GENO-DIVER
Genetic Simulation Toolkit

Jeremy T. Howard[1]
Francesco Tiezzi[1]
Jennie E. Pryce[2,3]
Christian Maltecca[1]

[1]North Carolina State University, Raleigh, NC, USA
[2]Department of Economic Development, Jobs, Transport and Resources, Victoria, Australia
[3]La Trobe University, Victoria, Australia.

Contents

Introduction

A variety of disciplines including conservation (McMahon et al. 2014), animal and plant breeding (De los Campos et al. 2013) and human genetics (Yang et al. 2010) are currently making use of large volumes of genetic marker information. In particular, within animal and plant breeding programs the use of genomic information to predict the genetic merit of individuals has become a routine practice (Jonas & Koning 2015; Morrell et al. 2012). This has resulted in a significant increase in the number of individuals within a herd/population having genomic information. Nonetheless, the ability to use this information to efficiently manage agricultural populations at the genomic level, both for preserving genetic diversity and lessening inbreeding depression, remains a challenge for the near future. Although the identification of lethal mutations of large effect segregating within livestock populations with genomic data is possible (VanRaden et al. 2011), optimal mating procedures that minimize the frequency of a large number of lethal and more importantly sublethal mutations across generations have not been fully implemented. As the popularity of genotyping breeding individuals increases across species, the possibility to utilize genomic information from multiple sources to manage genomic information will also increase. Methods that make effective use of information from multiple sources including performance, genome diversity and inbreeding load at the selection and/or mating step are in increasing need. The routine genotyping of individuals is a significant investment from several sectors of agriculture and the need to spread the costs across multiple avenues are of considerable practical interest.

The use of simulation is a low cost alternative to assess and validate proposed methods to predict genetic values or to compare alternative selection or mating strategies across time. A number of simulation programs have been developed that simulate breeding livestock/crops populations, but they primarily focus on testing strategies where the genetic architecture is based solely on a quantitative trait governed by additive effects (Sargolzaei & Schenkel 2009; Hickey et al. 2012; Cheng et al. 2015; Prez-Enciso & Legarra 2016). Currently there is no single self contained software that can simulate complex traits involving both quantitative and fitness components along with the ability to generate complex pedigrees and genomic information ranging from sparse marker to sequence information. Consequently, determining how various selection and management practices impact the fitness and the overall genomic variability of a population undergoing selection for a quantitative trait remains challenging. Furthermore, the precise genetic architecture of complex traits is largely unknown although in general it is assumed that complex traits are affected by variation in a large number of genes, most of which have individually minor effects (Weiss 2008). Knowledge on the demographic history of a population and the extent of linkage disequilibrium is also being generated from dense marker panels or sequence information (McKay et al. 2007; Ai et al. 2013; Porto-Neto et al. 2014). As more information is generated on the genetic architecture of complex traits and variation across the genome, simulation can be employed to more effectively predict how current selection and management practices will impact future generations.

Disclaimer

This document outlines how to run the simulation program Geno-Diver and describes the parameters utilized within the program. The software is free for academic and non-commercial use. The authors accept no responsibility for the accuracy of results obtained by using Geno-Diver software.

Geno-Diver is being updated with new tools and functions constantly. If you would like the program to do something that it currently does not do, don't hesitate to contact me at jthoward@ncsu.edu.

Please notify jthoward@ncsu.edu or cmaltec@ncsu.edu if you think the results are not correct or you have encountered a bug. We have written the program in a way for us to reconstruct the simulation scenario you ran to more effectively solve the problem.

The software was profiled using the Valgrind software (valgrind). In a few instances, calling functions within the Intel MKL library did result in "possibly lost" errors. Although, as outlined in the Intel MKL user manual, new buffers that the library allocates when an Intel MKL function gets called are not deallocated until the program ends. The use of mkl_free_buffers() and mkl_thread_free_buffers() were called at the end of the program and this fixed the majority of the issues.

The reference for the Geno-Diver software:

-

The reference for the MaCS software:

- Chen, G. K., P. Marjoram & J. D. Wall. 2009. Fast and flexible simulation of DNA sequence data. Genome Res. 19:136-142.

Download

Source code, executables and helper files are available at the GIT-HUB account.

Linux Executable Files:

- macs
- msformatter
- GenoDiver

Source Code Files:

- Animal.h
- AnimalFun.cpp
- PopulationSimulator.cpp
- Simulation_Functions.cpp
- makefile

Helper Files:

- Geno-Diver Manual
- ADSA-ASAS National Meeting Presentation
- Example1.txt
- Example2.txt
- Example3.txt
- Example4.sh
- complete_parameter_file.txt

Computing Environment

The code is written in the C++11 language using object-oriented techniques. The application has been tested to run on Linux platforms. The software makes use of two external libraries, Intel MKL and Eigen.

EIGEN Library:
EIGEN is freely available at: Eigen Site
Once at the site, download the latest stable release and uncompress it. For example, the current downloaded package is called "eigen-eigen-07105f7124f9.tar". To use it you just have to place it in the file where all of the other Geno-Diver files are located and uncompress the file. Once you uncompressed the file it will be a folder (e.g. "eigen-eigen-07105f7124f9"). The uncompressed file serves as your path in the makefile outlined below.

Intel MKL Library:
Intel MKL is a commercial library and is available for purchase. However, there is an opportunity to obtain the Intel MKL library (for Linux) free of charge for non-commercial use at the following website: Intel MKL Site
The Intel MKL can sometimes be tricky to download and link, but there is a step-by-step protocol within the folders that are downloaded or instructions are at Intel MKL Guide. Depending on the computing system you are running, a guide to linking the Intel MKL libraries are at Intel MKL Linking Guide.

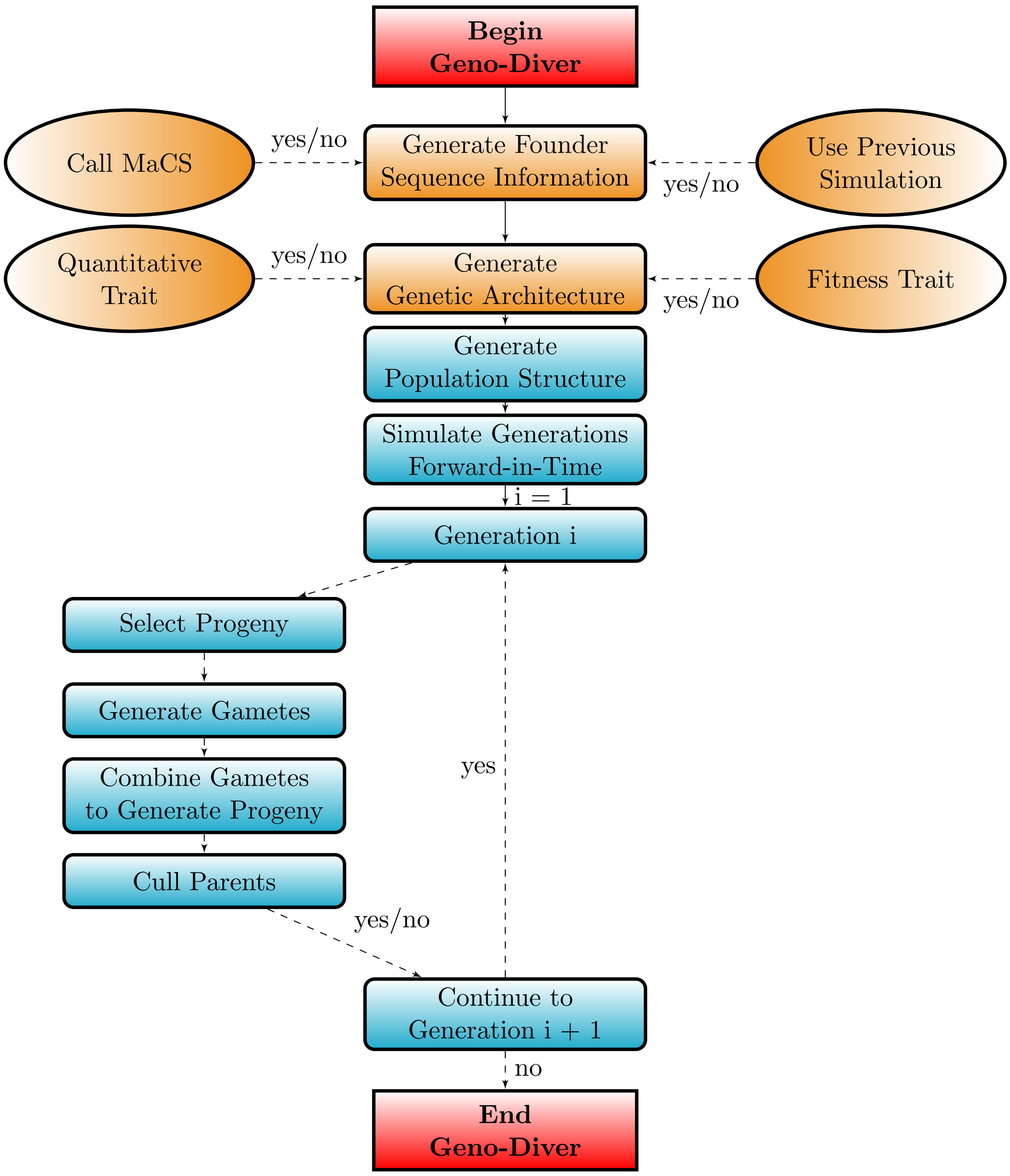
C++11 Version:
Versions of gcc 4.7 or newer support the C++11 standards. You need to install or update to the correct version of gcc using the standard package manager or installer, depending on what type of OS you are using. Some helpful websites include:
gcc helper 1 gcc helper 2

Compiling:
Once both EIGEN, Intel MKL libraries and gcc version 4.7 or newer have been correctly installed and the folders placed in the directory where all of the Geno-Diver source code files are located the last thing you have to do is change the path for EIGEN and Intel MKL libraries.

