

Fedorov Lab

User Guide for MAGEr1.pl

A software for population genetics simulations

This is the release 1 (r1) of MAGE. Next update (r2) will be in March 31, 2014

Available at: <http://bpg.utoledo.edu/~afedorov/lab/mage.html>

Alexei Fedorov's Laboratory
Department of Medicine, HSC
The University of Toledo
3000 Arlington Avenue
Toledo, OH 43614-5809
Phone: (419)383-5270
FAX: (419)383-3102

E-mail Address: alexei.fedorov@utoledo.edu

11/25/2013

MENU INDEX

1. [Abstract & Introductions](#)
2. [How does MAGE work? \(Flowchart & explanation\)](#)
 - [Genomes and individuals](#)
 - [Mutations](#)
 - [Meiotic recombinations and gametogenesis](#)
 - [Computation of a new generation of virtual individuals](#)
 - [Selection](#)
3. [Algorithm Details in MAGEr1.pl](#)
4. [Speed in MAGEr1.pl](#)
5. [Backup file in MAGEr1.pl](#)
6. [Mating Schemes](#)
7. [Current Versions of MAGEr1.pl](#)
8. [Step by Step illustration of MAGEr1.pl](#)
9. [Resources](#)
10. [References](#)
11. [Appendix](#)



Abstract

Mammalian genomes are replete with millions of polymorphic sites, among which those genetic variants that are co-located on the same chromosome and exist close to one another form blocks of linked mutations known as haplotypes. The linkage disequilibrium between polymorphic sites within haplotypes is periodically disrupted due to meiotic recombination events that cause an appearance of new haplotypes. Whole ensembles of such numerous haplotypes are subjected to evolutionary pressure, where mutations influence each other and should be considered as a whole entity – a gigantic matrix, unique for each individual genome. This idea was implemented into a computational approach, named Matrix Algorithms for Genome Evolution (MAGE) to model genomic changes taking into account all mutations in a population. MAGE has been tested for modeling of entire human chromosomes. The program can precisely mimic real biological processes that have an influence on genome evolution such as: 1) authentic arrangements of genes and functional genomics elements; 2) frequencies of various types of mutations in different nucleotide contexts; 3) non-random distribution of meiotic recombination events along chromosomes. Computer modeling with MAGE has demonstrated that the number of meiotic recombination events per gamete is among the most crucial factors influencing population fitness. In humans, these recombinations create a gamete genome consisting on an average of 48 pieces of corresponding maternal and paternal chromosomes. Such highly mosaic gamete structure allows preserving fitness of population under the intense influx of novel mutations (50-100 per individual) even when the number of mutations with deleterious effects is several times more abundant than those with beneficial effects.

Introduction

Matrix Algorithms for Genome Evolution (MAGE) presents a computer-simulation approach and possible solutions for the decades-long controversies in population genetics, where several opposing theories (e.g. Selectionism and Neutralism) modeling intricate dynamics of mutations have not yet reached a consensus. We



bring forth new software named MAGEr1.pl and MAGEr01.java that perform large-scale computational simulations of genome evolution for complex organisms under intense influx of mutations existing in reality. These advanced program packages allow consideration of multiple population parameters that have not yet been examined together. We acknowledge the existence of a number of similar computational algorithms created in the past, which are described in the Appendix. MAGE belongs to the forward simulator class of these algorithms. Our long-term strategic goal is to recreate the complex dynamics of mutations in the entire human genome taking into account all parameters as close to reality as possible. This includes consideration of non-random distribution of mutations with their real frequencies, complex arrangement of genes along chromosomes, considerable inhomogeneity of nucleotide compositions in different genomic sequences, hot and cold spots for meiotic recombination events, variations in the contributions of maternal and paternal alleles on phenotypes, etc. A combination of these goals have not been attacked by previous computer algorithms, therefore, we designed our own MAGE package. Our policy is to make all computer codes freely available to public and provide support from our web pages. We appreciate a broad collaboration on MAGE project with world-wide scientific community. Key biological questions, which we try to answer using MAGE modeling, are the following:

- What parameters are most important for maintaining the fitness of population under the intense influx of novel mutations when deleterious ones overwhelm beneficial?
- What biological forces have created GC-rich and GC-poor isochores in mammals and birds? (Testing Bias Gene Conversion theory.)
- What biological forces have shaped regularities in codon usage bias within vertebrates? (Testing our hypothesis from Nabiyouni et al. 2013, *Gene* 519:113-119).
- Why an average human gene has 3-7 mutually exclusive SNP haplogroups?



- How Mid-Range Inhomogeneity genomic regions have been created in animal genomes? (Reviewed by Fedorova and Fedorov 2011, *The Scientific World Journal: Genes & Genomics*. 2011, 11:842-854.)

[--return to MENU--](#)



How does MAGEr1.pl work?

The simplest scheme of MAGE is demonstrated in the *Figure 1* and its major steps are outlined below.

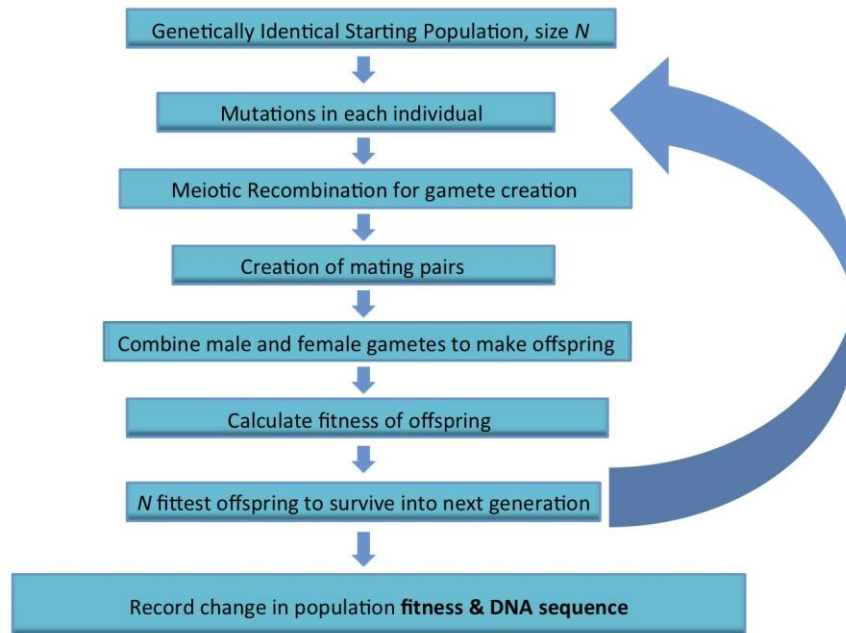


Figure 1 MAGE begins with a genetically identical population of size N . Genomic mutations occur in each individual, which are passed onto offspring. According to the mutations inherited, fitness is calculated for each offspring. The N fittest offspring become the next generation and the process repeats for thousands of generations.

A) **Genomes and individuals.** A large portion of a real genomic sequence (even whole chromosomes of human or other species) can be assigned as a reference genome for a model population. A user specifies the number of virtual individuals in the population (N). Each virtual individual is constructed as a diploid genome that descended as two haploid gamete genomes from its parents. For simplicity, in default initial conditions, all virtual individuals at the beginning share the same reference genome. In other words, there are no genetic variants between them (it is challenging for many users to assign thousands of initial SNPs, so this approach resolves this problem). As the program is running, mutations and recombinations

create diversity in the population. This is recorded by each virtual individual's genome being presented as an array of SNPs that make it different from the reference genome.

B) **Mutations.** Taking a user-defined integer parameter μ (number of novel mutations per gamete) MAGE creates mutations in the genomic sequences using random number generator for choosing mutation positions. Upon generation of a mutation, MAGE assigns a selection coefficient (s parameter) to the mutation using a user-defined s -distribution. We use non-normalized s -values, which means that s may be any number. Our algorithm is specifically design to model thousands of SNPs in each virtual individual. Thus a particular s -value for a single mutation does not matter, but should only make sense in comparison with other mutations. Distribution of all mutations of a virtual individual by their s -values determines the fitness of this individual. We use two methods to generate mutations (saturated and unsaturated algorithms are described in the next section → [Quick Link](#)).

