

# Package: simer (via r-universe)

September 19, 2024

**Title** Data Simulation for Life Science and Breeding

**Version** 0.9.0.5

**Date** 2024-9-18

**Description** Data simulator including genotype, phenotype, pedigree, selection and reproduction in R. It simulates most of reproduction process of animals or plants and provides data for GS (Genomic Selection), GWAS (Genome-Wide Association Study), and Breeding. For ADI model, please see Kao C and Zeng Z (2002) <[doi:10.1093/genetics/160.3.1243](https://doi.org/10.1093/genetics/160.3.1243)>. For build.cov, please see B. D. Ripley (1987) <ISBN:9780470009604>.

**License** Apache License 2.0

**URL** <https://github.com/xiaolei-lab/SIMER>

**BugReports** <https://github.com/xiaolei-lab/SIMER/issues>

**Imports** utils, stats, Matrix, methods, MASS, Rcpp, jsonlite

**LinkingTo** Rcpp, RcppArmadillo, RcppProgress, BH, bigmemory

**Depends** R (>= 3.5.0), bigmemory

**Suggests** knitr, igraph

**RoxygenNote** 7.2.3

**Encoding** UTF-8

**NeedsCompilation** yes

**Maintainer** Xiaolei Liu <x11198708@gmail.com>

**Author** Dong Yin [aut], Xuanning Zhang [aut], Lilin Yin [aut], Haohao Zhang [aut], Zhenshuang Tang [aut], Jingya Xu [aut], Xiaohui Yuan [aut], Xiang Zhou [aut], Xinyun Li [aut], Shuhong Zhao [aut], Xiaolei Liu [cre, aut, cph]

**Repository** CRAN

**Date/Publication** 2024-09-18 02:30:03 UTC

## Contents

annotation	3
build.cov	4
cal.eff	5
checkEnv	6
generate.map	7
generate.pop	8
geno.cvt1	9
geno.cvt2	9
genotype	10
getfam	11
GxG.network	12
IndPerGen	13
logging.initialize	14
logging.log	14
logging.print	16
mate	17
mate.2waycro	18
mate.3waycro	19
mate.4waycro	20
mate.assort	22
mate.backcro	23
mate.clone	24
mate.dh	26
mate.disassort	27
mate.randexself	28
mate.randmate	30
mate.selfpol	31
mate.userped	32
param.annot	34
param.geno	35
param.global	36
param.pheno	37
param.reprod	38
param.sel	39
param.simer	40
phenotype	41
pop.geno	43
pop.map	43
remove_bigmatrix	44
reproduces	45
selects	46
simer	47
simer.Data	48
simer.Data.Bfile2MVP	49
simer.Data.cHIBLUP	50
simer.Data.Env	52

simer.Data.Geno . . . . . 53  
 simer.Data.Impute . . . . . 55  
 simer.Data.Json . . . . . 56  
 simer.Data.Kin . . . . . 57  
 simer.Data.Map . . . . . 58  
 simer.Data.MVP2Bfile . . . . . 59  
 simer.Data.MVP2MVP . . . . . 60  
 simer.Data.Ped . . . . . 61  
 simer.Data.Pheno . . . . . 63  
 simer.Data.SELIND . . . . . 64  
 simer.Version . . . . . 65  
 write.file . . . . . 66

**Index** **68**

annotation *Annotation simulation*

**Description**

Generating a map with annotation information

**Usage**

annotation(SP, verbose = TRUE)

**Arguments**

SP a list of all simulation parameters.  
 verbose whether to print detail.

**Details**

Build date: Nov 14, 2018 Last update: Jul 10, 2022

**Value**

the function returns a list containing

- \$map\$pop.map** the map data with annotation information.
- \$map\$species** the species of genetic map, which can be "arabidopsis", "cattle", "chicken", "dog", "horse", "human", "maize", "mice", "pig", and "rice".
- \$map\$pop.marker** the number of markers.
- \$map\$num.chr** the number of chromosomes.
- \$map\$len.chr** the length of chromosomes.
- \$map\$qtn.model** the genetic model of QTN such as 'A + D'.
- \$map\$qtn.index** the QTN index for each trait.

**\$map\$qtn.num** the QTN number for (each group in) each trait.  
**\$map\$qtn.dist** the QTN distribution containing 'norm', 'geom', 'gamma' or 'beta'.  
**\$map\$qtn.var** the variances for normal distribution.  
**\$map\$qtn.prob** the probability of success for geometric distribution.  
**\$map\$qtn.shape** the shape parameter for gamma distribution.  
**\$map\$qtn.scale** the scale parameter for gamma distribution.  
**\$map\$qtn.shape1** the shape1 parameter for beta distribution.  
**\$map\$qtn.shape2** the shape2 parameter for beta distribution.  
**\$map\$qtn.ncp** the ncp parameter for beta distribution.  
**\$map\$qtn.spot** the QTN distribution probability in each block.  
**\$map\$len.block** the block length.  
**\$map\$maf** the maf threshold, markers less than this threshold will be exclude.  
**\$map\$recom.spot** whether to generate recombination events.  
**\$map\$range.hot** the recombination times range in the hot spot.  
**\$map\$range.cold** the recombination times range in the cold spot.

#### Author(s)

Dong Yin

#### Examples

```

# Generate annotation simulation parameters
SP <- param.annot(qtn.num = list(tr1 = 10))

# Run annotation simulation
SP <- annotation(SP)
  
```

---

build.cov

*Correlation building*

---

#### Description

To bulid correlation of variables.

#### Usage

```
build.cov(df = NULL, mu = rep(0, nrow(Sigma)), Sigma = diag(2), tol = 1e-06)
```

#### Arguments

df	a data frame needing building correlation.
mu	means of the variables.
Sigma	covariance matrix of variables.
tol	tolerance (relative to largest variance) for numerical lack of positive-definiteness in Sigma.

### Details

Build date: Oct 10, 2019 Last update: Apr 28, 2022

### Value

a data frame with expected correlation

### Author(s)

Dong Yin and R

### References

B. D. Ripley (1987) Stochastic Simulation. Wiley. Page 98

### Examples

```
df <- data.frame(tr1 = rnorm(100), tr2 = rnorm(100))
df.cov <- build.cov(df)
var(df.cov)
```

---

cal.eff

*QTN genetic effects*

---

### Description

Calculate for genetic effects vector of selected markers.

### Usage

```
cal.eff(  
  qtn.num = 10,  
  qtn.dist = "norm",  
  qtn.var = 1,  
  qtn.prob = 0.5,  
  qtn.shape = 1,  
  qtn.scale = 1,  
  qtn.shape1 = 1,  
  qtn.shape2 = 1,  
  qtn.ncp = 0  
)
```

**Arguments**

qtn.num	integer: the QTN number of single trait; vector: the multiple group QTN number of single trait; matrix: the QTN number of multiple traits.
qtn.dist	the QTN distribution containing 'norm', 'geom', 'gamma' or 'beta'.
qtn.var	the standard deviations for normal distribution.
qtn.prob	the probability of success for geometric distribution.
qtn.shape	the shape parameter for gamma distribution.
qtn.scale	the scale parameter for gamma distribution.
qtn.shape1	the shape1 parameter for beta distribution.
qtn.shape2	the shape2 parameter for beta distribution.
qtn.ncp	the ncp parameter for beta distribution.

**Details**

Build date: Nov 14, 2018 Last update: Apr 28, 2022

**Value**

a vector of genetic effect.

**Author(s)**

Dong Yin

**Examples**

```
eff <- cal.eff(qtn.num = 10)
str(eff)
```

---

checkEnv

*Environmental factor checking*

---

**Description**

Check the levels of environmental factors.

**Usage**

```
checkEnv(data, envName, verbose = TRUE)
```

**Arguments**

data	data needing check.
envName	the environmental factor name within the data.
verbose	whether to print detail.

**Details**

Build date: Sep 10, 2021 Last update: Apr 28, 2022

**Value**

data without environmental factors of wrong level.

**Author(s)**

Dong Yin

**Examples**

```
data <- data.frame(a = c(1, 1, 2), b = c(2, 2, 3), c = c(3, 3, 4))
envName <- c("a", "b", "c")
data <- checkEnv(data = data, envName = envName)
```

---

generate.map

*Marker information*

---

**Description**

Generate map data with marker information.

**Usage**

```
generate.map(  
  species = NULL,  
  pop.marker = NULL,  
  num.chr = 18,  
  len.chr = 1.5e+08  
)
```

**Arguments**

species	the species of genetic map, which can be "arabidopsis", "cattle", "chicken", "dog", "horse", "human", "maize", "mice", "pig", and "rice".
pop.marker	the number of markers.
num.chr	the number of chromosomes.
len.chr	the length of chromosomes.

**Details**

Build date: Mar 19, 2022 Last update: Apr 28, 2022

**Value**

a data frame with marker information.

**Author(s)**

Dong Yin

**Examples**

```
pop.map <- generate.map(pop.marker = 1e4)
str(pop.map)
```

---

generate.pop	<i>Population generator</i>
--------------	-----------------------------

---

**Description**

Generate population according to the number of individuals.

**Usage**

```
generate.pop(pop.ind = 100, from = 1, ratio = 0.5, gen = 1)
```

**Arguments**

pop.ind	the number of the individuals in a population.
from	initial index of the population.
ratio	sex ratio of males in a population.
gen	generation ID of the population.

**Details**

Build date: Nov 14, 2018 Last update: Apr 28, 2022

**Value**

a data frame of population information.

**Author(s)**

Dong Yin

**Examples**

```
pop <- generate.pop(pop.ind = 100)
head(pop)
```



---

geno.cvt1	<i>Genotype code convertor 1</i>
-----------	----------------------------------

---

**Description**

Convert genotype matrix from (0, 1) to (0, 1, 2).

**Usage**

```
geno.cvt1(pop.geno)
```

**Arguments**

pop.geno            genotype matrix of (0, 1).

**Details**

Build date: Nov 14, 2018 Last update: Apr 28, 2022

**Value**

genotype matrix of (0, 1, 2).

