Breeding Scheme Language

Shiori Yabe, Hiroyoshi Iwata, and Jean-Luc Jannink

13 February 2017

INDEX

Description

Package: BreedingSchemeLanguage

Type: Package

Title: Describe and simulate breeding schemes

Version: 1.0

Date: 2016-03-08

Author: Shiori Yabe, Hiroyoshi Iwata, and Jean-Luc Jannink

Maintainer: Shiori Yabe <syabe@affrc.go.jp>

Description: Users can simulate breeding schemes by using functions that imitate events in breeding.

License: GPL-3

Depends: R (>= 3.0.0), Rcpp (>= 0.11.0), snowfall

Imports:

   ggplot2,

   rrBLUP

LinkingTo: Rcpp

LazyLoad: yes

NeedsCompilation: yes

RoxygenNote: 5.0.1

Introduction

This documentation describes how to start and use the R package "BreedingSchemeLanguage". BreedingSchemeLanguage (BSL) was developed for breeders to conduct breeding simulations in a simple and flexible system. Users can use BSL under the R environment.

The BSL uses the coalescent-based whole genome simulator "GENOME" (Liang et al., 2006) to simulate a founder population. It is also possible to upload haplotypes of a founder population in hapmap format.

The BSL uses the R package "rrBLUP" (Endelman, 2011) in the function conducting genomic prediction.

Parallel processing of simulations depends on the multi-core R package "snow fall".

If you use the program, the appropriate citation is:

_____.

Reference for GENOME:

Liang L., Zöllner S., Abecasis G.R. (2006) GENOME: a rapid coalescent-based whole genome simulator.

Reference for rrBLUP:

Endelman, J.B. 2011. Ridge regression and other kernels for genomic selection with R package rrBLUP. The Plant Genome 4: 250-255.

Reference for snowfall:

snow(Simple Network of Workstations):

http://cran.r-project.org/src/contrib/Descriptions/snow.html

Required environment

Downloading and installing R is necessary to use the package. Users can download R on the web page:

The Comprehensive R Archive Network (CRAN)   https://cran.r-project.org

The BSL package needs Rtools (for Windows) or Xcode (for Mac OS X). These tools are used to install the BSL package from GitHub (https://github.com) and to compile C++ code on your PC. Compiling C++ code is automatically done when you install the BSL package.

Rtools: https://cran.r-project.org/bin/windows/Rtools/

Xcode: Mac App Store (You need your "Apple ID".)

Install BreedingSchemeLanguage

On your R screen, you can write the code below:

install.packages("devtools")
devtools::install_github("syabe/BreedingSchemeLanguage")

After installing the Breeding Scheme Language package, the only command needed to use it is:

library(BreedingSchemeLanguage)

Now, you can use the "Breeding Scheme Language" on your PC!

Describe breeding scheme

Help

You can call help pages two ways:

?FUNCTION

help(FUNCTION)

NOTE: FUNCTION should be replaced with the name of function you hope to ask.

Functions

- defineSpecies(loadData = NULL , importFounderHap = NULL, saveDataFileName = "previousData", nSim = 1, nCore = 1, nChr = 7, lengthChr = 150, effPopSize = 100, nMarkers = 1000, nQTL = 50, propDomi = 0, nEpiLoci = 0 , domModel = "HetHom")
- initializePopulation(simEnv, nInd = 100, gVariance = 1)
- phenotype(simEnv, errorVar = 1, popID = NULL)
- genotype(simEnv, popID = NULL)
- predictBreedVal(simEnv, popID = NULL, trainingPopID = NULL)
- select(simEnv, nSelect = 40, random = F, popID = NULL)
- cross(simEnv, nProgeny = 100, equalContribution = F, popID = NULL)
- selfFertilize(simEnv, nProgenyPerInd = 1, popID = NULL)
- doubledHaploid(simEnv, nProgeny = 100, popID = NULL)
- plotData(simEnv, ymax = NULL, add = F, addDataFileName = "plotData")
- outputResults(simEnv, summarize = T, directory = NULL, saveDataFileName = "BSLoutput")

The parameters in parenthesis represent the default parameter values.

For all functions other than defineSpecies, the first parameter is "simEnv", which is an R environment that is created by defineSpecies and that contains objects holding the simulated data.

Functions to initiate simulations

The function "defineSpecies" uses up to five parameters to define the overall simulation settings (i.e., loadData, importFounderHap, saveDataFileName, nSim, and nCore) and then eight parameters to define the genetic architecture of the species.

- loadData: If null, simulate new data, else a file name should be given and the function will attempt to load data from a file previously created by setting saveDataFileName to that file name.

- importFounderHap: If "loadData" is not null, this parameter has no effect. If "loadData" is null and "importFounderHap" contains a file name the function will attempt to load founder haplotype data from a hapmap format file. See details below.

