AP237/Bio251 Problem Set 4 Solutions

Written/compiled by: Benjamin Good and Anita Kulkarni

March 15, 2021

Problem 1: Measuring the DFE for de novo beneficial mutations, Part II

Part A

We would like to show that the MGF given in equation 20 is a good model of the data. At $\tau = 0$,

$$H(z|\hat{f}_{i,0}) \approx \exp\left[-\frac{z\hat{f}_{i,0}[1+(X_{i,0}-\overline{X}_0)\Delta t_0]}{1+z\kappa_0/D_0}\right] = \exp\left(-\frac{z\hat{f}_{i,0}}{1+z\kappa_0/D_0}\right)$$

$$\implies -\log H = \frac{z\hat{f}_{i,0}}{1+z\kappa_0/D_0} \implies -\frac{1}{\log H} = \frac{1}{z\hat{f}_{i,0}} + \frac{\kappa_0}{D_0\hat{f}_{i,0}} = \frac{D_0}{zR_{i,0}} + \frac{D_0\kappa_0}{D_0R_{i,0}} \implies -\frac{R_{i,0}}{\log H} = \frac{D_0}{z} + \kappa_0$$

How do we find H? If we choose only the lineages with exactly 50 reads at $\tau = 0$, then $R_{i,0} = 50$, and we can estimate H evaluating the empirical MGF at $\tau = 1$ (since all lineages with the same number of reads at $\tau = 0$ would be expected to have the same $p(\hat{f}_{i,1}|\hat{f}_{i,0})$), i.e.

$$\hat{H}(z) = \frac{1}{n}\sum_i \exp\left(-z\hat{f}_{i,1}\right)$$

evaluated for z at "typical" values of $1/\hat{f}_{i,1}$.

Now we calculate $\hat{H}$ as defined above. Actually, to make the numbers nicer, we (optionally) redefine

$$\hat{H}(z') = \frac{1}{n}\sum_i \exp\left(-z'R_{i,1}\right)$$

so that z' should actually be chosen around typical values of $1/R_{i,1}$, and our new fitting equation as

$$-\frac{50}{\log H} = \frac{D_0}{D_1}\frac{1}{z'} + \kappa_0$$

(this is how the (or one) sample code is written). Calculating $\hat{H}$ and fitting to typical values of z', we find that a linear fit works very well, that $\kappa_0 \approx 10.01$ (intercept), and that the fitted $D_0/D_1 \approx 2.618$ (slope) comes quite close to the actual $D_0/D_1 \approx 2.651$. Thus, the MGF in equation 20 appears to be a good model of the data (specifically, the conditional distribution is consistent with the approximation in equations 20 and 21).

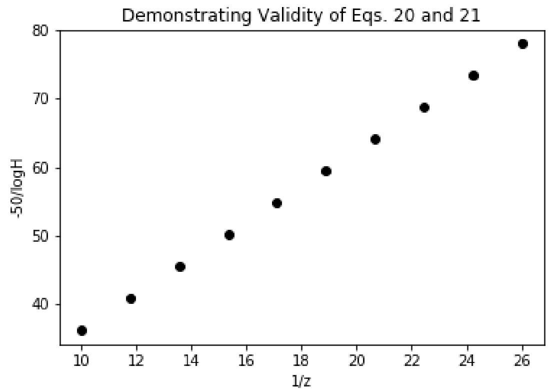


Figure 1: Relation between $-50/\log \hat{H}$ and $1/z$; we see a strong linear relation, demonstrating the validity of our MGF.

Part B

Since we assume that all lineages with $R_{i,\tau} \in [20, 60]$ remain neutral for all τ, let $X_{i,\tau} = 0$ for all τ for these lineages. We have

$$-\log H = \frac{z\frac{R_{i,\tau}}{D_\tau}\left(1 - \overline{X}_\tau \Delta t_\tau\right)}{1 + z\frac{\kappa_\tau}{D_\tau}} = \frac{zR_{i,\tau}\left(1 - \overline{X}_\tau \Delta t_\tau\right)}{D_\tau + z\kappa_\tau} \implies -\frac{R_{i,\tau}}{\log H} = \frac{1}{z}\frac{D_\tau}{1 - \overline{X}_\tau \Delta t_\tau} + \frac{\kappa_\tau}{1 - \overline{X}_\tau \Delta t_\tau}$$

