AP237/Bio251 Problem Set 3 Solutions

Written/compiled by: Benjamin Good and Anita Kulkarni

March 7, 2021

Problem 1: Heuristics for recessive mutations

Part A

The short-time approximation of our SDE is

$$f(\Delta t) = f(0) + sf^2(0)\Delta t + \sqrt{\frac{f(0)\Delta t}{2N}}Z$$

This approximation is valid up to logarithmic (order-of-magnitude) precision on a timescale $\sim \Delta t_{\text{reset}}$; to find the frequency boundary between drift-dominated and selection-dominated regimes, check whether deterministic or stochastic forces are dominant on this timescale (look for self-consistency), similar to what was done in class for the haploid case.

- If deterministic forces are dominant,

  $$f(0) \sim |f(\Delta t_{\text{reset}}) - f(0)| \sim |s|f^2(0)\Delta t_{\text{reset}} \implies \Delta t_{\text{reset}} \sim \frac{1}{f|s|}$$

  On this timescale, the contribution from drift is

  $$\sqrt{\frac{f}{2N}\frac{1}{f|s|}} = \sqrt{\frac{1}{2N|s|}}$$

  Dropping constant factors, we get that $|\Delta f_{\text{drift}}| \ll |\Delta f_{\text{sel}}| \sim f$ when

  $$f \gg \frac{1}{\sqrt{N|s|}}$$

- Similarly, perform a self-consistency check under the assumption that stochastic forces are dominant. Under this assumption,

  $$f \sim |\Delta f_{\text{drift}}| \sim \sqrt{\frac{f\Delta t_{\text{reset}}}{2N}} \implies \Delta t_{\text{reset}} \sim 2Nf$$

  On this timescale, the contribution from selection is $2Nsf^3$, and for this to be $\ll f$, we need $f \ll \frac{1}{\sqrt{2N|s|}}$. Thus, $|\Delta f_{\text{sel}}| \ll |\Delta f_{\text{drift}}| \sim f$ when

  $$f \ll \frac{1}{\sqrt{N|s|}}$$

  and we have self-consistency.

Drift dominates below $f \sim 1/\sqrt{N|s|}$, and selection dominates above $f \sim 1/\sqrt{N|s|}$. In order for selection to be effective in at least some part of frequency space, we need the frequency boundary to be $\ll 1$; rearranging, we find that $N|s| \gg 1$.

Part B

The following heuristic result derived in class is not haploid-specific (i.e. the derivation did not utilize any particular properties of the haploid SDE): a mutant with initial size f_0 drifts to final (boundary) size f with probability f_0/f on a timescale of $\sim Nf$ generations. Plugging in our results from part a, assuming $s \gg 1$, we get that a mutant with initial size $\sim \frac{1}{N}$ drifts to boundary size $\sim \frac{1}{\sqrt{Ns}}$ with probability $\sim \sqrt{\frac{s}{N}}$ on a timescale of $\sim \sqrt{\frac{N}{s}}$ generations.

Since the mutation is strongly beneficial, once its frequency reaches $f^* \sim 1/\sqrt{Ns}$, it is guaranteed to fix deterministically. How long will this part take? To get a rough estimate, solve the deterministic equation (in the low-frequency limit since this is more tractable) with $f(0) = f^*$:

$$\frac{\partial f}{\partial t} = sf^2 \implies f(t) = \frac{f^*}{1 - f^* st}$$

From this, we see that $f(t) = \frac{1}{2}$ when

$$t_{1/2} = \frac{1}{s}(\frac{1}{f^*} - 2) = \frac{1}{s}(\sqrt{Ns} - 2) \sim \sqrt{\frac{N}{s}}$$

Both the time to the drift boundary and subsequent deterministic time to frequency 0.5 are of order $\sqrt{N/s}$, so in general the time to fixation is of order $\sqrt{N/s}$ (with probability $\sim \sqrt{s/N}$, as we saw before). When $Ns \gg 1$, the fixation probability is smaller than that of the haploid case, and the fixation time is larger.

Part C

The results from part b for below the drift barrier still apply for strongly deleterious mutations (a strongly deleterious mutant will not grow much past the drift barrier): a strongly deleterious mutant with initial size $\sim \frac{1}{N}$ drifts to final size $\sim \frac{1}{\sqrt{N|s|}}$ with probability $\sim \sqrt{\frac{|s|}{N}}$ on a timescale of $\sim \sqrt{\frac{N}{|s|}}$ generations. Plugging in numbers, we get that a recessive mutation with fitness effect $s \approx -1$ in a population of $N = 10^6$ individuals will typically grow to maximum frequency $\sim 10^{-3}$ and exist for $\sim 10^3$ generations.

