GENOME: coalescent-based whole genome simulator

Liming Liang, Sebastian Zöllner, Goncalo Abecasis

November 17, 2011

Index

Introduction …….. 2

Download and Compile the program ……..... 3

Input parameters ……..... 3

Output ……..... 8

Implementation and Computational Issues …….……10

References……11

Introduction

This document describes how to use GENOME, a program to simulate sequences drawn from a population under the Wright-Fisher neutral model (Ewens 1979). The purpose of this program is to simulate sequences on the whole genome scale within practical time. The program can be used to study the sampling properties of any statistics from a genome-wide study or to evaluate the performance of any method that is applied to genome-wide scale data.

The program is based on a standard coalescent model (Hudson 1983, 1990; Donnelly & Tavaré 1995). Starting with the sampled sequences and moving backward in time, coalescent, recombination and migration events are simulated at each generation. These events could occur multiple times and could happen in the same generation. Each coalescent event is recorded and the resulting genealogy tree is constructed. Demographic events such as population bottlenecks and expansions or population merges and splits can also be simulated. In addition to uniform recombination rates, it is possible to allow recombination rates to vary so as to mimic the pattern of hotspots along the genome. After simulating a coalescent tree, mutations are placed along each branch. The number of mutations on each branch follows a Poisson distribution with mean equal to the product of the mutation rate and the branch length. The infinite-site mutation model is assumed, so no recurrent mutation can occur. The genealogy tree can also be output in Newick format, which is identical to that used by programs such as PHYLIP (Felsenstein 2005) and seq-gen (Rambaut & Grassly 1997).

The program is written in C++ and is portable to multiple operating systems. The following sections will describe how to download and compile the program and how to specify the parameters for the program.

If you use the program for your study, the appropriate **citation** is:

Liang L., Zöllner S., Abecasis G.R. (2006) GENOME: a rapid coalescent-based whole genome simulator. *Bioinformatics* **23**(12):1565-7.

Download and Compile the program

All relevant files are available at http://www.sph.umich.edu/csg/liang/genome/. The executable files are available for Linux, SunOS and Windows. If you need to compile the program for other platforms, download the archive genome.tar.gz to your computer and unpack it using the command: tar xvfz genome.tar.gz (if under UNIX) or use WinZip (if under Windows).

A makefile can be found in the folder "genome" after you unpack the source file. It should work for most UNIX like operating systems.

Input parameters

The program uses the following input format:
genome [parameter name_1] [value_1] [parameter name_2] [value_2] **...**

In the command line, if you only type the program's name "genome" without following any parameters, the program will show all the available parameters and their default values in "[ ]" as the following display:

GENOME: Whole Genome Coalescent Simulator. (2006) Liming Liang, Goncalo Abecasis

Parameters and default values:

-pop	number of subpopulations and size of subsamples [2 10 10]
-N	effective size of each subpopulation or the filename for population profile [10000]
-c	number of independent regions (chromosomes) to simulate [1]
-pieces	number of fragments per independent region [100]
-len	length in base of each fragment [10000]
-s	fixed number of SNPs per independent region (chromosome), -1 = number of SNPs follows Poisson distribution [-1]
-rec	recombination rate bewteen consecutive fragments or the filename for recombination rate distribution [0.0001]
-mut	mutation rate per generation per base pair [1e-08]
-mig	migration rate per generation per individual [0.00025]
-seed	random seed, -1 = use time as the seed [-1]
-tree	1=draw the genealogy trees, 0=do not output [0]
-maf	Output SNPs with minor allele frequence greater than [0]
-prop	To keep this proportion of SNPs with MAF < the value of -maf parameter [1]

The input parameters can be written in arbitrary order. If the default value of a parameter is desired, the parameter can be omitted. The detail of each parameter is described in next page.

-pop:

This parameter is used to specify how many subpopulations are simulated and the size of the samples to be drawn from each subpopulation. The first number following the parameter name "-pop" is the number of subpopulations and the following numbers are the sample sizes for each subpopulation. For example:

genome -pop 3 10 15 20

will simulate 3 subpopulations, 10 sequences are drawn from the first subpopulation, 15 sequences are from the second subpopulation and 20 from the third subpopulation.

-N

This parameter specifies the effective size (**number of chromosomes, or number of haploid individuals**) of each subpopulation. For example:

genome -pop 2 10 20 -N 10000

will simulate 2 subpopulation and each with an effective size of 10000.

