Breed_A_Male_fndrs,Female_founders=Breed_A_Female_fndrs,
ng=5,Selection=Selection,
litter_size=3,Display=TRUE)

# BREED B
Breed_B_Male_fndrs<-data.frame(number=50,select="rnd")
Breed_B_Female_fndrs<-data.frame(number=50,select="rnd")

# Selection and matings in Breed B
# Selection of 50 sires and 50 dam
# Selection criteria is "phen" for both sires and dams

Selection<-data.frame(matrix(NA, nrow=2, ncol=3))
names(Selection)<-c("Number","type","Value")
Selection$Number[1:2]<-c(50,50)
Selection$type[1:2]<-c("phen","phen")
Selection$Value[1:2]<-c("h","h")
Selection

Breed_B<-sample_hp(hp_out=historical,Male_founders=
Breed_B_Male_fndrs,Female_founders=Breed_B_Female_fndrs,
ng=5,Selection=Selection,
litter_size=3,Display=TRUE)

# # STEP 3: CROSSING BETWEEN BREED A AND B

  # Selection of founders in crossbreeding for Breed A
# Selection of 50 sires and 50 dams
        # from last generation of pop in step 2.
# Selection criteria is "rnd" for both sires and dams
founder_pop1<-data.frame(matrix(NA, nrow=2, ncol=3))
names(founder_pop1)<-c("size","generation","select")
founder_pop1[1,]<-c(50,5,"rnd")
founder_pop1[2,]<-c(50,5,"rnd")
founder_pop1

# Selection of founders in crossbreeding for Breed B
# Selection of 40 sires and 40 dams
# Selection criteria is "phen" for sires
# Selection criteria is "rnd" for dams

founder_pop2<-data.frame(matrix(NA, nrow=2, ncol=4))
names(founder_pop2)<-c("size","generation","select","value")
founder_pop2[1,]<-c(40,5,"phen","h")
founder_pop2[2,]<-c(40,5,"rnd","h") # "h" will be ignored as SC is "rnd"
founder_pop2

# Selection of animals from founder_pop1 and founder_pop2 to be crossed
founder_cross<-data.frame(matrix(NA, nrow=2, ncol=4))
names(founder_cross)<-c("pop","size","select","value")
founder_cross[1,]<-c("pop1",35,"tbv","h") # Select males from Breed A
```

```
founder_cross[2,]<-c("pop2",40,"phen","h")  # Select females from Breed B
founder_cross

# Selection scheme in Breed A to produce purebred replacement animals
Selection_pop1<-data.frame(matrix(NA, nrow=2, ncol=3))
names(Selection_pop1)<-c("Number","type","Value")
Selection_pop1$Number[1:2]<-c(70,70)
Selection_pop1$type[1:2]<-c("tbv","tbv")
Selection_pop1$Value[1:2]<-c("h","h")
Selection_pop1

# Selection scheme in Breed B to produce purebred replacement animals
Selection_pop2<-data.frame(matrix(NA, nrow=2, ncol=3))
names(Selection_pop2)<-c("Number","type","Value")
Selection_pop2$Number[1:2]<-c(40,40)
Selection_pop2$type[1:2]<-c("phen","phen")
Selection_pop2$Value[1:2]<-c("h","h")
Selection_pop2

# Selection scheme for crossing between A and B
Cross_design<-data.frame(matrix(NA, nrow=2, ncol=4))
names(Cross_design)<-c("pop","size","select","value")
Cross_design[1,]<-c("pop1",50,"phen","h")
Cross_design[2,]<-c("pop2",100,"phen","h")
Cross_design

# Save data for crossbred AB
output_cross<-data.frame(matrix(NA, nrow=1, ncol=1))
output_cross[,1]<-c(1)
output_cross

cross_AB<-xbreed(pop1=Breed_A,pop2=Breed_B,founder_pop1=
founder_pop1,founder_pop2=founder_pop2,
founder_cross=founder_cross,
Selection_pop1=Selection_pop1,Selection_pop2=Selection_pop2,
Cross_design=Cross_design,ng=2,litter_size=4,
saveAt="cross_pop",output_cross=output_cross,Display=TRUE)

## End(Not run)
```

Index

calc_LD, 2

make_hp, 4, *12*, *14, 15*
make_rp, *7*, 7, *17*, *22*

sample_hp, *7*, *9, 10*, 12, *17*, *22*

xbreed, 16, *17*