G-matrix Simulator -- Command Line Version

By Adam G. Jones

University of Idaho

Summary

The command-line version of the **G**-matrix simulator uses an individual-based model to simulate the evolution of the additive genetic variance-covariance matrix over hundreds or thousands of generations. The simulations are described at length elsewhere (see references below). The command-line version implements almost all of the features of the original Windows-based **G**-matrix simulation programs but without the graphical interface. Thus, the command-line version is suitable for use on any operating system with a C++ compiler and can be run on computers lacking graphical user interfaces (such as super computers or cloud instances). Users are also free to modify the source code as they see fit. The model is described in detail in the following publications:

Jones, A. G., S. J. Arnold, and R. Bürger. 2003. Stability of the **G**-matrix in a population experiencing pleiotropic mutation, stabilizing selection, and genetic drift. Evolution 57:1747-1760.

Jones, A. G., S. J. Arnold, and R. Bürger. 2004. Evolution and stability of the **G**-matrix on a landscape with a moving optimum. Evolution 58:1639-1654.

Jones, A. G., R. Bürger, S. J. Arnold, P. A. Hohenlohe, and J. C. Uyeda. 2012. The effects of stochastic and episodic movement of the optimum on the evolution of the **G**-matrix and the response of the mean to selection. Journal of Evolutionary Biology 25:2210-2231.

For a step-by-step guide on the development of this type of model in C++, see the following publication:

Jones, A. G. 2018. *C++ for Biologists: Evolutionary Models*. Independently published, Moscow, ID.

A hard-copy of this book can be purchased from Amazon, and a free pdf version is available for download from https://pipefishguysite.wordpress.com/jones-lab-publications/.

Installing the Simulator

You will need a standard C++ compiler to compile the source code into an executable for your operating system. The GNU g++ compiler is a good choice, and it should be included with most Linux distributions. On Linux, type g++ at the command line and hit “enter”. If g++ is not already installed, your operating system should tell you how to install it.

Make sure the source file (QGmodel.cpp) and the header files (MTwisterFunctions.h and simulation_engine.h) are in the current folder and type:

```
g++ QGmodel.cpp -o gmatrix_simulator
```

The compiler should create an executable named "gmatrix_simulator". To run this program, you might need to alter its permissions. Use:

```
chmod u+x gmatrix_simulator
```

The easiest way to run the program is to have a copy of it in your current directory (on Linux you would type `./gmatrix_simulator` to run it). Another option is to add its folder to your path (use Google to find out how) or to copy the compiled executable to a folder that is already in your path (often something like `/usr/local/bin`). We will assume the executable is in the current directory.

Running the Simulator

The simulator will generate a number of files in the current directory, so it will often be worth making a new directory for a series of runs. Then the simulator can be run easily by invoking the executable. In Linux, this task is accomplished by merely typing the name of the program:

```
./gmatrix_simulator
```

Invoking the program in this way will run it under the default parameter values. These parameter values will seldom be precisely what you want, so you will generally want to supply the simulator with arguments. The parameters and how to set them are shown in Table 1.

Table 1. Parameter values and the command-line argument used to change them. Default values and feasible ranges are also shown. In some cases, exceeding the feasible range may cause the program to crash (e.g., it might not allocate enough memory to handle the parameter value). In other cases, values outside the feasible range might cause unusual behavior or very slow run times. The final column provides a verbal description of the meaning of the parameter. In all cases, the parameters are best understood by consulting the Jones et al. (2003, 2004, 2012) papers.

Parameter	Argument	Default Value	Feasible Range	Description
Filename	--filename	outfile	Any admissible filename	This core filename will appear as part of every output file. Use it to identify the files associated with a specific run of the model.
No_Generations	--gens	1000	1-100,000	This parameter sets the number of experimental

				generations, during which summary statistics are recorded.
Carrying_Capacity	--carry_cap	256	50-5000	The number of adults in the population will be culled so that the adult population size does not exceed this value.
Female_Fecundity	--fecund	4	3-20	The number of progeny produced per female.
Max_Mating_Enc.	--mate_enc	50	1-500	The number of males a female will sample before she gives up and doesn't mate. If mating is random, she mates with the first male she encounters, so this parameter only matters when --pref_var is set to a value other than zero.
Gaussian_Pref_Var	--pref_var	0	0-200	This value sets the strength of Gaussian mating preferences by females. It is akin to a variance, so larger values produce weaker mate choice. Females choose on the basis of the value of trait 0 in encountered males. A value of 0 for this parameter results in random mating. When Gaussian preferences are invoked, a female's value for trait 1 determines her preferred male phenotype. That is, trait 0 is the male ornament and trait 1 is the female preference.
No_Loci_Trait0	--loci_trt0	0	0-50	Number of non-pleiotropic loci affecting trait 0.
No_Loci_Trait1	--loci_trt1	0	0-50	Number of non-pleiotropic loci affecting trait 1.
No_Loci_Pleiotrop	--loci_pleio	50	0-50	Number of pleiotropic loci affecting both traits.
Env_Variance_Trt0	--env_var0	1	0-10	Amount of environmental variance for trait 0.
Env_Variance_Trt1	--env_var1	1	0-10	Amount of environmental variance for trait 1.

