Haplotype Simulator V1.0

Table of Contents

Haplotype Simulator V1.0 ..... **1**
The simulation **1**
Running the programme. **4**
Double click on icon 4
From command line 4
Running without GUI 4
Output 4
Memory usage 5
Programme parameters: **5**
Recombination rate 5
Number of Generations 5
Population size 6
Chromosome Size 6
Number of Haplotypes. 6
Ploidy 7
Family Size 7
HotSpot Interval 8
Telomere Length 8
Interference Interval 8
Output Interval 8
Number of parallel processes 9
Number of replicates 9

The simulation

The process of recombination is simulated by taking two random chromosomes, and obtaining a number of breakpoints to insert from the user supplied

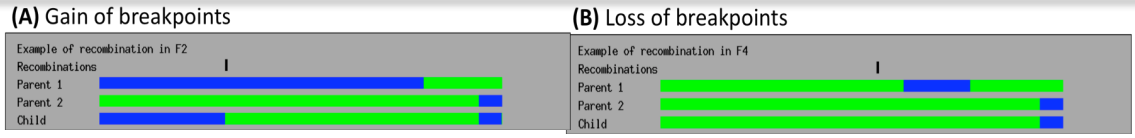

Fig S1. Recombination examples. The two panels show examples of A) gain or B) no gain of a breakpoint during recombination. In each panel the upper two bars represent the parental chromosomes and the lower bar represents the child chromosome. The position of the new break is shown by the solid black vertical bar. The child chromosome is created by taking the left of the chromosome from Parent 1 up to the position of the bar, it then takes the right of the chromosome from Parent 2 from the bar up to the end of the chromosome. If the two chromosomes are different at the break point the position of the break point will be identifiable in the child chromosome (Panel A). However if the two chromosomes are identical at the break position the position of the break will not be detectable in the child and the breakpoint will be effectively lost (Panel B). Hence if the two founder haplotypes are present at equal frequencies there is a 50% chance that a crossover will result in an observable breakpoint.

distribution. The child chromosome is constructed by taking from the first chromosome up to the first breakpoint then taking from the second chromosome up to the second breakpoint and so on up until the end of the chromosomes (Fig 1). At each breakpoint the parental chromosomes are examined and if the haplotypes are different at the breakpoint then a new break is added to the list of breaks on the child chromosome. If the two parental chromosomes have the same ancestral haplotype at the break point then no new break is recorded.

Chromosomes (or chromosome pairs for a diploid model) are selected at random for recombination with replacement. This means that the number of child chromosomes per parent chromosome will be Poisson distributed with about 37% of parent chromosomes having no child chromosomes and about 26% having two or more child chromosomes.

In the simulator the following parameters can be specified by the user: the chromosome size; the number of chromosomes in the population; the number of founder haplotypes; the number of generations to simulate; an interference interval between adjacent crossovers and the relative frequencies of 0, 1, 2 and 3 crossovers per meiosis and whether the population is diploid or haploid. The number of replicates can be specified in order to estimate the mean and standard deviation of the observations. The simulator generates graphs and tables that show simulated results alongside expected values obtained using Equation 1 and the two sets of values are in agreement (Main text Figure 2).

$$\lambda_g = r + rp\sum_{i=3}^{i=g}\left(\frac{2N_e-1}{2N_e}\right)^{(i-3)} = r + rp\left(\frac{1-((2N_e-1)/2N_e)^{(g-2)}}{1-((2N_e-1)/2N_e)}\right)$$

Equation 1 Where λ is the number of observable junctions, g is the generation number, r is the mean number of crossovers per meiosis, p is the probability of a recombination between two different haplotypes and N_e is effective population size and other symbols are as in equation 2 of the main text. At the F2 generation $\lambda = r$. The expression on the right is the standard closed form of the sum of an exponential series. The geometric series on the left is correct for $g > 2$, the expression on the right is correct for $g >= 2$ since it evaluates to r when $g = 2$. Generations are assumed not to overlap.

Interference interval had negligible effect on simulated haplotype block lengths, presumably because it is relatively rare to have more than one crossover in a meiosis (<10% of crossovers in mice [1]) and random crossovers would only be expected to be within a plausible (~20Mb) distance about 20% of the time and hence only about 2% of crossovers will be affected by interference. The ratios of 0, 1, 2 and 3 crossovers per meiosis also made no difference to simulated haplotype block lengths as long as the mean number of crossovers was held constant.