Problem 2: The molecular diversity of adaptive convergence

Part A

We calculate dN/dS by computing (# of observed nonsynonymous mutations/# of possible nonsynonymous mutations, from problem set 1)/(# of observed synonymous mutations/# of possible synonymous mutations, from problem set 1). The numbers of possible mutations from the posted problem set 1 solutions as of February 24, 2021 (3,059,233 possible synonymous mutations, 404,289 possible nonsense mutations, and 8,587,451 possible missense mutations) yield a dN/dS for missense mutations of 4.81 and dN/dS for nonsense mutations of 4.04. Other reasonable choices of numbers may yield ratios near roughly 4-5 and 3-4, respectively. Either way, these are quite far from 1 and we can confidently say that mutations in both classes are positively selected.

Part B

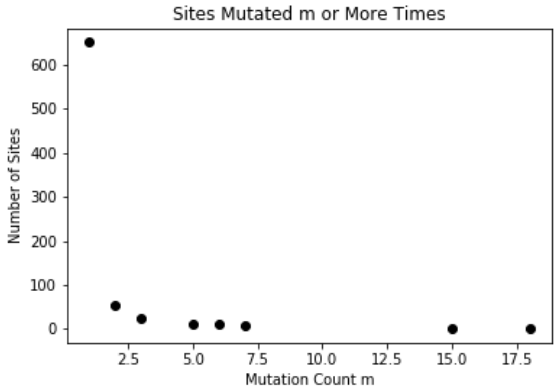


Figure 1: Number of sites mutated m or more times across all $n = 114$ replicates, plotted as a function of m.

If all 789 mutations were distributed evenly across the sites in the *E. coli* genome, then we would expect the number of mutations on each site to be roughly Poisson distributed. Given that the *E. coli* genome is $L = 4,629,812$ bp long, this distribution would have a mean of $\lambda = 789/4,629,812 \approx 1.7 \times 10^{-4}$. Thus, under our assumptions, the number of sites expected to have $\geq m$ mutations would be

$$L \times \left(1 - \sum_{k=0}^{m-1} \frac{\lambda^k e^{-\lambda}}{k!}\right)$$

Plugging in numbers, ≈ 788.9 sites would be expected to have ≥ 1 mutation, and ≈ 0.067 sites would be expected to have ≥ 2 mutations. Since $53 \gg 0.067$ sites had ≥ 2 mutations, we can safely say that sites with two or more mutations are likely to have experienced beneficial selection. Using this criterion, $192/789 \approx 24.3\%$ of observed mutation events are likely to have come from a beneficially selected site.

Part C

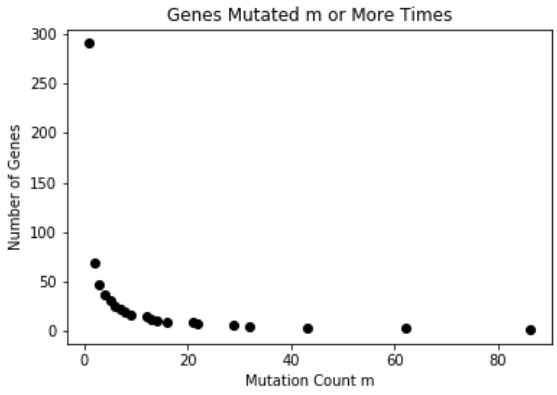


Figure 2: Number of genes mutated m or more times across all $n = 114$ replicates, plotted as a function of m.

Repeat a similar analysis as in part b. When counting synonymous, missense, nonsense, and within-gene indel mutations, 833 different genes were found to have mutated. Given that there are $L = 4,217$ genes in

the *E. coli* genome (and making the simplifying assumption that each gene is equally likely to mutate, i.e. is equally long), our Poisson distribution has $\lambda = 833/4217 \approx 0.198$. This yields ≈ 755.9 genes expected to have ≥ 1 mutation, ≈ 72.2 genes expected to have ≥ 2 mutations, and ≈ 4.67 genes expected to have ≥ 3 mutations. The observed values are 291, 69, and 46, respectively, and since $46 \gg 4.67$, we can safely say that genes with three or more mutations are likely to have experienced beneficial selection. Under this criterion, $565/833 \approx 67.8\%$ of observed mutation events are likely to have come from a beneficially selected gene.

Part D

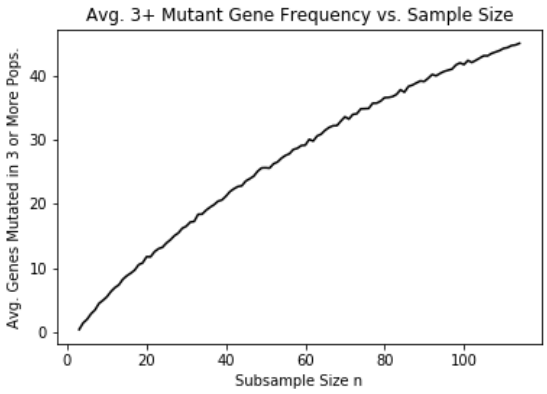


Figure 3: Empirical saturation curve, or average (over 100 trials) number of genes mutated (point or indel mutations within genes) in 3 or more randomly sampled populations of sample size $n = 3, \ldots, 114$.

The curve slows down but does not appear to fully saturate by $n = 114$.