In the makefile change lines 13 and 14 to the path which aligns to the updated EIGEN and MKL path.

After changing the path, type "make" on the command line. An executable file called "GenoDiver" is now in your working directory.

Overview of Program


Begin Geno-Diver
Call MaCS
yes/no
Generate Founder Sequence Information
yes/no
Use Previous Simulation
Quantitative Trait
yes/no
Generate Genetic Architecture
yes/no
Fitness Trait
Generate Population Structure
Simulate Generations Forward-in-Time
i = 1
Generation i
Select Progeny
Generate Gametes
Combine Gametes to Generate Progeny
Cull Parents
yes
yes/no
Continue to Generation i + 1
no
End Geno-Diver

Running the Program

At the current time executable files are available only for Linux operating environments and are located on the GitHub page (GENO-DIVER). To run the program place the following executable files in a folder:

- GenoDiver.
- macs.
- msformatter.

Before running the program, the file permission need to be checked. After verifying the permissions, a parameter file is generated and placed in the same folder as the previous three files. If you are new to the simulation program, multiple examples are on the GitHub page. The simulation program reads the parameter file by searching for keywords that are capitalized and then followed by a colon. Therefore any phrase that does not meet the search criteria is ignored when initializing parameters within the program.

To run the program type "./GenoDiver" and then the name of your parameter file. So for example, if the parameter file is named "parameterfile" then to run the program type "./GenoDiver parameterfile". During the simulation, the program outputs minimal comments on the progress. A more thorough description of the status of the program is printed to the log file (i.e. "log_file.txt").

After running the program it is a strongly recommended to check the parameters initialized at the top of the log file within the output folder. The log file contains a large amount of information and is a great tool to ensure that the parameters and outcome of the simulation is what is intended. If the program is not running correctly the log file should provide knowledge on why and where the simulation crashed or exited.

If you are unsure of the problem an e-mail can be sent to jthoward@ncsu.edu with a copy of the log file for me to replicate the results.

Program Parameters

A file that specifies all the parameters is outlined on the git-hub page ("complete_parameter_file.txt"). Not all of them are required for the program to run. A file with the mandatory parameters for the simulation to run is on the git-hub page ("Example1.txt"). All keywords should be in capital letters and the parameter(s) specified are separated by spaces. Only parameters after the keywords impact the simulation. Trailing spaces at the end of a parameter should be avoided.

A description of all the parameters specified in the program is outlined below. The appendix also contains further discussion and suggestions.

General Parameters

START

Description: - Determines where to start the simulation program.

Value:

- sequence: Starts at the sequence generation step.
- founder: Skips sequence generation and begins generating the founder population utilizing sequence information from a previous run.

Usage: - "START: sequence".

Type: - Mandatory.

Note:

The use of this option saves time and space due to the large size of sequence information from MaCS. If you are using a different effective population size or MaCS diversity metric you always have to start at the sequence step, while any replicate that has the same MaCS parameters can start at the founder step and use the previously generated haplotypes saved within the output folder. When calling multiple replicates, the simulation for any replicate after the first starts at the founder step by default.

OUTPUTFOLDER

Description:	- Directory name of files generated from simulation.
Value:	- Any valid folder name.
Usage:	- “OUTPUTFOLDER: GenoDiverFiles”.
Type:	- Optional. Default is ”GenoDiverFiles”.
Note:	If the directory contains information from a previous simulation, it may be deleted depending on the value of the START parameter. If the START parameter is “sequence” everything is deleted. If the START parameter is “founder” everything but sequence information is deleted. The sequence files could potentially take up a large amount of memory. Only one sequence file can be generated at a time unless the mac, msformatter and GenoDriver executables are in separate folders.

SEED

Description:	- Declares the seed number.
Value:	- Can be any integer value.
Usage:	- “SEED: 1501”.
Type:	- Optional. Default is the system time.
Note:	- The seed appears at the beginning of the log file if the user does not declare it.

THREAD

Description:	- Declares the number of threads used for parallel processing.
Value:	- Integer value based on number of cores available.
Usage:	- “NTHREAD: 4”.
Type:	- Optional. Default is 1.

REPLICATES

Description:	- Declares the number of replicates to produce and the seed is increased by 1 from the previous value for each one.
Value:	- Integer value.
Usage:	- “NREP: 1”.
Type:	- Optional. Default is 1.
Note:	- All important replicate files are in a folder within the output folder path called “replicates”. Each file has the seed number at the end to distinguish between replicates.

Genome and Marker Information

CHR

Description:	- Sets the number of chromosomes to generate.
Value:	- Integer from 1 to 31.
Usage:	- “CHR: 3”.
Type:	- Mandatory.

CHR_LENGTH

Description:	- Sets the length of each chromosome, in Megabases.
Value:	- Integer Value.
Usage:	- “CHR_LENGTH: 150, 150, 150”
Type:	- Mandatory.
Note:	- Can’t be any spaces after final number.

NUM_MARK

Description:	- Sets the number of evenly spaced markers for each chromosome.
Value:	- Integer Value.
Usage:	- “NUM_MARK: 4000, 4000, 4000”
Type:	- Mandatory.
Note:	- Quantitative and fitness QTL cannot be markers at the current time. Can’t be any spaces after the final number. The number of markers has to be less than the available number of mutations within a chromosome generated from MaCS.

MARKER_MAF

Description:	- Minimum allele frequency allowed for markers.
Value:	- Range from 0.0 to 0.50.
Usage:	- “MARKER_MAF: 0.075”
Type:	- Optional. Default is 0.05.

QTL

Description:
- The number of quantitative trait loci (QTL) for each chromosome. The QTL location is generated based on a uniform distribution from 0 to the length of the chromosome.

Value: - Integer. 5000 is max number of QTL and fitness trait loci (FTL).

Usage: - "QTL: 50"

Type: - Mandatory.

QUANTITATIVE_MAF

Description: - Minimum allele frequency allowed for QTL.

Value: - 0.0 to less than 0.5.

Usage: - "QUANTITATIVE_MAF: 0.05"

Type: - Optional. Default is 0.05.

FIT_LETHAL

Description:
- The number of lethal FTL for each chromosome. The lethal FTL location is generated based on a uniform distribution from 0 to the length of the chromosome.

Value: - Integer. 5000 is max number of QTL and FTL.

Usage: - "FIT_LETHAL: 25".

Type: - Optional. Default is 0.

Note: - Lethal FTL can't have covariance with the quantitative trait.

FIT_SUBLETHAL

Description:
- The number of sub-lethal FTL for each chromosome. The sub-lethal FTL location is generated based on a uniform distribution from 0 to the length of the chromosome.

Value: - Integer. 5000 is max number of QTL and FTL.

Usage: - "FIT_SUBLETHAL: 25".

Type: - Optional. Default is 0.

Note: - Sub-lethal FTL can have covariance with the quantitative trait.

FITNESS_MAF

Description:

- Minimum allele frequency allowed for FTL for lethal and sub-lethal. The first and second value pertain to lethal and sub-lethal FTL, respectively.

Value: - Range from 0.0 to 0.5.

Usage: - "FITNESS_MAF: 0.02 0.08".

Type: - Optional. Default is 0.02 and 0.08.

Note:

- The minimum value is 0.01. Using the default values, lethal FTL range from 0.01 to 0.02 and sub-lethal FTL range from 0.01 to 0.08. If the frequency of either lethal or sub-lethal FTL is too high, a large number of founders and progeny may not make it to breeding age. If the number of individuals is too small to generate the founder generation the simulation exits.

FOUNDER_HAPLOTYPES

Description:

- The number of haplotypes generated by MaCS. Need to ensure that it is greater than two times the total number of female animals needed in the founder population.

Value: - Integer Value.

Usage: - "FOUNDER_HAPLOTYPES: 4000".

Type: - Optional. Default is based on male & female number.

HAPLOTYPE_SIZE

Description:

- The number of markers contained within a non-overlapping haplotype window. Haplotype windows are used to generate haplotype-based relationship matrices and summary statistics.

Value: - Integer Value.

Usage: - "HAPLOTYPE_SIZE: 50".

Type: - Optional. Default is 50.

RECOMBINATION

Description:

- The type of distribution that generates the location of recombination events along the genome. The Poisson distribution generates the number of recombination events and the rate parameter fixed at 1.0 across all chromosomes.

Value:

- Uniform: Recombination sampled from a Uniform distribution from 0 to 1.0.
- Beta: Recombination sampled from a Beta distribution (0.5, 0.5) from 0 to 1.0. Recombinations occurs more often at the end of the chromosome than towards the middle.

Usage: - "RECOMBINATION: Uniform".

Type: - Optional. Default is Uniform.

QTL Distributions

ADD_QUAN

Description:

- The parameters for the gamma distribution that generate the un-scaled additive effect for QTL. The effects are scaled such that the sum of the QTL variances in the founder population is equivalent to the proportion of the variance that is due to the additive gene action specified. A complete description is in Appendix II.

Value:

- Shape: Shape of gamma distribution.
- Scale: Scale of gamma distribution.

Usage: - "ADD_QUAN: 0.4 1.66".

Type: - Optional. Default is 0.4 1.66.

DOM_QUAN

Description:

- The parameters for the normal distribution that generate the degree of dominance (h) for QTL. The effects are scaled such that the sum of the QTL variances in the founder population is equivalent to the proportion of the variance that is due to dominant gene action specified. A complete description is in Appendix II.

