Population Genetic Analyses of Genomic Data 2

USN: 303039534

1 Exercises

1.1 Phylogenetics

A python script was written to construct a tree graph from the data structure provided, storing the length of each edge and the label of each node. The script also includes a topological sort function that will sort the nodes non-uniquely such that for no pair of nodes a and b with indices i and j where $i < j$ will b be a parent of a. Using this function, we include in our parsing function a subroutine that counts for each node a how many total descendants it has and how many of these are leafnodes and internal nodes respectively. Finally, each node is also annotated with its total length from the rootnode.

These numbers are given in table 1 sorted from fewest to most total descendants. This also identifies node 5 as the root node and nodes 8,14,15,2,11,6,13 as leafnodes.

node	cumulative daughter nodes	internal nodes	leafnodes	distance from root
8	0	0	0	8.2
14	0	0	0	8.2
15	0	0	0	8.2
2	0	0	0	8.2
11	0	0	0	8.2
6	0	0	0	8.2
13	0	0	0	8.2
12	2	2	0	5.5
4	2	2	0	7.1
7	2	2	0	3.5
3	4	3	1	3.7
10	4	3	1	5.2
1	10	6	4	2.2
5	14	8	6	0

Table 1

b) We include in the final column of table 1 the distance of each node from the rootnode. This shows that the length to every leafnode from the root is 8.2 units. The tree is thus consistent with an evolutionary tree where branches represent time since all current individuals (leafnodes) are equidistant from their combined most recent common ancestor (the rootnode).

We can also show this visually by plotting the tree such that the length along the y-axis corresponds to the length L between two nodes. We label each node with its node label and give the result in figure 1

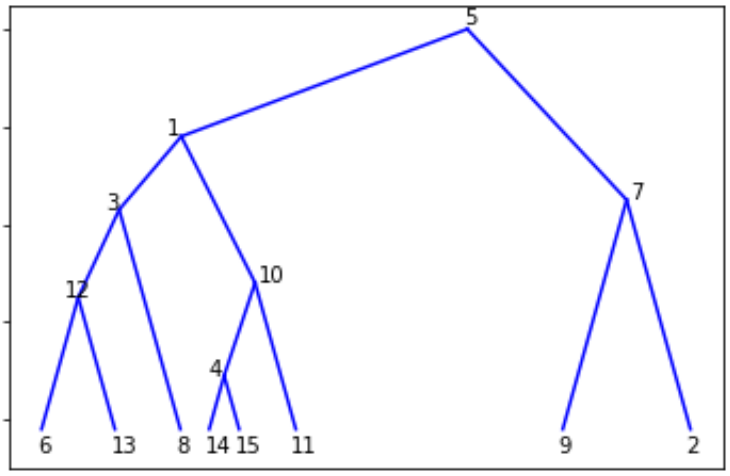


Figure 1: Visual representation of the tree provided in the assignment. The y axis is proportional to the branch length between nodes.

c)

Given our topological sort function, we can now easily print out a relabelled tree where every daughter node has a lower number than its parent, simply by labelling each node according to its position in the reverse of the topologically sorted list.

new label	original label	cumulative daughter nodes	internal nodes	leafnodes	distance from root
11	8	0	0	0	8.2
9	14	0	0	0	8.2
8	15	0	0	0	8.2
5	9	0	0	0	8.2
4	2	0	0	0	8.2
3	11	0	0	0	8.2
2	6	0	0	0	8.2
1	13	0	0	0	8.2
7	12	2	2	0	5.5
10	4	2	2	0	7.1
6	7	2	2	0	3.5
12	3	4	3	1	3.7
13	10	4	3	1	5.2
14	1	10	6	4	2.2
15	5	14	8	6	0

Table 2

Or in the format provided in the assignment:

14	12	1.5	13	3.0
12	7	1.8	11	4.5
10	9	1.1	8	1.1
15	14	2.2	6	3.5
6	5	4.7	4	4.7
13	10	1.9	3	3.0
7	2	2.7	1	2.7

Table 3

We can also use this list to plot a relabelled graph from which it is clear that the new labelling satisfies the condition that every daughter node has a lower number than its parent.

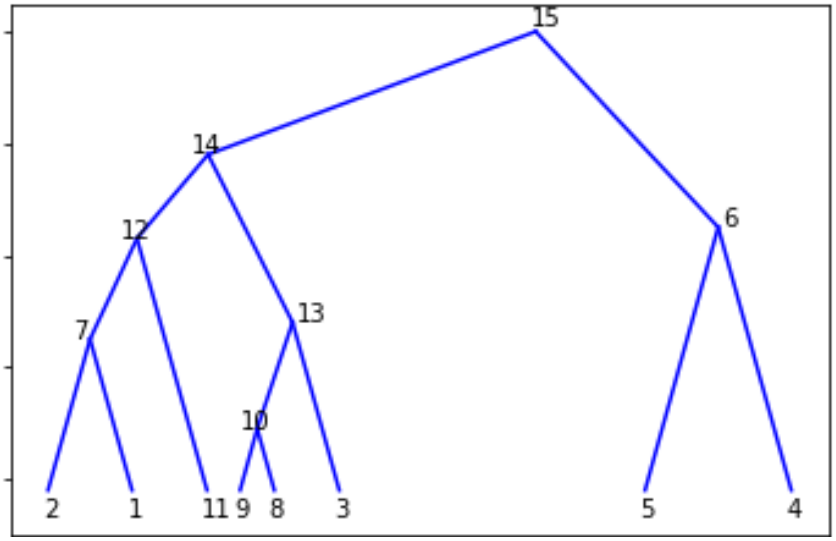


Figure 2

1.2 Time-dependent selection

We consider an organisms with 1000 polymorphic genes, each of which can be fixed in state 0 or 1. We let f_i^a denote the fitness advantage of gene i in state a and draw it from an exponential distribution with rate parameter 100, $f \sim \text{Exp}(\lambda = 100)$. We assume state 1 to be beneficial and let $f_i^0 = -f_i^1$, $f_i^1 > 0$. It is assumed that no more than one gene is polymorphic at any time. Given a mutation in a gene, the probability of fixation is given by

$$p_{fix}(f_i^a) = \frac{1 - e^{-2f_i^a}}{1 - e^{-2Nf_i^a}} \tag{1}$$

Where N is the population size. The lifespan of each polymorphism is assumed to follow equation 2 with N=100 and using the natural logarithm.

$$\min\left(10, \frac{2\log\left(N-1\right)}{919|f_i^a|}\right) \tag{2}$$

We note that this particular system can be implemented in two different ways.

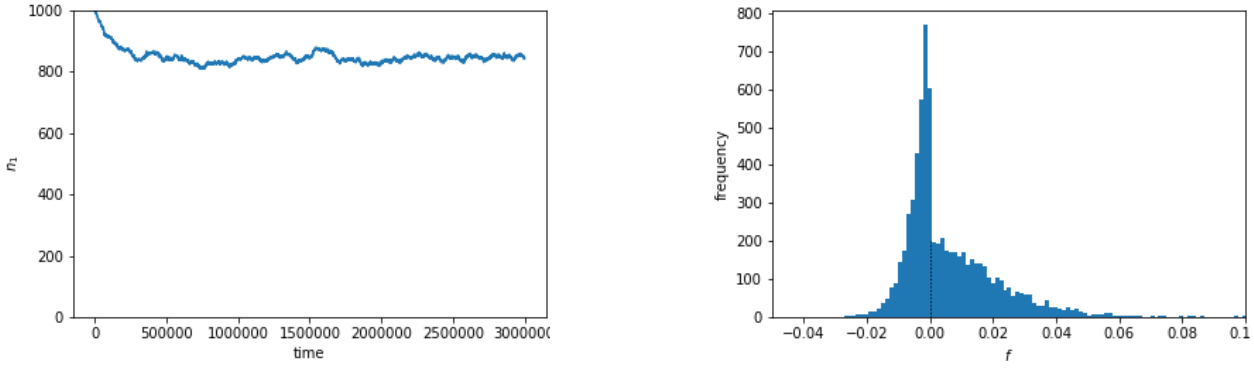
1. In the first implementation (scenario 1), we consider consecutive mutations between the 0 and 1 states of a gene to be independent such that whenever a mutation occurs in a gene in state 1, we draw a new fitness $f_i^0 = -|f_i|$ and whenever a mutation occurs in state 0, we draw a new fitness $f_i^1 = +|f_i|$.
2. In the other implementation (scenario 2), the relative fitnesses of states 0 and 1 are fixed and we draw all f_i^1 at the beginning of the simulation corresponding to the system switching between two pre-defined alleles over the course of the simulation. On one hand, this justifies always having $f_i^0 < 0$ and $f_i^1 > 0$, but on the other hand it is unlikely that a second mutation in gene i will exactly reverse the effect of the first mutation. In the following, we consider both of these interpretations.

EDIT: I answered this question prior to the clarification specifying that we are expected to consider only the second implementation. I have therefore retained a comparative analysis rather than rewriting my answer to the question.

We simulate these two systems for 3,000,000 timesteps and plot the number of loci in state 1 for both of the two approaches in figures 3a and 5a. We see that in both cases, an equilibrium state of $n_1 \approx 800$ is reached after 500,000 timesteps. While the present simulations were initiated with $n_1 = 1000$, the same equilibrium state is reached from different initial conditions.

b)

We start sampling the f values that lead to successful fixation after t=500,000 and run the simulation for an additional 2,500,000 timesteps. We then plot histograms showing the frequency of fixation for a given value of f in figures 3b and 5b for scenario 1 and 2 as described above.



(a) $n1$ as a function of simulated time

(b) Frequency of fitnesses f that fix in the population

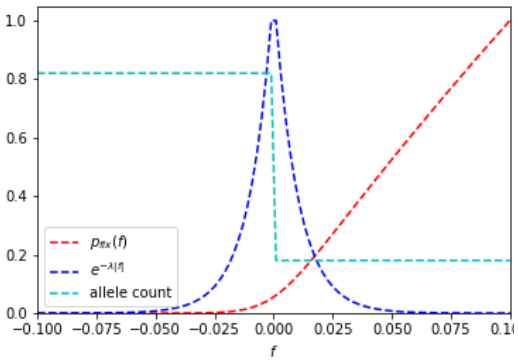
Figure 3: Simulation results when drawing a new f_i whenever a mutation occurs (scenario 1).

We can explain the shape of the distribution of frequencies of fixation in the case of resampling by combining three separate distributions (figure 4a). These are
1. The number of alleles in states 0 and 1 respectively, since a $0 \to 1$ transition is associated with $f > 0$ and vice versa.
2. The probability with which we draw a given value of $|f|$. This is given by the exponential distribution $e^{-\lambda|f|}$ with

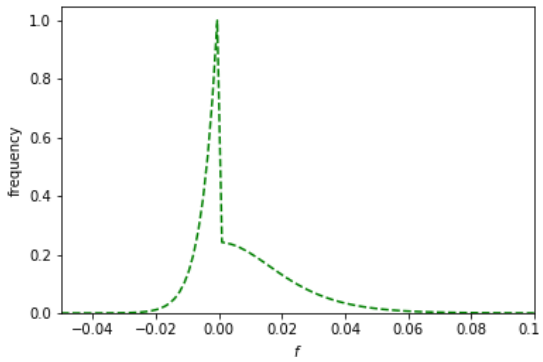
$\lambda = 100$.
3. The probability with which a mutation fixates which is given by equation 1.
If we multiply these three probability distributions, we arrive at a theoretical distribution that resembles the simulated data very closely (figure 4b), suggesting that this is a good explanation for the observed data. The frequency distribution is thus given by

$$y \propto n_1 * exp(-|f|) * \frac{1 - e^{-2f}}{1 - e^{-2Nf}} \quad if \quad f < 0 \tag{3}$$

$$y \propto n_0 * exp(-|f|) * \frac{1 - e^{-2f}}{1 - e^{-2Nf}} \quad if \quad f > 0 \tag{4}$$



(a) Distributions of alleles (cyan), frequency of drawing a given fitness (blue), and probability of fixation (red) as a function of fitness f.