**Author(s)**

Dong Yin

**Examples**

```
SP <- param.geno(pop.marker = 1e4, pop.ind = 1e2, incols = 2)
SP <- genotype(SP)
geno1 <- SP$geno$pop.geno$gen1
geno2 <- geno.cvt1(geno1)
geno1[1:6, 1:4]
geno2[1:6, 1:2]
```

---

geno.cvt2	<i>Genotype code convertor 2</i>
-----------	----------------------------------

---

**Description**

Convert genotype matrix from (0, 1, 2) to (0, 1).

**Usage**

```
geno.cvt2(pop.geno)
```

**Arguments**

pop.geno            genotype matrix of (0, 1, 2).

**Details**

Build date: Jul 11, 2020 Last update: Apr 28, 2022

**Value**

genotype matrix of (0, 1).

**Author(s)**

Dong Yin

**Examples**

```
SP <- param.geno(pop.marker = 1e4, pop.ind = 1e2, incols = 1)
SP <- genotype(SP)
geno1 <- SP$geno$pop.geno$gen1
geno2 <- geno.cvt2(geno1)
geno1[1:6, 1:2]
geno2[1:6, 1:4]
```

---

genotype

*Genotype simulation*

---

**Description**

Generating and editing genotype data.

**Usage**

```
genotype(SP = NULL, ncpus = 0, verbose = TRUE)
```

**Arguments**

SP                    a list of all simulation parameters.

ncpus                the number of threads used, if NULL, (logical core number - 1) is automatically used.

verbose              whether to print detail.

**Details**

Build date: Nov 14, 2018 Last update: Apr 28, 2022

**Value**

the function returns a list containing

**\$geno\$pop.geno** the genotype data.

**\$geno\$incols** '1': one-column genotype represents an individual; '2': two-column genotype represents an individual.

**\$geno\$pop.marker** the number of markers.

**\$geno\$pop.ind** the number of individuals in the base population.

**\$geno\$prob** the genotype code probability.

**\$geno\$rate.mut** the mutation rate of the genotype data.

**\$geno\$ld** whether to generate a complete LD genotype data when 'incols == 2'.

**Author(s)**

Dong Yin

**Examples**

```
# Generate genotype simulation parameters
SP <- param.geno(pop.marker = 1e4, pop.ind = 1e2)

# Run genotype simulation
SP <- genotype(SP)
```

---

getfam

*Family index and within-family index*

---

**Description**

Get indice of family and within-family

**Usage**

```
getfam(sir, dam, fam.op, mode = c("pat", "mat", "pm"))
```

**Arguments**

**sir** the indice of sires.

**dam** the indice of dams.

**fam.op** the initial index of family indice.

**mode** "pat": paternal mode; "mat": maternal mode; "pm": paternal and maternal mode.

**Details**

Build date: Nov 14, 2018 Last update: Apr 30, 2022

**Value**

a matrix with family indice and within-family indice.

**Author(s)**

Dong Yin

**Examples**

```
s <- c(0, 0, 0, 0, 1, 3, 3, 1, 5, 7, 5, 7, 1, 3, 5, 7)
d <- c(0, 0, 0, 0, 2, 4, 4, 2, 6, 8, 8, 6, 6, 8, 4, 8)
fam <- getfam(sir = s, dam = d, fam.op = 1, mode = "pm")
fam
```

---

GxG.network

*Genetic interaction network*

---

**Description**

Generate genetic interaction effect combination network.

**Usage**

```
GxG.network(pop.map = NULL, qtn.pos = 1:10, qtn.model = "A:D")
```

**Arguments**

pop.map	the map data with annotation information.
qtn.pos	the index of QTNs in the map data.
qtn.model	the genetic model of QTN such as 'A:D'.

**Details**

Build date: Mar 19, 2022 Last update: Apr 28, 2022

**Value**

a data frame of genetic interaction effect.

**Author(s)**

Dong Yin

**Examples**

```
pop.map <- generate.map(pop.marker = 1e4)
GxG.net <- GxG.network(pop.map)
head(GxG.net)
```

---

IndPerGen	<i>Individual number per generation</i>
-----------	---

---

**Description**

Calculate the individual number per generation.

**Usage**

```
IndPerGen(  
  pop,  
  pop.gen = 2,  
  ps = c(0.8, 0.8),  
  reprod.way = "randmate",  
  sex.rate = 0.5,  
  prog = 2  
)
```

**Arguments**

pop	the population information containing environmental factors and other effects.
pop.gen	the generations of simulated population.
ps	if $ps \leq 1$ , fraction selected in selection of males and females; if $ps > 1$ , $ps$ is number of selected males and females.
reprod.way	reproduction method, it consists of 'clone', 'dh', 'selfpol', 'randmate', 'randexself', 'assort', 'disassort', '2waycro', '3waycro', '4waycro', 'backcro', and 'userped'.
sex.rate	the sex ratio of simulated population.
prog	the progeny number of an individual.

**Details**

Build date: Apr 12, 2022 Last update: Apr 30, 2022

**Value**

the vector containing the individual number per generation.

**Author(s)**

Dong Yin

**Examples**

```
pop <- generate.pop(pop.ind = 100)
count.ind <- IndPerGen(pop)
```

---

logging.initialize      *Logging initialization*

---

**Description**

Initialize the logging process.

**Usage**

```
logging.initialize(module, outpath)
```

**Arguments**

module                  the module name.  
outpath                 the path of output files, Simer writes files only if outpath is not 'NULL'.

**Details**

Build date: Jul 11, 2020 Last update: Apr 28, 2022

**Value**

none.

**Author(s)**

Dong Yin

---

logging.log              *Logging*

---

**Description**

Print or write log.

**Usage**

```
logging.log(  
  ...,  
  file = NULL,  
  sep = " ",  
  fill = FALSE,  
  labels = NULL,  
  verbose = TRUE  
)
```

**Arguments**

...	R objects.
file	a connection or a character string naming the file to print to. If "" (the default), cat prints to the standard output connection, the console unless redirected by sink. If it is "lcmd", the output is piped to the command given by 'cmd', by opening a pipe connection.
sep	a character vector of strings to append after each element.
fill	a logical or (positive) numeric controlling how the output is broken into successive lines.
labels	a character vector of labels for the lines printed. Ignored if fill is FALSE.
verbose	whether to print detail.

**Details**

Build date: Jul 11, 2020 Last update: Apr 28, 2022

**Value**

none.

**Author(s)**

Dong Yin

**Examples**

```
logging.log('simer')
```

logging.print      *Logging printer*

---

### Description

Print R object information into file.

### Usage

```
logging.print(x, file = NULL, append = TRUE, verbose = TRUE)
```

### Arguments

x	a matrix or a list.
file	the filename of output file.
append	logical. If TRUE, output will be appended to file; otherwise, it will overwrite the contents of file.
verbose	whether to print details.

### Details

Build date: Feb 7, 2020 Last update: Apr 28, 2022

### Value

none.

### Author(s)

Dong Yin

### Examples

```
x <- list(a = "a", b = "b")
logging.print(x)
```



---

mate

*Mate*

---

### Description

Mating according to the indice of sires and dams.

### Usage

```
mate(pop.geno, index.sir, index.dam, ncpus = 0)
```

### Arguments

pop.geno	the genotype data.
index.sir	the indice of sires.
index.dam	the indice of dams.
ncpus	the number of threads used, if NULL, (logical core number - 1) is automatically used.

### Details

Build date: Nov 14, 2018 Last update: Apr 30, 2022

### Value

a genotype matrix after mating

### Author(s)

Dong Yin

### Examples

```
# Generate the genotype data
SP <- param.geno(pop.marker = 1e4, pop.ind = 1e2)
SP <- genotype(SP)
pop.geno <- SP$geno$pop.geno$gen1

# The mating design
index.sir <- rep(1:50, each = 2)
index.dam <- rep(51:100, each = 2)

# Mate according to mating design
geno.curr <- mate(pop.geno = pop.geno, index.sir = index.sir,
                 index.dam = index.dam)
geno.curr[1:5, 1:5]
```

---

mate.2waycro	<i>Two-way cross</i>
--------------	----------------------

---

### Description

Produce individuals by two-way cross.

### Usage

```
mate.2waycro(SP, ncpus = 0, verbose = TRUE)
```

### Arguments

SP	a list of all simulation parameters.
ncpus	the number of threads used, if NULL, (logical core number - 1) is automatically used.
verbose	whether to print detail.

### Details

Build date: Nov 14, 2018 Last update: Apr 30, 2022

### Value

the function returns a list containing

**\$reprod\$pop.gen** the generations of simulated population.

**\$reprod\$reprod.way** reproduction method, it consists of 'clone', 'dh', 'selfpol', 'randmate', 'randexself', 'assort', 'disassort', 'assort', 'disassort', '2waycro', '3waycro', '4waycro', 'backcro', and 'userped'.

**\$reprod\$sex.rate** the sex ratio of simulated population.

**\$reprod\$prog** the progeny number of an individual.

**\$geno** a list of genotype simulation parameters.

**\$pheno** a list of phenotype simulation parameters.

### Author(s)

Dong Yin

**Examples**

```

# Generate annotation simulation parameters
SP <- param.annot(qtn.num = list(tr1 = 10))
# Generate genotype simulation parameters
SP <- param.geno(SP = SP, pop.marker = 1e4, pop.ind = 1e2)
# Generate phenotype simulation parameters
SP <- param.pheno(SP = SP, pop.ind = 100)
# Generate selection parameters
SP <- param.sel(SP = SP, sel.single = "ind")
# Generate reproduction parameters
SP <- param.reprod(SP = SP, reprod.way = "2waycro")

# Run annotation simulation
SP <- annotation(SP)
# Run genotype simulation
SP <- genotype(SP)
# Run phenotype simulation
SP <- phenotype(SP)
# Two different breeds are cut by sex
SP$pheno$pop$gen1$sex <- rep(c(1, 2), c(50, 50))
# Run selection
SP <- selects(SP)
# Run two-way cross
SP <- mate.2waycro(SP)

```

---

mate.3waycro

*Three-way cross*


---

**Description**

Produce individuals by three-way cross.

**Usage**

```
mate.3waycro(SP, ncpus = 0, verbose = TRUE)
```

**Arguments**

SP	a list of all simulation parameters.
ncpus	the number of threads used, if NULL, (logical core number - 1) is automatically used.
verbose	whether to print detail.

