# COMPSCI 369 Assignment 3

# Due Tuesday June 5 2018

This assignment is worth 15% of your final grade and is marked out of 30. There are 6 bonus marks (3%) available for completing the final part. This means the best possible mark for this assignment is 36 out of 30 (18% out of 15%).

A notebook `tree.ipynb`, with helpful code is available on Canvas.

Write your submission in a Jupyter notebook and write any code in Python 3. You should submit the .ipynb file and a .html file with all output displayed. The primary document the markers will look at is the .html file.

In your report, include explanations of what you are doing and comments in the code. Where you are making mathematical derivations, show your working.

Submit your notebook as an .ipynb file and as an .html file with all output showing by Tuesday, June 5th, 2018, 11:59 pm on Canvas.

In this assignment you will write a method that simulates random coalescent trees, simulates DNA sequences on those trees, reconstructs trees from the simulated sequences and also infers a posterior distribution of the effective population size from the true coalescent trees. Each part is worth 6 marks (3%).

1. Write a method that simulates trees according to the Kingman coalescent model (described below) which takes as input the number of leaves, $n$, and the effective population size, $N_e$. Use the provided Python classes to represent a tree (available on Canvas).

   The Kingman coalescent model is a lineage death process backwards in time that provides a simple algorithm for simulating a gene tree of $n$ chromosomes from a larger population of $N_e$ chromosomes. From each leaf, start a lineage going back in time. Each pair of lineages coalesces at rate $1/N_e$. When there are $k$ lineages, the total rate of coalescence in the tree is $\binom{k}{2}/N_e$. Thus, we can generate a coalescent tree with $n$ leaves as follows:

   ```
   Set $k = n, t = 0$.
   Make $n$ leaf nodes with time $t = 0$ and labeled from 1 to $n$. This is the set of
       available nodes.
   While $k > 1$, iterate:
       Generate a time $t_k \sim \text{Exp}\left(\binom{k}{2}/N_e\right)$. Set $t = t + t_k$.
       Make a new node, $m$, with height $t$ and choose two distinct nodes, $i$ and $j$,
           uniformly at random from the set of available nodes. Make $i$ and $j$ the
           child nodes of $m$.
       Add $m$ to the set of available nodes and remove $i$ and $j$ from this set.
       Set $k = k - 1$.
   ```

   To check the correctness of your implementation, simulate 1000 trees and check that the mean height of the trees (that is, the time of the root node) agrees with the theoretical mean of $2N_e(1 - \frac{1}{n})$.

Use the provided `plot_tree` method to include a picture of a simulated tree with 10 leaves and $N_e = 100$ in your report.

2. Write a method to simulate sequences down a simulated tree according to the Jukes-Cantor substitution model. Your method should take a tree with $n$ leaves, sequence length $L$, and a mutation rate $\mu$. It should return either a matrix of sequences corresponding to nodes in the tree or the tree with sequences stored at the nodes.

   Your method should generate a uniform random sequence of length $L$ at the root node and recursively mutate it down the branches of the tree, using the node heights to calculate branch length. Pseudocode methods for random sequence generation and mutation down a lineage are provided at the end of this problem.

3. Write a method to calculate the Jukes-Cantor distance matrix, $d$, from a set of sequences, where $d_{ij}$ is the distance between the $i$th and the $j$th sequences. Recall that the Jukes-Cantor distance for sequences $x$ and $y$ is defined by

   $$d_{xy} = -\frac{3}{4}\log\left(1 - \frac{4f_{xy}}{3}\right)$$

   where $f_{xy}$ is the fraction of differing sites between $x$ and $y$. Since we will be dealing with short sequences, use the following definition of $f_{xy}$ so that the distances are well-defined:

   $$f_{xy} = \min\left(\frac{D_{xy}}{L}, 0.75 - \frac{1}{L}\right)$$

   where $D_{xy}$ is the number of differing sites between $x$ and $y$ and $L$ is the length of $x$.

   Include a simulated set of sequences of length $L = 50$ from the tree *leaves* and corresponding distance matrix in your report for a tree with $n = 10$, $N_e = 100$ and mutation parameter $\mu = 0.0015$.

4. Now simulate a tree with $n = 10$ and $N_e = 100$ and on that tree, simulate three sets of sequences with lengths $L = 50$, $L = 200$ and $L = 1000$, respectively, with fixed $\mu = 0.0015$. For each simulated set of sequences from the leaves, calculate the distance matrix and print it out.

   Now for each distance matrix, reconstruct the tree using the provided `compute_upgma_tree` method. Use the `plot_tree` method to include a plot of the original tree and a plot of each reconstructed tree.

   Comment on the quality of the reconstructions and the effect that increasing the sequence length has on the accuracy of the reconstruction.

5. Implement an MCMC algorithm to sample the posterior distribution of the effective population size $N_e$ given the tree simulated in the previous step. The proposal kernel should be a uniform random walk on the parameter $N_e$, so that $\delta \sim \text{Unif}(-10, 10)$ and $N_e' = N_e + \delta$. The lower boundary of zero can be treated by reflection or rejection. The target distribution is the coalescent likelihood:

   $$p(T|N_e) = \prod_{k=2}^{n}\left(\frac{1}{N_e}\exp\left(\frac{k(k-1)t_k}{2N_e}\right)\right)$$

   where $t_k$ is the time duration that the tree $T$ has $k$ lineages.

Plot the posterior distribution of $N_e$ from an MCMC chain of length 10,000 steps.

Comment on how you would get a better estimate of the effective population size.

6. BONUS MARKS.
   Run BEAST2 (`http://beast2.org`) on your largest simulated sequence data set, assuming constant-size coalescent, Jukes-Cantor substitution model, a strict clock and fixing the clock rate to the truth (0.0015). Plot the posterior distribution of the effective population size and comment on its coverage of the true value. Compare it to the result from the previous step.

## Pseudocode methods

The method randseq simulates a uniform random sequence. The method mutate mutates a given sequence according to the Jukes-Cantor model of mutation over a given length.

```
randseq(L)
  for (i in 1 to L)
    seq[i] = choice(['A','C','G','T'], [0.25,0.25,0.25,0.25])
  return seq
```

The mutate method below allows "mutations" from a base to itself. The uncorrected mutation rate ($\mu$ rather than $\frac{3}{4}\mu$) can therefore be used.

```
mutate(X,t,mu)
  L= X.length()
  \\ the number of mutations is Poisson with total rate L*mu*t
  numMutation = randpoiss(L*mu*t)
  \\ for each mutation, choose a site to mutate and mutate it
  for (i in 1 to numMutation)
    \\ choose a site
    site = ceiling(random()*L)
    \\ mutate that site
    X[site] =  choice(['A','C','G','T'], [0.25,0.25,0.25,0.25])
  return X
```