We can perform the same fitting procedure as in part a at all τ (except the final one) with all lineages with $R_{i,\tau} \in [20, 60]$ using the same definition of $\hat{H}$ as in part a. If the inferred slope and intercept of a given fit are m and b, respectively, then $\overline{X}_\tau$ can be estimated as

$$\overline{X}_\tau = \frac{1}{\Delta t_\tau}\left(1 - \frac{D_\tau}{m}\right)$$

and κ_τ can be estimated as

$$\kappa_\tau = \frac{D_\tau b}{m}$$

Averaging these estimates for all fits performed at a given τ and plotting them as a function of τ, we get the following:

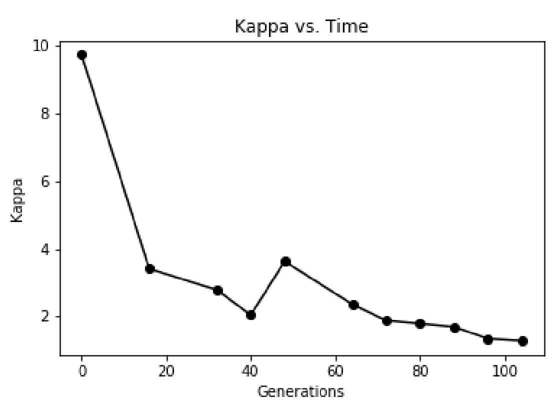


Figure 2: Change in inferred κ_τ over the course of the experiment.

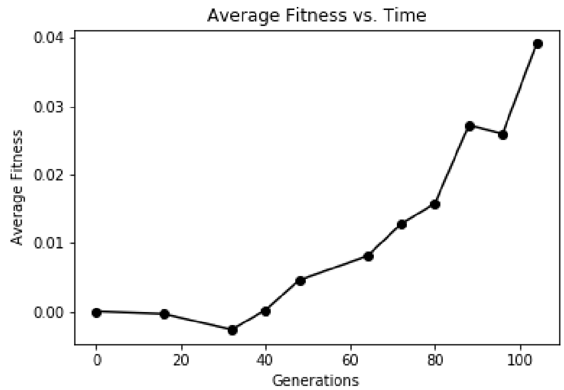


Figure 3: Change in inferred mean fitness over the course of the experiment.

Finally, we can estimate the fold change in frequency of a neutral lineage over the course of the experiment by computationally carrying out the integral

$$\exp\left(-\int_0^{t_f} \overline{X}(t')dt'\right) \approx 0.4$$

(derive this using $f(t) = f_0 \exp\left(\int_0^t (0 - \overline{X}(t'))dt'\right)$ for a neutral lineage). Thus, on average, we would expect to see the frequency of a neutral lineage drop by a factor of roughly 2.5 over the course of the experiment (of course, this is very noisy).

Part C

We have that

$$H(z|\hat{f}_{i,\tau}) = H(z|(\hat{f}^0_{i,\tau} + \hat{f}^s_{i,\tau})) = H_{\hat{f}^0_{i,\tau+1}}(z) H_{\hat{f}^s_{i,\tau+1}}(z)$$

where $H_{\hat{f}^0_{i,\tau+1}}(z)$ only depends on $\hat{f}^0_{i,\tau}$ and $H_{\hat{f}^s_{i,\tau+1}}(z)$ only depends on $\hat{f}^s_{i,\tau}$. Plug in the expressions for H:

$$H(z|\hat{f}_{i,\tau}) = \exp\left[-\frac{z\hat{f}^0_{i,\tau}(1-\overline{X}_\tau \Delta t_\tau)}{1+z\kappa_\tau/D_\tau}\right] \exp\left[-\frac{z\hat{f}^s_{i,\tau}\left[1+(s-\overline{X}_\tau)\Delta t_\tau\right]}{1+z\kappa_\tau/D_\tau}\right]$$