**Details**

Build date: Apr 11, 2022 Last update: Apr 30, 2022

**Value**

the function returns a list containing

**\$reprod\$pop.gen** the generations of simulated population.

**\$reprod\$reprod.way** reproduction method, it consists of 'clone', 'dh', 'selfpol', 'randmate', 'randexself', 'assort', 'disassort', '2waycro', '3waycro', '4waycro', 'backcro', and 'userped'.

**\$reprod\$sex.rate** the sex ratio of simulated population.

**\$reprod\$prog** the progeny number of an individual.

**\$geno** a list of genotype simulation parameters.

**\$pheno** a list of phenotype simulation parameters.

**Author(s)**

Dong Yin

**Examples**

```
# Generate annotation simulation parameters
SP <- param.annot(qtn.num = list(tr1 = 10))
# Generate genotype simulation parameters
SP <- param.geno(SP = SP, pop.marker = 1e4, pop.ind = 1e2)
# Generate phenotype simulation parameters
SP <- param.pheno(SP = SP, pop.ind = 100)
# Generate selection parameters
SP <- param.sel(SP = SP, sel.single = "ind")
# Generate reproduction parameters
SP <- param.reprod(SP = SP, reprod.way = "3waycro")

# Run annotation simulation
SP <- annotation(SP)
# Run genotype simulation
SP <- genotype(SP)
# Run phenotype simulation
SP <- phenotype(SP)
# Three different breeds are cut by sex
SP$pheno$pop$gen1$sex <- rep(c(1, 2, 1), c(30, 30, 40))
# Run selection
SP <- selects(SP)
# Run three-way cross
SP <- mate.3waycro(SP)
```

---

mate.4waycro

*Four-way cross process*

---

**Description**

Produce individuals by four-way cross.

**Usage**

```
mate.4waycro(SP, ncpus = 0, verbose = TRUE)
```

**Arguments**

SP	a list of all simulation parameters.
ncpus	the number of threads used, if NULL, (logical core number - 1) is automatically used.
verbose	whether to print detail.

**Details**

Build date: Apr 11, 2022 Last update: Apr 30, 2022

**Value**

the function returns a list containing

**\$reprod\$pop.gen** the generations of simulated population.

**\$reprod\$reprod.way** reproduction method, it consists of 'clone', 'dh', 'selfpol', 'randmate', 'randexself', 'assort', 'disassort', '2waycro', '3waycro', '4waycro', 'backcro', and 'userped'.

**\$reprod\$sex.rate** the sex ratio of simulated population.

**\$reprod\$prog** the progeny number of an individual.

**\$geno** a list of genotype simulation parameters.

**\$pheno** a list of phenotype simulation parameters.

**Author(s)**

Dong Yin

**Examples**

```
# Generate annotation simulation parameters
SP <- param.annot(qtn.num = list(tr1 = 10))
# Generate genotype simulation parameters
SP <- param.geno(SP = SP, pop.marker = 1e4, pop.ind = 1e2)
# Generate phenotype simulation parameters
SP <- param.pheno(SP = SP, pop.ind = 100)
# Generate selection parameters
SP <- param.sel(SP = SP, sel.single = "ind")
# Generate reproduction parameters
SP <- param.reprod(SP = SP, reprod.way = "4waycro")

# Run annotation simulation
SP <- annotation(SP)
# Run genotype simulation
SP <- genotype(SP)
# Run phenotype simulation
SP <- phenotype(SP)
```

```
# Four different breeds are cut by sex
SP$pheno$pop$gen1$sex <- rep(c(1, 2, 1, 2), c(25, 25, 25, 25))
# Run selection
SP <- selects(SP)
# Run four-way cross
SP <- mate.4waycro(SP)
```

---

mate.assort

*Assortative mating*


---

## Description

Produce individuals by assortative mating.

## Usage

```
mate.assort(SP, ncpus = 0, verbose = TRUE)
```

## Arguments

SP	a list of all simulation parameters.
ncpus	the number of threads used, if NULL, (logical core number - 1) is automatically used.
verbose	whether to print detail.

## Details

Build date: Sep 30, 2022 Last update: Sep 30, 2022

## Value

the function returns a list containing

**\$reprod\$pop.gen** the generations of simulated population.

**\$reprod\$reprod.way** reproduction method, it consists of 'clone', 'dh', 'selfpol', 'randmate', 'randexself', 'assort', 'disassort', '2waycro', '3waycro', '4waycro', 'backcro', and 'userped'.

**\$reprod\$sex.rate** the sex ratio of simulated population.

**\$reprod\$prog** the progeny number of an individual.

**\$geno** a list of genotype simulation parameters.

**\$pheno** a list of phenotype simulation parameters.

## Author(s)

Dong Yin

**Examples**

```

# Generate annotation simulation parameters
SP <- param.annot(qtn.num = list(tr1 = 10))
# Generate genotype simulation parameters
SP <- param.geno(SP = SP, pop.marker = 1e4, pop.ind = 1e2)
# Generate phenotype simulation parameters
SP <- param.pheno(SP = SP, pop.ind = 100)
# Generate selection parameters
SP <- param.sel(SP = SP, sel.single = "ind")
# Generate reproduction parameters
SP <- param.reprod(SP = SP, reprod.way = "assort")

# Run annotation simulation
SP <- annotation(SP)
# Run genotype simulation
SP <- genotype(SP)
# Run phenotype simulation
SP <- phenotype(SP)
# Run selection
SP <- selects(SP)
# Run random mating
SP <- mate.assort(SP)

```

---

mate.backcro

*Back cross*


---

**Description**

Produce individuals by back cross.

**Usage**

```
mate.backcro(SP, ncpus = 0, verbose = TRUE)
```

**Arguments**

SP	a list of all simulation parameters.
ncpus	the number of threads used, if NULL, (logical core number - 1) is automatically used.
verbose	whether to print detail.

**Details**

Build date: Apr 12, 2022 Last update: Apr 30, 2022

**Value**

the function returns a list containing

**\$reprod\$pop.gen** the generations of simulated population.

**\$reprod\$reprod.way** reproduction method, it consists of 'clone', 'dh', 'selfpol', 'randmate', 'randexself', 'assort', 'disassort', '2waycro', '3waycro', '4waycro', 'backcro', and 'userped'.

**\$reprod\$sex.rate** the sex ratio of simulated population.

**\$reprod\$prog** the progeny number of an individual.

**\$geno** a list of genotype simulation parameters.

**\$pheno** a list of phenotype simulation parameters.

**Author(s)**

Dong Yin

**Examples**

```
# Generate annotation simulation parameters
SP <- param.annot(qtn.num = list(tr1 = 10))
# Generate genotype simulation parameters
SP <- param.geno(SP = SP, pop.marker = 1e4, pop.ind = 1e2)
# Generate phenotype simulation parameters
SP <- param.pheno(SP = SP, pop.ind = 100)
# Generate selection parameters
SP <- param.sel(SP = SP, sel.single = "ind")
# Generate reproduction parameters
SP <- param.reprod(SP = SP, reprod.way = "backcro")

# Run annotation simulation
SP <- annotation(SP)
# Run genotype simulation
SP <- genotype(SP)
# Run phenotype simulation
SP <- phenotype(SP)
# Two different breeds are cut by sex
SP$pheno$pop$gen1$sex <- rep(c(1, 2), c(50, 50))
# Run selection
SP <- selects(SP)
# Run back cross
SP <- mate.backcro(SP)
```

---

mate.clone

*Clone*

---

**Description**

Produce individuals by clone.



**Usage**

```
mate.clone(SP, ncpus = 0, verbose = TRUE)
```

**Arguments**

SP	a list of all simulation parameters.
ncpus	the number of threads used, if NULL, (logical core number - 1) is automatically used.
verbose	whether to print detail.

**Details**

Build date: Nov 14, 2018 Last update: Apr 30, 2022

**Value**

the function returns a list containing

**\$reprod\$pop.gen** the generations of simulated population.

**\$reprod\$reprod.way** reproduction method, it consists of 'clone', 'dh', 'selfpol', 'randmate', 'randexself', 'assort', 'disassort', '2waycro', '3waycro', '4waycro', 'backcro', and 'userped'.

**\$reprod\$sex.rate** the sex ratio of simulated population.

**\$reprod\$prog** the progeny number of an individual.

**\$geno** a list of genotype simulation parameters.

**\$pheno** a list of phenotype simulation parameters.

**Author(s)**

Dong Yin

**Examples**

```
# Generate annotation simulation parameters
SP <- param.annot(qtn.num = list(tr1 = 10))
# Generate genotype simulation parameters
SP <- param.geno(SP = SP, pop.marker = 1e4, pop.ind = 1e2)
# Generate phenotype simulation parameters
SP <- param.pheno(SP = SP, pop.ind = 100)
# Generate selection parameters
SP <- param.sel(SP = SP, sel.single = "ind")
# Generate reproduction parameters
SP <- param.reprod(SP = SP, reprod.way = "clone")

# Run annotation simulation
SP <- annotation(SP)
# Run genotype simulation
SP <- genotype(SP)
# Run phenotype simulation
SP <- phenotype(SP)
```

```
# Run selection
SP <- selects(SP)
# Run clone
SP <- mate.clone(SP)
```

---

mate.dh

*Doubled haploid*


---

### Description

Produce individuals by doubled haploid.

### Usage

```
mate.dh(SP, ncpus = 0, verbose = TRUE)
```

### Arguments

SP	a list of all simulation parameters.
ncpus	the number of threads used, if NULL, (logical core number - 1) is automatically used.
verbose	whether to print detail.

### Details

Build date: Nov 14, 2018 Last update: Apr 30, 2022

### Value

the function returns a list containing

**\$reprod\$pop.gen** the generations of simulated population.

**\$reprod\$reprod.way** reproduction method, it consists of 'clone', 'dh', 'selfpol', 'randmate', 'randexself', 'assort', 'disassort', '2waycro', '3waycro', '4waycro', 'backcro', and 'userped'.