To simulate past demographic events such as population bottlenecks and expansion or population merges and splits, the user can specify the population history in a population profile (a text file) similar to the following toy example (this example is in the file "population.txt" included with the source distribution):

```
0     500       500       500
1-1 2-1 3-2
10   700       300
1-1 1-2 2-3
20   500       200       300
1-1 2-1 3-1
30   500
```

The odd lines specify the generation and the size of the populations. The first line "0 500 500 500" means at generation 0, there are three populations and the size of each population is 500. The third line "10 700 300" means that at generation 10 backward in time the number of populations changes from 3 to 2, the two populations are of size 700 and 300, respectively. So from generation 0 to 9, there are three populations and each of size 500. From generation 10 to 19, there are two populations of size 700 and 300. From generation 20 to 29, there are three populations of size 500, 200 and 300 respectively (figure 1).

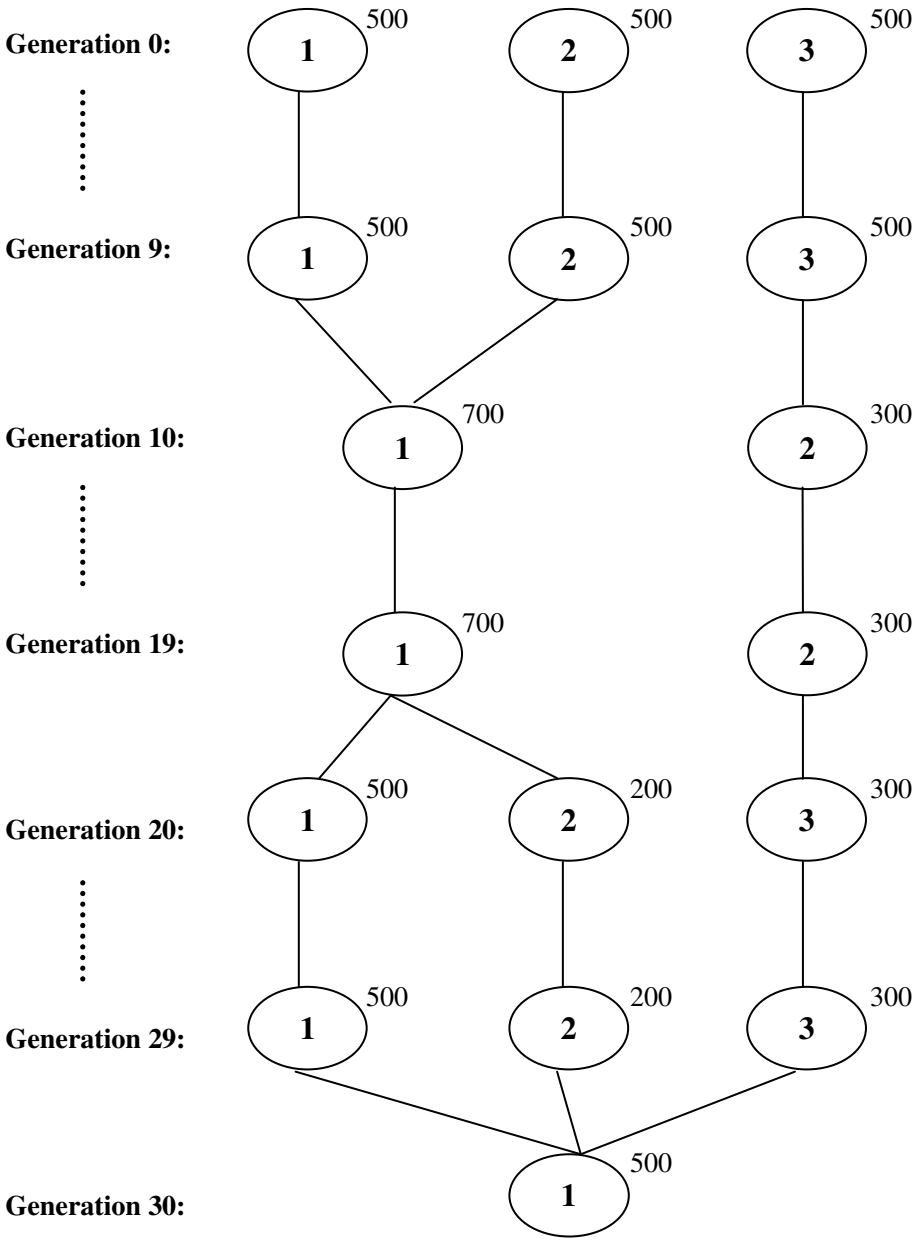
The even lines specify how the populations relate to their parent populations. The second line "1-1 2-1 3-2" means population 1, 2 and 3 at generation 9 are from population 1, 1 and 2 at generation 10, respectively. Similar interpretations apply the 4^{th} and the 6^{th} lines. The whole populations described in this example can be viewed in figure 1. Population 1 at generation 19 is from population 1 and 2 at generation 20. For a sequence in population 1 at generation 19, the program will randomly determine its parent from either population 1 or 2 at generation 20, with probability proportional to the size of the populations at generation 20. In this example, population 1 at generation 20 will be selected with probability 5/7 = 0.714 and population 2 with probability 2/7 = 0.286.