$$= \exp\left[-\frac{z\left[\hat{f}^0_{i,\tau} - \hat{f}^0_{i,\tau}\overline{X}_\tau \Delta t_\tau + \hat{f}^s_{i,\tau} + \hat{f}^s_{i,\tau} s \Delta t_\tau - \hat{f}^s_{i,\tau}\overline{X}_\tau \Delta t_\tau\right]}{1+z\kappa_\tau/D_\tau}\right]$$

$$= \exp\left[-\frac{z\hat{f}_{i,\tau}\left[1+\left(s\left(\hat{f}^s_{i,\tau}/\hat{f}_{i,\tau}\right)-\overline{X}_\tau\right)\Delta t_\tau\right]}{1+z\kappa_\tau/D_\tau}\right] \implies X_{i,\tau,\text{eff}} = s\frac{\hat{f}^s_{i,\tau}}{\hat{f}_{i,\tau}}$$

Part D

Use Bayes' theorem to come up with the following posterior odds ratio:

$$\frac{P(s,t_0|\{\hat{f}_{i,\tau+1}\})}{P(t_0=\infty|\{\hat{f}_{i,\tau+1}\})} = \frac{P(\{\hat{f}_{i,\tau+1}\}|s,t_0)P(s,t_0)}{P(\{\hat{f}_{i,\tau+1}\}|t_0=\infty)P(t_0=\infty)}$$

Plots of trajectory 14 are shown below. Either of the following could have been interpreted as "trajectory 14" depending on whether 14 was taken to be the barcode ID or a 0-based coordinate.

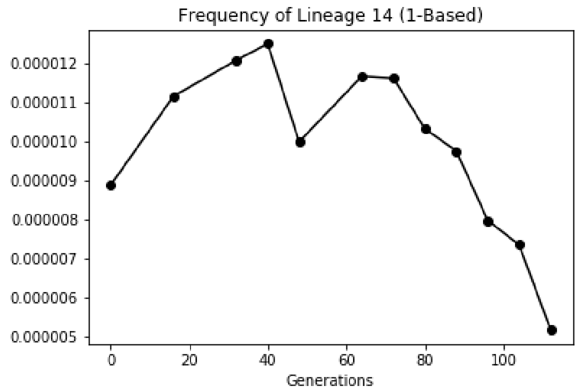


Figure 4: Trajectory of lineage 14 (barcode ID 14).

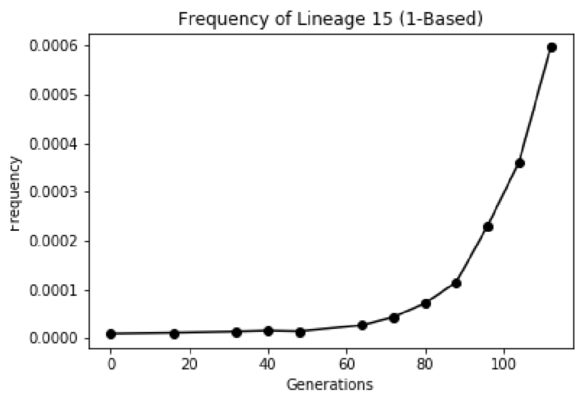
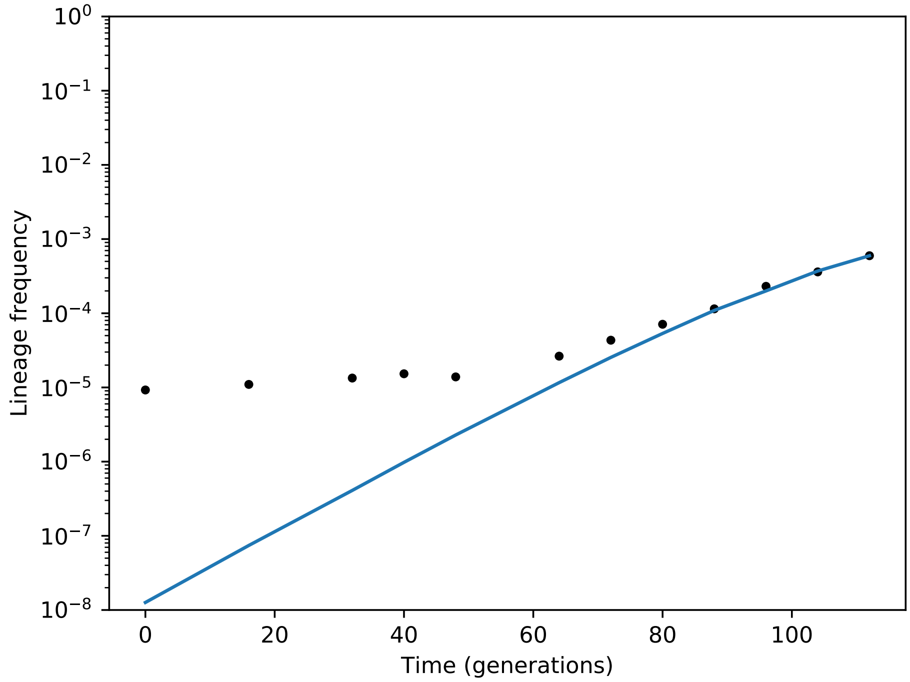