C) **Meiotic recombinations and gametogenesis.** Haploid genomes of gametes are generated for each virtual individual from its parents' chromosomes. The number of meiotic recombinations between its parents' chromosomes is an input parameter (r). Parameter r should be an integer or a real number in the (0..1) range. If $r < 1$ (let's assume it is 0.33) then two thirds of gametes do not have recombination between maternal and paternal chromosomes and one third of the gametes have one recombination event each. The recombination sites are defined by a random number generator, which might take into account the "hot-spots" and "cold-spots" for recombination events from the International HapMap Consortium genetic maps (specific option currently available for MAGEr01.java). According to the HapMap database of meiotic recombination events in humans, the total genetic length of all human autosomes is 35.25 morgans (Frazer et al. 2007, Nature 449: 851-861). This length does not include the partition of the genome on 23 chromosome pairs. For a computer modeling of genome evolution, all chromosomes may be spliced into a sole genomic sequence with the assignment to each of the chromosomal junctions a recombination hotspot with



the 50 centimorgan value. In this algorithmic procedure of chromosome concatenation, the order of chromosomes is not important and might be chosen randomly. The partition of a genome on N pairs of chromosomes increases the mosaic structure of gametes on an average by $N/2$ pieces of maternal and paternal DNA segments. Therefore, on an average, a human gamete is composed from $35.25 + 23/2 + xy \approx 48$ maternal and paternal chromosomal segments (where xy represents the average number of recombination events between two X chromosomes in females and recombination events between pseudoautosomal regions of X and Y chromosomes in males). Therefore, we frequently use $r=48$ in our MAGE simulations. However, it is important to note that fixed recombination sites (chromosomal junctions) and unfixed meiotic recombination sites might have different effects on mutational dynamics under specific parameters.

D) Generation of offspring for virtual individuals. By default and implicitly, 50% of randomly chosen virtual individuals are females and the rest are males. Different mating schemes for virtual individuals are possible as input options in the next release (r2) of MAGEr1.pl ([Quick Link](#)). In release 1 we use only one default scheme: random and permanent pairings between male and female virtual individuals. Their offspring have diploid genomes formed by two randomly chosen parents' gametes. The number of offspring per individual (α) is a user-defined input integer parameter and is the same for each pair.

E) Natural selection. The overall fitness of every created virtual individual in the MAGE algorithm is computed by taking into account all the mutations possessed by this individual. The current version of MAGE does not take into account mutual effects of mutations such as compensatory mutations and epistasis. MAGE calculates fitness for each gene by summing all the s -values of mutations within that gene. For example, assume that for a gene, its maternal allele is composed of a particular haplotype containing x number of SNPs and its paternal allele is composed of a different haplotype comprising y number of SNPs. The fitness of the maternal allele for the given gene (w_m) will be a sum of s -values for the entire set of x SNPs within this gene, while the fitness of the paternal allele (w_p) will be a sum of s -values for all y SNPs. The fitness of the gene in this example is calculated

from the w_m and w_p values and another user-defined input parameter, the dominance coefficient (h) using the formula: $w = \min(w_m, w_p) + h * \text{abs}(w_m - w_p)$. Since in reality an average mammalian gene has hundreds of genetic variants, our modeling parameter h is specific to a gene and not to a particular polymorphic site. In a co-dominance mode ($h=0.5$), the gene fitness is the average of the fitness of maternal and paternal alleles. For a recessive mode ($h=1$) the fitness is the maximum between w_m and w_p values (heterozygotes with one highly deleterious allele of a recessive gene are healthy), while for a dominant mode ($h=0$) the gene fitness is the minimum between w_m and w_p values (heterozygotes with one highly deleterious allele of a dominant gene are not healthy). Finally the overall fitness of the virtual individual is the sum of fitness of all genes inside the genome. In the selection phase of MAGE algorithm, the program picks the N fittest offspring and forms from them the new generation. This new generation replaces the previous one and the entire cycle repeats for a given number of generations.

MAGE regularly outputs the following parameters: Current generation, total fitness of the population, number of SNPs, number of fixed mutations (Fs). In addition, MAGE regularly outputs backup files that contain the current reference genome and arrays of all SNPs for each virtual individual. The latter arrays may be used to investigate mutation dynamics in the population with the highest precision.

[--return to MENU--](#)



Algorithm Details in MAGEr1.pl

A user chooses any curve for a distribution of mutations by their s-values, as demonstrated on the *Figure 2*. This curve should be normalized and tabulated in order to use it as an input table for MAGE algorithm. *Table 1* exemplifies such an input table obtained from the *Figure 2A*. The last column in this table represents the relative frequency of mutations with selection coefficient in the range ($s_{i-1} ..s_i$). The total sum of all these relative frequencies must be equal to 1. We have an auxiliary program to generate such table using Microsoft Excel ([Quick Link](#)).

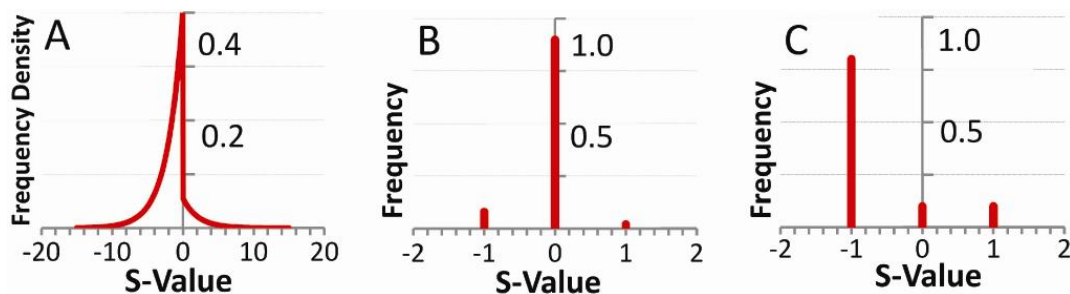


Figure 2. Distribution of mutations by their selection coefficients (s-values). A - Represents a continuous distribution of mutations by s that can range from -20 to +20 depending on their deleterious (negative s-values) or beneficial (positive s-values) effects. This curve represents 88% deleterious and 12% beneficial mutations. B - models a discrete distribution of mutations characterized predominantly by neutral mutations occurring at a frequency of 90% within the population while the remaining 10% is characterized by deleterious and beneficial mutations occurring in a ratio of 9:1. C - illustrates another discrete distribution for mutations, where the ratio of deleterious to beneficial mutations occurs again in the ratio of 9:1. However, this model is characterized by a preponderance of mutations with deleterious effects (81%). Neutral mutations in this case comprise 10% and beneficial - 9% of overall nucleotide changes occurring within the population.



S-Value	Frequency Density
-15	0.000405661
-14.9625	0.000412736
-14.9249	0.000419935
-14.8874	0.000427259
-14.8498	0.000434711
-14.8123	0.000442293
-14.7747	0.000450007
-14.7372	0.000457856
-14.6996	0.000465841
-14.6621	0.000473966
-14.6245	0.000482233
-14.587	0.000490644
-14.5494	0.000499201
-14.5119	0.000507908
-14.4743	0.000516766
-14.4368	0.00052578
-14.3992	0.00053495
-14.3617	0.00054428

Table 1 An example of the segment of selection coefficient table from the distribution curve *Figure 2A*. This table specified a group of discrete values (the s selection coefficient) and their corresponding frequency density.