The algorithm is implemented in a Java programme with a graphical user interface and can also be run from the command line. It generates results that are not significantly different from calculation using equation 1 over a broad range of conditions.

The programme is distributed as a jar file (HaplotypeSimulator.jar which is available as Dataset 1. Source files area available as Dataset 2.

HaplotypeSimulator is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version. See http://www.gnu.org/licenses/.

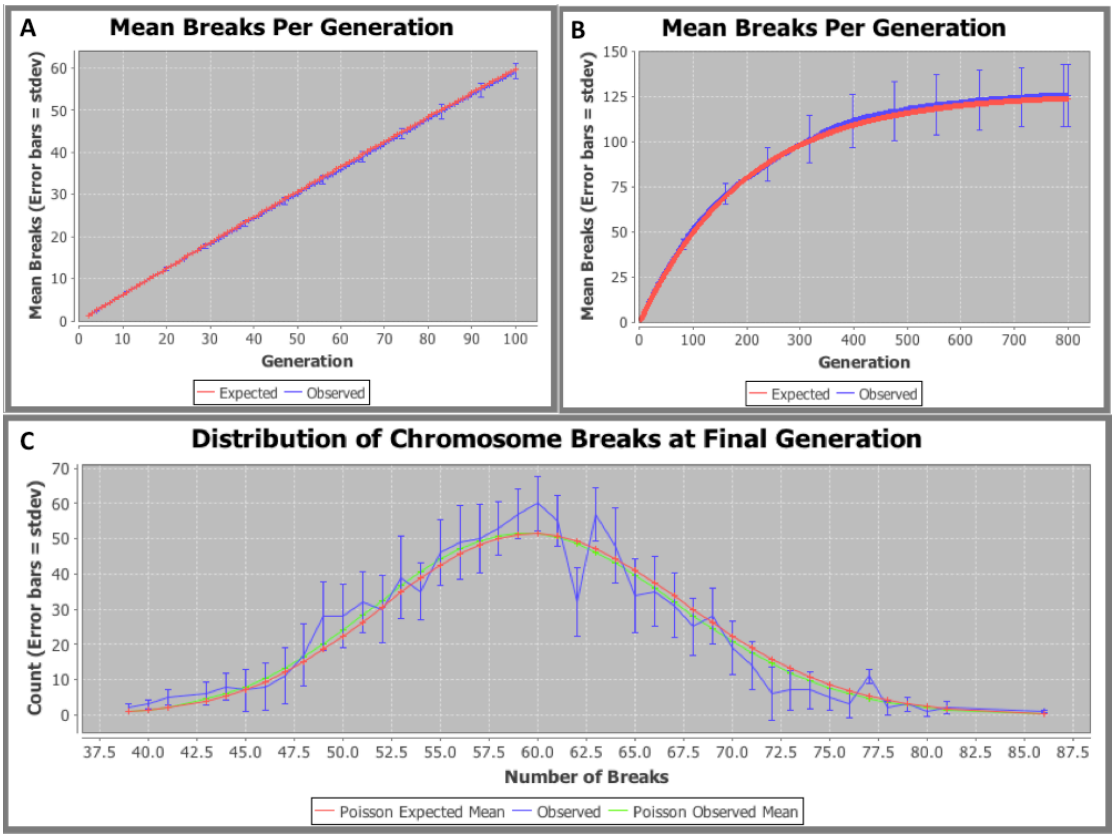

Figure S2. Output from the simulator. **A** shows the mean number of breaks on each chromosome over 100 generations, two founder haplotypes, a diploid population size of 500 (1000 chromosomes) and the human frequencies of 0, 1, 2 and 3 crossovers per meiosis which gives a mean recombination rate of 1.25 crossovers per chromosome per meiosis. In a population of this size the accumulation of breaks is almost linear for over 100 generations. The blue line with error bars represents the simulated numbers of breaks and the red line the expected number of breaks from equation 1. Error bars representing standard deviations are estimated from 30 replicate simulations. **B** shows a similar simulation to A except with a population size of 100 (200 chromosomes) over 800 generations during which time the chromosomes have almost drifted to fixation. **C** shows the distribution of the number of chromosomes with different numbers of break points in the 100th generation of the simulation shown in plot A. The blue line shows the distribution of breakpoints obtained by simulation, the redline shows a Poisson distribution with a mean calculated with equation 2 and the green line shows a Poisson distribution with the observed mean of the simulation.

