GeneEvolve Documentation

Rasool Tahmasbi
Institute for Behavioral Genetics,
University of Colorado, Boulder,
USA
Rasool.Tahmasbi@Colorado.edu

Matthew C. Keller
Institute for Behavioral Genetics,
University of Colorado, Boulder,
USA
matthew.c.keller@colorado.edu

version 2.0.0
Oct. 2017

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 What is *GeneEvolve*?

GeneEvolve is C++ code for simulating sequence- or SNP-level genetic data over large genomic regions in large populations, using an object-oriented approach. This allows compiling *GeneEvolve* on any computer platform, which supports standard C++ compiler.

Computer simulations are excellent tools for understanding the evolutionary and genetic consequences of complex processes that cannot be analytically derived and for creating realistic, individual-level genetic data . Unlike coalescent [5] based simulators, *GeneEvolve* runs forward-in-time, which allows it to provide a wide range of scenarios for selection, population size and structure, migration, recombination and familial effects. Forward time simulators also allows users to estimate population genetic parameters from among an entire population at any point in evolution rather than just from individuals who leave descendants.

GeneEvolve is fast and memory efficient simulator that can handle complex evolutionary scenarios and provides realistic, individual-level genetic data to the user.

1.2 How to download *GeneEvolve*

GeneEvolve can be download from https://github.com/rtahmasbi/GeneEvolve. The source codes, examples and this documentation are available freely for downloading.

There are three ways to download it:

1. Suggested method: You can run the following commands in the terminal:

```
git clone https://github.com/rtahmasbi/GeneEvolve
cd GeneEvolve
```

2. Or, go to https://github.com/rtahmasbi/GeneEvolve and then push the green button "Clone or Download" and then "Download ZIP".

3. Or, you can run the following commands:

```
wget https://github.com/rtahmasbi/GeneEvolve/archive/master.zip
unzip master
cd GeneEvolve-master
```

Note that for the Linux and Mac users, we added the compiled files (`GeneEvolve_Linux` and `GeneEvolve_Mac`) in the main directory.

1.3 How to compile *GeneEvolve*

GeneEvolve can be run on different platforms. We have included the compiled files in the main directory (`GeneEvolve_Linux` and `GeneEvolve_Mac`), but the source codes are also available and you can compile them for different platforms. After downloading it, you should go to its root directory (`GeneEvolve` or `GeneEvolve-master`) and simply type `make`. After successfully compiling, the `GeneEvolve` simulator will be in the `bin` subdirectory:

```
make
cd bin
./GeneEvolve --help
```

If you decide to compile *GeneEvolve* yourself, note that it depends on the following compilers or libraries:

```
GCC 4.8.1 # or higher versions
libStatGen
Eigen # http://eigen.tuxfamily.org/
```

If GCC is not installed on your computer, then the program will not compile, so you need to install it first. You can download GCC from https://gcc.gnu.org/. `libStatGen` and `Eigen` are with *GeneEvolve*.

While compiling, there may be several warnings from `libStatGen` source codes. These can be ignored. If you get the following error, it simply means you have not installed "OpenMP" for multithreading programming. This is not currently necessary and you can also ignore this error. *GeneEvolve* does not currently work with multithreading, although we are working on having it do so in future versions.

```
clang: error: unsupported option '-fopenmp'
make[1]: *** [../obj/omp/Simulation.o] Error 1
```

1.4 How to run *GeneEvolve*

GeneEvolve is a stand-alone program. Our aim was to make it simple and user friendly for different levels of user's knowledge. Users can create complex evolutionary and life history events by adding

and combining parameters. The minimum required parameters are *generation information* (such as population size, spousal correlation for mating, offspring distribution and selection function per generation), *haplotype information* (a file of phased haplotypes for the starting generation), *recombination map*, *cv list* (position of causal variants (CVs) and their additive and dominance effects), and the actual *CV's haplotype* (a file of phased haplotypes that only contains the CVs for the founder generation). These required parameters are listed below. Detailed explanations for each parameter are in the following chapters.

```
./GeneEvolve --file_gen_info [file] \
--file_hap_name [file] \
--file_recom_map [file] \
--file_cv_info [file] \
--file_cvs [file]
```

1.4.1 Required Parameters

The parameters shown above and listed below are all required, and are paths to files, the formats of which are described in detail in Chapters 2 and 4. Below, we briefly describe the information contained in each file.

`--file_gen_info` - A file with one row per generation plus an additional header row at the top. This file provides information about the population size, mating correlation, distribution of offspring per mating pair, and natural selection for each generation.

`--file_hap_name` - A file with one row per chromosome plus an additional header row at the top. Each row contains the paths to files containing phased haplotypes that will be used as the founder population at time 0. Currently, *GeneEvolve* uses as input the SHAPEIT file format (hap, legend, and indiv files); in the future, we will incorporate VCF file formats. For those wishing to use real sequence or SNP data as input (which we anticipate is the usual case), we recommend that you first phase your data using SHAPEIT or a similar program and then convert the files to SHAPEIT format. A number of programs can be used to convert from various phased file formats to SHAPEIT format (e.g., bcftools; https://samtools.github.io/bcftools/bcftools.html) For more information on using SHAPEIT, see the SHAPEIT website. Users wishing to use simulated data as input will need to write custom scripts to convert their data to these formats, which are described in more detail in Appendix C.

`--file_recom_map` - This file is used to calculate the probability of recombination events along the chromosomes. It contains physical distances (bp) and map distances (cM) at equal bp intervals along all chromosomes. Two such files are included in the Examples folder (one at 10 Kb intervals and another at 50 Kb intervals). They were created from deCODE recombination estimates (http://www.decode.com/addendum/). These files can be used for realistic human genome-wide simulations, although users are free to supply their own.

`--file_cv_info` - A file with one row per causal variant (CV) that describes the location and

additive and dominant effects of each CV.

`--file_cvs` - This is exactly like `--file_hap_name`, but contains information on phased haplotypes with varaints restricted to CVs rather than containing all variants. Thus, the legend/hap/indiv files to which this file points are necessarily subsets of the full genotype legend/hap/indiv files.

1.4.2 Additional Parameters

By default, var$[E] = 1$ and var$[A]$ and var$[D]$ are computed based on the additive and dominance effect sizes and the allele frequencies inputted by option `--file_cv_info`. However, a user can scale var$[A]$, var$[D]$ and var$[E]$ to any positive number. To do so (to control the actual variance of additive and dominance effects rather than allowing these to be consequences of the allelic effects and minor allele frequencies), user can use the following optional parameters
`--va [number] --vd [number]`

Similarly, for the unique, familial, shared sibling (common), and population specific environmental effects, user can specify the following parameters, respectively:
`--ve [number] --vf [number] --vc [number] --gamma [number]`

Additional optional parameters such as random mating, selection functions, migration and so forth, are described in Chapters 2 and 4. All the parameters and their default value are listed in Chapter 4.

1.5 Quick start

In this section, we present examples of simulating different evolutionary/life history scenarios. All required files are available in `Examples` directory. To run *GeneEvolve*, we first need phased genotype data in the SHAPEIT format. In this directory, we have simulated genotypes for 3 chromosomes. Clearly, real genotype datasets are typically much larger than this simulated data, but the provided data is useful for demonstration purposes.

1.5.1 Example 1 – Simple drift

In this example, we simulate a population with a founder population size of 2000 (number of rows in the `ref.chr1.indv` file that the `ex1.pop1.hap_sample_adress.txt` points to) and increase this to size 3000 for 10 generations (see `ex1.popinfo.txt` file below, which contains population information for each generation). Although a simple example, this type of simulation is useful for creating an independent genetic dataset that has the same genetic properties (MAF, LD, SNP density, etc.) as the founder population. Note, however, that the degree of relatedness will be considerably higher in your simulated samples unless you simulate a large population size, and even then, the population you simulate will have had a bottleneck of size equal to the founder population size. For highly realistic SNP or sequence data with low levels of relatedness, inbreeding, etc., we recommend that the founder population is as large as possible.

```
pop_size mat_cor offspring_dist selection_func selection_func_par1 selection_func_par2
3000 0 p thr 1 1
3000 0 p thr 1 1
3000 0 p thr 1 1
3000 0 p thr 1 1
3000 0 p thr 1 1
3000 0 p thr 1 1
3000 0 p thr 1 1
3000 0 p thr 1 1
3000 0 p thr 1 1
3000 0 p thr 1 1
```

ex1.popinfo.txt

As can be seen, the first line is header and there are 10 additional lines (for each generation) with a population of size 3000. The columns `thr 1 1` simply means no selection. The other parameters (columns 2-6) will be explained in Chapter 2.

To run *GeneEvolve* for a simple drift model without mutation or selection, use the code below. Because the user does not specify `--va` or `--vd`, var$[E] = 1$ by default and var$[A]$ and var$[D]$ are computed based on the additive and dominance effect sizes and the allele frequencies specified in the `cv.info` file. `seed` sets the random number generation seed so that simulation results can be rederived; if you do not specify it, the program will choose a random number as seed. `prefix` specifies that all output files will begin with "out.ex1".

Note: you should change `path` to the directory that the program *GeneEvolve* is located in. For example, if you unzip the `Example.zip` file, and then `cd Examples`, use `../bin/GeneEvolve` instead of `/path/GeneEvolve`. If you place *GeneEvolve* in an applications folder that exists in the `$PATH` (e.g., `/bin`; use `echo $PATH` to see which directories are in your `$PATH`) then you can simply type `GeneEvolve` without specifying the full path.

```
/path/GeneEvolve \
--file_gen_info ex1.popinfo.txt \
--file_hap_name par.pop1.hap_sample_address.txt \
--file_recom_map Recom.Map.b37.50KbDiff \
--file_cv_info cv.info \
--file_cvs par.pop1.cv_hap_files.txt \
--seed 12345 \
--prefix out.ex1
```

Code for Example 1

GeneEvolve creates a total of 18 new files in this case: one summary file (`out.ex1.pop1.summary`;), eleven phenotype files, one for generation 0 and ten for generations 1-10 (`out.ex1.info.pop1.gen*.txt`), and a `.hap` and `.indv` file for the final generation for each chromosome (total of six files for the three chromsomes simulated here).

The summary statistics for each generation are contained in the summary file. The columns show the variances of additive (A), dominance (D), genotypic ($G = A + D$), common sibling (C), unique environmental (E), parental (F), and total phenotypic ($P = A+D+C+E+F$), the narrow-sense heritability ($h^2 = \text{var}[A_g]/\text{var}[P_g]$) at each generation g, the variance of G at generation g

standardized by the variance of the founder population variance of G at time 0 ($\text{var}[G_g]/\text{var}[G_0]$), and the variances of the mating and selection values:

```
gen ph1_var_A ph1_var_D ph1_var_G ph1_var_C ph1_var_E ph1_var_F ph1_var_P ph1_h2 ph1_var_G_std var_mating_value
    var_selection_value
0 121.737 4.25635 124.514 0 1 0 125.767 0.967959 1 125.767 1
1 124.33 4.2101 128.371 0 1 0 128.459 0.967857 1.03098 128.459 1.02141
2 118.438 4.41779 123.912 0 1 0 124.835 0.948759 0.995165 124.835 0.992592
3 114.613 4.4722 118.065 0 1 0 119.402 0.959894 0.948203 119.402 0.949391
4 114.298 4.26211 119.248 0 1 0 120.49 0.948613 0.957702 120.49 0.958045
5 115.253 4.40189 118.133 0 1 0 118.57 0.972025 0.948753 118.57 0.942778
6 120.35 4.43166 124.973 0 1 0 125.557 0.958528 1.00369 125.557 0.998336
7 118.552 4.43403 121.746 0 1 0 122.913 0.964523 0.977766 122.913 0.977309
8 120.756 3.99147 126.503 0 1 0 126.909 0.951517 1.01597 126.909 1.00908
9 122.77 4.11834 127.32 0 1 0 127.775 0.960832 1.02254 127.775 1.01597
10 131.547 4.3614 133.714 0 1 0 134.045 0.981361 1.07388 134.045 1.06583
```

out.ex1.pop1.summary

In this example $\text{var}[E] = 1$ by default, and $\text{var}[A]$ and $\text{var}[D]$ are computed based on the additive and dominance effect sizes specified in the `cv.info` file. Here, the narrow-sense heritability is ∼0.96 across generations and by generation 10, $\text{var}[A]$ is 1.38% higher than it was in the founder population. Example 2 illustrates the more usual case where the user scales $\text{var}[A]$, $\text{var}[D]$ and $\text{var}[E]$.

For each generation, the program simulates phenotypes and in the `out.ex1.info.pop1.gen*.txt` files, reports two generations of pedigree information followed by the additive (A), dominance (D), genotypic ($G = A + D$), common sibling (C), unique environmental (E), parental (F), and total phenotypic ($P = A+D+C+E+F$) values for each individual. The final three columns contain mating value (MV), selection value (SV) and its selection function (see `out.ex1.info.pop1.gen10.txt` for this information from generation 10).

```
ID ID_Father ID_Mother ID_Fathers_Father ID_Fathers_Mother ID_Mothers_Father ID_Mothers_Mother sex ph1_A ph1_D
    ph1_G ph1_C ph1_E ph1_F ph1_P MV SV SV_f
1 975 1294 373 265 1110 1554 1 3.24744 2.80934 6.05678 0 1.3855 0 7.44228 7.44228 0.663852 1
2 975 1294 373 265 1110 1554 1 11.4525 1.13749 12.59 0 -0.467283 0 12.1227 12.1227 1.08121 1
3 2441 2388 320 1958 1120 1856 2 -11.974 1.46305 -10.511 0 0.0717792 0 -10.4392 -10.4392 -0.930636 1
4 2573 2291 1265 290 2976 2310 2 1.66111 -0.883735 0.777379 0 -0.738968 0 0.0384109 0.0384109 0.00365061 1
5 128 1330 2837 1525 2105 361 1 -4.46949 -0.294278 -4.76377 0 0.356487 0 -4.40728 -4.40728 -0.39277 1
6 128 1330 2837 1525 2105 361 2 -3.79678 2.25556 -1.54122 0 0.667059 0 -0.874165 -0.874165 -0.0777235 1
7 128 1330 2837 1525 2105 361 2 -3.43685 3.21396 -0.222884 0 -2.36603 0 -2.58891 -2.58891 -0.230627 1
8 128 1330 2837 1525 2105 361 2 6.41368 2.47784 8.89152 0 -0.58552 0 8.306 8.306 0.74087 1
9 128 1330 2837 1525 2105 361 1 14.5689 2.44939 17.0183 0 -1.078 0 15.9403 15.9403 1.42162 1
...
```

out.ex1.info.pop1.gen10.txt

Finally, *GeneEvolve* creates a `.indv` file (IDs of the people in the final generation, one row per person) and a `.hap` file (phased haplotype file, two columns per individual and one row per genetic variant) for each of the three chromosomes at the final generation.

1.5.2 Example 2 – Scaling the variance of additive, dominance and unique environmental effects

In order to control the variances of additive, dominance and unique environmental effects, we can use *GeneEvolve* to scale these variances in the founder population. The variances in subsequent generations may change due to natural selection, drift, assortative mating, migration, cultural transmission, and so forth. To scale the variances, run the following command:

```
/path/GeneEvolve \
--file_gen_info ex1.popinfo.txt \
--file_hap_name par.pop1.hap_sample_address.txt \
--file_recom_map Recom.Map.b37.50KbDiff \
--file_cv_info cv.info \
--file_cvs par.pop1.cv_hap_files.txt \
--va 3 \
--vd 1 \
--ve 2 \
--no_output \
--seed 12345 \
--prefix out.ex2
```

Code for Example 2

The summary statistics output file is:

```
gen ph1_var_A ph1_var_D ph1_var_G ph1_var_C ph1_var_E ph1_var_F ph1_var_P ph1_h2 ph1_var_G_std var_mating_value
    var_selection_value
0 3 1 3.88747 0 2 0 6.04642 0.496161 1 6.04642 1
1 3.1261 0.987175 4.20534 0 2 0 6.2702 0.498564 1.08177 6.2702 1.03701
2 3.17246 1.03002 4.22185 0 2 0 6.13293 0.517282 1.08602 6.13293 1.01431
3 3.15119 0.997206 4.08134 0 2 0 5.99266 0.525841 1.04987 5.99266 0.991109
4 3.13935 0.977009 4.01239 0 2 0 5.94886 0.527722 1.03213 5.94886 0.983865
5 3.28445 1.06132 4.48513 0 2 0 6.16831 0.532472 1.15374 6.16831 1.02016
6 3.19852 0.934098 4.08427 0 2 0 6.27223 0.509949 1.05063 6.27223 1.03735
7 3.10063 0.996814 4.14122 0 2 0 6.19148 0.50079 1.06528 6.19148 1.02399
8 3.0161 0.995782 4.13835 0 2 0 5.93437 0.508243 1.06454 5.93437 0.981467
9 2.96997 0.989681 3.92535 0 2 0 5.8508 0.507618 1.00974 5.8508 0.967646
10 3.11084 0.957658 4.27875 0 2 0 6.18205 0.503205 1.10065 6.18205 1.02243
```

out.ex2.pop1.summary

In this listing, the variances are scaled to $\text{var}[A] = 3$, $\text{var}[D] = 1$ and $\text{var}[E] = 2$ for generation zero (initial population at line 2), and therefore the narrow sense heritability should be $3/(3+1+2) = 0.5$. Since it is always possible that E and A have small correlation, then the narrow sense heritability is not exactly 0.5 at the generation 0, but it is close to it (0.496161). The variances for the next generations are computed and scaled according to the initial values. Note that we use the option `--no_output` to save disk space and time by not creating the output hap files.

1.5.3 Example 3 – No dominance effect

In order to work just with the additive effect and not dominance effect, we run the following command by setting `--vd 0`:

```
/path/GeneEvolve \
--file_gen_info ex1.popinfo.txt \
--file_hap_name par.pop1.hap_sample_address.txt \
--file_recom_map Recom.Map.b37.50KbDiff \
--file_cv_info cv.info \
--file_cvs par.pop1.cv_hap_files.txt \
--va 3 \
--vd 0 \
--ve 2 \
--avoid_inbreeding \
--no_output \
--seed 12345 \
--prefix out.ex3
```

Code for Example 3

The summary statistics output file is:

```
gen ph1_var_A ph1_var_D ph1_var_G ph1_var_C ph1_var_E ph1_var_F ph1_var_P ph1_h2 ph1_var_G_std var_mating_value
    var_selection_value
0 3 0 3 0 2 0 5.01803 0.597844 1 5.01803 1
1 3.13428 0 3.13428 0 2 0 5.05181 0.620426 1.04476 5.05181 1.00673
2 3.06076 0 3.06076 0 2 0 5.17631 0.591302 1.02025 5.17631 1.03154
3 3.07983 0 3.07983 0 2 0 5.12994 0.600365 1.02661 5.12994 1.0223
4 3.178 0 3.178 0 2 0 5.15433 0.616569 1.05933 5.15433 1.02716
5 3.09412 0 3.09412 0 2 0 5.08042 0.609028 1.03137 5.08042 1.01243
6 3.03889 0 3.03889 0 2 0 4.96056 0.612611 1.01296 4.96056 0.988547
7 3.04799 0 3.04799 0 2 0 4.95279 0.61541 1.016 4.95279 0.986999
8 2.96552 0 2.96552 0 2 0 5.01269 0.591602 0.988506 5.01269 0.998937
9 3.12052 0 3.12052 0 2 0 5.09312 0.612693 1.04017 5.09312 1.01496
10 3.29557 0 3.29557 0 2 0 5.28125 0.624012 1.09852 5.28125 1.05246
```

out.ex3.pop1.summary

In this listing, the variances are scaled to var$[A] = 3$, var$[D] = 0$ and var$[E] = 2$. The option `--avoid_inbreeding` is used to not let inbreeding between siblings and first cousins (see Chapter 4).

1.5.4 Example 4 – Assortative mating

In the assortative mating (AM) system modeled by *GeneEvolve*, couples choose each other based on their mating value, which is a function of phenotypes. For example, taller mates choose taller spouses according to some correlation. It has long been understood (Fisher, 1918 [3]) that this type of AM will increase the genetic variance and heritability of the trait under AM.

In *GeneEvolve*, the spousal correlation is between the mating values of mates. For a single phenotype, the mating phenotype value is the same as the phenotype value, but for simulating multiple phenotypes, the mating phenotype value will be a user-specified weighted combination of the phenotypes.

Note that when you scale the additive or dominance genetic variance using `--va` or `--vd`, the scaling factors (mean and standard devations) are computed in the first generation and then those same scaling factors are used every generation thereafter. This allows the additive or dominance genetic variance to change (increase with positive AM) over time. Obviously, if we had recomputed the scaling factors each generation, the genetic variances would remain constant.

In the following example we set the mating correlation equal to 0.5 between the mating values

of mates. You can specify this correlation in the second column of `ex4.popinfo.txt` file, for each generation.

```
/path/GeneEvolve \
--file_gen_info ex4.popinfo.txt \
--file_hap_name par.pop1.hap_sample_address.txt \
--file_recom_map Recom.Map.b37.50KbDiff \
--file_cv_info cv.info \
--file_cvs par.pop1.cv_hap_files.txt \
--va 1 \
--vd 0 \
--ve 1 \
--avoid_inbreeding \
--no_output \
--seed 12345 \
--prefix out.ex4
```

Code for Example 4

The population information for each generation is in the file `ex4.popinfo.txt`

```
pop_size mat_cor offspring_dist selection_func selection_func_par1 selection_func_par2
3000 0.5 p thr 1 1
3000 0.5 p thr 1 1
3000 0.5 p thr 1 1
3000 0.5 p thr 1 1
3000 0.5 p thr 1 1
3000 0.5 p thr 1 1
3000 0.5 p thr 1 1
3000 0.5 p thr 1 1
3000 0.5 p thr 1 1
```

ex4.popinfo.txt

The summary statistics output file is:

```
gen ph1_var_A ph1_var_D ph1_var_G ph1_var_C ph1_var_E ph1_var_F ph1_var_P ph1_h2 ph1_var_G_std var_mating_value
    var_selection_value
0 1 0 1 0 1 0 2.00736 0.498167 1 2.00736 1
1 1.16762 0 1.16762 0 1 0 2.1686 0.538419 1.16762 2.1686 1.08033
2 1.29397 0 1.29397 0 1 0 2.28725 0.565731 1.29397 2.28725 1.13943
3 1.3324 0 1.3324 0 1 0 2.43217 0.547824 1.3324 2.43217 1.21163
4 1.36359 0 1.36359 0 1 0 2.38339 0.572121 1.36359 2.38339 1.18733
5 1.34819 0 1.34819 0 1 0 2.27473 0.592681 1.34819 2.27473 1.13319
6 1.29679 0 1.29679 0 1 0 2.27644 0.569659 1.29679 2.27644 1.13405
7 1.26671 0 1.26671 0 1 0 2.23135 0.56769 1.26671 2.23135 1.11158
8 1.28567 0 1.28567 0 1 0 2.28535 0.56257 1.28567 2.28535 1.13849
9 1.33442 0 1.33442 0 1 0 2.33333 0.571897 1.33442 2.33333 1.16239
10 1.4494 0 1.4494 0 1 0 2.46232 0.588632 1.4494 2.46232 1.22665
```

out.ex4.pop1.summary

As can be seen, the variance of A increased from 1 to 1.449, very close to its predicted equilibrium value of $\frac{\text{var}[A_0]}{1-r_{mates}*h_g^2} = 1.322$, and as a result, heritability increased from 0.498 to 0.589.

