

Package ‘BitBreedingSim’

December 17, 2024

Type Package

Title Fast Breeding Simulation

Version 0.1.0

Author Minoru Inamori

Maintainer Minoru Inamori <inamori@ut-biomet.org>

Description Use bit operations to speed up breeding simulations.

License MIT License

Imports Rcpp (>= 1.0.5)

LinkingTo Rcpp

RoxygenNote 7.3.2

Encoding UTF-8

Suggests roxygen2

Roxygen list(markdown = TRUE)

Contents

addTraitA	2
addTraitAD	2
createBaseInfo	3
createOrigins	5
cross	5
getGenotypes	6
getInfo	7
getMap	7
getPhasedGenotypes	8
getPhasedIntGenotypes	8
getPhenotypes	9
getPopInfo	9
getPopNames	10
getTrait	10
joinPops	11
selectPop	11
writeVCF	12

Index**13**

addTraitA	<i>Add Trait with Additive Effects to BaseInfo</i>
-----------	--

Description

This function adds a trait with additive effects to the BaseInfo object.

Usage

```
addTraitA(info, name, mean, h2, sd = NULL, a = NULL, loci = NULL, num_loci = 1)
```

Arguments

info	External pointer to BaseInfo object
name	Name of the trait
mean	Phenotype mean
h2	Heritability
sd	Optional. Phenotype standard deviation
a	Optional. Numeric vector of additive effects
loci	Optional. List of loci in the form of a data frame with two columns: 'chrom' and 'marker'. For example: <pre>chrom <- c(3, 5, 10) marker <- c(1, 2, 1000) loci <- data.frame(chrom, marker)</pre>
num_loci	Optional. Number of loci (default is 1)

addTraitAD	<i>Add Trait with Additive and Dominance Effects to BaseInfo</i>
------------	--

Description

This function adds a trait with additive and dominance effects to the BaseInfo object.

Usage

```
addTraitAD(
  info,
  name,
  mean,
  sd = NULL,
  h2 = NULL,
  H2 = NULL,
  a = NULL,
  d = NULL,
  loci = NULL,
  num_loci = 1
)
```

Arguments

info	External pointer to BaseInfo object
name	Name of the trait
mean	Phenotype mean
sd	Optional. Phenotype standard deviation
h2	Optional. Narrow-sense heritability (proportion of variance due to additive genetic effects)
H2	Optional. Broad-sense heritability (proportion of variance due to all genetic effects, can be NULL)
a	Optional. Numeric vector of additive effects
d	Optional. Numeric vector of dominance effects
loci	Optional. List of loci in the form of a data frame with two columns: 'chrom' and 'marker'. For example: <pre>chrom <- c(3, 5, 10) marker <- c(1, 2, 1000) loci <- data.frame(chrom, marker)</pre>
num_loci	Optional. Number of loci (default is 1)

createBaseInfo

Create a BaseInfo object

Description

This function creates a BaseInfo object. If the seed is set to -1, a random seed is generated, resulting in different outcomes each time the function is called. If a specific seed is provided, the random number generation will be based on that seed, ensuring reproducible results.

Usage

```
createBaseInfo(
  chrom_maps = NULL,
  num_chroms = 10,
  num_markers = 1000,
  cM = 100,
  bp = 1e+06,
  seed = -1
)
```

Arguments

chrom_maps	A list of data.frames, each representing a chromosome map. Each data.frame should have two columns: 'cM' for centiMorgans and 'position' for base pair positions. The list should be named, with each name corresponding to a chromosome identifier (e.g., "chr1", "chr2", etc.). If chrom_maps is provided, the parameters num_chroms, num_markers, cM, and bp are ignored.
num_chroms	Optional. An integer. Number of chromosomes. Ignored if chrom_maps is provided. Default is 10.
num_markers	Optional. An integer. Number of markers per chromosome. Ignored if chrom_maps is provided. Default is 1000.
cM	A numeric. Optional. Length of each chromosome in centiMorgans. Ignored if chrom_maps is provided. Default is 100.
bp	An integer. Optional. Length of each chromosome in base pairs. Ignored if chrom_maps is provided. Default is 1000000.
seed	An integer. Optional. A seed for random number generation. Default is -1, which generates a random seed.

