SOFTWARE AND DATA RESOURCES

MoBPS - Modular Breeding Program Simulator

Torsten Pook,[*,†,1] **Martin Schlather,**[†,‡] **and Henner Simianer**[*,†]
[*]Department of Animal Sciences, Animal Breeding and Genetics Group, [†]Center for Integrated Breeding Research, University of Goettingen, 37075 Goettingen, Germany, and [‡]Stochastics and Its Applications Group, University of Mannheim, 68131 Mannheim, Germany
ORCID ID: 0000-0001-7874-8500 (T.P.)

ABSTRACT The R-package MoBPS provides a computationally efficient and flexible framework to simulate complex breeding programs and compare their economic and genetic impact. Simulations are performed on the base of individuals. MoBPS utilizes a highly efficient implementation with bit-wise data storage and matrix multiplications from the associated R-package miraculix allowing to handle large scale populations. Individual haplotypes are not stored but instead automatically derived based on points of recombination and mutations. The modular structure of MoBPS allows to combine rather coarse simulations, as needed to generate founder populations, with a very detailed modeling of todays' complex breeding programs, making use of all available biotechnologies. MoBPS provides pre-implemented functions for common breeding practices such as optimum genetic contributions and single-step GBLUP but also allows the user to replace certain steps with personalized and/or self-written solutions.

KEYWORDS
R-package
simulation
breeding
breeding program
resource management
population genetics

Breeding programs aim at improving the genetic properties of livestock and crop populations with respect to productivity, fitness and adaptation. Progress toward the target is limited by the available resources, but also negative effects, such as inbreeding depression or health issues, have to be avoided or at least controlled. Hence, the allocation of resources in a breeding program is a complex optimization problem. Additionally, population history, such as fluctuating population sizes and selection pressures, has an impact on the current genomic architecture and thus the potential for future improvement.

Over the years a variety of simulation tools have been developed to assist breeders to evaluate and optimize their breeding programs. A general problem of simulation studies is that the underlying genomic processes are highly complex and have to be simplified for modeling. In addition, users often have rather different objectives in mind when setting up their simulation studies. Since tools often do not provide the necessary flexibility to execute the specific breeding actions and/or it is not possible to export all necessary outputs, this commonly leads to the use of self-developed solutions that tend to be more error-prone, less sophisticated and computationally inefficient. The functionality of existing software for the simulation of breeding programs ranges from cohort based deterministic simulation that relies on expected gains like ZPLAN+ (Täubert *et al.* 2010) to applications on the base of the stochastic simulation of single individuals such as QMSim (Sargolzaei and Schenkel 2009) and AlphaSim (Faux *et al.* 2016). The functionality of each of these tools highly depends on the intended use. ZPLAN+ (Täubert *et al.* 2010) focuses on the economic impact from a macro-perspective. Since analytic formulas for cohorts are required, it has limitations when simulating complex mating schemes or when focusing on other quantities than genetic or economic gain. QMSim (Sargolzaei and Schenkel 2009) is able to simulate each individual meiosis but is lacking the flexibility in terms of design options in the breeding program, as it is mainly intended for use in population genetics. On the contrary, AlphaSim (Faux *et al.* 2016) provides a lot of flexibility in term of the design of the breeding program, especially for plant breeding and when the number of cohorts in the breeding program is small. However, AlphaSim lacks the efficiency to simulate complex and large scale populations (*e.g.*, as genotypes of all individuals are stored) and does not allow for the export of all potentially relevant results. The interested reader is referred to Sun *et al.* (2011) for an extended review on different simulators used in plant breeding.

Our goal was to develop a tool that combines the simulation of a historical population and the evaluation of a subsequent complex

Copyright © 2020 Pook *et al.*
doi: https://doi.org/10.1534/g3.120.401193
Manuscript received February 28, 2020; accepted for publication March 27, 2020; published Early Online March 30, 2020.
This is an open-access article distributed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.
[1]Corresponding author: University of Goettingen, Department of Animal Sciences, Center for Integrated Breeding Research, Animal Breeding and Genetics Group, Albrecht-Thaer-Weg 3, 37075 Goettingen, Germany. E-mail: torsten.pook@uni-goettingen.de

Downloaded from https://academic.oup.com/g3journal/article/10/6/1915/6026363 by Universitätsbibliothek Mannheim user on 04 April 2023

breeding program in a computationally efficient way. The Modular Breeding Program Simulator (MoBPS) is not only flexible in terms of parameters and design of breeding programs, but also allows the user to replace standard procedures of the package with own ones.

