Solutions for Problem Set 1

Sample code is provided at the end of the document.[1]

Problem 1: Molecular evolution and genetic diversity in the influenza virus

Part (a)

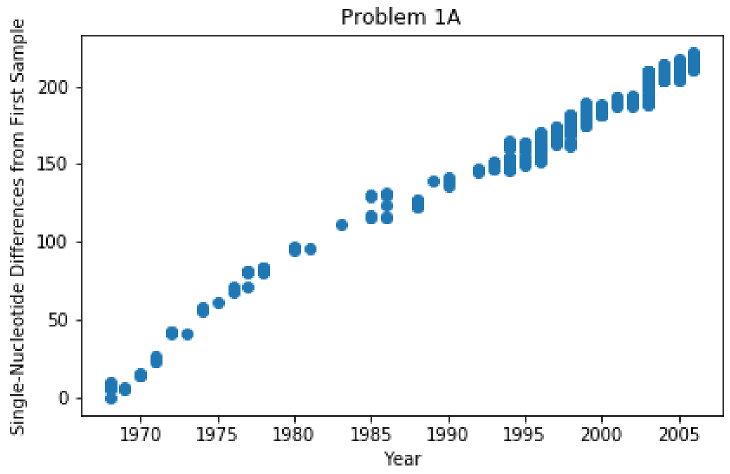


Figure 1: Number of single nucleotide differences between first HA gene sample (A/Aichi/2/1968) and others as a function of sampling year.

Approximately ∼200 (more like 210-220) differences accumulated over ~ 40 years. This corresponds to roughly 12-13% of the HA gene.

[1]Written by: Anita Kulkarni, Zhiru Liu and Benjamin Good (last updated on April 25, 2023). The codes were from a few years ago, when the problem numbers were a little bit different.

Part (b)

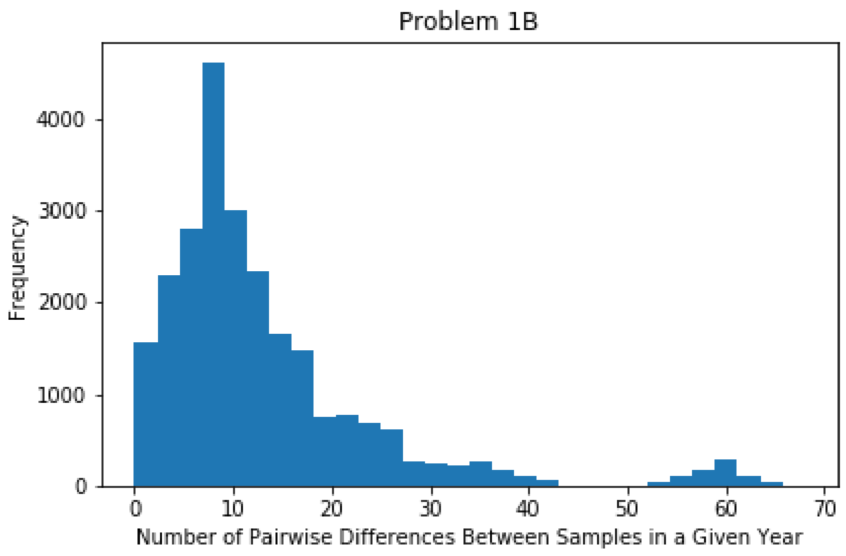


Figure 2: Distribution of number of genetic differences between all pairs of strains from the same year, aggregated across all years.

Most co-circulating strains vary at about 10 sites on average.

Bimodal distribution?
Some of you might be wondering – why is there a separate mode around 60 genetic differences in the above distribution? We can look into this by explicitly plotting the pairwise differences in each year separately, shown below. We see that the entire "second mode" is coming from year 2003, which probably had two lineages of Influenza A circulating (~ 60 differences between lineages, and ~ 10 differences within each lineage.)

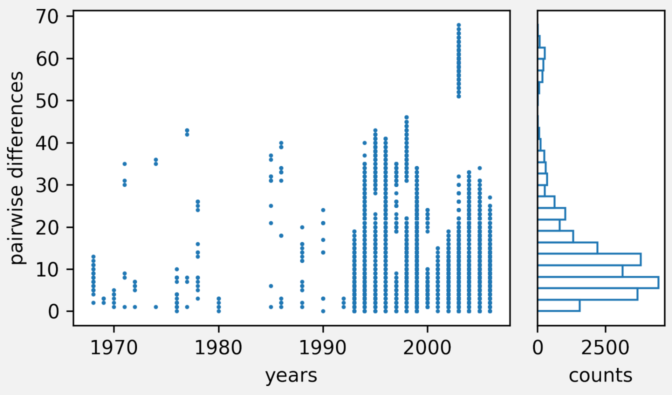


Figure 3: Genetic differences between pairs per year.

Comparing this to the slope from part (a), we see that this corresponds to a turnover time of ≈2-3 years.

Interpretation of the turnover time
From (a), we saw that the influenza population continues to diversify at a rate of ~ 6 differences per year. However, the amount of genetic diversity at any given year remains somewhat fixed at ~ 10 differences. Some of the older strains must be going extinct at a steady rate to maintain this level of diversity.
Here's an analogy: say university S has 4,000 incoming student each year, and enrolls roughly 16,000 students year to year. This means that 4,000 students graduate each year, and that the student population will completely turn over in 4 years. We are calculating the turnover time for the influenza population with much of the same reasoning here.

Problem 2: The Luria-Delbrück experiment

Part (a)

At $t = 0$, there are N_0 individuals without any antibiotic resistance, and at each subsequent time step t we assume that all $N(t)$ individuals divide, thus doubling the population ($2N(t)$ daughter cells). During this process, all daughter cells have an equal probability μ of acquiring a mutation; thus, the mean number of new mutations produced in generation t is

$$\theta(t) = \mu N_0 2^t$$

(Note that this formula is only valid for $1 \le t \le T$.)

Part (b)

If a mutation arises at generation t, then it will have $T - t$ generations to leave descendants by growing exponentially. Thus,

$$n(t) = 2^{T-t}$$

The total number of descendants left by all the mutants that arise at time t is $m(t)n(t)$; thus,

$$M_T = \sum_{t=1}^{T} m(t)n(t) = \sum_{t=1}^{T} 2^{T-t} m(t)$$

Part (c)

Mean of M_T:

$$\langle M_T \rangle = E\left[\sum_{t=1}^{T} 2^{T-t} m(t)\right] = \sum_{t=1}^{T} E[2^{T-t} m(t)] = \sum_{t=1}^{T} 2^{T-t} E[m(t)] = \sum_{t=1}^{T} 2^{T-t} \theta(t)$$

$$= \sum_{t=1}^{T} 2^{T-t} \mu N_0 2^t = \mu N_0 2^T \sum_{t=1}^{T} 1 = T \cdot N_0 \mu \cdot 2^T$$

Use a similar approach for the variance (i.e. properties of the variance of linear combinations of independent random variables, as each time point is independent), noting that since $m(t)$ is Poisson distributed its variance will be $\theta(t)$:

$$\text{Var}(M_T) = \text{Var}\left(\sum_{t=1}^{T} 2^{T-t} m(t)\right) = \sum_{t=1}^{T} \text{Var}(2^{T-t} m(t)) = \sum_{t=1}^{T} 2^{2T-2t} \text{Var}(m(t)) = \sum_{t=1}^{T} 2^{2T-2t} \theta(t)$$

$$= \sum_{t=1}^{T} 2^{2T-2t} \mu N_0 2^t = \mu N_0 2^{2T} \sum_{t=1}^{T} 2^{-t} = \mu N_0 4^T (1 - 2^{-T}) = \mu N_0 2^T (2^T - 1) \approx N_0 \mu \cdot (2^T)^2$$

The Fano factor is thus

$$F = \frac{2^T - 1}{T} \approx \frac{2^T}{T}$$

F which is larger than the Poisson limit by a factor of $2^T/T \gg 1$.