1.5.5 Example 5 – Random mating

Another mating system is random mating (RM), where couples are chosen completely at random (including potential inbreeding). Note that this implies a non-monogamous mating system, where parents are chosen randomly for each offspring. Therefore, full-siblings are rare (almost all siblings will be half-siblings). If you use `--RM`, the second column of `ex1.popinfo.txt` will not be evaluated, because in the RM system there is no mating correlation. Similarly, the parameter `--avoid_inbreeding` will be ignored.

```
/path/GeneEvolve \
--file_gen_info ex1.popinfo.txt \
--file_hap_name par.pop1.hap_sample_address.txt \
--file_recom_map Recom.Map.b37.50KbDiff \
--file_cv_info cv.info \
--file_cvs par.pop1.cv_hap_files.txt \
--va 1 \
--vd 0 \
--ve 1 \
--no_output \
--RM \
--seed 12345 \
--prefix out.ex5
```

Code for Example 5

The summary statistics output file is:

```
gen ph1_var_A ph1_var_D ph1_var_G ph1_var_C ph1_var_E ph1_var_F ph1_var_P ph1_h2 ph1_var_G_std var_mating_value
    var_selection_value
0 1 0 1 0 1 0 2.00736 0.498167 1 2.00736 1
1 1.0224 0 1.0224 0 1 0 1.9518 0.523824 1.0224 1.9518 0.97232
2 1.03915 0 1.03915 0 1 0 2.00955 0.517107 1.03915 2.00955 1.00109
3 1.0706 0 1.0706 0 1 0 2.08022 0.514656 1.0706 2.08022 1.0363
4 1.05634 0 1.05634 0 1 0 2.07964 0.507945 1.05634 2.07964 1.03601
5 1.1134 0 1.1134 0 1 0 2.11281 0.526977 1.1134 2.11281 1.05253
6 1.07844 0 1.07844 0 1 0 2.07896 0.518738 1.07844 2.07896 1.03567
7 1.09048 0 1.09048 0 1 0 2.1221 0.51387 1.09048 2.1221 1.05716
8 1.06629 0 1.06629 0 1 0 2.06115 0.517328 1.06629 2.06115 1.0268
9 1.04263 0 1.04263 0 1 0 2.04454 0.50996 1.04263 2.04454 1.01852
10 1.03664 0 1.03664 0 1 0 1.96993 0.526234 1.03664 1.96993 0.981352
```

out.ex5.pop1.summary

For RM, the variance of A does not change a lot and for each generation it is close to 1.

1.5.6 Example 6 – Exponential population growth

In *GeneEvolve* it is possible to simulate any scenario for population growth by modifying the first column of `ex6.popinfo.txt` file. Here is an example for exponential population size.

```
/path/GeneEvolve \
--file_gen_info ex6.popinfo.txt \
--file_hap_name par.pop1.hap_sample_address.txt \
--file_recom_map Recom.Map.b37.50KbDiff \
--file_cv_info cv.info \
--file_cvs par.pop1.cv_hap_files.txt \
--avoid_inbreeding \
--no_output \
--seed 12345 \
--prefix out.ex6
```

Code for Example 6

The population information for each generation is in the `ex6.popinfo.txt` file:

```
pop_size mat_cor offspring_dist selection_func selection_func_par1 selection_func_par2
500 0 p thr 1 1
1000 0 p thr 1 1
2000 0 p thr 1 1
4000 0 p thr 1 1
8000 0 p thr 1 1
16000 0 p thr 1 1
32000 0 p thr 1 1
```

ex6.popinfo.txt

1.5.7 Example 7 – Population bottleneck

In this example, we simulate a bottleneck population with a sharp reduction in population size to 200 in the 8th generation, increasing to 1000 by generation 14.

```
/path/GeneEvolve \
--file_gen_info ex7.popinfo.txt \
--file_hap_name par.pop1.hap_sample_address.txt \
--file_recom_map Recom.Map.b37.50KbDiff \
--file_cv_info cv.info \
--file_cvs par.pop1.cv_hap_files.txt \
--avoid_inbreeding \
--no_output \
--seed 12345 \
--prefix out.ex7
```

Code for Example 7

The population information for each generation is in the `ex7.popinfo.txt` file:

```
pop_size mat_cor offspring_dist selection_func selection_func_par1 selection_func_par2
3000 0 p thr 1 1
3000 0 p thr 1 1
3000 0 p thr 1 1
3000 0 p thr 1 1
3000 0 p thr 1 1
3000 0 p thr 1 1
3000 0 p thr 1 1
200 0 p thr 1 1
250 0 p thr 1 1
300 0 p thr 1 1
350 0 p thr 1 1
500 0 p thr 1 1
700 0 p thr 1 1
1000 0 p thr 1 1
```

ex7.popinfo.txt

1.5.8 Example 8 – Simulating multiple phenotypes

In this example, we simulate a population with two phenotypes. In *GeneEvolve* you can simulate any number of phenotypes simply by adding CV information with the parameters `--file_cv_info` and `--file_cvs` for each additional phenotype. Users can also control the genetic correlations between traits by specifying different proportions of overlapping CVs (or CVs that are in linkage disequilibrium with one another) in `--file_cv_info` and `--file_cvs` files. Roughly, for a highly polygenic trait, the genetic correlation is equal to the proportion of overlapping CVs between the two phenotypes, so long as the allelic effects and number of CVs are the same between the two traits. For different numbers of CVs, this same basic idea applies but the formula is slighly more involved.

```
/path/GeneEvolve \
--file_gen_info ex1.popinfo.txt \
--file_hap_name par.pop1.hap_sample_address.txt \
--file_recom_map Recom.Map.b37.50KbDiff \
--file_cv_info cv.info \
--file_cvs par.pop1.cv_hap_files.txt \
--file_cv_info cv2.info \
--file_cvs par.pop1.cv2_hap_files.txt \
--avoid_inbreeding \
--no_output \
--seed 12345 \
--prefix out.ex8
```

Code for Example 8

It is possible to scale the variance components for each phenotype using the scaling parameters `--va`, `--vc`, `--ve`, etc. multiple times (in order of the phenotypes), once for each phenotype. For example, `--va 3 --va 2 --vc 3 --vc 0` sets var$[A] = 3$ and var$[C] = 3$ for the first phenotype and sets var$[A] = 2$ and var$[C] = 0$ for the second phenotype. For more complex combinations, see Section 4.2.7.

The summary statistics output file is:

```
gen ph1_var_A ph1_var_D ph1_var_G ph1_var_C ph1_var_E ph1_var_F ph1_var_P ph1_h2 ph1_var_G_std ph2_var_A ph2_var_D
     ph2_var_G ph2_var_C ph2_var_E ph2_var_F ph2_var_P ph2_h2 ph2_var_G_std var_mating_value var_selection_value
0 121.737 4.25635 124.514 0 1 0 125.767 0.967959 1 130.018 11.6689 138.651 0 1 0 139.874 0.929537 1 274.225 1
1 118.679 4.26497 122.072 0 1 0 122.962 0.965167 0.980388 124.831 11.3065 140.252 0 1 0 141.203 0.88405 1.01155
    264.312 0.963851
2 123.79 4.30595 125.961 0 1 0 126.835 0.975988 1.01162 125.558 11.4168 134.576 0 1 0 135.147 0.929043 0.970607
    254.09 0.926575
3 131.966 4.14998 134.615 0 1 0 135.734 0.972234 1.08112 125.257 11.836 136.388 0 1 0 137.29 0.91235 0.983678
    257.118 0.937617
4 128.382 4.50141 132.511 0 1 0 133.582 0.961075 1.06422 124.566 12.2716 136.624 0 1 0 137.583 0.905391 0.985379
    252.358 0.920257
5 132.089 4.22107 135.051 0 1 0 135.992 0.971298 1.08463 125.213 12.4729 134.876 0 1 0 135.124 0.926654 0.972772
    253.536 0.924552
6 136.914 4.27789 140.285 0 1 0 141.585 0.967004 1.12666 119.734 11.6877 132.033 0 1 0 132.84 0.901337 0.952267
    256.361 0.934855
7 135.769 4.29335 138.741 0 1 0 139.948 0.970141 1.11426 126.53 12.0125 141.55 0 1 0 142.189 0.889869 1.02091
    265.264 0.967323
8 127.445 4.32883 133.773 0 1 0 135.277 0.942103 1.07436 127.87 12.0406 140.196 0 1 0 140.924 0.90737 1.01115
    258.645 0.943185
9 130.87 4.44239 135.356 0 1 0 136.162 0.961138 1.08707 123.857 12.1776 134.334 0 1 0 134.768 0.91904 0.968862
    263.027 0.959163
10 130.633 4.1655 133.544 0 1 0 134.924 0.968192 1.07252 130.319 11.9921 145.931 0 1 0 147.299 0.884723 1.0525
    270.496 0.986403
```

out.ex8.pop1.summary

As it can be seen, there are two phenotypes (with prefixes `ph1_` and `ph2_`). Note that in the multi-phenotype simulations, the selection value and mating value are the sum of the phenotypes. You can easily choose different weights for them, such that one phenotype contributes more or less to the mating or selection phenotype, using the parameters `--lambda` and `--omega` (see figure 2.6 and chapter 4). By default, `--lambda 1` and `--omega 1` for each phenotype, setting the weights to be equal across phenotypes.

1.5.9 Example 9 – Selection

In this example, we simulate a population with a phenotype under selection. For each generation, the user can choose different selection functions (to simulate directional, stabilizing, or threshold) via different parameters in the `ex9.popinfo.txt` file (columns 4-6) using the parameter `--file_gen_info`. Here, we simulate 5 generations of directional selection (using the logit and probit functions), two generations of stabilizing selection, and three generations of threshold selection.

```
/path/GeneEvolve \
--file_gen_info ex9.popinfo.txt \
--file_hap_name par.pop1.hap_sample_address.txt \
--file_recom_map Recom.Map.b37.50KbDiff \
--file_cv_info cv.info \
--file_cvs par.pop1.cv_hap_files.txt \
--avoid_inbreeding \
--va 1 \
--vd 0 \
--ve 1 \
--no_output \
--seed 12345 \
--prefix out.ex9
```

Code for Example 9

The population information for each generation is in the `ex9.popinfo.txt` file:

```
pop_size mat_cor offspring_dist selection_func selection_func_par1 selection_func_par2
3000 0 p logit 20 0
3000 0 p logit 10 5
3000 0 p logit 1 1
3000 0 p probit 0 1
3000 0 p probit 0 2
3000 0 p stab 0 1
3000 0 p stab 1 4
3000 0 p thr .8 3
3000 0 p thr .8 10
3000 0 p thr .9 20
```

ex9.popinfo.txt

For the definition of `logit`, `probit`, `stab` and `thr` functions see Section 2.6.

1.5.10 Example 10 – Haplotypes shared identical by descent

One the important and novel features of *GeneEvolve* is the ability to report the *haplotypes from the founder population* which can then be used to derive *shared identical by descent* (IBD) between all pairs of individuals in a sample using a separate program we have written available (see link below). By adding the parameter `--interval` to the command line, the program will create output files with ".int" extension that contains information on identities of haplotypes from the founder population.

Note: the .int files increase in size by roughly $2n\frac{cM}{100}$ rows per generation (where n is the sample size) for chromosomes of length cM. For shared haplotype simulations, we recommend that you not run on a large number (e.g., thousands) of generations or else the files can become very large.

```
/path/GeneEvolve \
--file_gen_info ex1.popinfo.txt \
--file_hap_name par.pop1.hap_sample_address.txt \
--file_recom_map Recom.Map.b37.50KbDiff \
--file_cv_info cv.info \
--file_cvs par.pop1.cv_hap_files.txt \
--seed 12345 \
--interval \
--prefix out.ex10
```

Code for Example 10

A part of the `out.ex10.pop1.gen10.chr1.int` file is:

```
h_ID chr hap st en hap_index root_pop
1 1 0 738555 1935359 1268 1
1 1 0 1935359 3701679 1184 1
1 1 0 3701679 3825508 989 1
1 1 0 3825508 7841616 1413 1
1 1 0 7841616 18568786 1519 1
1 1 0 18568786 24567649 454 1
1 1 0 24567649 26495876 891 1
1 1 0 26495876 34862278 2635 1
1 1 0 34862278 86048915 701 1
1 1 0 86048915 89852818 818 1
1 1 0 89852818 110453407 412 1
1 1 0 110453407 112133398 947 1
1 1 0 112133398 156580622 3530 1
1 1 0 156580622 156919514 896 1
1 1 0 156919514 175449373 160 1
1 1 0 175449373 177862000 2195 1
1 1 0 177862000 185522705 1961 1
1 1 0 185522705 208189978 1856 1
1 1 0 208189978 208239384 1054 1
1 1 0 208239384 217686627 3722 1
1 1 0 217686627 217797573 3584 1
1 1 0 217797573 218168972 3583 1
1 1 0 218168972 218277055 1131 1
1 1 0 218277055 218782909 3776 1
1 1 0 218782909 222689986 817 1
1 1 0 222689986 236621757 820 1
1 1 0 236621757 240768054 819 1
1 1 0 240768054 241314978 2945 1
1 1 0 241314978 245434911 3983 1
1 1 0 245434911 249238555 2121 1
1 1 1 738555 3701679 2933 1
1 1 1 3701679 5981788 2639 1
1 1 1 5981788 19061732 3459 1
1 1 1 19061732 19921491 3337 1
...
```

out.ex10.pop1.gen10.chr1.int

Based on this output, the first haplotype (0) of chromosome 1 for the individual with ID 1, consists of the start of the chromosome at position 738555 (base pair; bp), and ending at bp 1935359. The interval $[738555, 1935359)$ descended from the haplotype 1268 in the founder population (Simply, the `hap_index` column is the line number of the inputted initial .hap file). So, haplotype 0 of chromosome 1 of individual 1 is represented by half-open intervals of the form $[738555, 1935359)$, $[1935359, 3701679)$, $[3701679, 3825508)$, $[3825508, 7841616)$, $[7841616, 18568786)$, $[18568786, 24567649)$, $\cdots$ where these interval haplotypes descended down from founder haplotype IDs 1268, 1184, 989, 1413, 1519, 454 $\cdots$, respectively. A mathematical definition of this interval representation is presented in Section 2.3.

If segments of two chromosomes from two individuals were inherited from a common ancestral chromosome at a given location, then these segments are called identical by decent (IBD), that is, the segment has the same ancestral origin in these two individuals.

Given a ".int" file as input, the program "SharedHaplotypes" available at
https://github.com/rtahmasbi/IBG-SharedHaplotypes
can extract the true shared haplotypes between each two individuals and can report the true IBD segments.

1.5.11 Example 11 – Comprehensive example

For a comprehensive example, we simulate two populations, each with two phenotypes, in Chapter 4. The initial inputed genotype files are two different datasets. The variance components for each phenotype are scaled. Populations are under selection with different selection functions. The selection functions also change over time (over generations). The selection and mating values are computed with unequal weights. The mating system is assortative mating while avoiding inbreeding. There are also familial and siblings effects. The first phenotype has more effect on the familial effect. Populations migrate with different migration rates per generation.

1.6 Summary of features

In Table 1.1, we summarizes the basic features of *GeneEvolve*.

Table 1.1: Current features of *GeneEvolve*

- Specifiable duration of simulation (in synchronous generations)
- Univariate or multivariate phenotypes
- Additive and dominance genetic effects
- Multiple types of familial effects and unique environmental effects
- Migration between two or more populations
- Random mating or user-specified mate correlations (assortative mating)
- Monogamous or polygamous mating system
- Multiple genes and chromosomes, of arbitrary number and length
- Diploid populations
- Fixed or Poisson family size distribution
- User defined population size
- Phenotype-based natural selection of user-specified type
- Point and interval mutation rate
- Point and interval recombination rate
- Two types of familial environmental effects
- Ability to handle very large samples and simulate individual-level SNP or sequence data
- Input as hap (SHAPIT) format
- Output individual phenotype files and individual genotype files in hap, PLINK and plain text formats

Mating system: User can specify the correlation (which can change across time) between mates for the assortative mating system. If multiple phenotypes, the mating phenotype is a user-specified linear combination of them. Random mating (non-monogamous mating system where parents of offspring are chosen at random) is also possible. User can also choose monogamous or polygamous mating system.
Fecundity: In each generation, the offspring distribution can be uniform (constant number) or Poisson distribution.
Selection system: User can specify the strength and type (balancing selection, directional selection, inbreeding effects, and threshold selection) of natural selection acting on the selection

phenotype. If multiple phenotypes, the selection phenotype is a user-specified linear combination of them.

Recombination rate: *GeneEvolve* can handle variations allowed by defining hot spots or a genetic map.

Mutation rate: Similar to recombination rate, *GeneEvolve* can handle variation allowed by defining hot spots or a genetic map for mutation rate. Based on the user-specified mutation rate, the positions of mutations (which can affect the phenotype if they occur at user-specified sites) can also be stored for newly generated haplotypes.

Population growth: Population size can de defined by user at every time step, so it is easy to simulate bottlenecks.

Migration: By defining migration matrix per generation, a user can model Wright's Island model or general model.

Events: By specifying the migration matrix, selection strength and population size, user can simulate complex evolutionary events. Migration matrix, population size and selection strength can vary for each generation.

Additive and dominance effects: User specify the additive (a_j) and dominance (d_j) effects for each CV, according to [14]. A user can further specify the total additive and dominance genetic variances (in which case the user-specified additive and dominance effects are scaled appropriately to lead to the requested variances in the first generation; they may change thereafter due to drift, assortative mating, selection, etc.).

Familial effects: *GeneEvolve* can handle situations in which parents pass some proportion of their phenotypes to their offsprings ("vertical transmission"). Similarly, environmental effects shared by siblings (but not passed down from parents) can be simulated.

Output formats: Beside the standard output for all the phenotypic effects (total phenotype values, additive genetic values, dominant genetic values, etc.) per individual and per generation, *GeneEvolve* can create outputs for individual genotypes in SHAPEIT (".hap"), PLINK and plain text file formats. The plain text output is useful for investigating the lengths and locations of IBD shared segments. It also reports the summary statistics for each generation.

Chapter 2

Population Genetics Models

2.1 Population's basic information

The Sewall Wright and R. A. Fisher (for more info see [6]) model assumes discrete, non-overlapping generations $G_0, G_1, G_2, \ldots$. In *GeneEvolve* we also assume discrete and non-overlapping generations with size N_i for generation G_i.

2.1.1 Population size in *GeneEvolve*

For inputting the basic information of population, user should use the parameter

```
--file_gen_info [file_generaions_info.txt]
```

where `file_generaions_info.txt` is a text file with 6 columns and $n+1$ lines, where n is the number of generations to be simulated by *GeneEvolve*. This file should have a header and the first column is the population size. The structure of this file is listed in figure 2.1. *GeneEvolve* can simulate different population sizes in each generation by specifing them in the first column.

2.1.2 Number of offspring's distribution

The third column of file `file_generaions_info.txt` determines the distribution of offsprings per generation. It can be `p` for Poisson or `f` for fixed number of offsprings (see the third column of figure 2.1).

The mean Poisson distributed number of offspring in generation i is obtained by

$$\frac{N_{i+1}}{C_i},$$

where N_{i+1} is the user specified population size (first column of `file_generaions_info.txt`) for generation $i+1$ and C_i the number of couples at generation i.

For the fixed number of offsprings, we have

$$k = \text{round}(\frac{N_{i+1}}{C_i}).$$

```
pop_size mat_cor offspring_dist selection_func selection_func_par1 selection_func_par2
3000 0 p logit 20 0
3000 0 p logit 20 0
3000 0 p logit 20 0
3000 0 p logit 20 0
3000 0 p logit 20 0
3000 0 p logit 20 0
3000 0 p logit 20 0
3000 0 p logit 20 0
3000 0 p logit 20 0
3000 0 p logit 20 0
3000 0 p logit 20 0
3000 0 f logit 20 0
```

Figure 2.1: The structure of file_generaions_info.txt file.

For most situations of simulating realistic data, we recommend you use Poisson distributed number of offspring; fixed number of offspring can lead to significantly different population sizes than the user inputs due to rounding.

The selection function has a direct effect on C_i. If the selection function allows all individuals to marry (there is no selection), then C_i should be $N_{i+1}/2$ on average. So each family should have 2 offspring on average if the population size is constant. More stringent selection function can reduce the C_i and as the results, it increases the average family size, N_{i+1}/C_i. In Section 2.6, you can find more information about the selection function.

2.2 Haplotypes of the initial population

GeneEvolve works with phased haplotypes as input. For more information about haplotype file format, see the appendix C. Since the size of genome can be large, the haplotype information of each chromosome should be stored in separate files. For the initial population (generation 0), the user should specify a file containing the *address* or *path* to where the files of each chromosome are located. With the parameter `--file_hap_name [file.txt]`, user should address the `.hap`, `.legend`, and `.indv` files, respectively, for each chromosome per line. Figure 2.2 shows a typical example with 22 chromosomes. This file has header and the first column is chromosome number. Note that the file `.indv` has no header.

It is also possible for *GeneEvolve* to work with a subset of chromosomes. See the following example, where the user simulates just chromosomes 6, 10 and 22.

Example 2.1 (Working with few chromosomes)**.** If user wants to simulate a population with 3 chromosome (6, 10 and 22), he/she should prepare the following file for the parameter `--file_hap_name`.

```
chr hap legend indv
6 /path/chr6.hap /path/chr6.legend /path/chr6.indv
10 /path/chr10.hap /path/chr10.legend /path/chr10.indv
22 /path/chr22.hap /path/chr22.legend /path/chr22.indv
```

file.txt

```
chr hap legend indv
1 /path/chr1.hap /path/chr1.legend /path/chr1.indv
2 /path/chr2.hap /path/chr2.legend /path/chr2.indv
3 /path/chr3.hap /path/chr3.legend /path/chr3.indv
4 /path/chr4.hap /path/chr4.legend /path/chr4.indv
5 /path/chr5.hap /path/chr5.legend /path/chr5.indv
6 /path/chr6.hap /path/chr6.legend /path/chr6.indv
7 /path/chr7.hap /path/chr7.legend /path/chr7.indv
8 /path/chr8.hap /path/chr8.legend /path/chr8.indv
9 /path/chr9.hap /path/chr9.legend /path/chr9.indv
10 /path/chr10.hap /path/chr10.legend /path/chr10.indv
11 /path/chr11.hap /path/chr11.legend /path/chr11.indv
12 /path/chr12.hap /path/chr12.legend /path/chr12.indv
13 /path/chr13.hap /path/chr13.legend /path/chr13.indv
14 /path/chr14.hap /path/chr14.legend /path/chr14.indv
15 /path/chr15.hap /path/chr15.legend /path/chr15.indv
16 /path/chr16.hap /path/chr16.legend /path/chr16.indv
17 /path/chr17.hap /path/chr17.legend /path/chr17.indv
18 /path/chr18.hap /path/chr18.legend /path/chr18.indv
19 /path/chr19.hap /path/chr19.legend /path/chr19.indv
20 /path/chr20.hap /path/chr20.legend /path/chr20.indv
21 /path/chr21.hap /path/chr21.legend /path/chr21.indv
22 /path/chr22.hap /path/chr22.legend /path/chr22.indv
```

Figure 2.2: The structure of [file.txt] in [–file_hap_name].