Value:
- Mean: Mean of a normal distribution.
- Standard Deviation (SD): SD of a normal distribution

Usage: - "DOM_QUAN: 0.1 0.2".

Type: - Optional. Default is 0.1 0.2.

LTHA

Description:
- The parameters for the gamma distribution that generate the selection coefficients for lethal FTL. A complete description is in Appendix II.

Value:
- Shape: Shape of gamma distribution.
- Scale: Scale of gamma distribution.

Usage: - "LTHA: 1.6 0.1".

Type: - Optional. Default is 1.6 0.1.

LTHD

Description:
- The parameters for the normal distribution that generate the degree of dominance for the lethal FTL. A complete description is in Appendix II.

Value:
- Mean: Mean of a normal distribution.
- Standard Deviation (SD): SD of a normal distribution

Usage: - "LTHD: 0.05 0.1".

Type: - Optional. Default is 0.05 0.1.

SUBA

Description:
- The parameters for the gamma distribution that generate the selection coefficients for sublethal FTL. A complete description is in Appendix II.

Value:
- Shape: Shape of gamma distribution.
- Scale: Scale of gamma distribution.

Usage: - "SUBA: 0.1 0.2".

Type: - Optional. Default is 0.1 0.2.

SUBD

Description:

- The parameters for the normal distribution that generate the degree of dominance for sublethal FTL. A complete description is in Appendix II.

Value:

- Mean: Mean of a normal distribution.
- Standard Deviation (SD): SD of a normal distribution

Usage: - "SUBD: 0.3 0.1".

Type: - Optional. Default is 0.3 0.1.

COVAR

Description:

- Determines the relationship between the additive effect of the quantitative trait and selection coefficient for the sub-lethal trait. The relationship is a function of the number of QTL that is both quantitative and sub-lethal and the rank correlation between the effects. A complete description is in Appendix II.

Value:

- Proportion of Pleiotropic QTL: Proportion of QTL that are both quantitative and sub-lethal
- Genetic correlation: The rank correlation between QTL effects.

Usage: - "COVAR: 0.5 0.2".

Type: - Optional. Default is 0.0 0.0..

Note:

- Care needs to be taken to generate the desired relationship between the quantitative and fitness trait. Fitness values range from 0 to 1 and higher values leading to a lower fitness value. If the two traits are antagonistic under the scenario of high values being favorable, the correlation should be positive. One just needs to change the favorable direction of the quantitative trait to alter the interpretation.

Population Parameters

FOUNDER_Effective_Size

Description:

- Used to generate the population history of the haplotypes generated from MaCS. There are multiple default scenarios and represent a wide range of LD patterns, or one can specify a custom effective population size and population history. Each scenario has a slightly different population history parameter (i.e. "eN"). Lastly, the population history can be a single value with no historical population. A description is in Appendix III.

Value:

- Ne70: A scenario that generates a large amount of short LD.
- Ne100_Scen1: A scenario that generates moderate short LD.
- Ne100_Scen2: A scenario that generates moderate short LD.
- Ne250: A scenario that generates the minimal LD.
- Ne1000: A scenario that generates the minimal LD.
- CustomNe - Read in own population history parameters.
- Any value greater than 1: Utilizes the value as the effective population size and no population history assumed.

Usage: - "FOUNDER_Effective_Size: 50".

Type: - Mandatory.

MUTATION

Description:

Used in the MaCS software to generate scaled mutation parameter and in the simulation to generate new mutations as generations proceed.

Value:

- Mutation Rate: Probability of a new mutation occurring at a given base pair and follows the infinite-site model. A Poisson distribution with a rate parameter equal to the mutation rate times the length of the chromosome in nucleotides generates the total number of mutations occurring within a new gamete.
- The proportion of Mutations that can be QTL: Within the forward-in-time part of the simulation program, represents the number of non-neutral mutations that occurred out of the total number of new mutations. Each type of trait (i.e. quantitative, lethal, sub-lethal) has an equal chance of being chosen.

Usage: - "MUTATION: 2.5e-8 0.0".

Type: - Optional. Default is 2.5e-8 0.0.

VARIANCE_A

Description:	- Proportion of variance due to additive gene action.
Value:	- 0.0 to 1.0.
Usage:	- "VARIANCE_A: 0.25".
Type:	- Mandatory.

VARIANCE_D

Description:	- Proportion of variance due to dominant gene action.
Value:	- 0.0 to 1.0.
Usage:	- "VARIANCE_D: 0.05".
Type:	- Mandatory.
Note:	- Care needs to be taken in determining the additive and dominance variance and its implications on the number of QTL that display over-dominance or partial-dominance. Parameters to change that impact the dominance variance include the QTL MAF frequency, mean and standard deviation of the normal distribution that generates the degree of dominance parameter and the ratio of additive to dominance variance.

Selection and Mating Parameters

GENERATIONS

Description:	- Determines the number of generations to simulate.
Value:	- Any integer value.
Usage:	- "GENERATIONS: 10".
Type:	- Mandatory.
Note:	- The maximum number of generations vary depending on the size of your computer memory. At the current time if a quantitative trait is simulated the breeding values are generated by building the mixed model equations and solving them, which can take up a large amount of memory. It is advisable to start at a small number of generations and then build up to determine how much memory the scenario will use.

INDIVIDUALS

Description:

- Determines the number of males and females in each generation and replacement rate for parents each generation. Care should be taken on picking the number of offspring to have enough for the next generation. If the number of animals falls below the input male or female value, the program exits.

Value:

- Male Number: Number of males within each generation.
- Male Replacement: Proportion of males that are culled and replaced each generation.
- Female Number: Number of females within each generation.
- Female Replacement: Proportion of females that are culled and replaced each generation.

Usage: - “INDIVIDUALS: 50 0.2 600 0.2”.

Type: - Mandatory.

PARITY_MATES_DIST

Description:

- Determines the distribution of the number of mating pairs a sire has for each age group. A Beta distribution, which is parameterized by two parameters, is used to generate the distribution of mating pairs for a given age group. To generate the number of mating pairs by age class, the cumulative distribution function (CDF) is split into quadrants based on the number of age classes that occur within a generation. The total number of mating pairs within an age class is the proportion that falls within the CDF quadrant for a given age class.

Value: - Both parameters have to be positive values.

Usage: - “PARITY_MATES_DIST: 1 1”.

Type:

- Optional. The default is both parameters being 1, which is very similar to a uniform distribution, such that all age classes have the same proportion of mating pairs.

PROGENY

Description:

- Determines the number of progeny produced by each mating pair. The number of progeny per mating pair may not be the actual number of progeny produced if fitness QTL segregates in the population. The progeny of a mating pair makes it to breeding age if the fitness value of the progeny is greater than a random value derived from a uniform distribution ranging from 0 to 1. The fitness value of the progeny is a function of the multiplicative effect of all lethal and sub-lethal QTL genotype effects.

Value: - Ranges from 1 to 10.

Usage: - "PROGENY: 4".

Type: - Mandatory.

MAXFULLSIB

Description:

- Determines the maximum number of full-sib progeny selected within a family. Once the maximum number is reached the full-sib with the lowest selection criteria is no longer selected and the next best animal is selected. This process is repeated across all families until all families are below the value.

Value: - Ranges from 1 to number of progeny.

Usage: - "MAXFULLSIB: 2".

Type: - Optional. Default set at number of progeny.

SELECTION

Description: - The metric used to select individuals and favorable direction.

Value:

- Select: random, phenotype, true_bv or ebv.
- Direction: high or low.

Usage: - "SELECTION: ebv high".

Type: - Mandatory.

Note:

- For random selection, the direction does not impact the results and therefore the direction value does not matter.

SOLVER_INVERSE

Description:

- Parameters that specify which relationship matrix is used to estimate breeding values, how they will be solved and how the inverse will be calculated. Separated by spaces. Only the first variable is mandatory. The rest are set to the default values.

Value:

- Relationship Matrix:
  - pedigree: constructed using genealogical information.
  - genomic: constructed from genomic information as outlined in Van Raden (2008) and computing strategies were constructed based on Aguilar et al. (2011).
  - ROH: constructed based on shared ROH haplotypes and is constructed similar to Howard et al. (2016).
- Solver:
  - direct: Uses Cholesky decomposition for matrix inversion. When only simulating a small number of generations this is advised.
  - pcg: Uses the iterative preconditioned conjugate gradient (PCG) method and is faster than direct when the number of animals is large **(Default)**.
- Genomic Inverse:
  - cholesky: update previous inverse based on the algorithm presented by Meyer et al. (2012) **(Default)**.
  - recursion: utilizing the sequential update algorithm presented by Misztal et al. (2014).

Usage: - "SOLVER_INVERSE: pedigree'.

Type: - Only mandatory for the first variable.

Note:

Parameters that specify inverse calculations are for genomic-based relationships. Calculation of pedigree-based relationships is generated based on Meuwissen & Luo (1992).

MATING

Description: - Parameters that decide how animals are mated.

Value:

- random: males and females are randomly mated.
- random5: relationships ≥ 0.5 are not allowed to mate otherwise same as random.
- random25: relationships ≥ 0.25 are not allowed to mate otherwise same as random5.
- random125: relationships ≥ 0.125 are not allowed to mate otherwise same as random25.
- minPedigree: minimize parent co-ancestries based on pedigree.
- minGenomic: minimize parent co-ancestries based on genomic information (Van Raden 2008).
- minROH: minimize parent co-ancestries based on ROH information (Howard et al. 2016).

Usage: - "MATING: random5".

Type: - Mandatory.