This suggests that we should be able to distingiush between the induction and mutation hypotheses by calculating this Fano factor from the observed data. If the variance in the number of colonies seen is much larger than the mean (i.e. there are a few plates with very many colonies and the rest have no or very few colonies), then the mutation hypothesis would be supported. If the mean and variance are similar (i.e. almost all of the plates have just a few colonies), then the induction hypothesis would be supported.

Part (d)

First, calculate $\langle \overline{M}_T \rangle$:

$$\langle \overline{M}_T \rangle = \frac{1}{n}\sum_{i=1}^{n} \langle M_{T,i} \rangle = \frac{n}{n}\mu N_0 T 2^T = T \cdot N_0 \mu \cdot 2^T$$

Next, calculate Var$(\overline{M}_T)$:

$$\text{Var}(\overline{M}_T) = \text{Var}\left(\frac{1}{n}\sum_{i=1}^{n} M_{T,i}\right) = \frac{1}{n^2}\text{Var}\left(\sum_{i=1}^{n} M_{T,i}\right) = \frac{1}{n}\text{Var}(M_T) = \frac{1}{n}\cdot N_0\mu \cdot 2^T(2^T-1)$$

After some algebra, we get that the coefficient of variation is

$$c_V = \frac{1}{T}\sqrt{\frac{1-2^{-T}}{n\mu N_0}} \approx \frac{1}{T\sqrt{nN_0\mu}}$$

Setting this equal to ϵ and solving for n, we get that we need

$$n \approx \frac{1-2^{-T}}{N_0\mu\epsilon^2T^2} \approx \frac{1}{N_0\mu\epsilon^2T^2}$$

independent experiments to get $O(\epsilon)$ relative error. When $N_0\mu \ll 1$, the CV and n get very large; lots of experiments are needed to precisely estimate $\langle M_T \rangle$ and Var(M_T) as the mutation rate gets small.

Real numbers
It's nice to get some intuition of the numbers here. For most bacteria, the mutation rate is about $\mu = 10^{-9}$ per site per generation. For final population size of 10^8 cells and growth period of 7 generations, and a desired error rate of 10%, we get

$$n \approx \frac{1}{10^6 \cdot 10^{-9} \cdot 0.01 \cdot 50} \approx 2000$$

That's a lot!

Part (e)

Calculate $\theta_<(t|n)$, or the total number of mutations among all n populations expected to arise before generation t:

$$n\sum_{j=1}^{t-1}\theta(j) = n\sum_{j=1}^{t-1}N_0\mu 2^j = nN_0\mu\sum_{j=1}^{t-1}2^j = nN_0\mu(2^t-2)$$

Set this equal to 1 to find t^*:

$$nN_0\mu(2^{t^*}-2)=1 \implies 2^{t^*}=\frac{1}{nN_0\mu}+2$$
$$\implies t^*=\log_2\left(\frac{1}{nN_0\mu}+2\right)\approx\log_2\left(\frac{1}{nN_0\mu}\right) \tag{1}$$

By definition, this critical time t^* only makes sense when $t^*\leq T$, which requires that

$$nN_0\mu 2^T \geq 1 \tag{2}$$

(in other words, we should typically expect to have at least one mutation in one of the replicates by the end of the experiment)

Using these expressions, we find that the typical mean is given by

$$\langle\overline{M}_T\rangle_{\text{typ}} = \langle M_T\rangle_{\text{typ}} = \sum_{t=1}^{T}2^{T-t}\hat{\theta}(t|n) = \sum_{t=t^*}^{T}2^{T-t}\theta(t) = N_0\mu 2^T(T-t^*+1)$$

and the typical variance is

$$\text{Var}(\overline{M}_T)_{\text{typ}} = \frac{1}{n}\text{Var}(M_T)_{\text{typ}} = \frac{1}{n}\sum_{t=t^*}^{T}2^{2T-2t}N_0\mu 2^t = \frac{1}{n}N_0\mu 2^T\left[2^{T-t^*+1}-1\right]$$

The coefficient of variation is therefore given by

$$c_V = \frac{\sqrt{\text{Var}(\overline{M}_T)_{\text{typ}}}}{\langle\overline{M}_T\rangle_{\text{typ}}} = \sqrt{\frac{2^{T-t^*+1}-1}{(T-t^*+1)^2 nN_0\mu 2^T}} \tag{3}$$
$$= \sqrt{\frac{2^T nN_0\mu - 1}{(T-t^*+1)^2 nN_0\mu 2^T}} \tag{4}$$
$$\approx \frac{1}{T-t^*} \tag{5}$$

where we've dropped a lot of 1's assuming T to be large. We see that the $1/N\mu$ dependence is gone now, as long as we choose the experimental condition such that $1\leq t^*\leq T$. Plugging in our previous expression for t^*, we see that

$$c_V = \frac{1}{T+\log_2(nN_0\mu)} \sim \frac{1}{\log(n)} \tag{6}$$

So the coefficient of variation does not decay as $n^{-1/2}$ as we would expect from the central limit theorem, but instead displays a much slower logarithmic decay.

Part (f)

We know that $M'_T \sim$ Poisson $\left(N_0 \frac{M_T}{N_T}\right)$ = Poisson $\left(\frac{M_T}{2^T}\right) \equiv$ Poisson(R) (we're defining a new random variable $R \equiv M_T/2^T$). Then:

$$\langle M'_T \rangle = E[\text{Poisson}(R)] = E[R] = \frac{1}{2^T} E[M_T] = \frac{1}{2^T} N_0 \mu T 2^T = N_0 \mu T$$

To find the variance of M'_T, use the formula $\text{Var}(X) = E[\text{Var}(X|Y=y)] + \text{Var}(E[X|Y=y])$:

$$\text{Var}(M'_T|R) = R(=r) \implies E[R] = N_0 \mu T$$

$$E[M'_T|R] = R(=r) \implies \text{Var}(R) = \frac{1}{4^T} \text{Var}(M_T) = N_0 \mu (1 - 2^{-T})$$

$$\implies \text{Var}(M'_T) = N_0 \mu (T + 1 - 2^{-T})$$

So

$$F = 1 + \frac{1 - 2^{-T}}{T} \approx 1 + \frac{1}{T}$$

which approaches the Poisson limit of $F \approx 1$ when $T \gg 1$. For example, for a dilution factor of $2^T = 100$, we have $T \approx 6.7$ and $F \approx 1.14$ – a relatively small deviation from the Poisson approximation we used in class.

