simuPOP: a forward-time population genetics simulation environment

Bo Peng and Marek Kimmel

Department of Statistics, Rice University

For further information, please contact *bpeng@rice.edu*. More information about simuPOP can be found at *http://simupop.sourceforge.net*. A mailing list is devoted to the discussion of simuPOP, its email address is:*simupop-list@lists.sourceforge.net*. This work has been published at *bioinformatics* (Bo and Kimmel, 2005).

Introduction

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.

Why forward-time?

Backward-time (Coalescent, Kingman, 1982) based 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 forward-time simulations begin in genetic studies.

	Backward-time (Coalescent)	Forward-time
Efficiency	**Yes**	No
Intuitiveness	No	**Yes**
Flexibility	Yes	**Excellent!**
Demographies	Yes	**Any**
Selection	No	**Any**
Recombination	Limited	**Any**
Migration	Yes	**Any**
Mutation	Yes	**Any**
Ancestral information	Discard	**Keep**
Trace the evolutionary process	No	**Yes**
Population information	No, sample based	**Yes**
Study population property	No	**Yes**
Resample from the resulting population	No	**Yes**
Test ascertainment methods	No	**Yes**
Application area	mostly around sample generation	**unlimited**
Avalability of programs	**Plenty**	EasyPOP (Balloux 2001) FPG (hey 2004) Now comes simuPOP!

What is simuPOP?

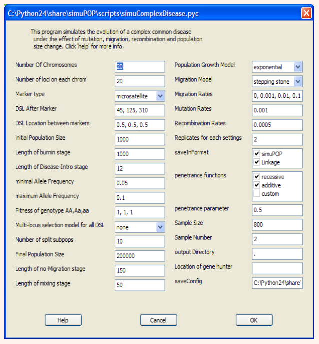
simuPOP is a forward-time population genetics simulation environment based on Python, an 'interpreted, interactive, object-oriented and extensible' language. simuPOP consists of a large number of Python objects and functions, including *population*, *mating schemes*, *operators* (objects that manipulate 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.

FEATURES

- ✓ *Scripting.* simuPOP is provided as a set of Python libraries, and is therefore backed by a full-blown object-oriented programming language. 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.
- ✓ *Comprehensiveness.* simuPOP consists of more than 70 operators and a lot more functions that cover all important aspects of genetic studies. These include mutation, migration, recombination, quantitative trait, selection, penetrance, ascertainment, statistics calculation, pedigree tracing, visualization and load/save in many popular formats.
- ✓ *Efficiency.* simuPOP is written in *C++*. With extensive optimization, simuPOP can run very large simulations at reasonable speed. A typical simulation of 200,000 individuals with 400 markers, evolving with mutation, migration, recombination etc. will take about 30min on a P4 2.8GHz PC.
- ✓ *Openness.* simuPOP is distributed free of charge under GPL license. The source code is available at simuPOP website.

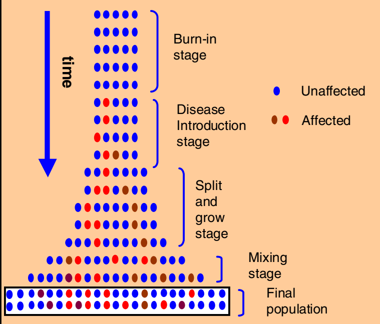
An example

The *simuComplexDisease.py* (above figure) script simulates the evolution of a complex disease under a four-stage evolutionary scenario. In a typical simulation, individuals have 20 chromosomes with 400 microsatellite markers and 5 binary disease susceptibility loci (DSL). The simulation starts with a small population and goes through four stages of evolution processes including burn-in, disease introduction, split and growth, and mixing. Recombination, mutation, selection, migration and demographic changes shape the genotype of the resulting large multi-generation populations in which samples can be drawn using different ascertainment methods.


C:\Python24\share\simuPOP\scripts\simuComplexDisease.pyc

This program simulates the evolution of a complex common disease under the effect of mutation, migration, recombination and population size change. Click 'help' for more info.

Number Of Chromosomes	20	Population Growth Model	exponential
Number of loci on each chrom	20	Migration Model	stepping stone
Marker type	microsatellite	Migration Rates	0, 0.001, 0.01, 0.1
DSL After Marker	45, 125, 310	Mutation Rates	0.001
DSL Location between markers	0.5, 0.5, 0.5	Recombination Rates	0.0005
initial Population Size	1000	Replicates for each settings	2
Length of burnin stage	1000	saveInFormat	✓ simuPOP ✓ Linkage
Length of Disease-Intro stage	12	penetrance functions	✓ recessive ✓ additive ☐ custom
minimal Allele Frequency	0.05		
maximum Allele Frequency	0.1		
Fitness of genotype AA,Aa,aa	1, 1, 1	penetrance parameter	0.5
Multi-locus selection model for all DSL	none	Sample Size	800
Number of split subpops	10	Sample Number	2
Final Population Size	200000	output Directory	.
Length of no-Migration stage	150	Location of gene hunter	
Length of mixing stage	50	saveConfig	C:\Python24\share\



An interactive Python session

```
>>> from simuPOP import *
>>> from simuRPy import *
>>> simu = simulator(
...   population(size=1000, ploidy=2,
...     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. The third command creates a simulator with three replicates of a diploid population. 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.

Acknowledgments

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. 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.

Literature cited

Balloux,F. (2001) EASYPOP (Version 1.7):Acomputer 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

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

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

Peng,B. and Kimmel,M. (2005) simuPOP: a forward-time population genetics simulation environment, *bioinformatics*, **21**(18):3686-3687.

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