Value

An external pointer to a BaseInfo object.

Examples

```
# Create a BaseInfo object with a random seed
base_info_random <- createBaseInfo()
getInfo(base_info_random)

# Create a BaseInfo object with a specific seed for reproducible results
base_info_reproducible <- createBaseInfo(seed = 123)
getInfo(base_info_reproducible)

# Create a chromosome map with 100 cM and 1 Mbp, containing 1000 markers
f <- function(x) { (x^3 / (1 + x^2) + 8/5) * 500 / 16 }
cM <- sapply(1:1000, function(i) f(i/250 - 2))
position <- sapply(1:1000, function(i) i * 1000)
chrom_map <- data.frame(cM, position)
chrom_maps <- replicate(10, chrom_map, simplify = FALSE)
```

```
names(chrom_maps) <- paste0("chr", 1:10)
info <- createBaseInfo(chrom_maps, seed = 123)
getInfo(info)
```

createOrigins	<i>Create origins for a Population object</i>
---------------	---

Description

Create origins for a Population object

Usage

```
createOrigins(num_inds, info, name_base)
```

Arguments

num_inds	An integer. The number of individuals.
info	An external pointer to a BaseInfo object.
name_base	A string. The base name for individuals.

Value

An external pointer to a Population object.

cross	<i>Cross two Population</i>
-------	-----------------------------

Description

Cross two Population

Usage

```
cross(num_inds, mothers, fathers, name_base, num_threads = 0)
```

Arguments

num_inds	An integer. The number of individuals.
mothers	An external pointer to a Population object.
fathers	An external pointer to a Population object.
name_base	A string. The base name for individuals.
num_threads	Optional. An integer. The number of threads to be used. If not specified, the function will use the maximum number of available threads.

Value

An external pointer to a Population object.

Examples

```
# Assuming 'mothers' and 'fathers' are valid Population objects
new_population <- cross(100, mothers, fathers, "prog_")
summary(new_population)
```

getGenotypes

Get genotypes from a Population object

Description

This function retrieves the genotypes from a given Population object. The genotypes are represented in a matrix format where rows correspond to samples and columns correspond to markers.

Usage

```
getGenotypes(pop)
```

Arguments

pop An external pointer to a Population object.

Details

Genotype encoding:

- 0/0 is encoded as -1
- 0/1 is encoded as 0
- 1/1 is encoded as 1

Value

A matrix of genotypes where rows are samples and columns are markers.

getInfo	<i>Get the values of a BaseInfo object</i>
---------	--

Description

Get the values of a BaseInfo object

Usage

```
getInfo(info)
```

Arguments

info An external pointer to a BaseInfo object

Value

The list of the values of a BaseInfo object

getMap	<i>Retrieve Genetic Map Information</i>
--------	---

Description

This function retrieves the genetic map information from a BaseInfo object.

Usage

```
getMap(info)
```

Arguments

info An object of class BaseInfo. This object contains the genetic map information.

Value

A list of data frames, each representing a chromosome. Each data frame contains two columns:

- cM: The centiMorgan positions of the markers.
- position: The base pair positions of the markers.

Examples

```
## Not run:  
# Assuming `info` is a valid BaseInfo object  
map <- getMap(info)  
print(map)  
  
## End(Not run)
```

getPhasedGenotypes	<i>Get phased genotypes from a Population object</i>
--------------------	--

Description

This function retrieves the genotypes from a given Population object. The genotypes are represented in a matrix format where rows correspond to markers and columns correspond to samples.

Usage

```
getPhasedGenotypes(pop)
```

Arguments

pop	An external pointer to a Population object.
-----	---

Details

Genotype is 0|0, 0|1, 1|0, or 1|1

Value

A matrix of genotypes where rows are samples and columns are markers.

getPhasedIntGenotypes	<i>Get phased integer genotypes from a Population object</i>
-----------------------	--

Description

This function retrieves the genotypes from a given Population object. The genotypes are represented in a matrix format where rows correspond to samples and columns correspond to markers. Each sample has two rows: the first row represents the maternal allele and the second row represents the paternal allele.