The existence of millions of genetic variations between individual organisms means that it is absolutely impossible to experimentally measure the selection coefficient (s) for a given mutation, unless this mutation is highly deleterious or beneficial. Theoretically, the exact value of s for a mutation should present the rate of advantage/disadvantage of this mutation over a wild type (HARTL and CLARK 2007). However, since s is an immeasurable parameter, MAGE algorithm assigns **non-normalized values of s** for generated mutations. For simplicity, MAGE may assign integer values of s (e. g. -10; -9; -8, ... +1; +2) to mutations. Such absolute values of s are not of primary importance, yet the distribution of these s -values among all mutations is crucial.

A) Two algorithms for generation of mutations.

SATURATION MODE

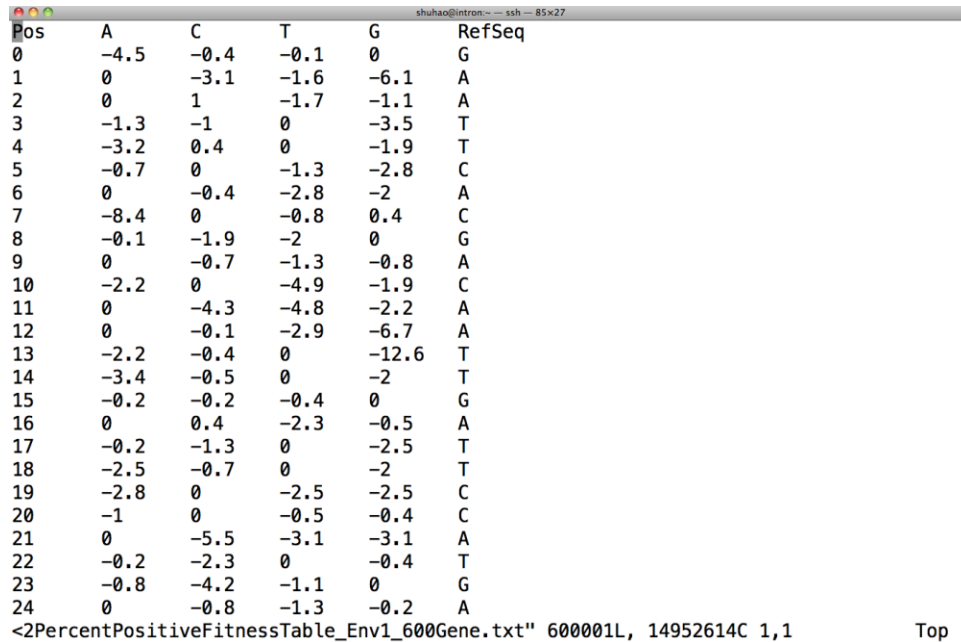
In the saturation mode the s -distribution table initially must be used as input for the MAGEr01.java (java program as available in the resources part [Quick Link](#)) that creates a complete table of selection coefficients for every possible



mutation for each genomic position (see *Table 2*). To obtain this table a user should activate MAGEr01.java code in the following command line:

Command line: `java MAGEr01 example.in`

The java program code and the example.in are available in the resources part ([Quick Link](#)). The code that creates the *Table 2* lies in the FitnessCalc class.



```

shuhao@intron ~ -- ssh -- 85x27
Pos  A      C      T      G      RefSeq
0    -4.5  -0.4  -0.1  0      G
1     0    -3.1  -1.6  -6.1  A
2     0     1    -1.7  -1.1  A
3    -1.3  -1     0    -3.5  T
4    -3.2  0.4   0    -1.9  T
5    -0.7  0     -1.3  -2.8  C
6     0    -0.4  -2.8  -2     A
7    -8.4  0     -0.8  0.4   C
8    -0.1  -1.9  -2     0     G
9     0    -0.7  -1.3  -0.8  A
10   -2.2  0     -4.9  -1.9  C
11   0    -4.3  -4.8  -2.2  A
12   0    -0.1  -2.9  -6.7  A
13  -2.2  -0.4  0    -12.6 T
14  -3.4  -0.5  0     -2    T
15  -0.2  -0.2  -0.4  0     G
16   0    0.4   -2.3  -0.5  A
17  -0.2  -1.3  0    -2.5  T
18  -2.5  -0.7  0     -2    T
19  -2.8  0     -2.5  -2.5  C
20  -1     0    -0.5  -0.4  C
21   0    -5.5  -3.1  -3.1  A
22  -0.2  -2.3  0    -0.4  T
23  -0.8  -4.2  -1.1  0     G
24   0    -0.8  -1.3  -0.2  A
<2PercentPositiveFitnessTable_Env1_600Gene.txt" 600001L, 14952614C 1,1
Top

```

Table 2 An example of a segment of Selection Coefficient Table that is required for the saturation mode.

The explanations for the Selection Coefficient Table are as follows: The first column of this table represents the numerical position of every nucleotide in the reference genome, while the last column represents the particular nucleotide (*N*, which could be A, C, G, or T) in the reference genome. The second column (nucleotide A), third column (nucleotide C), fourth column (nucleotide G), and the fifth column (nucleotide T) represent the *s*-values when nucleotide *N* mutates into A, C, G, or T accordingly. These *s*-values are obtained using MAGEr01 java program that produces specific numbers based upon the tabulated and normalized input table (*Table 1* as an example). For

example, when in the position #2 the reference nucleotide A mutates to nucleotide G, the selection coefficient for this $A_2 \rightarrow G_2$ mutation will be -1.7. Eventually, if reverse mutation occurs, the selection coefficient for it must be reversed (+1.7). When this model is used for a modeling a vast number of generations the fitness of the population eventually reaches saturation (equilibrium). For example, the most beneficial mutations may be already fixed in the population and the fitness at that point cannot be improved. The time when the system reaches its equilibrium depends on the size of the genome and the number of mutations per gamete.

UNSATURATED MODE

In this mode a random number generator creates a site for a mutation and also chooses nucleotide for this change. Then, next programming stage utilizes another random number generator to generate a selection coefficient for this mutation based on the same normalized tabulated input table (see Table 1). For example, for a trivial case of s -distribution with only three options: 90% ($s=0$); 9% ($s=-1$) and 1% ($s=+1$) a reference nucleotide G in a position 1483 will be mutated to C and the program will assign to this mutation a selection coefficient $s = -1$ with a 10% probability. Eventually this new C nucleotide at this position 1483 may mutate back to the G nucleotide. For this event the unsaturated model does not remember the previous s -value, and it generates a brand-new s -value again from the same input table and random number generator. Thus there is a 10% chance that this reverse mutation will also be assigned a selection coefficient $s = -1$. This unsaturated model very roughly reflects the evolution where a reverse mutation usually occurs in a different already mutated nucleotide context and, thus, may have different effect. Importantly, this model preserves the user defined ratio between deleterious and beneficial mutations that will never be changed.



NOTE: For a limited number of generations the unsaturated and saturation mode are identical. A user will start see the difference between them when mutations be fixed in about 20% of genomic positions.

B) Settings for either mutations or recombination coming first

Questions about either mutations or recombination coming first are brought forth in the development of MAGEr1.pl. It happens while considering novel mutations for a newly created individual. There are two possible situations. In The first case, mutations occur after recombination. The individuals that are being formed by the combination of both paternal and maternal gametes will acquire the same mutation number (2μ). In the second case, mutations occur before recombination and different gametes may have different numbers of novel mutations due to random sampling.

In MAGEr1.pl, at the very beginning of our code, we provide a variable to customize the settings of whether mutations happen before or after recombinations. The codes with its comments are shown below:

```
$mutations_before=0; #when $mutations_before =1, mutations will be created before recombinations. Alternatively when the variable is set to 0, mutations occur after recombinations.
```

[--return to MENU--](#)



Speed in MAGEr1.pl

Nearly every simulation program nowadays has to deal with efficiency issues or in other words, with the running time for the program. In MAGEr1.pl, the running time mostly depends on the user's settings which include: the population size (N), number of off spring (α), number of mutations per gamete (μ), number of recombination (r), the dominance coefficient (h). Two parameters of μ and N are strongly associated with the number of SNPs in a population. With a large number of SNPs, the memory requirements increase implying a longer run time for the program. A user may roughly estimate that by doubling the μ or N values the program will be twice slower. Also, r and h may also affect the number of SNPs in the population and, thus, change the speed of the program.

