CSE 5860: Computational Problems in Evolutionary Genomics, Spring 2014
Course Project: Compute the likelihood of the data by simulation

Ruhua Jiang

1. Project Goal

In this project, I will study how to compute the likelihood of the data via genealogy simulation. The content is mostly based on the Ch.8 of *Coalescent Theory: An Introduction*, by John Wakeley, 2008, so I won't repeat too much formulas and background knowledge here.

2. Method and Result

2.1 Probability of Segregating Site

Let's first use monte carlo method to get the probability of segregating site:

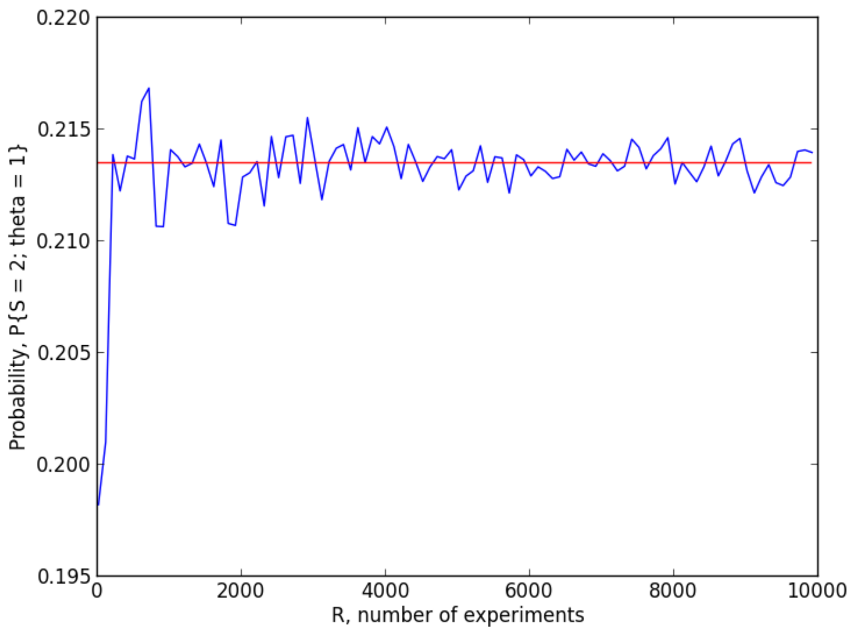

Fig 1. Monte Carlo Simulation, Probability of segregating site

The sample size is set to zero, and theta is set to 1. The red line is the exact probability in theory, P{S=2; theta =1} =0.2135. The blue line is the probability with increase of the number of experiments. From the Fig.1, we can see the result of monte carlo simulation approaching the exact result with the increase of R, however, not in a convergent way. In practise, the number of experiments required will depend on the quantity we want to compute.
segSiteEst.py is the program for doing that experiment.

2.2 Compute Likelihood of Full Data via importance sampling
In the following experiments, we will consider the probability of an entire data set of sequences under the infinite-sites model of mutation. We will use Giffiths and Tavare's method. The method is essentially an importance sampling method.

If we generate genealogy randomly as what we do in 2.1, we will find out lots of genealogy we generated is not compatible with the observed sequence data, which means their probability is zero and they are useless. In Giffiths and Tavare's method, we can generate genealogy from a proposal distribution, which guarantee it always compatible with the data. It use a Markov process back in time, making sure every transition is guaranteed to theose that are compatible with the data. A single realization of the Markov chain is corresponds to a series fo mutation events and coalescent events in the ancestry of the sample. For more details, please refer to Ch.8 of *Coalescent Theory: An Introduction*, by John Wakeley, 2008.

As we know, data that conform to the infinite-sites mutation model without recombination always have a unique perfect phylogeny tree. Gusfield(1991) gives an efficient algorithm for construct it. Here we won't deal with that part, the input of our program is the tree, which in essence represent the same information as the sequence data with the requirement we stated above.