Part E

The probability (exact, based on the binomial distribution) of observing m mutations in gene i (with probability p_i of being mutated in a given population) across ≥ 3 populations in an experiment with n total populations is:

$$1 - (1 - p_i)^n - np_i(1 - p_i)^{n-1} - \frac{n(n-1)}{2}p_i^2(1 - p_i)^{n-2}$$

Another decent approximation based on the Poisson distribution is

$$1 - e^{-np_i}\left(1 + np_i + \frac{n^2p_i^2}{2}\right)$$

Three theoretical saturation curves (the above probability plotted for different values of n) for $p_i = \frac{3}{114}, \frac{5}{114}, \frac{10}{114}$ are given below.

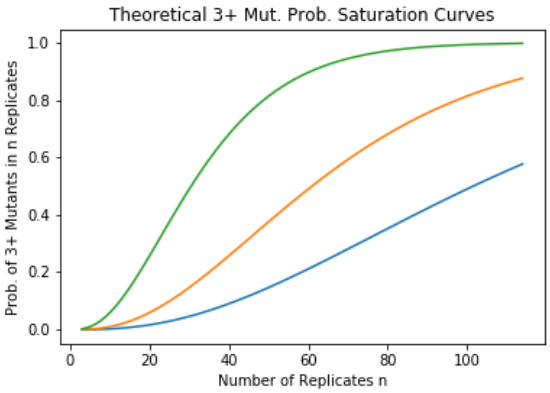


Figure 4: Theoretical saturation curves as described in the text. The blue curve corresponds to $p_i = \frac{3}{114}$, the orange curve corresponds to $p_i = \frac{5}{114}$, and the green curve corresponds to $p_i = \frac{10}{114}$.

For $p_i = \frac{3}{114}$, approximately 57.7% of beneficial genes will be detected in an experiment with $n = 114$ replicates, for $p_i = \frac{5}{114}$, approximately 87.5% will be detected, and for $p_i = \frac{10}{114}$, approximately 99.7% will be detected (these were calculated using the Poisson approximation). Our empirical saturation curve roughly resembles the curve for $p_i = \frac{5}{114}$ (except at small sample sizes, of course different genes could have different p_i's). 45 beneficial genes were detected for $n = 114$, and if these correspond to 87.5% of total beneficial genes, then ≈ 50 genes are likely to be beneficial in this environment.

Part F

We find that 43 total replicates have a (non-structural) mutation in *rho*, 29 total replicates have a mutation in *iclR*, and 20 replicates have mutations in both *rho* and *iclR*. By chance alone, we would expect

$$114 \times \frac{43}{114} \times \frac{29}{114} \approx 11$$

replicates to have mutations in both genes, which is only half as big as the observed value. Statistical significance can be assessed using Fisher's exact test — i.e., the probability that we observe 20 or more lines with both mutations by chance is given by

$$P = \sum_{k=20}^{29} \frac{\binom{43}{k}\binom{114-43}{29-k}}{\binom{114}{29}} \; 10^{-4} \tag{1}$$

This suggests that *iclR* mutations do tend to be more beneficial on a background of *rho* than without. However, 9 replicates still have a mutation in *iclR* alone, which is still significantly beneficial under our original 3-replicate threshold, suggesting that *iclR* are not exclusively beneficial in the presence of ρ.

Problem 3: Measuring the DFE for *de novo* beneficial mutations, Part I

We'll make use of the fact that this serial dilution experiment is equivalent to a diffusion model with an effective population size $N_e \sim N_0\Delta$. We'll then consider each of the four criteria in reverse order:

1. Each barcoded lineage will start at a characteristic frequency $f_0 \sim 1/B$. Genetic drift will require a time of order $\sim N_e f_0 = N_e/B$ generations to substantially perturb the frequency of these lineages, so we need to make sure that the total experimental duration is less than this time:

$$T \lesssim \frac{N_e}{B} \tag{2}$$

2. Conversely, beneficial mutations will require $\sim 1/s_b$ generations to substantially change the lineage frequency, so we want to make sure that the total duration is longer than this time. Combining with the condition above, this yields

$$\frac{1}{s_b} \lesssim T \lesssim \frac{N_e}{B} \tag{3}$$

3. Each barcoded lineage will produce $\sim (N_e/B) U_b s_b T$ successful beneficial mutations over the course of the experiment. We'll want this number to be $\ll 1$ so there is a small probablity of producing two beneficial mutations in the same lineage. Let's pick 0.01 for concretness (i.e., 99% of putatively adaptive barcodes will contain a single beneficial mutation). This leads to a condition,

$$\frac{N_e}{B} U_b s_b T \lesssim 0.01 \tag{4}$$

4. Finally, we want to make sure we have $\gtrsim 1000$ lineages with at least one mutation. If each of the B lineages produces a beneficial mutation with probability $\sim (N_e/B) U_b s_b T$, this requires that

$$N_e U_b s_b T \gtrsim 1000 \tag{5}$$

Now to plug in some numbers. If $s_b \sim 10^{-2}$, then the second condition requires that