A user should keep in mind that a considerable portion of SNPs eventually will be fixed in a model population. Yet, computationally, these fixed SNPs will be kept in the arrays of SNPs for each individual and their constant accumulation considerably slows down the speed of the program. To avoid this predicament, MAGE periodically looks for all fixed mutations, removes them from the SNPs arrays and, simultaneously, creates a new reference genome that includes all fixed mutations. Such fixation by default occurs every **10** generations (the number can be defined by the user).

An example of MAGE speed is provided below. The specific setting for this running is shown in the table on the left.

Number of Individuals	24
Number of Offspring	5
Number of Genes	600
Mutations per gamete	5
Recombination/gamete	1
Dominance coefficient	0.5
Gene length (nts)	1,000
Before/after mutation	0 (after)

For the parameters shown in the table on the left, it took MAGEr1.pl 5 hours and 36 minutes for running up to 10,000 generations on a Linux AMD workstation (AMD Opteron(tm) Processor 6128 HE, and 128 Gb of RAM). Note that number of genes and gene lengths also influence the



speed of the program execution. The larger the genome the more time it takes to accomplish the calculations.

[--return to MENU--](#)



Backup file in MAGEr1.pl

MAGE can be used for processing extremely large and even whole genome sequences. The larger the sequence size, the more increased will be the burden on computational time and memory constraints. It is possible that such computational constraints are exhausted depending on the user-defined parameters that subsequently procrastinates the program run to indefinite delays. Thus, for this purpose, MAGE regularly creates its backup files (as shown in *Figure 3*) in the output directory every time it runs. The backup files contain arrays of SNPs for each virtual individual. These data may be used to investigate mutation dynamics in the population with the highest precision.

By default, MAGE creates back-up files A and B for every **20** generations. This number can be user modified as shown below:

```
#shu Create the Backup file
if ($g % 20 == 0){
    if ($g % 40 == 0){
        $tag = 'B';
        open (BACKUPB, ">$result/backup_B");
        BACKUPB -> autoflush;
    }else {
        $tag = 'A';
        open (BACKUPA, ">$result/backup_A");
        BACKUPA -> autoflush;
    }
    {... Further computational blocks}
}
```

It can be seen that the above code actually creates two backup files A and B. By doing so, MAGE provides another level of protection in case of program premature termination while creating one of these backup files.



```

shuhao@intron:~/MAGE_MatingScheme/P-200-C-5-U-20-R-10-M-1-H-0.5 -- ssh -- 127x37
4700 53252 599 200 5 599999 1000 20 10 0.5 0 0
pi1 1 0 0 46
pi1 1 0 1 G
pi1 1 0 2 0
pi1 1 1 0 26
pi1 1 1 1 T
pi1 1 1 2 -1
pi1 1 2 0 99
pi1 1 2 1 G
pi1 1 2 2 -1
pi1 1 3 0 198
pi1 1 3 1 C
pi1 1 3 2 -1
pi1 1 4 0 195
pi1 1 4 1 C
pi1 1 4 2 1
pi1 1 5 0 250
pi1 1 5 1 C
pi1 1 5 2 -1
pi1 1 6 0 523
pi1 1 6 1 G
pi1 1 6 2 -1
pi1 1 7 0 796
pi1 1 7 1 T
pi1 1 7 2 -1
pi1 1 8 0 757
pi1 1 8 1 C
pi1 1 8 2 0
pi1 1 9 0 599
pi1 1 9 1 T
pi1 1 9 2 -1
pi1 1 10 0 642
pi1 1 10 1 G
pi1 1 10 2 0
pi1 1 11 0 247
pi1 1 11 1 G

```

Figure 3 Screenshot for the illustration of the backup file.

The first part is only the first line that presents the general information regarding the experiment. All the parameters are described as follows:

- 1st number specifies the generation that has been saved. In this example, the number 4700 in the figure means that the program saved the information for the 4700th generation;
- 2nd number represents the total number of nucleotides that has been fixed until 4700 generations;
- 3rd number represents the number of genes in the genome;
- 4th number represents the size of the population that is being run;
- 5th number represents the number of offspring per couple;
- 6th number represents the size of the reference genome;
- 7th number represents the length of each gene;
- 8th number represents the number of mutations per gamete;
- 9th number represents the number of recombination events per gamete;



- 10th number represents the dominance coefficient;
- 11th number represents the starting fitness value for the consensus genome for the population;
- 12th number represents the setting mode for mutations which may occur before or after recombination events. For example, when this parameter is 0, mutations will be created after recombination events; otherwise it will be created before recombination.

The second part is a matrix composed of SNPs information for each virtual individual (see *Figure 3* from the 2nd line and below).

There are 5 columns in total. *Pi1* represents the paternal chromosome of the first individual. The second column specifies the position of the current gene in the genome. In the above example, 1 denotes the second gene (the first gene is 0, because counting in Perl starts from zero). The third column represents the consecutive order of the SNP in the gene. The fourth column has only three values: 0, 1, or 2.

Value 0 means that the next column represents the position of the SNP in the gene; value 1 means that the next column specifies the mutant nucleotide; while value 2 means that the next column represents the s-value for this mutation. The MAGEr1.pl program keeps this information as multidimensional array @pi.

The structure of this array is the following:

```
#{pi} [#gene] [#SNP] [Position, Nucleotide, SelectionCoefficient]
```

Below is the example of the conversion of the backup table into @pi array:

pi1	1	0	0	46
pi1	1	0	1	G
pi1	1	0	2	0

The array may be reconstructed from this table as follows:

```
#{pi1}[1][0][0] = 46;    #{pi1}[1][0][1] = 'G';    #{pi1}[1][0][2] = 0.
```



The number 46 represents the position of the SNP in the second gene. Next value specifies that mutant nucleotide in this position is the nucleotide G. Finally, 0 means that the selection coefficient for this SNP is 0.

The third part is the fitness table for the selection coefficient for the whole genome. It has been shown as *Figure 4* and illustrated on the website:

<http://bpg.utoledo.edu/~afedorov/lab/mage.html> in the MAGEr1.pl_Video part.

```

shuhao@intron:~/MAGE_MatingScheme/P-200-C-5-U-20-R-10-M-1-H-0.5 -- ssh -- 90x39
mi200 598 2 1 T
mi200 598 2 2 -1
mi200 598 3 0 394
mi200 598 3 1 G
mi200 598 3 2 -1
0 -1 -1 -1 -1 G
1 -1 -1 -1 -1 T
2 -1 0 -1 -1 A
3 -1 1 1 1 A
4 -1 0 1 -1 G
5 -1 0 0 1 A
6 -1 -1 -1 0 A
7 0 -1 -1 -1 A
8 -1 -1 -1 -1 C
9 -1 -1 -1 0 C
10 -1 -1 -1 -1 C
11 0 -1 -1 1 T
12 1 -1 -1 -1 A
13 -1 -1 -1 0 A
14 -1 -1 -1 -1 A
15 1 -1 -1 -1 A
16 -1 0 -1 -1 A
17 -1 -1 1 -1 A
18 -1 -1 -1 -1 A
19 -1 0 -1 -1 A
20 -1 -1 0 -1 A
21 -1 -1 -1 -1 A
22 -1 -1 1 -1 A
23 -1 0 -1 0 A
24 -1 -1 -1 -1 A
25 -1 1 -1 0 A
26 -1 -1 -1 -1 A
27 -1 -1 -1 -1 A
28 -1 -1 -1 -1 T
29 -1 -1 -1 0 A
30 -1 -1 -1 1 C
31 -1 -1 -1 -1 A
32 -1 -1 -1 -1 A
3692181,1 86%

```

Figure 4 An example of the transition from the second part into the third part of the backup file.