2.3 Recombination and linkage

2.3.1 Introduction

Genetic *recombination*, also called *crossing over*, refers to genetic events that occur during the formation of sperm and egg cells. During the early stages of cell division in meiosis, two chromosomes of a homologous pair may exchange segments, producing *recombinant chromosomes* in germ cells that are a mix of the two parental chromosomes (see 2.3). For example, if one homologous chromosome has a *haplotype* (genetic sequence on the same chromosome) AB, and another homologous chromosome has a haplotype ab, one of the gamete cells, because of recombination, may have a chromosome with genotype Ab. Such gametes are called recombinants. The proportion of recombinants is called the *recombination rate* between these two loci, which is 1/2 if two loci are on two different chromosomes, and thus segregate independently.

The *genetic distance* (also called *map distance* as opposed to *physical distance*) between two loci is defined as the average number of crossovers between the loci per meiosis. The unit of genetic distance is the *centiMorgan* (cM). Two loci are 1 cM apart if on average there is one crossover occurring between these two loci on a single strand for every 100 meiosis. The distribution of recombination events varies between the sexes: females have on average 1.65-fold more recombination events when they make eggs than males do when they make sperms. Even on a single chromosome, recombination rate is highly uneven, and there exists *recombination hotspots* with peak recombination rate hundreds or thousands times that of the surrounding regions.

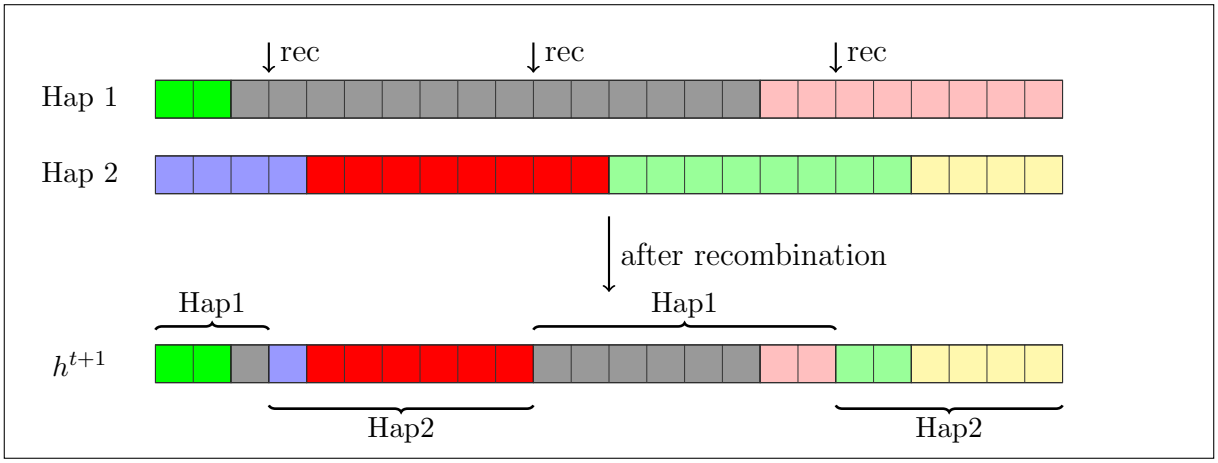


Figure 2.3: Recombination model: three recombination occurred at different positions. Each color code is an ancestral IBD.

Given a reference panel of haplotypes $H_N = \{h_1, \ldots, h_{2N}\}$ as input, where each haplotype is typed at L bi-allelic sites, that is $h_i = (h_{i,1}, \ldots, h_{i,L})$ and $h_{i,j} \in \{0, 1\}$, the program chooses mates based on the specified mating system, and for the newly simulated child, each haplotype is a recombination of parent's haplotypes, where the recombination rate is taken from a user-supplied recombination map file specifying genetic distances. Precisely, assume that at generation t, the haplotypes of one parent are h_i^t and h_{i+1}^t. These haplotypes are stored as a continues sequence of half open intervals, i.e., $h_i^t = \{[1, r_1) \cup [r_1, r_2) \cdots \cup [r_{v-1}, L]\}$ and $h_{i+1}^t = \{[1, s_1) \cup [s_1, s_2) \cdots \cup [s_{u-1}, L]\}$, for any arbitrary numbers u and v. Now assume that, for a child's newly formed chromosome, a recombination occurs at position w. If $r2 \leq w < r3$ and $s3 \leq w < s4$, then the new recombined haplotype becomes

$$h_i^{t+1} = \{[1, r_1) \cup [r_1, r_2) \cup [r_2, w) \cup [w, s_4) \cup [s_4, s_5) \cdots \cup [s_{u-1}, L]\}. \tag{2.1}$$

It is also possible that several recombinations occurs; the idea is the same. For a graphical illustration see the figure 2.3.

Clearly, working with a continuous sequence of intervals is much faster and more memory efficient than working with real genotypes each generation, allowing *GeneEvolve* to simulate even large SNP/sequence datasets across many generations. All that needs to be stored for each individual is their haplotype intervals, from which their CVs can be quickly calculated and hence their phenotype. Based on the user-specified mutation rate, the position of new mutations are also stored for the newly generated haplotypes.

2.3.2 Recombination in *GeneEvolve*

Different locations across the genome have different recombination rates. A high-resolution recombination map of the human genome was estimated by Kong et. al. [7] and is available online (`Recom_Map.zip` file in the github root) and is supplied as the `Recom.Map.b37.50KbDiff` file in the `Example` folder supplied with *GeneEvolve*. This map provides the average cM distance every 50000 bp; more fine-grained maps can also be supplied to *GeneEvolve* if the user wishes, although the error in estimating cM distances is probably too great for this to make much of a difference.

The recombination file is specified using the parameter `--file_recom_map [file_recom.txt]`. The file `file_recom.txt` has a header with 3 columns: chromosome number, base-pair distance and cM distance.

Example 2.2 (Recombination map file format)**.** In the following file, you can see part of a equidistant genetic map (cM reported every 10k bp). By definition, the probability of recombination in a chunk is the difference between its two cM distances divided by 100. These probabilities are computed automatically by *GeneEvolve* program.

```
chr bp    cM
1 1128555 1.13368814337268
1 1138555 1.14905703198189
1 1148555 1.15742981154837
1 1158555 1.16581577645964
1 1168555 1.17462263654929
1 1178555 1.18341927550241
1 1188555 1.19221591445554
1 1198555 1.20104594872684
1 1208555 1.20989254327934
1 1218555 1.21873913783184
```

file_recom.txt

For example, the probability occurrence of one recombination in the interval $[1148555, 1158555)$ is equal to

$$\frac{1.16581577645964 - 1.15742981154837}{100} = 0.0000839.$$

If a recombination event occurs in this interval, *GeneEvolve* randomly chooses at precisely which basepair (from among the 10k positions with equal probability) the recombination occurred.

2.4 Simulating complex quantitative traits

Computer programs that can simulate genotypes with phenotypes based on user-specified disease or quantitative trait models are useful in genetic studies. They can be used to evaluate statistical power when planning a study design based on the proposed sample size, the assumed genotypic relative risks (GRR), and allele frequencies. They are also useful for evaluating type I error rates for new statistical association tests and power comparisons between the new tests and other existing tests. *GeneEvolve* can simulate several phenotypes, simultaneously, based on existing publicly available SNP (e.g., UK Biobank, dbGaP, etc.) or sequence (UK10K, 1000 genome, etc.) datasets.

2.4.1 Genotypic value for single locus

If we could replicate a particular genotype in a number of individuals and measure them under environmental conditions normal for the population, their mean environmental deviation would be zero, and their mean phenotypic value would consequently be equal to the genotypic value of that particular genotype. This is the meaning of the genotypic value of an individual. In principle it is measurable, but in practice it is not, except when we are concerned with a single locus where the genotypes are phenotypically distinguishable, or with the genotypes represented in highly inbred lines [2].

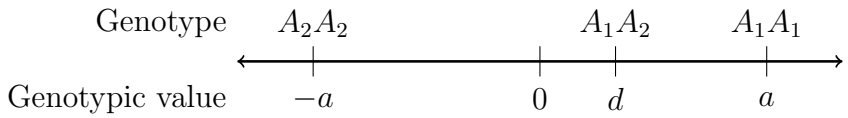


Figure 2.4: Arbitrary assigned genotypic value.

Table 2.1: Values of genotypes in a two-allele system, measured as deviations from the population mean. The population mean is $a(p-q)+2pqd$, and $\alpha = a + d(q-p)$.

	Genotypes		
	A_1A_1	A_1A_2	A_2A_2
Frequencies	p^2	$2pq$	q^2
Assigned values	a	d	$-a$
Deviation from population mean	$2q(a-pd)$	$a(q-p)+d(1-2pq)$	$-2p(a+qd)$
or	$2q(\alpha-qd)$	$\alpha(q-p)+2pqd$	$-2p(\alpha+pd)$
Breeding value	$2q\alpha$	$(q-p)\alpha$	$-2p\alpha$
Dominance deviation	$-2q^2d$	$2pqd$	$-2p^2d$

Considering a single locus with two alleles A_1 and A_2, we call the genotypic value of one homozygote $+a$, that of the other homozygote $-a$ and that of the heterozygote d (we adopt the convention that A_1 is the increasing allele). We thus have a scale of genotypic values as in figure 2.4. The origin, or point of zero value, on this scale is midway between the values of the two homozygotes. The value d of the heterozygote depends on the degree of dominance. If there is no dominance, $d = 0$; if A_1 is dominance over A_2, d is positive, and if A_2 is dominant over A_1, d is negative. If dominance is complete, d is equal to $+a$ or $-a$, and if there is overdominance, d is greater than $+a$ or less than $-a$. The degree of dominance may be expressed as d/a. The deviation from population mean is summarized in Table 2.1.

2.4.2 Phenotypic model in *GeneEvolve*

For simplicity, we assume here a single phenotype. For several phenotypes, the idea is the same, and we discuss how to simulate multiple phenotypes later. For each individual i, the phenotypic value P_i is a random variable defined as

$$P_i = A_i + D_i + E_i + F_f + C_f + \gamma_p \tag{2.2}$$

where A_i and D_i are the additive and dominance genetic values for individual i. The terms F_i, E_i, C_f, and γ_p are, respectively, familial, unique, shared sibling (common), and population specific environmental values for person i in family f in population p. For familial, shared sibling (common), and population specific environmental effects, see Sections 2.7, 2.8 and 2.9, respectively.

To simulate phenotypes, users can specify m *causal variants* (CVs), $\{\text{cv}_j\}_{j=1}^m$, and their additive (a_j) and dominance (d_j) effects. The value a_j and d_j are defined in Section 2.4.1. For each individual i, the additive term A_i is a sum across causal variant genotype scores multiplied by their additive effect size (α_j), i.e.,

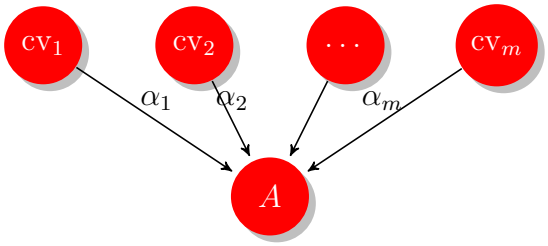


Figure 2.5: Additive term

$$A_i = \sum_{j=1}^{m} (x_{ij} - 2p_j)\alpha_j, \tag{2.3}$$

where

$$\alpha_j = a_j + d_j(q_j - p_j),$$

is the additive effect size, p_j is frequency of the increasing allele (A_1) with $q_j = 1 - p_j$ the frequency of the decreasing allele, $x_{ij} \in \{0, 1, 2\}$ is the genotypic value for cv_j (the number of A_1 alleles), and m is the number of CVs (see Figure 2.5). One choice would be to sample a_j and d_j such that $a_j \sim N(0, \sigma_a^2/(2p_jq_j))$ and $d_j \sim N(0, \sigma_d^2/(2p_jq_j)^2)$, for arbitrary positive numbers σ_a^2 and σ_d^2 - this would ensure that variance explained per CV is irrespective of its MAF, and hence that the effect sizes (a_j and d_j) for rare variants are larger than for common variants (which might often be true for traits under purifying selection). Other choices are possible; for example, $a_j \sim N(0, \sigma_a^2)$ would mean the CV effect size does not depend on MAF (the case ostensibly under no selection) and therefore the variance explained per CV is greater for common variants.

For each individual i, the dominance term D_i is,

$$D_i = \sum_{j=1}^{m} t_{ij} d_j, \tag{2.4}$$

where

$$t_{ij} = \begin{cases} -2p_j^2 & \text{if } x_{ij} = 0 \\ 2p_jq_j & \text{if } x_{ij} = 1 \\ -2q_j^2 & \text{if } x_{ij} = 2 \end{cases}.$$

For more information, see [2, 14].

Assuming no LD between CVs, for the additive term we have

$$\begin{aligned} \text{var}[A_i] &= \sum_{j=1}^{m} \alpha_j^2 \text{var}[x_{ij} - 2p_j] \\ &= \sum_{j=1}^{m} [a_j + d_j(q_j - p_j)]^2 (2p_jq_j), \end{aligned} \tag{2.5}$$

since $\text{var}[x_{ij}] = 2p_jq_j$.

For the dominance term,

$$\mathbb{E}[t_{ij}] = -2p_j^2 \times q_j^2 + 2p_jq_j \times 2p_jq_j - 2q_j^2 \times p_j^2 = 0,$$

and

$$\begin{aligned}\text{var}[t_{ij}] =&(-2p_j^2)^2 \times q_j^2 + (2p_jq_j)^2 \times 2p_jq_j + (-2q_j^2)^2 \times p_j^2\\ =&4p_j^4 \times q_j^2 + 4p_j^2q_j^2 \times 2p_jq_j + 4q_j^4 \times p_j^2\\ =&4p_j^2q_j^2[p_j^2 + 2p_jq_j + 2q_j^2]\\ =&4p_j^2q_j^2.\end{aligned}$$

Therefore,

$$\begin{aligned}\text{var}[D_i] =& \sum_{j=1}^{m} d_j^2 \text{var}[t_{ij}]\\ =& \sum_{j=1}^{m} (2p_jq_j)^2 d_j^2.\end{aligned} \tag{2.6}$$

2.4.3 Heritability

Heritability is the proportion of phenotypic variation explained by genetic effects. Broad–sense heritability (H^2) refers to the inclusion of all potential sources of genetic variation (additive, dominance, epistatic, maternal and paternal effects). Here, because we only include additive and dominance effects:

$$H^2 = \frac{V_G}{V_P} = \frac{V_A + V_D}{V_P},$$

where $V_A := \text{var}[A]$ and $V_D := \text{var}[D]$.

The ratio of additive genetic variation to the total phenotypic variation observed is the narrow–sense heritability (h^2):

$$h^2 = \frac{V_A}{V_P}.$$

2.4.4 Simulating phenotypes in *GeneEvolve*

To simulate a phenotype, you should specify the parameters

```
--file_cv_info [cv_info.txt] --file_cvs [cvs.txt]
```

The file `cv_info.txt` contains CV information and has a header with 4 columns: chromosome number, base-pair distance, additive (a) and dominance (d) effects (see Example 2.3).

The file `cvs.txt` has no header and contains the address (path) to haplotype files containing only the CVs. This file has just 2 columns: chromosome number and address of CV haplotype file (see Example 2.4).

Example 2.3 (CV information file format)**.** The file format for `--file_cv_info [cv_info.txt]` is illustrated in the following listing.

```
chr pos a d
1 4328476 1.18585978544321 -1.54676435875486
1 4500436 -0.128733310867769 1.11559331900794
1 7736097 -0.379038685626258 -1.12984403982323
1 14418448 0.0959318783552277 0.527555965661557
1 15825195 1.89998223576411 -0.724345249882114
1 17889690 -0.368424537129164 -0.120587409943792
...
22 42504679 0.990334634266996 -0.698799892553054
22 44338134 1.26982516250768 0.990334634266996
22 47450911 1.8384670663176 -0.625110331654888
22 49411595 1.44121811394195 0.878797016360936
```

cv_info.txt

Example 2.4 (CVs file format)**.** The file format for `--file_cvs [cvs.txt]` is illustrated in the following listing.

```
1 /path/CVs.chr1.hap
2 /path/CVs.chr2.hap
...
22 /path/CVs.chr22.hap
```

cvs.txt

For example, assume that we set $a_j = +2$ and $d_j = 0$ for the jth CV. Clearly, the genotypic value x_{ij} for individual i can be 0, 1, 1, or 2 (in equation 2.3), if the value of hap file for this CV be (0,0), (0,1), (1,0) or (1,1), respectively (See the hap file format in Appendix C). So, if the haplotype for individual i is $(1, 1)$, then setting $a_j = +2$ increases the value of A_i and setting $a_j = -2$ decreases its value. The value for the haplotypes (0 or 1) are constant over generations (not a function of allele frequency) and they are predetermined in the reference haplotype file.

2.4.5 Scaling VA and VD

In Equations 2.5 and 2.6 we compute the additive and dominance variances. It is possible in *GeneEvolve* to scale them to any arbitrary positive number using the following parameters:
`--va [number] --vd [number]`

This will force var$[A]$ and var$[D]$ to be whatever is specified. If there is more than one phenotype, the user can add more scaling parameters; the first `--va [number]` corresponds to the first phenotype, the second to the second, and so forth.

2.4.6 Simulating multivariate phenotypes

To simulate k phenotypes (multivariate phenotypes) with different CVs, user should specify the parameters `--file_cv_info [cv_info.txt]` and `--file_cvs [cvs.txt]` in the command line, k times, in order of the phenotypes.

Example 2.5 (Simulating 3 phenotypes)**.** In the following listing, *GeneEvolve* will create 3 phenotypes, where their CVs are listed in different files.

```
GeneEvolve --file_gen_info gen.info --file_hap_name hap_add.txt \
--file_recom_map map.txt \
--file_cv_info cv_info_p1.txt --file_cvs cvs_p1.txt  \
--file_cv_info cv_info_p2.txt --file_cvs cvs_p2.txt  \
--file_cv_info cv_info_p3.txt --file_cvs cvs_p3.txt
```

Simulating 3 phenotypes

To create genetic overlap (genetic correlations) between phenotypes, the user can specify different proportions of overlapping CVs or CVs that are in linkage disequilibrium with one another. Specifically, the expected genetic correlation, r_g, between two traits (assuming effect sizes are the same between them) is $r_g = \frac{m_c}{\sqrt{m_1}\sqrt{m_2}}$ where m_c is the number of CVs in common between the traits, m_1 is the number of CVs in trait 1, and m_2 is the number of CVs in trait 2 (see McNemar, 1949).

2.5 Random and non-random mating systems

Mating systems can affect the way that alleles are combined in individuals in a population. Outcrossing organisms put together new combinations of genes rapidly, leading to many different genotypes within populations (and creating high genotype diversity) and the potential for rapid adaptation in a changing environment.

For true random mating, where each child's parents are chosen at random (and therefore parents are not monogamous), users should use the parameter `--RM`. Note that this leads to very few full siblings and many half-siblings. If this is done, then the second column in file `file_generaions_info.txt` inputted in parameter `--file_gen_info [file_generaions_info.txt]` will not be used and `--avoid-inbreeding` will be ignored.

For monogamous matings systems, the user should not use the `--RM` argument, but should instead specify some correlation between mates specified in the `--file_gen_info` file. In assortative mating (mating based on some phenotype), *GeneEvolve* creates a random variable called mating value, MV, by combing all the k phenotypes as

$$\text{MV}_i = \sum_{j=1}^{k} \omega_j P_{ij}, \tag{2.7}$$

where P_{ij} is the jth phenotype for individual i and ω_j's are some coefficients (see Figure 2.6).

Mates chose each other based on the mating values, MV, and have all offspring together (monogamy). User can specify the correlation (which can change across time) between mating values of mates in the second column of file specified in the `--file_gen_info` parameter. Their correlation is

$$\mu = \text{corr}[\text{MV}_{FA}, \text{MV}_{MO}]. \tag{2.8}$$

Therefore, if users wish to simulate a monogamous mate system where mates are phenotypically uncorrelated, they should use 0's in the second column (which gives the mate correlation) in the file specified in the `--file_gen_info` command. If the user wants a non-zero mate correlation in a monogamous mating system, this number should be used instead in the second column of this file. Currently, *GeneEvolve* does not allow for non-zero mate correlations in non-monogamous mating systems.

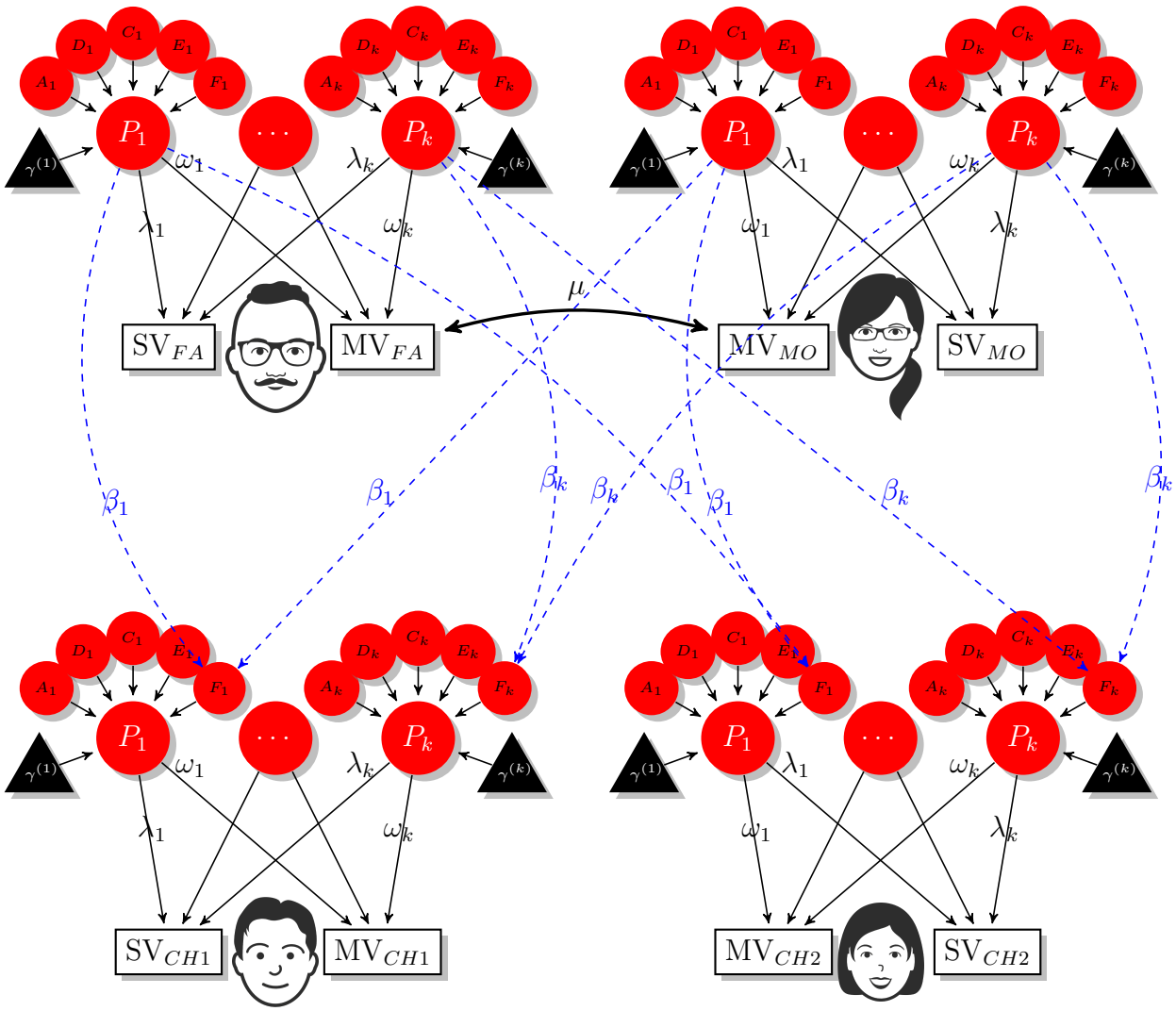


Figure 2.6: Mating path diagram and phenotypes with `--vt_type 1`.