$$T \gtrsim 100 \tag{6}$$

It's always easier if we run a shorter experiment, so let's see how far we can get with $T \sim 100$. If $U_b \sim 10^{-5}$, then the last condition requires that

$$N_e \gtrsim 10^8 \tag{7}$$

Meanwhile, the 3rd and 4th conditions together require that

$$B \gtrsim 10^5 \tag{8}$$

By choosing $N_e = 10^8$ and $B = 10^5$, we see that the first condition is satisfied. All four conditions are then satsified. To implement this in a serial dilution experiment, we'd want to make sure that $s\Delta t \ll 1$. This can be achived by taking $\Delta t = 10$ and a bottleneck size of $N_0 = 10^7$, and running the experiment for $T/\Delta t = 10$ days.

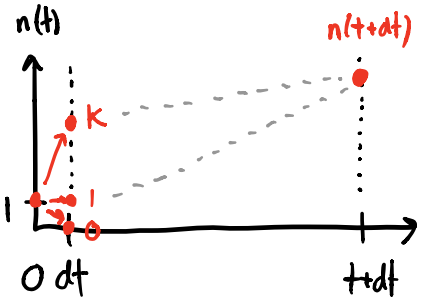
Finally, the sequencing depth must be chosen to be sufficiently high that we can resolve the relevant selection pressures. For a barcode at frequency $\sim 1/B$, the relative error in our frequency estimate will be of order $\sim \sqrt{B/D}$. If we want this to be no more than 10% at each timepoint, we will require

$$D \gtrsim 100B \tag{9}$$

or about $\sim 10^7$ reads per timepoint. All 10 timepoints for a single replicate could therefore fit on a single lane of Illumina sequencing ($\sim 10^8$ total reads).

Problem 4:

(a) By considering what happens in timeslice $[0, dt]$, we have:



so $\Pr[1 \to 0] = (1+d)dt$,

$\Pr(1 \to k) = (1+b)dt$

$\Pr(1 \to 1) = 1 - \Pr(1 \to 0) - \Pr(1 \to k) = 1 - (2+b+d)dt$

then

$$\left\langle e^{-z n(t+dt)} \right\rangle = \Big\langle \underbrace{\left[1-(2+b+d)dt\right] e^{-z n(t)}}_{\text{no change}} \cdot \underbrace{(1+d)dt \cdot e^{-z\cdot 0}}_{\text{death}} + \underbrace{(1+b)dt \cdot e^{-z\sum_{i=1}^{k} n_i(t)}}_{\text{birth}} \Big\rangle$$

same process $(1 \to n(t))$ but now w/ only t gens

$$\Rightarrow H(z,t+dt) = \left[1-(2+b+d)dt\right]H(z,t) + (1+d)dt + (1+b)dt\, H(z,t)^k$$

$\Rightarrow$ Taylor expanding for small dt:

$$\boxed{\frac{\partial H(z,t)}{\partial t} = (1+b)H(z,t)^k + (1+d) - (2+b+d)H(z,t)}$$

(b) When $k=2$, this equation reduces to

$$\frac{dH}{dt} = (1+b)H^2 + (1+d) - (2+b+d)H$$

$\Rightarrow$ can solve in Mathematica or by hand.

optional

$\Rightarrow$ in latter case, define $\Phi \equiv 1-H$.

$\Rightarrow$ then $\dfrac{d\bar{\Phi}}{dt} = -(1+b)(1-\bar{\Phi})^2 - (1+d) + (2+b+d)(1-\bar{\Phi})$

$= (b-d)\bar{\Phi} - (1+b)\bar{\Phi}^2$

$= (b-d)\bar{\Phi}\left[1 - \dfrac{(1+b)}{b-d}\bar{\Phi}\right]$

$\Rightarrow$ logistic eq w/ $r = b-d$ & $K = \dfrac{b-d}{1+b}$

$\Rightarrow$ solution $\bar{\Phi}(t) = \dfrac{\bar{\Phi}(0)\, e^{(b-d)t}}{1 + \bar{\Phi}(0)\cdot\frac{b-d}{1+b}\left(e^{(b-d)t} - 1\right)}$

$\Rightarrow$ $\boxed{H(t) = 1 - \dfrac{[1-H(0)]\, e^{(b-d)t}}{1 + (1-H(0))\,\frac{b-d}{1+b}\left(e^{(b-d)t} - 1\right)}}$

optional

$\Rightarrow$ initial condition is $H(0) = e^{-z}$, so we have

$$H(z,t) = 1 - \frac{(1-e^{-z})\, e^{(b-d)t}}{1 + (1-e^{-z})\left(\frac{b-d}{1+b}\right)\left(e^{(b-d)t} - 1\right)}$$

To compare w/ the diffusion model from class, we need to look @ the mutation frequency, $f \equiv \frac{n}{N}$.