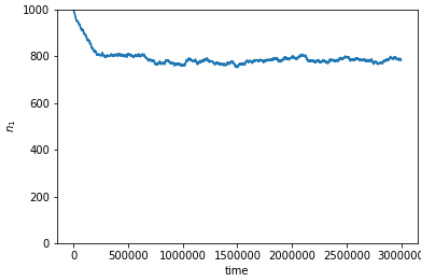
(b) Product of the three distributions in (a). This distribution explains the observed form of the simulated data (figure 3b).

Figure 4

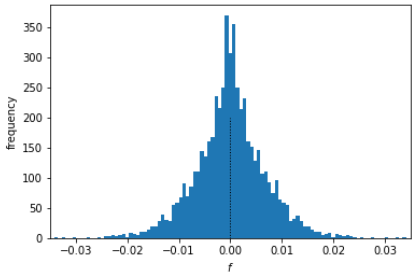
If instead we fix the values of f at the beginning of the simulation (scenario 2), consecutive mutations are no longer independent and we instead observe a symmetric distribution of frequencies of fixation (figure 5b). We can understand this by considering what happens at a single site i. Whenever this site is in state 1, the probability of fixation is $p_1 = p_{fix}(f_0^i)$ and whenever it is in state 0 the probability of fixation is $p_0 = p_{fix}(f_1^i)$. This probability can be interpreted as a frequency of switching, and the average time of switching is thus the inverse of this frequency. However, because the state must switch back from 1 to 0 to switch from 0 to 1, in the limit of $t \to \infty$ where many switches occur, the average switching time for site i is given by the mean switching time from $0 \to 1$ and $1 \to 0$. Denoting this mean switching time τ_i,

$$\tau_i \propto \frac{1}{2p_1} + \frac{1}{2p_0} = \frac{1 - e^{-2Nf_i^0}}{2 - 2e^{-2f_i^0}} + \frac{1 - e^{-2f_i^1}}{2 - 2e^{-2Nf_i^1}} \tag{5}$$

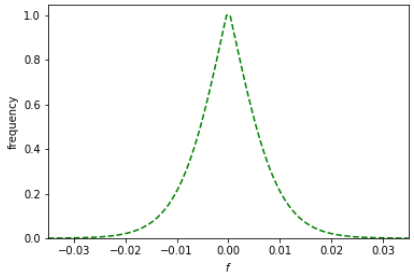
The average frequency of switching for site i is thus $\omega \propto \frac{1}{\tau_i}$. Since the distribution of fitnesses is given by $e^{-\lambda|f_i|}$, the probability distribution of fixation as a function of f is therefore given by $\frac{1}{\tau_i} e^{-\lambda|f_i|}$. This has been plotted in figure 5c and we see that it fits very well with the simulated data.



(a) n_1 as a function of simulated time



(b) Frequency of fitnesses f that fix in the population



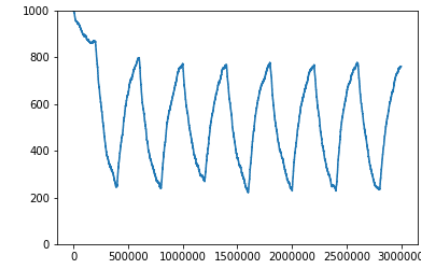
(c) Predicted frequencies of fixation given by $\frac{e^{-\lambda|f_i|}}{\tau_i}$.

Figure 5: Simulation result when f_i are fixed at the beginning of the simulation (scenario 2).

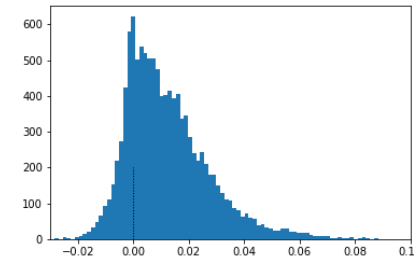
c)

We now let the environment change at a frequency determined by a poisson distribution with rate $\tau = 5 * 10^{-6}$. Upon a change in environment, fitnesses of alleles 1 and 0 reverse such that $f_i^1 < 0$ and $f_i^0 > 0$ for all i. We see in figures 6a and 7a that this leads to oscillations in n_1 as allele 1 becomes alternatingly more and less fit compared to allele 0 at each site.

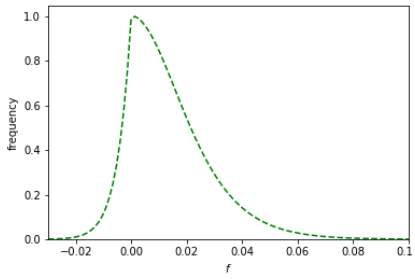
In the limit of $t \to \infty$ this averages the populations of state 0 and state 1 and leads to the resampling and fixed fitness strategies having equivalent mean behavior since consecutive fixations can now occur with the same change in fitness Δf for a single site i in the fixed fitness model. This is illustrated in figures 6 and 7, where we also note that we still sample fitness space less thoroughly with fixed fitnesses.



(a) n_1 as a function of simulated time

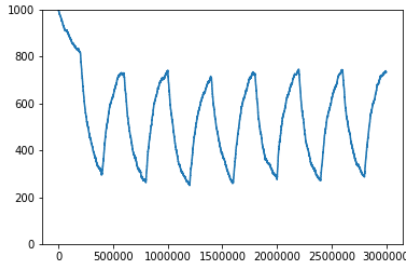


(b) Frequency of fitnesses f that fix in the population

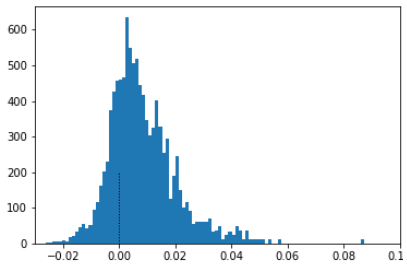


(c) Predicted frequencies of fixation given by $e^{-\lambda|f|}p_{fix}(f)$

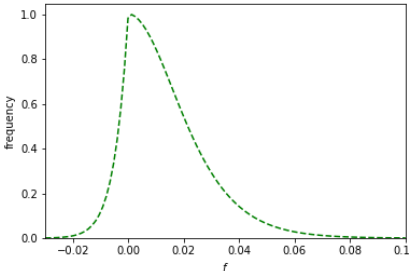
Figure 6: Simulated and predicted behavior of a system with resampling and changing environment (scenario 1).



(a) $n1$ as a function of simulated time



(b) Frequency of fitnesses f that fix in the population



(c) Predicted frequencies of fixation given by $e^{-\lambda|f|}p_{fix}(f)$

Figure 7: Simulated and predicted behavior of a system with constant $|f_i|$ and changing environment (scenario 2).

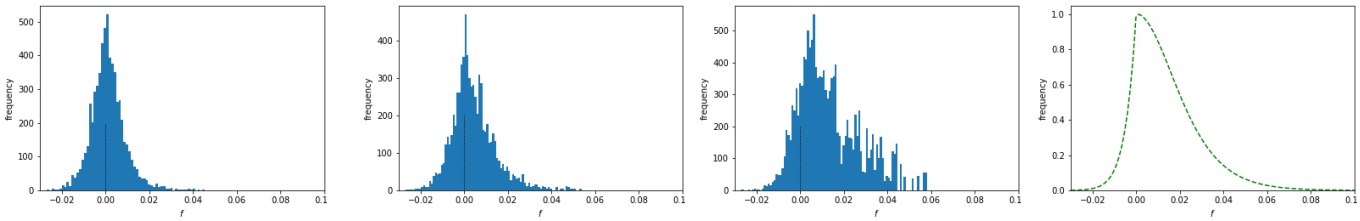
In the long time limit of a changing environment, $\langle n_1 \rangle = \langle n_2 \rangle$ where $\langle\rangle$ denotes a time average. The frequency

distribution is thus given by equations 3 and 4 with $n_1 = n_2 = 500$, giving

$$y \propto exp(-|f|) * \frac{1 - e^{-2f}}{1 - e^{-2Nf}} \tag{6}$$

This has been plotted in figures 6c and 7c and we see that this theoretical distribution fits well with the data in both cases.

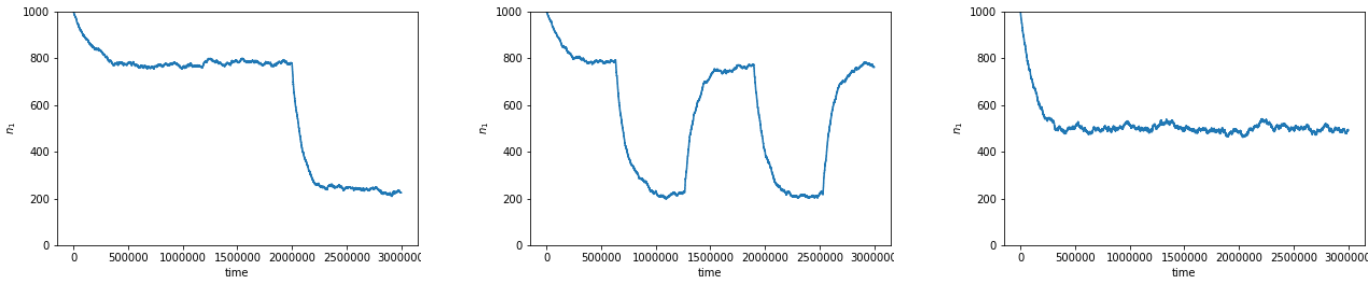
d) As τ becomes large, changes in the environment become increasingly frequent, making the time average approximation increasingly accurate. This leads to the distribution of fitnesses that fix becoming increasingly similar to equation 6 for both scenarios 1 and 2.



(a) $\tau = 10^{-7}$ (b) $\tau = 10^{-6.5}$ (c) $\tau = 10^{-4}$ (d) Predicted as $\tau, n \to \infty$

Figure 8: Frequency of fitnesses f that fix in the population for a changing environment with fixed fitnesses $|f|$ (scenario 2). As τ increases, this increasingly resembles the predicted distribution resulting from a time average of the populations n_1 and n_0.

This can also be rationalized by looking at the n_1 curves where for small τ we only see a single or a few transitions, whereas for large τ, transitions occur very rapidly and the equilibrium state has $n_1 \approx n_2 \approx 500$. Since the rapidly changing environment leads to $\langle f_i^0 \rangle = \langle f_i^1 \rangle$ over the timescale of mutagenesis, the frequency of fixation is simply proportional to the sampling frequency $exp(-|f|)$ times the probability of fixation $\frac{1 - e^{-2f}}{1 - e^{-2Nf}}$ in both states 0 and 1.



(a) $\tau = 10^{-7}$ (b) $\tau = 10^{-6.5}$ (c) $\tau = 10^{-4}$

Figure 9: n_1 as a function of simulated time for a changing environment with fixed $|f_i|$. As τ increases, oscillations become increasingly fast, allowing us to take a time average of the populations n_1 and n_0.

