Supplementary material for "An individual-based model simulator for transposable elements dynamics and evolution"

Felipe Figueiredo* Claudio Struchiner

March 2, 2011

Contents

1 Methods and models

The simulator is flexible in construction and able to produce a realistic scenario in which transposition events occur in finite populations that agree with the Wright-Fisher model [1]. Transposable elements

*ffigueiredo@ioc.fiocruz.br

sequences evolve according to Kimura's infinite-sites model [2], and an arbitrary model of molecular evolution [3].

1.1 Population

The population has two independent structures: gender and age. We draw inspiration from the SIR model to structure the population in compartments. The age structure is also discrete, as described below.

The population is divided into **Wild** and **GM** (genetically modified) compartments of hosts similarly to the SIR compartmental model classes *Susceptible* and *Infected*. There's also a **Deceased** compartment, in analogy to the *Recovered* compartment, where individuals from earlier generations are kept for future reference. Differently from the SIR model a given individual does not change from one compartment to another. Instead, the compartments' dynamics evolve through generations of sexual reproduction of the host population.

The age structure is defined by two classes, regarding wether or not a host is mature enough to reproduce. Each new individual remains non-reproductive for a maturation period, after which it becomes mature and enter the reproductive pool in the population.

Age of sexual maturity and maximum lifespan are passed to the simulator as configuration parameters by the user. Default parameters for maturation and lifespan reflect cage insect populations and thus were chosen to avoid generations overlap.

Default age parameters	
Initial age	**0**
Reproductive age	**3**
Maximum age	**4**

The growth rate of the population is determined at the beginning of each generation from an arbitrary ecological model (see below), and from this rate the net income of new individuals for the next generation is obtained. The necessary amount of reproductive encounters to reach the appropriate number of new individuals for the next generation is then determined by dividing the total income by the expected number of offspring per couple as indicated in equation (1). Note that assuming a non-monogamic species, this is equal to the number of females expected to mate in this generation, as there should always be enough males to impregnate any available fertile females.

$$\text{reproductive encounters} = \frac{\text{total income of new hosts}}{\text{expected number of offspring}} \tag{1}$$

Couples are then formed from the available mature population to satisfy the necessary number of reproductive encounters. We are modelling a host species that is not monogamous and strongly r-selected (mosquitoes), so males are chosen from the population with replacement and females are chosen without replacement.

The host fitness is the total offspring count that host has generated. The mean fitness over the population for that generation is defined as the arithmetic mean of the fitnesses of all live individuals at the end of the generation.

The offspring amount for each reproductive encounter should be chosen randomly around the mean fixated for the population. From this number it should be subtracted a quantity proportional to the impact TEs might have on the host reproductivity. For simplicity, however, we consider in this simulation a fixed initial amount of offspring for each couple, from which the fitness impact is subtracted. We postpone the definition of fitness impact ~~is~~ and how it is calculated to the transposition model section.

Migration can be introduced in the simulator by the use of several geographic patches. Each of these patches is a micro-environment of its own, with the same compartment structure. For simplicity, in this simulation, we consider only one geographic patch.

1.1.1 Ecological Model

The simulator is modular in the sense that the ecological and transposition models are implemented and considered independently from the rest of the system. This way, many models can be included in the simulator in order to suit particular demands for each of these sections. Indeed, any population model can be used in this system provided it can be represented in the form:

$$p_{n+1} = F(p_n) \tag{2}$$

where F is a function that depends on the current population size p_n. It is not required that F be deterministic, i.e., depends only on p_n. It can optionally depend on a random variable ξ_n to introduce stochasticity in the growth rate:

$$p_{n+1} = F(p_n, \xi_n) \tag{3}$$

As stated above, it is not required that the model can be written explicitly as a differential or difference equation. The only requisite is that the input is the current population size, and the output is the expected quantity for the next generation.

We implemented two population models of interest in the simulator: the logistic equation

$$\dot{p} = rp\left(1 - \frac{p}{K}\right) \tag{4}$$

and the Hassel equation [4]

$$\dot{p} = rp\left(1 + \frac{p}{K}\right)^{-\beta} \tag{5}$$

where $r, K, \beta > 0$. The parameter r is the reproductive ratio, K is the support capacity of the population and β is the intra-specific competition exponent. For $\beta = 1$, this model is identical to the logistic model.