Input Data:

```
X [[4, 0], [3, 4, 0], [2, 1, 3, 4, 0], [0]]
n [2, 1, 1, 1]
theta = 1
```

Below is single realization of the markov chain:

```
======
X [[4, 0], [3, 4, 0], [2, 1, 3, 4, 0], [0]]
n [2, 1, 1, 1]
N 5
Index of coalesce, mutate1 , mutate2:  [0] [2] []
fxn: 0.12
DEBUG_FXN_LIST: [0.12]
Three probability terms: 0.08 0.04 0.0
Transit probability list: [0.6666666666666666, 0.3333333333333333]
Event:coalesce

======
X [[4, 0], [3, 4, 0], [2, 1, 3, 4, 0], [0]]
n [1, 1, 1, 1]
N 4
Index of coalesce, mutate1 , mutate2:  [] [2] []
fxn: 0.0625
DEBUG_FXN_LIST: [0.12, 0.0625]
Three probability terms: 0 0.0625 0.0
```

```
Transit probability list: [1.0]
Event:first kind mutate

======
X [[4, 0], [3, 4, 0], [1, 3, 4, 0], [0]]
n [1, 1, 1, 1]
N 4
Index of coalesce, mutate1 , mutate2:  [] [] [(2, 1)]
fxn: 0.0625
DEBUG_FXN_LIST: [0.12, 0.0625, 0.0625]
Three probability terms: 0 0.0 0.0625
Transit probability list: [1.0]
Event:second kind mutate

======
X [[4, 0], [3, 4, 0], [0]]
n [1, 2, 1]
N 4
Index of coalesce, mutate1 , mutate2:  [1] [] []
fxn: 0.125
DEBUG_FXN_LIST: [0.12, 0.0625, 0.0625, 0.125]
Three probability terms: 0.125 0.0 0.0
Transit probability list: [1.0]
Event:coalesce

======
X [[4, 0], [3, 4, 0], [0]]
n [1, 1, 1]
N 3
Index of coalesce, mutate1 , mutate2:  [] [] [(1, 0)]
fxn: 0.111111111111
DEBUG_FXN_LIST: [0.12, 0.0625, 0.0625, 0.125, 0.1111111111111111]
Three probability terms: 0 0.0 0.111111111111
Transit probability list: [1.0]
Event:second kind mutate

======
X [[4, 0], [0]]
n [2, 1]
N 3
Index of coalesce, mutate1 , mutate2:  [0] [] []
fxn: 0.222222222222
DEBUG_FXN_LIST: [0.12, 0.0625, 0.0625, 0.125, 0.1111111111111111, 0.2222222222222222]
```

```
Three probability terms: 0.222222222222 0.0 0.0
Transit probability list: [1.0]
Event:coalesce

======
X [[4, 0], [0]]
n [1, 1]
N 2
Index of coalesce, mutate1 , mutate2:  [] [] [(0, 1)]
fxn: 0.25
DEBUG_FXN_LIST: [0.12, 0.0625, 0.0625, 0.125, 0.1111111111111111, 0.2222222222222222,
0.25]
Three probability terms: 0 0.0 0.25
Transit probability list: [1.0]
Event:second kind mutate

======
X [[0]]
n [2]
N 2
result is: 3.61689814815e-07
```

The X and n is defined same as in the textbook. There are three possible transitions, which are coalescent event, mutation event (mutate1) that doesn't change the haplotype count, and the mutation event(mutate2) that change the haplotype count. The three transition probabilities are list on "Three probability terms". We can generate a random number from 0 to 1, use it to decide what is the next state in the Markov chain. "fxn" correspond to the formula 8.24.

We can verify above process step by step, and it works correctly.

Important Sampling Result

We can repeat this process by realiza the Markov chain r times, say r = 10000, the result is:

```
Genealogy simulation starts ...
Method: importance sampling.
Number of replicates: 10000
Probability is: 7.90972743056e-07
```

Python Code **simulation.py** is used for that experiment.