Figure 5: Trajectory of lineage 15 (barcode ID 15) or lineage 14 in 0-based coordinates.

The fitted version of lineage 14 in 0-based coordinates looks like:



with a best fit selection coefficient of $s \approx 10\%$.

Part E

The code below processed the first 1000 barcodes in ∼3 seconds and found 96 beneficial barcodes. At this rate, we estimate that it will take about a half hour to process the entire set of 5×10^5 barcodes.

Consistent with this estimate, running the full dataset took about 22 minutes to run and found ∼12,000 beneficial barcodes.

Part F

We want to find a t^* such that by the end of the experiment (time t_f), the frequency of a neutral lineage would be of roughly the same order of magnitude as $f(t_f|s,t^*)$:

$$\frac{c}{N_b s}\exp\left[\int_{t^*}^{t_f}\left(s-\overline{X}(t')\right)dt'\right] \approx f_0 \exp\left(-\int_0^{t_f}\overline{X}(t')dt'\right)$$

$$\implies \frac{c}{f_0 N_b s}\exp\left(\int_{t^*}^{t_f} s dt'\right)\exp\left(-\int_{t^*}^{t_f}\overline{X}(t')dt'\right) = \exp\left(-\int_0^{t_f}\overline{X}(t')dt'\right)$$

From part b, we know that the right hand side is approximately 0.4, which means that $\exp\left(-\int_{t^*}^{t_f}\overline{X}(t')dt'\right)$ must be between 0.4 and 1. Since this is a rough order of magnitude calculation, removing a term of this order will not significantly affect the final answer (provided that s isn't too large, technically), so

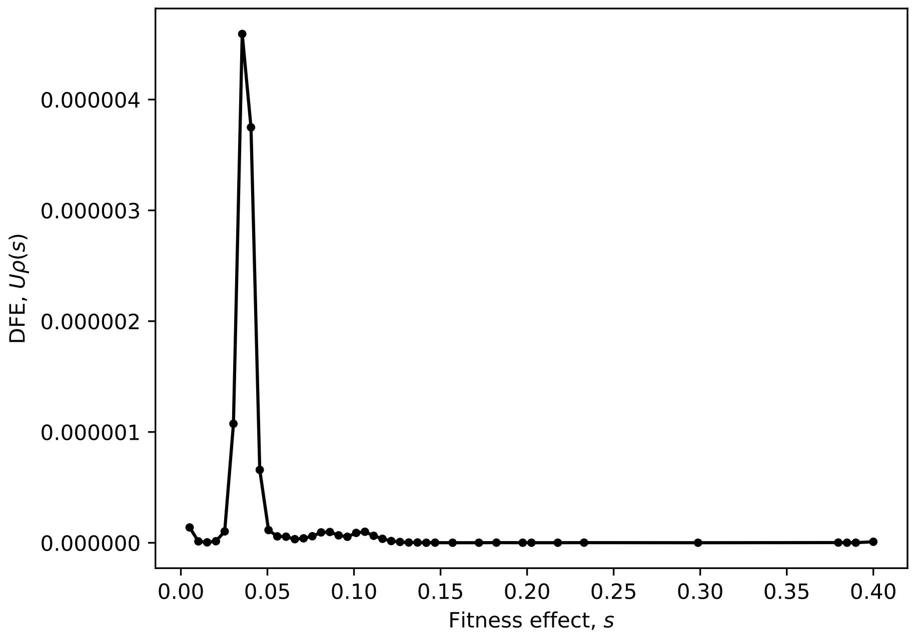
$$s(t_f - t^*) = \log\left(\frac{0.4 f_0 N_b s}{c}\right) \approx \log\left(\frac{f_0 N_b s}{c}\right) \implies t^* \approx t_f - \frac{1}{s}\log\left(\frac{f_0 N_b s}{c}\right)$$