1.1.2 Age structure

The population has an age structure that can be interpreted in two discrete categories. The behavior of such models has been thoroughly studied [5], and

There are two age classes of obvious interest: before and after reaching the reproductive age.

Leslie matrix 2×2

1.2 Genomic model

1.2.1 The genome

The genome is composed by two chromosomes, and each chromosome is represented discretely as a list of insertion sites that act as *loci* for TEs. Each insertion site can therefore be empty or occupied by a TE. To reflect the fact that some insertions can cause deleterious or fatal mutations on the host, we categorize the insertion sites into three classes: fatal (*killer sites*), severe (*severe sites*) and neutral (*neutral sites*). Under the hypothesis that within a species genomes from different hosts should bear more similarities than differences, the number of insertion sites in each of the above classes should not vary much within a species.

We consider in this model that the total number of sites in each of the three categories is species-determinant. Each chromosome has $\eta = k + s + n$ *loci*, each of which can contain one or zero TEs. We can reorder these sites, without loss of generality, in such a way that the first k sites are *killer sites*, meaning they disrupt essential genes in a way they render the cell useless; the next s are *severe sites*, which may disrupt non-essential genes, or metabolic pathways, and the remaining n are of *neutral sites*.

1.2.2 Recombination

A model of crossing-over recombination is optionally used to promote additional variability of chromosome content. It is implemented in three variants called *1-step*, *2-step* and *all-step*. The resulting gamete retains the original size in insertion sites, and each site is filled consulting one of the parent's corresponding site from one of the parental gamete.

In the *1-step* model a site is chosen at random and this site is the cutting point between the parts coming from each chromosome. The resulting gamete will be filled up to this site identical to the first chromosome and then the remaining sites are merged from the second chromosome.

The *2-step* model is similar to the above, but there are two cutting points instead of one. This means the first and third sections of the gamete are merged from the first chromosome and the second is merged from the other chromosome.

There's also an *all-step* model that abstracts from the two models above in the sense that every site is a cutting point. This way every site is taken from a random chromosome preserving only the order from which it came. This will provide maximum variability considering recombination events.

If none of the above recombination models is selected, the resulting gamete will be equal to one of the parental chromosomes. Hence one of the parental chromosomes is chosen at random.

1.2.3 The evolutionary model

Over time mutations occur and are accumulated in the genome of simulated hosts. Whenever a mutation occurs a substitution drawn from an evolutionary model. The evolutionary model is any model or function that, given an input nucleotide sequence, outputs a similar homologous sequence.

We implemented a simple model that follow the assumptions of the Jukes-Cantor substitution model [6]. All mutations are substitutions and equally likely to occur. As a result no special assumption is made over sites in the sequence which are more likely to change, nor transition/transversion bias. *All sites are considered equally likely to change as well as all nucleotides changes are considered equally likely to occur.*

As happens with the ecological section, almost any evolutionary model can be implemented in the simulator. The requirement here is that the input is the original sequence, and the output is the mutated sequence. How many changes, or which changes are allowed are completely up to the model in question, which provides

1.3 Transposition model

Several transposition models can be implemented in the system. We implemented both neutral selection exponential model [7], and a transposition model that takes into account deleterious effects by transposition [8].

1.3.1 Neutral model

Here we describe the simplest transposition model we implemented in the simulator. This model is thoroughly described in [7]. It doesn't take into account any regulation due to selection on the host population, so the only regulation imposed on the TE family, besides self-regulation, is the excision of TE copies.

This model is basically the combination of two exponential models, one for the the acquisition of new copies and one for

$$\Delta c_n = (u_n - e)c_n \tag{6}$$

where c_n is the TE count at time n, u_n is a decreasing positive function of n and $e > 0$. According to this model, the TE can invade the host population provided u_n is larger than e for small n [7], and can fixate if u_n never decreases below e. Furthermore, TE copy number reaches an equilibrium if u_n converges to e for large n.

1.3.2 Activity cycle

The initial phase of the TE invasion, should be regarded as the period during which the TE is most active. Otherwise, genetic drift may lead the TE to extinction [9]. After this initial phase, if the TE succeeds in fixating in the host population, it must decrease its activity so as not to disrupt too much the host's fertility.