**\$reprod\$sex.rate** the sex ratio of simulated population.

**\$reprod\$prog** the progeny number of an individual.

**\$geno** a list of genotype simulation parameters.

**\$pheno** a list of phenotype simulation parameters.

### Author(s)

Dong Yin

## Examples

```
# Generate annotation simulation parameters
SP <- param.annot(qtn.num = list(tr1 = 10))
# Generate genotype simulation parameters
SP <- param.geno(SP = SP, pop.marker = 1e4, pop.ind = 1e2)
# Generate phenotype simulation parameters
SP <- param.pheno(SP = SP, pop.ind = 100)
# Generate selection parameters
SP <- param.sel(SP = SP, sel.single = "ind")
# Generate reproduction parameters
SP <- param.reprod(SP = SP, reprod.way = "dh")

# Run annotation simulation
SP <- annotation(SP)
# Run genotype simulation
SP <- genotype(SP)
# Run phenotype simulation
SP <- phenotype(SP)
# Run selection
SP <- selects(SP)
# Run doubled haploid
SP <- mate.dh(SP)
```

---

mate.disassort	<i>Disassortative mating</i>
----------------	------------------------------

---

## Description

Produce individuals by disassortative mating.

## Usage

```
mate.disassort(SP, ncpus = 0, verbose = TRUE)
```

## Arguments

SP	a list of all simulation parameters.
ncpus	the number of threads used, if NULL, (logical core number - 1) is automatically used.
verbose	whether to print detail.

## Details

Build date: Sep 30, 2022 Last update: Sep 30, 2022

**Value**

the function returns a list containing

**\$reprod\$pop.gen** the generations of simulated population.

**\$reprod\$reprod.way** reproduction method, it consists of 'clone', 'dh', 'selfpol', 'randmate', 'randexself', 'assort', 'disassort', '2waycro', '3waycro', '4waycro', 'backcro', and 'userped'.

**\$reprod\$sex.rate** the sex ratio of simulated population.

**\$reprod\$prog** the progeny number of an individual.

**\$geno** a list of genotype simulation parameters.

**\$pheno** a list of phenotype simulation parameters.

**Author(s)**

Dong Yin

**Examples**

```
# Generate annotation simulation parameters
SP <- param.annot(qtn.num = list(tr1 = 10))
# Generate genotype simulation parameters
SP <- param.geno(SP = SP, pop.marker = 1e4, pop.ind = 1e2)
# Generate phenotype simulation parameters
SP <- param.pheno(SP = SP, pop.ind = 100)
# Generate selection parameters
SP <- param.sel(SP = SP, sel.single = "ind")
# Generate reproduction parameters
SP <- param.reprod(SP = SP, reprod.way = "disassort")

# Run annotation simulation
SP <- annotation(SP)
# Run genotype simulation
SP <- genotype(SP)
# Run phenotype simulation
SP <- phenotype(SP)
# Run selection
SP <- selects(SP)
# Run random mating
SP <- mate.assort(SP)
```

---

mate.randexself

*Random mating excluding self-pollination*

---

**Description**

Produce individuals by random mating excluding self-pollination.

**Usage**

```
mate.randexself(SP, ncpus = 0, verbose = TRUE)
```

**Arguments**

SP	a list of all simulation parameters.
ncpus	the number of threads used, if NULL, (logical core number - 1) is automatically used.
verbose	whether to print detail.

**Details**

Build date: Nov 14, 2018 Last update: Apr 30, 2022

**Value**

the function returns a list containing

**\$reprod\$pop.gen** the generations of simulated population.

**\$reprod\$reprod.way** reproduction method, it consists of 'clone', 'dh', 'selfpol', 'randmate', 'randexself', 'assort', 'disassort', '2waycro', '3waycro', '4waycro', 'backcro', and 'userped'.

**\$reprod\$sex.rate** the sex ratio of simulated population.

**\$reprod\$prog** the progeny number of an individual.

**\$geno** a list of genotype simulation parameters.

**\$pheno** a list of phenotype simulation parameters.

**Author(s)**

Dong Yin

**Examples**

```
# Generate annotation simulation parameters
SP <- param.annot(qtn.num = list(tr1 = 10))
# Generate genotype simulation parameters
SP <- param.geno(SP = SP, pop.marker = 1e4, pop.ind = 1e2)
# Generate phenotype simulation parameters
SP <- param.pheno(SP = SP, pop.ind = 100)
# Generate selection parameters
SP <- param.sel(SP = SP, sel.single = "ind")
# Generate reproduction parameters
SP <- param.reprod(SP = SP, reprod.way = "randexself")

# Run annotation simulation
SP <- annotation(SP)
# Run genotype simulation
SP <- genotype(SP)
# Run phenotype simulation
SP <- phenotype(SP)
```

```
# Run selection
SP <- selects(SP)
# Run random mating excluding self-pollination
SP <- mate.randexself(SP)
```

---

mate.randmate	<i>Random mating</i>
---------------	----------------------

---

### Description

Produce individuals by random-mating.

### Usage

```
mate.randmate(SP, ncpus = 0, verbose = TRUE)
```

### Arguments

SP	a list of all simulation parameters.
ncpus	the number of threads used, if NULL, (logical core number - 1) is automatically used.
verbose	whether to print detail.

### Details

Build date: Nov 14, 2018 Last update: Apr 30, 2022

### Value

the function returns a list containing

**\$reprod\$pop.gen** the generations of simulated population.

**\$reprod\$reprod.way** reproduction method, it consists of 'clone', 'dh', 'selfpol', 'randmate', 'randexself', 'assort', 'disassort', '2waycro', '3waycro', '4waycro', 'backcro', and 'userped'.

**\$reprod\$sex.rate** the sex ratio of simulated population.

**\$reprod\$prog** the progeny number of an individual.

**\$geno** a list of genotype simulation parameters.

**\$pheno** a list of phenotype simulation parameters.

### Author(s)

Dong Yin

## Examples

```
# Generate annotation simulation parameters
SP <- param.annot(qtn.num = list(tr1 = 10))
# Generate genotype simulation parameters
SP <- param.geno(SP = SP, pop.marker = 1e4, pop.ind = 1e2)
# Generate phenotype simulation parameters
SP <- param.pheno(SP = SP, pop.ind = 100)
# Generate selection parameters
SP <- param.sel(SP = SP, sel.single = "ind")
# Generate reproduction parameters
SP <- param.reprod(SP = SP, reprod.way = "randmate")

# Run annotation simulation
SP <- annotation(SP)
# Run genotype simulation
SP <- genotype(SP)
# Run phenotype simulation
SP <- phenotype(SP)
# Run selection
SP <- selects(SP)
# Run random mating
SP <- mate.randmate(SP)
```

---

mate.selfpol

*Self-pollination*

---

## Description

Produce individuals by self-pollination.

## Usage

```
mate.selfpol(SP, ncpus = 0, verbose = TRUE)
```

## Arguments

SP	a list of all simulation parameters.
ncpus	the number of threads used, if NULL, (logical core number - 1) is automatically used.
verbose	whether to print detail.

## Details

Build date: Nov 14, 2018 Last update: Apr 30, 2022

**Value**

the function returns a list containing

**\$reprod\$pop.gen** the generations of simulated population.

**\$reprod\$reprod.way** reproduction method, it consists of 'clone', 'dh', 'selfpol', 'randmate', 'randexself', 'assort', 'disassort', '2waycro', '3waycro', '4waycro', 'backcro', and 'userped'.

**\$reprod\$sex.rate** the sex ratio of simulated population.

**\$reprod\$prog** the progeny number of an individual.

**\$geno** a list of genotype simulation parameters.

**\$pheno** a list of phenotype simulation parameters.

**Author(s)**

Dong Yin

**Examples**

```
# Generate annotation simulation parameters
SP <- param.annot(qtn.num = list(tr1 = 10))
# Generate genotype simulation parameters
SP <- param.geno(SP = SP, pop.marker = 1e4, pop.ind = 1e2)
# Generate phenotype simulation parameters
SP <- param.pheno(SP = SP, pop.ind = 100)
# Generate selection parameters
SP <- param.sel(SP = SP, sel.single = "ind")
# Generate reproduction parameters
SP <- param.reprod(SP = SP, reprod.way = "selfpol")

# Run annotation simulation
SP <- annotation(SP)
# Run genotype simulation
SP <- genotype(SP)
# Run phenotype simulation
SP <- phenotype(SP)
# Run selection
SP <- selects(SP)
# Run self-pollination
SP <- mate.selfpol(SP)
```

---

mate.userped

*User-specified pedigree mating*

---

**Description**

Produce individuals by user-specified pedigree mating.



**Usage**

```
mate.userped(SP, ncpus = 0, verbose = TRUE)
```

**Arguments**

SP	a list of all simulation parameters.
ncpus	the number of threads used, if NULL, (logical core number - 1) is automatically used.
verbose	whether to print detail.

**Details**

Build date: Apr 12, 2022 Last update: Apr 30, 2022

**Value**

the function returns a list containing

**\$reprod\$pop.sel** the generations of simulated population.

**\$reprod\$reprod.way** reproduction method, it consists of 'clone', 'dh', 'selfpol', 'randmate', 'randexself', 'assort', 'disassort', '2waycro', '3waycro', '4waycro', 'backcro', and 'userped'.

**\$reprod\$sex.rate** the sex ratio of simulated population.

**\$reprod\$prog** the progeny number of an individual.

**\$geno** a list of genotype simulation parameters.

**\$pheno** a list of phenotype simulation parameters.

**Author(s)**

Dong Yin

**Examples**

```
# Generate annotation simulation parameters
SP <- param.annot(qtn.num = list(tr1 = 10))
# Generate genotype simulation parameters
SP <- param.geno(SP = SP, pop.marker = 1e4, pop.ind = 1e2)
# Generate phenotype simulation parameters
SP <- param.pheno(SP = SP, pop.ind = 100)
# Generate reproduction parameters
SP <- param.reprod(SP = SP, reprod.way = "userped")

# Run annotation simulation
SP <- annotation(SP)
# Run genotype simulation
SP <- genotype(SP)
# Run phenotype simulation
SP <- phenotype(SP)
# Run user-specified pedigree mating
SP <- mate.userped(SP)
```

---

param.annot                      *Annotation parameters generator*

---

## Description

Generate parameters for annotation data simulation.