Problem 3: The *E. coli* genome

Part (a)

The genome is 4,629,812 bp long. The relative fractions are all roughly the same:

- A: 24.64%
- T: 24.59%
- C: 25.42%
- G: 25.35%

GC content
In the literature, people refer to the total proportion of G-C bases as "GC content." In this *E. coli*, the GC content is about half; however, GC content can vary a lot among species (bacterial or not), and even within the chromosome (e.g. 35% to 60% across 100-Kb fragments in humans, cf Wikipedia).

Part (b)

20-mer occurrence distribution (this gives a general idea; note the log scale):

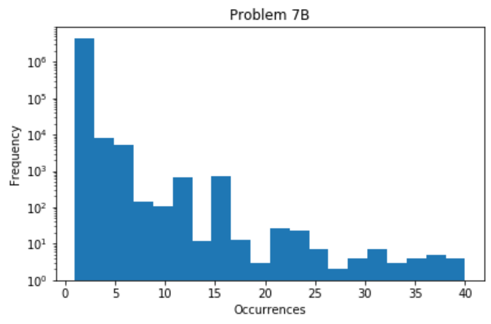


Figure 4: 20-mer occurrence distribution in *E. coli* genome.

99.2% of unique 20-mers appear only once (as a fraction of all unique 20-mers), and 97.4% of 20-mers in the E. coli genome appear exactly once. This shows that most (specifically 97.4%) sites in the E. coli genome can be uniquely identified by a 20 bp sequence.

PCR primers
Usually, primers used in PCR reactions are about 20bp long. The total number of possible 20-mer ($\sim 10^{12}$) is orders of magnitude larger than the genome size of most organisms (including humans, $\sim 10^9$ bp), so we expect primers of this size to have good specificity to locate a particular genome region.

Part (c)

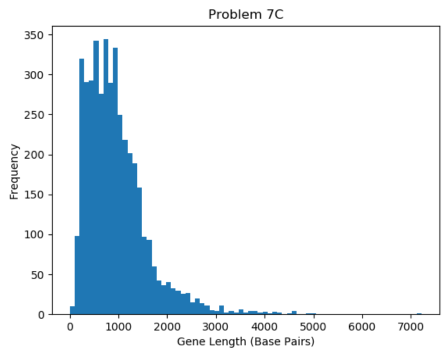


Figure 5: Distribution of gene lengths in *E. coli* genome.

4,217 genes account for 86.76% of the total genome length, and 52.15% of genes are transcribed in the reverse direction.

Coding region size
In contrast, the protein coding region in higher organisms accounts for a much smaller percentage of the genome. In humans, this number is only about $1-2\%$. In some sense, bacterial genomes are much more “streamlined”, because they have much smaller genome size and have to be more efficient.

Part (d)

The number of possible synonymous mutations (in the coding region) is 3,059,233, nonsense mutations is 404,289, and missense mutations is 8,587,451.

My number is different from above! Do not worry to much about the exact numbers here – answers may vary depending on the definition of these three classes. For example, when a stop codon mutated into a stop codon, did you count that as synonymous or nonsense? (Both options makes *sense*!) However, the rough proportion of these three classes should be consistent between reasonable implementations – about three times as many missense as synonymous, and much fewer nonsense mutations.

Problem 4: Single Locus Simulations

We use the serial dilution model introduced in class as the microscopic model for the single locus simulation. At the start of each cycle k, the number of mutant is $N_m(k)$ and the number of wild type is $N_w(k)$, and the mutant frequency is

$$f(k) = \frac{N_m(k)}{N_m(k) + N_w(k)} \tag{7}$$

After a day of exponential growth, the total number of cells become roughly N_f. $\gamma \equiv N_f/N_0$ is the dilution factor, which in turn determines the number of generations per day $\Delta t = \log_2(\gamma)$. We could calculate the fraction of mutant type at the end of the day to be

$$\tilde{f}(k) = \frac{f(k)e^{s\Delta t}}{f(k)e^{s\Delta t} + (1 - f(k))} \tag{8}$$

Because new mutations are sufficiently rare, we can make the additional assumption that each wild type cell acquires the mutation independently. Then, the starting population sizes of the mutant and the wild type follow Poisson distributions

$$\begin{aligned} N_m(k+1) &\sim \text{Poisson}\big(N_0\tilde{f}(k)\big) + \text{Poisson}\big(N_0\mu\Delta t(1-\tilde{f}(k))\big) \\ N_w(k+1) &\sim \text{Poisson}\big(N_0(1-\mu\Delta t)(1-\tilde{f}(k))\big) \end{aligned}$$

Note that $N_m + N_w$ is a random number, and need not equal to N_0 exactly. But for large N_0, the spread of this random number is small compared to the mean N_0. Finally, we could update the mutant frequency of the new day as

$$f(k+1) = \frac{N_m(k+1)}{N_m(k+1) + N_w(k+1)} \tag{9}$$

Part (a)

In Fig.6, we first explored the neutral dynamics. When plotted on the linear time scale, each trajectory varies greatly and the fixation/extinction event could take place at any location of the plot. However, on log scale, it becomes clear that the rough time for fixation/extinction is of order $T = N_0$. We will stick to the log time scale when we scan over different combination of parameters.

In Fig.7, we plotted nine different parameter combinations. The most notable feature of the plots is that, the larger N_0 and s, the smaller the fluctuation among different trajectories. For $N_0 = 10^6$, the two beneficial cases become essentially deterministic. The selection coefficient also determines how fast the sweep finishes. Roughly, the sweep time scales as $1/s$. Lastly, the fixation/extinction time for neutral simulations scales with N_0.

Part (b)

Now the mutation is turned on with rate $\mu = 10^{-5}$. Four deleterious simulations are shown in Fig.8 separately to avoid overlapping. None of the run reaches high frequency, as is

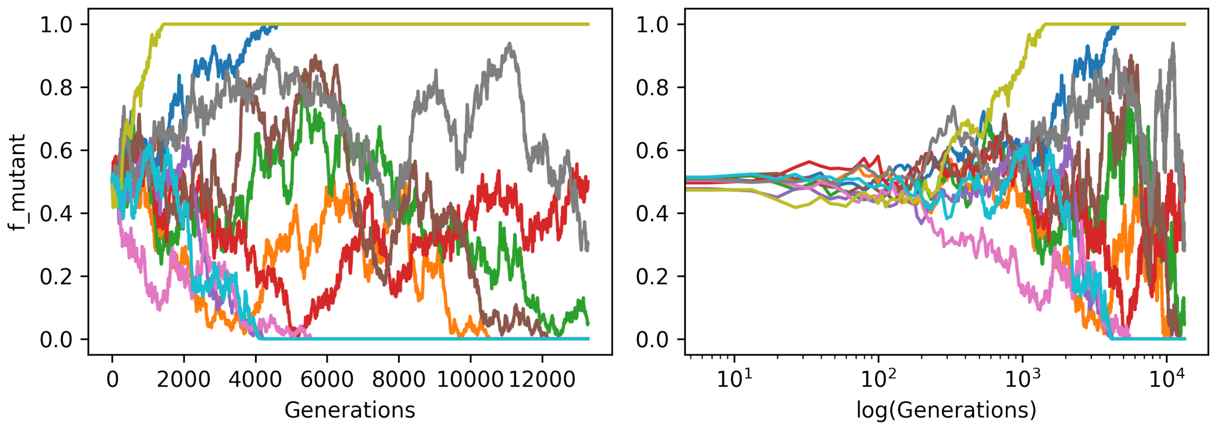


Figure 6: Single locus simulation with $N_0 = 10^3$, $\gamma = 100$, $\mu = s = 0$. 10 replicates are shown here.

expected for a deleterious mutation. However, because of the continuous input of new mutants, there is always some mutants in the population. In fact, the order of magnitude for the fraction of mutants is $\mu/|s| = 10^{-2}$. This dynamics is known as *deleterious mutation-selection-drift balance*.