Running the programme.

The programme was written with Java 1.6 and has not been tested on other versions. The open source packages JFreechart [2] is used for plotting graphs and Batik (http://xmlgraphics.apache.org/batik/) for generating SVG vector graphic output files. All necessary software is packaged within the jar file and there are no external dependencies.

Double click on icon

The jar file can be run from any directory by clicking on the icon.

From command line

Alternatively it can be started from the command line with the command:

```
java -jar path/HaplotypeSimulator.jar
```

Exploring the application

The application opens on a demo page. Click start to obtain a graphical demo of the simulation process. Use the drop down selection to choose a model to simulate and run it with the default parameters. It will switch to an output view at the end of the run. Explore the different outputs.

Running without GUI

If a configuration filename is given as an argument on the command line then the graphical user interface will not open and the programme will run and return immediately using the parameters in the configuration file. This option makes it possible to incorporate the program into a larger package or pipeline. An example configuration file 'config.txt' is included in the distribution.

```
java -jar HaplotypeSimulator.jar config.txt
```

If the specified config file is not found the programme will request a new filename and wait for it to be entered. No sanity checking is applied to parameters in the config file and inappropriate values or combinations of values will crash or hang the programme. (Some basic sanity checks are applied to input from the GUI)

Output

At the end of a run the GUI will switch to the output tab, which shows summary statistics, a plot of the mean number of breaks at each generation and a plot of the distribution of break counts at the final generation. In each case observed and expected values are presented. If drift has been significant then the observed distribution of break counts at the final generation will differ substantially from the Poisson distribution that is presented.
At the end of a run the two plots in the output tab will be available to print out in W3C standard Scalable Vector Graphics (SVG) file format from the File menu. SVG files can be converted to EPS and PDF in the open source Inkscape package (http://inkscape.org/). The tabulated means of observed and expected values are also available to print out as a tab delimited text file.

Memory usage

The programme is allocated a maximum 500Mb of heap space by default. Longer (>2000 generations) or larger (Population size > 2000) can cause the programme to run out of memory.
Additional heap space can be allocated by using the parameter -XmxNg in the command line where "N" is the number of Gb of memory to be allocated

-Xmx4g would increase maximum heap space to 4Gb

```
java –Xmx4g –jar HaplotypeSimulator.jar
```

for more details see eg http://javarevisited.blogspot.com/2011/05/java-heap-space-memory-size-jvm.html

Programme parameters:

Model

Intercross, backcross or introgression models are available. The intercross is suitable for simulating an intercross for a mapping experiment in a model organism but also for long term evolution of any outbred population. The backcross is only relevant to model organism as it assumes that a constant supply of chromosomes from one of the founders is available. The introgression would be relevant for livestock breeding where one allele is introduced into a breed from another population as well as for the creation of congenic lines in model organisms.

Recombination rate

The recombination rate is entered by setting the relative integer numbers of 0, 1, 2 and 3 crossovers per meiosis. The values for humans[3] and mice[1] are available by selecting radio buttons. For humans the relative numbers are 51, 110, 100 and 10 for 0, 1, 2 and 3 crossovers. This example can be written as 0=>51, 1=> 110, 2=>100, 3=>10. The weighted average of these values gives the mean recombination rate, which for humans is 1.25 crossovers per meiosis. The weighted average is obtained by multiplying each number of crossovers by its count and then dividing the sum of those products by the sum of the counts.

$$\mu = \frac{1}{\sum c_i} \sum i c_i$$

The distribution of 0, 1, 2 and 3 crossovers makes no difference to the numbers of breaks observed provided that the mean is kept constant. For example 0=>1, 1=> 1, 2=>1, 3=>0 gives a mean recombination rate of 1 and will give rise to the same number of breaks as 0=>0, 1=> 1, 2=>0, 3=>0 which also gives a mean recombination rate of 1.

Number of Generations

Minimum 2; Maximum 10,000. The number of generations for which the simulation will run.
The population size and number of generations are the major determinants of memory usage and speed. When using large combinations of these two parameters it may be necessary to increase the amount of memory available to

the programme (see Running the Programme -> Memory usage above for how to do this).