## Usage

```
param.annot(SP = NULL, ...)
```

## Arguments

SP                      a list of all simulation parameters.  
 ...                     one or more parameter(s) for map simulation.

## Details

Build date: Feb 24, 2022 Last update: Jul 10, 2022

## Value

the function returns a list containing

**\$map\$pop.map** the map data with annotation information.  
**\$map\$species** the species of genetic map, which can be "arabidopsis", "cattle", "chicken", "dog", "horse", "human", "maize", "mice", "pig", and "rice".  
**\$map\$pop.marker** the number of markers.  
**\$map\$num.chr** the number of chromosomes.  
**\$map\$len.chr** the length of chromosomes.  
**\$map\$qtn.model** the genetic model of QTN such as 'A + D'.  
**\$map\$qtn.index** the QTN index for each trait.  
**\$map\$qtn.num** the QTN number for (each group in) each trait.  
**\$map\$qtn.dist** the QTN distribution containing 'norm', 'geom', 'gamma' or 'beta'.  
**\$map\$qtn.var** the standard deviations for normal distribution.  
**\$map\$qtn.prob** the probability of success for geometric distribution.  
**\$map\$qtn.shape** the shape parameter for gamma distribution.  
**\$map\$qtn.scale** the scale parameter for gamma distribution.  
**\$map\$qtn.shape1** the shape1 parameter for beta distribution.  
**\$map\$qtn.shape2** the shape2 parameter for beta distribution.  
**\$map\$qtn.ncp** the ncp parameter for beta distribution.  
**\$map\$qtn.spot** the QTN distribution probability in each block.

- \$map\$len.block** the block length.
- \$map\$maf** the maf threshold, markers less than this threshold will be exclude.
- \$map\$recom.spot** whether to generate recombination events.
- \$map\$range.hot** the recombination times range in the hot spot.
- \$map\$range.cold** the recombination times range in the cold spot.

### Author(s)

Dong Yin

### Examples

```
SP <- param.annot(qtn.num = list(tr1 = 10))
str(SP)
```

---

param.geno	<i>Genotype parameters generator</i>
------------	--------------------------------------

---

### Description

Generate parameters for genotype data simulation.

### Usage

```
param.geno(SP = NULL, ...)
```

### Arguments

- SP a list of all simulation parameters.
- ... one or more parameter(s) for genotype simulation.

### Details

Build date: Feb 21, 2022 Last update: Jul 4, 2022

### Value

the function returns a list containing

- \$geno\$pop.geno** the genotype data.
- \$geno\$incols** '1': one-column genotype represents an individual; '2': two-column genotype represents an individual.
- \$geno\$pop.marker** the number of markers.
- \$geno\$pop.ind** the number of individuals in the base population.
- \$geno\$prob** the genotype code probability.
- \$geno\$rate.mut** the mutation rate of the genotype data.
- \$geno\$clld** whether to generate a complete LD genotype data when 'incols == 2'.

**Author(s)**

Dong Yin

**Examples**

```
SP <- param.geno(pop.marker = 1e4, pop.ind = 1e2)
str(SP)
```

---

 param.global

*Global parameters generator*


---

**Description**

Generate parameters for global options.

**Usage**

```
param.global(SP = NULL, ...)
```

**Arguments**

SP	a list of all simulation parameters.
...	one or more parameter(s) for global options.

**Details**

Build date: Apr 16, 2022 Last update: Jul 4, 2022

**Value**

the function returns a list containing

**\$replication** the replication times of simulation.

**\$seed.sim** simulation random seed.

**\$out** the prefix of output files.

**\$outpath** the path of output files, Simer writes files only if outpath is not 'NULL'.

**\$out.format** 'numeric' or 'plink', the data format of output files.

**\$pop.gen** the generations of simulated population.

**\$out.geno.gen** the output generations of genotype data.

**\$out.pheno.gen** the output generations of phenotype data.

**\$useAllGeno** whether to use all genotype data to simulate phenotype.

**\$ncpus** the number of threads used, if NULL, (logical core number - 1) is automatically used.

**\$verbose** whether to print detail.

**Author(s)**

Dong Yin

**Examples**

```
SP <- param.global(out = "simer")
str(SP)
```

---

param.pheno	<i>Phenotype parameters generator</i>
-------------	---------------------------------------

---

**Description**

Generate parameters for phenotype data simulation.

**Usage**

```
param.pheno(SP = NULL, ...)
```

**Arguments**

SP	a list of all simulation parameters.
...	one or more parameter(s) for phenotype simulation.

**Details**

Build date: Feb 21, 2022 Last update: Jul 4, 2022

**Value**

the function returns a list containing

**\$pheno\$pop** the population information containing environmental factors and other effects.

**\$pheno\$pop.ind** the number of individuals in the base population.

**\$pheno\$pop.rep** the repeated times of repeated records.

**\$pheno\$pop.rep.bal** whether repeated records are balanced.

**\$pheno\$pop.env** a list of environmental factors setting.

**\$pheno\$phe.type** a list of phenotype types.

**\$pheno\$phe.model** a list of genetic model of phenotype such as "T1 = A + E".

**\$pheno\$phe.h2A** a list of additive heritability.

**\$pheno\$phe.h2D** a list of dominant heritability.

**\$pheno\$phe.h2GxG** a list of GxG interaction heritability.

**\$pheno\$phe.h2GxE** a list of GxE interaction heritability.

**\$pheno\$phe.h2PE** a list of permanent environmental heritability.

**\$pheno\$phe.var** a list of phenotype variance.  
**\$pheno\$phe.corA** the additive genetic correlation matrix.  
**\$pheno\$phe.corD** the dominant genetic correlation matrix.  
**\$pheno\$phe.corGxG** the GxG genetic correlation matrix.  
**\$pheno\$phe.corPE** the permanent environmental correlation matrix.  
**\$pheno\$phe.corE** the residual correlation matrix.

### Author(s)

Dong Yin

### Examples

```
SP <- param.pheno(phe.model = list(tr1 = "T1 = A + E"))
str(SP)
```

---

param.reprod

*Reproduction parameters generator*

---

### Description

Generate parameters for reproduction.

### Usage

```
param.reprod(SP = NULL, ...)
```

### Arguments

SP a list of all simulation parameters.  
 ... one or more parameter(s) for reproduction.

### Details

Build date: Apr 6, 2022 Last update: Jul 4, 2022

### Value

the function returns a list containing

**\$reprod\$pop.gen** the generations of simulated population.  
**\$reprod\$reprod.way** reproduction method, it consists of 'clone', 'dh', 'selfpol', 'randmate', 'randexself', 'assort', 'disassort', '2waycro', '3waycro', '4waycro', 'backcro', and 'userped'.  
**\$reprod\$sex.rate** the male rate in the population.  
**\$reprod\$prog** the progeny number of an individual.

**Author(s)**

Dong Yin

**Examples**

```
SP <- param.reprod(reprod.way = "randmate")
str(SP)
```

param.sel

*Selection parameters generator***Description**

Generate parameters for selection.

**Usage**

```
param.sel(SP = NULL, ...)
```

**Arguments**

SP                    a list of all simulation parameters.  
 ...                    one or more parameter(s) for selection.

**Details**

Build date: Apr 6, 2022 Last update: Jul 4, 2022

**Value**

the function returns a list containing

**\$sel\$pop.sel** the selected males and females.**\$sel\$ps** if  $ps \leq 1$ , fraction selected in selection of males and females; if  $ps > 1$ ,  $ps$  is number of selected males and females.**\$sel\$dexpr** whether the sort order is decreasing.**\$sel\$sel.crit** the selection criteria, it can be 'TBV', 'TGV', and 'pheno'.**\$sel\$sel.single** the single-trait selection method, it can be 'ind', 'fam', 'infam', and 'comb'.**\$sel\$sel.multi** the multiple-trait selection method, it can be 'index', 'indcul', and 'tmd'.**\$sel\$index.wt** the weight of each trait for multiple-trait selection.**\$sel\$index.tdm** the index of tandem selection for multiple-trait selection.**\$sel\$goal.perc** the percentage of goal more than the mean of scores of individuals.**\$sel\$pass.perc** the percentage of expected excellent individuals.

**Author(s)**

Dong Yin

**Examples**

```
SP <- param.sel(sel.single = "ind")
str(SP)
```

---

param.simer

*Parameter generator*

---

**Description**

Generate parameters for Simer.

**Usage**

```
param.simer(SP = NULL, ...)
```

**Arguments**

SP	a list of all simulation parameters.
...	one or more parameter(s) for simer.

**Details**

Build date: Apr 17, 2022 Last update: Jul 4, 2022

**Value**

the function returns a list containing

**\$global** a list of global parameters.

**\$map** a list of marker information parameters.

**\$geno** a list of genotype simulation parameters.

**\$pheno** a list of phenotype simulation parameters.

**\$sel** a list of selection parameters.

**\$reprod** a list of reproduction parameters.

**Author(s)**

Dong Yin

**Examples**

```
SP <- param.simer(out = "simer")
str(SP)
```



---

phenotype	<i>Phenotype simulation</i>
-----------	-----------------------------

---

**Description**

Generate single-trait or multiple-trait phenotype by mixed model.

**Usage**

```
phenotype(SP = NULL, verbose = TRUE)
```

**Arguments**

SP	a list of all simulation parameters.
verbose	whether to print detail.