The formula for the DFE is given by

$$U_b \rho(s)\delta s \approx \frac{c n(s)}{N_b s \int_0^{t^*(s)} e^{-\overline{X}(t)} dt}$$

.

Applying this formula to the results from the full dataset, we obtain the following estimate of the DFE:



Problem 2: Genealogies from sequences of neutral mutations

Note: there are lots of trees that are compatable with the sequences listed in parts (a-d) and part (f). Here we've listed just one set of possibilities that work.

Part A

Assume A is ancestral and the red line indicates a mutation from A to T.

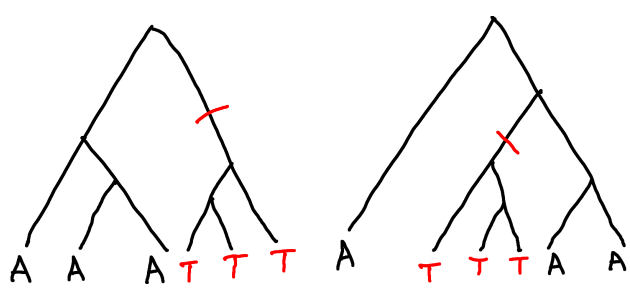


Figure 6: Genealogies for mutation pattern a.

Part B

Assume AG is ancestral, the red line denotes a mutation from A to T, and the blue line denotes a mutation from G to C.

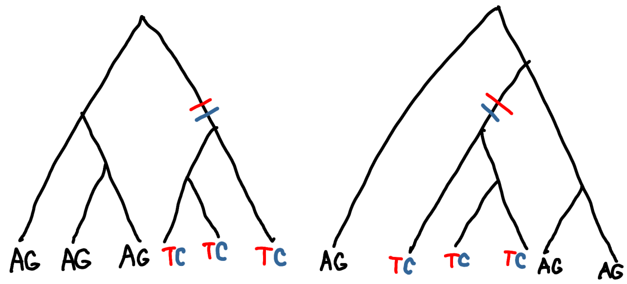


Figure 7: Genealogies for mutation pattern b.

Part C

Once again, assume AG is ancestral, the red line denotes a mutation from A to T, and the blue line denotes a mutation from G to C.

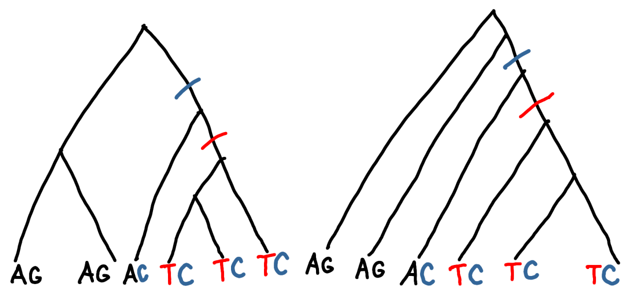


Figure 8: Genealogies for mutation pattern c.

Part D

There are 2 variable sites and 4 distinct haplotypes spread across 6 individuals, so there cannot be a consistent genealogy where each mutation happens only once. To see this, consider the 6 unique binary trees for $n = 6$:

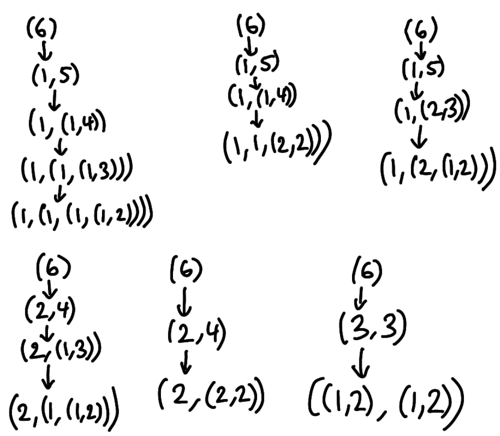


Figure 9: Unique binary trees for $n = 6$.