We thus see that as τ increases, the proportion of fixed mutations that are beneficial (i.e. has $f > 0$) increases until it reaches the limiting distribution given by equation 6. In the case of constant fitnesses (scenario 2), a better fit to the predicted frequencies would be obtained with an increased number of genes.

e) *Drosophila* have a relatively short generation time of ≈2 weeks. Changes in the environment due to factors such as changing seasons thus occur over a timescale that allows for fixation of mutations in small populations within a single season. This scenario is qualitatively similar to the scenario considered above, where e.g. alleles that are more beneficial during the summer can arise and fix during the warmer months while alleles that are more beneficial in the winter can arise and fix during the colder months. As exemplified by figures 8 and 9, this will lead to the population having a large proportion of beneficial mutations if the relative timescales of the generation time (and thus time of fixation) and change in environment are appropriate.

As is evident from figure 9, this can lead to there being a number of beneficial mutations at any one time, but at a later time these may turn out to be detrimental and thus be cleared from the population. This does not lead to the mean fitness of the population increasing over long timescales, but it does lead to the population being fitter at a given time of year, although there will be a delay corresponding to the fixation time of the mutations.

This particular scenario is likely to be more similar to the model with resampling of fitnesses rather than fixed fitnesses since we can imagine there to be a set of mutations $\{1_j\}$ that are more beneficial during the summer and a set of mutations $\{0_j\}$ that are more beneficial during the winter, and while the population can change between these two states, it is unlikely that mutations are exactly reversed. Instead, the population will likely sample genotype space in a more continous manner where mutations in a given gene can have different fitness advantages depending on how they affect the resulting protein and its expression level.

notes to self

$$n1 \cdot p(a = 1 \rightarrow a = 0) = n2 \cdot p(a = 0 \rightarrow a = 1) \tag{7}$$

This implies that the integral of the right side of the above figure must be equal to the integral of the left side. We can thus calculate the expected n1 analytically.

1.3 Ancestral inference

a)

We start by considering the genealogy at a single haploid locus ignoring recombination and mutation. An example of such a system might be human mtDNA. A simple theory we can use to calculate the time to the most recent common ancestor (TMRCA) for a set of n individuals given a total population size of N in this scenario is the Kingman coalescent. We start by assuming that reproduction follows a Wright-Fisher (WF) model and thus satisfies the following conditions:

1. Nonoverlapping generations
2. Random reproduction (Poission distributed offspring count)
3. Constant population size N
4. Random mating

Now let $p(n)$ be the probability that none of n individuals share a parent in the previous generation. We can use this to also calculate $1 - p(n)$, the probability that a coalescence event does occur in the previous generation. Together, these two parameters allow us to calculate the distribution of $T(i)$ which we define as the time required for the first two of i individuals to share a common ancestor. We assume that at most one coalescence event occurs per generation. Finding an expression for $T(i)$ also allows us to calculate the time to the most recent common ancestor $TMRCA$ of all n individuals as

$$TMRCA = \sum_{i=2}^{n} T(i) \tag{8}$$

Considering only $n = 2$ individuals from a population of size N, the probability that they share an ancestor in the previous generation is $(1-p(2)) = 1/N$ since mating is random in the WF model. This gives $p(2) = 1-1/N$. Consider now the case where $n = 3$. In this case, we can calculate the probability of no shared ancestors as the probability of the first two individuals not having a shared ancestor $(1 - 1/N)$ times the probability of the third individual not sharing ancestors with the first two $(1 - 2/N)$:

$$p(3) = (1 - 1/N) * (1 - 2/N) \tag{9}$$

Generalizing this to arbitrary n gives $p(n) = p(n-1) * (1 - (n-1)/N)$. Using this inductive argument together with $p(2) = 1/N$, we get

$$p(n) = \prod_{i=1}^{n-1} 1 - i/N = 1 - \sum_{i=1}^{n-1} \frac{i}{N} + \mathcal{O}(1/N^2) = 1 - \frac{(n-1)n}{2N} + \mathcal{O}(1/N^2) \tag{10}$$

This is the probability that no individuals in a sample of n share any ancestors in the previous generation given a population size of N. Conversely we can also calculate the probability that at least two of the n individuals share an ancestor in the previous generation which to first order is

$$1 - p(n) \approx \frac{n(n-1)}{2N} \tag{11}$$

In the WF model, generations are independent such that the probability that n individuals share no ancestors in t generations is $p(n|t) = p(n)^t$ Hence the probability that the first coaslescence event for the n individuals occurs in generation t, $C(t|n)$, is the probability that no coalescence has occurred before then

$$p(n)^t \approx (1 - \frac{n(n-1)}{2N})^t = 1 - \frac{n(n-1)}{2N}t + O(1/N^2) \approx e^{-\frac{n(n-1)}{2N}t} \tag{12}$$

times the probability that coalescence occurs in a given generation $(1 - p(n) \approx \frac{n(n-1)}{2N})$. We can thus write this probability distribution as

$$C(t|n) = p(n)^t(1 - p(n)) \approx \frac{n(n-1)}{2N} e^{-\frac{n(n-1)}{2N}t} \tag{13}$$

This is an exponential distribution with parameter $\lambda = \frac{n(n-1)}{2N}$,

$$p(t) = \lambda e^{-\lambda t} \tag{14}$$

In the present case the above assumptions are justified since N is expected to be large, on the order of $10^3 - 10^4$, and n is on the order of 10^1.

This distribution has a mean of $E[T(i)] = 1/\lambda = \frac{2N}{i(i-1)}$ which is thus the expected time for two of the i lineages to converge. We note that the distribution also has a large variance of $Var[T(i)] = 1/\lambda^2 = (\frac{2N}{i(i-1)})^2$.

A reasonable question is now what value to choose for N. Here we could opt for the global mean effective population size of $N_e = 20,000$ quoted in the PG lecture notes. For the MPhil class with $n = 19$ individuals, this corresponds to the first ancestor from which a single pair of MPhil students share inherited DNA at this locus having lived $T(19) = \frac{2 * 20000}{19(19-1)} = 116.96$ generations ago with an identical standard deviation. The human generation time may be taken as approximately 25 years (Scally & Durbin (2012) suggest a range of 20-30 years). This corresponds to the ancestor in question having lived 2429 years ago.

However, we could also consider the fact that the majority of the MPhil class (17 of 19) is of European ancestry where we can instead take $N_e \approx 1000$ (Melé et al. (2012)). Since λ is linear in N, this corresponds to a common ancestor from which two individuals share DNA living only $T(17) = \frac{2 * 1000}{17(17-1)} * 25 = 163.4$ years ago.

If on the other hand we pick two students at random and ask when the most recent common ancestor from which they share DNA lived, the expected answer is $T(2) = N$ which is 1,000-20,000 generations depending on the effective population size we use for our calculation.

If instead we interpret the phrase "the most recent common ancestor from which any pair of the MPhil students share inherited DNA" to mean the most recent ancestor from which any pair of students we pick will have shared DNA at this locus, we need the TMRCA of the whole population. For a sample of n individuals in a population of size N, this is

$$E[TMRCA] = \sum_{i=2}^{n} E[T(i)] = \sum_{i=2}^{n} \frac{2N}{i(i-1)} = 2N \sum_{i=2}^{n} [\frac{1}{i-1} - \frac{1}{i}] = 2N(1 - \frac{1}{n}) \tag{15}$$

Given the size of the 2018/2019 MPhil class of 19 students with $N_e = 20000$, we thus get

$$E[TMRCA] = 2 * 20000(1 - 1/19) = 37,895 \text{ generations} = 947,368 \text{ years} \tag{16}$$

This is of course an unreasonable estimate since this predates the out-of-Africa event by an order of magnitude. If instead we use $N_e = 1000$ we get $TMRCA = 47368$ years which seems more reasonable.

However, when considering the most recent common ancestor from which we share DNA, we need to consider not just haploid loci like the mitochondria but also take into account the fact that most of our DNA is diploid. In the Kingman coalescent, this is easily achieved by substituting $N \to 2N$ and then letting $n = 38$ since we are now considering both the maternal and paternall copy in each individuals such that there are 38 copies of each locus in the sample (Hudson 1990).

This gives a time to the most recent coalescence event of

$$E[T(i = 38)] = \frac{4N}{i(i-1)} = \frac{4 * 20,000}{38(38-1)} * 25 = 1422 \text{ years} \tag{17}$$

For $N_e = 20,000$ and $E[T(i = 38)] = 71$ years for $N_e = 1,000$. This first coalescence point will be the most recent common ancestor contributing DNA at this locus to two individuals with probability $p = 36/37$ while in the remaining $1/37$ cases, the first coalescence will be of two genes from the same individual (assuming random mating).

When considering the time of the most recent common ancestor from which the whole MPhil cohort share inherited DNA, we need to identify the first node in the coalescent that joins 19 individuals. This is unlikely to be the first node with 19 descendants since some of the coalesced haploid genomes can come from the same individual. Considering

random sampling of the haploid genomes, the probability that all 19 individuals are represented by K genes sampled from a total of 38 with $K \in [19, 38]$ is given by

$$p_{all}(K) = \frac{N_{all}(K)}{\binom{38}{K}} = \frac{2^{38-K}19!}{(38-K)!(K-19)!}\frac{K!(38-K)!}{38!} = \frac{2^{38-K}K!19!}{38!(K-19)!} \tag{18}$$

Where $N_{all}(K)$ represents the number of ways to draw K samples from 38 haploid genomes such that all 19 individuals are represented. This distribution has been plotted in figure 10.

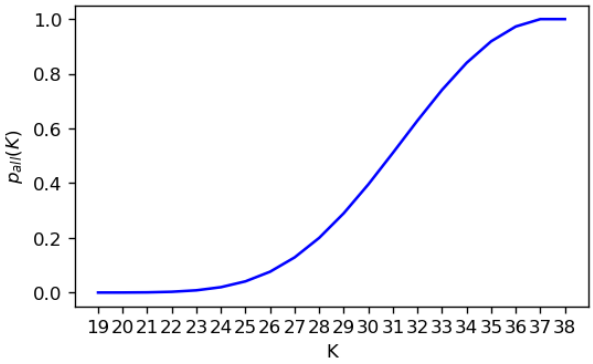


Figure 10: Probability that 19 indidividuals are represented when randomly sampling K genes from 38 haploid genomes.

We see that $p_{all}(K = 31) \approx 0.5$ and therefore for simplicity consider the most recent common ancestor of the full MPhil cohort to be the first ancestor of 31 leafnodes. Of course the time of this node does not easily follow from the analytical treatment of the coalescent above but can be extracted from simulations as wil be discussed later.

For a real system, in addition to population size and generation time we also need to consider the fact that there is a finite recombination rate. This leads to separate sections of the genome having non-identical genealogies which must all be considered. In this case, the time to the most recent common ancestor from which DNA is shared for a pair of students is the most recent coalescence event in any tree. The time to the most recent common ancestor from which the full cohort shares DNA at a single locus is the time to the first node joining 31 leafnodes in any tree.

The coalescent with recombination is not easily treated analytically but we can simulate it using msprime (Kelleher et al. 2016) which will be discussed further in part (c). In msprime, we do not simulate the full Ancestral Recombination Graph (ARG) since most common ancestor events will be between lineages with no overlapping ancestral material. Instead, only events leading to coalescence at loci of interest are simulated and stored.

In addition to the number of samples, $n = 38$, the population size N_e and the generation time $T_{gen} \approx 25$ years, we also need to estimate the recombination rate ρ for these simulations. In this report, we will use a default recombination rate of $\rho = 0.5\text{x}10^{-9}\,\text{bp}^{-1}\text{year}^{-1} = 40\,\text{generation}^{-1}$ as proposed by Scally and Durbin (2012).

b)

Additional factors affecting the estimate are factors violating the Wright-Fisher assumptions.