$$H_f(z) = \langle e^{-zf} \rangle = \langle e^{-z \cdot \frac{n}{N}} \rangle = H_n\left(\frac{z}{N}\right)$$

$$= 1 - \frac{(1-e^{-\frac{z}{N}})\, e^{(b-d)t}}{1 + (1-e^{-z/N})\,\frac{b-d}{1+b} \cdot \left(e^{(b-d)t} - 1\right)}$$

In limit that $N \gg 1$ and $b, d \ll 1$, this becomes:

$$H_f(z) \simeq 1 - \frac{z\, e^{(b-d)t}}{1 + \frac{(b-d)z}{N}\left(e^{(b-d)t} - 1\right)}$$

which is identical to our diffusion model w/

$$\boxed{s_e \equiv b-d \quad \text{and} \quad N_e = \frac{N}{2}}$$

(c) Substituting $H(z) \approx 1 - z\langle f(t)\rangle$ into (*) & expanding to lowest order in z:

$$-z\frac{d\langle f(t)\rangle}{dt} = (1+b)\left(1-z\langle f(t)\rangle\right)^k + (1+d) - (2+b+d)\left(1-z\langle f(t)\rangle\right)$$

$$\approx (1+b) - (1+b)kz\langle f(t)\rangle + (1+d) - (2+b+d)\left(1-z\langle f(t)\rangle\right)$$

$$\Rightarrow \quad \frac{d\langle f(t)\rangle}{dt} = \left[(k-1)(1+b) - (1+d)\right]\langle f(t)\rangle$$

$\Rightarrow$ long term growth rate is $\boxed{s_e \equiv (k-1)(1+b) - (1+d)}$

(d) @ long times, generating function approaches a constant value $H(z,t) \approx (1-p)$.

substituting into (a) & expanding for $p \ll 1$:

$$0 = (1+b)(1-p)^k + (1+d) - (2+b+d)(1-p)$$

$$\approx (1+b)\left(1 - pk + \frac{k(k-1)}{2}p^2\right) + (1+d) - (2+b+d)(1-p)$$

$$= -\left[(k-1)(1+b) - (1+d)\right]p + \frac{(1+b)k(k-1)}{2}p^2$$

using fact that $S_e \equiv (k-1)(1+b) - (1+d) \Rightarrow (k-1)(1+b) = 1+d+S_e$

this reduces to

$$0 = -S_e p + \frac{(1+d+S_e)k}{2}p^2$$

$$\Rightarrow \boxed{p = \frac{2S_e}{1+d+S_e} \cdot \frac{1}{k}}$$

this is smaller than equivalent fixation prob.
of $K=2$ case (same s_e) by factor of $\approx 1/K$.

To derive this solution, we assumed that $pK \ll 1$

$\Rightarrow$ this is self-consistent if

$$Kp = \frac{2s_e}{1+d+s_e} \ll 1 \quad \Rightarrow \quad d, s_e \ll 1.$$

$\Rightarrow$ breaks down when $s_e \sim O(1)$ ($pK \gtrsim 1$)

this makes sense: a single burst event leads
to fixation w/ high probability.

optional

in this case, can do other dominant balance

$$(1+b)(1-p)^K \ll (1+d), (2+b+d)(1-p)$$

$\Rightarrow$ $p = \frac{1+b}{2+b+d}$ (i.e. probability that you get a birth before you die.)

$\Rightarrow$ self consistency: $(1+b)\left(\frac{1+d}{2+b+d}\right)^k < (1+d)$

Problem 5:

(a) In lecture 3 (p 12 of whiteboard) we derived

$$st = \log\left(\frac{f(t)}{1-f(t)} \cdot \frac{1-f(0)}{f(0)}\right)$$

$$\Rightarrow t = \frac{1}{s}\log\left(\frac{f(t)}{1-f(t)} \cdot \frac{1-f(0)}{f(0)}\right)$$

for $f(0) = 0.1$ & $f(t) = 0.9 \Rightarrow t \approx \frac{4}{s}$

for $f(0) \simeq 10^{-2}$ & $f(t) \simeq 1-10^{-2} \Rightarrow$ $\boxed{T_{sw} \approx \frac{9}{s}}$

(b) If $s \approx 1\%$ and $\Delta t \approx 7$ gens per day in LTEE expt,

$T_{sweep} \simeq 900$ gens $= 130$ days

if same mutation in gut microbiome (1-10 gens/day)

$T_{sweep} = 900$ gens $= 90 - 900$ days

(C) Compared to fixation timescale, $T_{fix} \simeq \frac{2}{s} \log(Ns)$:

LTEE ($N_e \simeq 3 \times 10^7$, $\Delta t \simeq 7$ gens per day):

$T_{fix} = 2500$ gens $\simeq 360$ days
(1 yr)