METHODS

Simulations in MoBPS are ultimately based on the simulation of single individuals. In principle, this allows the user to control each singular mating and modify recombination or mutation rates for the respective meiosis. However, breeding programs in MoBPS can still be constructed in a modular form as a combination of cohorts and breeding actions. As breeding actions like phenotyping, selection, aging, or reproduction are typically applied on groups of individuals, the relevant individuals for each breeding action can be selected via three different keywords:

1. gen: all individuals of a certain generation
2. cohort: group of individuals generated via the same breeding action
3. database: all (or in principle specific) individuals of a certain generation and sex

Similar to the gene-flow concept (Hill 1974), a cohort describes a group of individuals with usually identical characteristics like age, sex and genetic origin. As the three types of groups can also be combined, handling of overlapping generations in a breeding program is possible. Cohorts and breeding actions are defined in a generic way and are parametrized, so that any breeding program of arbitrary complexity can be modeled as a suitable sequence of cohorts and breeding actions.

All data for a population is stored in a list that contains general and individual information. The general part provides information on the underlying genetics like the physical position of each marker, allelic variants or structure of the underlying genetic traits. The individual part contains information that is specific to the individual. Similarly to simulators like SBVB (Pérez-Enciso *et al.* 2017), haplotypes are stored for founder individuals only. For all other individuals only points of recombination and mutation, and their genetic origins are stored. In particular, haplotypes are not permanently stored but only derived when needed. Therefore, the required memory is minimized and only increases slightly with increasing marker density. When thousands of generations are simulated it is advisable to classify additional generations as new founders to reduce the number of recombinations and mutations to be stored in subsequent generations. The usefulness of this is highly dependent on the ratio between genome length and number of markers and on how many new founder genotypes have to be stored. As the population itself in a breeding program usually occupies only a small share of the required memory and less than one hundred generations are considered, benefits here are usually small, making this only relevant/required in large scale population genetic studies.

Simulation of multiple correlated traits with and without underlying QTL is supported. Classical additive, dominant and epistatic or pleiotropic QTL can be defined and any effect structure of multiple interacting loci is supported. Each locus has to be assigned with a position in Morgan and different recombination rates for subgroups (*e.g.*, males/females) are supported. Information on the number of markers can be manually entered or imported via a database (Ensembl, (Zerbino *et al.* 2017)), a map-file (Purcell *et al.* 2007) or a vcf-file (Danecek *et al.* 2011). For common species, exemplary map files are provided in the associated package MoBPSmaps (Pook 2019). Genotype data for a base population can be imported via PLINK (Purcell *et al.* 2007) and/or vcf-format (Danecek *et al.* 2011), sampled internally or generated by executing prior simulation in MoBPS and/or other tools (Chen *et al.* 2009; Sargolzaei and Schenkel 2009) to generate the required population structure. All breeding actions performed in the simulation can be tracked and assigned with costs to derive the expenses of the program. Different breeding programs can be compared in terms of their economic revenue or other target functions (*e.g.*, development of the inbreeding rate) one is interested in.

Common methods for selection such as optimal genetic contributions (Meuwissen 1997) are implemented and a variety of different packages for breeding value estimation can be switched on. This includes BGLR (Pérez and de los Campos 2014), sommer (Covarrubias-Pazaran 2016) and rrBLUP (Endelman 2011), as well as an efficient implementation for solving the mixed model (Henderson 1975) in the traditional GBLUP model (Meuwissen *et al.* 2001; VanRaden 2008) that is assuming known heritability and is using the R-package RandomFieldsUtils (Schlather *et al.* 2019) for the matrix inversion. Inputs for these packages such as the different pedigree and genomic relationship matrices (VanRaden 2008; Legarra *et al.* 2014; Martini *et al.* 2017) can be derived via highly efficient and fully-parallelized bit-wise matrix multiplications (R-package miraculix (Schlather 2020)). Non of the mentioned packages, however, is required to execute simulations in MoBPS. In particular, all functionality of the MoBPS R-package is still available when miraculix is not installed, with the downside of higher computing times and memory demands.