Note:

At the current time relationships are minimized when applicable using the simulated annealing algorithm. For avoidance mating's (i.e. random5, random25, random 125) any coancestry below the threshold is zeroed out and optimization is performed using the simulated annealing algorithm.

CULLING

Description:

- The metric used to cull individuals and maximum age an animal can remain in the population before being removed due to old age. The direction is the same as the selection direction.

Value:

- cull: random, phenotype, true_bv or ebv.
- max age: any number greater than 1.

Usage: - "CULLING: ebv 5".

Type: - Mandatory.

Output Options

OUTPUT_LD

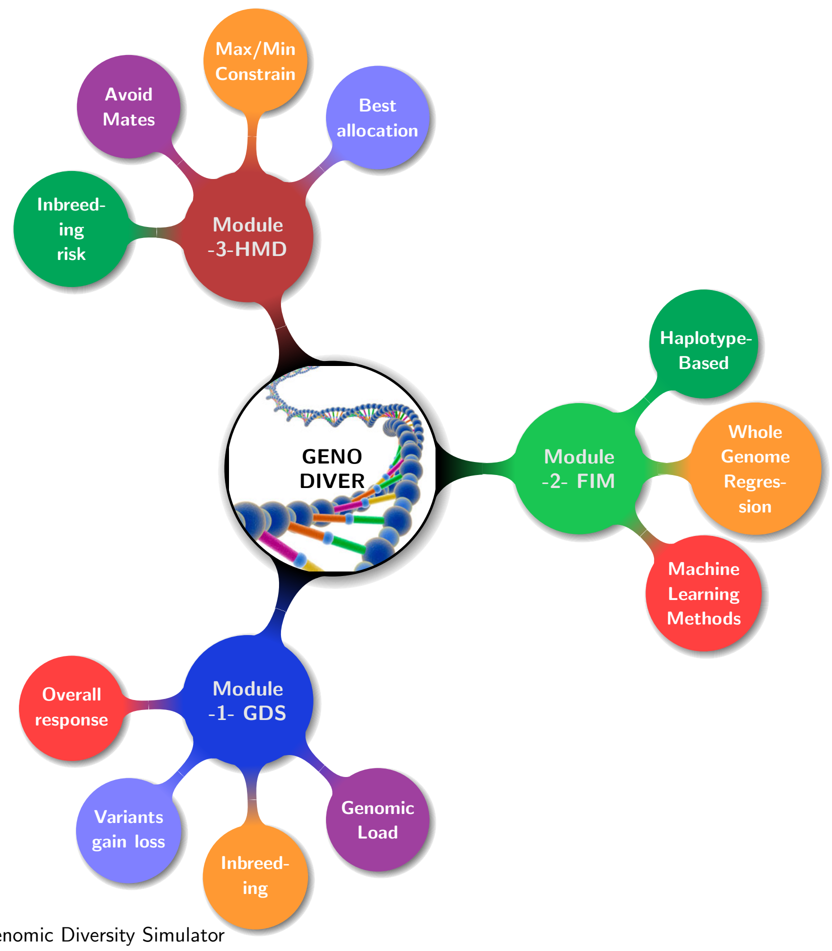
Description:	- Used to determine if you need to calculate the linkage disequilibrium decay based on the r2 metric for each generation.
Value:	- yes or no.
Usage:	-"OUTPUT_LD: no".
Type:	- Optional. Default is no.

GENOTYPES

Description:	- Used to determine if saving genotypes and at what generation to begin saving them. If you do not need the all the genotypes, this saves space and reduces running time.
Value:	- No or yes and if yes provide the generation at which to start exporting genotypes. Generation 0 is founder population.
Usage:	- "GENOTYPES: yes 0".
Type:	- Optional. Default is yes and starting at generation 0..

Incorporation of Future Modules

The overall goal of Geno-Diver is to be able to utilize either simulated or real data to see how current mating and selection strategies impact the performance, diversity and genetic load across the time horizon in the context of an individual herd or an entire population. To fulfill this goal multiple modules will be incorporated in the future including the use of external breeding value predictions, the use of advanced reproductive technologies, optimal contribution selection procedures and evolutionary algorithms. Furthermore, as more sophisticated methods get developed to identify and manage lethal recessive mutations and regions that give rise to inbreeding depression, these routines will be introduced into the simulation. A pictorial description is below.



● GDS: Genomic Diversity Simulator
● FIM: Functional Inbreeding Mapper
● HMD: Herd Mating Designer

Appendix I - Output Information

The folder that contains the information generated by the simulation program contains multiple files, and a description of each one is below.

Data Summary Files

File: Summary_Statistics_QTL

Generation: Generation number.
Quant_Founder_Start: Number of QTL from founder generation segregating.
Quant_Founder_Lost: Number of QTL from founder generation fixed.
Mutation_Quan_Total: Number of QTL from new mutations segregating.
Mutation_Quan_Lost: Number of QTL from new mutations fixed.
Additive_Var: True additive genetic variance based on Σ 2pq[a+d(q-p)]^2.
Dominance_Var: True dominance genetic variance based on Σ (2pqd)^2.
Fit_Founder_Start: Number of FTL from founder generation segregating.
Fit_Founder_Lost: Number of FTL derived from founder generation fixed.
Mutation_Fit_Total: Number of FTL derived from new mutations segregating.
Mutation_Fit_Lost: Number of FTL derived from new mutations fixed.
Avg_Haplotypes_Window: Mean haplotypes contained within a haplotype window
ProgenyDiedFitness: Number of progeny that died due to fitness.

File: Summary_Statistics_DataFrame_Inbreeding:

Generation: Generation number. ped_f: Mean pedigree based inbreeding parameter.
gen_f: Mean genomic relationship diagonal constructed based on Van Raden (2008).
h1_f: Mean diagonal of haplotype based relationship matrix (Hickey et al. 2012; H1).
h2_f: Mean diagonal of haplotype based relationship matrix (Hickey et al. 2012; H2).
h3_f: Mean diagonal of ROH based relationship matrix (Howard et al. Submitted).
homozy: Mean proportion homozygous (i.e. 1 - homozy = Observed Heterozygosity).
ExpHet: Expected Heterozygosity (i.e. Σ (1 - p^2 - q^2))
fitness: Mean multiplicative fitness value of an individual.
homozlethal: Mean number of homozygous FTL classified as lethal.
hetezlethal: Mean number of heterozygous FTL classified as lethal.
homozysublethal: Mean number of homozygous FTL classified as sub-lethal.
hetezsublethal: Mean number of heterozygous FTL classified as sub-lethal.
lethalequiv: Mean lethal equivalents (Lethal equivalents = Σ s for an animal).

File: Summary_Statistics_DataFrame_Performance:

Generation: Generation number.
phen: Mean (variance) phenotypic value.
ebv: Mean (variance) estimated breeding value.
gv: Mean (variance) true genotypic breeding value (Σ (a + d)).
bv: Mean (variance) true genotypic breeding value (Σ a).
dd: Mean (variance) true dominance deviation (Σ d).
res: Mean (variance) residual value.

File: LD_Decay:

A file that has the average correlation (r^2) between two SNP across a range of distances. The distances are in the first row and are in Kilobases. Each row after the first row corresponds to the generation, such that line 2 is generation 0, line 3 is generation 1, etc. The formula to calculate the D and (r^2) values is below and the subscript refers to either SNP marker 1 or 2.

$$D = (A_1B_1 \times A_2B_2) - (A_1B_2 \times A_2B_1)$$

$$r^2 = \frac{D^2}{p_1(1-p_1)p_2(1-p_2)}$$

Data Files

File: log_file.txt:

This file displays a great deal of information on specifics within each generation and it is advisable that one should look over it after you try a new simulation protocol.

File: Low_Fitness:

This file that describes the animals that died due to the fitness effect.

Sire: sire of dead progeny.

Dam: dam of dead progeny.

Fitness: Fitness value of individual.

QTL_Fitness: The genotypes for Fitness QTL for the individual.

File: Marker_Map:

chr: Chromosome location.
pos: Nucleotide position of marker.

File: Master_Genotypes:

A file that contains genotypic information for individuals that survived.
ID: Identification of individual.
Marker: Marker genotypes of individual (0-11; 2-22; 3-12; 4-21).
QTL: QTL genotypes of individual (0-11; 2-22; 3-12; 4-21).

File: QTL_new_old _Class:

A file that contains information on QTL effects and frequency across generations.
Chr: Chromosome location.
Pos: Nucleotide position of QTL.
Additive_Selective: If it is a QTL it refers to the additive effect and if it is a FTL it refers to the selection coefficient.
Dominance: The dominance effect for the QTL or degree of dominance for FTL.
Type: Refers to the type of loci (2 = quantitative trait; 4 = fitness lethal; 5 = fitness sub-lethal).
Gen: Generation at which the mutation occured.
Freq: Gene frequency across generations with a "_" as the delimeter.

File: Master_DataFrame:
A file that contains multiple statistics for individuals that survived.
ID: Identification of individual.
Sire: Sire Identification of individual.
Dam: Dam Identification of individual.
Sex: Sex of individual (0 = male and 1 = female).
Gen: Generation the animal was born.
Age: Age the animal was removed from the population either at the culling or selection stage.
Progeny: Number of progeny.
Dead: Number of dead progeny.
Ped_F: Pedigree based inbreeding metric.
Gen_F: Diagonal of genomic based relationship constructed based on Van Raden (2008).
Hap1_F: Diagonal of haplotype 1 based relationship matrix.
Hap2_F: Diagonal of haplotype 2 based relationship matrix.
Hap3_F: Diagonal of ROH based relationship matrix.
Homolethal: Number of homozygous lethal genotypes.
Heterlethal: Number of heterozygous lethal genotypes.
Homosublethal: Number of homozygous sub-lethal genotypes.
Hetersublethal: Number of heterozygous sub-lethal genotypes.
Letequiv: Lethal equivalent value.
Homozy: Proportion of the genome homozygous.
Fitness: Multiplicative Fitness value of the individual.
Phen: Phenotype.
EBV: Estimated Breeding Value.
Acc: Accuracy of EBV (not included yet).
GV: True genotypic value of individual (Σ (a + d)).
BV: True breeding value of individual (Σ a).
DD: True dominance deviation of individual (Σ d).
R: Residual value of individual.

