Massachusetts Institute of Technology
6.877 Computational Evolutionary Biology, Fall, 2005
Laboratory 2, Part I Addendum: coalescent exercises

Handed out: October 22, 2005 **Due: October 31, 2005**

This portion of the laboratory is an addendum to PART I, consisting of warm-up computer exercises with coalescent software. If you want to cut to the chase and get to the actual questions, they are a few pages on, in subsection 4.2 labeled "Questions", but I would urge that you try out the `ms` program as suggested in section 4.1.

Part 4: Warm-up using simple coalescent computer programs

There are three or four readily available computer programs that are used to work with coalescent analysis in conjunction with polymorphism data. Three have been mentioned and/or demonstrated so far in class: `ms, simcoal,` and `coalface.` There's also a java applet simulator for some very simple simulations, available at http://www.coalescent.dk/. Several other programs take the trees produced by these programs and then generate 'artificial' sequence data sets by sprinkling in Poisson-distributed mutations. In typical 'simulation' mode, given polymorphism data like the one you saw in the earlier sections of this lab, one can use the coalescent to simulate possible polymorphism pattern outcomes, and thus derive predicted polymorphism patterns, given various scenarios of population size changes, recombination, etc., all using basic parameters derived from the original data. We can then go back see which scenario is most compatible with the data – in the stronger sense of also obtaining confidence intervals for some of the test statistics we have seen above. (These might be otherwise quite hard to derive.)

Another mode is rather different: instead of exploring the parameter space 'by hand' we can use likelihood methods to try to find the 'best' estimates for effective population size, mutation rate, and so forth.

In this part of the lab, we shall deal only with the former, 'hand simulation' method to just 'get acquainted' with how to run a coalescent program and use it to simulate a set of sequences.

In particular, we will for the most part use Hudson's computer program `ms`, which simulates coalescent gene trees under a variety of possible population change, migration, recombination, and other historical scenarios. We use this program mostly because it is fast and accurate and can compile on almost any platform with a decent C compiler. Instructions for installing it are given below. (If you have access to either a Windows or a Linux box, you might want to install `CoalFace`, since that has a very nice interface that `ms` lacks, and can easily display the trees graphically, compute nucleotide diversity statistics, etc. – in short, you can use it as a check on your work with the `ms` program if you want, even though we won't use it in this part of the lab. The `ms` program is not so user friendly: it outputs trees in a list format that must be piped other programs; ditto for its descriptive statistics. However, it has more options/flexibility in terms of describing recombination and structured population scenarios. The information for downloading and installing it are given at the end of this document.)

4.1 Installing and running the `ms` program.

There are two options for installing the `ms` program. First, there are pre-compiled binaries for Mac OS X, and RedHat linux systems (including pre-compiled versions of `ms, stats,` and `sample_stats`).

These are available for download in the labs section:

MacOS X version
RedHat linux version

Once you've downloaded and unpacked the appropriate file and run `tar` you'll have a new directory (or folder) `msdir` with the three programs in it, and you should be ready to go. These programs are intended to be run from a unix terminal window, so once your have downloaded the executable you'd actually run them in the following way, redirecting output (as usual you might add the directory where you've downloaded the binaries to your PATH environment variable so you can avoid the **`./ms`** usage):

```
>./ms <input parameters>  > <myoutputfile>
```

If you have any difficulties at all running these compiled binaries, or if you need to use a Windows machine, you'll have to do a simple compilation of the source code, which is available in both .gz and .zip formats in the labs section:

Unix source tar.gz file
Macintosh source tar.gz file
Windows source .zip file

The program is simple to compile on any platform with any current Ansi C compiler (there are a few changes to be noted for proper compilation under Mac OS X, however, so be sure you download the MacOS X files if that's what you intend to compile it on). Again, once you've downloaded, uncompressed, and untarred the relevant file, you'll have a new directory or Folder `msdir`. You should then do the following to compile all three programs (any Ansi C compiler ought to do):

```
gcc –o3 –o ms ms.c streec.c rand1.c -lm
gcc –o stats stats.c -lm
gcc –o sample_stats sample_stats.c tajd.c -lm
```

Here I've chosen a particular random number generator – I suggest you use this one so your results will 'agree,' with other folks', but see the `ms` documentation about the choices (it's not really of concern here).