The simulations in MoBPS are based on two main functions: *creating.diploid()* and *breeding.diploid()*. Here, *creating.diploid()* initializes the base-line population and *breeding.diploid()* performs breeding actions on an existing population list. As a simple example consider the following script:

```
library (MoBPS)
pop <- creating. diploid(nsnp =10000, nindi =100, chr.nr =5, chromosome.length = 2, n.additive =50, n.dominant = 10, var.target = 1, name.cohort="Founder")
pop <- breeding.diploid(pop, heritability = 0.5, new.bv. observation="all")
pop <- breeding.diploid(pop, bve = TRUE)
pop1 <- breeding.diploid(pop, breeding.size = 100, selection.size = c(20,20), selection.criteria ="bve", selection.m.cohorts ="Founder_M", selection.f.cohorts ="Founder_F", name.cohort ="Offspring")
pop2 <- breeding.diploid(pop, breeding.size = 100, selection.size = c(5,20), selection.criteria="bve", selection.m.cohorts ="Founder_M", selection.f.cohorts ="Founder_F", name.cohort ="Offspring")
```

Via this code, we first generate a base population containing 100 individuals with 10,000 markers. The underlying genome consists of 5 chromosomes with a length of 2 Morgan each and equidistant markers. Furthermore, we generated a single trait that is impacted by 50 purely additive QTL and 10 dominant QTL and scale QTL effect to result in a trait with genomic variance of 1.

In the next step, we initialize a breeding action to generate phenotypes for all individuals in the population with an assumed heritability of 0.5. Next, a breeding value estimation is performed. Since no cohorts are selected, the last (and only) generation of the population list will be considered for the breeding value estimation. Lastly, we generate 100 offspring by random mating. Here, two scenarios are considered with different selection intensity with the top 5/20 male and top 20 female individuals being used for reproduction, leading to gains of 0.663/0.922 genomic standard deviations and an increase in inbreeding in terms of average kinship of 0.0062/0.0156 (Figure 1). In principle, all three breeding actions

Downloaded from https://academic.oup.com/g3journal/article/10/6/1915/6026363 by Universitätsbibliothek Mannheim user on 04 April 2023

performed via *breeding.diploid()* could have also been executed in a joint step. For a full list of all possible breeding actions and available parameters we refer to our user manual (available at https://github.com/tpook92/MoBPS).

For a quick overview of the simulated population, the function *summary()* can be used:

```
summary(pop1)
Population size:
Total: 200 Individuals
Of which 100 are male and 100 are female.
There are 2 generations and 4 unique cohorts.

Genome Info:
There are 5 unique chromosomes.
In total there are 10000 SNPs.
The genome has a total length of 10 Morgan.
The genome has a physical size of about: 1 GB

Trait Info:
There is 1 modeled trait.
The trait has underlying QTL
The trait is named: Trait 1
```

A variety of functions is provided to export required information such as the phenotypes (*get.pheno()*), the genotypes (*get.geno()*) and the pedigree (*get.pedigree()*) for selected individuals from the population list. These functions are thoroughly described in chapter 7 of the user manual (available at https://github.com/tpook92/MoBPS). In addition, functions to derive rates of inbreeding (*kinship.emp()*), development of breeding values (*bv.development()*) or changes in allele frequency over time (*analyze.population()*) are provided to further analyze the resulting population list.