Population size

Minimum 2; Maximum 10,000. For a diploid population this is the number of chromosome pairs that will be used for the simulation. For a haploid population it is the number of chromosomes. Since mating is random this is the effective population size (N_e) except if the Family size parameter is used when the effective population size will be less than the entered population size.

Chromosome Size

The chromosome size in base pairs: default 100,000,000; minimum 10, maximum 1,000,000,000. This parameter is not used in the calculation of the number of breakpoints and is only used to obtain haplotype lengths by dividing the given chromosome size by the simulated number of break points. However using very small values will lead to increased likelihood of multiple recombinations at the same site and is equivalent to increasing the Hot Spot Interval parameter. Eg a 100,000,000bp chromosome with a 10,000bp Hot Spot interval will give the same number of possible break points as a 10,000bp chromosome with a 1bp Hot Spot Interval.

Number of Haplotypes.

Number of founder haplotypes: default 2; minimum 2; maximum 1,000. The frequency of each haplotype is equal by default but can be set manually by the user if unequal founder haplotype frequencies are required. Note that it is the number of chromosomes with each haplotype that must be set and the numbers must sum to twice the population size (for a diploid model).
The effect of founder haplotype count on haplotype length is proportional to (1-1/h) where h is haplotype count if founder frequencies are equal and proportional to

$$\sum_{i=1}^{i=h}(1-f_i)f_i$$ (Equation 2) if frequencies are variable. Under either assumption increasing the founder haplotype count above about 10 has a very modest impact on mean haplotype length.
In the first generation of an intercross each chromosome is deemed to be a single haplotype. In this generation recombination of two chromosomes with the same haplotype is prohibited.
In the first generation of an intercross chromosomes are only permitted to recombine with different haplotypes in order to model a structured breeding programme such as mapping experiment between inbred lines.
An exhaustive list of all possible haplotype combinations in 1% increments was obtained for between 2 and 9 haplotypes, in order to obtain the distribution of *p* the probability that haplotypes will be different in a random intercross. The distribution of haplotype frequencies used to obtain the distribution of *p* is shown in figure 1 of the main paper.

Ploidy

For a Haploid model enter 1; for a Diploid model enter 2. If a diploid model is used then pairs of chromosomes will be created to represent each individual in the population.

Family Size

The number of child chromosomes created per parental chromosome on each occasion that a chromosome is selected for recombination. With a family size of 1 and a diploid model 2 child chromosomes are created each time a parental pair of chromosomes are selected for recombination. Since pairs are selected at random with replacement the number of times a given pair are selected will be Poisson distributed with about 37% of parent chromosomes having no child chromosomes and about 26% having two or more child chromosomes.
With a family size of 2 then 4 child chromosomes will be created each time a chromosome pair is selected and hence only half as many chromosomes will be selected for recombination and more chromosome pairs will have no offspring. In this case the numbers of child chromosomes per parent chromosome will no longer be Poisson distributed.
As implemented, the family size should more properly be described as litter size since it is the number of offspring from a given pair of individuals in the population when that pair is selected at random to mate. Either member of the pair can be selected again and will be randomly paired with another individual, which will almost certainly be different from the first.
The expected number of breaks with increased family size can be calculated since increasing family size reduces effective populations size by a constant amount[4] as shown in equation 3.

$$N_e = \frac{4N}{(\mathrm{var}(g) + \bar{g}^2 - \bar{g})}$$

Equation 3. g is number of gametes contributed by each parent and will be two for a diploid organism in a stable population[4].

In a randomly mating population with mean offspring per pair of parents = 1, each individual will participate in a mean of two distinct pairings and will contribute one gamete to each of two children. The number of offspring per individual will have a Poisson distribution and $\mathrm{var}(g) = \bar{g} = 2$ and $N_e = N$.
However in a constant size population if pairings are stable and produce offspring at random then more individuals will have no offspring and those that do will have more offspring leading to an increase in the variance of g. The variance of g under these conditions is $2f$ where f is family size but g will remain constant at 2 if population size is constant.

$$\mathrm{var}(g)_f = \sum_{i=0}^{i=\infty}\left(\frac{1}{f}p(i)(\bar{g}-i)^2\right) + (\bar{g}-0)^2(1-\frac{1}{f}) = 2f \qquad \text{Equation 3.1}$$