As shown in figure 4, the third part has 6 columns in total. The first column is the position of the nucleotide. The 2nd to 5th columns contain the selection coefficients for “A, C, T, G” respectively. The last one is the sequence of the current reference genome (remember that the reference genome is periodically updated by including all fixed mutations).



Mating Schemes

Mating schemes may influence on the population fitness and the distribution of SNPs. For the purpose of exploring the effects of mating schemes on the population parameters, we have created 7 different mating schemes in the release 2 of MAGEr1.pl (will be available on March 31st 2014). Following are the illustrations for the different types of mating schemes.

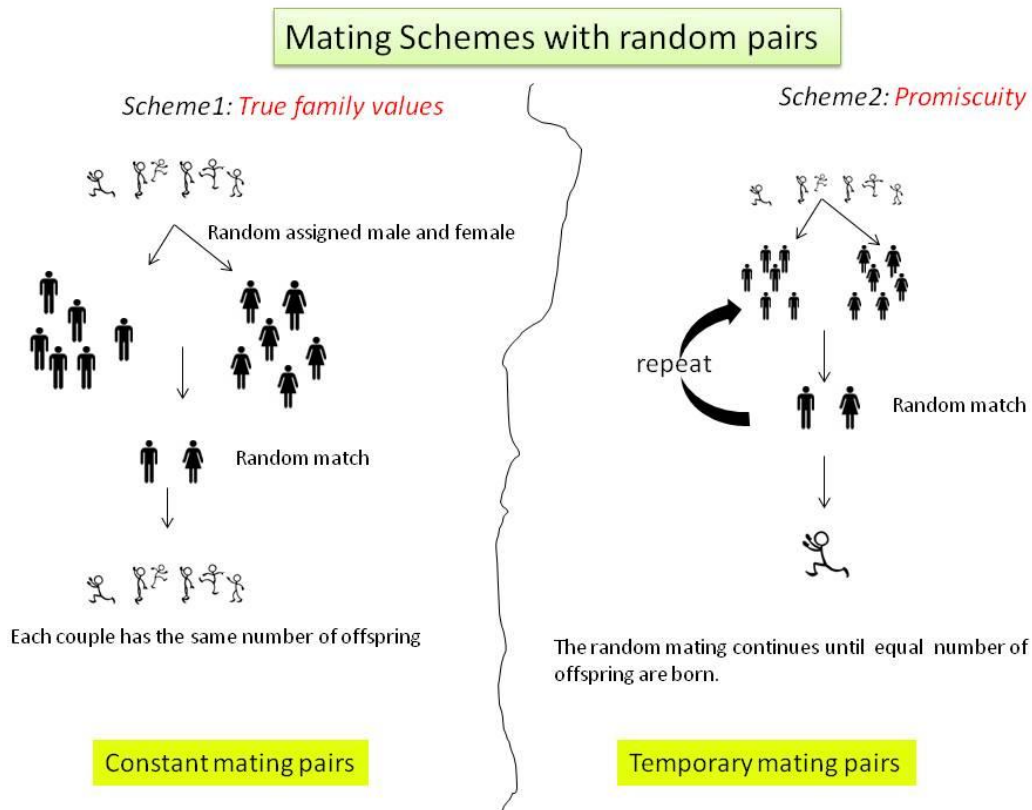


Figure 6 Simplified flowcharts for Scheme1 and Scheme2. Scheme1 has fixed couples (every individual participates in one pair) in every generation while couples in Scheme2 are temporary (every individual participates in multiple pairs, which number equal to the number of offspring per individual) and produce one offspring a time. The total number of offspring per virtual individual is same for each generation.



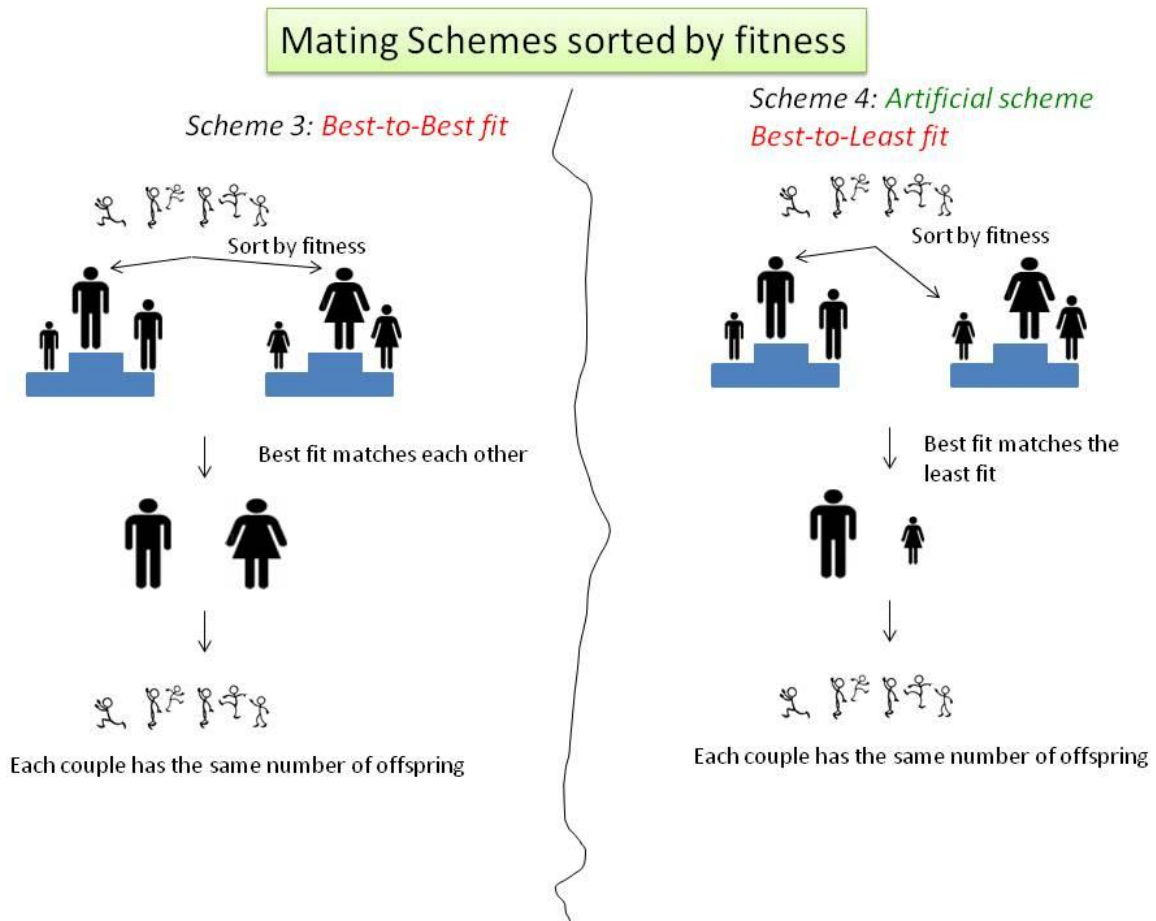


Figure 7 Simplified flowcharts for Scheme3 and Scheme4. The matching between the male and female are considered based on their fitness. The Scheme 4 is artificial and does not exist in nature.