Mut_Var_Trait0	--mut_var0	0.05	0-2	The variance of the distribution from which new mutational effects are drawn for loci affecting trait 0.
Mut_Var_Trait1	--mut_var1	0.05	0-2	The variance of the distribution from which new mutational effects are drawn for loci affecting trait 1.
Mut_Correlation	--mut_corr	0	-0.9-0.9	The correlation coefficient for the bivariate normal distribution from which mutations at pleiotropic loci are drawn.
Mutation_Rate	--mut_rate	0.0002	0-0.01	The per-locus mutation rate.
Omega_Trait0	--exp_w00	49	2-200	The width of the individual selection surface for trait 0 during the experimental generations. Larger numbers result in weaker stabilizing selection.
Omega_Trait1	--exp_w11	49	2-200	The width of the individual selection surface for trait 1 during the experimental generations.
Selection_Corr	--exp_sel_corr	0	-0.9-0.9	The strength of correlational selection during the experimental generations.
Optimum_Trait0	--exp_opt0	0	-10-10	The location of the trait 0 optimum at the beginning of the experimental generations.
Optimum_Trait1	--exp_opt1	0	-10-10	The location of the trait 1 optimum at the beginning of the experimental generations.
Sex_Limited_Sel	--exp_sex_lim	false	true or false	When false, selection acts on both traits in both sexes. When true, trait 0 is selected only in males and trait 1 is selected only in females.
No_Initial_Gens	--init_gens	1000	1-20,000	The number of initial generations of stabilizing selection, during which the population reaches a mutation-selection-drift

				quasi-equilibrium. No data are collected during these generations.
Init_Omega_Trait0	--init_w00	49	2-200	The strength of selection on trait 0 during the initial generations.
Init_Omega_Trait1	--init_w11	49	2-200	The strength of selection on trait 1 during the initial generations.
Init_Sel_Corr	--init_sel_corr	0	-0.9-0.9	The strength of correlation selection during the initial generations.
Init_Opt_Trt0	--init_opt0	0	-10-10	The position of the trait 0 optimum during the initial generations.
Init_Opt_Trt1	--init_opt1	0	-10-10	The position of the trait 1 optimum during the initial generations.
Init_Sex_Lim_Sel	--init_sex_lim	false	true or false	Determines whether or not selection is sex-limited during the initial generations (see above).
No_Initial_Gens_Moving_Opt	--init_gens_directional_sel	0	0-5,000	Another optional set of initial generations during which the optimum is allowed to move. This parameter allows the population to equilibrate relative to the moving optimum before data are collected.
Peak_Shift_Interval	--peak_shift_interval	1	1-500	The number of generations that elapse between each movement of the optimum.
Trait0_Peak_Shift	--trt0_peak_shift	0	0-2	The amount the trait 0 optimum moves each time it shifts.
Trait1_Peak_Shift	--trt1_peak_shift	0	0-2	The amount the trait 1 optimum moves each time it shifts.
Trait0_Optimum_Var	--opt0_variance	0	0-0.5	The variance of a Gaussian random number that will be added to the trait 0 optimum each time it moves.
Trait1_Optimum_Var	--opt1_variance	0	0-0.5	The variance of a Gaussian random number that will be added to the trait 1 optimum each time it moves.

Move_Peak_Repeatedly	--move_peak_repeatedly	false	true or false	This parameter determines whether the peak moves only once (at the generation specified by Peak_Shift_Interval) or multiple times (with the interval between moves given by Peak_Shift_Interval).
Replications	--reps	5	1-50	The total number of replicate simulations to run under this set of parameter values.

Using Shell Scripts to Run the Program

With so many parameters to set, it is nearly hopeless to run the program by typing a command into the command line. The best way to run it is to use a shell script. Here, I provide an example of the format of a BASH shell script that will run the program for a single set of parameters. Shell scripts to replicate the runs that provided data for Table 1 in Jones et al. (2003) and Tables 1 and 2 in Jones et al. (2004) are available in the GitHub repository associated with this program.

```
#!/bin/bash

./gmatrix_simulator \
--filename jones_table1_row1 \
--reps 20 \
--init_gens 10000 \
--init_w00 9 \
--init_w11 9 \
--init_sel_corr 0 \
--init_opt0 0 \
--init_opt1 0 \
--init_sex_lim false \
--gens 2000 \
--carry_cap 256 \
--fecund 4 \
--mate_enc 50 \
--pref_var 0 \
--loci_trt0 0 \
--loci_trt1 0 \
--loci_pleio 50 \
--env_var0 1 \
--env_var1 1 \
--mut_var0 0.05 \
--mut_var1 0.05 \
--mut_corr 0 \
--mut_rate 0.0002 \
--exp_w00 9 \
--exp_w11 9 \
--exp_sel_corr 0 \
--exp_opt0 0 \
--exp_opt1 0 \
--exp_sex_lim false \
--move_peak_repeatedly false \
--trt0_peak_shift 0 \
```

```
--trt1_peak_shift 0 \
--peak_shift_interval 1 \
--init_gens_directional_sel 0 \
--opt0_variance 0 \
--opt1_variance 0
```

A text file containing the lines shown above can be executed as a BASH script in Linux. To make the script executable, use `chmod u+x my_script.sh`, where "my_script.sh" is the script's filename.

Visualizing the G-matrix in R

The GitHub repository for the **G**-matrix command-line program also contains an R-script that can be used to visualize the **G**-matrix over the course of a simulation run. To use it, install the package known as "shape" [install.packages("shape")]. Then load the R script (Gmatrix_Plotter_R.R) into R. Make sure the **G**-matrix output file (something like "outfile_run_1.csv") is in your R working directory. Edit the script, so that your output filename is used by this line of the script:

```
data <- read.csv("jones_2004_table1_row3_run_1.csv")
```

The other parameters you can change in the script include "draw_width", which specifies how wide the drawing area will be. Change this value so that the output fits on your screen. You can also change the "draw_interval", which determines how many generations to skip between renderings of the **G**-matrix. Finally, you can change "sleep_time", which controls how long R will pause on each **G**-matrix. If you set "sleep_time" to a value less than about 0.2, you will experience undesirable graphical glitches.

Then, to run the script, use the command (within R):

```
source("Gmatrix_Plotter_R.R")
```