**Details**

Build date: Nov 14, 2018 Last update: Apr 28, 2022

**Value**

the function returns a list containing

**\$pheno\$pop** the population information containing environmental factors and other effects.

**\$pheno\$pop.ind** the number of individuals in the base population.

**\$pheno\$pop.rep** the repeated times of repeated records.

**\$pheno\$pop.rep.bal** whether repeated records are balanced.

**\$pheno\$pop.env** a list of environmental factors setting.

**\$pheno\$phe.type** a list of phenotype types.

**\$pheno\$phe.model** a list of genetic model of phenotype such as "T1 = A + E".

**\$pheno\$phe.h2A** a list of additive heritability.

**\$pheno\$phe.h2D** a list of dominant heritability.

**\$pheno\$phe.h2GxG** a list of GxG interaction heritability.

**\$pheno\$phe.h2GxE** a list of GxE interaction heritability.

**\$pheno\$phe.h2PE** a list of permanent environmental heritability.

**\$pheno\$phe.var** a list of phenotype variance.

**\$pheno\$phe.corA** the additive genetic correlation matrix.

**\$pheno\$phe.corD** the dominant genetic correlation matrix.

**\$pheno\$phe.corGxG** the GxG genetic correlation matrix.

**\$pheno\$phe.corPE** the permanent environmental correlation matrix.

**\$pheno\$phe.corE** the residual correlation matrix.

**Author(s)**

Dong Yin

**References**Kao C and Zeng Z (2002) <<https://www.genetics.org/content/160/3/1243.long>>**Examples**

```

# Prepare environmental factor list
pop.env <- list(
  F1 = list( # fixed effect 1
    level = c("1", "2"),
    effect = list(tr1 = c(50, 30), tr2 = c(50, 30))
  ),
  F2 = list( # fixed effect 2
    level = c("d1", "d2", "d3"),
    effect = list(tr1 = c(10, 20, 30), tr2 = c(10, 20, 30))
  ),
  C1 = list( # covariate 1
    level = c(70, 80, 90),
    slope = list(tr1 = 1.5, tr2 = 1.5)
  ),
  R1 = list( # random effect 1
    level = c("l1", "l2", "l3"),
    ratio = list(tr1 = 0.1, tr2 = 0.1)
  )
)

# Generate genotype simulation parameters
SP <- param.annot(qtn.num = list(tr1 = c(2, 8), tr2 = 10),
  qtn.model = "A + D + A:D")
# Generate annotation simulation parameters
SP <- param.geno(SP = SP, pop.marker = 1e4, pop.ind = 1e2)
# Generate phenotype simulation parameters
SP <- param.pheno(
  SP = SP,
  pop.ind = 100,
  pop.rep = 2, # 2 repeated record
  pop.rep.bal = TRUE, # balanced repeated record
  pop.env = pop.env,
  phe.type = list(
    tr1 = "continuous",
    tr2 = list(case = 0.01, control = 0.99)
  ),
  phe.model = list(
    tr1 = "T1 = A + D + A:D + F1 + F2 + C1 + R1 + A:F1 + E",
    tr2 = "T2 = A + D + A:D + F1 + F2 + C1 + R1 + A:F1 + E"
  ),
  phe.var = list(tr1 = 100, tr2 = 100)
)

```

```
# Run annotation simulation
SP <- annotation(SP)
# Run genotype simulation
SP <- genotype(SP)
# Run phenotype simulation
SP <- phenotype(SP)
```

---

pop.geno	<i>Raw genotype matrix from outside in simdata</i>
----------	--

---

**Description**

Raw genotype matrix from outside in simdata

**Usage**

```
data(simdata)
```

**Format**

matrix

**Examples**

```
data(simdata)
dim(pop.geno)
head(pop.geno)
```

---

pop.map	<i>Map file from outside in simdata</i>
---------	---

---

**Description**

Map file from outside in simdata

**Usage**

```
data(simdata)
```

**Format**

list

**Examples**

```
data(simdata)
dim(pop.map)
head(pop.map)
```

remove\_bigmatrix      *Big.matrix removing*

---

### Description

Remove big.matrix safely.

### Usage

```
remove_bigmatrix(x, desc_suffix = ".geno.desc", bin_suffix = ".geno.bin")
```

### Arguments

x                      the filename of big.matrix.  
desc\_suffix          the suffix of description file of big.matrix.  
bin\_suffix            the suffix of binary file of big.matrix.

### Details

Build date: Aug 8, 2019 Last update: Apr 30, 2022

### Value

TRUE or FALSE

### Author(s)

Haohao Zhang and Dong Yin

### Examples

```
library(bigmemory)
mat <- filebacked.big.matrix(
  nrow = 10,
  ncol = 10,
  init = 0,
  type = 'char',
  backingpath = ".",
  backingfile = 'simer.geno.bin',
  descriptorfile = 'simer.geno.desc')

remove_bigmatrix(x = "simer")
```

reproduces

*Reproduction***Description**

Population reproduction by different mate design.

**Usage**

```
reproduces(SP, ncpus = 0, verbose = TRUE)
```

**Arguments**

SP	a list of all simulation parameters.
ncpus	the number of threads used, if NULL, (logical core number - 1) is automatically used.
verbose	whether to print detail.

**Details**

Build date: Nov 14, 2018 Last update: Apr 29, 2022

**Value**

the function returns a list containing

**\$reprod\$pop.gen** the generations of simulated population.

**\$reprod\$reprod.way** reproduction method, it consists of 'clone', 'dh', 'selfpol', 'randmate', 'randexself', 'assort', 'disassort', '2waycro', '3waycro', '4waycro', 'backcro', and 'userped'.

**\$reprod\$sex.rate** the male rate in the population.

**\$reprod\$prog** the progeny number of an individual.

**\$geno** a list of genotype simulation parameters.

**\$pheno** a list of phenotype simulation parameters.

**Author(s)**

Dong Yin

**Examples**

```
# Generate annotation simulation parameters
SP <- param.annot(qtn.num = list(tr1 = 10))
# Generate genotype simulation parameters
SP <- param.geno(SP = SP, pop.marker = 1e4, pop.ind = 1e2)
# Generate phenotype simulation parameters
SP <- param.pheno(SP = SP, pop.ind = 100)
# Generate selection parameters
```

```

SP <- param.sel(SP = SP, sel.single = "ind")
# Generate reproduction parameters
SP <- param.reprod(SP = SP, reprod.way = "randmate")

# Run annotation simulation
SP <- annotation(SP)
# Run genotype simulation
SP <- genotype(SP)
# Run phenotype simulation
SP <- phenotype(SP)
# Run selection
SP <- selects(SP)
# Run reproduction
SP <- reproduces(SP)

```

---

selects

*Selection*


---

### Description

Select individuals by combination of selection method and criterion.

### Usage

```
selects(SP = NULL, verbose = TRUE)
```

### Arguments

SP	a list of all simulation parameters.
verbose	whether to print detail.

### Details

Build date: Sep 8, 2018 Last update: Apr 30, 2022

### Value

the function returns a list containing

**\$sel\$pop.sel** the selected males and females.

**\$sel\$ps** if  $ps \leq 1$ , fraction selected in selection of males and females; if  $ps > 1$ ,  $ps$  is number of selected males and females.

**\$sel\$decr** whether the sort order is decreasing.

**\$sel\$sel.crit** the selection criteria, it can be 'TBV', 'TGV', and 'pheno'.

**\$sel\$sel.single** the single-trait selection method, it can be 'ind', 'fam', 'infam', and 'comb'.

**\$sel\$sel.multi** the multiple-trait selection method, it can be 'index', 'indcul', and 'tmd'.

**\$sel\$index.wt** the weight of each trait for multiple-trait selection.

**\$sel\$index.tdm** the index of tandem selection for multiple-trait selection.

**\$sel\$goal.perc** the percentage of goal more than the mean of scores of individuals.

**\$sel\$pass.perc** the percentage of expected excellent individuals.

### Author(s)

Dong Yin

### Examples

```
# Generate annotation simulation parameters
SP <- param.annot(qtn.num = list(tr1 = 10))
# Generate genotype simulation parameters
SP <- param.geno(SP = SP, pop.marker = 1e4, pop.ind = 1e2)
# Generate phenotype simulation parameters
SP <- param.pheno(SP = SP, pop.ind = 100)
# Generate selection parameters
SP <- param.sel(SP = SP, sel.single = "ind")

# Run annotation simulation
SP <- annotation(SP)
# Run genotype simulation
SP <- genotype(SP)
# Run phenotype simulation
SP <- phenotype(SP)
# Run selection
SP <- selects(SP)
```

---

simer

*Simer*

---

### Description

Main function of Simer.

### Usage

```
simer(SP)
```

### Arguments

SP                    a list of all simulation parameters.

### Details

Build date: Jan 7, 2019 Last update: Apr 29, 2022

**Value**

the function returns a list containing

**\$global** a list of global parameters.

**\$map** a list of marker information parameters.

**\$geno** a list of genotype simulation parameters.

**\$pheno** a list of phenotype simulation parameters.

**\$sel** a list of selection parameters.

**\$reprod** a list of reproduction parameters.

**Author(s)**

Dong Yin, Lilin Yin, Haohao Zhang, and Xiaolei Liu

**Examples**

```
# Generate all simulation parameters
SP <- param.simer(out = "simer")

# Run Simer
SP <- simer(SP)
```

---

simer.Data

*Data handling*

---

**Description**

Make data quality control for genotype, phenotype, and pedigree.

**Usage**

```
simer.Data(jsonList = NULL, out = "simer.qc", ncpus = 0, verbose = TRUE)
```

**Arguments**

jsonList	a list of data quality control parameters.
out	the prefix of output files.
ncpus	the number of threads used, if NULL, (logical core number - 1) is automatically used.
verbose	whether to print detail.

**Details**

Build date: May 26, 2021 Last update: Apr 28, 2022



**Value**

the function returns a list containing

**\$genotype** the path of genotype data.

**\$pedigree** the filename of pedigree data.

**\$selection\_index** the selection index for all traits.

**\$breeding\_value\_index** the breeding value index for all traits.

**\$quality\_control\_plan** a list of parameters for data quality control.

**\$breeding\_plan** a list of parameters for genetic evaluation.

**Author(s)**

Dong Yin

**Examples**

```
# Read JSON file
jsonFile <- system.file("extdata", "04breeding_plan", "plan1.json", package = "simer")
jsonList <- jsonlite::fromJSON(txt = jsonFile, simplifyVector = FALSE)

## Not run:
# It needs 'plink' and 'hiblup' software
jsonList <- simer.Data(jsonList = jsonList)

## End(Not run)
```

---

`simer.Data.Bfile2MVP` *simer.Data.Bfile2MVP: To transform plink binary data to MVP package*

---

**Description**

transforming plink binary data to MVP package.