One such factor is changing population size, and this is an important factor since we know that the global population has increased significantly over the past 100,000 years. As discussed above, the effective population size N_e is usually used in population genetics rather than the actual population size. However, even the use of an appropriately time-averaged Ne for the process of interest can be erroneous since the time-variation of N_e can affect our coalescent both qualitatively and quantitatively.

The effects of varying N_e can be quantified by considering Tajima's D. Denoting the mutation rate by μ, we can calculate the expected genetic difference rate between individuals as
$E(observed\ difference\ rate) = \theta_\pi = 2N\mu$
θ_π thus provides one measure of genetic diversity in terms of the pairwise genetic differences between individuals.

We can also quantify genetic diversity based on the number of segregating sites S.

$$E[S] = 2\mu \sum_{i=2}^{n} iT(i) = \theta \sum_{i=1}^{n-1} \frac{1}{i} \approx \theta \log(n) \tag{19}$$

This allows us to define Watterson's theta as another measure of genetic diversity.

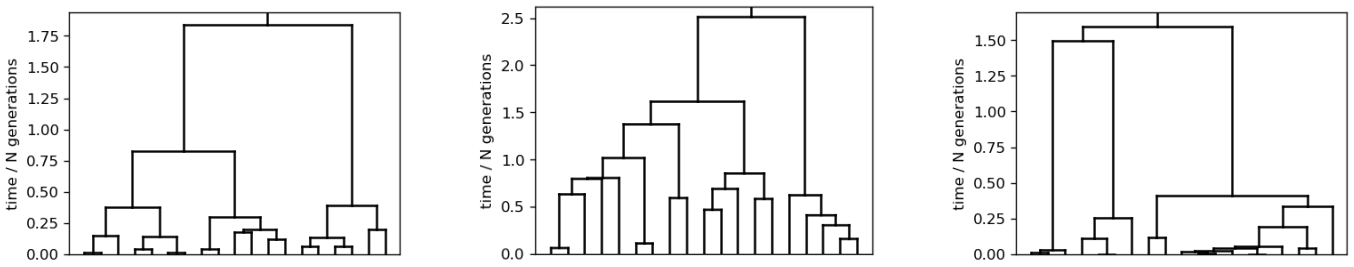
$$\hat{\theta}_S = \frac{S}{\sum_{i=1}^{n-1} \frac{1}{i}} \tag{20}$$

Under the Wright-Fisher assumptions $\theta_\pi = \theta_S$, and Tajima's D for quantifying deviations from these assumptions is therefore defined as

$$D = \frac{\hat{\theta}_\pi - \hat{\theta}_S}{\sqrt{(\hat{\theta}_\pi - \hat{\theta}_S)}} \tag{21}$$

This is the scaled difference between the mean number of pairwise differences between individuals and the number of segregating sites. It is therefore sensitive to the number of rare mutations and thus to N_e. If D is positive there is a deficiency of rare mutations in the current population, suggesting excess recent coalescences and thus a recent small Ne corresponding to a declining population

In this case, the most recent branches of the evolutionary tree would be shortened relative to the more distant branches which would be elongated. This might lead us to underestimate TMRCA if we base it on current population sizes and overestimate TMRCA if we base it on ancient population sizes. On the contrary if D is negative, there is an excess of rare mutations and recent branches of the coalescent will be longer than expected due to an increasing population size. In this case, we might overestimate TMRCA if we base it on current population sizes and underestimate TMRCA if we base it on ancient population sizes. These effects are illustrated in figure 11 for a set of simple simulations of haploid genomes.



(a) $D = 0$ (b) $D < 0$ (c) $D > 0$

Figure 11: Example trees for constant population size (D = 0), increasing population size (D < 0) and decreasing population size (D > 0) with $n = 19$. Trees were simulated by drawing sequential coalescence events from the exponential distribution in equation 13.

Since $E[TMRCA]$ is linear in N, the magnitude of the effects of changing population size could be rather large if we are considering long time periods over which the population size has changed significantly. However, if we use data for our effective population size that takes the changing population size into account, this should be mitigatable. This is hard to do in practice since our TMRCA events depend on Ne, but Ne depends on how much of human history we are considering.

Another violation of the Wright-Fisher model that may affect our estimate is population structure. The WF model assumes random mating in the population which is not true in structured populations. This approximation may be reasonable in isolated populations, but when considering larger populations it can make a significant difference. As already discusse in section (a), if we for example assume that all MPhil students are of European ancestry, this allows us to reduce the effective population size for the calculation by a factor of 10-20 compared to the case where we consider the global population, and this in turn affects the calculated TMRCA by an order of magnitude.

Additionally, if the cohort consists of people from different subpopulations, we might need further details in our model to take this into account. This is indeed the case in the present calculation since in the actual MPhil cohort, there are people of both Chinese and Indian descent, suggesting that we should consider the out-of-africa population more

broadly rather than merely the European population. There are no MPhil students of recent African descent and we therefore do not have to consider the global effective population size.

Population structure affects not just the TMRCA estimate but also the form of the genealogy. In the present case where the majority of our sample is of European descent and a minority of Asian descent, we expect a smaller TMRCA for the Asian than European subpopulation, and a TMRCA of the full cohort that is bounded from below by the time of migration separating the two subpopulations. Further complications arise in the case of non-zero migration between such subpopulations as in e.g. the island hopping model also described by Hudson. The effects of separated populations and migration can be taken into account by simulating the system with variable and separate N_e but this is hard to solve analytically. We therefore defer further discussions of this to section (c).

c)

To come up with an estimate and range of the time to the most recent common ancestor from which the MPhil students share inherited DNA, we use the msprime software by Kelleher et al.

We start by considering the pair of MPhil students that share the most recent common ancestor from which they both inherit DNA at a locus. From a), we know that it is likely that the first coalescence event will have taken place between two of the 17 students with European ancestry rather than the two students with Asian ancestry.

A simple guess at the most recent common ancestor to two MPhil students might thus be obtained from a simple consideration of the number of ancestors of an individual which is 2^g going g generations back. The total number of ancestors of the European part of the cohort is thus $17 * 2^g$ assuming no overlap between ancestors or generations. We can now consider the probability of a common ancestor as the probability of drawing two identical individuals when drawing $n = 17 * 2^g$ from a generation size of $N \approx 40$million, ignoring the probability that these are ancestors of the same individual. This probability is given by

$$P_{shared} = 1 - \frac{N!}{(N-n)!N^n} \tag{22}$$

This probability is plotted as a function of the number of generations in figure 12

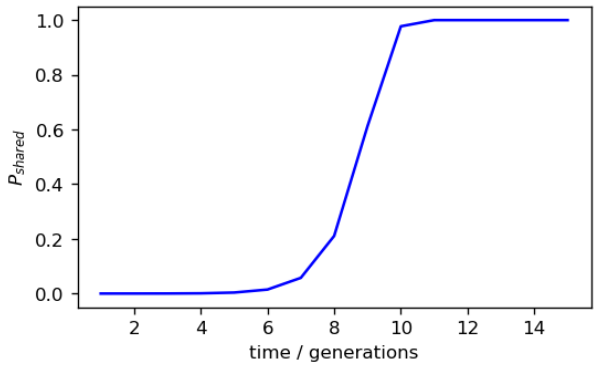


Figure 12: Probability that $n = 17 * 2^g$ ancestors g generations ago are not distinct.

We thus see that there is a higher than 50% probability that there is a shared ancestor 9 generations ago corresponding to 225 years ago, with a 90% confidence interval of 175-250 years. This estimate might have to be modified slightly if the ancestors of the North-American part of the MPhil cohort emmigrated from Europe more than 250 years ago. However, with an increasing number of ancestors it becomes increasingly unlikely that a common ancestor event also represents a coalescence event. In addition, it also becomes increasingly likely that the 2^g ancestors of a single individual are non-unique. Taken together, this suggests that our naive estimate might be an underestimate. We therefore return to the Kingman coalescent for an estimate that takes more details of the system into account.

The above considerations suggest that we are working with a relatively modern population, and it is thus unlikely to be appropriate with Ne=1000 as in section (a). Instead, we (somewhat arbitrarily) let Ne=100,000 (with Ne=1000, we get a most recent common ancestor living less than 1 generation ago). Running 10 simulations, this gives a most recent common ancestor from which at least two MPhil students inherited DNA at the same locus having lived 335 years ago with a range of 190 to 541 years. If we were to also vary parameters such as effective population size and recombination rate, the variability in our estimate would be even greater.

To find the most recent common ancestor that has contributed DNA to the entire MPhil cohort assuming a homogenous population with random mating, we instead run a simulation with Ne=1,000. We simulate the full tree with n=38 leaves and for each locus find the most recent common ancestor of all 19 individuals by finding the first node that is the ancestor of at least K leafnodes, with K drawn from equation 23 for each site.

This gives a value of 473 generations corresponding to 11825 years since the most recent common ancestor to the entire cohort. When repeating this simulation 50 times, we get a mean of 13038 and standard deviation of 1136 for our estimate assuming a homogenous and randomly mating population. The resulting distribution is shown in figure 13

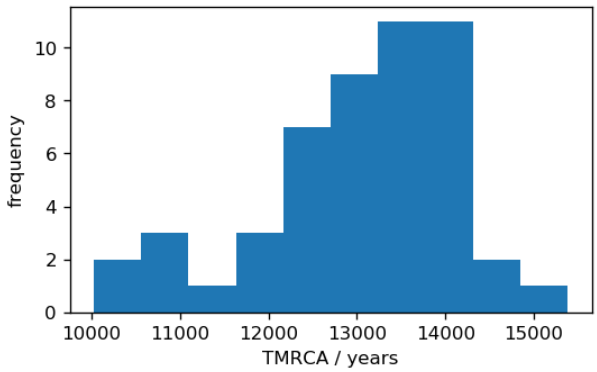


Figure 13: Distribution of TMRCA for 38 samples and Ne=1000.

However, to provide a range for our estimate we should include additional uncertainty in the parameters used and therefore repeat our simulation with $Ne \in [500, 3000]$, $t_{gen} \in [20, 30]$ and $\rho \in [0.25 * 10^{-9}, 1 * 10^{-9}]$. We run 100 simulations drawing all three parameters uniformly at random in the given ranges for each simulation and plot a histogram of the result in figure 14

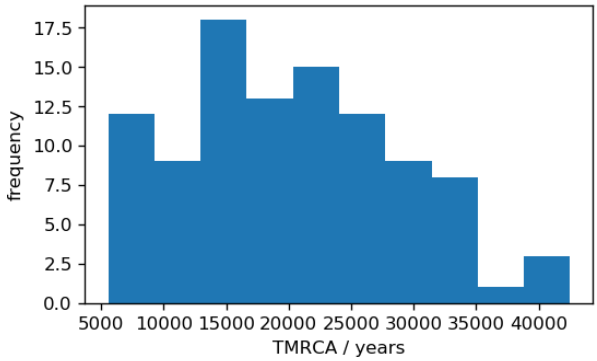


Figure 14: Distribution of time to the most recent ancestor from which all MPhil students share DNA at a single locus, drawing parameters uniformly at random as described in the main text.

This gives us an expected value of 20,273 years, standard deviation of 8723 years and 90% confidence interval of 7720 to 34993 years. We could also have considered the possibility that all pairs of MPhil students will share DNA from a single ancestor, but that different pairs of students share DNA at different loci. However, this is considered beyond the scope of the present assignment.

Finally we attempt to take into account population structure by considering Asian and European populations to have been completely separated since the European-Asian divergence 40,000-80,000 years ago (Scally & Durbin). It is clear from our previous simulations and calculations that at this point many loci in the current European MPhil cohort will have a single ancestor, and similarly the Asian MPhil cohort will likely have a single ancestor at many genetic loci.