Transposable elements also have some degeneration process (FIXME: CITE), which we take into account in the model in the form of an aging score, that quickly drops to zero as transposition events occur. When it reaches zero, it renders the TE inactive. This *status score* is also used to determine whether of not the TE is actively transposing.

Hosts can be born dead due to deleterious transposition, and otherwise have lower offspring count.

Lifespan can also be decreased if the impact is too great, albeit not fatal (as proposed in [7]). We do not include this characteristic in the model.

1.4 The population genealogy

We are interested in analyzing sequence data from an individual to reconstruct the phylogenetic relations between its TEs. Therefore it becomes necessary to compare the reconstructed history with the real parental history we simulated.

Each host is identified by a name defined as a serial number, and carries the names of both parents. A simple recursive breadth-first algorithm is used to recover the names of parents, grandparents and so on. As such the genealogy can be fully reconstructed up to any valid time interval. The population stored in the **Deceased** compartment can be consulted to create genealogies of different depths after the execution of the simulation, or inspect each individual for it's genome, in order to compare the elements present in the sampled individual, and its ancestors.

1.5 Sources of stochasticity

Although some features of the simulator involve deterministic models the overall behavior is stochastic. This happens due to both sampling effects and the fact that the system is an individual based model, or agent based model [10]. The uncertainty related can be estimated with straightforward statistics techniques. The following are sources of such uncertainty:

1. sampling of individuals in finite population
2. recombination
3. replication and excision of TEs
4. mutations (substitutions) within TE sequences

The item 1 is related to Wright-Fisher models [1]. The item 4 is related to the evolutionary model used to promote variability across generations. The item 2 is related to the recombination model used to promote additional variability to the chromosome pool. The item 3 is related to the availability of new TEs in the chromosomes (as defined by different *loci* and sequence).

2 Implementation details

Here we describe the overall structure of the simulator. More specific details are available in the API documentation, provided with the software.

The simulator is an Object-Oriented system, with individual hosts and TEs each represented by objects. It's main interface with the user is provided by an ASCII configuration file that must be present in directory of the simulation. The simulation sub-directory tree will be created, and data and log files written by the simulator as required.

2.1 Population models

2.1.1 Implementation of the logistic model

We will use the logistic equation as an example to describe the implementation because of its simplicity; the implementation Hassel for the Hassel equation is analogous.

This differential equation is discretized as a single iteration of Euler's method, with a step size $h = 1$. This results in the following formula:

$$p_{n+1} - p_n = \Delta p = Ceiling\left(\frac{r \times p \times \left(1 - \frac{p}{K}\right)}{\text{offspring count}}\right) \tag{7}$$

where $Ceiling(x)$ ($Floor(x)$) is the smallest (greatest) integer greater (less) than or equal to x. Note that this implementation already takes into account the normalization by the total offspring count as per equation (1), so the result is the number of couples, instead of the total income of new individuals.

This model is deterministic so each input value always renders the same output value. All values calculated for deterministic models are cached to avoid the performance toll of unnecessary repeated calculations. This cache is implemented as a hash[1], and is consulted beforehand in each iteration.

2.2 Transposition models

2.2.1 SKR model

$$c_{t+1} = Ceiling(c_t + c_t \times T_0 \times U(c_t)) \tag{8}$$

$$U_1(c) = 2^{(-c/C_{0.5})} \tag{9}$$

$$U_2(c) = 1 - \frac{c^5}{(C_{0.5}^5 + c^5)} \tag{10}$$

$$U_3(c) = 1 + (c - \frac{0.5}{C_{0.5}}) \tag{11}$$

2.3 Availability

The simulator has been tested in GNU/Linux systems running Ubuntu Linux. Being t should run on any system for which its dependencies are available. Those include MacOS X and Windows

2.4 Data structures

The model is implemented in Perl5, using its object-oriented paradigm. The main classes define objects for the population, the hosts that constitute the population and the TEs that populate each host genome.

Two classes inherit methods from Bioperl classes [11]. The TE class inherits the `Bio::Seq` class, and the Host class inherits the `Bio::SeqIO` class. This guarantees the simulator can import and export sequences using a great range of sequence formats. It also uses the abstract model of representing DNA sequences provided by the BioPerl project in a way that unifies the usage of any sequence format (Fasta, genbank, embl, swissprot, etc) so that any of these formats can be used at the discretion of the user. We provide a wrapper for including header information in the Fasta format, which we describe in detail in a later section.