- saveDataFileName: Name under which to save newly-simulated data. No file suffix needs to be given to this name

- nSim: Number of repeats of the simulation

- nCore: Simulations can be processed in parallel if the number given here is greater than 1

- nChr: Number of chromosomes of the species

- lengthChr: Length of each chromosome in cM (all chromosomes have the same length)

- effPopSize: Effective population size of the base population. An idealized population is assumed leading up to the beginning of the simulation

- nMarkers: Number of markers observable for making predictions

- nQTL: Number of genetic effects controlling the target trait. If there is no epistasis, this is also the number of QTL. Under epistasis, the expected number of causal loci will be nQTL times (nEpiLoci + 1). The nEpiLoci parameter is defined below

- probDomi: Probability of a QTL locus exhibiting dominance

- nEpiLoci: Expected number of interacting loci contributing to each effect

- domModel: The two options are "HetHom" (the default) a dominance model that gives opposite effects to heterozygotes versus homozygotes, or "Partial", a dominance model that requires specifying a degree of dominance of the derived allele.

ImportFounderHap details. This feature uses a hapmap format with loci in rows, eleven (11) locus annotation columns, and then one column for each haplotype. Annotation columns as follows. Column 1: locus names. Column 2: possible genotypes (e.g. "A/G"). Column 3: chromosome number. Column 4: locus position in cM. If users specify QTL information, it is as follows. Column 7: the ID of the effect (more than one locus can contribute multiplicatively to one effect if all loci have the same effect ID). Column 8: action type (0 for additive, 1 for dominance, in between for partial dominance). Column 9: the effect of the QTL.

The function "initializePopulation" creates a founder population for breeding:
- nPop: Size of the founder population
- gVariance: Genetic variance in the founder population

Breeding functions

The function "phenotype" causes a phenotyping trial to be run:
- errorVar: Environmental error variance of the trial
- popID: Population ID to be phenotyped (Default is the last population created)

The function "genotype" causes marker genotypes to become available for the specified population.
- popID: Population ID to be genotyped (Default is all individuals in the breeding scheme)

The function "predictBreedVal" performs genomic prediction using phenotype and genotype data previously generated in the breeding scheme:
- popID: Population ID to be predicted (Default is the latest population created)
- trainingPopID: A vector of population IDs to be used to train a prediction model (Default is all populations having phenotype data)

The function "select" conducts selection in the defined population:

- nSelect: Number of individuals to select
- popID: Population ID to be selected (Default is the last population created when random=T. When random=F, default is the last evaluated or predicted population)
- random: If T, individuals are selected at random. If F, the selection criterion depends on previous breeding activities: if the last activity was "phenotype", then selection will be on phenotypes; if the last activity was "predictBreedVal", then selection will be on predicted breeding values.

Mating functions

The "cross" function conducts random mating among parents.
- nProgeny: Number of progeny to generate by random mating
- equalContribution: If TRUE, all individuals are used the same number of times as parents. The number of progeny should be larger than the number of parents in popID. This setting increases the effective population size for a given number of progeny. If FALSE, pairs of parents are chosen at random for each progeny.
- popID: Population ID to be used as parents (Default is the last population created)
- popID2: ID of a second population to be used as parents. If not null, the mating design will force one parent to be taken from popID and one parent from popID2 for each cross. Thus, popID2 can be used to simulate reciprocal recurrent selection.

The "selfFertilize" function implements inbreeding.
- nProgeny: Number of selfed progeny to make. Parent in the population are used as evenly as possible to make the progeny (i.e., each parent is used minimally floor(nProgeny / nParents) times with remaining progeny allocated at random.
- popID: Population ID to be used as parents (Default is the last population created)

The "doubledHaploid" function makes doubled haploids progeny.
- nProgeny: Number of doubled haploid progeny to make, allocated as in selfFertilize.
- popID: Population ID to be used as parents (Default is the last population created)

Result functions

The function "plotData" draws a figure of the genotypic value through generations of breeding. The figure shows population means of each simulation replication and the mean value over repeated simulations (given by the nSim parameter in "defineSpecies" function).

- ymax: Maximum genotypic value on the y-axis of the figure.
- add: If TRUE results will be added to previous data obtained from the "addDataFileName" file (see below)
- addDataFileName: String giving the name of a file from which to obtain data from a previous simulation. Also, results used in making this plot will be saved to that data file.   No file suffix needs to be given to this name
- popID: Optional vector of population IDs allowing the user to specify which populations to plot. If not given, the default will be to plot each population's mean genotypic value when it was first created by "cross", "selfFertilize", or "doubledHaploid" and before any lines were selected out of it.

The function "outputResults" saves the results.

