BIOINFORMATICS APPLICATIONS NOTE Vol. 21 no. 18 2005, pages 3686–3687
doi:10.1093/bioinformatics/bti584

Genetics and population analysis

simuPOP: a forward-time population genetics simulation environment

Bo Peng* and Marek Kimmel

Department of Statistics, Rice University, 6100 Main Street, MS138, Houston, TX 77005, USA

Received on May 13, 2005; revised on July 8, 2005; accepted on July 13, 2005
Advance Access publication July 14, 2005

ABSTRACT

Summary: simuPOP is a forward-time population genetics simulation environment. The core of simuPOP is a scripting language (Python) that provides a large number of objects and functions to manipulate populations, and a mechanism to evolve populations forward in time. Using this R/Splus-like environment, users can create, manipulate and evolve populations interactively, or write a script and run it as a batch file. Owing to its flexible and extensible design, simuPOP can simulate large and complex evolutionary processes with ease. At a more user-friendly level, simuPOP provides an increasing number of built-in scripts that perform simulations ranging from implementation of basic population genetics models to generating datasets under complex evolutionary scenarios.

Availability: simuPOP is freely available at http://simupop.sourceforge.net, distributed under GPL license.

Contact: bpeng@rice.edu

INTRODUCTION

Coalescent-based (Kingman, 1982) methods have dominated the area of genetic dataset generation because of their efficiency and flexibility. In contrast, forward-time simulations, although simpler as an idea, are computationally inefficient and have been used primarily for teaching purposes. Only recently, owing to the exponential growth of the power of personal computers, did the use of serious simulation programs such as easyPOP (Balloux, 2001) and FPG (Hey, 2004, http://lifesci.rutgers.edu/heylab/HeylabSoftware.htm.) begin in genetic studies (Balloux and Goudet, 2002).

Unlike coalescent-based approaches, forward-time simulations keep track of complete ancestral information. This gives forward simulations a wider application area if evolutionary processes themselves rather than their outcome are of interest (Calafell *et al*., 2001) or if population-level properties are studied (Balloux and Goudet, 2002). Forward-time simulations are also more flexible in the sense that any genetic or environmental factors can be applied to a forward-evolving population, while coalescent simulations (e.g. SIMCOAL, http://cmpg.unibe.ch/software/simcoal/) are still complicated for simple genetic forces such as selection (Fearnhead, 2003). Therefore, selection as well as complex mating schemes and recombination are the main strengths of simuPOP.

It is relatively easy to implement a special purpose forward-time simulation program. On the other hand, a one-for-all simulation program is difficult to design due to its wide application area. For example, neither easyPOP nor FPG can handle demographic changes and customized genetic forces, and are difficult to use. To solve this problem, a new approach has been introduced.

*To whom correspondence should be addressed.

FEATURES AND BASIC USAGE

simuPOP is a forward-time population genetics simulation environment based on Python, an ‘interpreted, interactive, object-oriented and extensible’ language. simuPOP consists of a large number of Python objects and functions, including population, mating schemes, operators (objects that manipulate populations) and simulators to coordinate the evolutionary processes. It is the users’ responsibility to write a Python script to glue these pieces together and form a simulation. simuPOP distinguishes itself from other programs in the following respects:

Scripting. simuPOP is provided as a set of Python libraries, and is therefore backed by a full-blown object-oriented programming language. All key elements of simuPOP are objects with their own data elements and member functions. Users can run a simulation interactively using a Python shell or write an arbitrarily complex Python script and run it as a batch file.

Flexibility. simuPOP does not impose any limit on the size of genome, population, ploidy number, demographic model, mating type, etc. Using a large number of standard and hybrid (Python-assisted) operators, plus the ability to extend simuPOP in Python, users can simulate almost arbitrarily complex evolutionary processes.

Integration. Owing to the ‘glue language’ nature of Python, it is easy to integrate simuPOP with other languages and programs. For example, users can call any R function from Python/simuPOP for the purposes of visualization and statistical analysis, using R and a Python module RPy. This feature is shown in the following simuPOP session:

```
>>> from simuPOP import *
>>> from simuRPy import *
>>> simu=simulator(
...   population(size=1000,loci=[2]),
...   randomMating(),rep=3 )
>>> simu.evolve(
...   preOps=[initByValue([1,2,2,1])],
...   ops=[
...     recombinator(rate=0.1),
...     stat(LD=[0,1]),
...     varPlotter("LD[0][1]",numRep=3,
...       ylim=[0,.25],xlab="generation",
...       ylab="D",title="LD Decay")],
...   end=100 )
```

The first two lines import simuPOP and simuRPy modules. `simuPOP.py` is the standard simuPOP module. `simuRPy.py` provides R-related operators like `varPlotter`. The third command creates a simulator with three replicates of a diploid population with 1000 individuals, each having one chromosome with two loci. Random mating will be used to generate offspring. The last command uses the `evolve` function to evolve the populations for 100 generations, subject to four operators.