Using the `ms` program.

A complete pdf documentation file is located in the labs section. Running the program with `ms –h` will produce a summary of all the command line arguments.

In its most basic mode, the user supplies a command line in the form:

```
ms nsam nreps -t θ
```

The above line shows the simplest usage of `ms` that generates samples under the basic neutral model, with constant population size, no recombination, panmixis (free interbreeding), and an infinite-sites model. In this case there are three arguments to `ms`: `nsam`, `nreps`, and, following the switch "`-t`", the parameter θ. The two arguments, `nsam` and `nreps` are required and must appear in this order.(Although there are exceptions, most of the switches can appear in any order.) `nsam` is the number of copies of the locus in each sample, and nreps is the number of independent samples to generate. The third parameter here is the mutation parameter, $\theta = (4N_0\mu)$ where N_0 is the diploid population size and where μ is the neutral mutation rate for the entire

locus. **Remember** that for most purposes, this basic parameter is 'small', between 0 and, say, 20!
At least one of the options, `-t –s,` or `-T` must be used. The latter two options are described later. After `nsam` and `nreps`, any or all other switches with their parameters can be read from a file using `-f` filename.
Example:

```
ms 4 2 -t 5.0
```

In this case, the program will output 2 samples, each consisting of 4 chromosomes (or 'sequences'), generated assuming that $\theta = 5.0$.

The output from the example command in the previous section would look like this (the exact output will depend on the random number generator) :

```
ms 4 2 -t 5.0
27473 36154 10290
//
segsites: 4
positions: 0.0110 0.0765 0.6557 0.7571
0010
0100
0000
1001
//
segsites: 5
positions: 0.0491 0.2443 0.2923 0.5984 0.8312
00001
00000
00010
11110
```

The first line of the output is the command line. The second line shows the random number seeds. Following these two lines are a set of lines for each sample. Each sample is preceded by a line with just "//" on it. That line is followed by "segsites:" then the number of polymorphic sites in the sample. Following that line is a line that begins with "positions:" which is followed by the positions of each polymorphic site, on a scale of (0,1). The positions are randomly and independently assigned from a uniform distribution. (With recombination, the distribution is somewhat more complex.) Following the positions, the haplotypes (sequence blocks) for each of the samples is given, as a string of 0's and 1's. The zeroes denote the 'ancestral form', while the ones denote the mutant or 'derived state'. (Recall that the infinite sites model only admits a binary change at any one site position.) A sample line is omitted if there are no mutations from the initial state of all zeroes.

```
ms nsam nreps -T
```

When the option `-T` is used the trees representing the history of the sampled chromosomes are output. For example, the command line `ms 5 2 -T` results in the following output:

```
ms 5 2 -T
3579 27011 59243
//
((2:0.074,5:0.074):0.296,(1:0.311,(3:0.123,4:0.123):0.187):0.060);
//
(2:1.766,(4:0.505,(3:0.222,(1:0.163,5:0.163):0.059):0.283):1.261);
```

This output represents the trees for two samples. The tree format is in a commonly accepted particular list form that we'll study more closely when we look at phylogenetics in the latter part of the course. This particular list form is called Newick format, and is used by the Phylip

phylogenetic program and a number of applications. The branch lengths are in units of $4N_0$ generations. The sampled chromosomes are labeled 1, 2 ... corresponding or ordered, sampled chromosomes. This labeling is irrelevant for unstructured population models, but with island models described later, the labeling can be important. With recombination a tree is output for each segment within which no recombination has occurred in the history of the sample.

```
ms 5 10 -t 6.0 | grep segsites | cut -f 2 -d ' ' > SegSites.txt
```