- summarize: If TRUE results averaged over simulation replications will be saved. If FALSE, all data from breeding simulations will be saved
- directory: If NULL data will be saved in the R working directory. If a string giving the name of a directory, data will be saved there. When summarize = F, extensive data is saved so that dedicating a directory to it may be wise
- saveDataFileName: String giving the name of a file in which simulation results are saved.   No file suffix needs to be given to this name

Population ID

Initial population ID = 0

Population ID will be incremented by these functions:

1. select
2. cross

3. selfFertilize

4. doubledHaploid

Be careful that:

select() creates a new population that is a subset of the candidate population

Example 1
"Phenotypic selection & genomic selection"

Code:

1. Phenotypic mass selection (Fig. a)

```
simEnv <- defineSpecies(nSim = 5)
initializePopulation(simEnv)
phenotype(simEnv)
select(simEnv)
cross(simEnv)
phenotype(simEnv)
select(simEnv)
cross(simEnv)
phenotype(simEnv)
select(simEnv)
cross(simEnv)
plotData(simEnv)
```

2. Genotypes aiding breeding value estimation on existing phenotypic data (Fig. b)

```
simEnv <- defineSpecies(loadData="previousData")
initializePopulation(simEnv)
phenotype(simEnv)
genotype(simEnv)
predictBreedVal(simEnv)
select(simEnv)
cross(simEnv)
phenotype(simEnv)
genotype(simEnv)
predictBreedVal(simEnv)
select(simEnv)
```

```
cross(simEnv)
phenotype(simEnv)
genotype(simEnv)
predictBreedVal(simEnv)
select(simEnv)
cross(simEnv)
plotData(simEnv, add=T)
```

(a) / (b) Selection scheme diagram
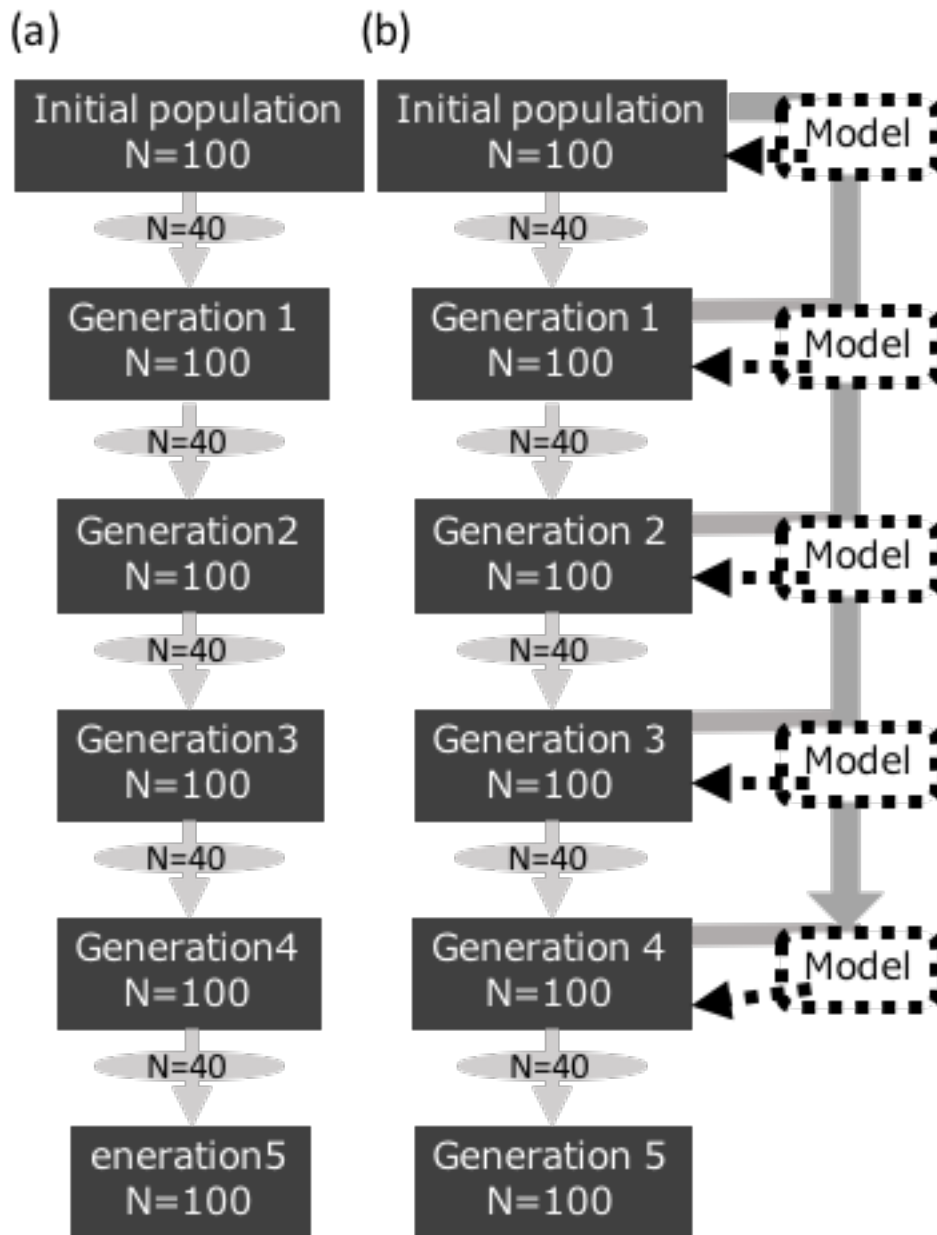
Simulation result:

Scheme 1 is phenotypic selection, and Scheme 2 is genomic selection.

Bold lines represent the mean values of simulation trials (in this case, 5 generations of selection),

and each thin line is the result of a simulation trial.