genomicSimulation: fast R functions for stochastic simulation of breeding programs

Kira Villiers[1,*], Eric Dinglasan[1], Ben J. Hayes[1], and Kai P. Voss-Fels[1,2]

[1]Queensland Alliance for Agriculture and Food Innovation, The University of Queensland, St Lucia, QLD, Australia
[2]Department of Grapevine Breeding, Hochschule Geisenheim University, Geisenheim, Germany.
[*]To whom correspondence should be addressed.
Contact address: k.villiers@uqconnect.edu.au

Abstract

Simulation tools are key to designing and optimising breeding programs that are many-year, high-effort endeavours. Tools that operate on real genotypes and integrate easily with other analysis software are needed for users to integrate simulated data into their analysis and decision-making processes. This paper presents genomicSimulation, a fast and flexible tool for the stochastic simulation of crossing and selection on real genotypes. It is fully written in C for high execution speeds, has minimal dependencies, and is available as an R package for integration with R's broad range of analysis and visualisation tools. Comparisons of a simulated recreation of a breeding program to the real data shows that the tool's simulated offspring correctly show key population features. Both versions of genomicSimulation are freely available on GitHub: The R package version at https://github.com/vllrs/genomicSimulation/ and the C library version at https://github.com/vllrs/genomicSimulationC

1 Introduction

When real breeding schemes and genetic improvement programs are high-effort, many-year undertakings, simulation tools offer valuable opportunity to judge feasibility or balance tradeoffs in design so as to get the most out of the real programs (Li *et al.*, 2012). Before the last few years, the available genetic simulation tools as described by Sun *et al.* (2011) tended to be standalone tools with limited opportunities to incorporate external analysis tools, and/or tools that generated artificial genomes from parameters rather than using real data. While this is helpful for statistical analyses or broad evaluations of breeding strategies (for example, Hickey *et al.* (2014)), one of the key problems in optimising breeding scheme design is parent selection, which is highly dependent on the real parents that are available (Sun *et al.*, 2011; Li *et al.*, 2012; Bernardo, 2020). Bernardo (2020) makes a strong case for the need for simulation tools focused on operating on real genotypes data, real genome maps, and real phenotypic

performance data.

This paper introduces genomicSimulation, a fast, flexible R package for the stochastic simulation of breeding programs on real genotype data. It was built to provide high-speed crossing simulation capabilities that can be easily integrated with R's wide range of visualisation and genetic analysis tools. genomicSimulation loads and runs based on user-provided marker maps, founder genotypes, and (optional) lists of allele effect values, so can easily simulate breeding schemes on real maps and candidate founders. It simulates meiosis without mutation on alleles at discrete positions provided in the genome map. Phase of resulting genotypes is tracked. genomicSimulation works as a scripting tool, with functions for performing targeted crosses, random crosses, doubling haploids and selfing. genomicSimulation's inbuilt genotypic value calculator uses an additive model of marker effects. Alternatively, users can script their own custom selection methods in R. The package has no dependencies beyond C standard libraries. All core functionality is written in C in order to achieve high execution speeds.

For even faster simulations, genomicSimulation's underlying C library (in itself a fully-functional simulation tool) is also available. It is distinguished from the R package only by the lack of default parameters and lack of R vector data output options.

The package was originally developed for use on self-pollinated crop species, but its flexible set of crossing operations allow it to be used more broadly. genomicSimulation has full documentation, use guides, and is in ongoing development.

2 Methods

The first step in using genomicSimulation is a call to the setup function (load.data). Two data files are required to set up the tool: a matrix file contain the alleles of at least one founder genotype at a set of discrete positions, and a linkage map to situate those positions in the model of the genome. If the phase at heterozygous positions in the founder genotypes is not known, it is randomised on load. Alleles can be any non-space character, which allows for the package to be used on abstracted Mendelian genes as well as SNPs.

