mpop: C++ objects for forward population genetic simulation

Joseph Pickrell

November 18, 2008

Contents

1 Introduction

Mpop is a set of C++ objects and executables for running forward population genetic simulations of haplotypes using a Wright-Fisher model. It allows for incorporation of natural selection and arbitrary demographic histories.

2 Installation and Usage

The motivation for mpop was that one might want to simulate haplotypes through a complex demographic history with multiple populations, and add a selected mutation at some point in that history. It provides two ways for doing this: first, and most simply, is a standalone program (`mpop`) that reads in a set of haplotypes, simulates them forwards in time in a constant population size according to some parameters, and outputs them. By stringing sets of calls to this command together, arbitrarily complex demographies can be produced.

Alternatively, the objects themselves can be employed by the user to create their own custom demography in a single program. This is much more flexible; some examples will be given later.

2.1 Compilation

Mpop requires both the c++ boost and gsl libraries. Also, and this is important, all examples assume that the user can initialize the simulations with a neutral population generated using either ms or cosi. These programs are available (as of the time of this writing) at `http://home.uchicago.edu/~rhudson1/` and `http://www.broad.mit.edu/~sfs/cosi/`, respectively. Once these dependencies are installed, extract the source to a directory with `tar -xvf mpop.tar`. Then enter the directory `src` and check the Makefile to ensure that it's properly configured for your system. Finally, compile with `make mpop`.

2.2 Input and output format

The input format for mpop is either ms format (see link to ms above) or a modified ms format, which is also the output format. An example of the modified format follows (from the file `example/mpop.example`):

```
mpop 2000 0.5 startfreq:  0.0005 mu:  0.0001 r:  0.0001 s:  0.01 h:  0
1227029722

//
derived_sites:  31
positions:  0.0007 0.0754 0.1187 0.121 0.1293 0.1721 0.1816 0.1965 0.1974 0.2193
0.2249 0.2684 0.4269 0.4308 0.4512 0.5 0.519 0.5566 0.5766 0.583 0.6583 0.8021
0.8115 0.8329 0.8618 0.8919 0.8953 0.8964 0.9356 0.9427 0.98
00000100000000000000000000000101
...
```

In words, the first line contains the relevant parameters– the name of the program (mpop), the number of haplotypes, the position of the allele under selection, the initial frequency of the selected allele, the mutation rate, recombination rate, selection coefficient, and dominance coefficient. The second line contains the seed. The fifth line contains the number of derived sites (NOTE: this is different from ms, which only contains segregating sites. Mpop stores the positions of fixed sites to allow for comparisons of F_{ST}). The sixth line contains the positions of the derived sites, and the seventh line on are the haplotypes–derived alleles are coded as 1, and ancestral alleles as 0.

2.3 Running mpop

In this section, the command line arguments to mpop are described. It is assumed that the user has initialized a population in ms format called `ms.example`.

`-i <filename>` gives the name of the input file (in ms or mpop format).
`-o <filename>` gives the name of the output file.
`-m <double>` gives the mutation rate. So to add a mutation rate of 0.001, use the argument

`-m 0.001`.
`-r <double>` gives the recombination rate. So to add a recombination rate of 0.001 use the argument `-r 0.001`.
`-S` is a flag that tells the program to add a new selected site. NOTE: only one selected site is allowed per population.
`-s <double>` and `-h <double>` give the selection coefficient and dominance parameter (the parameterization used here is that the fitnesses of the three genotypes are 1, 1+hs, and 1+s). So to set the selection coefficient to 0.01 and make the effect additive, use `-s 0.01 -h 0.5`.
`-N <int>` changes the population size. So to change the population size to 100, use `-N 100`.
`-O <double>` adds selection on an old mutation with some frequency. So to add a selected site on a standing polymorphism of frequency 0.1, use `-O 0.1`. NOTE: this has not been carefully tested.
`-g <int>` gives the number of generations to simulate.

Note that when a selected site is added, the population is simulation *conditional on the selected site not being lost*. Putting it all together, a sample command line might look something like this:

```
mpop -i ms.example -o new.example -N 100 -m 0.00001 -r 0.00001 -S -s 0.01 -h 0.5
-g 100
```

2.4 Constructing complex demographies

Mpop commands like the one above can be used together to construct complex demographies. For example, the following commands would take an input ancestral population and add a selected site, have the population split in two populations of different sizes, and include and instantaneous bottleneck in one of the two:

```
mpop -i ms.example -o tmp.pop -N 100 -S -m 0.0001 -r 0.0001 -s 0.01 -h 0.5 -g
100
mpop -i tmp.pop -o pop1.out -N 500 -g 40
```

```
mpop -i tmp.pop -o pop2.bottleneck -N 50 -g 0
mpop -i pop2.bottleneck -o pop2.out -N 100 -g 40
```

Note that once the parameters have been defined, they do not need to be included in every command line. Only use command line arguments if you need to change a parameter. To perform multiple simulations, use a script (eg. in python or perl) to iterate through these commands.

3 Computational considerations

Forward-time simulations are necessarily more computationally intensive than coalescent simulations, and require representation of the entire population throughout the entire demography. One way to save computing time is to rescale all parameters. That is, divide all population sizes and numbers of generations by some factor λ, and increase the mutation and recombination rates by that same factor. In that way, the population genetic parameters with remain the same (ie. $\theta = 4N\mu$ and $\rho = 4Nr$ will not change), but the speed of the simulations will increase approximately with λ^2. Note that this does affect times to fixation of selected alleles somewhat (data not shown); a relatively small λ is preferable; I have used $\lambda = 5$ for simulations of human data.

Even with this speedup, thousands of simulations of large genomic segments (~500kb) for thousands of generations are not feasible without access to a computing cluster. Smaller regions and geneologies are likely feasible on single processors.

4 Bug reports, questions, comments

Please note that this code was *not* written by professional programmers, and whenever possible, compare results to theoretical expectations to ensure it's being reasonable. Direct all comments, questions, or bug reports to Joe Pickrell (`pickrell@uchicago.edu`).