Data availability

An executable version of MoBPS and the associated R-packages miraculix (Schlather 2020), RandomFieldsUtils (Schlather *et al.* 2019) and MoBPSmaps (Pook 2019) for Windows and Linux are freely available at https://github.com/tpook92/MoBPS. This directory also contains an comprehensive user manual explaining the functionality of all input parameters and utility functions in MoBPS. A frozen version of the R-packages MoBPS (v1.4.87), MoBPSmaps (v0.1.7), miraculix (v0.9.10), RandomFieldsUtils (v0.5.9), and our user manual at submission are also provided there. The MoBPS R-package can be directly installed within your R session via following commands:

```
install.packages(”devtools”)
devtools::install_github(”tpook92/MoBPS”, subdir=”pkg”)
```

RESULTS AND DISCUSSION

The package MoBPS is completely written in R (R Core Team 2017) so that all functionalities for genetic applications are platform independent. The R-packages miraculix (Schlather 2020) can be activated in MoBPS and leads to more efficient data storage and shorter simulation times. In particular vector multiplications with genetics data (0,1,2) are performed via bitwise operations on a whole register (128/256 bit) using SSE2/AVX2. Computing times are similar to the ones in PLINK (Purcell *et al.* 2007) with one fourth of the memory usage. The interested reader is referred to Schlather (2020) for extended benchmark-testing on miraculix. The storage of founder genotypes is 4 times as efficient as in AlphaSim (Faux *et al.* 2016) and 32/64 times as efficiency as the use of integer/double variables to store haplotypes.

Even though basically all information regarding each individual is stored, the required memory in MoBPS is still relatively low as a highly efficient storage structure is used. Haplotypes of founders and details on the origin of the segment between points of recombination are stored bitwise. E.g. the simulation of 20 generations with 50,000 cows with 50,000 markers and breeding value estimation via GBLUP takes 26.2 hr using 24 cores on a server cluster with Intel E5-2650 (2X12 core 2.2GHz) processors. At peak, 65 GB of memory was used. The main share of this was required for the storage of the genomic relationship matrix whereas the resulting population list, containing more than a million individuals, only had a size of about 0.44 GB. The biggest proportion of the computing time is used for breeding value estimation (25.3 hr, 96.4%). The generation of new animals took 55 min (3.5%, 304 animals per second using a single core). All other parts needed negligible computing time (132 sec, 0.1%). Computing times for most parts (except breeding value estimation) increase linearly with the number of individuals. This highly efficient storage structure therefore also allows for the simulation of historical populations with thousands of

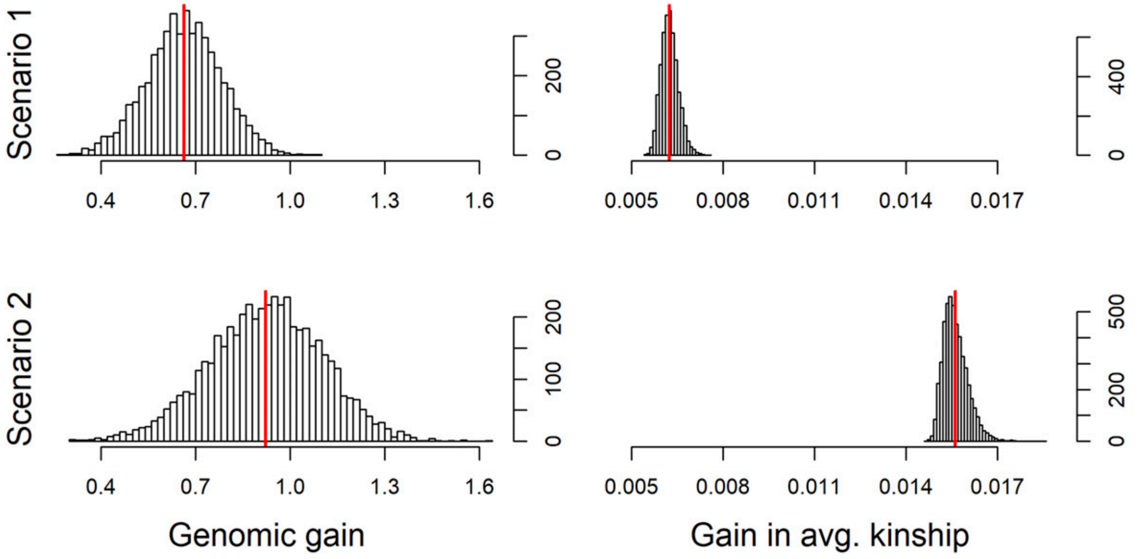

Figure 1 Resulting genetic gain and increase in inbreeding with low (Scenario 1) and high (Scenario 2) selection pressure for 5,000 simulation runs each.