To simulate 10, 20 and 30 sequences from the three populations and the evolution follows the above history, the user can specify the parameters as:

genome -pop 3 10 20 30 -N population.txt

The number of populations by parameter -pop needs to be consistent with generation 0 in the population profile.

Figure 1: Population history specified in the file "population.txt"



-c

The parameter specifies the number of independent regions to simulate. To mimic a genome-wide study, one might want to simulate the whole genome which consists of multiple chromosomes. This parameter can be used as the number of chromosomes.

-pieces

This parameter specifies the number of fragments per independent region. The recombination events can only occur between fragments but not within a fragment. For a short sequence, this parameter can be the number of base pairs. For a very long sequence, this parameter can be combined with the **-len** parameter below to avoid the limit of the available memory of the computer. For example, a sequence of 150 Mb can be specified by parameters combination "-pieces 15000 -len 10000". Each fragment will have a genealogy tree. The similarity of the trees will depend on the recombination rate with the higher recombination rate the less similar of the trees.

-len

The parameter specifies the length in base pair per fragment. The product of the "-len" and "-pieces" parameters is the total length in base pair of the whole independent region. The product of the "-len", "-pieces" and "-c" parameters is the total length of the sequence.

-s

This parameter specifies the number of SNPs needed per independent region. If this is a non-positive number (<=0), the number of mutations of a branch follows a Poisson distribution with mean equal to the product of the mutation rate per base pair (specified by "-mut"), the fragment length (specified by "-len") and the branch length (in the unit of generation).

If this is a positive number, the program will place a fixed number (specified by this parameter) of mutations on the resulting genealogy tree. The probability that a mutation is placed on a branch is proportional to the length of the branch.

-rec

This parameter specifies the recombination rate between consecutive fragments. For example, If one wants to simulate a sequence of 150Mb, the parameters could be "-pieces 15000 -len 10000 -rec 0.0001", if the recombination rate of 10^{-8}/bp is assumed.

The program can allow recombination rates to vary so as to mimic the pattern of hotspots along the genome. User can specify the frequency of different recombination rates or assign the recombination rates between fragments. For example, "-rec

recombination.txt" will use the frequency information in the file "recombination.txt". An example of the recombination profile is given below (also in the file "recombination.txt"):

```
Rec     Freq
0.0001      9
0.001       1
```

The first line in the file is the title and the "Freq" header tells the program this file specifies the frequency of recombination rate. From the second line, it is the recombination rate and its frequency. The first number needs to be the recombination rate, the second number is the relative frequency (the sum of the second column is not necessary to be 1, the program will normalize them). The program will randomly determine the recombination rate between consecutive fragments according to the frequency. If the user wants to assign recombination rates between fragments, this information can be specified in a text with title "Rec Pos". An example is in "recombination-pos.txt":

```
Rec         Pos
0.0001      1
0.001       3
0.5         5
0.7         6
0.9         7
```

The first data line "0.0001 1" means after the 1st fragment, the recombination rates between consecutive fragments will be 0.0001. So the recombination rate between fragments 1 and 2 and the recombination rate between fragments 2 and 3 will be set to 0.0001. Similarly, the recombination rate between fragments 3 and 4 and between fragments 4 and 5 will be set to 0.001. The rate between 7 and 8, and afterward will be set to 0.9.

-mut

This parameter specifies the mutation rate per base pair per generation.

-mig

This parameter specifies the migration rate per generation per individual.

-seed

This parameter specifies the random seed being used. If the value for this parameter is non-positive (<=0), the program uses the system time as the random seed. In the output of the program, the random seed used will be displayed. By using a fixed seed, it should be possible to reproduce output from previous runs.

-tree

This parameter indicates whether the genealogy trees should be output with value "1" for output and "0" for not output.

-maf

SNPs with minor allele frequency greater than this parameter will be output. The output of remaining SNPs will be controlled by the –prop parameter.

-prop

The parameter specifies what proportion of SNPs with minor allele frequency < the value specified in the -maf parameter will be output.

Output

The following is the output that would be produced by the command line:
genome -pop 1 3 -len 100 -tree 1 -pieces 3 -len 10000

GENOME-0.2: Whole Genome Coalescent Simulator. (2009.2) Liming Liang, Goncalo Abecasis

Parameters and effective values:
-pop number of subpopulations and size of subsamples [1 3]
-N effective size of each subpopulation or the filename for population profile [10000]
-c number of independent regions (chromosomes) to simulate [1]
-pieces number of fragments per independent region [3]
-len length in base of each fragment [10000]
-s fixed number of SNPs per independent region (chromosome), -1 = number of SNPs follows Poisson distribution [-1]
-rec recombination rate bewteen consecutive fragments or the filename for recombination rate distribution [0.0001]
-mut mutation rate per generation per base pair [1e-08]
-mig migration rate per generation per individual [0.00025]
-seed random seed, -1 = use time as the seed [-1]
-tree 1=draw the genealogy trees, 0=do not output [1]
-maf Output SNPs with minor allele frequency greater than [0]
-prop To keep this proportion of SNPs with MAF < the value of -maf parameter [1]