The documentation of the API is bundled in the code with the POD (Plain Old Documentation) format, which can be accessed by the ordinary means of any Perl distribution. The

[1] sometimes referred to as an associative array

2.4.1 Host data structure

The structure inherited from `Bio::SeqIO` provide methods to collect and save sequences from a file, which inspired an implementation for a file-based storage. Besides these general-purposed methods we defined some object attributes to store meta-data related to each individual like its name, age, gender, fitness, the two chromosomes the file name to which the `Host` object is associated.

Each of the two chromosomes is an array, and each of its positions stores a reference for a TE object if one is present. Methods for consulting the number or position of TEs and de-reference them are provided in the class API.

2.4.2 Population data structure

The object attributes for the `Population` class store information related to where the files are stored in disk, and some statistics that are frequently consulted.

2.4.3 TE data strucuture

Besides the structure inherited from `Bio::Seq`, which includes methods for setting and retrieving several meta-data as the sequence string, header information, etc.

2.4.4 Meta-data format for Fasta sequences

We default to using sequences in Fasta format. To this end, we propose a protocol for inserting meta-data into the Fasta header and a set of corresponding parser (writer) methods in order to retrieve (save) the corresponding information.

```
>string:int:int:real
```

The first field (string) denotes the **TE description**. It can vary in size and can contain spaces. Usually it comprises the whole Fasta header if a real sequence is used, as acquired from any genomic database, should one such sequence be used.

The next two fields that contain integer numbers. The first is the **site** position in the chromosome, and the second defines in which **chromosome** this particular TE is located. The total number of sites available can be chosen by the user, and the default is 1000 positions (from 0 to 999). There are always two chromosomes (denoted by the numbers 0 and 1).

The last field is a real number, in the interval $[0, 1]$. It describes the **status score** of activity for that TE, where any non-zero value means the TE is active. Each new copy generated from this copy should have a lower score, effectively reproducing a deactivation function for the TE family.

References

[1] Daniel L. Hartl and Andrew G. Clark. *Principles of population genetics*. Sinauer Associates, 3 edition, January 1998.

[2] Fumio Tajima. Infinite-allele model and infinite-site model in population genetics. *Journal of Genetics*, 75(1):27–31, 1996.

[3] Ziheng Yang. *Computational Molecular Evolution (Oxford Series in Ecology and Evolution)*. Oxford University Press, USA, December 2006.

[4] M. P. Hassell. Density-dependence in single-species populations. *Journal of Animal Ecology*, 44(1):pp. 283–295, 1975.

[5] R. M. Nisbet and W. S. C. Gurney. *Modelling fluctuating populations*. John Wiley, Chichester. US, 1982.

[6] T. H. Jukes and C. R. Cantor. *Evolution of protein molecules*. Academy Press, 1969.

[7] A. Le Rouzic and G. Deceliere. Models of the population genetics of transposable elements. *Genet Res*, 85(3):171–81, 2005.

[8] Claudio J Struchiner, Margaret G Kidwell, and Jose M C Ribeiro. Population Dynamics of Transposable Elements: Copy Number Regulation and Species Invasion Requirements. *Journal of Biological Systems*, 13(4):455–475, Dec 2005.

[9] A. Le Rouzic and P. Capy. The first steps of transposable elements invasion: parasitic strategy vs. genetic drift. *Genetics*, 169(2):1033–43, 2005.

[10] Eric Bonabeau. Agent-based modeling: Methods and techniques for simulating human systems. *Proceedings of the National Academy of Sciences of the United States of America*, 99(Suppl 3):7280–7287, 2002.

[11] J. E. Stajich, D. Block, K. Boulez, S. E. Brenner, S. A. Chervitz, C. Dagdigian, G. Fuellen, J. G. Gilbert, I. Korf, H. Lapp, H. Lehvaslaiho, C. Matsalla, C. J. Mungall, B. I. Osborne, M. R. Pocock, P. Schattner, M. Senger, L. D. Stein, E. Stupka, M. D. Wilkinson, and E. Birney. The Bioperl toolkit: Perl modules for the life sciences. *Genome Res*, 12(10):1611–8, 2002.

A API description

The following sections document the API of the simulator, which is available for scripting arbitrary simulations in the Perl language. The main classes are described here in this document, and the classes of ...