Downloaded from https://academic.oup.com/g3journal/article/10/6/1915/6026363 by Universitätsbibliothek Mannheim user on 04 April 2023

generations and undergone population dynamics such as genetic bottlenecks, migration or mutational drift.

The flexible and efficient environment of MoBPS allows for the simulation of a variety of different and potential large-scale breeding programs. For exemplary scripts of more complex breeding programs we refer to the user manual. Exemplary simulations are given for the effect of gene editing in a cattle breeding program (Simianer *et al.* 2018), the simulation of a multi-parent advanced generation intercross in maize (Pook *et al.* 2019), an introgression scheme in chicken (Ha *et al.* 2017) and the generation of a base population with a hard sweep. A further advantage of MoBPS compared to other simulation tools is its flexible structure that allows the user to substitute single steps of the breeding program with a customized approach. For this consider the following example to execute one owns breeding value estimation:

```
genos <- get.geno(pop, gen = 1)
y <- get.pheno(pop, gen = 1)
indi_names <- colnames(genos)
# Execute one owns function to perform
# the breeding value estimation
y_hat <- own.method.for.bve (genos, y)
# Enter BVEs in the population-list
pop <- insert.bve(pop, bves = cbind (indi_names, y_hat))
```

Even though a simulation study can never fully reflect reality and is relying on model assumptions, the use of a simulation study comes with major benefits and still allows the user to draw important conclusions. In contrast to reality the underlying truth in a simulation study is known, and therefore new methods can be thoroughly evaluated and compared to existing ones. Furthermore, the effects of particular breeding actions on a variety of output dimension can be assessed and compared. This in turn can be used to derive an ideal resource allocation and optimize potentially highly complex breeding scenarios in a setting that can be evaluated multiple times and without constrains both in terms of money and time.

ACKNOWLEDGMENTS

This package was developed in the context of the European Union's Horizon 2020 Research and Innovation Program under grant agreement n°677353 IMAGE.

LITERATURE CITED

Chen, G. K., P. Marjoram, and J. D. Wall, 2009 Fast and flexible simulation of dna sequence data. Genome Res. 19: 136–142. https://doi.org/10.1101/gr.083634.108

Covarrubias-Pazaran, G., 2016 Genome-assisted prediction of quantitative traits using the r package sommer. PLoS One 11: e0156744. https://doi.org/10.1371/journal.pone.0156744

Danecek, P., A. Auton, G. Abecasis, C. A. Albers, E. Banks *et al.*, 2011 The variant call format and vcftools. Bioinformatics 27: 2156–2158. https://doi.org/10.1093/bioinformatics/btr330

Endelman, J. B., 2011 Ridge regression and other kernels for genomic selection with r package rrblup. Plant Genome 4: 250–255. https://doi.org/10.3835/plantgenome2011.08.0024

Faux, A.-M., G. Gorjanc, R. C. Gaynor, M. Battagin, S. M. Edwards *et al.*, 2016 Alphasim: Software for breeding program simulation. Plant Genome 9: plantgenome2016.02.0013. https://doi.org/10.3835/plantgenome2016.02.0013

Ha, N.-T., T. Pook, C. Dierks, S. Weigend, R. Preisinger *et al.*, 2017 A simulation approach to optimize breeding programs with application to the introgression of the blue egg color into a high performing layer line. 10th European Symposium on Poultry Genetics 2017: 107.