Supplementary Files

File: CH*SNP.txt:
- Haplotype sequence for each chromosome simulated from MaCS.

File: MAP*.txt:
- map file corresponding to haplotypes sequence in CH*SNP.txt.

File: FounderGenotypes:
- Genotypes across chromosomes for each founder. The line number corresponds to the founder ID and the first column represents the row number of the two haplotypes that created the genotype, followed by the genotype string.

File: G_Matrix:
- The Genomic relationship matrix in binary format.

File: Ginv_Matrix:
- The inverse of the Genomic relationship matrix in binary format.

File: Linv_Matrix:
- The inverse of the cholesky matrix from the previous generation that is utilized to construct inverse relationship matrix using the method outlined by Meyer et al. (2013) to

obtain the inverse. This file is in binary format.

File: Pheno_GMatrix:

- Dataframe utilized in constructed genomic relationship matrix.

File: Pheno_Pedigree:

- Used in constructing pedigree relationship matrix.

File: Previous_Beta_PCG:

- Estimates of solutions for previous generation.

File: SNPFreq:

- Frequency of SNP across all chromosomes derived from MaCS.

Appendix II - Generation of Effects

The generation of effects for the quantitative and fitness traits are important parameters that can have a large impact on the simulation results. The methods to generate effects for both types of traits is similar to previous articles and methods other simulation programs have used. At the current time, the sampling of additive effects is from a gamma distribution and dominances effects are simulated from a normal distribution to generate the covariance between quantitative and fitness traits. The use of alternative distributions to generate effects and allow for covariance to occur between the two traits will occur in the future.

Quantitative Trait:

The additive effect (a), defined as half the difference in genotypic value between alternative homozygotes, is generated from a gamma distribution. The default parameters for the gamma distribution(0.4,1.66) result in an L-shaped distribution of QTL effects and implies that the majority of effects are small and a few have large effects. The gamma distribution only generates positive values, therefore, with equal probability, one of the two alleles was chosen to be positive or negative based on a binomial distribution (p = 0.5).

The dominance effect, defined as the deviation of the value of the heterozygote from the mean of the two homozygotes, was generated using a multistep procedure. Independence between additive and dominance effects is the classical treatment (Falconer & Mackay, 1996) and it is convenient because it allows orthogonally of the additive and dominance estimates. However, this independence is contradictory with the phenomena of inbreeding depression and hybrid vigor that indicates dominance is directional (Lynch & Walsh, 1998) and results from real data (Wellmann & Bennewitz 2011; Wellmann & Bennewitz 2012), which suggest an a priori dependency between additive and dominance effects. Therefore, the degree of dominance (h) is sampled from a normal distribution, which allows for the user to vary the proportion of positive or negative dominance effect by altering the mean. Next, dominance effects (d) were generated by multiplying the degree of dominance by the absolute value of the additive effect ($d = h|a|$). The use of this simulation method results in the additive and dominance effects to now be dependent on each other. Lastly, the choice of parameters specifying the normal distribution and the minor allele frequency for the quantitative QTL has an impact on the proportion of dominance effects that display partial or over-dominance. The proportion that displays partial or over-dominance is outlined in the log file near the beginning of the log file.

Fitness Trait:

The generation of fitness effects was divided into lethal and sub-lethal genetic architectures to allow for full flexibility. The distribution of fitness effects and their associated frequency in the genome have been hypothesized to come from two competing results from the literature. The first one is based on the results obtained by (Mukai et al., 1972) and is what we called the "Mukai scenario", where mutations are assumed to be numerous and of small effect. The second hypothesis is based on more recent results from mutation-accumulation studies and assume that mutations are considerable less frequent but of larger effect (Caballero & Keightley, 1994; Garcia-Dorado & Caballero, 2000). For both lethal and sub-lethal FTL the fitness was defined as relative fitness and is parameterized by two coefficients and they include the selection coefficient (s) and the dominance coefficient (h). The s value measures how much worse the unfit allele is, compared to the fittest allele. The h value measures the degree of dominance that the heterozygote shows regarding the reduced fitness compared to the unfit homozygote (Wright 1931). The normalization procedure forces the fittest homozygote genotype to have a value of 1, and the other homozygote genotype has a value of 1 - s. Lastly, heterozygote genotypes have a fitness value of 1 - hs.

The selection coefficient was generated from a gamma distribution with different parameters for lethal and sublethal. The logfile outlines the mean selection coefficient for the lethal and sub-lethal FTL. As a reference when altering the shape and scale parameter, the mean of a gamma distribution is the shape X scale.

The dominance coefficient was generated from a normal distribution with different parameters for the lethal and sublethal. The absolute value of the sample is taken as the dominance coefficient. The logfile outlines the mean dominance coefficient for the lethal and sub-lethal FTL. As a reference when altering the shape and scale parameter, the mean of a gamma distribution is the shape X scale.

The fitness of an individual was then calculated as the multiplicative effect of each fitness genotype across both lethal and sub-lethal FTL with a maximum value of 1 and minimum of 0. A value closer to 1 has a higher fitness and is more likely to survive. An animal survived if their individual fitness value was larger than a random number from a uniform distribution (0,1) if it was lower the animal died.

Covariance Between Traits:

The correlation between the quantitative trait and the fitness trait can be due linkage or pleiotropy. Setting the COVAR parameters both to 0 results in linkage to be the only possible source of correlation between fitness and quantitive traits. Setting the COVAR parameters to a value greater than 0 results in a pleiotropic correlation between the additive effects for the quantitative trait and the selection coefficient for the sub-lethal fitness traits. The scaling of quantitative traits results in the additive effects for the quantitative trait to change and therefore covariance was generated based on Trivariate Reduction algorithm. The Trivariate Reduction algorithm only allows the correlation to be positive. For example, high values for the quantitive trait would result in the two traits being antagonistic based on a positive correlation. One just needs to change the favorable direction of the quantitative trait to alter the interpretation.

Trivariate Reduction for Gamma1 (a_1,b_1) and Gamma2 (a_2,b_2)
- Correlation (ρ) Bounded between: $0 \leq \rho \leq \min(a_1,a_2) / \sqrt{a_1 a_2}$.
- Steps:
1.) Generate $Y_1 \sim$ gamma($a_1 - \rho\sqrt{a_1 a_2}$,1)
2.) Generate $Y_2 \sim$ gamma($a_2 - \rho\sqrt{a_1 a_2}$,1)
3.) Generate $Y_3 \sim$ gamma($\rho\sqrt{a_1 a_2}$,1)
4a.) Generate Value for Gamma1: $b_1(Y_1 + Y_3)$
4b.) Generate Value for Gamma2: $b_2(Y_2 + Y_3)$

The Y_3 value generate the covariance between the two traits. For FTL that have a covariance with the quantitative trait the Y_2 value is sampled for each FTL within an iteration and the rank correlation is calculated. Once the rank correlation gets within a 1.5 percent of the value specified it then generates the selection coefficient and dominance values using the current iterations Y_2 values.

Appendix III - MaCS Sequence Simulation

The MaCS program (Markovian Coalescence Simulator; Chen et al. 2009) generates the founder genome. Before using the program, it is advisable to understand the coalescent process and a good review paper is Hudson (1991) and Chapter 5 of Charlesworth & Charleworth (2010). We have chosen to employ a coalescent simulator (specifically MaCS) in this step due to the flexibility of the approach in generating haplotype sequences for a wide range of population scenarios in terms of the size and structure of the ancestral population across time and genome scenarios with varying mutation and recombination rates. We have chosen the default scenarios to resemble options specified by AlphaSim (Hickey & Gorjanc, 2012) as they represent typical agricultural species LD patterns.

There are 5 default scenarios that represent a range of linkage disequilibrium (LD) patterns that can be called within the simulation, as outlined in the figure below. The five scenarios are called by specifying either "Ne70", "Ne100_Scen1", "Ne100_Scen2", "Ne250" or "Ne1000" after the FOUNDER_Effective_Size parameter in the parameter file. The user can input a custom effective population size and historical population parameters by using "CustomNe" as the parameter. An easy way to generate custom parameters for MaCS is to utilize a default scenario that resembles the pattern you are wanting and to change the effect population size of the population and determine how the LD pattern changes. If this is specified the program looks for a file called "CustomNe" within the folder where the execution of the program occurred. The file should contain two rows, with the first one being the effective population size parameter and the last one being the historical population size parameters. Lastly, if the program only reads an integer value, then the value is the effective population size with no population history.

The generation of sequence information may take some time to compute. The files generated from the program may be large. Due to this, it is advisable only to generate sequence data once for a given scenario and then adjust narrow-sense heritability, broad sense heritability, selection or mating parameters and start with generating the founder generation.

The default scenarios are below. An illustration of the LD decay associated with each scenario is on the following page:

Ne70:

- Effective population size = "70".
- Historical population parameters: "-eN 0.18 0.71 -eN 0.36 1.43 -eN 0.54 2.14 -eN 0.71 2.86 -eN 0.89 3.57 -eN 1.07 4.29 -eN 1.25 5.00 -eN 1.43 5.71".