On the other hand, for a strongly beneficial mutation $s = 10^{-2}$, the mutant always sweep through the population (Fig.9). The start of the sweep is stochastic, and is determined by the population size N_0 as well as the selection strength.

Connection to theoretical analysis
After seeing how to analyze the single-locus SDE in class, could you refer back to these simulation results and interpret them using theoretical intuitions? Some of the main features are already outlined above. The heuristic picture introduced in lecture 7&8 will be especially useful. In particular, think about

- What is the timescale for genetic drift in each of the simulations?
- At what frequency does selection become important in each of the simulations (i.e. the drift barrier)?
- How fast does beneficial mutations sweep to fixation? How does that scale with s?

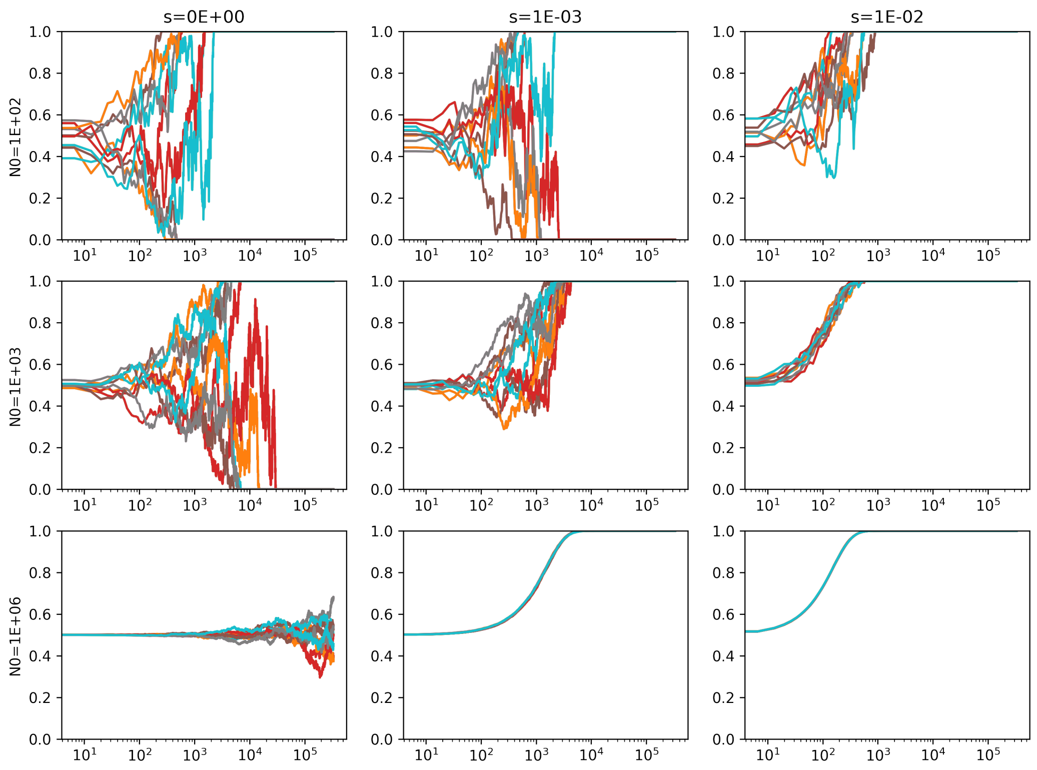


Figure 7: Scanning over 9 parameter combinations. Here, $N_0 = 10^2, 10^3, 10^6$, $s = 0, 10^{-3}, 10^{-2}$, $\mu = 0$, and a dilution factor $\gamma = 100$. 10 replicates are shown for each combination.

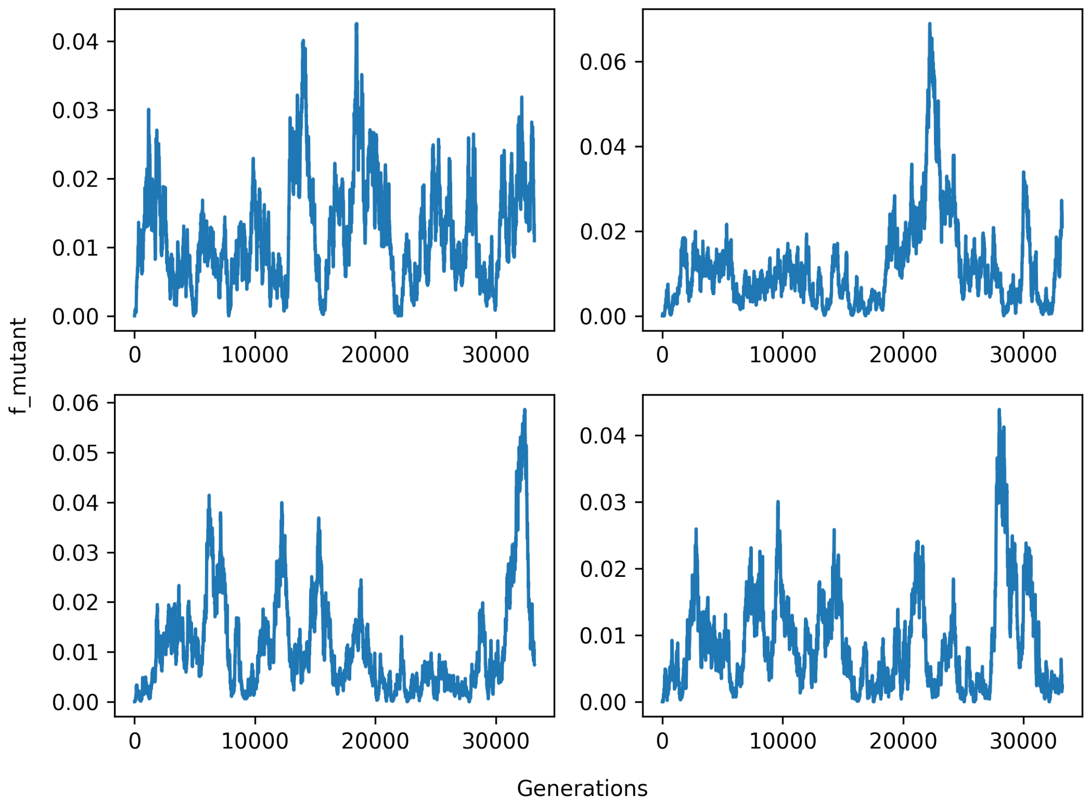


Figure 8: Trajectories of deleterious mutants. $N_0 = 10^4$, $\mu = 10^{-5}$, $s = -10^{-3}$ and the initial mutant population is zero.