The simulation stores every simulated genotype as a sequence of characters in memory.

To create a cross, the simulation generates a gamete independently from each of the two parents. No distinction is made between male and female parent. Karlin and Liberman (1979)'s count-location method is used to simulate meiosis. Specifically, the following process is followed: First, the number of crossovers to occur in each chromosome is drawn from a Poisson distribution whose parameter is the distance in Morgans between the first and last positions tracked on that chromosome (Figure 1a). The positions of those crossovers along the chromosome are then drawn from a uniform distribution (Figure 1b). Finally, a random logical value (0 or 1) is drawn to choose which of the two possible gametes to use (Figure 1c).

The count-location method of simulating meiosis ignores crossover interference.

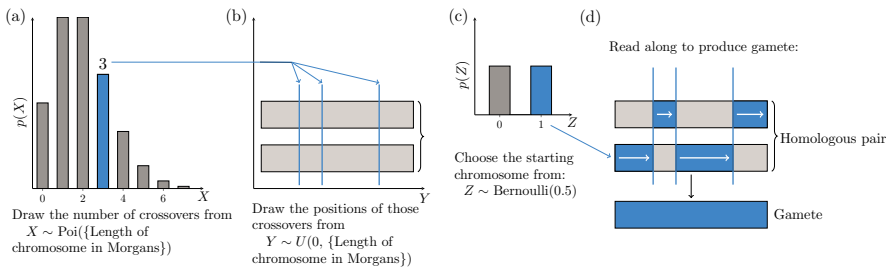

Figure 1: Meiosis simulation procedure in genomicSimulation uses a count-location strategy. The following steps are performed for every homologous pair of chromosomes. (Blue coloration shows a sample procedure.) (a) First, the number of crossovers is drawn from a Poisson distribution with expectation matching the length of the chromosome in Morgans. (b) Next, the positions of each of those crossovers are drawn uniformly across the length of the chromosome. (c) A final random draw determines which of the two resultant gametes is chosen. (d) The gamete is created by reading along the chosen starting chromosome, swapping to the other of the pair whenever a crossover point is encountered.

The specific distributions chosen add the additional assumptions that the number of crossovers is proportional to the chromosome length, and that crossovers are uniformly distributed along the chromosome. This choice of distributions and assumptions is shared with existing simulation tool Plabsoft (Maurer *et al.*, 2007).

During setup, there is also the option to load an input file containing the numeric effects of particular alleles. This enables the use of the internal breeding value calculator. Breeding values are calculated using an additive model of effects: the breeding value of a given genotype is the sum across all marker positions of each marker's two alleles' effect values (if such allele effects are loaded).

2.1 Features

At any time after the initial setup command, more external genotypes can be loaded using the command load.more.genotypes(), and the simulation's optional stored set of marker effects can be substituted for another with load.different.effects(). With these features, users can simulate the introduction of new genetic material into an ongoing breeding program, calculate genotypic values for multiple traits with separate sets of marker effects, and/or manually simulate environmental fluctuations across years and/or locations by using different sets of marker effects.

A range of functions are available for simulating the production of offspring. These include cross.randomly(), self.n.times(), make.doubled.haploids(), and cross.all.pairs(). To carry out specific crosses or crossing plans, users can call cross.combinations() with vectors laying out the parents to cross.

Every genotype loaded or produced in genomicSimulation is allocated to a group. Every function in the R package returns 0 if it does not create or modify any groups, or the group identifier (a positive integer) of whatever group was populated.