The time to convergence for any single locus prior to the migration event will be on the order of $2N \approx 4000$ generations with an exponential distribution as discussed in (a). From our msprime simulations we expect on the order of 10,000

independently segregating loci with a single common ancestor 50,000 years ago. We therefore find the expected time to coalescence prior to migration as the 10^{-4} percentile of this exponential distribution, giving a value of $t_{coal} = -\ln(\frac{1-1/10^4}{2(2-1)/4N}) = 0.2\text{generations} = 5years$. This suggests that the time to coalescence between the Asian and European MPhil lineages for the first locus to coalesce will be neglible after the migration event.

We verify this by running an msprime simulation with a migration event 50,000 years ago and find the TMRCA of the entire MPhil cohort at a single locus to have lived 2000.1 generations ago corresponding to 50,002.5 years. A genealogy for an example locus from this simulation is given in figure 15

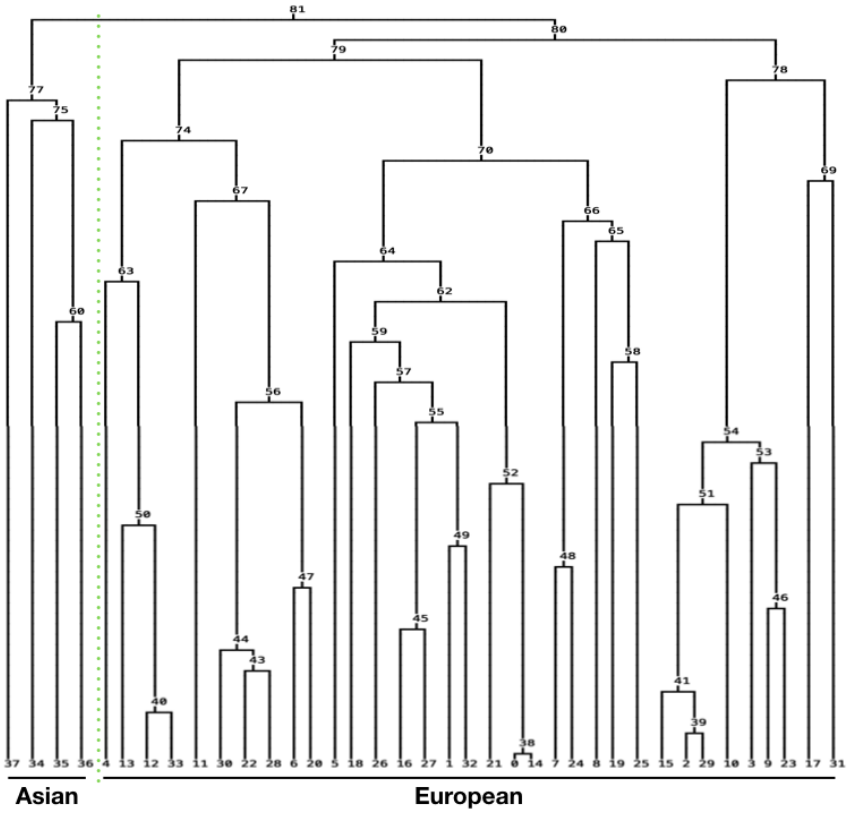


Figure 15: Example genealogy generated by msprime with a migration event 2000 generations ago separating the Asian samples (34, 35, 36, 37) from the European samples (0-33)

Hence in reality, the most recent ancestor contributing a section of DNA to the entire MPhil cohort is likely to have lived $\approx 50,000$ years ago with a range of 40,000-80,000 years as dictated entirely by the uncertainty in the European-Asian divergence. A low rate of migration between these populations since 50,000 years ago means that there is in practice a finite probability that the TMRCA is lower if this has caused mixing of the lineages.

In summary my best estimate at a most recent common ancestor that has contributed DNA to any two MPhil students lived 335 years ago with a range of 190 to 541 years while the most recent common ancestor that has contributed DNA to all MPhil students under the hypothesis of a homogenous population lived 20,273 years ago with a range of 7720 to 34,993 years. Taking into account the Asian-European divergence, my best estimate of the time to the most recent ancestor contributing a genomic segment to all MPhil students is 50,000 years with a range of 40,000 to 80,000 years corresponding to the time and uncertainty of the European-Asian divergence. If we pick a specific pair of students and want the TMRCA contributing shared genetic material to these two specific individuals, this is the expectation value of an msprime simulation with n=2 which yields 11378 years and a range of 2899 to 24680 years over 10 simulations if they are both of European descent, while we also expect a coalescence time of 40,000-80,000 years in this case if one is of European and one of Asian descent.

However, as the above analyses show, these estimates are strongly dependent on Ne, population structure, ρ and the mean generation time. In particular, the effective population size and population structure of the relevant subpopula-

tions have been very hard to estimate accurately, but while the estimates used in the present assignment may not be the most appropriate, they do at least illustrate how coalescent theory can be used to investigate common ancestor events for genetic loci in the absence and presence of recombination.

R. R. Hudson "Gene genealogies and the coalescent process" *Oxford Surverys in Evolutionary Biology* (1991)
J. Kelleher et al. "Efficient Coalescent Simulation and Genealogical Analysis for Large Sample Sizes" *PLOS Computational Biology* (2016)
M. Melé et al. "Recombination Gives a New Insight in the Effective Population Size and the History of the Old World Human Populations" *Molecular Biology and Evolution* (2012)
A. Scally & R. Durbin "Revising the human mutational rate: implications for understanding human evolution" *Nature Reviews Genetics* (2012)

1.4 Neanderthal introgression

a)

First we note a few key features of the problem at hand. Firstly, mitochondria are uniparentally, maternally inherited and we can thus consider the mitochondrial origin as a single trait analogous to the haploid single-locus case above. Secondly, we assume that there is no selective pressure such that mitochondria are randomly inherited. This means that if the population has a fraction c of female individuals with Neanderthal mitochondria and $1-c$ female individuals with modern human mitochondria, there is a probability $c\ (1-c)$ that a given offspring receives Neanderthal (modern human) mitochondria. We also assume random mating, no population structure, non-overlapping generations and ignore mutations.

Following the argument of Nordborg (1998), when investigating a sample of n modern humans we consider the distribution $g_{nk}(t = 60,000\text{years})$ that our n current samples had k ancestors t years ago. For a given value of k, we can then calculate the probability that none of these ancestors had Neanderthal DNA as $p_k = (1-c)^k$ given the assumptions above. If none of the ancestors of our current samples had Neanderthal mitochondria, we will see no Neanderthal mitochondrial DNA in our $n = 1,000,000$ modern humans and vice versa. Finally this allows us to calculate the probability that none of the n modern humans have Neanderthal mitochondria as

$$P(0|c,n,t) = \sum_{k=1}^{n} g_{nk}(t)(1-c)^k \tag{23}$$

From Tavaré (1983) $g_{nk}(t)$ is given by

$$g_{nk}(t) = \sum_{i=k}^{n} \rho_i^0(t) \frac{(2i-1)(-1)^{i-k}\, k_{(i-1)}\, n_{[i]}}{k!\,(i-k)!\, n_{(i)}} \tag{24}$$

Where $\rho_i^0(t) = \exp\left(-i(i-1)t/(2N)\right)$ and we measure time in generations. $a_{(j)}$ and $a_{[j]}$ are defined as $a\,(a \pm 1)\,...\,(a \pm (j-1))$

To verify our implementation we first reproduce the results of Nordborg for a fixed population size with mixing times (t_m) of 100,000 and 30,000 years (figure 16). We see that in both cases the probability of observing no Neanderthal mitochondria in a modern sample of size $n = 986$ with a female population size of $N = 3400$ and a generation time of 20 years decreases monotonously from 1 to 0 as c increases from 0 to 1.

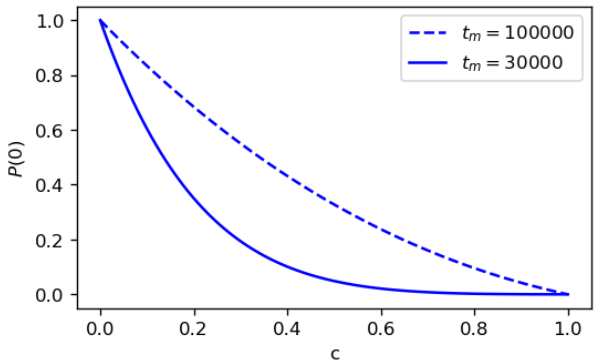


Figure 16: Probability of oberserving no Neanderthal DNA in a modern sample of size $n = 986$ as a function of c for two different mixing times (reproduced from Nordborg 1998).

We now return to the problem given in the assignment with $t = 60,000$ years. We assume a generation time of 25 years as in question 3 and thus let $t = 2400$ generations. Using equations 23 and 24 with $c = 0.01$, $n = 1,000,000$ and $t = 2,400$ with a population size of 20,000 and thus a female population size of N=10,000, we get $P(0|N = 10,000) = 0.917$. This suggests a relatively small probability of observing Neanderthal DNA in modern humans under these hypotheses. Our calculation is thus consistent with a hypothesis of population mixing 60,000 years ago. If instead we let N=3400 as in Nordborg, we get $P(0|N = 3400) = 0.968$, suggesting an even higher likelihood of not observing Neanderthal mitochondria in our modern sample. To further investigate the relationship between N and $P(0|N)$, we plot $P(0)$ as a function of N in figure 17.

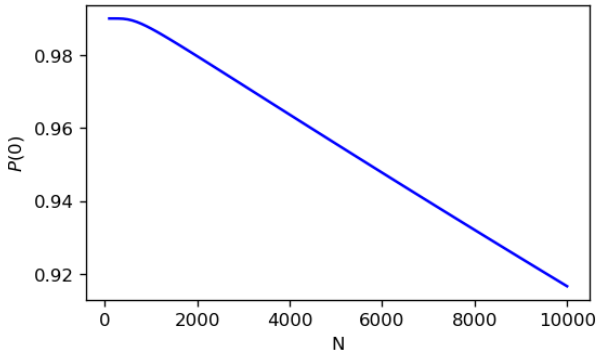


Figure 17: Probability of oberserving no Neanderthal DNA in a modern sample of size $n = 1,000,000$ as a function of female population size N over t=2400 generations.

We see that after an initial plateau, $P(0)$ is approximately linearly decreasing in N. We can understand this intuitively since the expected number of ancestors of our sample increases with increasing N, and this in turn increases the probability that at least one of these had Neanderthal DNA, decreasing $P(0)$. The initial plateau at $P(0) = 0.99$ is the range of N for which we expect only a single ancestor of our sample.

However, we also note that all probabilities are relatively high. In light of section 1.3, this is understandable since we expect only a few ancestors 60,000 years ago even for N=10,000, and these are unlikely to have had any Neanderthal DNA with c=0.01. We therefore conclude that with $c = 0.01$, the observation that none of our sample has Neanderthal mitochondria is a likely scenario.

b)

If we let $c = 0.10$, the equivalent probabilities are $P(0|N = 10000, c = 0.10) = 0.407$ and $P(0|N = 3400, c = 0.10) = 0.718$. In this case, the term $(1 - c)^k$ in equation 23 becomes small much faster, resulting in a lower probability that no ancestors had Neanderthal mitochondria. To further investigate the effect of c on $P(0)$, we follow the example of Nordborg and plot in figure 18 $P(0)$ as a function of c for both $N = 10,000$ and $N = 3400$.