Equation 3.1 Variance of gametes (g) for a given family size (f) where $p(i)$ is the Poisson probability of i given an expected of 1. $\bar{g} = 2$ in a stable population. The summation on the left gives the variance of the number of gametes from individual parents. The right hand term corrects for those individual who do not

become parents and do not contribute gametes, (hence the 0 in the first parentheses of the right hand term).

$$N_e = \frac{4N}{(2f + \bar{g}^2 - \bar{g})}$$ Equation 3.2

By using equations 3.2 to correct N_e for family size, we can predict the mean number of breaks in a randomly mating stable population with discrete generations with any mean family size.

Selfing

In mapping experiments using inbred lines the F0 animals are set up to only permit crosses between different haplotypes. In subsequent generations mating is random. If selfing in the first generation is permitted (selfing = 1) then mating is completely randomized. If it is not permitted then pairs of haplotypes will be selected at random until different haplotypes are selected. If the numbers of each haplotype set by the user are not all equal then it is essential that selfing is permitted since the effect of prohibiting selfing is to equalize haplotype frequencies. Consider the case of 2 haplotypes with frequencies set by the user at 0.75 and 0.25 then if selfing is prohibited the frequencies in the actual recombinations will be 0.5. With more than 2 founder haplotypes the problem is less extreme depending on frequencies but will nevertheless cause observed and expected to diverge.

HotSpot Interval

The interval between recombination hot spots in base pairs. Minimum 0 maximum 50,000,000. If setting either the Chromosome length or the HotSpot Interval to a value such that c/h <= 10 (where c is chromosome length and h is hotspot interval) then the new value will be rejected and the previous value will be retained.

Telomere Length

The length of the region at the ends of chromosomes where crossovers are prohibited. Minimum 1; maximum10,000,000.

Interference Interval

The minimum permitted distance between adjacent crossovers at a meiosis. Minimum 0; maximum 100,000,000. In the GUI the interference interval cannot be more than 25% of chromosome length in order to make it possible to have up to three crossovers in a meiosis.

Output Interval

The number of generations between outputting summary statistics. Increasing the value will reduce clutter in the output and can dramatically increase speed of execution. A simulation of an intercross with a population size of 100 over 1000 generations took 11s with Output Interval of 10 generations and 60s with an Output Interval of 1.

Number of parallel processes

The number of threads that will be used for the simulation Minimum 1; maximum 4. Additional threads are used for the GUI and other purposes. Performance testing shows that increasing the number of threads beyond four makes negligible difference to run time.

Number of replicates

The number of replicate simulations to run with the same parameters: Minimum 1; maximum 100. Programme output will be the means and standard deviations of the replicate runs. Run time scales linearly with number of replicates. Increasing number of replicates has only a small effect on memory requirements.

Errors and Logging

Error messages from the application are written to SimulatorLog.txt. If problems are experienced please send this file to harry at liv.ac.uk together with the parameters used.
A log of configuration parameters and times for each replicate is maintained in HaplotypeSimulatorLog.txt.

Output

On completion four sets of plots are available:
Breaks Output shows a table and two plots.
A table of the mean number of breaks (junctions) and mean haplotype length at each generation.
A plot of the mean and expected number of junctions and the variance of the number of junctions at each generation.
A plot of the observed and expected distribution of the number of junctions in the final generation. Two expected distributions are presented: a) using the expected mean obtained using Equation 1; b) using the observed mean of the simulation.
Haplotype Output shows to column plots
A column plot of the mean haplotype length in the final generation for each founder haplotype with error bars showing standard deviation
A column plot of the frequency of each haplotype in the final generation with error bars showing standard deviation.
Haplotypes Distribution shows a summary column plot of the length distribution for all haplotypes and individual plots for each haplotype in the final generation. Each plot also shows the expected distribution of haplotype lengths assuming an exponential distribution.
Haplotype Lengths shows the mean and standard deviation of haplotype length in each generation.

Algorithm for Distribution of Probabilities of Recombination

The distribution of probabilities of a recombination between different haplotypes was found using a stand-alone command line programme HapFreqs.jar.