Mean execution time (sec)	Load 50 genotypes of 5000 SNPs	Perform 10^5 random crosses	Get resulting genotypes	Get resulting breeding values
C library	0.66	4.26	67.99	70.13
R package in RStudio	0.70	2.58	13.30	81.09

Table 1: Average execution time (in seconds) across 6 repeats of tasks in genomicSimulation v0.2, running on a consumer-model laptop with 8GB RAM and Intel i5-7200U CPU @ 2.50GHz. The tasks timed are: (1) loading 50 genotypes of 5000 SNPs, (2) perform 10^5 random crosses between those genotypes with one offspring each, (3) save the 10^5 genotypes from task 2 to a file (for the C version) or an R dataframe (for the R version), and (4) calculate and save the breeding values of the 10^5 genotypes from task 2 to a file (for the C version) or an R dataframe (for the R version). Note that the R version of the tool also shares the C library's functionality for saving output to files. The time taken to save output to files is comparable across R and C versions.

The function select.by.gebv() performs truncation selection on breeding values calculated by the internal breeding value calculator. An interface for custom selection methods is also offered: this involves using data from see.group.data() and some R scripting to identify the best individuals, followed by passing those individuals' indexes to make.group() to move them into a new 'selected' group. Further options for group manipulation include combine.groups(), break.group.into.families(), break.group.into.halfsib.families(), and a range of functions to randomly partition a group (perhaps to demarcate male/female offspring, or to create subgroups of specific sizes for complex breeding project designs).

Simulated data can be saved to tab-separated text files or pulled into the user's R environment as vectors. The current data types available as output are genotypes, haplotypes, allele counts, immediate parents, full known ancestry, breeding values (calculated using the internal additive trait), and local breeding values over certain blocks/subsets of positions (calculated using the internal additive trait).

More in-depth description of features and usage examples are provided in the R package vignette, R documentation, C library guides, and C library documentation.

2.2 Run Speed

Table 1 shows some average runtimes of simulation tasks. Most constraints on speed are imposed by file read/write operations, so scripts that minimise the number of outputs saved will run faster.

3 Results

A structured durum wheat population from Alahmad *et al.* (2019) was simulated using genomicSimulation to assess the simulation. The structured population was developed through the nested association mapping (NAM) design presented in Figure 2a. This design was simulated with genomicSimulation using the cross.combinations() and self.n.times() commands to produce 10 fam-

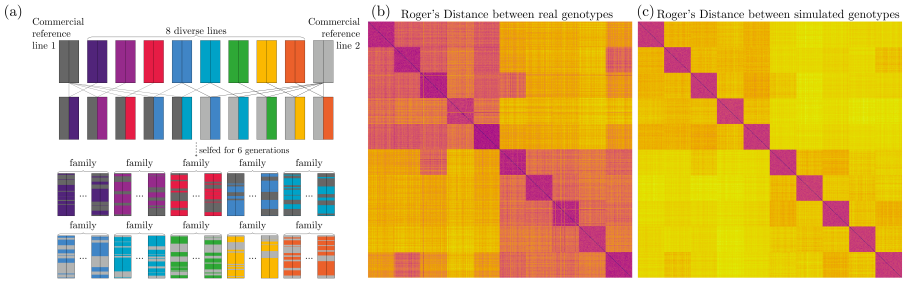

Figure 2: (a) The crossing plan of a structured population developed by Alahmad *et al.* (2019). The matrices of Roger's Genetic Distance between all (b) real imputed genotypes, (c) simulated genotypes, of the final-generation offspring resulting from that crossing design.

ilies of 100 genotypes in the final generation. The simulated genotypes were then compared to the real imputed genotypes. R packages SelectionTools and ComplexHeatmaps were used to produce the heatmaps shown in Figure 2b-c.

The population structure of ten family relationships and two shared-elite-parent relationships is clearly visible in both heatmaps. The distributions of genetic distances also show the same profiles (Supplementary Figure 2). However, the real population shows higher variation within families, as shown in the wider range of base colours in the heatmaps and in Supplementary Figure 1. The real population was developed with selection for several traits during line development, including height variation, reasonable flowering time and early vigour (S Alahmad 2021, personal communication, 27 January). These selection pressures could not be reasonably mimicked in simulation, and so may well account for the differences in level of variance observed.