Ne100_Scen1:

- Effective population size = "100".
- Historical population parameters: "-eN 0.06 2.0 -eN 0.13 3.0 -eN 0.25 5.0 -eN 0.50 7.0 -eN 0.75 9.0 -eN 1.00 11.0 -eN 1.25 12.5 -eN 1.50 13.0 -eN 1.75 13.5 -eN 2.00 14.0 -eN 2.25 14.5 -eN 2.50 15.0 -eN 5.00 20.0 -eN 7.50 25.0 -eN 10.00 30.0 -eN 12.50 35.0 -eN 15.00 40.0 -eN 17.50 45.0 -eN 20.00 50.0 -eN 22.50 55.0 -eN 25.00 60.0 -eN 50.00 70.0 -eN 100.00 80.0 -eN 150.00 90.0 -eN 200.00 100.0 -eN 250.00 120.0 -eN 500.00 200.0 -eN 1000.00 400.0 -eN 1500.00 600.0 -eN 2000.00 800.0 -eN 2500.00 1000.0".

Ne100_Scen2:

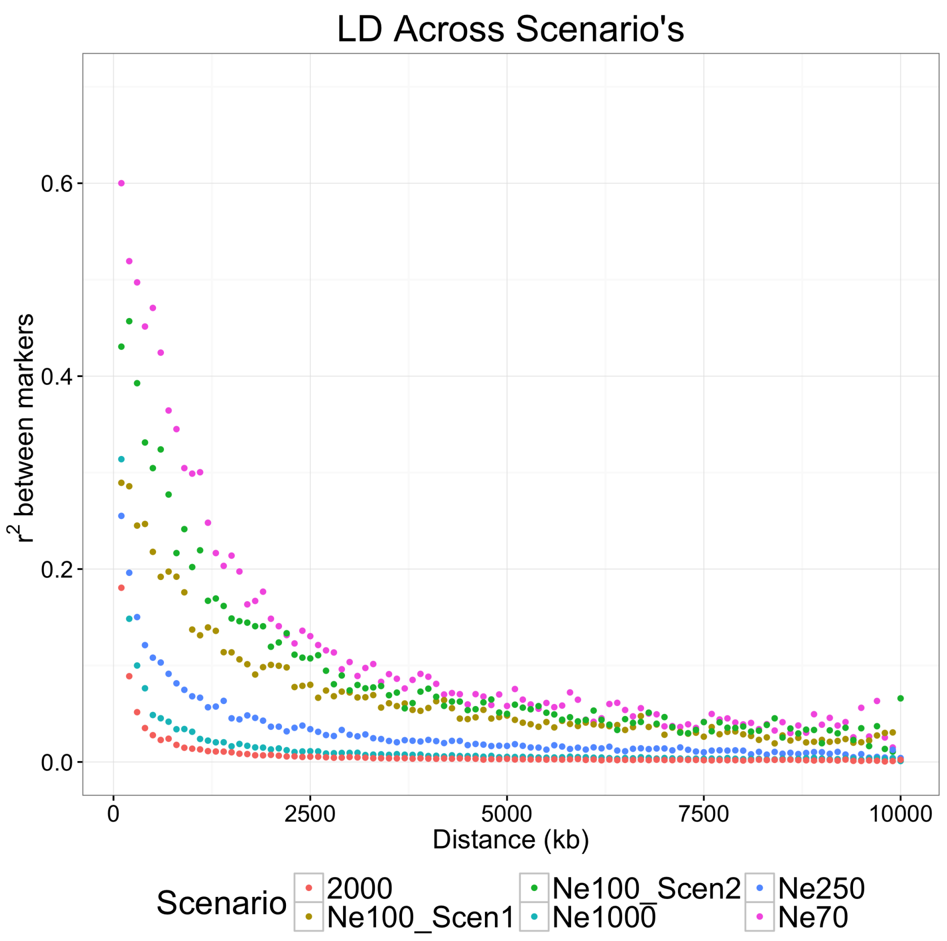
- Effective population size = "100".
- Historical population parameters: "-eN 50.00 200.0 -eN 75.00 300.0 -eN 100.00 400.0 -eN 125.00 500.0 -eN 150.00 600.0 -eN 175.00 700.0 -eN 200.00 800.0 -eN 225.00 900.0 -eN 250.00 1000.0 -eN 275.00 2000.0 -eN 300.00 3000.0 -eN 325.00 4000.0 -eN 350.00 5000.0 -eN 375.00 6000.0 -eN 400.00 7000.0 -eN 425.00 8000.0 -eN 450.00 9000.0 -eN 475.00 10000.0".

Ne250:

- Effective population size = "250".
- Historical population parameters: "-eN 0 1.04 -eN 0 1.08 -eN 0 1.12 -eN 0 1.16 -eN 0.01 1.2 -eN 0.03 1.6 -eN 0.05 2.0 -eN 0.1 2.8 -eN 0.2 4.8 -eN 0.3 5 -eN 0.4 5.2 -eN 0.5 5.4 -eN 0.6 5.6 -eN 0.7 5.7 -eN 0.8 5.8 -eN 0.9 5.9 -eN 1 6 -eN 1 4 -eN 2 8 -eN 3 10 -eN 4 12 -eN 5 14 -eN 6 16 -eN 7 18 -eN 8 20 -eN 9 22 -eN 10 24 -eN 20 28 -eN 40 32 -eN 60 36 -eN 80 40 -eN 100 48 -eN 200 80 -eN 400 160 -eN 600 240 -eN 800 320 -eN 1000 400".

Ne1000:

- Effective population size = "1000".
- Historical population parameters: "-eN 0.50 2.00 -eN 0.75 2.50 -eN 1.00 3.00 -eN 1.25 3.20 -eN 1.50 3.50 -eN 1.75 3.80 -eN 2.00 4.00 -eN 2.25 4.20 -eN 2.50 4.50 -eN 5.00 5.46 -eN 10.00 7.37 -eN 15.00 9.28 -eN 20.00 11.19 -eN 25.00 13.10 -eN 50.00 22.66 -eN 100.00 41.77 -eN 150.00 60.89 -eN 200.00 80.00".


LD Across Scenario's
r^2 between markers
0.0
0.2
0.4
0.6
0
2500
5000
7500
10000
Distance (kb)
Scenario
2000
Ne100_Scen1
Ne100_Scen2
Ne1000
Ne250
Ne70

Literature Cited

- Aguilar, I., I. Misztal , A. Legarra & S. Tsuruta. 2011. Efficient computation of the genomic relationship matrix and other matrices used in single-step evaluation. J. Anim. Breed. Genet. 128:422-428.

- Ai H., L. Huang, J. Ren. 2013. Genetic diversity, linkage disequilibrium and selection signatures in chinese and Western pigs revealed by genome-wide SNP markers. PLoS One 8(2):e56001.

- Caballero A. & P. Keightley. 1994. A pleiotropic nonadditive model of variation in quantitative traits. Genetics 138:883-900.

- De los Campos, G., J. M. Hickey, R. Pong-Wong, H. D. Daetwyler, & M. P. L. Calus. 2013. Whole-Genome Regression and Prediction Methods Applied to Plant and Animal Breeding. Genetics, 193, 327-345.

- Charlesworth, B., & D. Charlesworth. 2010. Elements of Evolutionary Genetics. Greenwoord Village, Colorado, USA: Roberts and Company.

- Chen, G. K., P. Marjoram & J. D. Wall. 2009. Fast and flexible simulation of DNA sequence data. Genome Res. 19(1):136-42.

- Cheng H., D. J. Garrick, R. L. Fernando. 2015. XSim: simulation of descendants from ancestors with sequence data. G3 5:1415-1417.

- Falconer D. S. & T. F. S. Mackay. Introduction to quantitative genetics. 4th ed. New York, NY: Longman Scientific and Technical; 1996.

- Garcia-Dorado, A. & A. Caballero. 2000. On the average degree of dominance of deleterious spontaneous mutations. Genetics, 155, 1991?2001.

- Hickey, J. M., & G. Gorjanc. 2012. Simulated Data for Genomic Selection and Genome-Wide Association Studies Using a Combination of Coalescent and Gene Drop Methods. G3 2:425-427.

- Howard J. T., F. Tiezzi1, Y. Huang, K.A. Gray & C. Maltecca. The use of alternative genomic metrics in nucleus herds to manage the diversity of purebred and crossbred animals. Genet. Sel. Evol. 48:91.

- Hudson R. R. 1990. Gene genealogies and the coalescent process. Oxford Surveys in Evolutionary Biology. 7:1-45.

- Jonas, E. & D. -J. de Koning. 2015. Genomic Selection Needs to Be Carefully Assessed to Meet Specific Requirements in Livestock Breeding Programs. Front. Genet., 6, 49.