E. coli in gut ($N_e \sim 10^{12}$, $\Delta t \simeq 1-10$ gens/day)

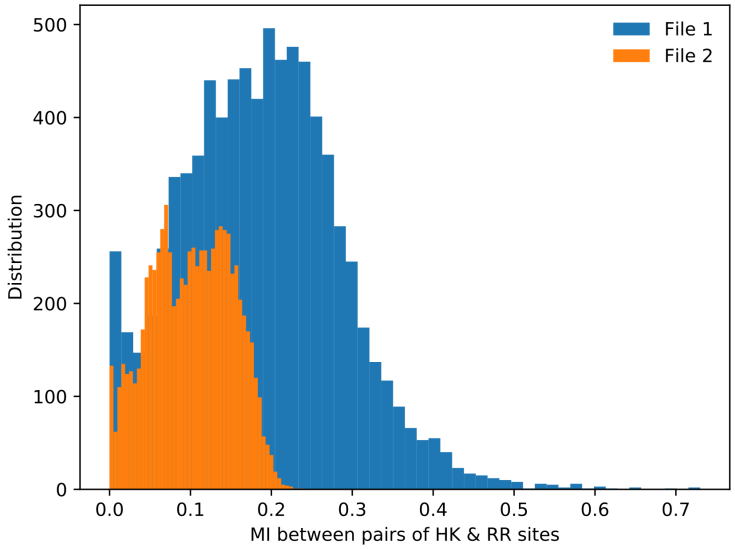
$T_{fix} = 4600$ gens $\simeq 460 - 4600$ days
(1 - 10 yrs!)

(d) Based on this answer, an individual would need to adhere to their diet for 1-10yrs before this loss of function mutation would sweep to high frequency!

In comparison, it could take as short as 3mo for the strain to sweep from 1% frequency.

Problem 6:

(a) After calculating the MI values for each pair of HK & RR sites in each file, the distribution of values is given by:



File 1 appears to have higher average & max MI values, so we conclude that it corresponds to the proper pairing of HK & RR proteins.

(b) The maximum MI value occurs @ sites

HK 27 + RR 10 (zero-based)

$\Rightarrow$ MI $\approx$ 0.73, 209 unique AA pairs.

(sites 27+14 are nearly indistinguishable w/ MI $\approx$ 0.728)

$\Rightarrow$ thus, if we wanted to alter the specificity of HK w/ one AA change, we'd want to change HK27.

(c) At mutation-selection balance, the frequency of the valley genotype saturates to $f_v \approx \frac{N}{s_d}$.

$\Rightarrow$ thus, $N f_v \cdot N = \frac{N N^2}{s_d}$ double mutants

will be produced every generation.

$\Rightarrow$ each one fixes w/ probability $p_{fix} \approx \frac{1}{N}$,

so the substitution rate of valley crossing pairs is given by $\boxed{N^2/5d}$ per generation.

$\Rightarrow$ if typical bacterial generation times are $\sim 1\text{-}10$ / day & bacterial mut'n rates are $\sim 3\times10^{-10} - 3\times10^{-9}$ / codon / gen

then after $\sim 4\times10^9$ years, we'd expect a total of

$$n = (4\times10^9 \text{ yrs})\left(\frac{365 \text{ days}}{\text{yr}}\right)\left(\frac{1\text{-}10 \text{ gens}}{\text{day}}\right)\left(\frac{10^{-19} - 10^{-17}}{5d}\right)\times 1000$$

$$= \frac{10^{-4} - 10^{-1}}{5d}$$ successful valley crossings across all 1000 gene families.

$\Rightarrow$ for relatively weak costs ($5d \sim 10^{-2}$) this works out to $n \sim 10^{-2} - 10^{1}$ successful crossings.

$\Rightarrow$ if HK+RR function is essential ($s_d \approx 1$)

$\Rightarrow$ $n \sim 10^{-4}$-10^{-1} successful crossings.

we can compare this to the ~ 200 unique AA pairs @ the HK27 RR10 sites in the dataset.

based on these #s, this form of "neutral" valley crossing does not seem like a super plausible explanation.