2.6 Natural selection

Natural selection is the differential survival and reproduction of individuals due to differences in phenotype. It is a key mechanism of evolution, the change in heritable traits of a population over time. Natural variation occurs among the individuals of any population of organisms. Many of these differences do not affect survival or reproduction, but some differences may improve the chances of survival and reproduction of a particular individual.

Natural selection is a process that favors certain traits or trait values over others. Selection can be positive (or advantageous) or negative (or purifying) and has a profound impact on the evolution of the human population. In addition, selection can be balancing in which the genotypes have a mixture of positive and negative selection pressures so that there is no net effect of selection on the individual alleles.

For each individual i, the selection value can be computed from k phenotypes as

$$\mathrm{SV}_i = \sum_{j=1}^{k} \lambda_j P_{ij}, \tag{2.9}$$

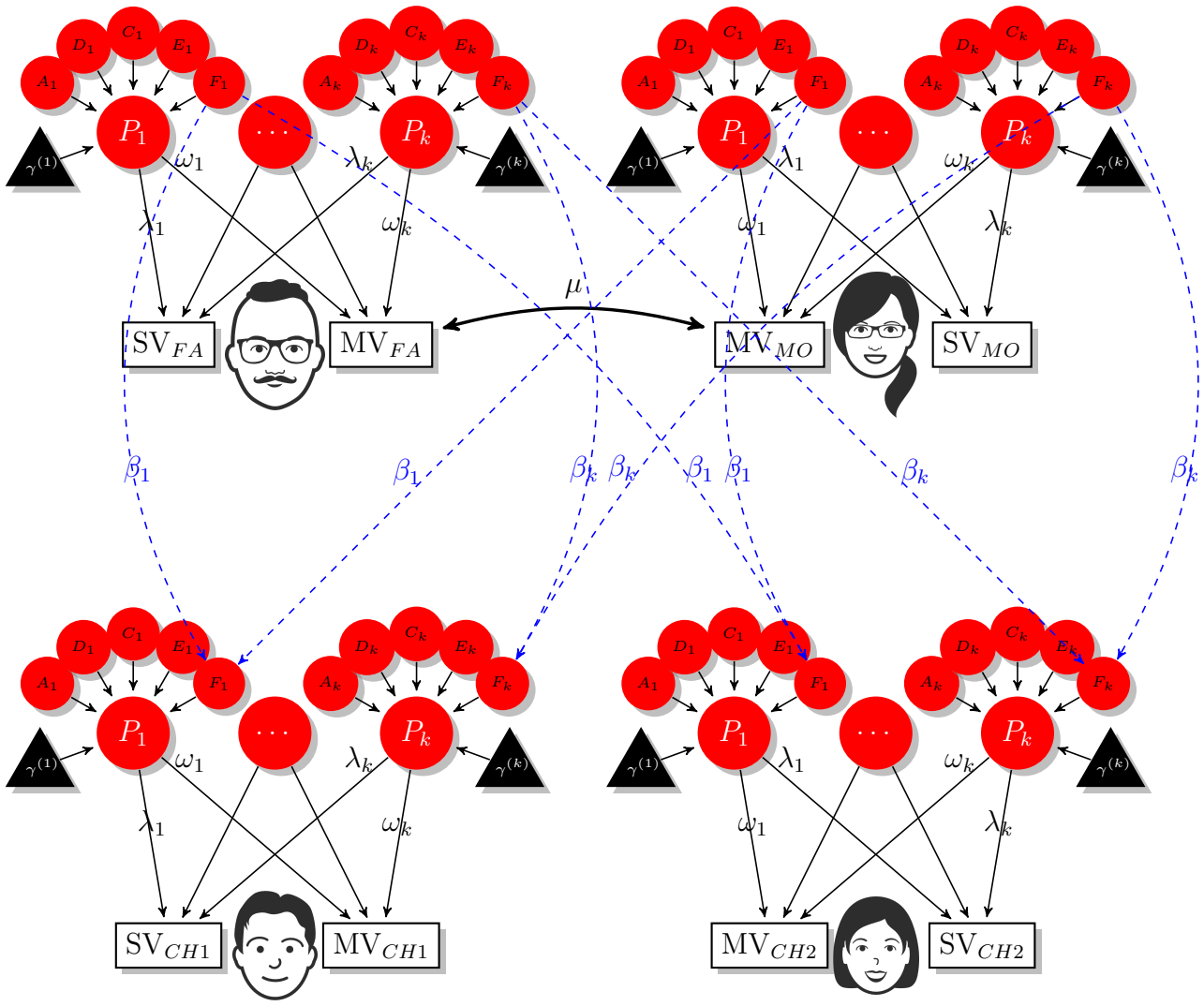


Figure 2.7: Mating path diagram and phenotypes with `--vt_type 2`.

where P_{ij} is the jth phenotype for individual i and λ_j's are some user define coefficients (see Figure 2.6).

2.6.1 No selection

In order to have no selection, you can easily use the `thr 1 1` in columns 4-6 of the file specified in the `--file_gen_info` parameter (see Example 1.5.1).

2.6.2 Directional selection

In population genetics, *directional selection* is a mode of natural selection in which one direction of phenotypic values is favored over the other direction, causing the allele frequency to shift over time in the direction of that phenotype. Under directional selection, the advantageous alleles increases as a consequence of differences in survival and reproduction among different phenotypes. The increases are independent of the dominance of the allele, and even if the allele is recessive, it will eventually become fixed, although at a slower rate.

Based on the individuals selection value and selection function, *GeneEvolve* decides to let them mate or not. User can define the following parameters as selection function in file specified in the `--file_gen_info` parameter (columns 4-6):

```
logit p1 p2
probit p1 p2
stab p1 p2
thr p1 p2
```

where $p1$ and $p2$ are its parameters. For more information about `--file_gen_info`, see Figure 2.1.

For the `logit` function, the probability of mating is computed from inverse *logit* function, i.e.,

$$\mathbb{P}[\text{mating}] = \frac{\exp(p_1 + p_2 \text{SV}_i)}{1 + \exp(p_1 + p_2 \text{SV}_i)}. \tag{2.10}$$

For the `probit` function, the probability of mating is computed from inverse *probit* function, i.e., normal CDF with mean p_1 and standard deviation p_2.

2.6.3 Stabilizing selection

For the `stab` function (stabilizing selection), the probability of mating is computed from the normal PDF with mean p_1 and standard deviation p_2.

$$\mathbb{P}[\text{mating}] = \frac{1}{\sqrt{2\pi} p_2} \exp\left[-\frac{1}{2}\left(\frac{\text{SV}_i - p_1}{p_2}\right)^2\right]. \tag{2.11}$$

2.6.4 Threshold selection

For the `thr` function (threshold selection), the probability of mating is computed from

$$\mathbb{P}[\text{mating}] = \begin{cases} p_1 & \text{if } \text{SV}_i < p_2 \\ 1 & \text{if } \text{SV}_i \geq p_2 \end{cases}. \tag{2.12}$$

2.7 Familial effect

Sometimes non-genetics characters run in families and influence the phenotypes of offsprings. For example, families with higher education may have more educated offspring for reasons that are purely or partly environmental in nature (e.g., due to inherited wealth). We can model this familial effect in *GeneEvolve*, easily.

For each offspring i, the familial effect can be computed from the following equation

$$F_i = \beta(P_i(FA) + P_i(MO)), \tag{2.13}$$

where $P_i(FA)$ and $P_i(MO)$ are parent's phenotypes (see Figure 2.6) and β will automatically calculated for generation 1 such that the V_F becomes equal to the user specified `--vf [number]`. Specifically, $\beta = \sqrt{v_f/(2V_P)}$, where v_f is user specified and V_P is calculated form sample. This case is equal to using the option `--vt_type 1`.

If a user sets `--vt_type 2`, then *GeneEvolve* can also compute the familial effect from equation

$$F_i = \beta(F_i(FA) + F_i(MO)), \tag{2.14}$$

where $F_i(FA)$ and $F_i(MO)$ are parent's familial effect (see Figure 2.7). In this case, $\beta = \sqrt{v_f/(2V_F)}$.

Note that *GeneEvolve* assumes the individuals at generation zero are independent, so there is no familial effect at generation zero. So at the generation 1 where the program simulate some relatives, then this parameter will show up.

2.8 Shared sibling (common) effect

Shared sibling (common) effect is defined as the environmental effects shared by groups of individuals, for example effects shared by groups of relatives that are not due to genetic effects.

For each sibling i in a family f, we add a random number c_f to its phenotype. The generated random number c_f comes from a standard Gaussian distribution with mean zero and the user specified variance VC. User can define VC in *GeneEvolve* by assigning a positive number in `--vc [VC]`.

If there are more than one phenotype, user can define different values for VC for each phenotype.

Note that *GeneEvolve* assumes the individuals at generation zero are independent, so there is no common effect at generation zero. So at the generation 1 where the program simulate some siblings, then this parameter will show up.

2.9 Environmental effects specific to each population

The aim of this subsection is to define the γ_p term in Equation 2.2 (Figure 2.8). For simplicity, assume there is just one phenotype. For k phonetypes, the idea is the same and user should input $\gamma^{(i)}$ for $i \in \{1, \ldots, k\}$ (see Figure 2.6).

Assume that there are P populations. For a user-specified γ, the environmental effects specific to a population $p \in P$, is γ_p, which can be obtained from solving the following equation

$$\text{var}(Y) = (1 + \gamma)\text{var}(X), \tag{2.15}$$

where, X is all the phenotypes obtained from the combined populations, i.e.

$$Y = \bigcup_{p \in P} \bigcup_{i \in \text{pop}(p)} S_{i,p},$$

and

$$X = \bigcup_{p \in P} \bigcup_{i \in \text{pop}(p)} T_{i,p},$$

where for each individual i in population p,

$$S_{i,p} = A_i + D_i + F_i + E_i + C_f + b_p, \tag{2.16}$$

$$T_{i,p} = A_i + D_i + F_i + E_i + C_f, \tag{2.17}$$

where

$$b_p = \Gamma(\frac{2(p-1)}{P-1} - 1).$$

After solving for Γ, the environmental effects specific to population p, becomes

$$\gamma_p = \Gamma(\frac{2(p-1)}{P-1} - 1). \tag{2.18}$$

We use the Newton-Raphson's method to solve the equation 2.15. Equation 2.15 simply says that we can add the constants γ_p to each population p in order to increase (decrease) the overall variance (variance of all the populations together) by the factor γ, while the variance of each population kept constant.

2.10 Simulating several populations

In *GeneEvolve* you can easily simulate several population. For simulating the second population, user should use the parameter `--next_population` in order to distinguish between populations parameters. There is no limit in the number of populations, but you should use the parameter `--next_population` to separate them. In particular, *GeneEvolve* considers all the parameters prior to the `--next_population` as a set of parameters for the first population and treat it as one population with multiple phenotypes. The parameters after `--next_population` (and possibly before the second `--next_population`) will be considered for the second population and so on.

2.11 Population structure

2.11.1 Introduction

A population may have substructures – differences in genetic and environmental effects, in mating, or selection among its subpopulations. Furthermore, exchange of individuals (migration) may differ across subpopulations and may be asymetric. To model migration, assume that a population consists of p subpopulations and that the proportion of individuals migrating from subpopulation j to subpopulation i each generation is m_{ij}. As a result there is a matrix of gene flow parameters, called the backward *migration matrix*, that describes the gene flow pattern among subpopulations (see Table 2.2). The proportion of non-migrants (or residents) for subpopulation i is given by m_{ii} (see figure 2.8). Each row of this matrix sums to unity because it describes the proportion coming from every other possible subpopulation to that particular subpopulation or

$$\sum_{j=1}^{p} m_{ij} = 1.$$

The columns of the matrix will generally not sum to unity, and the matrix will generally not be symmetric (i.e., $m_{ij} \neq m_{ji}$.). The migration matrix is denoted by

$$\mathrm{M} = \begin{bmatrix} m_{11} & m_{12} & \dots & m_{1p} \\ m_{21} & m_{22} & \dots & m_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ m_{p1} & m_{p2} & \dots & m_{pp} \end{bmatrix}.$$

Each of the subpopulations may have a different frequency of A_1, and let us indicate the allele frequency in the jth subpopulation as q_j. Therefore, the frequency of A_1 in the ith subpopulation after gene flow is

$$q_i' = \sum_{j=1}^{p} m_{ij} q_j.$$

Table 2.2: The migration matrix

Subpopulations in generation $t+1$	Subpopulations in generation t				Total
	1	2	...	p	
1	m_{11}	m_{12}		m_{1p}	1
2	m_{21}	m_{22}		m_{2p}	1
$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$
p	m_{p1}	m_{p2}		m_{pp}	1

We can symbolize the process of allele frequency change over all the subpopulations by using matrix notation. First we can indicate the migration matrix as $\boldsymbol{M}$ and the vector of allele frequency for the different subpopulations in generation t with $\boldsymbol{Q}_t$. Therefore,

$$\boldsymbol{Q}_{t+1} = \boldsymbol{M}\boldsymbol{Q}_t.$$

It is also possible that the migration matrix $\boldsymbol{M}$ changes over generation, so we denote it by $\boldsymbol{M}_t$.

$$\boldsymbol{Q}_{t+1} = \boldsymbol{M}_t\boldsymbol{Q}_t.$$

2.11.2 Population structure in *GeneEvolve*

The migration matrix can be inputed to *GeneEvolve* using the parameter

```
--file_migration [file_migration.txt]
```

The inputed file has no header and it has n rows, where n in the number of generations. This file should have k^2 columns as follows where k is the number of populations:

$$m_{11}, m_{12}, \ldots, m_{1k}, m_{21}, \ldots, m_{2k}, \ldots, m_{k1}, \ldots, m_{kk}$$

Therefore, the user can use different migration matrix $\boldsymbol{M}_t$ for each generation.

Example 2.6 (Migration file format)**.** In the following listing file, there are 10 generations with 2 subpopulations.

```
1 .0 .05 .95
.9 .1 .05 .95
.8 .2 .05 .95
.9 .1 .05 .95
.9 .1 .05 .95
.9 .1 .05 .95
.9 .1 .05 .95
.9 .1 .05 .95
.9 .1 .05 .95
.9 .1 .05 .95
```

file_migration.txt

For generation 3, we have

$$\boldsymbol{M}_3 = \begin{bmatrix} .8 & .2 \\ .05 & .95 \end{bmatrix}$$

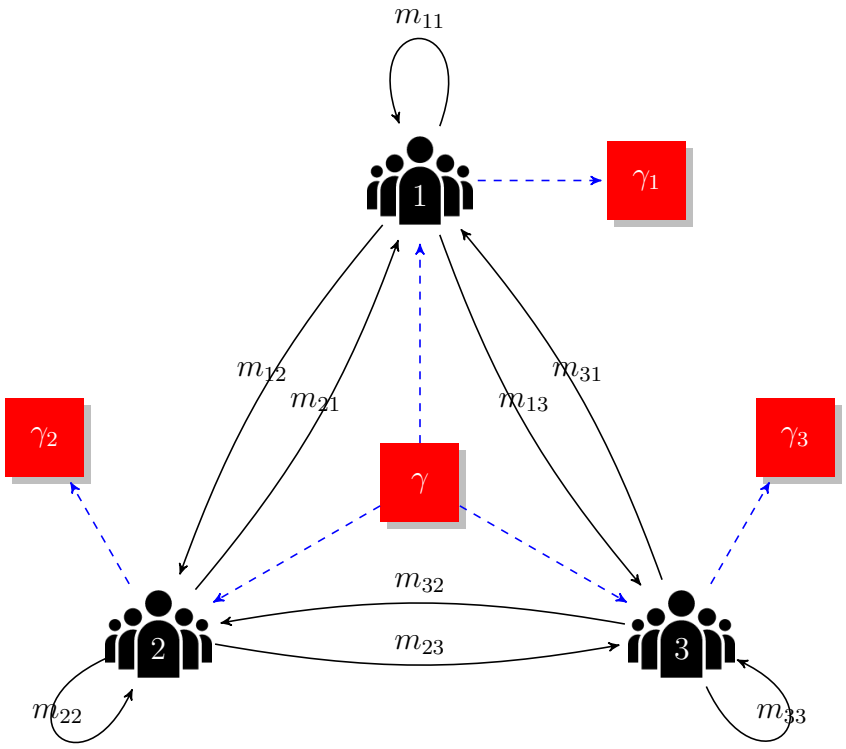


Figure 2.8: Path diagram for migration (m_{ij}) and environmental effects specific to each population (γ_i)

Chapter 3

Results

GeneEvolve is a stand-alone program. Its main advantages are its speed, memory efficiency, ability to simulate realistic SNP and/or sequence data given potentially complex evolutionary events, and ease of use. *GeneEvolve* has been developed and tested under a Linux and Mac environments. It can also be installed on a Windows platform.

In this section, we briefly review the performance of *GeneEvolve* and estimate several population genetics parameters of interest from simulated data and compare these to expected parameters based on population genetics theory.

3.1 Time and memory

In order to show the application of *GeneEvolve* on real data with large populations, we use a combined sample of 33,253 individuals (after cleaning and phasing genotypes) from 9 whole-genome SNP datasets in the NCBI dbGaP Database [8, 13] and, a sample of 3,781 whole-genome sequences from the UK10K project [10, 1] as the initial reference panels (see Acknowledgements).

The time and memory used by *GeneEvolve* per generation under different population sizes are reported in Table 3.1, with both SNP and sequence data as reference panels. In this table, we just report the time used by the main body of simulation and do not consider the reading and writing whole genome SNP/sequence data, which depend on the computer system configurations.

Table 3.1: Average time (seconds) and memory used (Megabytes) in *GeneEvolve* per generation. For the whole-genome SNP data 320,926 SNPs are used and for the whole-genome sequence data 22,989,093 SNPs are used.

Population size	Type	Spousal correlation	Memory used (MB)	Time (s)*
3,000	SNP	0	28.8	5.8
3,000	SNP	0.4	29.6	5.5
30,000	SNP	0	254.9	57.7
30,000	SNP	0.4	257.3	56.2
300,000	SNP	0	2,526.4	1,121.8
300,000	SNP	0.4	2,530.1	991.8
300,000	SEQ	0	2,566.3	1,277.5

* The time for reading and writing genome SNP/sequence data is not considered.

So *GeneEvolve* takes similar time regardless of the marker density (SNP vs. sequence data). For sample size of 300K, it takes ∼20 minutes per generation and uses ∼2.5 Gb memory. However, the time to write out data depends entirely on the marker density. The memory used for 10k, 20k and 30k individuals are plotted in Figure 3.1, starting from UK10K as the reference panel, and using 1000 CVs.

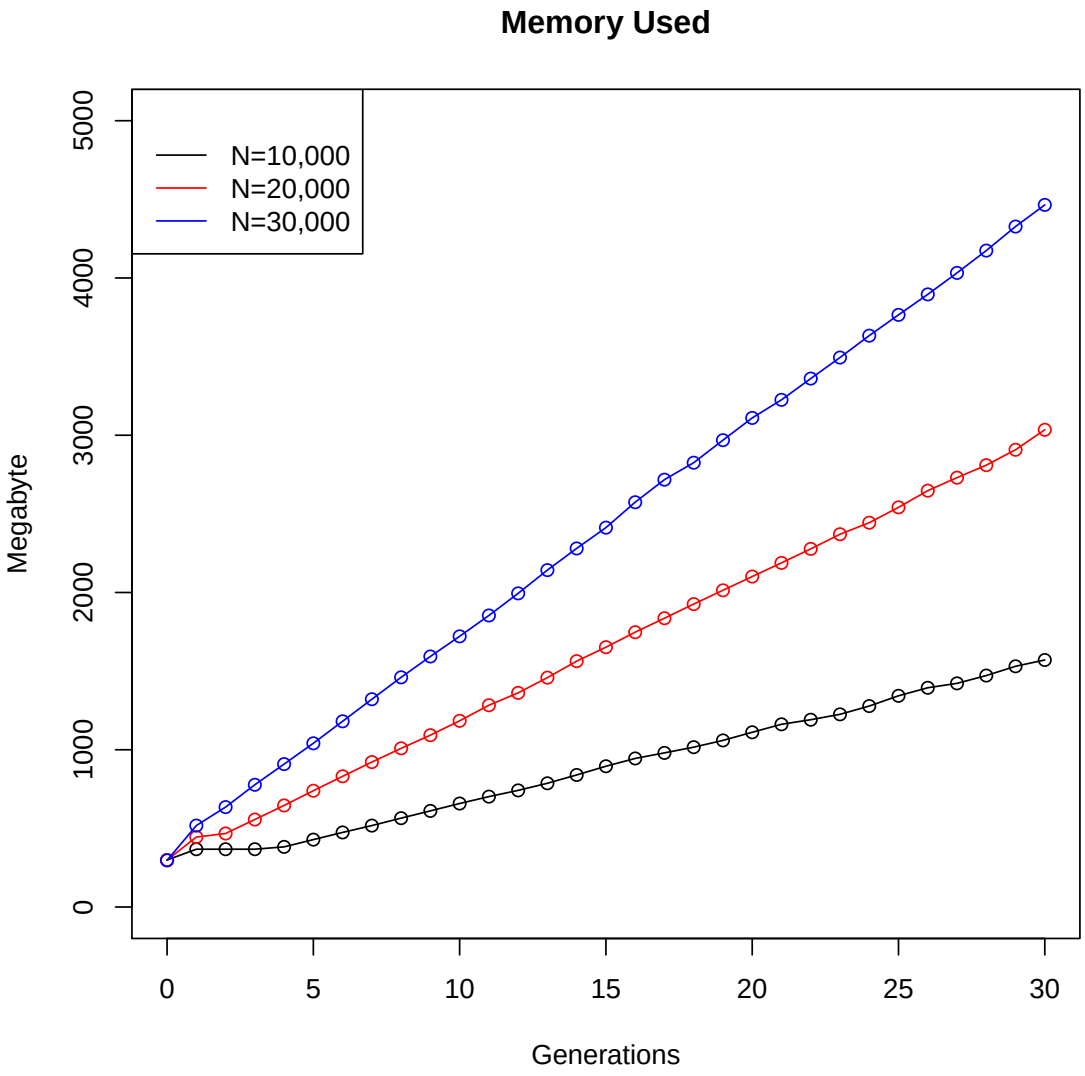


Figure 3.1: The memory used (Megabytes) for 3 populations over 30 generation.

3.2 Minor allele frequency

Under random mating and no selection model, the changes in minor allele frequency (MAF) for five randomly selected SNPs over 40 generation is plotted in Figure 3.2. In this example, we used a simulated genotype as the initial reference panel.

3.3 The effects of genetic drift on genetic variation

The *heterozygosity* (or allelic diversity) of the population in generation t is defined as

$$H_t^0 = \frac{2X_t(2N - X_t)}{2N(2N-1)} = \frac{2N}{2N-1} 2p_t(1-p_t), \tag{3.1}$$

where X_t is the number of copies of allele A in generation t, N is the population size, and $p_t = X_t/2N$. As is clear from the first definition, the heterozygosity is equal to the probability of sampling both alleles when we sample two chromosomes at random and without replacement from the population.