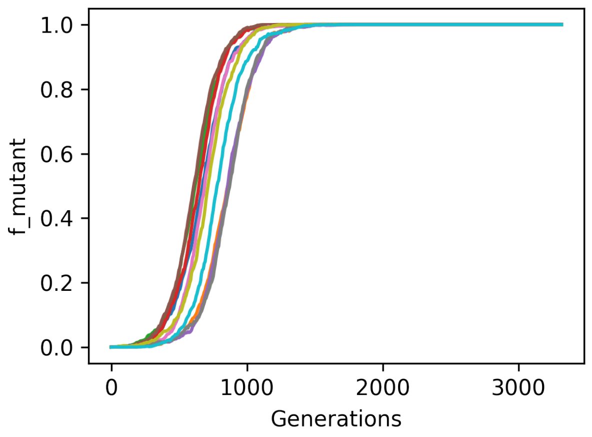
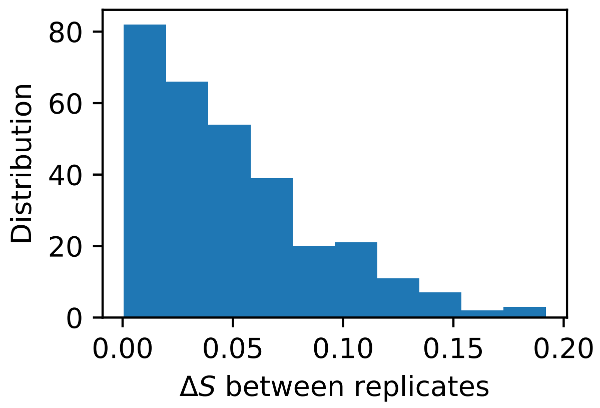


Figure 9: Trajectories of beneficial mutants. $N_0 = 10^4$, $\mu = 10^{-5}$, $s = 10^{-2}$ and the initial mutant population is zero.

Problem 5: Competitive fitness in a long term evolution experiment in E. coli

Part (a)

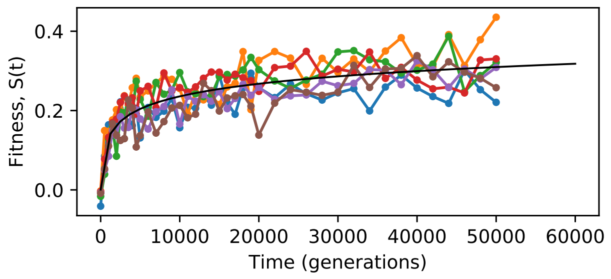
The observed distribution of differences between replicate fitness measurements is shown below:



This shows that the typical errors are on the order of $\sigma_S \sim 5\%$.

Part (b)

The estimated fitness trajectories for each population are shown in the colored lines below:

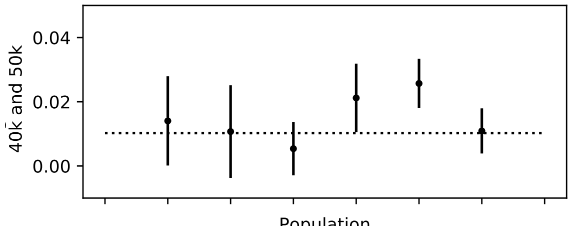


Part (c)

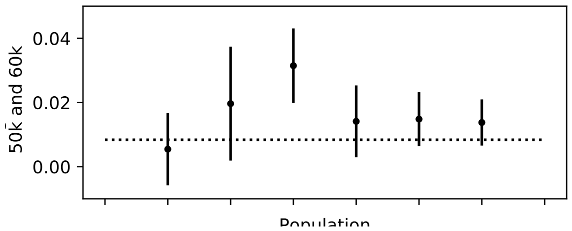
The predicted trajectory, $X(t) = X_c \log(1 + v_0 t/X_c)$ is shown in the solid black line in the plot in part (b) for $X_c \approx 4.6 \times 10^{-2}$ and $v_0 = 7.7 \times 10^{-4}$. It seems to be roughly consistent with the data. The predicted fitness gain between generation $40,000$ and $50,000$ is $\Delta X \approx 0.01$, which is well within the range of uncertainty on the individual fitness measurements above.

Part (d)

Using the more highly replicated fitness assays in the provided file, the fitness gains between generation 40,000 and 50,000 are shown for different replicate popluations below:



The error bars denote ± 2 standard errors, and the dashed line indicates the prediction from the theoretical model in Part (c). An analogous figure for generation 50,000 to 60,000 is shown below:



In both cases, the error bars exclude 0 for most of the replicate populations, suggesting that fitness is still improving in the Lenski experiment. (We note however, that the statistical support for the predicted fitness trajectory is much weaker.)

Problem 6: Pooled fitness assay

Part (a)

Given that each cycle is of length Δt and each strain grows as $N_k(t) = N_k(0)e^{s_k t}$,

$$f_k(\Delta t) = \frac{N_k(\Delta t)}{\sum_{i=1}^K N_i(\Delta t)} = \frac{N_k(0)e^{s_k\Delta t}}{\sum_{i=1}^K N_i(0)e^{s_i\Delta t}} = \frac{\frac{1}{\sum_{i=1}^K N_i(0)} e^{s_k\Delta t} N_k(0)}{\frac{1}{\sum_{j=1}^K N_j(0)} \sum_{i=1}^K N_i(0)e^{s_i\Delta t}} = \frac{f_k(0)e^{s_k\Delta t}}{\sum_{i=1}^K f_i(0)e^{s_i\Delta t}}$$

Part (b)

If we neglect noise, the frequencies calculated in part a will be conserved during the dilution step and the next growth phase will be deterministic; the calculation will be akin to that of part a once again. The calculation is also not dependent on completing a full growth phase Δt. It is thus easy to see that

$$f_k(2 \text{ cycles}) = \frac{f_k(0)e^{2s_k\Delta t}}{\sum_{i=1}^K f_i(0)e^{2s_i\Delta t}}$$

$$f_k(n \text{ cycles}) = \frac{f_k(0)e^{ns_k\Delta t}}{\sum_{i=1}^K f_i(0)e^{ns_i\Delta t}}$$

$$f_k(t) = \frac{f_k(0)e^{s_k t}}{\sum_{i=1}^K f_i(0)e^{s_i t}}$$

Part (c)

Shift all $s_i \to s_i' = s_i + c$:

$$f_k'(t) = \frac{f_k(0)e^{(s_k+c)t}}{\sum_{i=1}^K f_i(0)e^{(s_i+c)t}} = \frac{f_k(0)e^{s_k t}e^{ct}}{e^{ct}\sum_{i=1}^K f_i(0)e^{s_i t}} = f_k(t)$$

Only knowing $f_k(t)$ (tracking strain frequencies over time) is not enough to determine the absolute values of s_k.