A single mutation affecting exactly 3 organisms (here, either mutating A to T or G to C) would need to happen where there is a (1,2), but we see that all instances of these necessitate that either the other nucleotide is the same in all 3 first-site-mutant organisms, or there is exactly a 4/2 split in the frequencies at the other site (of which the 2 must both have the same mutant first site), neither of which is true in our scenario. So single mutations at each site cannot give rise to our scenario.

Part E

The diagram from part d helps us find the right tree architecture for this scenario (ancestral sequence is AGTG).

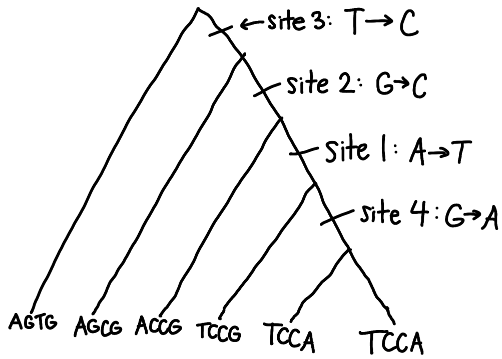


Figure 10: Genealogy for mutation pattern e.

Problem 3

(a) There are two different ways we can do this part:

Method 1: In Problem 5 of PSET 1, we showed that a collection of N_0 strains w/ fitnesses x_i evolve under selection as

$$f_i(t) = \frac{e^{x_i t}}{\sum_{j=1}^{N_0} e^{x_j t}}$$

the mean fitness of the population is therefore given by:

$$\bar{x}(t) = \sum_{i=1}^{N} x_i f_i(t) = \frac{\sum_{i=1}^{N_0} x_i e^{x_i t}}{\sum_{j=1}^{N_0} e^{x_j t}}$$

taking derivatives, we find that:

$$\partial_t \bar{x}(t)\Big|_{t=0} = \left(\frac{\sum_{i=1}^{N_0} x_i^2 e^{x_i t}}{\sum_{j=1}^{N_0} e^{x_j t}} - \frac{\sum_{i=1}^{N_0} x_i e^{x_i t} \sum_{j=1}^{N_0} x_j e^{x_j t}}{\left(\sum_{j=1}^{N_0} e^{x_j t}\right)^2} \right)\Bigg|_{t=0}$$

$$= \frac{1}{N_0}\sum_{i=1}^{N_0} x_i^2 - \left(\frac{1}{N_0}\sum_{i=1}^{N_0} x_i\right)^2$$

$$= \int x^2 f(x)\,dx - \left(\int x f(x)\,dx\right)^2$$

where $f(x) \sim \text{Gaussian}(0, V)$ is the fitness distribution of the hybrid offspring.

This shows that $\boxed{\partial_t \bar{x}\big|_{t=0} = V}$

Method 2: we can do the same thing directly from our multi-locus SDE model w/o mutation & recombination:

$$\left\langle \frac{\partial \bar{x}(t)}{\partial t} \right\rangle \equiv \left\langle \frac{\partial}{\partial t} \sum_{\vec{g}} x(\vec{g}) f(\vec{g}) \right\rangle = \left\langle \sum_{\vec{g}} x(\vec{g}) \frac{\partial f(\vec{g})}{\partial t} \right\rangle$$

$$= \left\langle \sum_{\vec{g}} x(\vec{g}) \left[(x(\vec{g}) - \bar{x}) f(\vec{g}) + \sqrt{\frac{f(\vec{g})}{N}}\,\eta(\vec{g}) - f(\vec{g}) \sum_{\vec{g}'} \sqrt{\frac{f(\vec{g}')}{N}}\,\eta(\vec{g}') \right\rangle$$

$$= \left\langle \sum_{\vec{g}} X(\vec{g})^2 f(\vec{g}) \right\rangle - \left\langle \left(\sum_{\vec{g}} X(\vec{g}) f(\vec{g}) \right)^2 \right\rangle + 0 + 0$$

@ $t=0 \Rightarrow f(\vec{g},0) = \sum_{i=1}^{N_0} \frac{1}{N_0} \delta_{\vec{g},\vec{g}_i}$

where $\vec{g}_i$ is the genotype of the ith hybrid founder.