The first operator `initByValue` is applied to all populations before evolution. It initializes all individuals with the same genotype `12/21`. The other three operators will be applied at every generation. `recombinator` will recombine parental chromosomes with the given recombination rate during the generation of offspring; `stat` will calculate standard linkage disequilibrium between the first and second loci. The result of this operator will be stored in a local variable space of each population and be retrieved and plotted by `varPlotter`, which uses R for plotting. When `evolve` is called, a graphics window will be fired and will display the dynamics of LD values for all three replicates.

DISCUSSION

simuPOP is large, consisting of >70 operators and a lot more functions that cover all important aspects of genetic studies. These include mutation (k-allele, stepwise, generalized stepwise and hybrid), migration (arbitrary, can create new subpopulation), recombination (uniform or nonuniform), quantitative trait (single, multi-locus or hybrid), selection (implemented as relative probability of mating, single-locus, additive, multiplicative or hybrid multi-locus models), penetrance (single, multi-locus or hybrid), ascertainment (case–control, affected sibpairs or random), statistics calculation (including but not limited to allele, genotype, haplotype, heterozygote number and frequency; expected heterozygosity; bi-allelic and multi-allelic D, D' and r^2 linkage disequilibrium measures; F_{st}, F_{it} and F_{is}), pedigree tracing, visualization (using R or other Python modules) and load/save in text, XML, Fstat or Linkage format. Each of these operators accepts a number of parameters that allow it to be applied at any given stage of a life-cycle, at any generation(s) and so on. The mating schemes are also flexible in that arbitrary demographic models (patterns of population size changes) can be specified and the number of offspring per mating event can be constant or follow a random distribution. Although simuPOP currently focuses on random mating in non-overlapping generations, age-structured populations and cooresponding mating schemes are under development and will allow simulations of overlapping generations and continuous-time reproduction. The detailed descriptions can be found in the simuPOP user's guide and reference manual, along with several real examples.

Simulation of real evolutionary processes is not always easy. Parameters like mutation or migration rate are hard to determine and

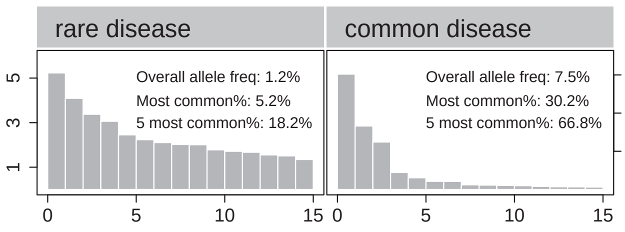

Fig. 1. Simulations testing the common disease common variant hypothesis (Reich and Lander, 2001; Peng and Kimmel, 2005). Allellic spectra of a rare and a common disease 500 generations after rapid linear population expansion from 10^4 to 10^6 x-axis: 15 most common disease alleles. y-axis: percentage among disease alleles.

initialization is surprisingly difficult. This is because many models evolve from a certain (e.g. linkage or mutation-drift) equilibrium state that cannot be initialized easily. A common approach is to use a burn-in period to allow a population to reach equilibrium from a random or uniform initial state.

The mere idea of having to write a program may drive most time-limited researchers away. In recognition of this, simuPOP is bundled with an increasing number of scripts that can be executed without knowing the underlying language. These scripts (under scripts directory) are equipped with graphical user interfaces and perform simulations ranging from implementation of simple population genetics models in standard textbooks to generating datasets of a complex disease under complicated evolutionary scenarios. More scripts will be added, hopefully with user contributions, to make simuPOP suitable for a wider audience.

ACKNOWLEDGEMENTS

The authors thank Dr. François Balloux, author of easyPOP (Balloux, 2001), for his many suggestions and Dr. William Amos for his insights about the userfriendliness of simuPOP. Many thanks should go to the SWIG and the Python user community. Their prompt responses to the author's numerous emails were essential to the making of simuPOP. This work was supported in part by a training fellowship from the W.M. Keck Foundation to the Gulf Coast Consortia through the Keck Center for Computational and Structural Biology.

Conflict of Interest: none declared.

REFERENCES

Balloux,F. (2001) EASYPOP (Version 1.7): A computer program for population genetics simulation. *J. Hered.*, **92**, 301–302.

Balloux,F. and Goudet,J. (2002) Statistical properties of population differentiation estimators under stepwise mutation in a finite island model. *Mol. Ecol.*, **11**, 771–783.

Calafell,F. *et al*. (2001) Haplotype evolution and linkage disequilibrium: a simulation study. *Hum. Hered.*, **51**, 85–96.

Fearnhead,P. (2003) Ancestral processes for non-neutral models of complex diseases. *Theoret. Popul. Biol.*, **63**, 115–130.

Hey,J. (2004) A computer program for forward population genetic simulation.

Kingman,J. (1982) The coalescent. *Stochastic Proc. Appl.*, **13**, 235–248.

Reich,D.E. and Lander,E.S. (2001) On the allelic spectrum of human disease. *Trends Genet.*, **17**, 502–510.

Peng,B. and Kimmel,M. (2005) On the allelic spectrum of human diseases, a simulation study. *In preparation*.