Part (d)

Let strain 0 be the wildtype with $s_0 = 0$. Then

$$\frac{f_k(t)}{f_0(t)} = \frac{f_k(0)e^{s_k t}}{f_0(0)e^{s_0 t}} = \frac{f_k(0)}{f_0(0)}e^{s_k t} \implies \frac{N_k(t)}{N_0(t)} = \frac{N_k(0)}{N_0(0)}e^{s_k t} \implies \frac{N_k(t_2)}{N_0(t_2)}\frac{N_0(t_1)}{N_k(t_1)} = e^{s_k(t_2-t_1)}$$

$$\implies s_k = \frac{1}{t_2 - t_1}\log\left(\frac{N_k(t_2)N_0(t_1)}{N_0(t_2)N_k(t_1)}\right)$$

Part (e)

The initial frequency of the wildtype (assume $s_0 = 0$) is f_0 and the initial frequencies of the rest of the strains are $(1 - f_0)/K$ (since K is very large, it is safe to have K in the denominator instead of $K - 1$). Plug these into our formula for f_k:

$$f_k(t) = \frac{\frac{1-f_0}{K} e^{s_k t}}{f_0 + \frac{1-f_0}{K} \sum_{i=1}^{K} e^{s_i t}}$$

Now we want to find the "average" frequency trajectory for the trajectory of the focal strain k. Since K is very large, by the central limit theorem, $\frac{1}{K}\sum_{i=1}^{K} e^{s_i t}$ can safely be approximated by the mean $\langle e^{s_i t}\rangle = e^{\frac{1}{2}\sigma^2 t^2}$ (technically we would need to pull out $e^{s_k t}$ from the sum since presumably s_k is fixed/known, but this shouldn't matter much when K is very large):

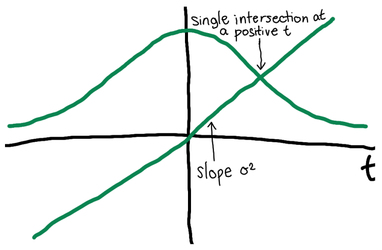
$$\langle f_k(t)\rangle \approx \frac{\frac{1-f_0}{K} e^{s_k t}}{f_0 + (1-f_0) e^{\frac{1}{2}\sigma^2 t^2}}$$

Check if this is monotonic by looking for maxima/minima:

$$\frac{\partial \langle f\rangle}{\partial t} = \frac{1-f_0}{K} \frac{1}{\left(f_0 e^{s_k t} + (1-f_0) e^{\frac{1}{2}\sigma^2 t^2 - s_k t}\right)^2} \left(-s_k f_0 e^{-s_k t} + (1-f_0)(\sigma^2 t - s_k) e^{\frac{1}{2}\sigma^2 t^2 - s_k t}\right) = 0$$

$$\implies -s_k f_0 + (1-f_0)(\sigma^2 t - s_k) e^{\frac{1}{2}\sigma^2 t^2} = 0 \implies \sigma^2 t = s_k \left(\frac{f_0}{1-f_0} e^{-\frac{1}{2}\sigma^2 t^2} + 1\right)$$

If $s_k > 0$ and $t > 0$, then both sides of the equation are positive and we get the following situation:



So the frequency trajectory has a maximum at the t^* that solves the following equation:

$$t^* = \frac{s_k}{\sigma^2} \left(\frac{f_0}{1-f_0} e^{-\frac{1}{2}\sigma^2 t^{*2}} + 1\right)$$

Problem 7: Experimental evolution in a chemostat (in theory)

Part (a)

At steady state, n and c become constant, and the time derivatives $\partial_t n$ and $\partial_t c$ become zero. We can set Eq. 8 to zero and find that

$$r(c^*) = \delta\,. \tag{10}$$

Notice that this equilibrium growth rate does not depend on c_{in}!

Part (b)

Similar as above, we use the steady state condition to set Eq. 9 to zero and find that

$$\delta c_{\text{in}} - \delta c^* - \frac{r(c^*)n^*}{V} = 0 \tag{11}$$

$$\Rightarrow n^* = V(c_{\text{in}} - c^*) \approx V c_{\text{in}}\,. \tag{12}$$

Again, n^* does not depend on the growth function $r(c)$. This is because no matter what the function form of $r(c)$ is, the nutrient concentration will adjust to c^* such that $r(c^*) = \delta$.

Part (c)

The "adiabatic limit" is a common approximation in physics; here, it assumes that c changes much more rapidly compared to n, and at any given moment, nutrient concentration will approach a (temporary) steady state value given by $\partial_t c = 0$. This steady state value of c still changes over time, because of the slower dynamics of $n(t)$. The nice thing about this approximation is that we can reduce the number of dynamical variables in this system to only one, i.e. $n(t)$.

$$\partial_t c = 0 \approx \delta c_{\text{in}} - r(c)n/V \Rightarrow r(c) = \delta c_{\text{in}} V/n \tag{13}$$

Plugging the above into the equation for $n(t)$, we get

$$\partial_t n = \delta(c_{\text{in}} V - n) \tag{14}$$

The solution to this equation is an exponential relaxation toward the equilibrium value n^* we found in Part (b),

$$n(t) = n^* + (n(0) - n^*)e^{-\delta t}\,. \tag{15}$$

The rate of this exponential relaxation is simply set by δ.

Part (d)

With two strains, we need to keep track of three dynamical variables, n_{wt}, n_{mut} and c. We modify the model to be

$$\partial_t n_{\text{wt}} = r(c) n_{\text{wt}} - \delta n_{\text{wt}} \tag{16}$$

$$\partial_t n_{\text{mut}} = (1+s) r(c) n_{\text{mut}} - \delta n_{\text{mut}} \tag{17}$$

$$\partial_t c = \delta c_{\text{in}} - \delta c - \frac{1}{V}\left[r(c) n_{\text{wt}} + (1+s) r(c) n_{\text{mut}}\right] \tag{18}$$

Similar to Part (c), we set the equation for c to zero,

$$r(c)\left[n_{\text{wt}} + (1+s) n_{\text{mut}}\right] = \delta c_{\text{in}} V \tag{19}$$

To find how the total population size changes, we add together Eq.(16) and (17):

$$\partial_t N = r(c) n_{\text{wt}} + (1+s) r(c) n_{\text{mut}} - \delta N \tag{20}$$

$$\Rightarrow \partial_t N = \delta c_{\text{in}} V - \delta N \tag{21}$$

where we have used Eq.(19) in the last step. (Notice that again, the explicit form of $r(c)$ did not matter.)

Part (e)

Since the equation for total population size is the same as the single-strain case, the steady state population size should be n^*. If at time 0, $N(0) = n^*$, then the population size will remain at this steady state level. The only thing that will change is the composition of wild type v.s. mutants. We can derive an effective model for the mutant frequency as the following:

$$\partial_t f(t) = \frac{\partial_t n_{\text{mut}}}{N} = (1+s) r(c) \frac{n_{\text{mut}}}{N} - \delta \frac{n_{\text{mut}}}{N} \tag{22}$$

$$= (1+s) r(c) f(t) - \delta f(t) \tag{23}$$

Using the adiabatic condition again (19),

$$\partial_t f = \frac{(1+s)\delta c_{\text{in}} V}{n_{\text{wt}} + (1+s) n_{\text{mut}}} f - \delta f \tag{24}$$

$$= \frac{(1+s)\delta N}{N + s n_{\text{mut}}} f - \delta f \tag{25}$$

$$= \delta \frac{(1+s)}{1+sf} f - \delta f \tag{26}$$

Expanding the fraction to lowest order in s, we finally get

$$\partial_t f = \delta(1 + s - sf) f - \delta f \tag{27}$$

$$= \delta s f (1-f) \tag{28}$$

which has the same form as the serial dilution model in class.