It is well known that under the Wright–Fisher model, the expected heterozygosity $h(t) = \mathbb{E}[H_t^0]$ decreases geometrically at rate $(1 - 1/2N)$:

$$h(t) = \left(1 - \frac{1}{2N}\right)^t h(0). \tag{3.2}$$

In this section, we simulate a population of size 33,253 individuals under the Wright–Fisher model for 100 generations starting from NCBI dbGaP Database [8, 13] as the reference panel. In order to check the effects of genetic drift on genetic variation, we computed the heterozygosity (eq 3.1) and the the expected heterozygosity (eq 3.2).

For each SNP, we define the mean square difference between equations 3.1 and 3.2 as

$$MSE = \frac{1}{g}\sum_{t=1}^{g}(H_t^0 - h(t))^2, \tag{3.3}$$

where g is the number of generations (here, $g = 100$).

In Table 3.2 we categorized SNPs based on their MAF decile and reported the MSE summary statistics for each decile. This table just shows the results of chromosome 22. Clearly, the difference between heterozygosity and its expected value after 100 generations of neutral drift is negligible for each quantile category and for all the SNPs (last line).

Table 3.2: Summary statistics for MSEs in each category of SNPs (chromosome 22).

Percentile	Mean	Std	Median	Min	Max	# SNPs
10	1.40e-04	1.63e-04	8.59e-05	6.21e-06	1.32e-03	477
20	1.57e-04	1.79e-04	9.23e-05	7.80e-06	1.17e-03	476
30	1.79e-04	2.25e-04	9.63e-05	8.47e-06	1.68e-03	476
40	1.77e-04	2.07e-04	1.05e-04	8.84e-06	1.55e-03	480
50	1.53e-04	1.68e-04	9.25e-05	8.33e-06	1.16e-03	473
60	1.28e-04	1.46e-04	7.72e-05	5.64e-06	1.04e-03	478
70	9.74e-05	1.23e-04	5.12e-05	4.10e-06	1.07e-03	475
80	5.33e-05	6.33e-05	3.34e-05	1.40e-06	5.08e-04	477
90	2.06e-05	2.81e-05	1.21e-05	4.77e-07	2.71e-04	477
100	4.65e-06	1.05e-05	1.25e-06	3.53e-08	1.32e-04	476
All	1.11e-04	1.61e-04	5.21e-05	3.53e-08	1.68e-03	4766

From equation 3.2, we can infer that if N is large, then $h(t) \approx h(0)$. So, we also simulated the chromosome 1 for a small population with 100 individuals per generation starting from NCBI dbGaP Database [8, 13] as the reference panel. We then categorize their MAF in groups $[f - .01, f + .01]$ for $f \in \{0.1, 0.2, 0.3, 0.4\}$. The mean of MAF for 100 generations is depicted in Figures 3.3, 3.4, 3.5 and 3.6. As can be seen, they are close to their theoretical value.

3.4 LD structure

In this section, we simulate a population of size 10,000 individuals under the Wright–Fisher model for 30 generations starting from NCBI dbGaP Database [8, 13] as the reference panel. In order to check the LD structure, we computed the r^2 in PLINK using the following parameters for chromosome 1:

```
--r2 --ld-window-r2 0 --ld-window 100 --ld-window-kb 100000
```

We then divided the pairwise estimated r^2 (LDs) uniformly in 20 groups with equal number of SNPs. Table 3.3 summarized some statistics. For each group, we computed the following quantities:

$$MSE = \frac{1}{N}\sum_{i\leq j}(r^2_{ij}(30) - r^2_{ij}(0))^2, \tag{3.4}$$

where N is the number of pairwise LDs in each group, and $r^2_{ij}(30)$ and $r^2_{ij}(0)$ are the LD between SNPs i and j at generation 30 and 0, respectively. The min absolute difference ($\min AD$) and max absolute difference ($\max AD$) are defined as

$$\min AD = \min_{i,j} |r^2_{ij}(30) - r^2_{ij}(0)|, \tag{3.5}$$

$$\max AD = \max_{i,j} |r^2_{ij}(30) - r^2_{ij}(0)|. \tag{3.6}$$

In the last column of Table 3.3, we computed the correlation between LDs at generation 30 and 0, i.e.,

$$\rho = \text{corr}[r^2_{ij}(30), r^2_{ij}(0)]. \tag{3.7}$$

This table shows that there is no meaningful difference between the LD structure at generation 30 and 0, although there are small changes in the their values due to the stochastic nature of recombinations. The overall correlation is .99 and the maximum absolute difference between r^2 values is 0.28. Figure 3.7 plots the histogram of the difference of r^2 values at the initial and the last generations.

To check graphically, we also selected 39 SNPs around a randomly selected SNP and computed their pairwise LDs at the initial population and the last generated population (generation 30). The LD heatmap for generation 0 and generation 30 are plotted in Figures 3.8 and 3.9, respectively. As expected, there are a little changes in the colors.

3.5 Change in additive genetic variance under assortative mating

To compare the estimated variance of the additive genetic variance under assortative mating with its theoretical value, we plot 3 simulation runs with population size of 30,000 in Figure 3.10 for spousal correlation of 0.4. As can be seen, the estimated variance of the additive term under assortative mating is close to its theoretical value.

3.6 Speed and memory comparison

In this section, we checked the speed and memory used by *GeneEvolve* and compared it with SLiM [9], quantiNemo [11] and simuPOP [12]. Table 3.4 summarized their features and abilities using the format presented in [4]. These benchmark simulators need to run for thousands of generations in order to reach

Table 3.3: Summary statistics for the differences in LD structure at the initial and the last generations (chromosome 1).

Group	N	MSE	min AD	max AD	ρ
1	128244	7.85e-05	2.13e-09	1.71e-01	0.9940
2	128244	8.92e-05	1.85e-09	2.05e-01	0.9940
3	128244	9.82e-05	2.36e-09	2.00e-01	0.9933
4	128244	9.08e-05	1.01e-10	2.43e-01	0.9936
5	128244	1.07e-04	7.30e-10	1.66e-01	0.9945
6	128244	8.54e-05	0.00e+00	1.75e-01	0.9938
7	128244	1.04e-04	2.27e-09	1.98e-01	0.9943
8	128244	9.40e-05	0.00e+00	1.65e-01	0.9936
9	128244	1.16e-04	5.00e-10	1.92e-01	0.9942
10	128244	1.03e-04	7.84e-10	2.52e-01	0.9937
11	128244	1.05e-04	8.80e-09	2.05e-01	0.9948
12	128244	9.98e-05	4.51e-10	1.81e-01	0.9937
13	128244	1.04e-04	0.00e+00	2.19e-01	0.9938
14	128244	1.14e-04	2.00e-10	2.22e-01	0.9940
15	128244	1.03e-04	1.56e-09	1.81e-01	0.9948
16	128244	8.05e-05	0.00e+00	1.78e-01	0.9940
17	128244	1.00e-04	6.00e-10	2.35e-01	0.9923
18	128244	1.13e-04	1.09e-09	2.82e-01	0.9931
19	128244	7.50e-05	2.00e-09	1.68e-01	0.9944
20	128244	8.15e-05	5.60e-09	1.51e-01	0.9937
All	2564892	9.71e-05	0.00e+00	2.82e-01	0.9939

the equilibrium (e.g., SLiM suggests to run for 10,000 generations), but *GeneEvolve* can obtain the results after 30 generations since it starts with an at-equilibrium initial reference panel. So for a fair comparison we should compare the memory and time after 10,000 generations for these benchmark simulators and 30 generation for *GeneEvolve*. On the other hand, simulating a large population with 100,000 SNPs and running for 10,000 generation is impossible for these these benchmarks (except for SLiM).

To the best of our knowledge, the computational complexity of some of the recent forward-in-time simulators is $\mathcal{O}(mnpg)$, where m is the number of loci, n is the number of individuals, p is the number of simulated populations and g is the number of generations, but *GeneEvolve* runs in $\mathcal{O}(npg)$ which is not a function of the number of loci, m. So, *GeneEvolve* can simulate real sequence data panels as well as the SNP panels.

Table 3.5 compares the speed and memory with different number of individuals (n) and SNPs (m) for one simulated population. For SLiM, we simulated with $g = 10,000$ and for other software, we used $g = 100$. Based on Table 3.5, *GeneEvolve* can simulate populations more efficiently.

3.7 Limitations

While we have tried to make *GeneEvolve* as comprehensive, easy to use, and fast as possible, no simulation program is suitable for all purposes. *GeneEvolve* excels at creating realistic genomic data quickly, but it has some limitations. Here, we list the limitations of *GeneEvolve* (roughly in what we consider the order

of importance). Many of these limitations we plan to address in the near future.

Addressable limitations:

1. In the current version of *GeneEvolve*, mutations occur just at the available SNP or sequence panel and cannot occur at new sites. So if a mutation occurs in an existing variant, then its value will change, e.g., from G to C. However, it cannot occur at a site that is invariant in the founder population. This means that *GeneEvolve* is currently not well suited to mimicking extremely rare and de novo mutations in sequence data. We plan to address this limitation in the future, but we expect that it will create a computational burden for users who wish to use this feature.

2. The size in memory and time increase linearly with number of generations because the number of haplotype termini from the founder population grows linearly with the number of generations since the founding population. This makes *GeneEvolve* currently unsuitable for studying long-term evolutionary processes that take thousands of generations to reach equilibrium (e.g., mutation-selection). One possible solution can be to run *GeneEvolve* for a small number (e.g., 100) generations, save the output, and then start from the saved output. This process can be repeated as many as needed, although it is not currently user-friendly to do this. We plan to write helper scripts for accomplishing this task in the future.

3. No ability to write out a random sample of the simulated genotypes for each generation. Users might want to simulate a large population (e.g., $> 1M$) to get realistic levels of relatedness, rare mutations, etc., but only want to work with a small subset of these individuals due to space limitations and write time. Currently, *GeneEvolve* writes out all members of the final generation, but we are currently working to change this.

4. *GeneEvolve* currently writes out either all generations or only the final generation; it does not write out multiple specified generations (e.g., the final two generations in order to establish pedigrees). We are currently working on a command that allows this. We also hope to include a helper program that can link the IDs across consecutive generations to create pedigrees.

5. No ability to simulate epistatic effects at the moment. This is not a fundamental difficulty to do, we just haven't gotten around to it. We hope to implement epistatic effects in the future.

6. No ability to simulate correlated environmental factors. This should be easy to add in future revisions.

7. No ability to simulate other causes of mate assortment other than primary phenotypic. Again, this should be easy to add in future revisions.

8. No frequency dependent selection. While users can adjust the allelic effects themselves to simulate heterozygote advantage (by making $d > a$ for a phenotype under directional selection) and antagonstic pleiotropy (by the user including CVs with opposite effects on phenotypes under selection in a multivariate setting), it does not currently have the ability to simulate frequency dependent selection, where the fitness of a phenotype depends on its frequency. We may add this feature in the future.

9. No ability to write out subsets of variants. Currently, *GeneEvolve* writes out all variants that were included in the founder population. This seems like a minor issue, however, because the user can simply include only those variants of interest in the founder population. We do not currently plan to allow *GeneEvolve* to write out subsets of variants but are willing to listen to why this might be needed.

Inherent limitations:

GeneEvolve depends on the user supplying a founder population of finite size. This means that *GeneEvolve* generates simulated data that has gone through at least one generation of population size equal to the sample size of the founder population supplied by the user. With current publicly available SNP datasets now > 500k individuals, this is unlikely to be a major problem for simulating realistic SNP data, but current sequence datasets are typically much smaller (e.g., < 10k individuals). We don't see this as a huge impediment to creating realistic data because (a) the user can simply run enough generations that the founder population has minimal impact on the phenomena of interest; (b) the effective population size of humans is $\sim$ 10k anyway; (c) genomic datasets are growing rapidly; and (d) population bottlenecks of a single generation typically do not have a huge influence on genetic diversity. Nevertheless, this is something that users need to keep in mind when simulating data for specific purposes (e.g., distributions of rare variants) that depend on the founder size.

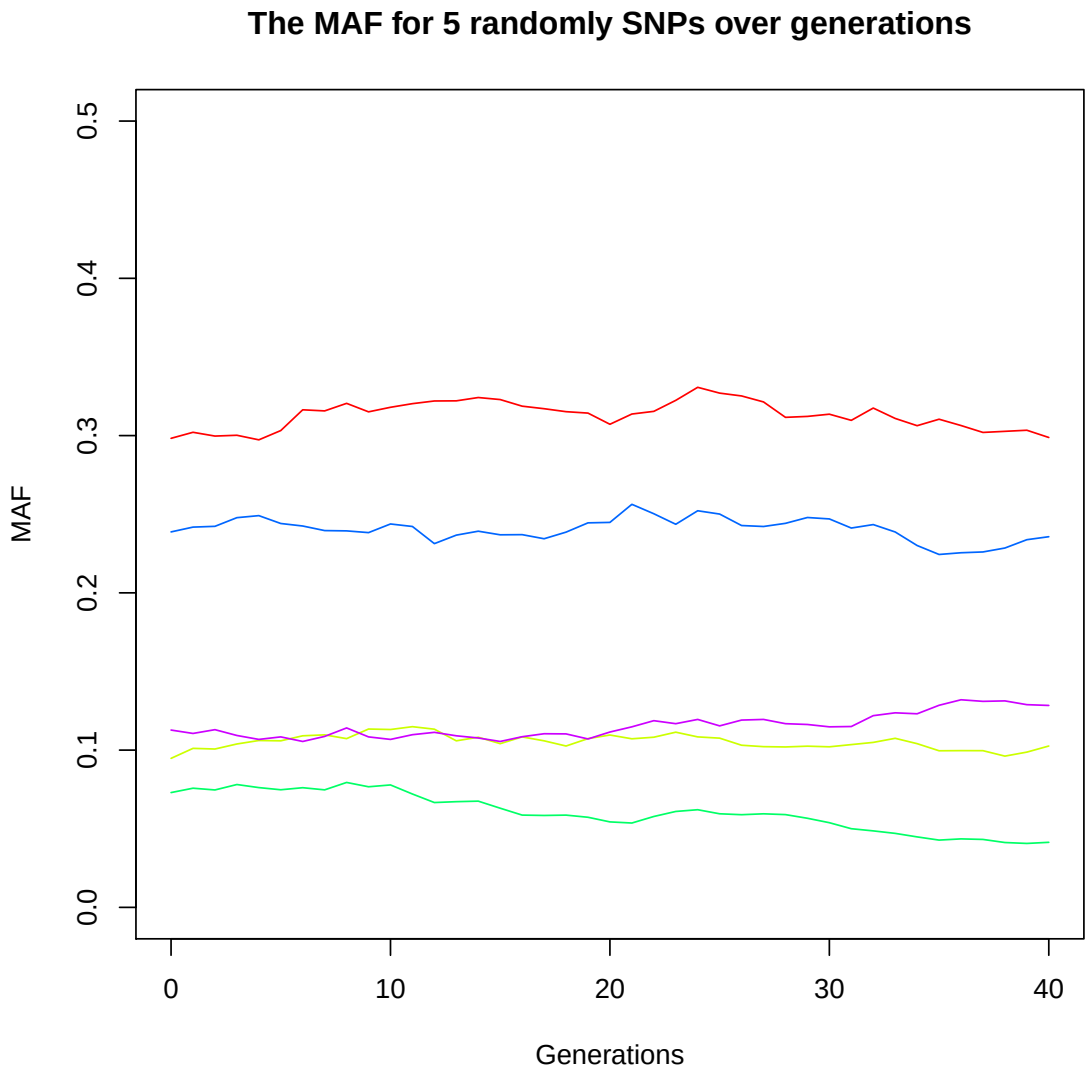


Figure 3.2: Minor allele frequency (MAF) for five randomly selected SNPs over 40 generation.

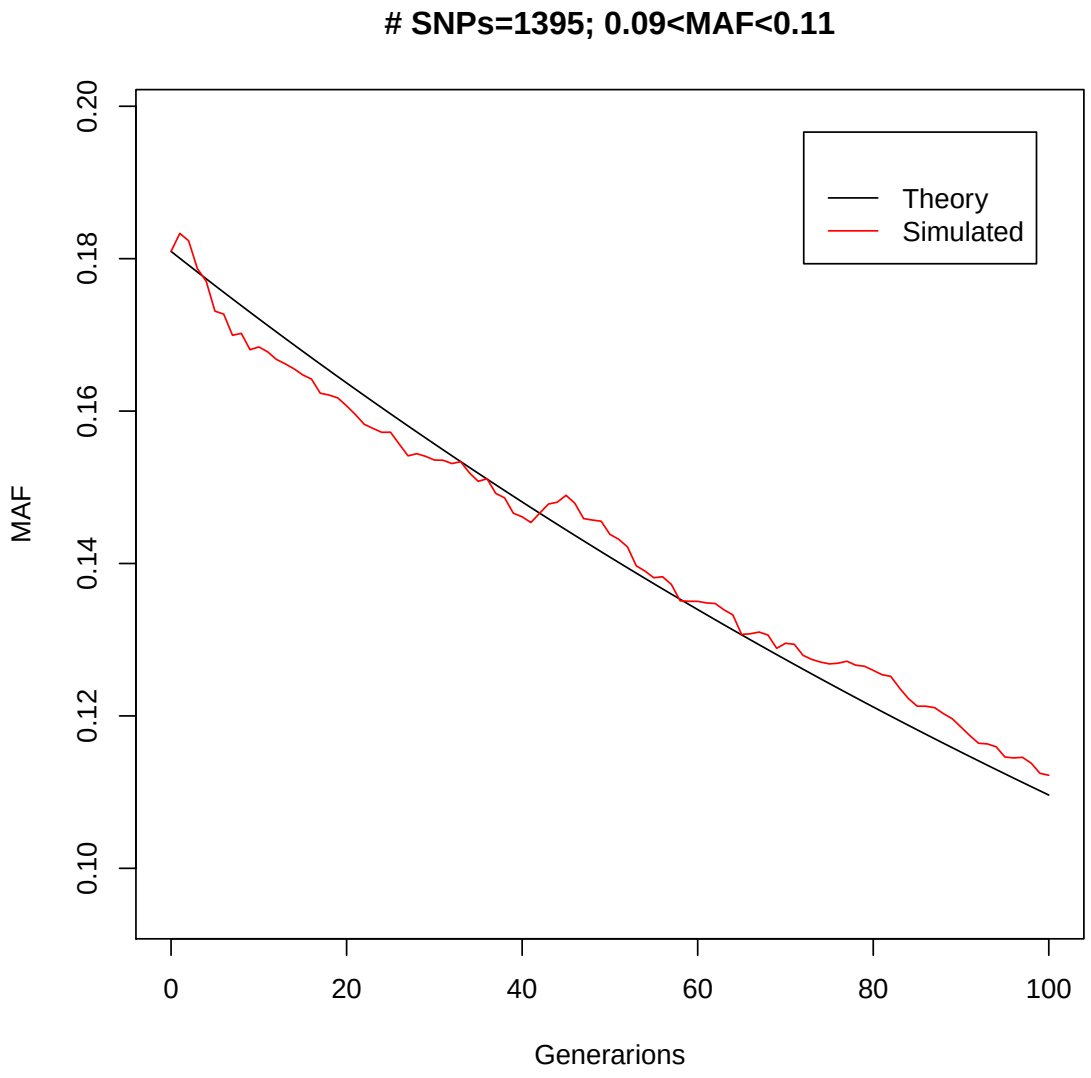


Figure 3.3: Mean of genetic drift for SNPs with MAF in $(0.09, 0.11)$. Population size is 100.

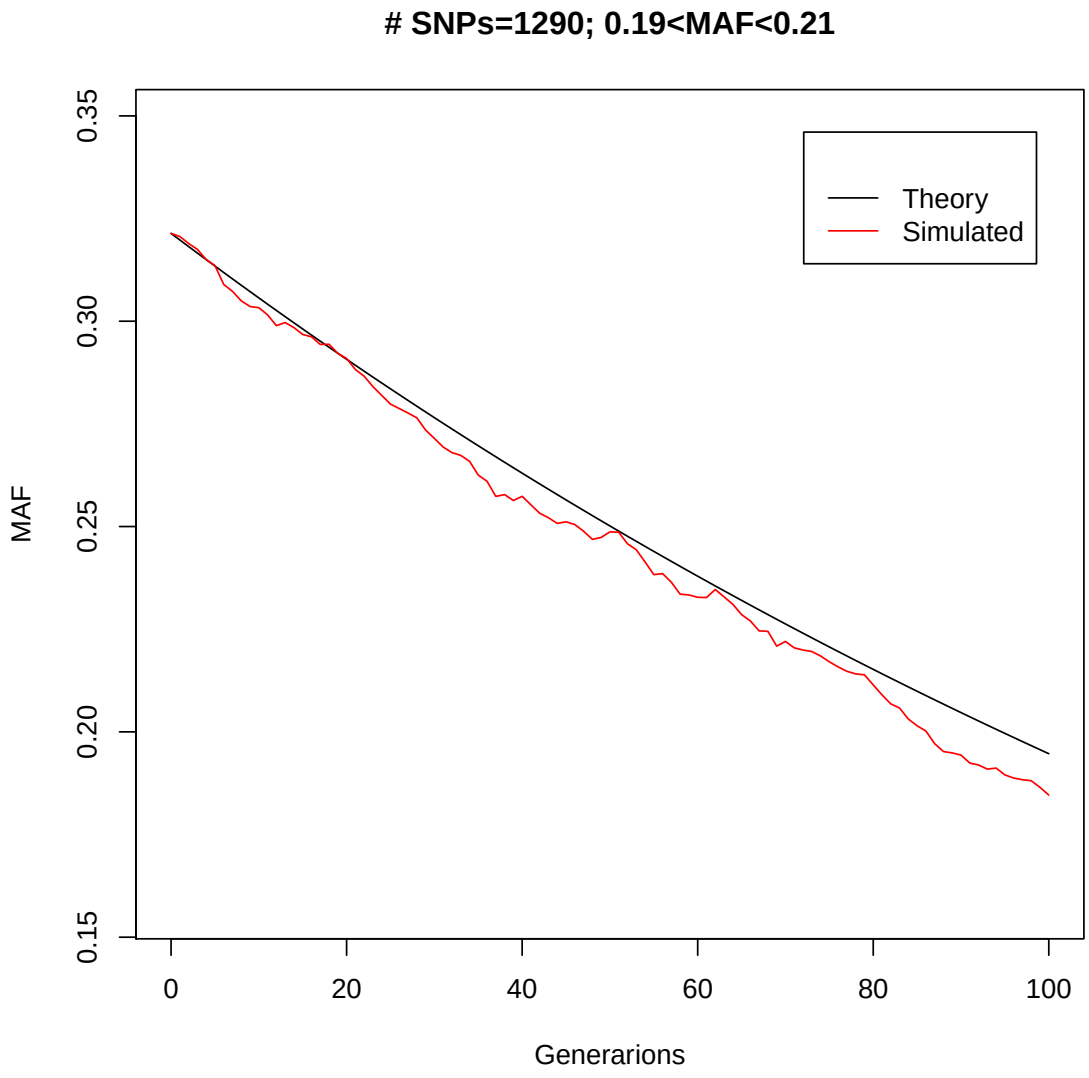


Figure 3.4: Mean of genetic drift for SNPs with MAF in $(0.19, 0.21)$. Population size is 100.