Mating Schemes sorted by fitness & variable offspring numbers

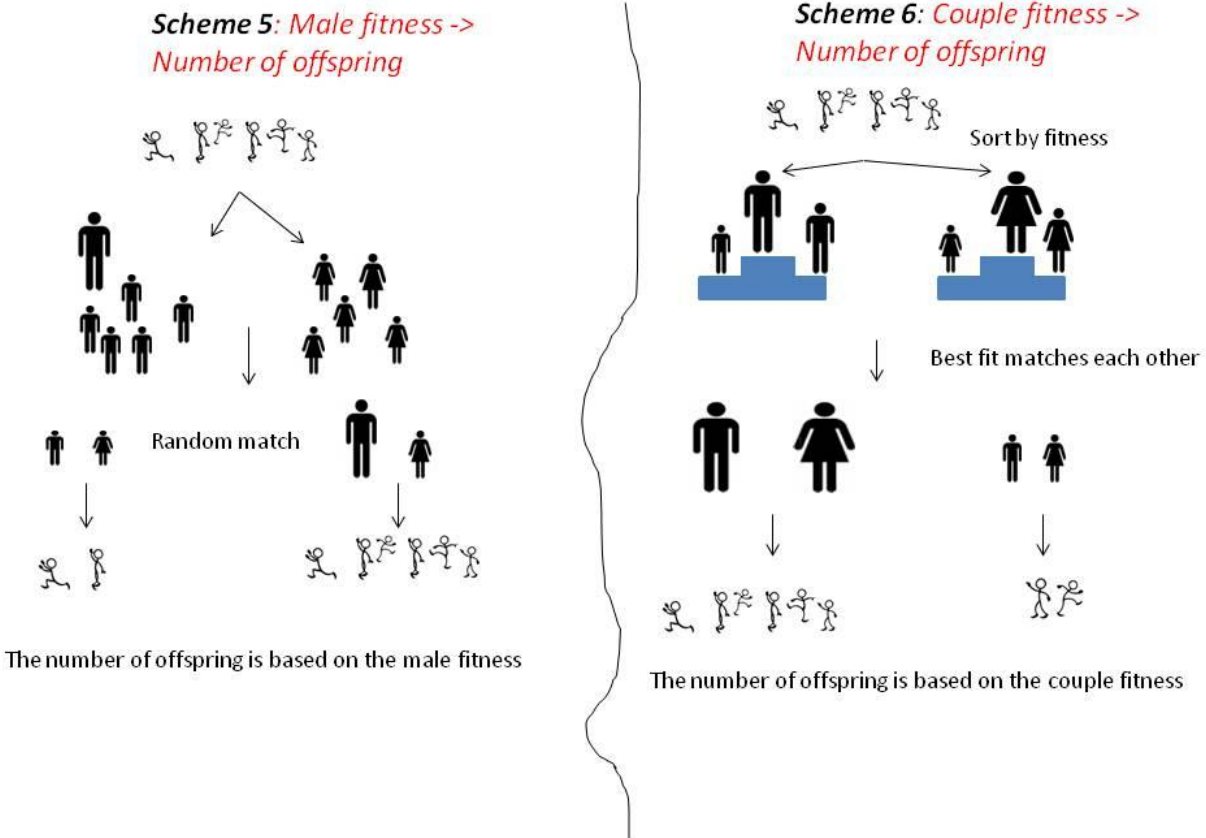


Figure 8 Simplified flowcharts for Scheme5 and Scheme6. The mating pairs are established based on the fitness of individuals as in Schemes 3 & 4. Additionally, the number of offspring for the pair depends on the fitness of parents. However, the total number of offspring per generation is the fixed parameter and is equal to $N \times \alpha / 2$. Scheme5 is mainly based on the fitness of the male while Scheme6 consider both male and female.



Mating Schemes: **Herd – Only the Best male is reproduced**

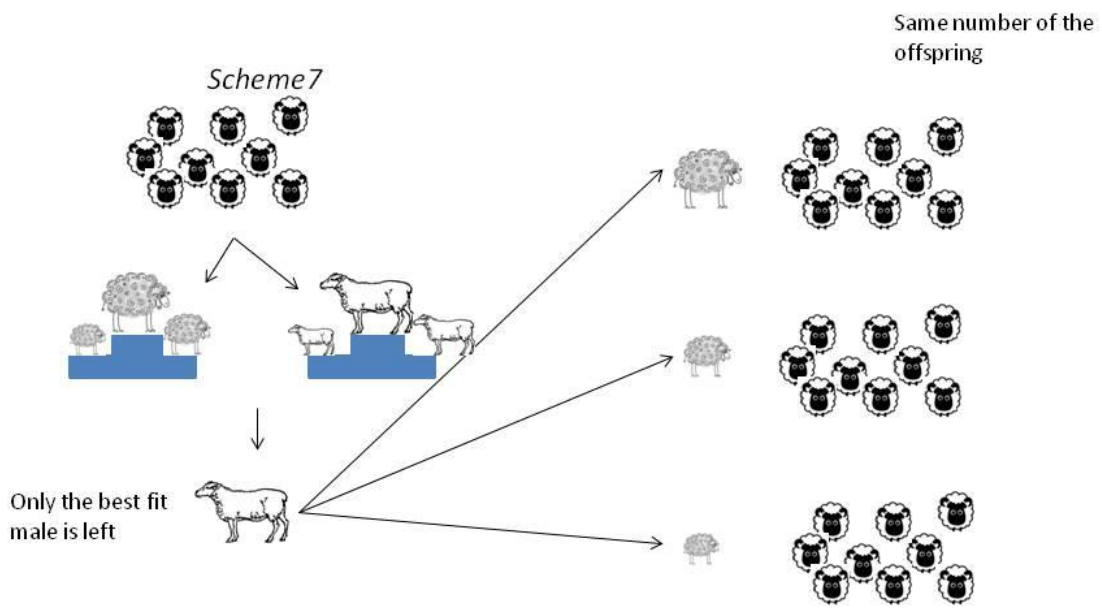


Figure 9 Simplified flowcharts for Scheme7. This mating scheme most mimics the herd reproducing structure. In this mating scheme, only the fittest male is involved in the sexual reproduction of the next generation with multiple female partners.

[--return to MENU--](#)



Current Versions of MAGE

Currently, there are two available versions of MAGE written in different programming languages (Perl and Java: MAGEr1.pl and MAGEr01.java). The lesser sophisticated version of MAGE is written in Perl with the intention to investigate or explore the algorithms for Population genetics. The advanced object-oriented java version of MAGE not only improves the calculation efficiency (e.g. by implementing multiple threads in the running of MAGE) of the program but also makes it much more flexible. Some genome features like codon model (intra codon recombination), codon usage bias and recombination hot spots can be realized only by using the advanced java version of MAGE by embedding components and features as “Objects” into the program. In the following sections we explain how to use MAGEr1.pl.

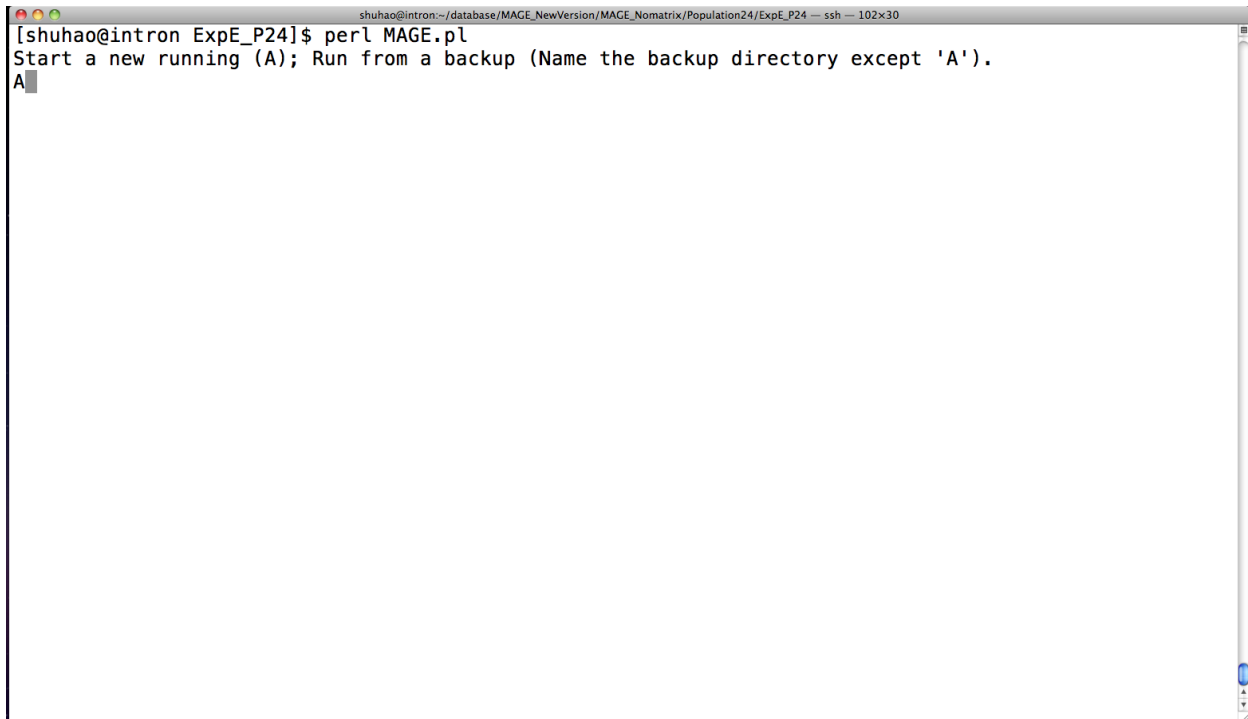
[--return to MENU--](#)