4.3 Questions

Now we can test some theory. Let's use coalescent simulations to estimate the average number of segregating sites when $n = 5$, and theta = 6.0, by generating 10,000 replicate samples, using unix to snip out the segregating sites from the output, and passing that to a simple statistical analysis program that is also in the same package. Selecting out the second column via the 'cut' command will give us just the mean value and the standard deviation. (The `–d` argument is used because the output from

```
ms 5 10000 -t 6.0 | grep segsites | cut -f 2 -d ' ' | stats
```

Remember Watterson found analytically that the expected number of segregating sites ought to be:

$$E[S_n] = \theta \sum_{i=1}^{n-1} \frac{1}{i}$$

Question 8.
Work out what the expected value should be for n=5 and this particular value of theta. Then compare the theoretical result to what you get via the simulation – please provide both answers. How much closer does your estimated value come to the true value of theta as you increase the number replicates by one and then two orders of magnitude?

Note that the `stats` program will also print out and particular percentile statistic for the data on a particular column fed into it, including the median and, say, the 99th percentile of the distribution, via this form:

```
stats 0.5 0.99
```

Thus, you could actually use the output from this program to feed a graphing program to view the distribution of any quantity of interest (see the next paragraph below, e.g.).

For other sample statistics, including Tajima's D, we can use the `sample_stats` program by Hudson in the same package. The program will compute the nucleotide diversity π, the number of segregating sites, Tajima's D, and two additional statistics we haven't yet discussed.

```
ms 30 4 -t 3.0 | sample_stats
pi: 1.751724 ss: 6 D: 0.446936 thetaH: 1.282759H:      0.468966
pi: 1.705747 ss: 9 D:-0.774289 thetaH: 0.501149H:      1.204598
pi: 1.390805 ss: 6 D:-0.233099 thetaH: 1.022989H:      0.367816
pi: 3.156322 ss:15 D:-0.560417 thetaH: 3.119540H:      0.036782
```

Question 9.
Now you can use these three programs piped together to show that, say, for n=5, θ=6.0, $E[\pi]=\theta$, as expected by theory. Try this by generation 10,000 replicates with these values: first generate

the samples using `ms`, then pipe these through `sample_stats`; then pull the 2nd column from this output and feed it to `stats`. (You can use the unix 'cut' program as before, via `cut –f 2` to pull out the 2nd column from the output of `sample_stats` and thus get the mean value of π.)
Question 9(i) Please do this for several different replicate sizes and provide a table of $E[\pi]$, θ, and indicate how close the convergence is.

Question 9(ii) Because individuals are correlated through their common ancestry, increasing the number of individuals does not lead to proportional increases in the performance of an estimate. Theory predicts that as the sample size becomes infinite the standard deviation of our estimate of θ (based on nucleotide diversity) will be as follows:

$$sd = \sqrt{\frac{\theta}{3} + \frac{2\theta^2}{9}}$$

Using `ms` and the program pipeline as above, please test the theoretical predictions against simulation for θ=4.0. First, compute the theoretical limit. You might want to use, e.g., 10,000 simulated samples at a time, while increasing *n* and then plot the resulting standard deviation against *n*, in gnuplot, or Excel, matlab, etc. also indicating where the theoretical limit is. Here's the sample output you might get for θ=1.0.

```
TrueTheta  n     MeanEstimate    SDEstimate
1          2     0.979800        1.372655
1          3     0.984733        1.093850
1          4     1.006700        0.993173
1          5     1.005380        0.940195
…
1          12    0.989418        0.802780
1          13    0.998744        0.808657
1          14    0.991237        0.791793
1          15    1.001560        0.811949
```

Question 9(iii) Briefly comment on the implications of your results for sequence sample collection.