Sample code for Problem Set 3

```
# Code for Problem 2 of Problem Set 3

# -*- coding: utf-8 -*-
"""
Created on Mon Feb 17 00:47:03 2020

@author: Anita Kulkarni
"""

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

f = open("./data_files/problem_set_data/tenaillon_etal_2012_mutations.txt","r")
raw_data = f.readlines()
del(raw_data[0])

data = []
for i in range(len(raw_data)):
    l = raw_data[i].split(", ")
    # get rid of last two columns (allele, functional module)
    del(l[len(l)-1])
    del(l[len(l)-1])
    l[0] = int(l[0][4:len(l[0])]) # lineage number
    l[1] = int(l[1]) # mutation location
    l = tuple(l)
    data.append(l)

# Part A
n_synonymous = 0
n_missense = 0
n_nonsense = 0
for i in range(len(data)):
    if data[i][3] == 'synonymous':
        n_synonymous = n_synonymous + 1
    elif data[i][3] == 'missense':
        n_missense = n_missense + 1
    elif data[i][3] == 'nonsense':
        n_nonsense = n_nonsense + 1
possible_synonymous = 3059233
possible_missense = 8587451
possible_nonsense = 404289
S = n_synonymous/possible_synonymous
N1 = n_missense/possible_missense
N2 = n_nonsense/possible_nonsense
print(N1/S)
print(N2/S)

# Part B
point_mutation_sites = []
for i in range(len(data)):
    if data[i][3] == 'synonymous' or data[i][3] == 'missense' or data[i][3] == 'nonsense' or data[i]
```

```
        point_mutation_sites.append(data[i][1])
print(len(point_mutation_sites))
unique_mut_sites = list(set(point_mutation_sites))
m_sites = []
for val in unique_mut_sites:
    m_sites.append(point_mutation_sites.count(val))
unique_m_sites = list(set(m_sites))
unique_m_sites.sort(reverse=True)
m_sites_freq = []
for val in unique_m_sites:
    m_sites_freq.append(m_sites.count(val))
m_sites_freq_cumulative = []
cum_sum = 0
for val in m_sites_freq:
    cum_sum = cum_sum + val
    m_sites_freq_cumulative.append(cum_sum)
plt.plot(unique_m_sites, m_sites_freq_cumulative, 'ko')
plt.xlabel('Mutation Count m')
plt.ylabel('Number of Sites')
plt.title('Sites Mutated m or More Times')
plt.savefig('AP237_PS3_Problem2_1.png')
plt.show()
print(unique_m_sites)
print(m_sites_freq)
print(m_sites_freq_cumulative)

# Part C
mutations_genes = []
for i in range(len(data)):
    if (data[i][3] == 'synonymous' or data[i][3] == 'missense'
        or data[i][3] == 'nonsense' or data[i][3] == 'indel') and data[i][2] != 'intergenic':
        mutations_genes.append(data[i][2])
print(len(mutations_genes))
unique_mut_genes = list(set(mutations_genes))
m_genes = []
for val in unique_mut_genes:
    m_genes.append(mutations_genes.count(val))
unique_m_genes = list(set(m_genes))
unique_m_genes.sort(reverse=True)
m_genes_freq = []
for val in unique_m_genes:
    m_genes_freq.append(m_genes.count(val))
m_genes_freq_cumulative = []
cum_sum = 0
for val in m_genes_freq:
    cum_sum = cum_sum + val
    m_genes_freq_cumulative.append(cum_sum)
plt.plot(unique_m_genes, m_genes_freq_cumulative, 'ko')
plt.xlabel('Mutation Count m')
plt.ylabel('Number of Genes')
plt.title('Genes Mutated m or More Times')
plt.savefig('AP237_PS3_Problem2_2.png')
plt.show()
print(unique_m_genes)
```

```
print(m_genes_freq)
print(m_genes_freq_cumulative)

# Part D
# first make a list of lists of genes in each replicate
gene_lists = []
rep = 1
i = 0
while i < len(data):
    g = []
    while data[i][0] == rep:
        if data[i][2] != 'intergenic' and data[i][3] != 'structural':
            g.append(data[i][2])
        i = i + 1
        if i >= len(data):
            break
    if i < len(data):
        rep = data[i][0]
    gene_lists.append(g)
# create empirical saturation curve
num_subsets = 100
avg_mutated_genes = []
for n in range(3, 115):
    total_avg_3_mut = 0
    for i in range(num_subsets):
        s = random.sample(gene_lists, n)
        s_flattened = []
        for l in s:
            s_flattened = s_flattened + list(set(l)) # 3 or more *populations*
        u_genes = list(set(s_flattened))
        u_genes_freq = []
        for val in u_genes:
            u_genes_freq.append(s_flattened.count(val))
        num_mut_3 = 0
        for item in u_genes_freq:
            if item >= 3:
                num_mut_3 = num_mut_3 + 1
        total_avg_3_mut = total_avg_3_mut + num_mut_3
    avg_mutated_genes.append(total_avg_3_mut/num_subsets)
print(avg_mutated_genes[len(avg_mutated_genes)-1])
plt.plot(np.arange(3, 115), avg_mutated_genes, 'k-')
plt.xlabel('Subsample Size n')
plt.ylabel('Avg. Genes Mutated in 3 or More Pops.')
plt.title('Avg. 3+ Mutant Gene Frequency vs. Sample Size')
plt.savefig('AP237_PS3_Problem2_3.png')
plt.show()

# Part E
n = np.arange(3, 115)