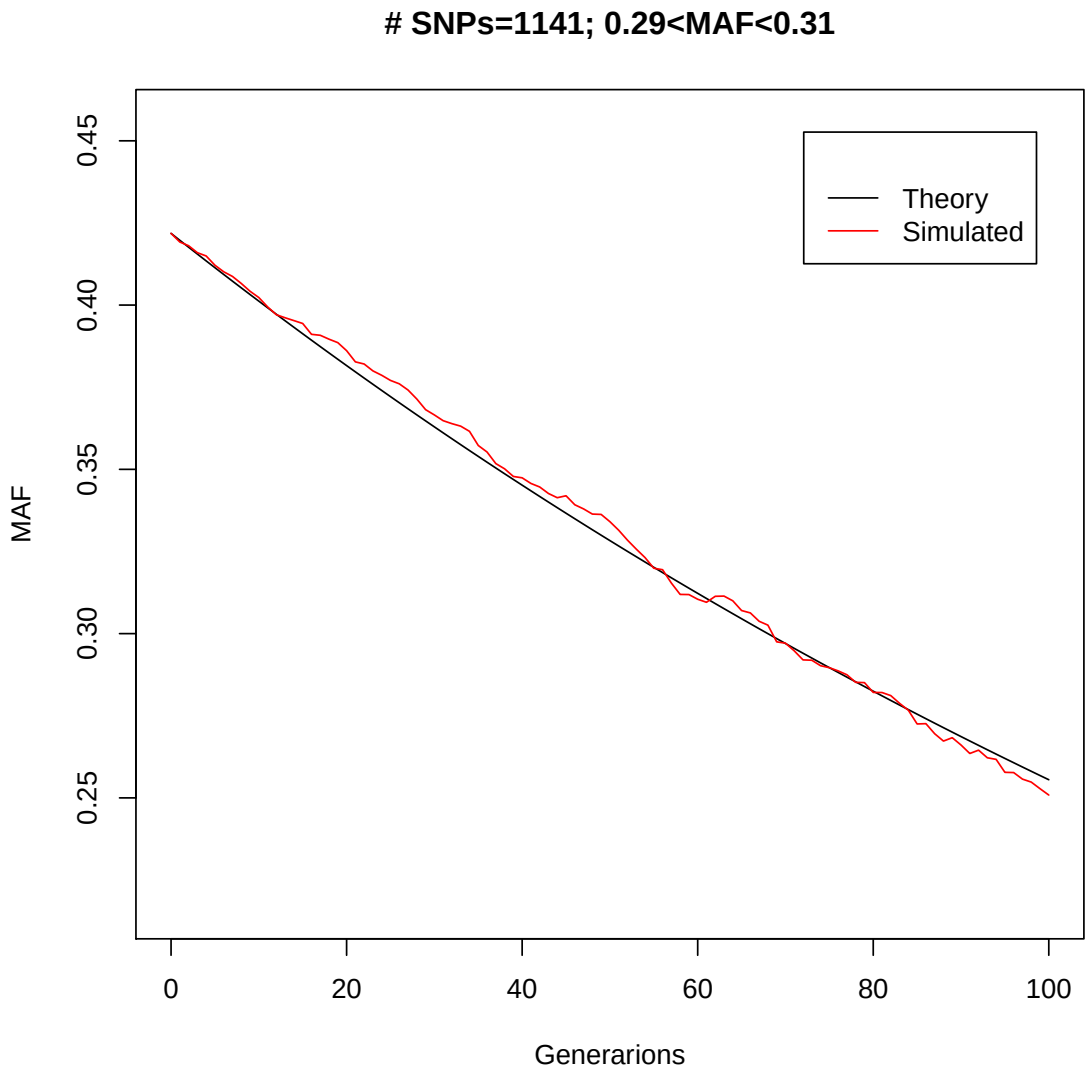


Figure 3.5: Mean of genetic drift for SNPs with MAF in $(0.29, 0.31)$. Population size is 100.

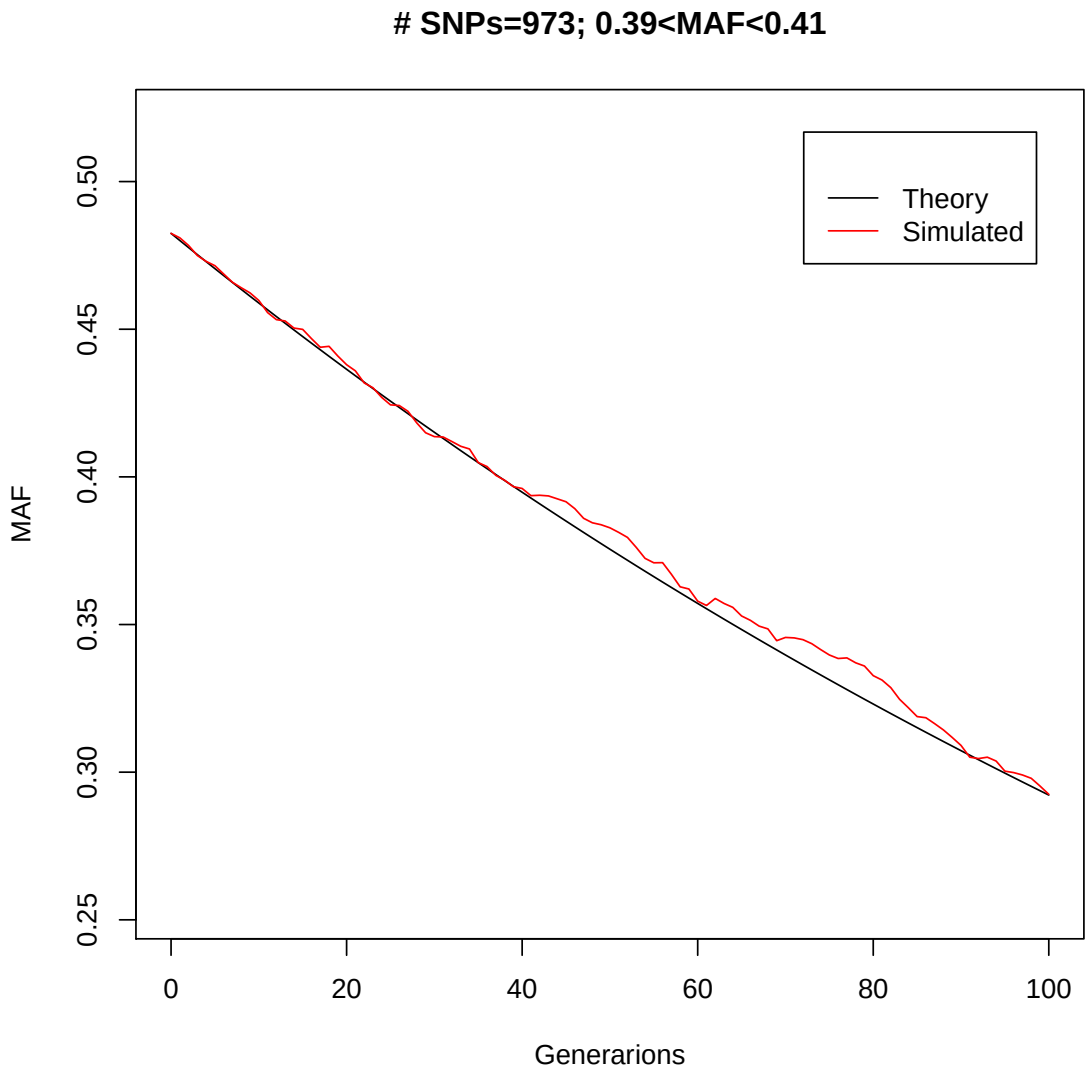


Figure 3.6: Mean of genetic drift for SNPs with MAF in $(0.39, 0.41)$. Population size is 100.

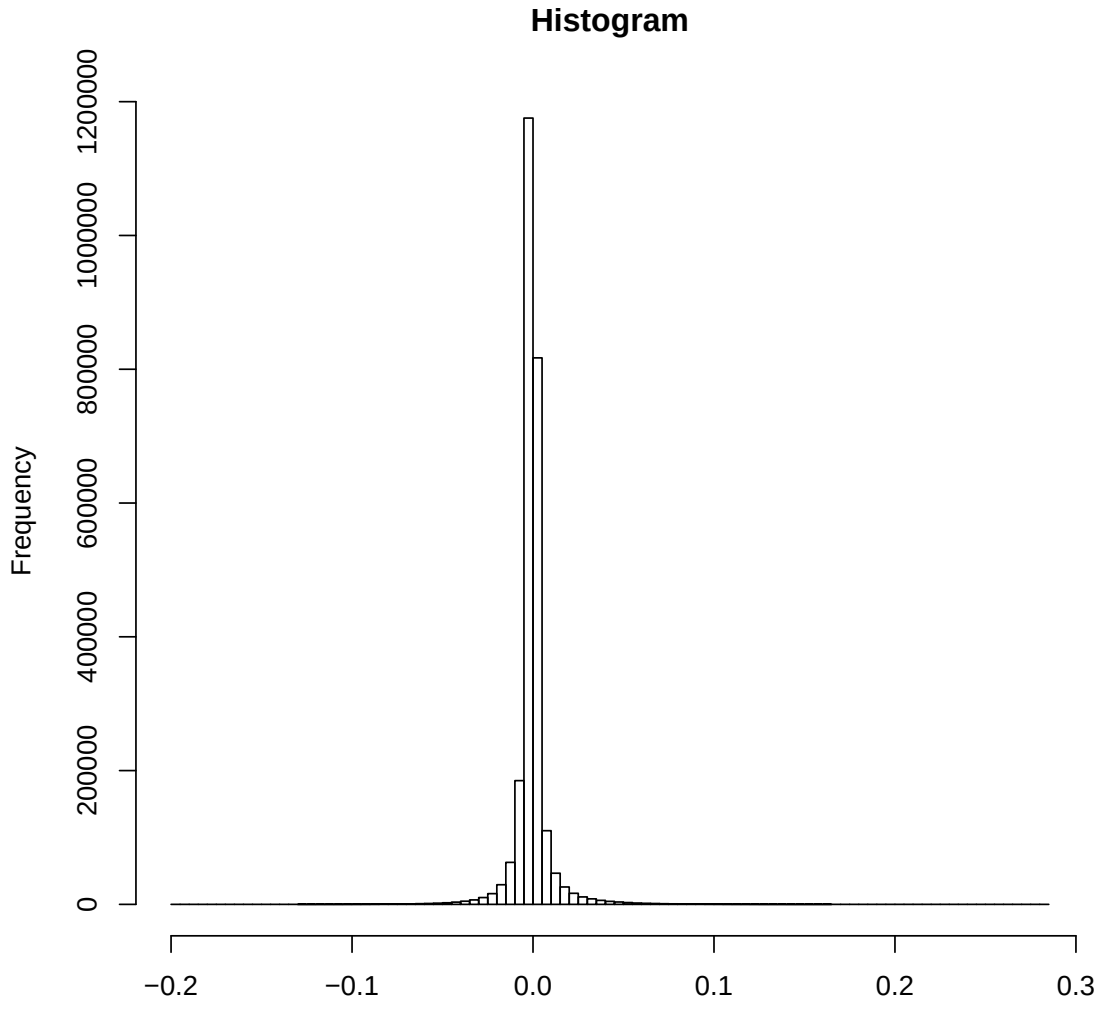


Figure 3.7: Histogram of the difference of LD (r^2) values at the initial and the last generations.

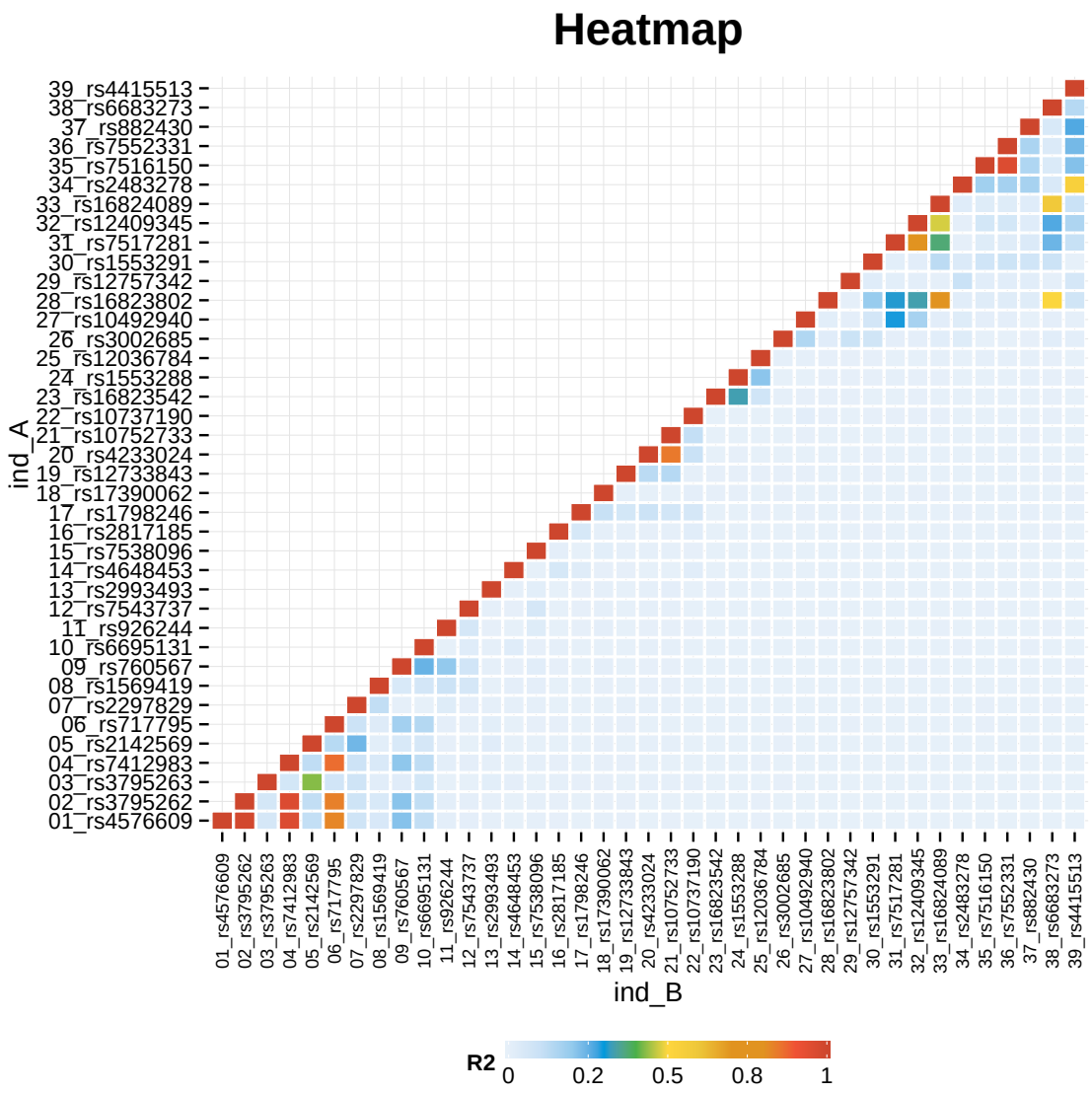


Figure 3.8: LD heatmap for generation 0 for a random segment of the genome.

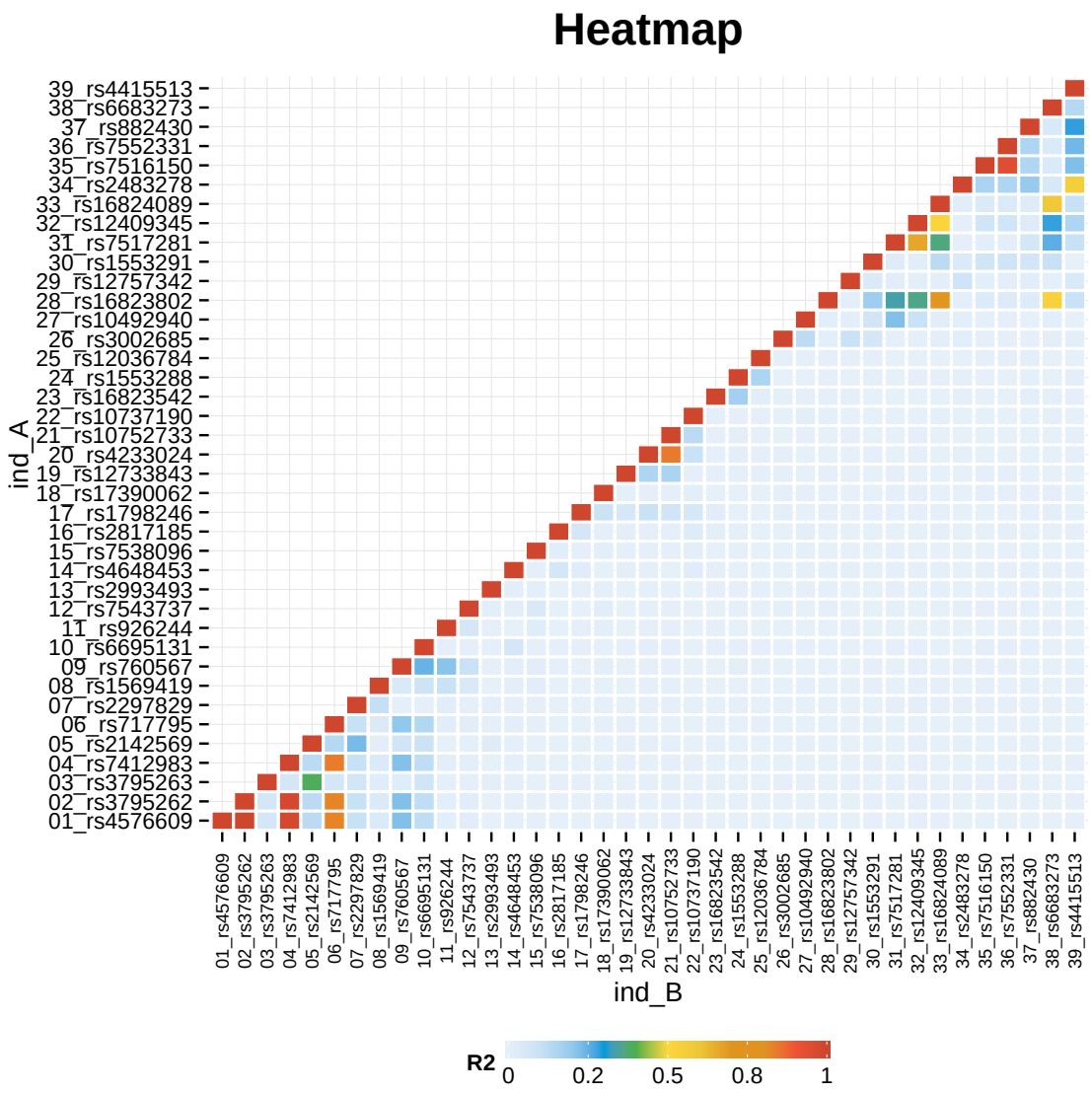


Figure 3.9: LD heatmap for generation 30 for a random segment of the genome.

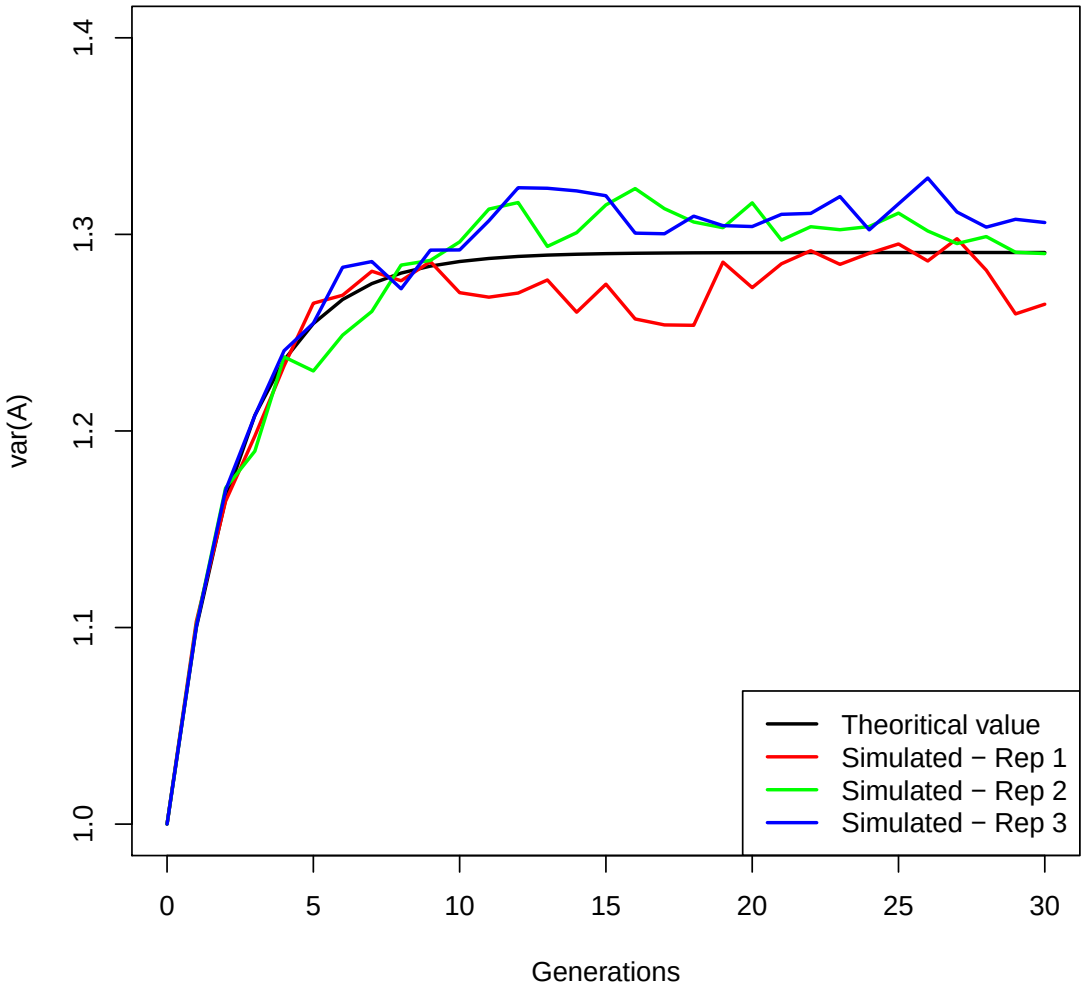


Figure 3.10: Comparing the simulated variance of additive term under assortative mating for three replications (red, green and blue) with its theoretical value (black line).

Table 3.4: Summary of the features and abilities.