Question 10. Exponentially growing populations and Tajima's D
Recall that we would like to use Tajima's D to detect 'statistically significant' deviations from the neutral model, but mentioned that growing populations can also 'fool' this test statistic. In this question, we'd like to briefly explore this issue, and how we can use coalescent models to test this. First, we need to know how the program handles population growth simulation.

Simulating population size changes.
To specify that demographic parameters change at specific times in the past, the `-e` switches are used. These switches are: `-eG, -eg, -eN, -en, -em, -ema, -es` and `-ej`. In each case the first parameter following the switch is the time when the demographic change occurred, measured from the present in units of $4N_0$ generations. In all cases, the parameter values specified apply to the time interval immediately farther in the past from the time point specified. (The term "past-ward" will be used to indicated farther in the past.) The arguments that follow the time parameter specify subpopulations and other relevant parameters, as indicated in the following list:

It is important to note that ms generates genealogical histories by working back from the present, and that each of these `-e` commands changes the parameters for the period immediately pastward (farther back in the past from) the time point specified. For example,

```
ms 10 30 -t 4.0 -eN 0.2 .02
```

specifies that the population size was constant at size N_0 from the present back to time $0.2 * 4N_0$, and farther back in time the population size was $0.02 * 4N_0$. The population size change was instantaneous and occurred at time $0.2 * 4N_0$ generations before the present.

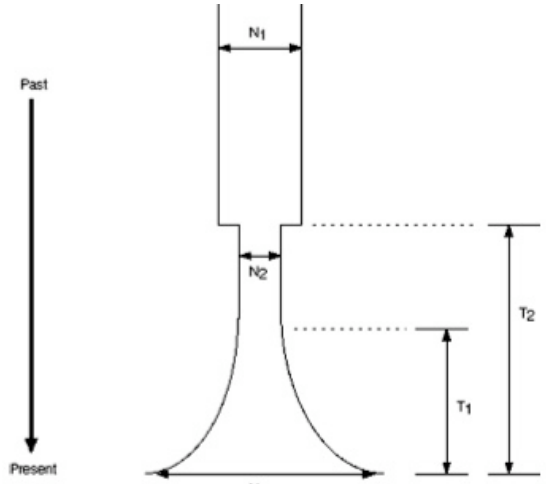
An example. Suppose we have one panmictic population that underwent a size reduction, followed by a period of constant size, followed by population expansion – in short, possibly much like the current human population (putting to one side possible migration events). This demographic history is shown in figure 1 below. To be concrete let us suppose that the population sizes are N_1 = 10,000, N_2 = 5,000, and N_3 = 20,000. Also suppose that the neutral mutation rate is 10^{-8} per site per generation and that we are considering a segment 8,000 base pairs long. In this case, if we take N_0 to be 20,000, we have $\theta = 4 * 20,000 * 10^{-8} * 8,000 = 6.40$. Suppose that T_1 is 16,000 generations, or 16,000/(4*20,000) = 0.2 in units of $4N_0$ generations. Similarly, suppose that T_2 is 24,000 generations or 0.3 in units of $4N_0$ generations. To specify the exponential growth on the command line we need to calculate the growth parameter α. To obtain α we solve the equation,

$$5,000 = 20,000 * \exp^{-\alpha 0.2}, \text{ which is } \alpha = -(1/0.2) * \log(5,000/20,000) = 6.93.$$

The command line to generate 1000 samples of size 15 is thus written as follows:

```
ms 15 1000 -t 6.4 -G 6.93 -eG 0.2 0.0 -eN 0.3 0.5
```

The phrase `-G 6.93` specifies that the population decreases as we go back in time according to the equation, $N(t) = N_0 \exp^{-6.93t}$. The phrase `-eG 0.2 0.0` specifies that the growth rate changes to zero at time 0.2, and the population size preceding this time will be $N_0 \exp^{-6.93*0.2} = N_0 * 0.25$. The phrase `-eN 0.3 0.5` specifies that the population size before 0.3 times units was half of N.



Courtesy of Richard Hudson. Used with permission.