Part (f)

Assuming that the vessel is well-mixed, then the probability that a particular cell gets diluted out is

$$\mathrm{Pr}[\mathrm{diluted}] = \delta \Delta t \tag{29}$$

The total number of wild-type or mutant cells getting diluted each timestep follow a binomial distribution (n_{wt} or n_{mut} coin flips with above probability). The variance is then

$$\mathrm{Var}(n_{\mathrm{wt}}) = n_{\mathrm{wt}} \delta \Delta t (1 - \delta \Delta t) \approx n_{\mathrm{wt}} \delta \Delta t \tag{30}$$

(same form for n_{mut}.) This is similar to the Poisson noise in the serial dilution model in class.

Sample code for Problem Set 1

```python
# Code for Problem 1 of Problem Set 1

# -*- coding: utf-8 -*-
"""
Created on Tue Jan 21 01:05:31 2020

@author: Anita Kulkarni
"""

import matplotlib.pyplot as plt

f = open("../data_files/problem_set_data/influenza_HA_dna_sequences.fasta", "r")
data = f.readlines()
sequences = []  # sequences is a list of tuples (year integer, DNA sequence string)

for i in range(0, len(data), 2):  # lines alternate between label (year, location, et
    year = int(data[i][-5:-1])
    seq = data[i+1][:-1]
    sequences.append((year, seq))

ref_seq = sequences[0][1]  # first sequence (Aichi, 1968) is reference sequence
print(len(ref_seq))

sequences.sort(key=lambda tup: tup[0])  # sort list of tuples by year

def compare_seq(s0, s1):  # number of differences between two sequences
    diff = max((len(s0),len(s1))) - min((len(s0),len(s1)))  # difference in length
    for i in range(min((len(s0), len(s1)))):
        if s0[i] != s1[i]:
            diff = diff + 1
    return diff

num_differences = []
years = []
for i in range(len(sequences)):
    num_differences.append(compare_seq(ref_seq, sequences[i][1]))
    years.append(sequences[i][0])
plt.plot(years, num_differences, 'o')
plt.xlabel("Year")
plt.ylabel("Single-Nucleotide Differences from First Sample")
plt.title("Problem 1A")
plt.tight_layout()
plt.savefig("AP237_PS1_Problem1A.png")
plt.show()

unique_years = list(set(years))
unique_years.sort()
# number of sequences for each year
year_counts = [years.count(unique_years[y]) for y in range(len(unique_years))]

pointer = 0
pairwise_diffs = []
for i in range(len(unique_years)):
    year_sequences = []
    for j in range(year_counts[i]):
        year_sequences.append(sequences[pointer+j][1])
    pointer = pointer + year_counts[i]
    for s in range(year_counts[i]):
        if year_counts[i]-s >= 1:
            for S in range(s+1, year_counts[i]):
                pairwise_diffs.append(compare_seq(year_sequences[s], year_sequences[S
plt.hist(pairwise_diffs, bins=30)
plt.xlabel("Number of Pairwise Differences Between Samples in a Given Year")
plt.ylabel("Frequency")
plt.title("Problem 1B")
plt.tight_layout()
plt.savefig("AP237_PS1_Problem1B.png")
plt.show()
```

```python
# Code for Problem 7 of Problem Set 1

# -*- coding: utf-8 -*-
"""
Created on Tue Jan 14 23:17:40 2020

@author: Anita Kulkarni
"""

import numpy as np
import matplotlib.pyplot as plt

f1 = open("../data_files/ecoli_reference_genome.fasta", "r")
f1.readline()
genome = f1.readline()
f1.close()
f2 = open("../data_files/ecoli_genes.txt", "r")
genes = f2.readlines()
del(genes[0])
for i in range(len(genes)):
    genes[i] = genes[i].split(", ")
    del(genes[i][0])
    genes[i][0] = int(genes[i][0])-1
    genes[i][1] = int(genes[i][1])
    if genes[i][2][0] == 'f':
        genes[i][2] = 1
    else:
        genes[i][2] = 0
f2.close()

len_genome = len(genome)
print(len_genome)
letter_counts = [0,0,0,0] #A, T, C, G
for i in range(len_genome):
    if genome[i] == 'A':
        letter_counts[0] = letter_counts[0] + 1
    elif genome[i] == 'T':
        letter_counts[1] = letter_counts[1] + 1
    elif genome[i] == 'C':
        letter_counts[2] = letter_counts[2] + 1
    elif genome[i] == 'G':
        letter_counts[3] = letter_counts[3] + 1
letter_counts = np.array(letter_counts)
letter_frequencies = letter_counts/np.sum(letter_counts)
print(letter_frequencies)
```

```python

twenty_mers = {}
for i in range(0, len_genome-20):
    seq = genome[i:i+20]
    if seq in twenty_mers:
        twenty_mers[seq] = twenty_mers[seq] + 1
    else:
        twenty_mers[seq] = 1
twenty_mer_frequencies = list(twenty_mers.values())
occ_list = list(set(twenty_mer_frequencies))
occ_list.sort()
twenty_mer_freq_dist = []
for val in occ_list:
    twenty_mer_freq_dist.append(twenty_mer_frequencies.count(val))
print(occ_list)
print(twenty_mer_freq_dist)
print(twenty_mer_freq_dist[0]/sum(twenty_mer_freq_dist))

num_genes = len(genes)
print(num_genes)
gene_lengths = []
num_reverse = 0
for i in range(len(genes)):
    gene_lengths.append(genes[i][1]-genes[i][0])
    if genes[i][2] == 0:
        num_reverse = num_reverse + 1
print(sum(gene_lengths))
print(sum(gene_lengths)/len(genome))
print(num_reverse/num_genes)
gene_lengths_unique = list(set(gene_lengths))
gene_lengths_unique.sort()
gene_lengths_freq_dist = []
for val in gene_lengths_unique:
    gene_lengths_freq_dist.append(gene_lengths.count(val))
plt.hist(gene_lengths, bins=np.arange(0, 99*int(gene_lengths_unique[len(gene_lengths_
plt.xlabel("Gene Length (Base Pairs)")
plt.ylabel("Frequency")
plt.title("Problem 7C")
plt.savefig("AP237_PS1_Problem7C.png")
plt.show()

codons = {'TTT':'F', 'TCT':'S', 'TAT':'Y', 'TGT':'C',
          'TTC':'F', 'TCC':'S', 'TAC':'Y', 'TGC':'C',
          'TTA':'L', 'TCA':'S', 'TAA':'STOP', 'TGA':'STOP',
          'TTG':'L', 'TCG':'S', 'TAG':'STOP', 'TGG':'W',
          'CTT':'L', 'CCT':'P', 'CAT':'H', 'CGT':'R',
          'CTC':'L', 'CCC':'P', 'CAC':'H', 'CGC':'R',
          'CTA':'L', 'CCA':'P', 'CAA':'Q', 'CGA':'R',
          'CTG':'L', 'CCG':'P', 'CAG':'Q', 'CGG':'R',
          'ATT':'I', 'ACT':'T', 'AAT':'N', 'AGT':'S',
          'ATC':'I', 'ACC':'T', 'AAC':'N', 'AGC':'S',
          'ATA':'I', 'ACA':'T', 'AAA':'K', 'AGA':'R',
          'ATG':'M', 'ACG':'T', 'AAG':'K', 'AGG':'R',
          'GTT':'V', 'GCT':'A', 'GAT':'D', 'GGT':'G',
          'GTC':'V', 'GCC':'A', 'GAC':'D', 'GGC':'G',
          'GTA':'V', 'GCA':'A', 'GAA':'E', 'GGA':'G',
          'GTG':'V', 'GCG':'A', 'GAG':'E', 'GGG':'G'}

syn = -sum(gene_lengths) # don't double-count replacements of same nucleotide
non = 0
mis = 0
nuc = ['A','T','C','G']
for i in range(len(genes)):

    forward_gene = genome[genes[i][0]:genes[i][1]]
    reversed_gene = genome[genes[i][1]:genes[i][0]:-1]

    #print len(forward_gene)
    #print len(reversed_gene)
    if genes[i][2] == 1: # non-reversed gene
        gene = forward_gene
    else: # reversed gene
        gene = reversed_gene
    for j in range(0, len(gene), 3):
        codon = gene[j:j+3]
        aa = codons[codon]
        for n in nuc:
            new_codon = ''.join((n, gene[j+1], gene[j+2]))
            new_aa = codons[new_codon]
            if new_aa == aa:
                syn = syn + 1
            elif new_aa == 'STOP':
                non = non + 1
            else:
                mis = mis + 1
            new_codon = ''.join((gene[j], n, gene[j+2]))
            new_aa = codons[new_codon]
            if new_aa == aa:
                syn = syn + 1
            elif new_aa == 'STOP':
                non = non + 1
            else:
                mis = mis + 1
            new_codon = ''.join((gene[j], gene[j+1], n))
            new_aa = codons[new_codon]
            if new_aa == aa:
                syn = syn + 1
            elif new_aa == 'STOP':
                non = non + 1
            else:
                mis = mis + 1
print(syn)
print(non)
print(mis)
```

```
### Code for Problem 4 on Problem Set 1

import numpy
import pylab
from math import log
import sys


# Part A

# Load data from file
file = open("../data_files/LTEE_ancestor_fitness_assays.txt","r")
file.readline() # ignore header
records = []
for line in file:
    items = line.split(",")
    population = items[0].strip()
    t = float(items[1])

    NE0 = float(items[2]) # evolved strain counts at time 0
    NA0 = float(items[3]) # ancestor strain counts at time 0
    NEF = float(items[4]) # evolved strain counts at time 1
    NAF = float(items[5]) # ancestor strain counts at time 1

    records.append((population,t,NE0,NA0,NEF,NAF))


fitness_data_map = {}
dt = numpy.log2(100.0) # these measurements were carried out with 100-fold dilution
# Collate by population and timepoint
for population,t,NE0,NA0,NEF,NAF in records:

    if population not in fitness_data_map:
        fitness_data_map[population] = {}

    if t not in fitness_data_map[population]:

        fitness_data_map[population][t] = []

    # Calculate fitness:

    s = 1.0/dt * log(NEF/NAF/(NE0/NA0))

    fitness_data_map[population][t].append(s)