- Lynch, M. & B. Walsh. 1998. Genetics and analysis of quantitative traits. Vol. 1. Sunderland, MA: Sinauer.
- McKay, S. D., R. D. Schnabel, B. M. Murdoch, L. K. Matukumalli, J. Aerts, W. Coppieters, D. Crews, E. Dias Neto, C. A. Gill, C. Gao, H. Mannen, P. Stothard, Z. Wang, C. P. Van Tassell, J. L. Williams, J. F. Taylor, S. S. Moore. 2007. Whole genome linkage disequilibrium maps in cattle. BMC Genet. 8:74.
- McMahon, B. J., E. C. Teeling, J. Hoglund. 2014. How and Why Should We Implement Genomics into Conservation? Evol. Appl., 7, 999-1007.
- Meuwissen, T. H. E., & Z. Luo. 1992. Computing inbreeding coefficients in large populations. Genet Sel Evol. 24(4): 305-313.
- Meyer, K., B. Tier & H. U. Graser. 2013. Technical note: Updating the inverse of the genomic relationship matrix. J. Anim. Sci. 91:2583-2586.
- Misztal, I., A. Legarra & I. Aguilar. 2014. Using recursion to compute the inverse of the genomic relationship matrix.
- Morrell, P. L., E. S. Buckler, J. Ross-Ibarra. 2012. Crop genomics: advances and applications. Nat. Rev. Genet. 13, 85-96.
- Mukai, T., S. I. Chigusa, L. E. Mettler, J. F. Crow. 1972. Mutation rate and dominance of genes affecting viability in drosophila melanogaster. Genetics 72:333-355.
- Pérez-Enciso M., & A. Legarra. 2016. A combined coalescence gene-dropping tool for evaluating genomic selection in complex scenarios (ms2gs). J. Anim. Breed. Genet. 133(2):85-91.
- Porto-Neto L. R., T. S. Sonstegard, G. E. Liu, D. M. Bickhart, M. V. Da Silva, M. A. Machado, Y. T. Utsunomiya, J. F. Garcia, C. Gondro, C. P. Van Tassell. 2013. Genomic divergence of zebu and taurine cattle identified through high-density SNP genotyping. BMC Genomics. 14:876.
- Sargolzaei, M. & F. S. Schenkel. 2009. QMSim: a large-scale genome simulator for livestock. Bioinformatics, 25, 680-1.
- VanRaden P. M. 2008. Efficient Methods to Compute Genomic Predictions. J. Dairy Sci., 91, 4414-4423.
- VanRaden P. M., K. M. Olson, D. J. Null, J. L. Hutchison. 2011. Harmful recessive effects on fertility detected by absence of homozygous haplotypes. J. Dairy Sci., 94, 6153-6161.

- Wellmann, R. & J. Bennewitz. 2011. The contribution of dominance to the understanding of quantitative genetic variation. Genet Res (Camb) 93:139?154.

- Wellmann, R. & J. Bennewitz. 2012. Bayesian models with dominance effects for genomic evaluation of quantitative traits. Genet Res (Camb) 94:21?37.

- Wright, S. 1931. Evolution in Mendelian populations. Genetics 16:97-159.

- Yang, J., B. Benyamin, B. P. McEvoy, S. Gordon, A. K. Henders, D. R. Nyholt, P. A. Madden, A. C. Heath, N. G. Martin, G. W. Montgomery, M. E. Goddard, P. M. Visscher. 2010. Common SNPs explain a large proportion of the heritability for human height. Nat Genet., 42, 565-569.

Example 1 (Quantitative Trait - Single Progeny)

```
--------|     Example 1 Parameter File      |--------
--------|        Starting Parameters        |--------
START: sequence
--------| Genome and Marker Information |--------
CHR: 3
CHR_LENGTH: 150 150 150
NUM_MARK: 4000 4000 4000
QTL: 50
--------|        Population Parameters       |--------
FOUNDER_Effective_Size: Ne70
VARIANCE_A: 0.35
VARIANCE_D: 0.05
--------| Selection and Mating Parameters |--------
GENERATIONS: 10
INDIVIDUALS: 50 0.2 400 0.2
PROGENY: 1
SELECTION: ebv high
SOLVER_INVERSE: genomic
MATING: random5
CULLING: ebv 5
```

The simulation starts by creating sequence data for 3 chromosomes. The genome simulated has a high degree of short-range LD (Ne70). This type of effective population size scenario does not create a lot of available markers. Due to this, the SNP panel only contained 12,000 marker (i.e. 4,000 markers per chromosome). For each chromosome, the genome contained 50 randomly placed QTL. The genome does not contain any FTL. The narrow and broad sense heritability for the trait is 0.35 and 0.40, respectively. The phenotypic variance is by default set at 1.0 and therefore the residual variance is 0.6. For each generation, a total of 50 males and 400 females are in the population. A total of 10 and 80 (0.2 replacement rate) male and female parents, respectively, are culled and replaced by new progeny each generation. Animals with a high estimated breeding value (EBV) were selected or culled each generation. The EBV are solved using an animal model based on a genomic relationship matrix. Lastly, each mating pair produced 1 progeny. The number of generations simulated is 10.

Overview of Results:

- Before looking at any results or while the program is running go into the directory "GenoDiverFiles" and look over what parameters got called at the beginning of the log file.
- When simulating dominance be sure to check to see if the number that display over-dominance or partial dominance is what you were wanting. In this case with the given narrow and broad sense heritability, the proportion of QTL displaying over-dominance is small (around 15 %).
- The table below outlines the change in multiple parameters as the generations proceed and were all found in the following files:
  - "Summary_Statistics_DataFrame_Inbreeding".
  - "Summary_Statistics_DataFrame_Performance".
- The values outlined below may be a little different than yours because I did not specify a seed.

Generation	Phenotype	EBV	Pedigree Inbreeding	Genomic Inbreeding
0	-0.041	0.000	0	0.994
1	0.460	0.436	0	1.007
2	0.644	0.635	0.00063	1.022
3	0.883	0.854	0.00016	1.039
4	1.131	1.095	0.00231	1.062
5	1.371	1.312	0.00432	1.080
6	1.534	1.530	0.01239	1.133
7	1.723	1.699	0.01072	1.151
8	1.925	1.896	0.01267	1.186
9	2.043	2.072	0.01880	1.229
10	2.190	2.238	0.01634	1.257

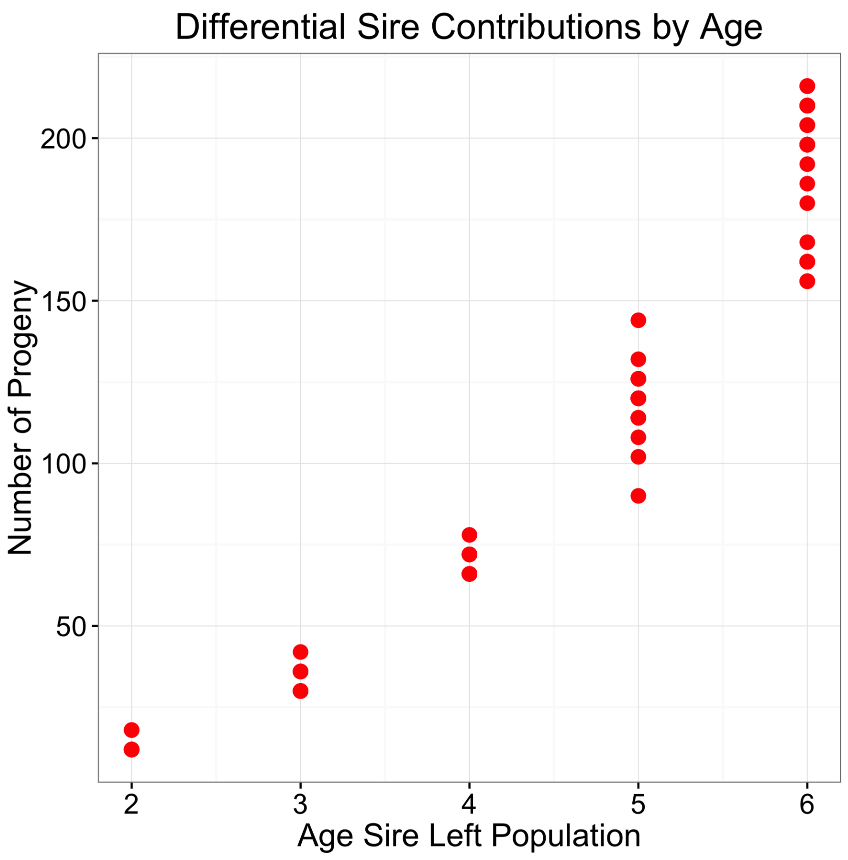
Example 2 (Quantitative Trait - Multiple Progeny)

```
−−−−−−−|     Example 2 Parameter File      |−−−−−−−
−−−−−−−|         Starting Parameters         |−−−−−−−
START: sequence
−−−−−−−| Genome and Marker Information |−−−−−−−
CHR: 3
CHR_LENGTH: 150 150 150
NUM_MARK: 4000 4000 4000
QTL: 50
−−−−−−−|         Population Parameters       |−−−−−−−
FOUNDER_Effective_Size: Ne70
VARIANCE_A: 0.35
VARIANCE_D: 0.05
−−−−−−−| Selection and Mating Parameters |−−−−−−−
GENERATIONS: 10
INDIVIDUALS: 50 0.2 300 0.2
PROGENY: 6
PARITY_MATES_DIST: 2.0 1.0
MAXFULLSIB: 2
SELECTION: ebv high
SOLVER_INVERSE: pedigree
MATING: random125
CULLING: ebv 5
```

The simulation from Example 2 is similar concerning its genome, markers and genetic architecture to Example 1. A few of the added complexities of the current simulation is that the number of progeny per mating pair is 6 and therefore to limit selecting entire families the MAXFULLSIB parameter is 2. Furthermore, the number of matings allowed to a sire is dependent on the age of the animal and in the current setting older animals get a larger proportion of the total number of matings compared to younger animals. For each generation, a total of 50 males and 300 females are in the population. The mating design is avoiding matings with parental co-ancestries greater than 0.125 and mating below that at random.

Overview of Results:

- The log file from lines 139 to 160 describe the Beta distribution (2.0, 1.0) that generated the number of matings given to a specific age group. As outlined in the figure below, as an animals ages (approaches the max value 1.0) a greater proportion of the distribution contains that region.
- Furthermore, within each generation, the number of siblings selected with each family is outlined in the log file. The use of a simple Linux ”sed” statement can be utilized to grab the section of the log file that are important to the user.
- The figure below depicts the number of offspring the sire produced as a function of when the sire left the population and generated from data contained in the “Master_DataFrame” file. As shown below the number of offspring produced as a function of when the sire left the herd is not linear due to the particular Beta distribution utilized (2.0, 1.0).