Scaled recombination rate = 4.0000e+00
Scaled migration rate = 5.0000e+00
Scaled mutation rate = 6.0000e+00

```
*** POPULATION PROFILE ***
The size of each population is fixed to 10000
Sizes of populations = 10000
*** END OF POPULATION PROFILE ***

random seed=1321554181

Genealogy trees for chromosome:1
The tree for fragment 1= (3:5906,(1:1223,2:1223):4683);
The tree for fragment 2= (3:11740,(1:1223,2:1223):10517);
The tree for fragment 3= (3:8997,(1:1223,2:1223):7774);
Simulate mutation for Chromosome 1
Coalescent Memory free
Mutation process initialized
Mutation assigned
Mutation Memory allocated
Internal memory free
SNP genetic position scaled to [0,1]:
0 0 0.5 0.5 0.5 1 1
Chromosome done
Return chromosome done

Done simulating ARG ...
Total number of SNPs = 7
Samples:
POP1: 0101101
POP1: 0101101
POP1: 1010010
```

The program will first print out the effective values for the parameters and followed by the rates of recombination, mutation and migration scaled by 2*effective population size (**the value specified by –N parameter, i.e. number of chromosomes or number of haploid individuals**). The random seed being used is also displayed.

When the -tree parameter is “1”, the genealogy trees for each fragments is displayed. In this output three trees are displayed in the Newick format. The last output is the simulated sequence leading by the population index. The ancestral allele is represented by “0” and the mutated allele is “1”.

When population history is specified in a text file, e.g. the population.txt file is used, the program will print out the population history as following:

```
*** POPULATION PROFILE ***
Maximum total size of populations at all generations: N = 1500
Total size of populations at generation 0: N = 1500
```

```
Generation = 0
Sizes of populations= 500 500 500
Populations change backward in time:
  From 1 to 1
  From 2 to 1
  From 3 to 2

Generation = 10
Sizes of populations= 700 300
Populations change backward in time:
  From 1 to 1 2
  From 2 to 3

Generation = 20
Sizes of populations= 500 200 300
Populations change backward in time:
  From 1 to 1
  From 2 to 1
  From 3 to 1

Generation = 30
Sizes of populations= 500
*** END OF POPULATION PROFILE ***
```

If varying recombination rates are specified, e.g. recombination.txt is used, the program will print out the recombination profile and the recombination rates between fragments similar as the following example (if 10 fragments are specified in -pieces parameter):

```
*** RECOMBINATION RATES PROFILE ***
Recombination_Rate        Cumulative_frequency
1e-04                      0.9
0.001                      1
The recombination rates between fragments are:
      (1) 1e-04 (2) 1e-04 (3) 1e-04 (4) 1e-04 (5) 0.001 (6) 1e-04 (7) 0.001 (8) 1e-04 (9) 1e-04 (10)
*** END OF RECOMBINATION RATES PROFILE ***
```

Implementation and Computational Issues

The program is written in C++ and is portable to a variety of operating systems. The random number generator uses the Mersenne Twister Code (Matsumoto & Nishimura 1998) as the source of uniform (0, 1) random deviate. The simulator is implemented as a C++ function "genome()", it is convenient for people if they want to incorporate the simulator into their simulation programs.

An example to call the function genome() can be found in main.cpp.

References

Ewens WJ (1979), *Mathematical Population Genetics*. Springer, Berlin

Donnelly, P., Tavaré, S. (1995) Coalescents and genealogical structure under neutrality. *Annu. Rev. Genet.* 29: 401-421

Felsenstein, J. (2005) PHYLIP (Phylogeny Inference Package) version 3.6. Distributed by the author. Department of Genome Sciences, University of Washington, Seattle.

Hudson, R.R. (1983) Properties of a neutral allele model with intragenic recombination. *Theoretical Population Biology*, 23:183–201.

Hudson, R.R. (1990) Gene genealogies and the coalescent process, *Oxford Surveys in Evolutionary Biology* Vol 7: 1-44.

Matsumoto, M., Nishimura, T. (1998) Mersenne Twister: A 623-dimensionally equidistributed uniform pseudorandom number generator. *ACM Trans. on Modeling and Computer Simulation* 8(1):3-30

Rambaut, A., Grassly, N.C. (1997) Seq-gen: An application for the monte carlo simulation of dna sequence evolution along phylogenetic trees. *Comput. Appl. Biosci.* 13:235–238.