```

```python
# Calculate differences between replicate measurements
deltas = []
for population in sorted(fitness_data_map):
    for t in sorted(fitness_data_map[population]):
        ss = fitness_data_map[population][t]

        # Look at all distinct pairs of replicates
        for i in xrange(0,len(ss)):
            for j in xrange(i+1,len(ss)):
                delta = numpy.fabs(ss[i]-ss[j])
                deltas.append(delta)

pylab.figure(figsize=(3,2))
pylab.xlabel('$\Delta S$ between replicates')
pylab.ylabel('Distribution')
pylab.hist(deltas)
pylab.savefig('problem_4_a.pdf',bbox_inches='tight')

# Part B and C
pylab.figure(figsize=(5,2))
pylab.xlabel('Time (generations)')
pylab.ylabel('Fitness, S(t)')
for population in sorted(fitness_data_map):
    ts = []
    savgs = []
    for t in sorted(fitness_data_map[population]):
        ss = numpy.array(fitness_data_map[population][t])
        savg = ss.mean()
        ts.append(t)
        savgs.append(savg)


    pylab.plot(ts,savgs,'.-')

theory_ts = numpy.linspace(1,60000)
Xc = 4.6e-02
v0 = 7.7e-04
theory_ss = Xc*numpy.log(1+v0*theory_ts/Xc)
pylab.plot(theory_ts,theory_ss,'k-',linewidth=1)
pylab.savefig('problem_4_bc.pdf',bbox_inches='tight')

sys.stdout.write("Predicted gain from 40k to 50k is: %g\n" % ( Xc*numpy.log(1+v0*5e04

# Part D
# Load other data file

# Load data from file
file = open("../data_files/LTEE_40k_fitness_assays.txt","r")
file.readline() # ignore header
records = []
for line in file:
    items = line.split(",")
    population = items[0].strip()
    t = float(items[1])

    NE0 = float(items[2]) # evolved strain counts at time 0
    NA0 = float(items[3]) # ancestor strain counts at time 0
    NEF = float(items[4]) # evolved strain counts at time 1
    NAF = float(items[5]) # ancestor strain counts at time 1

    records.append((population,t,NE0,NA0,NEF,NAF))


fitness_data_map = {}
dt = 3*numpy.log2(100.0) # these measurements were carried out over 3 days
# Collate by population and timepoint
for population,t,NE0,NA0,NEF,NAF in records:

    if population not in fitness_data_map:
        fitness_data_map[population] = {}

    if t not in fitness_data_map[population]:

        fitness_data_map[population][t] = []

    # Calculate fitness:

    s = 1.0/dt * log(NEF/NAF/(NE0/NA0))

    fitness_data_map[population][t].append(s)

for min_t,max_t in [(4e04,5e04),(5e04,6e04)]:
    pylab.figure(figsize=(5,2))
    pylab.ylabel('Fitness gain between \n %dk and %dk' % (min_t/1000,max_t/1000))
    pylab.xlabel('Population')
    pylab.ylim([-0.01,0.05])
    current_idx = 0
    for population in sorted(fitness_data_map):
        current_idx+=1

        ss = numpy.array(fitness_data_map[population][min_t])
        s0 = ss.mean()
        ds0 = ss.std()/(len(ss)*1.0)**0.5

        ss = numpy.array(fitness_data_map[population][max_t])
        sf = ss.mean()
        dsf = ss.std()/(len(ss)*1.0)**0.5

        s = sf-s0
        ds = (ds0**2+dsf**2)**0.5

        pylab.plot([current_idx],[s],'k.')
        pylab.plot([current_idx,current_idx],[s-2*ds,s+2*ds],'k-')


    # theory line
    predicted_ds = Xc*numpy.log(1+v0*max_t/Xc)-Xc*numpy.log(1+v0*min_t/Xc)

    pylab.plot([0,current_idx+1],[predicted_ds,predicted_ds],'k:')
    pylab.gca().set_xticklabels([])
    pylab.savefig('problem_4_d_%dk.pdf' % (max_t/1000))
```