If there are h founder haplotypes at equal frequencies instead of just two, then the probability (p) of a given haplotype recombining with a different haplotype is (1-1/h). If frequencies of founder haplotypes are not all equal then the frequency of recombination of haplotype h is f_h, the frequency of that haplotype, and the frequency of recombination of that haplotype with haplotypes of a different class is ($1 - f_h$). Hence for a specific locus with h haplotypes in a specific population, the probability of haplotypes recombining with different haplotypes is:

$$p = \sum_{i=1}^{i=h} (1 - f_i) f_i$$

Equation 2. The probability in a population of the two haplotypes at a breakpoint being different. f_i is the frequency of the i^{th} haplotype

In order to find the distribution of probabilities of a recombination between different haplotypes in all populations or across all loci then in principle it is necessary to find all possible combinations of haplotype frequencies. However the order of frequencies is not meaningful and makes no difference to the population probability of a recombination between different haplotypes. Therefore the problem can be reduced to two subsidiary problems:

1) Identification of all unique unordered sets of h frequencies that sum to 1
2) Identification of the number of possible combinations of the frequencies in each set.

In order to make the calculations tractable only discrete frequencies in the range 0.1-0.99 were evaluated. Frequencies were converted to percentages and the search was conducted for sets of h percentages that summed to 100.

The number of unique ordered combinations of h positive integers that sum to 100

$$\frac{99!}{(h-1)!(100-h)!}$$

Equation 4.1. The number of unique combinations of h numbers between 1 and 99

The number of unique ordered combinations of a given set of n numbers is:

$$\frac{n!}{\prod_{i=1}^{i=x} a_i!}$$

Equation 4.2. The number of unique combinations of n numbers where numbers can be used more than once. a_1 to a_x are the numbers of occurrences of numbers 1 to x

Therefore it is only necessary to identify the unique sets and not all possible combinations to obtain a complete set of possible population probabilities of a recombination. An algorithm for efficiently identifying all such sets was identified by first conducting an exhaustive search for all such subsets for 4 haplotypes (Table 1). It was observed that:

4 numbers
1,31,32,36
1,31,33,35
1,31,34,34
1,32,32,35

1,32,33,34
1,33,33,33
2,02,02,94

Table 1. Part of observed sequence of unique sets of 4 numbers

1) The first set of 4 numbers that summed to 100 was 1,1,1,97 and this was the initial state

```
while( n4 > n3 +1){
        n4 --
        n3++
        i = 2
        while(ni  >=  ni+1 && i > 0){
                i--;
        }
        ni  ++
increment(i){
        i++
while(i < 4){
        ni  = ni-1
        i++
}
```

$$n_4 = 100 - \sum_{i=1}^{i=3} n_i$$

```
if(n4 > n2){
        next unique set identified
}
else{
        i = 2;
while(ni >= ni +1){
        i—
}
increment(i)
}
```

When a unique set of numbers was found the number of combinations of that set was obtained using Equation 4.2. The algorithm was repeated until the number of unique combinations equaled the global total found with equation 4.1. This end point occurred when all haplotype frequencies were equal.

The algorithm was implemented in a Java programme HapFreqs.jar, which is available from GitHub together with the source code. The programme also calculated the mean and mode probability of a recombination across all possible frequency sets and generated data for a histogram of the probability distribution.

It was noted that a close approximation ($r^2 = 0.997$) to the most likely set of frequencies was obtained using the Geometric distribution (equation 4.3) (Table 2).

$$\frac{\sum_{i=1}^{i=h} p_i(1-p_i)}{\sum_{i=0}^{i=h-1} p_i}$$

Equation 4.3 Most probable probability of a recombination where h is number of haplotypes and p_i is the Geometric probability mass function $(1/h)(1\text{-}1/h)^{i-1}$

Calculated	Observed
0.494	0.510
0.609	0.610
0.682	0.670
0.732	0.720
0.769	0.760
0.797	0.790
0.818	0.810

Table 2 comparison of the most likely probability of a recombination obtained by calculation using equation 4.3 and observed from the algorithm.

1. Broman, K.W., Rowe, L.B., Churchill, G.A. & Paigen, K. *Genetics* **160**, 1123-31 (2002).
2. Object Refinery Limited. JFreeChart: The official home page of the JFreeChart project. Vol. 2011 (2009).
3. Jensen-Seaman, M. *Genome Research* **14**, 528-538 (2004).
4. Crow, J. & Denniston, C. *Evolution* **42**, 482-495 (1988).