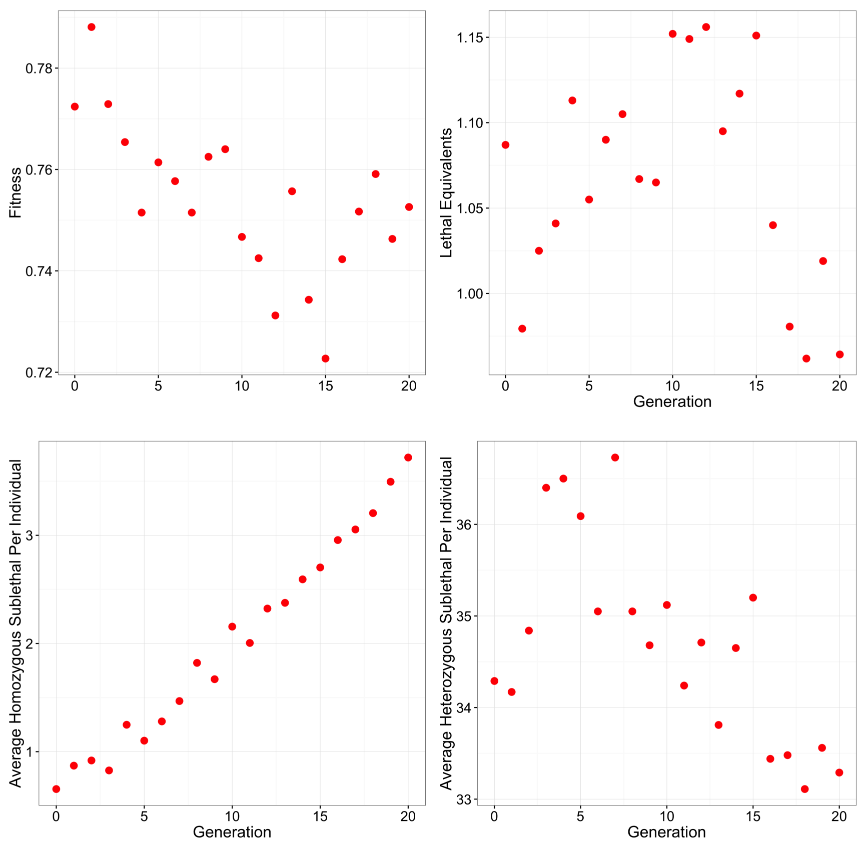
Example 3 (Quantitative Trait + Fitness Correlated)

```
--------|      Example 3 Parameter File      |--------
--------|        Starting Parameters         |--------
START: sequence
THREAD: 4
--------| Genome and Marker Information |--------
CHR: 3
CHR_LENGTH: 150 150 150
NUM_MARK: 4000 4000 4000
QTL: 100
FIT_LETHAL: 25
FIT_SUBLETHAL: 100
--------|        Population Parameters       |--------
FOUNDER_Effective_Size: Ne70
VARIANCE_A: 0.20
VARIANCE_D: 0.05
COVAR: 0.5 0.2
--------| Selection and Mating Parameters |--------
GENERATIONS: 20
INDIVIDUALS: 50 0.2 250 0.2
PROGENY: 1
SELECTION: ebv high
SOLVER_INVERSE: pedigree
MATING: random
CULLING: ebv 5
```

This simulation highlights how a quantitative and a fitness trait can be simulated with a certain proportion (i.e. 50 %) of the loci have both a quantitative and fitness effect. The simulation from Example 3 is similar regarding its genome, markers and genetic architecture to Example 2.

Overview of Results:

- The number of progeny that died due to fitness and the number of the FTL purged from the population is in Summary_Statistics_QTL file. Furthermore, some of the results are displayed graphically below.
- The log file contains the mean selection coefficient and degree of dominance for the lethal and sublethal FTL.
- When simulating fitness effects, a certain proportion of the progeny will die due to fitness and if the population does not have enough progeny to stay at the value given the program exits. Therefore, careful consideration of the number and magnitude of fitness effects along with the number of progeny produced per mating pair needs to be carefully considered when constructing the parameter file.

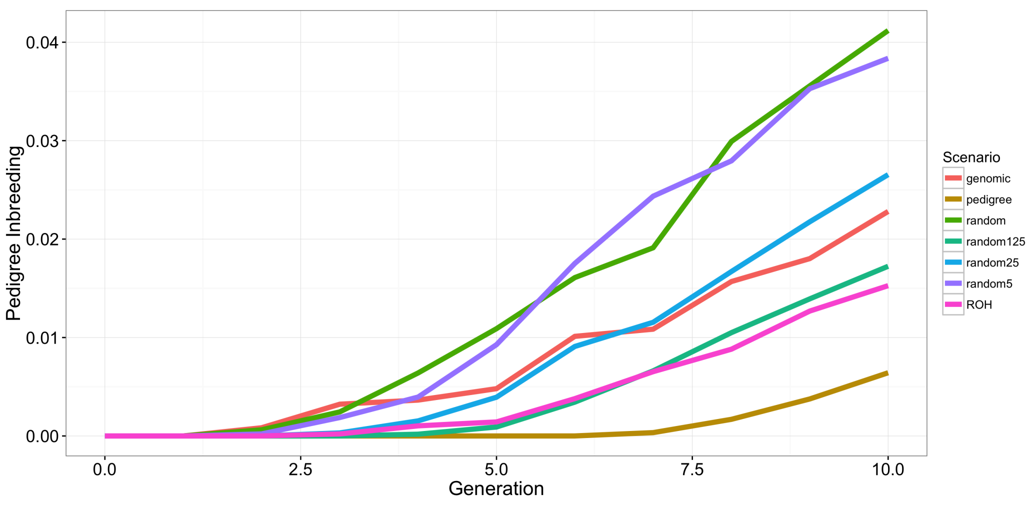


Example 4 (Quantitative Trait Minimize Inbreeding)

```
--------|     Example 4 Parameter File      |--------
--------|         Starting Parameters         |--------
START: sequence
THREAD: 4
--------| Genome and Marker Information |--------
CHR: 3
CHR_LENGTH: 150 150 150
NUM_MARK: 4000 4000 4000
QTL: 100
--------|         Population Parameters        |--------
FOUNDER_Effective_Size: Ne70
VARIANCE_A: 0.35
VARIANCE_D: 0.05
--------| Selection and Mating Parameters |--------
GENERATIONS: 10
INDIVIDUALS: 50 0.2 600 0.2
PROGENY: 1
SELECTION: ebv high
SOLVER_INVERSE: pedigree
MATING: random
CULLING: ebv 5
```

Outlined above is the initial parameter file and is very similar to Example 2. A bash script outlined below is used to change parameters in the parameter file across multiple scenarios and then output the results to a directory. Across all simulations, they use the same founder sequence by first starting the random mating simulation from the sequence and then all remaining simulations started from the founder generation. Furthermore, the same seed was utilized across all scenarios and therefore the only thing that changes across simulation replicates was how animals were mated. The graph below highlights the change across generations in pedigree across scenarios.

```
# Delete old parameter file and director to place files if their
rm -rf ./Example4.txt | | true
rm -rf ./Example4_Output | | true
mkdir Example4_Output

# run Geno Driver using the initial parameter file
./GenoDiver Example4.txt
# Move inbreeding folder to permanent location
mv ./GenoDiverFiles/Summary_Statistics_DataFrame_Inbreeding ./Example4_Output/inbreeding_random

# Now do random 5 and start with founder
sed -i '/START: sequence/c\START: founder' Example4.txt
sed -i '/MATING: random/c\MATING: random5' Example4.txt
./GenoDiver Example4.txt
mv ./GenoDiverFiles/Summary_Statistics_DataFrame_Inbreeding ./Example4_Output/inbreeding_random5

# Now do random 25
sed -i '/MATING: random5/c\MATING: random25' Example4.txt
./GenoDiver Example4.txt
mv ./GenoDiverFiles/Summary_Statistics_DataFrame_Inbreeding ./Example4_Output/inbreeding_random25

# Now do random 125
sed -i '/MATING: random25/c\MATING: random125' Example4.txt
./GenoDiver Example4.txt
mv ./GenoDiverFiles/Summary_Statistics_DataFrame_Inbreeding ./Example4_Output/inbreeding_random125

# Now do minimize pedigree
sed -i '/MATING: random125/c\MATING: minPedigree' Example4.txt
./GenoDiver Example4.txt
mv ./GenoDiverFiles/Summary_Statistics_DataFrame_Inbreeding ./Example4_Output/inbreeding_pedigree

# Now do minimize genomic
sed -i '/MATING: minPedigree/c\:MATING minGenomic' Example4.txt
./GenoDiver Example4.txt
mv ./GenoDiverFiles/Summary_Statistics_DataFrame_Inbreeding ./Example4_Output/inbreeding_genomic

# Now do minimize genomic
sed -i '/MATING: minGenomic/c\MATING: minROH' Example4.txt
./GenoDiver Example4.txt
mv ./GenoDiverFiles/Summary_Statistics_DataFrame_Inbreeding ./Example4_Output/inbreeding_ROH

# Change back to original Example4.txt parameter file
sed -i '/MATING: minROH/c\MATING: random' Example4.txt
```


0.04
0.03
0.02
0.01
0.00
Pedigree Inbreeding
0.0
2.5
5.0
7.5
10.0
Generation
Scenario
genomic
pedigree
random
random125
random25
random5
ROH