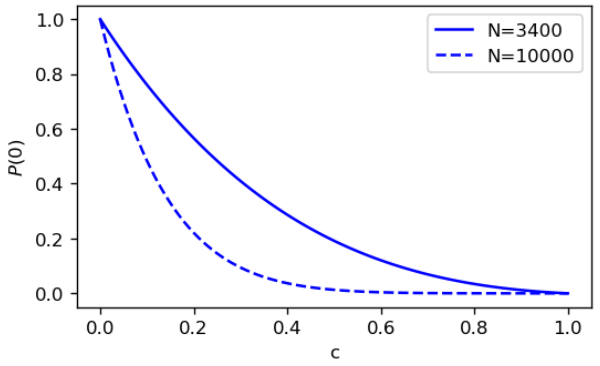


Figure 18: Probability of oberserving no Neanderthal DNA in a modern sample of size $n = 1,000,000$ as a function of c after t=2400 generations.

As expected, we see that $P(0|N = 10,000)$ very rapidly drops to 0 with c since with $N = 10,000$ the expected number of ancestors 2,400 generations ago is $\langle g_{10000,k} \rangle = 8.67$ and $(1 - c)^{8.67}$ rapidly goes to zero. In contrast $\langle g_{3400,k} \rangle = 3.19$ and the N=3400 curve therefore goes to zero more slowly. However, even with a population size of 10,000 and c=0.10, we cannot reject the hypothesis of population mixing 60,000 years ago on the basis of these simulations.

The reason why the calculated $P(0)$ values are quite high even with large numbers of modern people is that the entire population is likely to have only few ancient ancestors at the time of mixing as described above. This in turn results in $P(0)$ flattening as $n \to \infty$ as illustrated in figure 19.

The limiting value of $P(0|n)$ as $n \to \infty$ for a given c, N is the probability that mitochrondrial DNA is completely lost from the population due to genetic drift (Tavaré 1983). We thus see that as soon as the sample size exceeds $\approx 1,000$

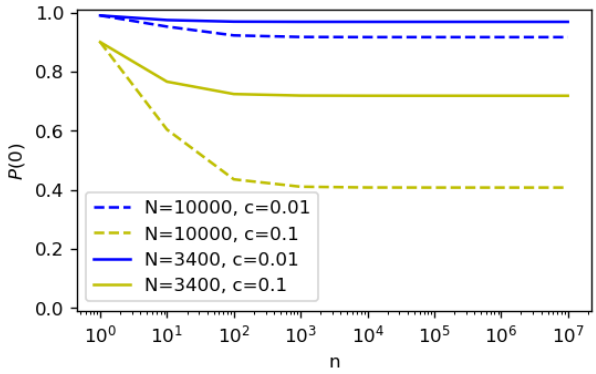


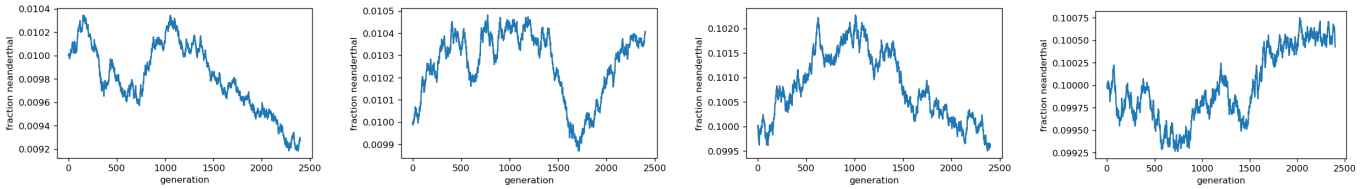
Figure 19: Probability of oberserving no Neanderthal DNA in a modern sample as a function of n with t=2400 generations and various N and c values.

the probability that none of the sample contains Neanderthal mitochrondria is indistinguishable from the probability that the Neanderthal mitochrondria have been lost from the entire population, and increasing the modern sample size thus does not improve the statistical power of our analysis or our ability to determine whether a mixing event has taken place. We also note that in many of these cases our assumptions are obviously violated since we are sampling upwards of 1,000,000 individuals from a constant population of size 10,000, and this may affect our results at high n.

To investigate the time-dependency of the loss of Neanderthal mitochondria in the population, we can run an explicit simulation of the number of Neanderthal mitochondria present in each generation. We do this by drawing at each generation the number of female individuals with Neanderthal mitochondria (n_i) from a binomial distribution

$$n_{i+1} \sim \text{binomial}(Ne, n_i/Ne) \tag{25}$$

We proceed to do this throughout the 2400 generations and plot the mean fraction of Neanderthal mitochondria over 10,000 simulations as a function of time in figure 20.



(a) Ne=3400, c=0.01 (b) Ne=10,000, c=0.01 (c) Ne=3400, c=0.10 (d) Ne=10,000, c=0.10

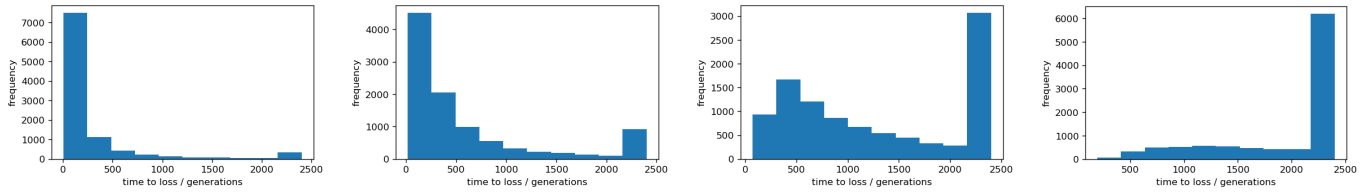
Figure 20: Mean fraction of Neanderthal mitochondria over 10,000 simulations as a function of time since mixing.

Interestingly, we see that even though the proportion of Neanderthal mitochondria in any single simulation is very variable, the mean fraction of Neanderthal mitochrondria over 10,000 simulations remains approximately constant throughout the simulation. This is of course because $\langle n_{i+1} \rangle = n_i$ at every generation.

Consistent with the above considerations of allele loss, when looking at individual simulations we often observe complete loss of Neanderthal mitochrondria and thus fixation of modern human mitochondria. We show this in figure 21 by plotting histograms of the time to loss of the last Neanderthal mitochrondrion.

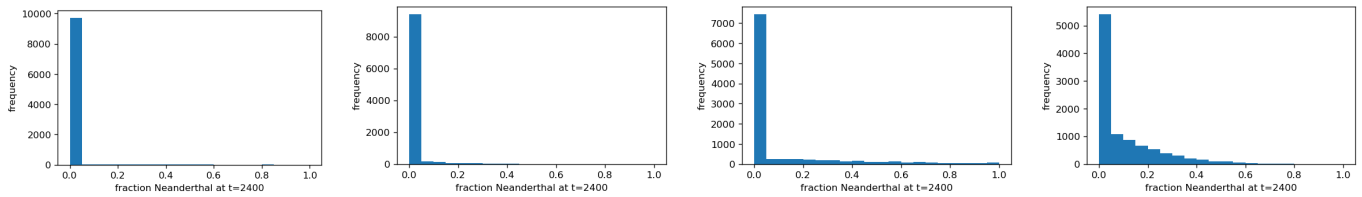
Finally we also plot in figure 22 histograms of the proportion of Neanderthal mitochondria at the end of each simulation. We see that in the simulations where these are retained, they have often achieved relatively high frequencies which prevents loss by genetic drift. However, fixation of Neanderthal mitochrondria is rare with any of the parameter sets used.

From figure 21 we also see that the proportion of simulations where Neanderthal mitochondria are lost are very similar to what we arrived at using the approach of Nordborg. Given our large sample size of $n = 10^6$ and finite $c(t = 2400)$ (figure 22), we can ignore the neglible probability that Neanderthal mitochrondria are retained in the population but not present in the sample as also discussed above. A direct comparison is given in table 4 where we see that the results



(a) Ne=3400, c=0.01 P(0)=0.9698 (b) Ne=10,000, c=0.01 P(0)=0.9132 (c) Ne=3400, c=0.10 P(0)=0.7186 (d) Ne=10,000, c=0.10 P(0)=0.4079

Figure 21: Histogram of the time at which the last Neanderthal mitochondrion is lost from the population. Bars at 2400 represent the number of simulations where Neanderthal mitochondria are retained in the population. P(0) is 1 minus this fraction and is given under each histogram.



(a) Ne=3400, c=0.01 (b) Ne=10,000, c=0.01 (c) Ne=3400, c=0.10 (d) Ne=10,000, c=0.10

Figure 22: Histograms of the fraction of Neanderthal mitochondria at the end of the simulation.

from the simulation and equation 23 agree to two decimal places. This suggests that we can use such simulations with high accuracy to model more complex scenarios that are not easily treated analytically.

Ne	c	calculated P(0) (Nordborg)	simulated P(0)
3400	0.01	0.968	0.970
10,000	0.01	0.917	0.916
3400	0.10	0.718	0.719
10,000	0.10	0.407	0.408

Table 4: Predicted P(0) using Nordborg's calculation and our simple simulation.

Given the consistency of table 4, we can use similar simulations to easily investigate the effect of changing population size on P(0). In this case, we again use similar parameters to Nordborg, letting Ne increase from 3400 to 10^9 over 2000 generations starting 50,000 years ago according to $Ne = \max(3400, 3400 \exp(0.00630 * (t - 399))$. The resulting probabilities are given in table 5 for c=0.01 and c=0.10 together with the corresponding probabilities for constant population sizes.

We thus see that with the more realistic increasing population size, it becomes significantly less likely that all Neanderthal mitchondria are lost from the population. However, P(0) is still greater than 0.25 for c=0.10 suggesting that even with this higher mixing ratio we still cannot with high probability discard the hypothesis that *Homo sapiens* and Neanderthals mixed 60,000 years ago followed by loss of Neanderthal mitochrondria.

Finally we plot P(0) as a function of c for the exponentially increasing population, running 10,000 simulations for each value of c (figure 23).

We see that while there is a substantial probability of loss for $c < 0.20$, this probability decreases to $P(0) < 0.01$ for $c > 0.33$. On the basis of our simulations, we thus conclude that there is unlikely to have occurred a mixing event between Neanderthals and modern humans with a mixing ratio of Neanderthals higher than 20%, but a mixing ratio of 1-2% as suggested by autosomal DNA analyses is very plausible.

c)

Other biological factors influencing the estimate include selection, mating bias and population structure.

One might expect that if the majority of the population after mixing consisted of *Homo sapiens* and the majority of nuclear DNA therefore had modern human origins, there would be a selective pressure against Neanderthal mitochon-

Ne	c	simulated P(0)
3400	0.01	0.970
10,000	0.01	0.916
exponential	0.01	0.876
3400	0.10	0.719
10,000	0.10	0.408
exponential	0.10	0.278

Table 5: Predicted P(0) for constant and exponentially growing population sizes.

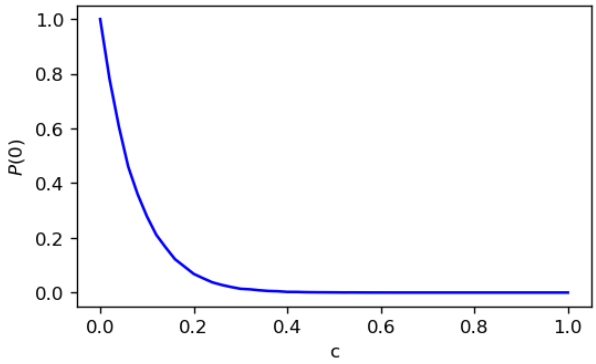


Figure 23: Probability of oberserving no Neanderthal DNA in a modern sample as a function of c with exponential popoulation growth as described in the main text.

dria. This is the case since a number of metabolic processes depend on coordination between nuclear and mitochondrial proteins, and the expression levels of these must therefore be appropriately balanced. This selective pressure would lead to a higher probability of lines with Neanderthal mitochondrial ancestry dying out, and would therefore increase P(0), the probability of not observing any Neanderthal mitochondria in the modern sample. In the simulated model, this selective pressure could for example be included by using an augmented c that is not merely the proportion of the population with Neanderthal mitochondria but also takes into account the probability that an individual with Neanderthal mitochondria will achieve reproductive success.