Step by Step illustrations of MAGE

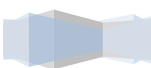
Input data

1st screenshot for running MAGEr1.pl in the command line.



```
shuhao@intron:~/database/MAGE_NewVersion/MAGE_Nomatrix/Population24/ExpE_P24 — ssh — 102x30
[shuhao@intron ExpE_P24]$ perl MAGE.pl
Start a new running (A); Run from a backup (Name the backup directory except 'A').
A
```

Upon executing MAGE program, the user can either start an entirely new run or chose to continue from a previously saved backup point. If one choses a new run, capital “A” must be input from the keyboard.



Selection Coefficient file

2nd & 3rd screenshots for running MAGEr1.pl in the command line.

```

shuhao@intron:~/database/MAGE_NewVersion/MAGE_Nomatrix/Population24/ExpE_P24 — ssh — 102x30
[shuhao@intron ExpE_P24]$ perl MAGE.pl
Start a new running (A); Run from a backup (Name the backup directory except 'A').
A
Please choose a Selection Coefficient file.

```

After selecting the option “A”, the program will require the user to define a selection coefficient file. The selection coefficient file is a matrix containing all possible mutations occurring at any position within the genome accompanied with their selection coefficient values s as shown below and also demonstrated on Table 2.

```

shuhao@intron:~/database/MAGE_NewVersion/MAGE_Nomatrix/Population24/ExpE_P24 — ssh — 102x30
Position    A      C      T      G
0           -1     -1     -1     -1     A
1           -1     -1     -1     -1     A
2           -1     0      -1     -1     A
3           -1     1      1      1      A
4           -1     0      1      -1     A
5           -1     0      0      1      A
6           -1     -1     -1     0      A
7           0      -1     -1     -1     A
8           -1     -1     -1     -1     C
9           -1     -1     -1     0      A
10          -1     -1     -1     1      C
11          0      -1     -1     1      A
12          1      -1     -1     -1     A
13          -1     -1     -1     0      A
14          -1     -1     -1     -1     A
15          1      -1     -1     -1     A
16          -1     0      -1     -1     A
17          -1     -1     1      -1     A
18          -1     -1     -1     -1     A
19          -1     0      -1     -1     A
20          -1     -1     0      -1     A
21          -1     -1     -1     -1     A
22          -1     -1     1      -1     A
23          -1     0      -1     0      A
24          -1     -1     -1     -1     A
25          -1     1      -1     0      A
26          -1     -1     -1     -1     A
27          -1     -1     -1     -1     A
"ExpE.txt" 600001L, 12010112C
1,1 Top

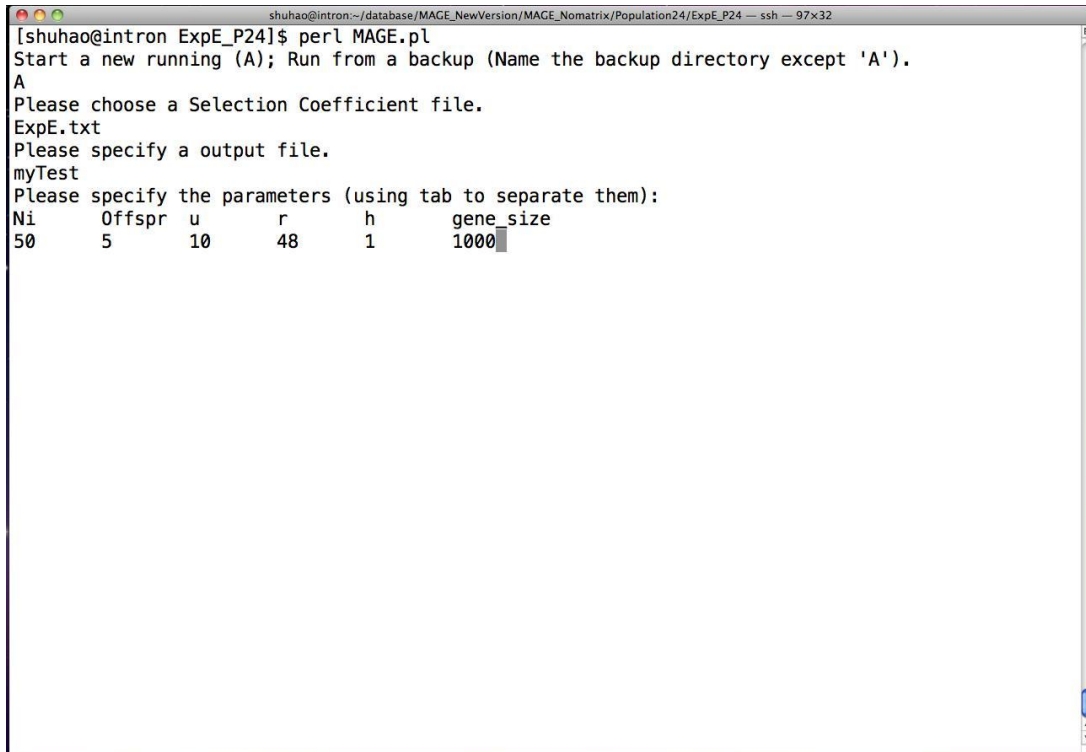
```



Input Parameters

4th screenshot for running MAGEr1.pl in the command line.

The primary parameters for running MAGE are set up as follows:



```

shuhao@intron ~/database/MAGE_NewVersion/MAGE_Nomatrix/Population24/ExpE_P24 — ssh — 97x32
[shuhao@intron ExpE_P24]$ perl MAGE.pl
Start a new running (A); Run from a backup (Name the backup directory except 'A').
A
Please choose a Selection Coefficient file.
ExpE.txt
Please specify a output file.
myTest
Please specify the parameters (using tab to separate them):
Ni      Offspr  u      r      h      gene_size
50      5        10     48     1      1000

```

- Number of individuals (Population size, Ni)

MAGE does not have any upper limit for the population size. However, the larger the size of population, the longer will be the time taken.

- Number of off spring (Offspr)

Each individual can have different number of off springs. In the Perl version of MAGE, this setting is constant, implying that each individual will have the same number of off spring that is once user-defined. In the Java version however, the off spring number for each individual is related to its fitness value. The total number of off spring in both cases still equals $Ni * \text{Offspr}$.