Henderson, C. R., 1975 Best linear unbiased estimation and prediction under a selection model. Biometrics 31: 423–447. https://doi.org/10.2307/2529430

Hill, W. G., 1974 Prediction and evaluation of response to selection with overlapping generations. Anim. Sci. 18: 117–139. https://doi.org/10.1017/S0003356100017372

Legarra, A., O. F. Christensen, I. Aguilar, and I. Misztal, 2014 Single step, a general approach for genomic selection. Livest. Sci. 166: 54–65. https://doi.org/10.1016/j.livsci.2014.04.029

Martini, J. W. R., N. Gao, D. F. Cardoso, V. Wimmer, M. Erbe *et al.*, 2017 Genomic prediction with epistasis models: On the marker-coding-dependent performance of the extended gblup and properties of the categorical epistasis model (ce). BMC Bioinformatics 18: 3. https://doi.org/10.1186/s12859-016-1439-1

Meuwissen, T. H. E., 1997 Maximizing the response of selection with a predefined rate of inbreeding. J. Anim. Sci. 75: 934–940. https://doi.org/10.2527/1997.754934x

Meuwissen, T. H. E., B. J. Hayes, and M. E. Goddard, 2001 Prediction of total genetic value using genome-wide dense marker maps. Genetics 157: 1819–1829.

Pérez, P., and G. de los Campos, 2014 Genome-wide regression & prediction with the BGLR statistical package. Genetics 198: 483–495. https://doi.org/10.1534/genetics.114.164442

Pérez-Enciso, M., N. Forneris, G. de Los Campos, and A. Legarra, 2017 Evaluating sequence-based genomic prediction with an efficient new simulator. Genetics 205: 939–953. https://doi.org/10.1534/genetics.116.194878

Pook, T., 2019 Mobps: Modular breeding program simulator: Available at https://github.com/tpook92/mobps; r-package version 1.4.14.

Pook, T., M. Schlather, G. de los Campos, M. Mayer, C. C. Schoen *et al.*, 2019 Haploblocker: Creation of subgroup specific haplotype blocks and libraries. Genetics 212: 1045–1061. https://doi.org/10.1534/genetics.119.302283

Purcell, S., B. Neale, K. Todd-Brown, L. Thomas, M. A. R. Ferreira *et al.*, 2007 Plink: A tool set for whole-genome association and population-based linkage analyses. Am. J. Hum. Genet. 81: 559–575. https://doi.org/10.1086/519795

R Core Team, 2017 R: A language and environment for statistical computing.

Sargolzaei, M., and F. S. Schenkel, 2009 Qmsim: A large-scale genome simulator for livestock. Bioinformatics 25: 680–681. https://doi.org/10.1093/bioinformatics/btp045

Schlather, M., 2020 Efficient calculation of the genomic relationship matrix. bioRxiv. https://doi.org/10.1101/2020.01.12.903146 (Preprint posted January 14, 2020)

Schlather, M., R. Furrer, and M. Kroll, 2019 Randomfieldsutils: Utilites for the simulation and analysis of random fields: Available at https://github.com/tpook92/mobps; r-package version 0.5.9.

Simianer, H., T. Pook, and M. Schlather, 2018 Turning the page - the potential of genome editing in breeding for complex traits revisited. World Congress on Genetics Applied to Livestock p. 190.

Sun, X., T. Peng, and R. H. Mumm, 2011 The role and basics of computer simulation in support of critical decisions in plant breeding. Mol. Breed. 28: 421–436. https://doi.org/10.1007/s11032-011-9630-6

Täubert, H., F. Reinhardt, and H. Simianer, 2010 Zplan+, a new software to evaluate and optimize animal breeding programs. World Congress on Genetics Applied to Livestock p. 950.

VanRaden, P. M., 2008 Efficient methods to compute genomic predictions. J. Dairy Sci. 91: 4414–4423. https://doi.org/10.3168/jds.2007-0980

Zerbino, D. R., P. Achuthan, W. Akanni, M. R. Amode, D. Barrell *et al.*, 2017 Ensembl 2018. Nucleic Acids Res. 46: D754–D761. https://doi.org/10.1093/nar/gkx1098

Communicating editor: A. Lipka

Downloaded from https://academic.oup.com/g3journal/article/10/6/1915/6026363 by Universitätsbibliothek Mannheim user on 04 April 2023