In the model, we have also assumed random mating which may not be the case. We might expect biased mating of Neanderthals with Neanderthals and *Homo sapiens* with *Homo sapiens*. In this case, the full population could be modelled as two sub-populations with migration, and if modern humans are descended from the *Homo sapiens* sub-population, this would again decrease the probability of observing Neanderthal mitochondria in the modern sample and thus increase P(0) since the effective c would be smaller than the full Neanderthal mixing ratio.

This already touches upon the topic of population structure where we have assumed a single homogenous population in the above which may not be the case. The presence of subpopulations would for example increase the probability of loss of Neanderthal mitochondria by reducing the effective population size even if there is random mating within the subpopulations. From Nordborg and the considerations in section 1.3 and part (b), we also know changing population size to be important for the coalescent and the probability of allele loss. In the final section of part (b), we have attempted to take into account the growing global population instead of using an average effective population size, but this could also be modelled in more detail. .

In summary, the proposed scenario of population mixing 60,000 years ago appears to be consistent with the observation that no modern humans contain Neanderthal mitochondria based on calculations and simulations with a range of population sizes and mixing ratios. In contrast, Neanderthal DNA is still found in the nuclear genome where the coalescence timescale is slower, loss from genetic drift is slower, and recombination allows for decoupling of different loci.

M. Nordborg "On the Probability of Neanderthal Ancestry" *American Journal of Human Genetics* (1998)
S. Tavaré "Line-of-Descent and Genealogical Processes and Their Application in Population Genetics Models" *Theoretical Population Biology* (1984)

Appendix

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Code for manipulating and plotting trees
"""

import matplotlib.pyplot as plt
import numpy as np
import networkx as nx


def topological_sort(G, relabel = False):
    '''function for sorting the nodes of a tree such that
    all parent nodes in the resulting list have a lower index than their
    child nodes'''
    nodes = [int(n) for n in G.nodes]
    inds = {}
    for i, n in enumerate(nodes): inds[n] = i #start with default order

    done = False
    while not done:
        #repeat until the nodes are sorted
        done = True
        #run through all pairs of nodes and swap indices if wrong order
        for i in nodes:
            for j in nodes:
                if j in G.neighbors(i):
                    a = inds[i]
                    b = inds[j]
                    if a > b:
                        inds[i] = b
                        inds[j] = a
                        done = False

    for node, ind in inds.items():
        nodes[ind] = node

    if not relabel: return nodes
    else: #relabel nodes as described in assignment
        nodes.reverse()
        for i, n in enumerate(nodes):
            print(n, i)
            G.node[n]['newlabel'] = str(i+1)

        nodes.reverse()
        return nodes, G




def parse_tree(fname = "tree.dat"):
    '''parse data format given in assignment'''
    G = nx.DiGraph() #use the nx module to store information

    f = open(fname, 'r')
    for line in f:
```

```
        n, d1, l1, d2, l2 = line.split()
        G.add_node(int(n))
        G.add_edge(int(n), int(d1), length = float(l1))
        G.add_edge(int(n), int(d2), length = float(l2))
    f.close()

    #topo = topological_sort(G)
    topo, G = topological_sort(G, relabel = True)

    G.root = topo[0]
    G.node[G.root]['rootlength'] = 0
    G.node[G.root]['tier'] = 0


    for node in topo[1:]: #calculate length below root node; start from top
        pred = list(G.predecessors(node))[0]
        G.node[node]['rootlength'] = G[pred][node]['length'] + G.node[pred]['rootlength']
        G.node[node]['tier'] =  G.node[pred]['tier']+1

    topo.reverse() #reverse list

    for node in topo: #calculate number of nodes below each node; start from bottom
        ntot = 0
        nleaves = 0
        nint = 0
        for daughter in G.neighbors(node):
            ntot += G.node[daughter]['ntot']+1 #also add for this node
            nleaves += G.node[daughter]['nleaves']
            nint += G.node[daughter]['nint']
            if len(list(G.neighbors(daughter))) == 0: nleaves += 1 #daughter is a leaf
            else: nint += 1 #daughter is internal


        G.node[node]['ntot'] = ntot #total nodes below this node
        G.node[node]['nleaves'] = nleaves #leaves below this node
        G.node[node]['nint'] = nint #internal nodes below this node

    return G

def print_tree(G, relabel = False):

    nodes = np.array([n for n in G.nodes])
    ntots = np.array([G.node[n]['ntot'] for n in G.nodes])
    args = np.argsort(ntots)

    nodesn = nodes[args]
    ntots = ntots[args]

    print('\n')
    if relabel:
        for n in nodesn:
            lab = G.node[n]['newlabel']
            print(lab,"&", n, '&', G.nodes[n]['ntot'], '&', G.nodes[n]['nleaves'],
                  '&', G.nodes[n]['nint'], '&', G.nodes[n]['rootlength'], '\\'+'\\')

    else:
        for n in nodesn:
```

```
            print(n,"&", G.nodes[n]['ntot'], '&', G.nodes[n]['nleaves'],
                  '&', G.nodes[n]['nint'], '&', G.nodes[n]['rootlength'], '\\'+'\\')
    print('\n')

    ls = np.array([G.node[n]['rootlength'] for n in G.nodes])
    args = np.argsort(-ls)
    if relabel:
        for n in nodesn: #print in same format as assignment; default order
            lab = G.node[n]['newlabel']
            sucs = list(G.successors(n))
            if len(sucs) > 0:
                print('{:5}'.format(str(lab)),
                      '{:5}'.format(str(G.node[sucs[0]]['newlabel'])),
                      '{:5}'.format(str(G.edges[(n, sucs[0])]['length'])),
                      '{:5}'.format(str(G.node[sucs[1]]['newlabel'])),
                      '{:5}'.format(str(G.edges[(n, sucs[1])]['length'])))

    if relabel:
        print('\n')
        for n in [1,3,4,5,7,10,12]: #print in same format and order as assignment
            lab = G.node[n]['newlabel']
            sucs = list(G.successors(n))
            if len(sucs) > 0:
                print('{:5}'.format(str(lab)),
                      '{:5}'.format(str(G.node[sucs[0]]['newlabel'])),
                      '{:5}'.format(str(G.edges[(n, sucs[0])]['length'])),
                      '{:5}'.format(str(G.node[sucs[1]]['newlabel'])),
                      '{:5}'.format(str(G.edges[(n, sucs[1])]['length'])))



def draw_tree(G, relabel = False):

    #topo = list(nx.topological_sort(G))
    topo = list(topological_sort(G))


    xs = {topo[0]: 0}

    plt.figure()
    if relabel:plt.text(-0.05, 0.08, str(G.node[G.root]['newlabel']))
    else: plt.text(-0.05, 0.08, str(G.root))

    slopes = [0.5, 0.4,0.3,0.2,0.1]

    for node in topo:
        y1 = -G.node[node]['rootlength']
        x1 = xs[node]
        daughters = list(G.neighbors(node))
        if len(daughters) == 1:
            y2 = -G.node[daughters[0]]['rootlength']
            x2 = x1
            xs[daughter[0]] = x2
            plt.plot([x1, x1], [y1, y2], 'b-')
            plt.text(x2+0.1, y2, str(daughters[0]))
        elif len(daughters) == 2:
            newxs = [xs[node]-1, xs[node]+1]
```

```
            newxs = [-1, 1]
            for i, daughter in enumerate(daughters):
                y2 = -G.node[daughter]['rootlength']
                slope = -1/(y2+1.5)
                x2 = x1+(y1-y2)*newxs[i]*slope#slopes[G.node[node]['tier']]
                #x2 = newxs[i]
                xs[daughter] = x2

                plt.plot([x1, x2], [y1, y2], 'b-')
                if relabel: s = G.node[daughter]['newlabel']
                else: s = str(daughter)
                if len(list(G.neighbors(daughter))) == 0:
                    plt.text(x2-0.05, y2-0.5, s)
                elif newxs[i] > 0:
                    plt.text(x2+0.03, y2, s)
                else:
                    plt.text(x2-0.16, y2, s)

    plt.xticks([], [])
    plt.ylim(-9, 0.5)
    if relabel: plt.savefig("figures/tree_relabelled.png")
    else: plt.savefig("figures/tree.png")
    plt.show()

G = parse_tree() #parse input file
print_tree(G) #print table of nodes and lengths
print_tree(G, relabel = True) #print relabelled table
#draw_tree(G) #plot tree
#draw_tree(G, relabel = True) #plot relabelled tree

#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Code for time-dependent selection questions
"""

import random
import numpy as np
import matplotlib.pyplot as plt
from numpy.random import poisson



def pfix(f, N=100):
    '''returns the probability of fixation given values of f and N'''
    return (1-np.exp(-2*f))/(1-np.exp(-2*N*f))


def plot_summary(ts, n1s, fixed, fname='test', xlim = [-0.05,0.1]):
    '''plot a series of graphs given values of n1 as a function of t'''

    #plot timeseries
    plt.figure()
    plt.plot(ts, n1s)
    plt.ylim(0,1000)
    plt.xlabel('time')
    plt.ylabel('$n_1$')
    plt.savefig('figures/'+fname+'_selection_trajec.png')
```

```
    plt.show()

    #plot histograms
    bins = np.linspace(min(fixed), max(fixed), 100)
    plt.figure()
    plt.hist(fixed, bins=bins)
    plt.plot([0,0], [0,200], 'k:', linewidth = 1)
    plt.xlim(xlim[0], xlim[1])
    plt.xlabel('$f$')
    plt.ylabel('frequency')
    plt.savefig('figures/'+fname+'_selection_hist2.png')
    plt.show()

    '''
    also plot predicted distributions
    ns = [n1, 1000-n1]
    xs = bins#np.linspace(-0.02, 0.08, 100)
    ps = [6*pfix(x) for x in xs]
    exps = [np.exp(-lambd*abs(x)) for x in xs]
    counts = [ ns[int((1+np.sign(x))/2)]/1000 for x in xs]

    combs = [20*exps[i]*ps[i]*counts[i] for i in range(len(xs))]

    plt.figure()
    plt.plot(xs, ps, 'r--')
    plt.plot(xs, exps, 'b--')
    plt.plot(xs, counts, 'c--')
    #plt.plot(xs, combs, 'g--')
    plt.legend(['$p_{fix}(f)$','$e^{-\lambda |f|}$','allele count'])
    plt.savefig('figures/'+fname+'_selection_dists.png')
    plt.show()

    plt.figure()
    plt.plot(xs, combs, 'g--')
    plt.savefig('figures/'+fname+'_combined_dists.png')
    plt.show()
    '''

def get_dists(xs, lambd=100):
    '''return theoretical distribution'''
    n1 = 820 #calculate analytically
    ns = [n1, 1000-n1]

    ps = np.array([pfix(x) for x in xs])
    ps /= np.amax(ps)

    exps = np.array([np.exp(-lambd*abs(x)) for x in xs])
    exps /= np.amax(exps)

    counts = np.array([ ns[int((1+np.sign(x))/2)]/1000 for x in xs])

    return ps, exps, counts

def plot_dists():
    '''plot expected n1 distributions in the different scenarios'''
    xs = np.linspace(-0.1, 0.1, 100)
    ps, exps, counts = get_dists(xs)
```