4 Conclusion

Simulation is a valuable tool to inform the choices needed to carry out a breeding program. Various simulation tools in this area exist. Some, such as QU-GENE (Podlich and Cooper, 1998) and ADAM-Plant (Liu *et al.*, 2019), are standalone programs, but an increasing number are R packages. These include Plabsoft (Maurer *et al.*, 2007), BSL (Yabe *et al.*, 2017), MoBPS (Pook *et al.*, 2020), and AlphaSimR (Gaynor *et al.*, 2020). genomicSimulation joins these ranks as a simple, flexible, freely-available tool designed for simulating from real genotypes and genome maps, and analysing the results with R's wide range of specialised libraries. Its approachable syntax, easy interoperability with other R libraries, and automatically-fast execution (thanks to an underlying C library implementation) make it a useful complement to existing simulation tools. The package should be straightforward to install thanks to its lack of dependencies beyond C standard libraries. Documentation and use guides are available. Development is ongoing.

Acknowledgements

The authors thank Samir Alahmad and Lee Hickey for providing access to the wheat NAM population dataset.

Funding

This work was carried out under a University of Queensland Summer Student Project.

References

Alahmad, S., El Hassouni, K., Bassi, F.M., Dinglasan, E., Youssef, C., Quarry, G., Aksoy, A., Mazzucotelli, E., Juhász, A., Able, J.A., Christopher, J., Voss-Fels, K.P., Hickey, L.T. (2019), A Major Root Architecture QTL Responding to Water Limitation in Durum Wheat, *Frontiers in Plant Science* **10**(436).

Bernardo, R. (2020), Reinventing quantitative genetics for plant breeding: something old, something new, something borrowed, something BLUE, *Heredity* **125**(6): 375-385.

Gaynor, R.C., Gorjanc, G., Hickey, J.M. (2020), AlphaSimR: An R-package for Breeding Program Simulations, *bioRxiv: 2020.2008.2010.245167.*

Hickey, J.M., Dreisigacker, S., Crossa, J., Hearne, S., Babu, R., Prasanna, B.M., Grondona, M., Zambelli, A., Windhausen, V.S., Mathews, K., Gorjanc, G. (2014) Evaluation of genomic selection training population designs and genotyping strategies in plant breeding programs using simulation, *Crop Science* **54**(4), 1476-1488.

Karlin, S. and U. Liberman (1979), A natural class of multilocus recombination processes and related measures of crossover interference, *Advances in Applied Probability* **11**(3), 479-501.

Li, X., Zhu, C., Wang, J., Yu, J. (2012), Computer Simulation in Plant Breeding, *Advances in Agronomy* **116**, 219-264.

Liu, H., Tessema, B. B., Jensen, J., Cericola, F., Andersen, J.R., Sørensen, A.C. (2019) ADAM-Plant: A software for stochastic simulations of plant breeding from molecular to phenotypic level and from simple selection to complex speed breeding programs, *Frontiers in Plant Science* **9**(1926).

Maurer, H.P., Melchinger, A.E., Frisch, M. (2007) Population genetic simulation and data analysis with Plabsoft, *Euphytica* **161**(1-2), 133-139.

Podlich, D.W. and M. Cooper (1998) QU-GENE: a simulation platform for quantitative analysis of genetic models, *Bioinformatics* **14**(7), 632-653.

Pook, T., Schlather, M., Simianer, H. (2020) MoBPS - Modular Breeding Program Simulator, *G3 (Bethesda, Md.)* **10**(6), 1915-1918.

Sun, X., Peng, T., Mumm, R.H. (2011) The role and basics of computer simulation in support of critical decisions in plant breeding, *Molecular Breeding* **28**(4), 421-436.

Yabe, S., Iwata, H., Jannink, J. (2017) A Simple Package to Script and Simulate Breeding Schemes: The Breeding Scheme Language, *Crop Science*, **57**, 1347-1354.