def sat_func(l):
    return 1 - np.exp(-l)*(1+l+(0.5*l*l))
sat_1 = sat_func((3/114)*n)
sat_2 = sat_func((5/114)*n)
sat_3 = sat_func((10/114)*n)
```

```
plt.plot(n, sat_1)
plt.plot(n, sat_2)
plt.plot(n, sat_3)
plt.xlabel('Number of Replicates n')
plt.ylabel('Prob. of 3+ Mutants in n Replicates')
plt.title('Theoretical 3+ Mut. Prob. Saturation Curves')
plt.savefig('AP237_PS3_Problem2_4.png')
plt.show()

# Part F
rho_iclR_simultaneous = 0
rho = 0
iclR = 0
for i in range(len(gene_lists)):
    if ('rho' in gene_lists[i]) and ('iclR' in gene_lists[i]):
        rho_iclR_simultaneous = rho_iclR_simultaneous + 1
    if 'rho' in gene_lists[i]:
        rho = rho + 1
    if 'iclR' in gene_lists[i]:
        iclR = iclR + 1
print(rho_iclR_simultaneous)
print(rho)
print(iclR)
```

```
# Code for Problem 6 of Problem Set 3

import sys
import numpy
import pylab
from math import log

hk_len = 71
rr_len = 116

total_data = []
hks = []
file=open("../data_files/skerker_etal_hk_alignment.txt","r")
for line in file:
    hks.append(line.strip())
    total_data.extend(line.strip())
file.close()

total_data = set(list(total_data))

#for item in sorted(total_data):
#    print item
#print len(total_data)

pylab.figure(1)
pylab.xlabel('MI between pairs of HK & RR sites')
pylab.ylabel('Distribution')
for file_idx in [1,2]:

    MI_matrix = numpy.zeros((hk_len, rr_len))
    num_AA_pairs = numpy.zeros((hk_len,rr_len))
    I_vector_hk = numpy.zeros(hk_len)
    I_vector_rr = numpy.zeros(rr_len)
    rrs = []
    file=open("../data_files/skerker_etal_rr_alignment_%d.txt" % file_idx, "r")
    for line in file:
        rrs.append(line.strip())
    file.close()

    for i in xrange(0,hk_len):
        #print i
        for j in xrange(0,rr_len):
            hk_aas = {}
            rr_aas = {}
            joint_aas = {}
            total = 0
            for hk,rr in zip(hks,rrs):
                hk_aa = hk[i]
                rr_aa = rr[j]
                if hk_aa == '-' or rr_aa == '-':
                    pass
                else:
                    if hk_aa not in hk_aas:
                        hk_aas[hk_aa] = 0
```

```
                    hk_aas[hk_aa]+=1
                    if rr_aa not in rr_aas:
                        rr_aas[rr_aa] = 0
                    rr_aas[rr_aa]+=1
                    if (hk_aa,rr_aa) not in joint_aas:
                        joint_aas[(hk_aa,rr_aa)] = 0
                    joint_aas[(hk_aa,rr_aa)] += 1
                    total += 1.0

            MI = 0
            I_hk = 0
            I_rr = 0
            if True: #total > 1167: # Skerker et al thresholded on no more than 10% gaps

                # normalize
                for aa in hk_aas.keys():
                    hk_aas[aa] *= 1.0/total
                for aa in rr_aas.keys():
                    rr_aas[aa] *= 1.0/total
                for aa1,aa2 in joint_aas.keys():
                    joint_aas[(aa1,aa2)] *= 1.0/total

                for aa1,aa2 in joint_aas.keys():
                    MI += joint_aas[(aa1,aa2)]*log(joint_aas[(aa1,aa2)]/hk_aas[aa1]/rr_aas[aa2])/log(

                for aa in hk_aas.keys():
                    I_hk += -1*hk_aas[aa]*log(hk_aas[aa])/log(2)

                for aa in rr_aas.keys():
                    I_rr += -1*rr_aas[aa]*log(rr_aas[aa])/log(2)

            num_AA_pairs[i,j] = len(joint_aas.keys())
            MI_matrix[i,j] = MI
            I_vector_hk[i] = I_hk
            I_vector_rr[j] = I_rr

    max_MI = MI_matrix.max()
    for i,j in zip(*numpy.nonzero(MI_matrix>=(max_MI*0.9))):
        print i,j,MI_matrix[i,j],num_AA_pairs[i,j]

    print "File %d, Mean MI=%g, Max MI=%g" % (file_idx, MI_matrix.mean(), MI_matrix.max())

    print "Argmax:", numpy.unravel_index(numpy.argmax(MI_matrix, axis=None), MI_matrix.shape)

    pylab.hist(MI_matrix.flatten(),bins=50,label=('File %d' % file_idx))
    #pylab.figure()
    #pylab.title('Mutual information matrix')
    #pylab.ylabel('HK Position')
    #pylab.xlabel('RR Position')
    #c = pylab.pcolor(MI_matrix,vmin=0,vmax=0.8)
    #(pylab.gcf()).colorbar(c, ax=pylab.gca())

    #pylab.figure()
    #pylab.plot(I_vector_hk)
```

```
    #pylab.ylabel('HK Entropy')
    #pylab.xlabel('Position')
    #pylab.figure()
    #pylab.plot(I_vector_rr)
    #pylab.ylabel('HK Entropy')
    #pylab.xlabel('Position')
    #pylab.figure()
    #pylab.plot(MI_matrix.max(axis=1))
    #pylab.ylabel('Max Mutual information w/ RR')
    #pylab.xlabel('HK Position')
    #pylab.figure()

#pylab.show()
pylab.legend(loc='upper right',frameon=False)
pylab.savefig('problem_6_a.pdf',bbox_inches='tight')
```