	SLiM	quantiNemo	simuPOP	*GeneEvolve*
SEC[a]	-	P	P, I	-
Migration[b]	UD, IM	IM, Pr, SS, UD, DD, Sx	IM, SS, UD, Sx	IM, UD
Mating system[c]	RM, AP, PP, PS	RM, SM, M, PG, PS, CS	RM, M, Pg, PS, H, PP, SM, Pa, AS	M, AP, RM, Pg, RP
Fecundity[d]	Poi, FI	Poi, CN	B, Poi, CN, Sx, UUD, FI	CN, Poi
Life cycle[e]	-	-	O, AR, AM, PM	-
Population growth[f]	UD	K, AS	UD	UD, Ex, PSp
Events allowed[g]	Sel, CE, MM, GR	MM, PopS, Sel, EH	MM, PopS, FF, RH, Sel, MP	MM, CE, RP, FF, PopS, Sel
Selection[h]	MBS, DS, TS, MPG	MPG, DS, BM, TV, EB	MBS, MMS, EB, ME, FS, CB, SP, MP, DS, B, TS, TV	DS, MPG, MBS, TS, B, IF, EB, SBS, SP, TS
Mutation[i]	SNP, KA	M, RIO, H	M, SEQ, SNP, H, In	SNP
Recombination[j]	Y	V, Y	V, Y	V, Y
Platform	C++	C++	Python	C++

[a]Spatially explicit considerations. Populations (P) or individuals (I) can be modeled on a lattice. — [b]Migration or dispersal. DD, density dependence; IM, Wright's Island model; Pr, propagule pool; SS, stepping stone; Sx, sex-specific migration rates; UD, user-defined matrix. — [c]Mating system. AP, assortative based on phenotype; AS, age- or stage-specific; CS, complete self-fertilizing (selfing); H, haplodiploid; M, monogamous; Pa, polyandrous; Pg, polygamous; PP, defined per population; PS, partial selfing; RM, random mating; Sx, can define specific sex ratios; SM, single male. — [d]Fecundity. B, binomial distribution; CN, constant number; FI, influenced by fitness; Poi, Poisson distribution; Sx, sex-influenced; UDD, user-defined distribution. — [e]Life cycle. AR, user-defined age of reproduction; AM, user-defined age of sexual maturity; O, overlapping; PM, user-defined percentage mortality at each stage. — [f]Population growth. AS, age- or stage-specific carrying capacities; Ex, exponential growth; K, carrying capacity; PSp, population-specific; Poi, carrying capacity is determined each generation by a Poisson distribution; UD, user-defined population size every time step. — [g]Events allowed. CE, colonization or extinction (change in number of populations); EH, extinction or harvesting; FF, population fission (splitting) or fusion (admixture); GR, population growth rate; MM, migration matrix; MP, mating probabilities (between sexes and stages); PopS, population size; RP, recombination parameters; Sel, selection strength. — [h]Selection. B, balancing selection; CB, codon-based selection; DS, directional selection; Ep, epistasis; EB, environment-based or population specificity instead of global fitness; FS, frequency-dependent selection; IF, inbreeding affects fitness; ME, multiplicative effects; MBS, multiple biallelic sites; MMS, multiple multilocus selection; MPG, selection on multiple phenotypes or genetic values (which are determined by quantitative trait loci (QTLs)); SBS, single biallelic site; SP, selection on single phenotype; TS, threshold selection removes all individuals with a phenotype above a threshold – stochastic selection removes individuals above threshold with set probabilities; TV, time variable. — [i]Mutation. H, heterogeneity in mutation among sites; KA, k allele model; M, microsatellites; RF, restriction fragment length polymorphisms (RFLPs); RIQ, mutations at a QTL can have a random or incremental effect; Seq, sequence; SNP, single-nucleotide polymorphism. — [j]Recombination. N, no; Un, unknown; V, variation allowed by defining hot spots or a genetic map; Y, yes.

Table 3.5: Time and memory (Megabyte) comparisons for different number of individuals (n) and SNPs (m).

n	m	SLiM*		simuPOP		quantiNemo		*GeneEvolve*	
		Mem	Time	Mem	Time	Mem	Time	Mem	Time
10k	1k	76.95	00:02:20	199.79	00:01:11	1560.47	00:00:51	704.29	00:19:12
10k	2k	109.28	00:02:47	238.16	00:02:37	2323.50	00:01:13	703.90	00:19:41
10k	3k	130.83	00:03:24	276.51	00:03:27	3086.62	00:01:32	707.43	00:19:37
10k	4k	150.21	00:04:31	314.93	00:04:33	3849.71	00:01:47	705.74	00:20:29
10k	5k	150.32	00:05:17	353.34	00:05:45	4612.67	00:02:10	704.61	00:19:16
10k	10k	260.27	00:10:20	545.35	00:11:23	8428.16	00:03:29	700.26	00:19:44
10k	20k	433.16	00:22:15	929.19	00:22:34	16 131.80	00:06:32	704.98	00:22:48
10k	30k	618.66	00:32:54	1312.75	00:34:09	23 720.98	00:09:53	705.97	00:21:12
10k	40k	799.29	00:52:01	1696.80	00:45:00	31 388.30	00:13:12	704.61	00:20:39
10k	50k	1195.26	01:14:34	2080.54	00:57:00	38 977.48	00:17:00	704.49	00:22:09
10k	100k	1855.60	02:51:44	3999.53	01:50:39	77 157.79	00:35:24	699.30	00:22:47
10k	200k	4360.07	08:48:50	7837.62	03:45:42	153 439.31	01:29:36	703.86	00:20:37
10k	300k	6497.29	16:39:31	11 677.63	05:35:16	229 721.80	01:58:23	706.38	00:20:29
10k	400k	killed	>24h	15 513.80	07:31:23	306 082.45	02:35:03	700.36	00:19:49
10k	500k	killed	>24h	19 349.77	09:13:23	382 364.94	03:25:16	703.14	00:19:18
100k	1k	1771.16	00:46:33	553.49	00:11:21	15 470.53	00:10:09	6885.07	02:22:23
100k	2k	1915.82	00:51:46	935.11	00:22:48	23 100.07	00:13:03	6891.88	02:19:34
100k	3k	1538.64	00:50:51	1316.85	00:34:09	30 729.55	00:17:04	6891.93	02:22:10
100k	4k	1312.61	00:44:54	1698.56	00:45:06	38 359.07	00:19:56	6872.79	02:18:40
100k	5k	2035.03	00:52:42	2080.32	00:55:47	45 988.56	00:22:16	6884.25	02:11:44
100k	10k	1335.31	00:50:28	3988.82	01:53:26	84 136.19	00:44:58	6891.82	02:27:04
100k	20k	1401.01	01:04:37	7805.95	03:44:41	160 669.36	01:32:52	6880.55	02:27:15
100k	30k	1687.16	01:22:20	11 622.73	05:33:55	236 828.54	02:08:09	6885.18	02:25:31
100k	40k	1858.50	01:47:19	15 440.02	07:36:46	313 244.23	02:45:41	6894.66	02:16:27
100k	50k	2147.29	02:03:24	19 256.98	09:25:03	389 403.26	03:30:24	6884.18	02:23:43
100k	100k	3528.81	04:21:12	38 342.10	18:39:08	770 967.12	06:44:53	6886.73	02:15:28
100k	200k	5561.63	10:38:47	killed	>24h	>1000GB	killed	6893.66	02:13:19
100k	300k	9523.55	16:57:08	killed	>24h	>1000GB	killed	6875.15	02:13:25
100k	400k	killed	>24h	killed	>24h	>1000GB	killed	6892.97	02:15:39
100k	500k	killed	>24h	killed	>24h	>1000GB	killed	6889.97	02:18:32

*For SLiM, we simulated $g = 10,000$ generations and for other software, we used $g = 100$.

Chapter 4

Parameters Reference

Here is the list of all the parameters used in *GeneEvolve*.

```
GeneEvolve \
--file_gen_info par.pop1.info.txt \
--file_hap_name par.pop1.hap_sample_address.txt \
--file_recom_map Recom.Map.b37.50KbDiff \
--file_mutation_map mutation.Map.b37.50KbDiff \
--file_cv_info par.pop1.cv1_info.txt \
--file_cvs par.pop1.cv1_hap_files.txt \
--va 1 \
--vd .2 \
--vc .1 \
--ve 1 \
--vf .1 \
--omega .7 \
--lambda 1 \
--file_cv_info par.pop1.cv2_info.txt \
--file_cvs par.pop1.cv2_hap_files.txt \
--va 2 \
--va .1 \
--vc .1 \
--ve 1 \
--vf .1 \
--omega .3 \
--lambda 1 \
--next_population \
--file_gen_info par.pop2.info.txt \
--file_hap_name par.pop2.hap_sample_address.txt \
--file_cv_info par.pop2.cv1_info.txt \
--file_cvs par.pop2.cv1_hap_files.txt \
--file_cv_info par.pop2.cv2_info.txt \
--file_cvs par.pop2.cv2_hap_files.txt \
--file_recom_map Recom.Map.b37.50KbDiff \
--file_mutation_map mutation.Map.b37.50KbDiff \
--va 2 \
--vd .2 \
--vc 1 \
--ve 1 \
--vf 0 \
```

```
38 --va 2 \
39 --vd .2 \
40 --vc 1 \
41 --ve 1 \
42 --vf 0 \
43 --omega 1 \
44 --omega 1 \
45 --lambda 1 \
46 --lambda 1 \
47 --file_migration par.migration.txt \
48 --prefix out1 \
49 --gamma 1 \
50 --gamma 1\
51 --avoid_inbreeding
```

In the above listing, we simulate 2 populations, each with 2 phenotypes. In the following subsections, we explain each of the parameters, briefly.

4.1 The order of the parameters

For simulating one population and one phenotype, the order of the parameters do not matter. So for example, you can put `--va` prior to the `--file-cv_info`. But, with multiple phenotypes and one populations, *GeneEvolve* sets the first parameter for the first phenotype and the second parameter for the second phenotype and so on. For example, if you use `--va 3 --va 2`, then the additive variances for the first and second phenotypes will scaled to 3 and 2, respectively. For multiple phenotypes and multiple populations, *GeneEvolve* considers all the parameters prior to the `--next_population` as a set of parameters for the first population and treat it as one population with multiple phenotypes. The parameters after `--next_population` (and possibly before the second `--next_population`) will be considered for the second population and so on.

Note that the number of phenotypes should be the same for each population.

4.2 Parameters

4.2.1 `--file_gen_info`

This is a required parameter with no default value. You should use it for each population.

```
--file_gen_info par.pop1.generaions_info.txt
```

where `par.pop1.generaions_info.txt` is a text file with 6 columns and $n + 1$ lines, where n is the number of generations to be simulated by *GeneEvolve*. This file should have a header and the first column is the population size. The structure of this file is listed in Figure 2.1. *GeneEvolve* can simulate different population sizes in each generation, by specifying some numbers in the first column.

If you want to simulate p populations ($p > 1$), then you need p of this file (possibly with different parameters) for each population with the same number of lines (generations), i.e., the number of generations should be the same for all the populations.

4.2.2 --file_ref_vcf

If the reference panel (population 0) is in `vcf` format, the you should use it for each population.

4.2.3 --file_hap_name

If the reference panel (population 0) is in `hap` format, the you should use it for each population. This file addresses hap, legend and sample files for the reference panel. The reference panel can be SNP or sequence data. Since the SNP/sequence reference panels can be huge, so they should be separated for each chromosome.

```
--file_hap_name par.pop1.hap1_sample_address.txt
```

where `par.pop1.hap1_sample_address.txt` is a text file with 4 columns: chromosome number, the address of the initial hap, legend and sample files.

If you want to simulate p populations ($p > 1$), then you should use this parameter p times for each population. You can use different initial hap files for for each population.

4.2.4 --file_recom_map

This is a required parameter with no default value. You should use it for each population.

```
--file_recom_map Recom.Map.b37.50KbDiff
```

where `Recom.Map.b37.50KbDiff` is a text file with 4 columns, counting the cM distance for all the snap panel.

If you want to simulate p populations ($p > 1$), then you should use this parameter p times for each population. You can use different recombination map files for for each population.

4.2.5 --file_cv_info

This is a required parameter with no default value. You should use it for each phenotype.

```
--file_cv_info par.pop1.cv1_info.txt
```

where the CV information are in file `par.pop1.cv1_info.txt`.

Clearly, If you want to simulate m phenotypes ($m > 1$), then you should use this parameter m times for each phenotype with different sets of CVs. Importantly, you can control the correlation between phenotypes by choosing overlap sets of CVs.

4.2.6 --file_cvs

This is a required parameter with no default value. You should use it for each phenotype.

```
--file_cvs par.pop1.cv1_hap_files.txt
```

where the actual haplotype address are saved in file `par.pop1.cv1_info.txt`.

Clearly, If you want to simulate m phenotypes ($m > 1$), then you should use this parameter m times for each phenotype with different sets of CVs.

4.2.7 `--va [-1]`

This is an optional parameter with -1 as the default value, where -1 means its value will be computed based on the user-specified additive and dominance effects and will not be scaled. We know that variance is not negative, but this value is useful when you want to simulate three phenotypes and you want to scale just the first and third one (in fact -1 is a flag). For example `--va 3 --va -1 --va 2` means the the additive variance of the first and second phenotype should be scaled to 3 and 2, respectively, while the second variance will not scaled. You can use it for each phenotype. You will also get an error if you set `--va 0`.

Note that when you scale the additive or dominance genetic variance using `--va` or `--vd`, the scaling factors (mean and standard devations) are computed in the first generation and then those same scaling factors are used every generation thereafter. This allows the additive or dominance genetic variance to change (e.g., increase with positive AM or decrease due to selection) over time. The user inputted values here therefore only provide the starting place (in generation 0) of these values. Obviously, if we had recomputed the scaling factors each generation, the genetic variances would remain constant.

```
--va 2
```

GeneEvolve will transforms the additive variance to 2, in generation zero.

4.2.8 `--vd [-1]`

This is an optional parameter with -1 as the default value, where -1 means its value will be computed based on the user-specified additive and dominance effects and will not be scaled. We know that variance is not negative, but this value is useful when you want to simulate three phenotypes and you want to scale just the first and third one (in fact -1 is a flag). For example `--vd 3 --vd -1 --vd 2` means the the dominance variance of the first and second phenotype should be scaled to 3 and 2, respectively, while the second variance will not scaled. You can use it for each phenotype. You can also set `--vd 0`.

```
--vd 0.1
```

GeneEvolve will transforms the dominance variance to 0.1, in generation zero.

4.2.9 `--vc [0]`

This is an optional parameter with 0 as the default value. You can use it for each phenotype.

```
--vc 0.1
```

GeneEvolve will transforms the variance of sibling environment (common) effect to 0.1, in all generations.

4.2.10 `--ve [1]`

This is an optional parameter with 1 as the default value. You can use it for each phenotype. You can also set `--ve 0`.

```
--ve 1
```

GeneEvolve will transforms the environmental effect variance to 1, in all generations.

4.2.11 --vf [0]

This is an optional parameter with 0 as the default value. You can use it for each phenotype.

```
--vf 0.1
```

GeneEvolve will transforms the familial effect variance to 0.1, in generation zero.

4.2.12 --RM

This is an optional parameter with no default value.

```
--RM
```

There are two mating systems in *GeneEvolve*: random mating (parents for all offspring are chosen completely at random) and assortative mating (monogamous mating where parents are paired such that a user-specified correlation exist between mates' phenotypes). Assortative mating is the default mating system for *GeneEvolve*, where the spousal correlation is specified in the `--file_gen_info` file. Note that monogamous mating systems where parents are uncorrelated should be specified with a 0 in the second column of the `--file_gen_info` file. User can choose random (non-monogamous) mating system, where parents of each offspring are chosen completely at random, using `--RM`. In this case the spousal correlation specified in the `--file_gen_info` file and the parameter `avoid_inbreeding` will be ignored. Note that there is currently no way to specify non-monogamous mating system and assortative mating simultaneously in *GeneEvolve*.

4.2.13 --avoid_inbreeding

This is an optional parameter. If you use it, then *GeneEvolve* avoids inbreeding (avoids sibs and cousin mating).

```
--avoid_inbreeding
```

This parameter will be ignored if you use the random mating system by `--RM`.

4.2.14 --next_population

This is a required parameter when you want to simulate more than one population.

```
--next_population
```

This parameter will used to give the next population information. Clearly, you should use it $p-1$ times, if you want to simulate p populations, since we do not use it for the first population.

4.2.15 --omega [1]

This is an optional parameter with 1 as the default value. It is the coefficient relating the observed phenotype P to the phenotype upon which mates select each other (MV) (see Figure 2.6). You can use it for each phenotype; for a single phenotype, it does not matter what you set this value to so long as it is not 0; for multiple phenotypes, it tells *GeneEvolve* how to weight each phenotype's influence on the mating phenotype. Note that if the correlation between mates is set to 0 in the second column of `--file_gen_info`, OR if the `--RM` is used, this parameter has no effect.

```
--omega .3
```

4.2.16 `--lambda [1]`

This is an optional parameter with 1 as the default value. It is the coefficient relating the observed phenotype P to the phenotype under natural selection (SV) (see Figure 2.6). You can use it for each phenotype; for a single phenotype, it does not matter what you set this value to so long as it is not 0; for multiple phenotypes, it tells *GeneEvolve* how to weight each phenotype's influence on the selection phenotype. Note that if you do not specify selection in the `--file_gen_info`, file, this parameter has no effect.

```
--lambda .7
```

4.2.17 `--file_mutation_map`

This is an optional parameter with no default value.

```
--file_mutation_map mutation.Map.b37.50KbDiff
```

where `mutation.Map.b37.50KbDiff` is a text file with 3 columns: chromosome number, starting region (bp), and mutation rate, where the last column is the probability of one mutation in that region. The starting point for each region is presented in the second column and the end point is the starting point of the next line. This file has a header.

If you want to simulate p populations ($p > 1$), then you should use this parameter p times for each population. You can use different mutation map files for for each population.

In the current version of *GeneEvolve*, mutations occur just in the available SNP panel and do not create a new variable. So if a mutation occurs in a variant, then its value will change, e.g., form G to C. If a rare variant goes extinct, then its value for all the individuals will be zero (or one). Therefore, the legend file is always as same as the first generation legend file.

4.2.18 `--gamma [0]`

This is an optional parameter with 0 as the default value. It is the environmental effects specific to each population (see Section 2.9). If you set it to zero or not use it, then *GeneEvolve* will not generate population specific environmental effects.

```
--gamma 1
```

4.2.19 `--file_migration`

This is a required parameter if you simulate more than one populations.

```
--file_migration par.migration.txt
```

The inputed file `par.migration.txt` has no header and it has n rows, where n in the number of generations. This file should have k^2 columns as follows where k is the number of populations:

$$m_{11}, m_{12}, \ldots, m_{1k}, m_{21}, \ldots, m_{2k}, \ldots, m_{k1}, \ldots, m_{kk},$$

where m_{ij} is the probability of migrating from population i to j. By this format, a user can use different migration matrix per generation. For example in the following listing file, the probability of migrating for 10 generations and 2 populations are listed.

```
1 .0 .05 .95
.9 .1 .05 .95
.8 .2 .05 .95
.9 .1 .05 .95
.9 .1 .05 .95
.9 .1 .05 .95
.9 .1 .05 .95
.9 .1 .05 .95
.9 .1 .05 .95
.9 .1 .05 .95
```

file_migration.txt

As and example, for generation 3, the migration matrix is

$$\boldsymbol{M}_3 = \begin{bmatrix} .8 & .2 \\ .05 & .95 \end{bmatrix}.$$

4.2.20 `--out_hap`

The output file format. This option tells *GeneEvolve* to save the simulated genotypes in hap format. *GeneEvolve* can output in different formats at the same time.

4.2.21 `--out_plink` and `--out_plink01`

The output file format. This option tells *GeneEvolve* to save the simulated genotypes in plink format. For more information about plink file formats, see the appendix C.

4.2.22 `--out_vcf`

The output file format. This option tells *GeneEvolve* to save the simulated genotypes in vcf format.

4.2.23 `--out_interval`

The output file format. This option tells *GeneEvolve* to save the simulated genotypes in interval format.

The "interval" output format is the true identity by descent information for haplotypes. For more information about interval file format, see the appendix C.

4.2.24 `--file_output_generations [filename]`

A file containing the generation numbers per line for saving the simulated genotypes.

4.2.25 `--prefix [out]`

This is an optional parameter with "out" as the default value. This parameter determines the prefix for the output files.

```
--prefix example1
```

Acknowledgements

For the base population used to simulate Figure S2 and Table S2, we used a sample of 33,253 individuals from 9 datasets in the NCBI dbGaP Database of Genotypes and Phenotypes: the Health and Retirement Study (dbGaP accession phs000428.v1.p1), the Genetic Epidemiology of COPD (dbGaP accession phs000179.v4.p2), the Chronic Renal Insufficiency Cohort Study (dbGaP accession phs000524.v1.p1), the Geisinger eMERGE Abdominal Aortic Aneurysm Project (dbGaP accession phs000387.v1.p1), the Geisinger eMERGE MyCode Project (dbGaP accession phs000381.v1.p1), the Genes and Blood Clotting Study (dbGaP accession phs000304.v1.p1), the Polycystic Ovary Syndrome (PCOS) Genetics study (dbGaP accession phs000368.v1.p1), the WHI GARNET study (dbGaP accession phs000315.v6.p3, and the WHIMS+ study (dbGaP accession phs000675.v2.p3) from the NIH Genotype and Phenotype database.

For the base population used to simulate the last row of Table S2, we used the UK10K project sequence data with a sample of 3,781 individuals (with accession numbers EGAD00001000776).