Figure 1. Example of strong bottleneck followed by exponential expansion, as it relates to the arguments to be supplied to the `ms` program.

Now on to the actual question. We can obtain a distribution for a sample statistic like Tajima's D as follows. The command:

```
ms 30 10000 -t 3.0 -eN .2 0.1 | sample_stats | cut -f 6 | stats .025 .5 .975
```

will output the estimated mean, standard deviation, 2.5th percentile, median, and the 97.5th percentile of the distribution of Tajima's D for a model in which the population has experienced

a recent rapid increase in size. (In this example, the population size was one-tenth the current size until $0.2 * 4N_0$ generations ago. Since Tajima's D is output in the 6th column of the output of `sample_stats` we use "`cut -f 6`" to get the D values.) This gives the output as a tab-delimited table.

Now we can generate 'simulated' Tajima D scores and recover their distribution under the 'expanding population' scenario. Here are the questions.

Question 10(i). Run the program as described and print out, then plot, the distribution of Tajima's D for this given set of population parameters (you'll have to 'fill in' more percentiles to get a better looking graph). Compare this to the value you get when you run the same simulation without any population growth. Now go back to Question 5(v) and Question 7 of this lab and reflect a bit…

Question 10(ii). In Question 7, we saw that mitochondrial DNA in one sample population led to a Tajima's D value less than –2, which is the usual cut-off on one side (given a beta distribution) for significance at the 0.05 level, so we can reject the null model with this confidence level – there is supposedly a less than a 5% probability that this difference between nucleotide diversity and the segregating sites estimator could be due to chance alone, thus possibly rejecting the neutrality of the mutations. However, there are other elements to the null model, in particular, that of population growth. Can expanding population growth ever get us in trouble here?

In order to investigate this question see if you can find any combination of changes in the population demographic parameters above (e.g., the rate of the expansion, how long ago it was) that could 'push' the score below –2.0. (You want to start with the same parameter values as in Question 10(i). However don't simulate 10,000 samples. Instead, start with, say, 50 samples and just run the output of the `ms` program through `sample_stats` to see if you can find a Tajima's D value greater than +2 or less than –2. You might also try experimenting with changing the θ value, which you may recall is a compound parameter of population size and mutation rate, and see if you can find a combination with demographics that works.) Of course, just finding one odd-ball example doesn't make the case; rather, we want to know how likely this is, due to chance. For this, you'll have to create 10,000 sample data sets as before. Please describe the outcome of your results along with a brief description of their implications with respect to this particular context: what combination of segregating sites, time expansion, etc. leads to high positive or negative D values? What about in this mitochondrial DNA context – if we simulate lots of samples, what is the chance that population size changes alone could account for a score below –2? Please explain what you've found out.

4.3 Appendix: Installing and running the `CoalFace` program.

CoalFace is a simulation of the coalescent process with the visual display of gene genealogies, developed by Wayne Delport & Michael Cunningham at the University of Pretoria. CoalFace has been developed predominantly as a teaching tool, yet some basic coalescent simulation analyses can be performed. Both Windows and Linux (Intel) executables are available at http://www.up.ac.za/academic/genetics/staff/Bloomer/Research/software.htm.

To install, just download, uncompress, and then either just double-click the CoalFace app (in the CoalFace Windows folder that is produced) or run the shell script CoalFace (in Linux) in the CoalFace directory. From the first "Main" tab you can select the population size and the number of simulation repetitions to run, as with `ms`. However, you don't select the theta parameter in the same way; instead, you have to select a mutation rate in conjunction with the (effective)

population size, which is done under the “Sequence” tab in the “mutation rate” box. You also want to tick the “draw mutations on genealogy” checkbox on the “Main” tab so you can see the mutations generated on the coalescent tree. Statistical output from the coalescent simulation is placed into a file you specify under the “Files” tab, viz., the “Output File” box. After a run, that file will contain various statistics on (simulated) nucleotide diversity, etc.