- Number of novel mutation per gamete (u)
- Number of recombinations between the paternal and maternal genomes (r)



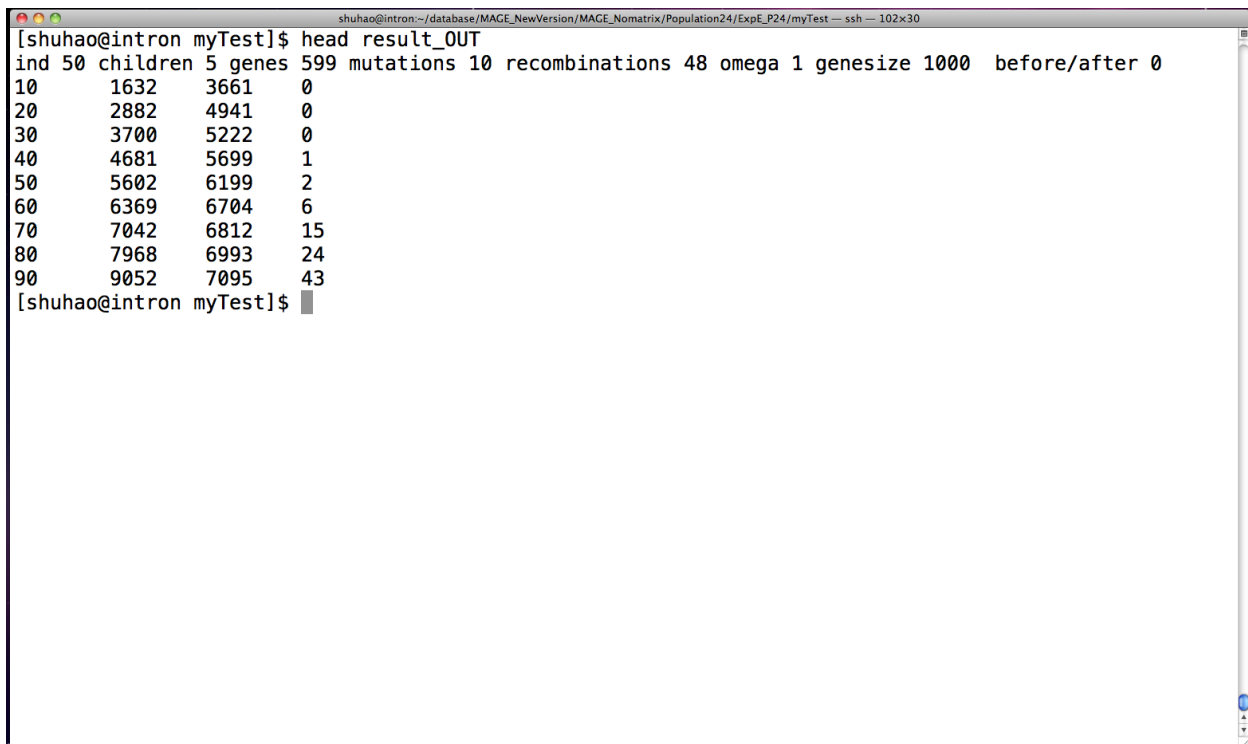
- Dominance coefficients (h)
- gene_size

Depending on the total length of genome in the matrix, the size of each gene (gene_size parameter) allows MAGE to mimic multiple genes.

Output files

The Output directory contains 3 separate files. The first two are for backup use, named “backup_A” and “backup_B”. As shown in the following figure, a third file “result_OUT” contains all the detailed results for MAGE.

5th screen shot for MAGE output file.



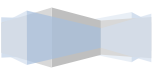
```
[shuhao@intron:~/database/MAGE_NewVersion/MAGE_Nomatrix/Population24/ExpE_P24/myTest -- ssh -- 102x30]
[shuhao@intron myTest]$ head result_OUT
ind 50 children 5 genes 599 mutations 10 recombinations 48 omega 1 genesize 1000 before/after 0
10      1632      3661      0
20      2882      4941      0
30      3700      5222      0
40      4681      5699      1
50      5602      6199      2
60      6369      6704      6
70      7042      6812     15
80      7968      6993     24
90      9052      7095     43
[shuhao@intron myTest]$
```

As displayed above, the first line in the “result_OUT” contains the information for MAGE input parameters. The columns in the subsequent lines denote the results of the following parameters from left to right: the number of generations, total



fitness for the whole population, number of SNPs for the entire population, and number of fixed mutations.

[--return to MENU--](#)



Resources

- Genome Evolution Model (GEM): Design and Application (extensive description of MAGEr01 Java Version)

https://etd.ohiolink.edu/ap:10:0::NO:10:P10_ACCESSION_NUM:mco1290550446

- Codes for MAGE Perl Version

http://bpg.utoledo.edu/~afedorov/lab/DOWNLOADS_WEB/MAGE.txt

- Codes for MAGE Java Version

http://bpg.utoledo.edu/~afedorov/lab/DOWNLOADS_WEB/MAGE.java.txt

- Pseudocode for MAGE Java Version

http://bpg.utoledo.edu/~afedorov/lab/DOWNLOADS_WEB/MAGE_Java_pseudocode.pdf

- MAGE Perl Version Video Illustration

<http://bpg.utoledo.edu/~afedorov/lab/videos/MAGE.mov>

- Exponential Decay Distribution.xls

http://bpg.utoledo.edu/~afedorov/lab/DOWNLOADS_WEB/Exponential_Decay_Distribution.xls

[--return to MENU--](#)



References

Carvajal-Rodriguez A (2008) GENOMEPOP: A program to simulate genomes in populations. *Bmc Bioinformatics* 9.

Hernandez RD (2008) A flexible forward simulator for populations subject to selection and demography. *Bioinformatics* 24(23):2786-2787.

Chadeau-Hyam M, et al. (2008) Fregene: Simulation of realistic sequence-level data in populations and ascertained samples. *Bmc Bioinformatics* 9.

Sanford J BJ, Brewer W, Gibson P, Remine W (2007) Mendel's Accountant: A biologically realistic forward-time population genetics program. *SCPE* 8:147-165.

[--return to MENU--](#)



Appendix

List of alternative software for computer simulations of genome evolution:

Backward Simulators

Simulator	Reference
ms	(Hudson, 2002)
SNPsim	(Posada and Wiuf, 2003)
SimCoal2	(Laval and Excoffier, 2004)
Coasim	(Mailund et al., 2005)
Cosi	(Schaffner et al., 2005)
GeneArtisan	(Wang and Rannale, 2005)
FastCoal	(Marjoram and Wall, 2006)
Recodon	(Arenas and Posada, 2007)
msHot	(Hellenthal and Stephens, 2007)
GENOME	(Liang et al., 2007)
MIcoalsim	(Ramos-Onsins and Mitchell-Olds, 2007)
MaCS	(Chen et al., 2009)
AquaSplatche	(Samuel Neuenschwander, 2006)
Splatche	(Currat et al, 2004)
CoalFace	(W Delpont, 2006)
IBDsim	(R Leblois et al, 2009)
SelSim	(CCA Spencer, 2008)
Serial Netevolve	(P Buendia, 2006)
Bayesian Serial SimCoal	(CNK Anderson, 2005)
fastsimcoal	(L Excoffier, 2011)

Forward Simulators

Simulator	Reference
simuPOP	(Peng and Kimmel, 2005)
genomeSIM	(Dudek et al., 2006)
Nemo	(Guillaume and Rougemont, 2006)
FREGENE	(Hoggart et al., 2007)
GenomePop/ GenomePop2	(Carvajal-Rodriguez, 2008)
genomeSIMLA	(Edward et al., 2008)
SFS_CODE	(Hernandez, 2008)
ForSim	(Lambert et al., 2008)
QuantiNEMO	(Neuenschwander et al., 2008)
ForwSim	(Padhukasahasram et al., 2008)
BottleSim	(CH Kuo, 2003)
Cdpop	(EL Landguth, 2010)
Easypop	(F Balloux, 2001)
Mendels Accountant	(J Sanford, 2001)
Pedagog	(JA Coombs, 2010)
QMSim	(M Sargolzaei, 2009)
RmetaSim/ metasim	(DC Richter, 2008)
KernelPop	(AE Strand, 2007)
Spip/ Spip_m	(EC Anderson, 2005)

[--return to MENU--](#)