Appendix A

Genotype simulation for an initial population

To run *GeneEvolve*, we first need genotype data. Here, is a simple R code for simulating genotypes for 3 chromosomes. Usually users will be inputting real, phased SNP or sequence data into GeneEvolve, but this simulated data is useful for getting started and/or for educational purposes.

```
#####################################################
# create genotypes in hap, legend, indv format
# creating CVs
#####################################################
set.seed(12345)
NCHR <- 3
NSNP <- rep(1000,NCHR) #number SNPs per chromosome
NCV <- rep(100,NCHR) #number CVs per chromosome
NIND <- 2000
VAR.A <- 1
VAR.D <- .1
VAR.A2 <- 1 # for the second CV set
VAR.D2 <- .3 # for the second CV set
INCLUDE.CHRS <- 1:NCHR

# We create this map in order to use the real genomic map distance
# genotypes and cvs should be in the range of genomic map
# These numbers came from the range of genomic map
map.pos <- matrix(ncol=2,nrow=22)
map.pos[ 1 ,] <-  c(738555,249238555)
map.pos[ 2 ,] <-  c(1,242900000)
map.pos[ 3 ,] <-  c(1,197900000)
map.pos[ 4 ,] <-  c(1,1.91e+08)
map.pos[ 5 ,] <-  c(1,180750000)
map.pos[ 6 ,] <-  c(105878,170955878)
map.pos[ 7 ,] <-  c(567276,159167276)
map.pos[ 8 ,] <-  c(64984,146364984)
map.pos[ 9 ,] <-  c(88894,141088894)
map.pos[ 10 ,] <-  c(26070,135526070)
map.pos[ 11 ,] <-  c(102856,135002856)
map.pos[ 12 ,] <-  c(91619,133841619)
map.pos[ 13 ,] <-  c(19198564,115148564)
```

```
map.pos[ 14 ,] <-  c(20326742,105826742)
map.pos[ 15 ,] <-  c(22684095,102484095)
map.pos[ 16 ,] <-  c(1263,90201263)
map.pos[ 17 ,] <-  c(1,81100000)
map.pos[ 18 ,] <-  c(12535,78062535)
map.pos[ 19 ,] <-  c(160912,59160912)
map.pos[ 20 ,] <-  c(1,6.3e+07)
map.pos[ 21 ,] <-  c(15107860,48157860)
map.pos[ 22 ,] <-  c(17096300,51246300)



NCHR <- length(NSNP)
cv.info <- c()
cv2.info <- c()

for (ichr in 1:NCHR)
{
   print("----------------------------------------")
   nsnp_chr <- NSNP[ichr]
   p <- runif(nsnp_chr,min=.05,max=.95)
   hap <- matrix(0, nrow=nsnp_chr, ncol=2*NIND)
   for (isnp in 1:nsnp_chr)
   {
      hap[isnp,] <- rbinom(2*NIND,1,p[isnp])
   }
   # creating ref.hap file
   write.table(hap, file=paste("ref.chr",ichr,".hap",sep=''), quote=FALSE,row.names=FALSE, col.names=
     FALSE)
   # creating ref.legend file
   legend.id <- paste("rs",1:nsnp_chr,sep='')
   legend.pos <- sort(sample(map.pos[ichr,1]:map.pos[ichr,2], nsnp_chr, replace = FALSE))
   legend.al0 <- rep("C", nsnp_chr)
   legend.al1 <- rep("T", nsnp_chr)
   write.table(cbind(legend.id,legend.pos,legend.al0,legend.al1), file=paste("ref.chr",ichr,".legend",sep=''),
     quote=FALSE,row.names=FALSE, col.names=c("id","pos","al0","al1"))
   # creating ref.indv file
   write.table(1:NIND, file=paste("ref.chr",ichr,".indv",sep=''), quote=FALSE,row.names=FALSE, col.names
     =FALSE)
   ####
   # creating cv.hap file
   ncv_chr <- NCV[ichr]
   # cv set 1
   cv_chr <- sort(sample(1:nsnp_chr, ncv_chr, replace = FALSE))
   cvs <- hap[cv_chr,]
   write.table(cvs, file=paste("cv.chr",ichr,".hap",sep=''), quote=FALSE,row.names=FALSE, col.names=
     FALSE)
   cv.pos <- legend.pos[cv_chr]
   cv.maf <- apply(matrix(c(cvs),nrow=2*NIND),2,mean)
   cv.maf[which(cv.maf>.5)] <- 1-cv.maf[which(cv.maf>.5)]
   cv.a <- rnorm(ncv_chr,mean=0,sd=sqrt(VAR.A))
   cv.d <- rnorm(ncv_chr,mean=0,sd=sqrt(VAR.D))
   cv.info.chr <- cbind(ichr,cv.pos,cv.maf,cv.a,cv.d)
   cv.info <- rbind(cv.info,cv.info.chr)
```

```
    # cv set 2
    cv2_chr <- sort(sample(1:nsnp_chr, ncv_chr, replace = FALSE))
    cvs2 <- hap[cv2_chr,]
    write.table(cvs2, file=paste("cv2.chr",ichr,".hap",sep=''), quote=FALSE,row.names=FALSE, col.names=
      FALSE)
    cv2.pos <- legend.pos[cv2_chr]
    cv2.maf <- apply(matrix(c(cvs2),nrow=2*NIND),2,mean)
    cv2.maf[which(cv2.maf>.5)] <- 1-cv2.maf[which(cv2.maf>.5)]
    cv2.a <- rnorm(ncv_chr,mean=0,sd=sqrt(VAR.A2)) # var can be a function of pq
    cv2.d <- rnorm(ncv_chr,mean=0,sd=sqrt(VAR.D2)) # var can be a function of (pq)^2
    cv2.info.chr <- cbind(ichr,cv2.pos,cv2.maf,cv2.a,cv2.d)
    cv2.info <- rbind(cv2.info,cv2.info.chr)
}

colnames(cv.info) <- c("chr","pos","maf","a","d")
write.table(cv.info[,c("chr","pos","a","d")], file=paste("cv.info",sep=''), quote=FALSE,row.names=FALSE,
    col.names=TRUE)

colnames(cv2.info) <- c("chr","pos","maf","a","d")
write.table(cv2.info[,c("chr","pos","a","d")], file=paste("cv2.info",sep=''), quote=FALSE,row.names=FALSE,
    col.names=TRUE)

########### --file_cvs
b <- paste("cv.chr",INCLUDE.CHRS,".hap",sep='')
b <- cbind(INCLUDE.CHRS,b)
write.table(b,file=paste("par.pop1.cv_hap_files.txt",sep=''),quote=FALSE,row.names=FALSE,col.names=
    FALSE)

b <- paste("cv2.chr",INCLUDE.CHRS,".hap",sep='')
b <- cbind(INCLUDE.CHRS,b)
write.table(b,file=paste("par.pop1.cv2_hap_files.txt",sep=''),quote=FALSE,row.names=FALSE,col.names=
    FALSE)


########### --file_hap_name
a1 <- paste("ref.chr",INCLUDE.CHRS,".hap",sep="")
a2 <- paste("ref.chr",INCLUDE.CHRS,".legend",sep="")
a3 <- paste("ref.chr",INCLUDE.CHRS,".indv",sep="")
a <- cbind(INCLUDE.CHRS,a1,a2,a3)
write.table(a,file=paste("par.pop1.hap_sample_address.txt",sep=''),quote=FALSE,row.names=FALSE,col.
    names=c("chr","hap","legend","sample"))
```

create.genotypes.R

To create population information file, you can also use the following code:

```
# creating population info

#############################################################
#DEFINE WILDCARDS
#Must run this section. These need to be changed as you see fit
#############################################################

NUM.GENERATIONS <- 10    #number of generations
POP.SIZE <- rep(3000,NUM.GENERATIONS)  #population size over time
```

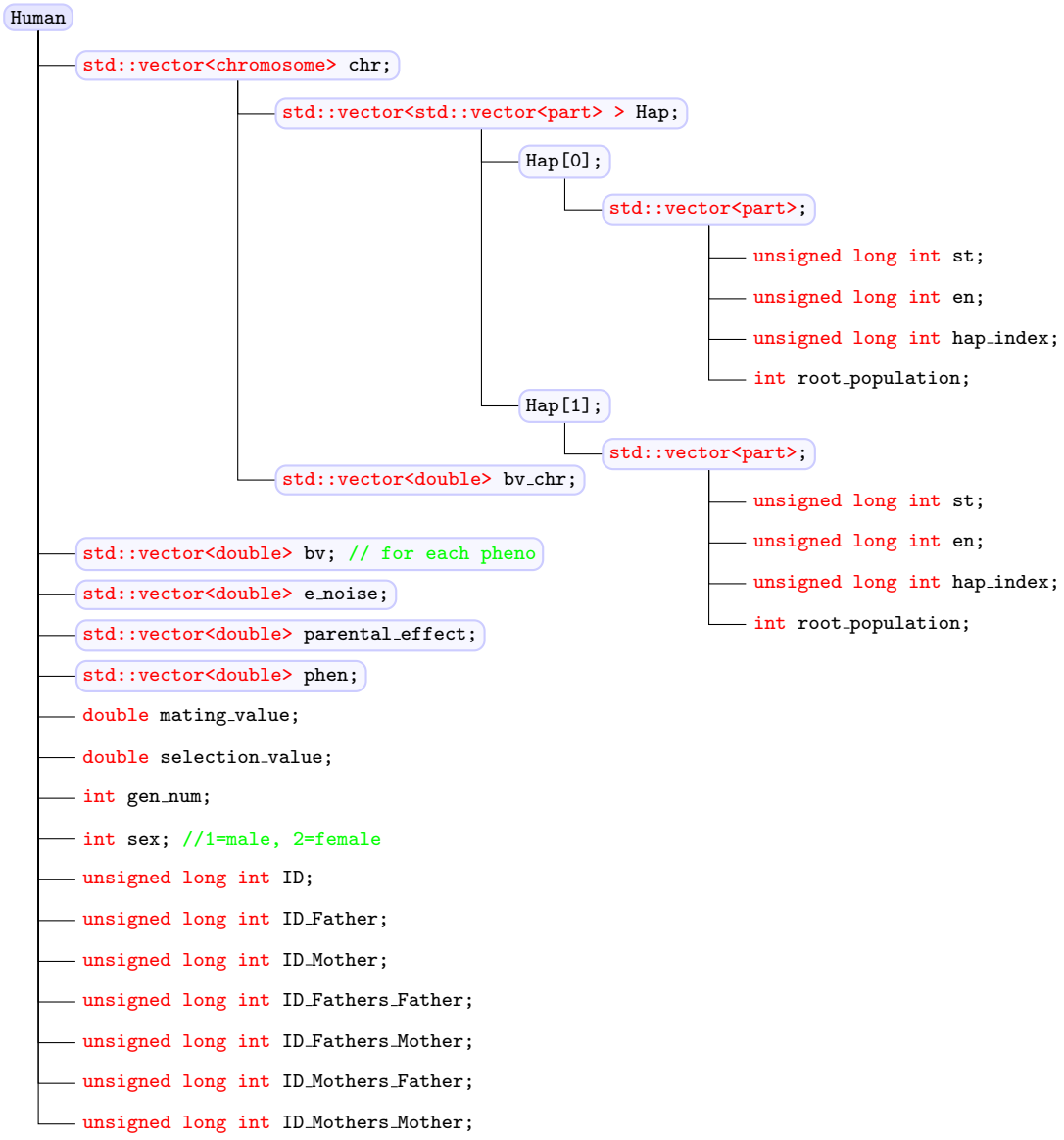
```
PHENO.MATE.COR <- rep(0,NUM.GENERATIONS)   #phenotypic correlation between mates at each
    generation
OFFSPRING_DIST <- rep("p",NUM.GENERATIONS) # p or P=Poisson distribution, f or F=fixed
    distribution
#OFFSPRING_DIST[NUM.GENERATIONS] <- "f" # last generation is fixed
SELECTION.FUNCTION <- rep("thr",NUM.GENERATIONS) #Selection function for each generation
SELECTION.FUNCTION.PAR1 <- rep(1,NUM.GENERATIONS) #Selection function for each generation
SELECTION.FUNCTION.PAR2 <- rep(1,NUM.GENERATIONS) #Selection function for each generation

##########popinfo.txt
write.table(cbind(POP.SIZE,PHENO.MATE.COR,OFFSPRING_DIST,SELECTION.FUNCTION,
    SELECTION.FUNCTION.PAR1,SELECTION.FUNCTION.PAR2), file=paste("ex1.popinfo.txt",sep=''),
    quote=FALSE, row.names=FALSE, col.names=c("pop_size","mat_cor","offspring_dist","selection_func", "
    selection_func_par1","selection_func_par2"))
```

An explanatory example

Appendix B

C++ Classes

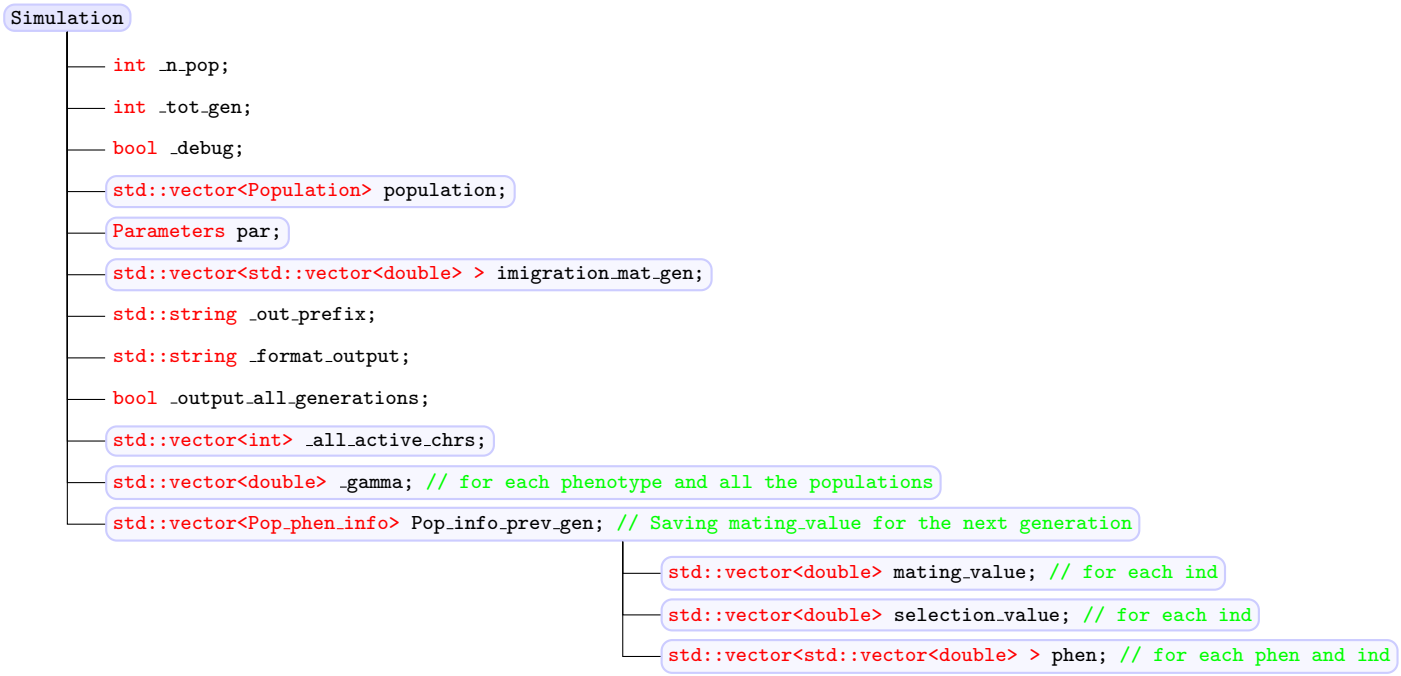
B.1 Human



B.2 Population

```
Population
    int _pop_num;
    int _nchr;
    std::vector<Phenotype_scheme> _pheno_scheme; // for each phenotype
        std::vector<CV_INFO> _cv_info; // for each chr
            std::vector<unsigned long int> bp;
            std::vector<double> maf;
            std::vector<double> alpha;
        std::vector<CV> _cvs; // for each chr
            std::vector<std::vector<bool> > val;
        std::vector<std::string> _name_cv_hap; // for each chr
        double _va;
        double _ve;
        double _vf;
        double _alpha; // mating value coefficient
        double _beta; // transmission of environmental effects from parents to offspring
        double _delta; // selection value coefficient
    std::vector<Human> h;
    std::vector<Couples_Info> _couples_info;
        unsigned long int pos_male; // pos human, not pos hap
        unsigned long int pos_female; // pos human, not pos hap
        bool inbreed;
        int num_offspring;

    std::vector<unsigned long int> _pop_size; // for each generation
    std::vector<double> _mat_cor; // for each generation
    std::vector<std::string> _offspring_dist; // for each generation
    std::string _selection_func; // --logit 0 1
    std::vector<double> _selection_func_pars; // --logit 0 1
    std::vector<std::vector<std::string> > _hap_legend_sample_name; // for each chr, with 3 columns
    std::vector<rMap> _rmap; // for each chr
    std::vector<std::vector<double> > _recom_prob; // for each chr
    std::vector<double> _var_bv_gen0; // for each phenotype
    std::vector<int> _all_active_chrs; // for each chr
    bool _avoid_inbreeding;
    bool _no_output;
    bool _output_all_generations;
    bool _debug;
    std::string _out_prefix;
    std::string _format_output;
    double _RM_percent; // Random mating percent (inds who have 2 spouses)
    std::vector<double> ret_var_mating_value; // for each gen
    std::vector<std::vector<double> > ret_var_phen; // for each pheno and gen
    std::vector<std::vector<double> > ret_var_bv; // for each pheno and gen
    std::vector<std::vector<double> > ret_var_parental_effect; // for each pheno and gen
```

B.3 Simulation



Appendix C

File Formats

C.1 Hap – Legend – Sample

C.1.1 .hap

No header.

This file is SPACE delimited. Each line corresponds to a single SNP. Each successive column pair (0, 1), (2, 3), (4, 5) and (6, 7) corresponds to the alleles carried at the 4 SNPs by each haplotype of a single individual. For example a pair "1 0" means that the first haplotype carries the B allele while the second carries the A allele as specified in the LEGEND file. The haplotypes are given in the same order than in the SAMPLE file. This file should have L lines and 2N columns, where L and N are the numbers of SNPs and individuals respectively.

	ind1.hap1	ind1.hap2	ind2.hap1	ind2.hap2	...	
snp1	0	0	1	0		
snp2	0	1	1	0		
⋮						
						nsnp × nhaps

C.1.2 .legend

Has header.

This file is SPACE delimited. The first line is a header line that describe the content of the file. Each line corresponds to a single SNP.

	col1	col2	col3	col4	
header	ID	pos	allele0	allele1	
snp1	rs17432784	17196300	T	C	
snp2	rs2845379	rs1807512	A	G	
snp3	rs17432784	17196300	G	T	
⋮					
					(nsnp+1) × 4

C.1.3 .sample

Has header.

It is SPACE delimited. The first line is a header line that describe the content of the file. Then, each line corresponds to a single individual.

	col1	col2	col3	col4	
header	sample	population	group	sex	
ind1	CEU1	CEU	EUR	1	
ind2	CEU2	CEU	EUR	2	
ind3	GBR1	GBR	EUR	2	
⋮					
					(nind+1) × 4

C.1.4 .impute.hap.indv

No header.

This file has just one column.

	col1	
ind1	5659883013-R02C01	
ind2	5648551130-R02C01	
ind3	5648560075-R02C01	
⋮		
		nind × 1

C.2 PLINK

There are two formats for PLINK: Binary format (.bed, .bim and .fam) or uncompressed format (.ped and .map).

C.2.1 .ped

No header.

Each line corresponds to a single individual.

1. Family ID
2. Sample ID
3. Paternal ID
4. Maternal ID
5. Sex (1=male; 2=female; other=unknown)
6. Affection (0=unknown; 1=unaffected; 2=affected)
7. Genotypes (space or tab separated, 2 for each marker. 0=missing)

	FID	IID	PID	MID	Sex	Aff	SNP1	SNP1	SNP2	SNP2	
IND1	fam1	ind1	0	0	1	2	A	C	C	T	
IND2	fam1	ind2	0	0	2	1	C	A	T	T	
IND2	fam2	ind1	0	0	1	1	C	C	C	T	
⋮											
											(nind) × (6+2*nsnp)

C.2.2 .map

No header.

It is SPACE delimited. Each line corresponds to a single SNP. Chromosome can be (1-22, X, Y or 0 if unplaced)

	chromosome	rs#	cM	bp	
snp1	1	rs123456	0	1234555	
snp2	1	rs234567	0	1237793	
snp3	1	rs233556	0	1337456	
⋮					
					(nsnp) × 4

C.2.3 .fam

No header.

It is SPACE delimited. Each line corresponds to a single individual.

1. Family ID ('FID')
2. Within-family ID ('IID'; cannot be '0')
3. Within-family ID of father ('0' if father isn't in dataset)
4. Within-family ID of mother ('0' if mother isn't in dataset)
5. Sex code ('1' = male, '2' = female, '0' = unknown)
6. Phenotype value ('1' = control, '2' = case, '-9'/'0'/non-numeric = missing data if case/control)

If there are any numeric phenotype values other than -9, 0, 1, 2, the phenotype is interpreted as a quantitative trait instead of case/control status. In this case, -9 normally still designates a missing phenotype;

	FID	IID	ID-Father	ID-mother	Sex	Phenotype	
ind1	1	child1	0	0	1	1	
ind2	1	child2	0	0	1	2	
ind3	2	child1	0	0	1	2	
⋮							
							(nind) × 6

C.3 Interval

Has header.

Each line corresponds to a single part of a haplotype. The columns are:

1. Human ID

2. Chromosome number
3. Haplotype (0 or 1)
4. Start position (bp)
5. End position (bp)
6. Index position of the haplotype in the initial generation (generation 0)
7. Root population index (zero based index)

In the following example, the first haplotype (haplotype 0) for chromosome 1 of an individual with ID 1, is saved as a continues union of intervals, where each interval comes from one ancestral root. The id of the ancestral root is given in the sixth column.

```
h_ID chr hap st en hap_index root_pop
1 1 0 738555 3981099 28927 1
1 1 0 3981099 4804034 16396 1
1 1 0 4804034 5046412 47008 1
1 1 0 5046412 5917830 19541 1
1 1 0 5917830 6241920 52672 1
1 1 0 6241920 7781935 1026 1
1 1 0 7781935 10744488 17586 1
1 1 0 10744488 14786766 15382 1
1 1 0 14786766 14932265 13305 1
1 1 0 14932265 30071798 62082 1
1 1 0 30071798 30565233 34178 1
1 1 0 30565233 35873811 15383 1
1 1 0 35873811 37925508 22152 1
1 1 0 37925508 41381101 14645 1
1 1 0 41381101 45089312 28882 1
1 1 0 45089312 57225089 14645 1
1 1 0 57225089 60927267 7988 1
1 1 0 60927267 65383367 41180 1
1 1 0 65383367 67645596 34866 1
1 1 0 67645596 94508106 49280 1
1 1 0 94508106 97900471 15372 1
1 1 0 97900471 101722348 22732 1
...
```

out1.pop1.gen20.chr1.int

Given a ".int" file as input, the program "SharedHaplotypes" available at
https://github.com/rtahmasbi/IBG-SharedHaplotypes
can extract the true shared haplotypes between each two individuals and can report the true identical by decent (IBD) haplotypes.

Bibliography

[1] Andy Boyd, Jean Golding, John Macleod, Debbie A Lawlor, Abigail Fraser, John Henderson, Lynn Molloy, Andy Ness, Susan Ring, and George Davey Smith. Cohort profile: the 'children of the 90s' – the index offspring of the avon longitudinal study of parents and children. *International journal of epidemiology*, 42(1):111–27, 2013.

[2] DS Falconer and TFC Mackay. *Introduction to Quantitative Genetics*. Longman, 4 edition, 1996.

[3] RA Fisher. The correlation between relatives on the supposition of Mendelian inheritance. *Transactions of the Royal Society of Edinburgh*, 52:399–433, 1918.

[4] Sean Hoban, Giorgio Bertorelle, and Oscar E Gaggiotti. Computer simulations: tools for population and evolutionary genetics. *Nature Reviews Genetics*, 13(2):110–122, 2012.

[5] John FC Kingman. The coalescent. *Stochastic processes and their applications*, 13(3):235–248, 1982.

[6] John FC Kingman. On the genealogy of large populations. *Journal of Applied Probability*, pages 27–43, 1982.

[7] Augustine Kong, Daniel F Gudbjartsson, Jesus Sainz, Gudrun M Jonsdottir, Sigurjon A Gudjonsson, Bjorgvin Richardsson, Sigrun Sigurdardottir, John Barnard, Bjorn Hallbeck, Gisli Masson, et al. A high-resolution recombination map of the human genome. *Nature genetics*, 31(3):241–247, 2002.

[8] Matthew D Mailman, Michael Feolo, Yumi Jin, Masato Kimura, Kimberly Tryka, Rinat Bagoutdinov, Luning Hao, Anne Kiang, Justin Paschall, Lon Phan, et al. The NCBI dbGaP database of genotypes and phenotypes. *Nature genetics*, 39(10):1181–1186, 2007.

[9] Philipp W Messer. SLiM: simulating evolution with selection and linkage. *Genetics*, 194(4):1037–1039, 2013.

[10] Alireza Moayyeri, Christopher J Hammond, Deborah J Hart, and Timothy D Spector. The UK adult twin registry (TwinsUK resource). *Twin Research and Human Genetics*, 16(01):144–149, 2013.

[11] Samuel Neuenschwander, Frédéric Guillaume, Jérôme Goudet, et al. quantiNemo: an individual-based program to simulate quantitative traits with explicit genetic architecture in a dynamic metapopulation. *Bioinformatics*, 24(13):1552–1553, 2008.

[12] Bo Peng and Marek Kimmel. simuPOP: a forward-time population genetics simulation environment. *Bioinformatics*, 21(18):3686–3687, 2005.

[13] Kimberly A Tryka, Luning Hao, Anne Sturcke, Yumi Jin, Zhen Y Wang, Lora Ziyabari, Moira Lee, Natalia Popova, Nataliya Sharopova, Masato Kimura, et al. NCBI's database of genotypes and phenotypes: dbGaP. *Nucleic acids research*, 42(D1):D975–D979, 2014.

[14] Zulma G Vitezica, Luis Varona, and Andres Legarra. On the additive and dominant variance and covariance of individuals within the genomic selection scope. *Genetics*, 195(4):1223–1230, 2013.