**Usage**

```
simer.Data.Bfile2MVP(
  bfile,
  out = "simer",
  maxLine = 10000,
  priority = "speed",
  type.geno = "char",
  threads = 10,
  verbose = TRUE
)
```

**Arguments**

bfile	Genotype in binary format (.bed, .bim, .fam).
out	the name of output file.
maxLine	the max number of line to write to big matrix for each loop.
priority	'memory' or 'speed'.
type.geno	the type of genotype elements.
threads	number of thread for transforming.
verbose	whether to print the reminder.

**Details**

Build date: Sep 12, 2018 Last update: July 25, 2022

**Value**

number of individuals and markers. Output files: genotype.desc, genotype.bin: genotype file in bigmemory format phenotype.phe: ordered phenotype file, same taxa order with genotype file map.map: SNP information

**Author(s)**

Haohao Zhang and Dong Yin

**Examples**

```
# Get bfile path
bfilePath <- file.path(system.file("extdata", "02plinkb", package = "simer"), "demo")

# Data converting
simer.Data.Bfile2MVP(bfilePath, tempfile("outfile"))
```

---

simer.Data.cHIBLUP      *Genetic evaluation*

---

**Description**

The function of calling HIBLUP software of C version.

**Usage**

```
simer.Data.cHIBLUP(
  jsonList = NULL,
  hiblupPath = "",
  mode = "A",
  vc.method = "AI",
  ncpus = 10,
  verbose = TRUE
)
```

**Arguments**

jsonList	the list of genetic evaluation parameters.
hiblupPath	the path of HIBLUP software.
mode	'A' or 'AD', Additive effect model or Additive and Dominance model.
vc.method	default is 'AI', the method of calculating variance components in HIBLUP software.
ncpus	the number of threads used, if NULL, (logical core number - 1) is automatically used.
verbose	whether to print detail.

**Details**

Build date: June 28, 2021 Last update: Apr 28, 2022

**Value**

the function returns a list containing

**\$randList** a list of estimated random effects.

**\$varList** a list of variance components.

**\$covA** the genetic covariance matrix for all traits.

**\$corA** the genetic correlation matrix for all traits.

**Author(s)**

Dong Yin

**Examples**

```
# Read JSON file
jsonFile <- system.file("extdata", "04breeding_plan", "plan1.json", package = "simer")
jsonList <- jsonlite::fromJSON(txt = jsonFile, simplifyVector = FALSE)

## Not run:
# It needs 'hiblup' software
gebvs <- simer.Data.cHIBLUP(jsonList = jsonList)

## End(Not run)
```

---

simer.Data.Env      *Environmental factor selection*

---

### Description

To find appropriate fixed effects, covariates, and random effects.

### Usage

```
simer.Data.Env(
  jsonList = NULL,
  hiblupPath = "",
  header = TRUE,
  sep = "\t",
  ncpus = 10,
  verbose = TRUE
)
```

### Arguments

jsonList	the list of environmental factor selection parameters.
hiblupPath	the path of HIBLUP software.
header	the header of file.
sep	the separator of file.
ncpus	the number of threads used, if NULL, (logical core number - 1) is automatically used.
verbose	whether to print detail.

### Details

Build date: July 17, 2021 Last update: Apr 28, 2022

### Value

the function returns a list containing

**\$genotype** the path of genotype data.

**\$pedigree** the filename of pedigree data.

**\$selection\_index** the selection index for all traits.

**\$breeding\_value\_index** the breeding value index for all traits.

**\$quality\_control\_plan** a list of parameters for data quality control.

**\$breeding\_plan** a list of parameters for genetic evaluation.

### Author(s)

Dong Yin

**Examples**

```
# Read JSON file
jsonFile <- system.file("extdata", "04breeding_plan", "plan1.json", package = "simer")
jsonList <- jsonlite::fromJSON(txt = jsonFile, simplifyVector = FALSE)

## Not run:
# It needs 'hiblup' software
jsonList <- simer.Data.Env(jsonList = jsonList)

## End(Not run)
```

---

```
simer.Data.Geno      Genotype data quality control
```

---

**Description**

Data quality control for genotype data in MVP format and PLINK format.

**Usage**

```
simer.Data.Geno(
  fileMVP = NULL,
  fileBed = NULL,
  filePlinkPed = NULL,
  filePed = NULL,
  filePhe = NULL,
  out = "simer.qc",
  genoType = "char",
  filter = NULL,
  filterGeno = NULL,
  filterHWE = NULL,
  filterMind = NULL,
  filterMAF = NULL,
  ncpus = 0,
  verbose = TRUE
)
```

**Arguments**

fileMVP	genotype in MVP format.
fileBed	genotype in PLINK binary format.
filePlinkPed	genotype in PLINK numeric format.
filePed	the filename of pedigree data.
filePhe	the filename of phenotype data, it can be a vector.
out	the prefix of output files.

genoType	type parameter in bigmemory, genotype data. The default is char, it is highly recommended *NOT* to modify this parameter.
filter	filter of genotyped individual.
filterGeno	threshold of sample miss rate.
filterHWE	threshold of Hardy-Weinberg Test.
filterMind	threshold of variant miss rate.
filterMAF	threshold of Minor Allele Frequency.
ncpus	the number of threads used, if NULL, (logical core number - 1) is automatically used.
verbose	whether to print detail.

### Details

Build date: May 26, 2021 Last update: Apr 28, 2022

### Value

the function returns files

**<out>.bed** the .bed file of PLINK binary format.

**<out>.bim** the .bim file of PLINK binary format.

**<out>.fam** the .fam file of PLINK binary format.

### Author(s)

Dong Yin

### Examples

```
# Get the prefix of genotype data
fileBed <- system.file("extdata", "02plinkb", "demo", package = "simer")

## Not run:
# It needs 'plink' software
simer.Data.Geno(fileBed=fileBed)

## End(Not run)
```

---

simer.Data.Impute      *Genotype data imputation*

---

## Description

Impute the missing value within genotype data.

## Usage

```
simer.Data.Impute(  
  fileMVP = NULL,  
  fileBed = NULL,  
  out = NULL,  
  maxLine = 10000,  
  ncpus = 0,  
  verbose = TRUE  
)
```

## Arguments

fileMVP	genotype in MVP format.
fileBed	genotype in PLINK binary format.
out	the name of output file.
maxLine	number of SNPs, only used for saving memory when calculate kinship matrix.
ncpus	the number of threads used, if NULL, (logical core number - 1) is automatically used.
verbose	whether to print detail.

## Details

Build date: May 26, 2021 Last update: Apr 28, 2022

## Value

the function returns files

**<out>.geno.desc** the description file of genotype data.

**<out>.geno.bin** the binary file of genotype data.

**<out>.geno.ind** the genotyped individual file.

**<out>.geno.map** the marker information data file.

## Author(s)

Dong Yin

**Examples**

```
# Get the prefix of genotype data
fileMVP <- system.file("extdata", "02plinkb", "demo", package = "simer")

## Not run:
# It needs 'beagle' software
fileMVPimp <- simer.Data.Impute(fileBed = fileBed)

## End(Not run)
```

---

simer.Data.Json      *Data quality control*

---

**Description**

Make data quality control by JSON file.

**Usage**

```
simer.Data.Json(
  jsonFile,
  hiblupPath = "",
  out = "simer.qc",
  dataQC = TRUE,
  buildModel = TRUE,
  buildIndex = TRUE,
  ncpus = 10,
  verbose = TRUE
)
```

**Arguments**

jsonFile	the path of JSON file.
hiblupPath	the path of HIBLUP software.
out	the prefix of output files.
dataQC	whether to make data quality control.
buildModel	whether to build EBV model.
buildIndex	whether to build Selection Index.
ncpus	the number of threads used, if NULL, (logical core number - 1) is automatically used.
verbose	whether to print detail.

**Details**

Build date: Oct 19, 2020 Last update: Apr 28, 2022



**Value**

the function returns a list containing

**\$genotype** the path of genotype data.

**\$pedigree** the filename of pedigree data.

**\$selection\_index** the selection index for all traits.

**\$breeding\_value\_index** the breeding value index for all traits.

**\$quality\_control\_plan** a list of parameters for data quality control.

**\$breeding\_plan** a list of parameters for genetic evaluation.

**Author(s)**

Dong Yin

**Examples**

```
# Get JSON file
jsonFile <- system.file("extdata", "04breeding_plan", "plan1.json", package = "simer")

## Not run:
# It needs 'plink' and 'hiblup' software
jsonList <- simer.Data.Json(jsonFile = jsonFile)

## End(Not run)
```

---

simer.Data.Kin

*simer.Data.EMMA: To construct EMMA kinship matrix*

---

**Description**

constructing EMMA kinship matrix.

**Usage**

```
simer.Data.Kin(
  fileKin = TRUE,
  fileMVP = "simer",
  out = NULL,
  method = "EMMA",
  sep = "\t",
  threads = 10,
  verbose = TRUE
)
```

### Arguments

fileKin	kinship that represents relationship among individuals, $n * n$ matrix, $n$ is sample size.
fileMVP	prefix for mvp format files.
out	prefix of output file name.
method	only "EMMA" method for now.
sep	separator for Kinship file.
threads	the number of cpu.
verbose	whether to print detail.

### Details

Build date: Apr 19, 2023 Last update: Apr 19, 2023

### Value

Output file: <out>.kin.bin <out>.kin.desc

### Author(s)

Haohao Zhang and Dong Yin

### Examples

```
# Get the prefix of genotype data
fileMVP <- system.file("extdata", "01bigmemory", "demo", package = "simer")

# Check map data
simer.Data.Kin(fileKin = TRUE, fileMVP = fileMVP, out = tempfile("outfile"))
```

---

simer.Data.Map

*simer.Data.Map: To check map file*

---

### Description

checking map file.

### Usage

```
simer.Data.Map(
  map,
  out = "simer",
  cols = 1:5,
  header = TRUE,
  sep = "\t",
  verbose = TRUE
)
```

**Arguments**

map	the name of map file or map object(data.frame or matrix).
out	the name of output file.
cols	selected columns.
header	whether the file contains header.
sep	separator of the file.
verbose	whether to print detail.