```
plt.figure()
plt.plot(xs, ps, 'r--')
plt.plot(xs, exps, 'b--')
plt.plot(xs, counts, 'c--')
#plt.plot(xs, combs, 'g--')
plt.ylim(0,1.05)
plt.xlim(np.amin(xs), np.amax(xs))
plt.legend(['$p_{fix}(f)$','$e^{-\lambda_|f|}$','allele_count'])
plt.xlabel('$f$')
plt.savefig('figures/selection_dists.png')
plt.show()


xs = np.linspace(-0.05, 0.1, 101)
ps, exps, counts = get_dists(xs)
combs = np.array([exps[i]*ps[i]*counts[i] for i in range(len(xs))])
combs /= np.amax(combs)
plt.figure()
plt.plot(xs, combs, 'g--')
plt.ylim(0,1.05)
plt.xlim(np.amin(xs), np.amax(xs))
plt.xlabel('$f$')
plt.ylabel('frequency')
plt.savefig('figures/all_combined_dists.png')
plt.show()

xs = np.linspace(-0.03, 0.1, 101)
ps, exps, counts = get_dists(xs)
combs_non = np.array([exps[i]*ps[i] for i in range(len(xs))])
combs_non /= np.amax(combs_non)
plt.figure()
plt.plot(xs, combs_non, 'g--')
plt.ylim(0,1.05)
plt.xlim(np.amin(xs), np.amax(xs))
plt.xlabel('$f$')
plt.ylabel('frequency')
plt.savefig('figures/no_n_combined_dists.png')
plt.show()


#with fixed fs, need to switch forward to switch back, so actual
#switching interval is given by the mean switching intervals of the two times frequency
#n1/n0 is given by the ratio of switching rates integrated over space
xs = np.linspace(-0.035, 0.035, 100)
L = len(xs)
ps, exps, counts = get_dists(xs)
sym = np.array([exps[i]/np.mean([1/ps[i], 1/ps[L-i-1]]) for i in range(len(xs))])
sym /= np.amax(sym)
plt.plot(xs, sym, 'g--')
plt.ylim(0,1.05)
plt.xlim(np.amin(xs), np.amax(xs))
plt.xlabel('$f$')
plt.ylabel('frequency')
plt.savefig('figures/sym_combined_dists.png')
plt.show()
```

```
def run_sim(N=100, lambd=100, changing = False, tlim=3000000,
            resample = False, fname='test', tau=5*10**(-6), xlim = [-0.05,0.1]):
    '''run simulation as described in the assinment.
    resample specifies which scenrio we consider, tau specifies the rate
    of change of the environment. tlim specifies the length of simulation'''

    change = 0 #start in default environment
    tswitch = poisson(1/tau) #timescale of switching

    f1s = np.random.exponential(1/lambd, 1000) #get an intial fitness at each locus
    f0s = -f1s #non-preferred allele

    states = np.ones(1000) #all loci start in state 1
    n1 = 1000 #we have 1000 loci in state 1

    t = 0 #initialize
    ts = [t]
    n1s = [n1]
    fixed = []

    while t < tlim:
        i = random.randint(0,999) #mutate random locus
        old = states[i]
        if old == 0: f = f1s[i] #switch to allele 1
        else: f = f0s[i] #switch to allele 0
        if resample: #draw new fitnesses
            if old == 0: f = np.random.exponential(1/lambd)
            else: f = -np.random.exponential(1/lambd)
            if change == 1: f = -f #change in environment

        p = pfix(f) #probability of fixation
        if random.random() < p: #fix allele
            new = 1-old
            states[i] = new
            n1 += new-old
            if t > 500000: #start collecting distribution after 500000 timesteps
                fixed.append(f)

        n1s.append(n1)
        dt = min(10, np.abs( 2*np.log(N-1)/(919*f) )) #fixation time
        t += dt
        ts.append(t)

        if changing:
            if t > tswitch:
                #change environment
                f1s = -f1s
                f0s = -f0s
                change = 1-change
                tswitch += poisson(1/tau) #update switching time

    plot_summary(ts, n1s, fixed, fname=fname, xlim = xlim)
```

```
#plot_dists() #plot expected distributions

#run simulations with constant environment
#run_sim(fname='constant', xlim = [-0.035, 0.035]) #scenario 2
#run_sim(resample = True, fname='resample', xlim=[-0.05,0.1]) #scenario 1

#run simulations with changing environments
#run_sim(fname='changing', changing=True, xlim = [-0.03, 0.1])
#run_sim(fname='changing_resample', changing=True, resample = True, xlim=[-0.03,0.1])



#run_sim(fname='tau5e7', changing=True, tau=5*10**(-7), resample=False, xlim = [-0.03, 0.1])
#run_sim(fname='tau5e5', changing=True, tau=5*10**(-6.5), resample=False, xlim = [-0.03, 0.1])
#run_sim(fname='tau5e3', changing=True, tau=5*10**(-3), resample=False, xlim = [-0.03, 0.1])

#questions: is the probability of fixation per unit time or absolute?
#how do we have a negative time to fixation?

#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Code for simulating coalescents without recombination
"""

import numpy as np
import matplotlib.pyplot as plt
import networkx as nx

def add_sucs(G, node, newleaves):
    '''return list of leafnodes in order of their ancestors'''
    for newnode in G.successors(node):
        sucs = list(G.successors(newnode))
        if len(sucs) == 0: newleaves.append(newnode)
        else:
            newleaves = add_sucs(G, newnode, newleaves)
    return newleaves


def sort_graph(G):
    '''sort graph and annotate with initial x values to plot'''
    nodes = np.array(list(nx.topological_sort(G)))
    root = nodes[0]
    newleaves = add_sucs(G, root, [])
    xs = {}
    for i, leaf in enumerate(newleaves):
        xs[leaf] = i/(len(newleaves)-1)
    nodes = np.setdiff1d(nodes, np.array(newleaves))
    times= np.array([G.node[node]['time'] for node in nodes])
    args = np.argsort(times)
    nodes = nodes[args]
    times = times[args]
    nodelist = newleaves+list(nodes)

    times = [0 for i in range(len(nodelist)-len(times))]+list(times)
    #print(nodelist)
    return nodelist, xs, times
```

```
def get_xs(G, xs, nodelist):
    '''sort the nodes according to their ancestors so branches don't cross'''
    not_added = True
    while not_added:
        not_added = False
        for i, node in enumerate(nodelist):
            parents = list(G.predecessors(node))
            if len(parents) > 0:
                parent = parents[0]
                if not parent in xs.keys():
                    sucs = list(G.successors(parent))
                    for suc in sucs:
                        if not suc in xs.keys():
                            #print('dont have', suc)
                            xs = get_xs(G, xs, nodelist[i+1:])
                            not_added = True
                    #print(sucs)
                    xs[parent] = np.mean([xs[s] for s in sucs])

    return xs


def draw_tree(G, relabel = False, fname = 'figures/drawtree_test.png'):
    '''plot the coalescent'''
    topo = list(nx.topological_sort(G))
    xs = {topo[0]: 0}
    nodelist, xs, times = sort_graph(G)
    xs = get_xs(G, xs, nodelist)
    plt.figure(figsize = (4,3))

    for node in nodelist:
        y1 = G.node[node]['time']
        x1 = xs[node]
        parents = list(G.predecessors(node))
        if len(parents) > 0:
            parent = parents[0]
            y2 = G.node[parent]['time']
            x2 = xs[parent]
            plt.plot([x1, x1], [y1, y2], 'k-')
            plt.plot([x1, x2], [y2, y2], 'k-')
        else:
            plt.plot([x1,x1], [y1, y1+0.1], 'k-')

    plt.xticks([], [])
    plt.ylim(0, max(times)+0.1)
    plt.xlim(-0.05, 1.05)
    plt.ylabel('time / N generations')
    if relabel: plt.savefig(fname, dpi = 120, bbox_inches = 'tight')
    else: plt.savefig(fname, dpi = 120, bbox_inches = 'tight')
    plt.show()


def sim_tree(n = 19, Print = True, dynamics = 'none'):
    '''simulate a coalescent with sample size n and N=1'''
    G= nx.DiGraph()
    G.add_nodes_from(range(1, n+1), time=0)
```

```
    t = 0 #start at time zero
    nnodes = n
    currnodes = [i for i in range(1,n+1)]

    while n > 1: #continue until everything has coalesced
        if dynamics == 'decreasing': #D > 0
            N = 5*(1/(1+np.exp(-2*(t-0.5))))-0.15)
        elif dynamics == 'increasing': #D < 0
            N = 10/(1+np.exp(2*t))
        else: N = 1 #D = 0

        rate = n*(n-1)/(2*N) #parameter of exponential distribution
        dt = np.random.exponential(scale = 1/rate) #next coalescence
        if Print: print(dt)
        t += dt #update time
        merge = np.random.choice(currnodes, 2, replace=False) #merge

        nnodes += 1
        G.add_node(nnodes, time=t) #update trees
        G.add_edge(nnodes, merge[0])
        G.add_edge(nnodes, merge[1])

        [currnodes.remove(node) for node in merge]
        currnodes.append(nnodes)
        n -= 1 #update number of lineages

    return G, t

def test_TMRCAs(ntrial = 100000, figname='figures/tmrca_hist.png'):
    #simulate ntrial trees and report TMRCA results
    tmrcas = []
    for i in range(ntrial):
        G, t = sim_tree(Print = False)
        tmrcas.append(t)

    plt.figure(figsize = (5,3))
    plt.hist(tmrcas, bins=50, density=True)
    plt.xlabel('TMRCA')
    plt.ylabel('frequency')
    plt.savefig(figname, dpi=120)
    print('mean:', np.mean(tmrcas), 'var:', np.var(tmrcas))

def plotdists():
    '''plot distributions of increasing and decreasing population sizes'''
    t = np.arange(0,3.1, 0.1)
    N1 = 10*(1/(1+np.exp(-2*(t-0.5))))-1.5
    N2 = 20/(1+np.exp(2*t))
    plt.plot(t, N1, 'b-')
    plt.plot(t,N2, 'g-')
    plt.xlabel('time / generations')
    plt.ylabel('N')
    plt.savefig('figures/changing_N.png')
    plt.show()


def plotsamps(n=19):
```

```
    '''plot the probability of sampling n individuals from K genomes'''
    Ks = range(n, 2*n+1)
    ps = [2**(2*n-K) * np.math.factorial(K) * np.math.factorial(n) /
          (np.math.factorial(2*n) * np.math.factorial(K-n)) for K in Ks ]
    plt.figure(figsize = (5,3))
    plt.plot(Ks, ps, 'b-')
    plt.xticks( Ks, Ks )
    plt.xlabel('K')
    plt.ylabel('$p_{all}(K)$')
    plt.savefig('figures/testsamp_K.png', bbox_inches = 'tight', dpi=120)
    plt.show()
    for i in range(len(Ks)):
        print(Ks[i], ps[i])

plotsamps(19)


#plotdists()

#G, t = sim_tree()
#draw_tree(G)
#test_TMRCAs()

#G, t = sim_tree(dynamics = 'decreasing')
#draw_tree(G, fname = 'figures/drawtree_decreasing2.png')

#G, t = sim_tree(dynamics = 'increasing')
#draw_tree(G, fname = 'figures/drawtree_increasing2.png')

#G, t = sim_tree(dynamics = 'constant')
#draw_tree(G, fname = 'figures/drawtree_constant2.png')
```