Usage

```
getPhasedIntGenotypes(pop)
```

Arguments

pop	An external pointer to a Population object.
-----	---

Details

Genotype is represented as integers: 0 or 1

Value

A matrix of genotypes where rows are samples and columns are markers. Each sample has two rows: the first row is the maternal allele and the second row is the paternal allele.

getPhenotypes	<i>Get phenotypes from a Population object</i>
---------------	--

Description

Get phenotypes from a Population object

Usage

```
getPhenotypes(pop, i)
```

Arguments

pop	An external pointer to a Population object.
i	An integer index representing the trait for which phenotypes are to be retrieved. The index should be between 1 and the total number of traits available in the Population object.

Value

A vector of phenotypes.

Examples

```
# Assuming 'pop' is a valid Population object and trait index 1 is valid
phenotypes <- getPhenotypes(pop, 1)
print(phenotypes)
```

getPopInfo	<i>Get information for a Population object</i>
------------	--

Description

Get information for a Population object

Usage

```
getPopInfo(pop)
```

Arguments

pop	An external pointer to a BaseInfo object.
-----	---

Value

A list containing the number of individuals and the number of chromosomes in the population.

getPopNames	<i>Get name data from a Population object</i>
-------------	---

Description

Get name data from a Population object

Usage

```
getPopNames(pop)
```

Arguments

pop	An external pointer to a Population object.
-----	---

Value

A data.frame containing the names, maternal names, and paternal names from the Population object.

Examples

```
# Assuming 'pop' is a valid Population object
name_data <- getPopNames(pop)
print(name_data)
```

getTrait	<i>Get a trait from a BaseInfo object</i>
----------	---

Description

Get a trait from a BaseInfo object

Usage

```
getTrait(info, i)
```

Arguments

info	An external pointer to a BaseInfo object.
i	An integer. The index of the trait to retrieve.

Value

The trait at the specified index.

joinPops	<i>Join multiple Population objects</i>
----------	---

Description

Join multiple Population objects

Usage

```
joinPops(...)
```

Arguments

... External pointers to Population objects.

Value

An external pointer to a new Population object containing the combined individuals.

Examples

```
# Assuming 'pop1', 'pop2', and 'pop3' are valid Population objects
combined_pop <- joinPops(pop1, pop2, pop3)
print(combined_pop)
```

selectPop	<i>Select individuals from a Population object</i>
-----------	--

Description

Select individuals from a Population object

Usage

```
selectPop(pop, indices_R)
```

Arguments

pop An external pointer to a Population object.
indices A vector of integer indices representing the individuals to be selected.

Value

An external pointer to a new Population object containing the selected individuals.

Examples

```
# Assuming 'pop' is a valid Population object and indices are valid
selected_pop <- selectPop(pop, c(1, 2, 3))
print(selected_pop)
```

`writeVCF`*Write Population to VCF file*

Description

This function writes a Population object to a VCF file.

Usage

```
writeVCF(pop, filename)
```

Arguments

<code>pop</code>	An external pointer to a Population object.
<code>filename</code>	A string specifying the path to the output VCF file.

Examples

```
# Assuming 'pop' is a valid Population object
writeVCF(pop, "output.vcf")
```

Index

addTraitA, [2](#)
addTraitAD, [2](#)

createBaseInfo, [3](#)
createOrigins, [5](#)
cross, [5](#)

getGenotypes, [6](#)
getInfo, [7](#)
getMap, [7](#)
getPhasedGenotypes, [8](#)
getPhasedIntGenotypes, [8](#)
getPhenotypes, [9](#)
getPopInfo, [9](#)
getPopNames, [10](#)
getTrait, [10](#)

joinPops, [11](#)

selectPop, [11](#)

writeVCF, [12](#)