**Details**

Build date: Sep 12, 2018 Last update: July 25, 2022

**Value**

Output file: <out>.map

**Author(s)**

Haohao Zhang and Dong Yin

**Examples**

```
# Get map path
mapPath <- system.file("extdata", "01bigmemory", "demo.geno.map", package = "simer")

# Check map data
simer.Data.Map(mapPath, tempfile("outfile"))
```

---

simer.Data.MVP2Bfile    *simer.Data.MVP2Bfile: To transform MVP data to binary format*

---

**Description**

transforming MVP data to binary format.

**Usage**

```
simer.Data.MVP2Bfile(
  bigmat,
  map,
  pheno = NULL,
  out = "simer",
  threads = 10,
  verbose = TRUE
)
```

**Arguments**

bigmat	Genotype in bigmatrix format (0,1,2).
map	the map file.
pheno	the phenotype file.
out	the name of output file.
threads	the number of threads used, if NULL, (logical core number - 1) is automatically used.
verbose	whether to print the reminder.

**Details**

Build date: Sep 12, 2018 Last update: July 20, 2022

**Value**

NULL Output files: .bed, .bim, .fam

**Author(s)**

Haohao Zhang and Dong Yin

**Examples**

```
# Generate bigmat and map
bigmat <- as.big.matrix(matrix(1:6, 3, 2))
map <- generate.map(pop.marker = 3)

# Data converting
simer.Data.MVP2Bfile(bigmat, map, out=tempfile("outfile"))
```

---

simer.Data.MVP2MVP      *Genotype data conversion*

---

**Description**

Convert genotype data from MVP format to MVP format.

**Usage**

```
simer.Data.MVP2MVP(fileMVP, genoType = "char", out = "simer", verbose = TRUE)
```

**Arguments**

fileMVP	the prefix of MVP file.
genoType	type parameter in bigmemory data. The default is 'char', it is highly recommended *NOT* to modify this parameter.
out	the prefix of output files.
verbose	whether to print detail.

**Details**

Build date: May 26, 2021 Last update: Apr 28, 2022

**Value**

the function returns files

**<out>.geno.desc** the description file of genotype data.

**<out>.geno.bin** the binary file of genotype data.

**<out>.geno.ind** the genotyped individual file.

**<out>.geno.map** the marker information data file.

**Author(s)**

Dong Yin

**Examples**

```
# Get the prefix of genotype data
fileMVP <- system.file("extdata", "01bigmemory", "demo", package = "simer")

# Convert genotype data from MVP to MVP
simer.Data.MVP2MVP(fileMVP, out = tempfile("outfile"))
```

---

simer.Data.Ped

*Pedigree data quality control*

---

**Description**

Data quality control for pedigree data.

**Usage**

```

simer.Data.Ped(
  filePed,
  fileMVP = NULL,
  out = NULL,
  standardID = FALSE,
  fileSir = NULL,
  fileDam = NULL,
  exclThres = 0.1,
  assignThres = 0.05,
  header = TRUE,
  sep = "\t",
  ncpus = 0,
  verbose = TRUE
)

```

**Arguments**

filePed	the filename of pedigree need correcting.
fileMVP	genotype in MVP format.
out	the prefix of output file.
standardID	whether kid id is 15-character standard.
fileSir	the filename of candidate sires.
fileDam	the filename of candidate dams.
exclThres	if conflict ratio is more than exclThres, exclude this parent.
assignThres	if conflict ratio is less than assignThres, assign this parent to the individual.
header	whether the file contains header.
sep	separator of the file.
ncpus	the number of threads used, if NULL, (logical core number - 1) is automatically used.
verbose	whether to print detail.

**Details**

Build date: May 6, 2021 Last update: Apr 28, 2022

**Value**

the function returns files

**<out>.ped.report** the report file containing correction condition.

**<out>.ped.error** the file containing pedigree error.

**<out>.ped** the pedigree file after correction.

**Author(s)**

Lilin Yin and Dong Yin

**Examples**

```
# Get the filename of pedigree data
filePed <- system.file("extdata", "05others", "pedigree.txt", package = "simer")

# Get the prefix of genotype data
fileMVP <- system.file("extdata", "01bigmemory", "demo", package = "simer")

# Run pedigree correction
simer.Data.Ped(filePed = filePed, fileMVP = fileMVP, out = tempfile("outfile"))
```

---

simer.Data.Pheno	<i>Phenotype data quality control</i>
------------------	---------------------------------------

---

**Description**

Data quality control for phenotype data.

**Usage**

```
simer.Data.Pheno(
  filePhe = NULL,
  filePed = NULL,
  out = NULL,
  planPhe = NULL,
  pheCols = NULL,
  header = TRUE,
  sep = "\t",
  missing = c(NA, "NA", "Na", ".", "-", "NAN", "nan", "na", "N/A", "n/a", "<NA>", "",
    "-9", 9999),
  verbose = TRUE
)
```

**Arguments**

filePhe	the phenotype files, it can be a vector.
filePed	the pedigree files, it can be a vector.
out	the prefix of output file.
planPhe	the plans for phenotype quality control.
pheCols	the column needing extracting.
header	the header of file.
sep	the separator of file.
missing	the missing value.
verbose	whether to print detail.

**Details**

Build date: June 13, 2021 Last update: Apr 28, 2022

**Value**

the function returns files

**<out>.phe** the phenotype file after correction.

**Author(s)**

Haohao Zhang and Dong Yin

**Examples**

```
# Get the filename of phenotype data
filePhe <- system.file("extdata", "05others", "phenotype.txt", package = "simer")

# Run phenotype correction
simer.Data.Pheno(filePhe = filePhe, out = tempfile("outfile"))
```

---

simer.Data.SELIND      *Selection index construction*

---

**Description**

The function of General Selection Index.

**Usage**

```
simer.Data.SELIND(jsonList = NULL, hiblupPath = "", ncpus = 10, verbose = TRUE)
```

**Arguments**

jsonList	the list of selection index construction parameters.
hiblupPath	the path of HIBLUP software.
ncpus	the number of threads used, if NULL, (logical core number - 1) is automatically used.
verbose	whether to print detail.

**Details**

Build date: Aug 26, 2021 Last update: Apr 28, 2022



**Value**

the function returns a list containing

**\$genotype** the path of genotype data.

**\$pedigree** the filename of pedigree data.

**\$selection\_index** the selection index for all traits.

**\$breeding\_value\_index** the breeding value index for all traits.

**\$quality\_control\_plan** a list of parameters for data quality control.

**\$breeding\_plan** a list of parameters for genetic evaluation.

**Author(s)**

Dong Yin

**References**

Y. S. Chen, Z. L. Sheng (1988) The Theory of General Selection Index. Genetic Report, 15(3): P185-P190

**Examples**

```
# Read JSON file
jsonFile <- system.file("extdata", "04breeding_plan", "plan1.json", package = "simer")
jsonList <- jsonlite::fromJSON(txt = jsonFile, simplifyVector = FALSE)

## Not run:
# It needs 'hiblup' software
jsonList <- simer.Data.SELIND(jsonList = jsonList)

## End(Not run)
```

---

simer.Version

*Simer version*

---

**Description**

Print simer version.

**Usage**

```
simer.Version(width = 60, verbose = TRUE)
```

**Arguments**

width            the width of the message.

verbose         whether to print detail.

**Details**

Build date: Aug 30, 2017 Last update: Apr 30, 2022

**Value**

version number.

**Author(s)**

Dong Yin, Lilin Yin, Haohao Zhang, and Xiaolei Liu

**Examples**

```
simer.Version()
```

---

write.file

*File writing*

---

**Description**

Write files of Simer.

**Usage**

```
write.file(SP)
```

**Arguments**

SP                    a list of all simulation parameters.

**Details**

Build date: Jan 7, 2019 Last update: Apr 30, 2022

**Value**

none.

**Author(s)**

Dong Yin

**Examples**

```
outpath <- tempdir()
SP <- param.simer(out = "simer")
SP <- simer(SP)
SP$global$outpath <- outpath
write.file(SP)
unlink(file.path(outpath, "180_Simer_Data_numeric"), recursive = TRUE)
```

# Index

## \* datasets

- pop.geno, [43](#)
- pop.map, [43](#)
  
- annotation, [3](#)
  
- build.cov, [4](#)
  
- cal.eff, [5](#)
- checkEnv, [6](#)
  
- generate.map, [7](#)
- generate.pop, [8](#)
- geno.cvt1, [9](#)
- geno.cvt2, [9](#)
- genotype, [10](#)
- getfam, [11](#)
- GxG.network, [12](#)
  
- IndPerGen, [13](#)
  
- logging.initialize, [14](#)
- logging.log, [14](#)
- logging.print, [16](#)
  
- mate, [17](#)
- mate.2waycro, [18](#)
- mate.3waycro, [19](#)
- mate.4waycro, [20](#)
- mate assort, [22](#)
- mate.backcro, [23](#)
- mate.clone, [24](#)
- mate.dh, [26](#)
- mate.disassort, [27](#)
- mate.randexself, [28](#)
- mate.randmate, [30](#)
- mate.selfpol, [31](#)
- mate.userped, [32](#)
  
- param.annot, [34](#)
- param.geno, [35](#)
- param.global, [36](#)
- param.pheno, [37](#)
- param.reprod, [38](#)
- param.sel, [39](#)
- param.simer, [40](#)
- phenotype, [41](#)
- pop.geno, [43](#)
- pop.map, [43](#)
  
- remove\_bigmatrix, [44](#)
- reproduces, [45](#)
  
- selects, [46](#)
- simer, [47](#)
- simer.Data, [48](#)
- simer.Data.Bfile2MVP, [49](#)
- simer.Data.CHIBLUP, [50](#)
- simer.Data.Env, [52](#)
- simer.Data.Geno, [53](#)
- simer.Data.Impute, [55](#)
- simer.Data.Json, [56](#)
- simer.Data.Kin, [57](#)
- simer.Data.Map, [58](#)
- simer.Data.MVP2Bfile, [59](#)
- simer.Data.MVP2MVP, [60](#)
- simer.Data.Ped, [61](#)
- simer.Data.Pheno, [63](#)
- simer.Data.SELIND, [64](#)
- simer.Version, [65](#)
  
- write.file, [66](#)