Thus,
$$\left\langle \frac{\partial \bar{x}(t)}{\partial t}\Big|_{t=0} \right\rangle = \frac{1}{N_0} \sum_{i=1}^{N_i} x_i^2 - \left(\frac{1}{N_0} \sum_{i=1}^{N_0} x_i \right)^2$$

$$= \int x^2 f(x)\,dx - \left(\int x f(x)\,dx \right)^2$$

$$= V \quad \text{as above.}$$

(b) Each of the N_0 founder lineages establishes w/ probability $\sim 2x_i$. Thus, the expected # of established lineages w/ fitness $\geq x^*$ is given by

$$n_>(x^*) = N_0 \int_{x^*}^{\infty} 2x f(x)\,dx = \int_{x^*}^{\infty} 2x \frac{1}{\sqrt{2\pi V}} e^{-\frac{x^2}{2V}} dx$$

$$= \frac{2N_0 V}{\sqrt{2\pi V}} e^{-x^2/2V} \Big|_{x^*}^{\infty} = N_0 \sqrt{V} \sqrt{\frac{2}{\pi}} e^{-\frac{x^{*2}}{2V}}$$

The typical maximum fitness will occur when $n_>(X_{max}) \sim 1$. Solving for X_{max}, we obtain:

$$\boxed{X_{max} \simeq \sqrt{2v} \cdot \log^{1/2}\left(N_0 \sqrt{v}\right)}$$

(c) If recombination is high enough that sites evolve independently, then the frequency of the + allele @ the ℓ^{th} site is satisfies the single locus equation:

$$\frac{\partial f_\ell}{\partial t} = s f_\ell(1-f_\ell) + \sqrt{\frac{f_\ell(1-f_\ell)}{N}}\,\eta(t) \approx s f_\ell(1-f_\ell)$$

where we have assumed that N is sufficiently large that individual founder lineages do not change much on our experimental timescales $\left(\frac{N_0 t}{N} \ll 1\right)$. The solution is our familiar logistic function,

$$f_\ell(t) = \frac{f_\ell(0)e^{st}}{1+f_\ell(0)(e^{st}-1)} = \frac{e^{st}}{1+e^{st}} \qquad \left(\text{since } f_\ell(0) \approx \tfrac{1}{2}\right)$$

the mean fitness then grows as

$$\bar{X}(t) = \sum_{\ell=1}^{L} \frac{s}{2} f_\ell(t) + \left(-\frac{s}{2}\right)(1-f_\ell)) = \sum_{\ell=1}^{L}\left(s f_\ell(t) - \frac{s}{2}\right)$$

$$= \frac{Ls}{2}\left(\frac{2e^{st}}{1+e^{st}} - 1\right) = \frac{Ls}{2}\cdot\frac{e^{st}-1}{e^{st}+1}$$

$$= V\cdot\frac{2}{s}\cdot\frac{e^{st}-1}{e^{st}+1}$$ which matches part (a) when $t=0$.

The mean fitness will reach X_{max} from part (b) when

$$\tilde{X}(t) = V\cdot\frac{2}{s}\cdot\frac{e^{st}-1}{e^{st}+1} = X_{max} \approx \sqrt{2V}\,\log^{1/2}(N_0\sqrt{V})$$

Since we have assumed $s\to 0$ & $L\to\infty$ w/ V held fixed, we can Taylor expand the exponentials in $\bar{X}(t)$, leaving

$$\bar{X}(t) \approx Vt = \sqrt{2V}\,\log^{1/2}(N_0\sqrt{V})$$

or

$$\boxed{t = \sqrt{\frac{2}{V}}\, \log^{1/2}\!\left(N_0 \sqrt{V}\right)}$$

Since this expression is finite as $s \to 0$ & $L \to \infty$, we will always have $st \ll 1$ on this timescale, and

$$f_\ell(t) \approx \frac{1}{2} + \mathcal{O}(st) \quad \text{(i.e. allele freqs have barely changed)}$$

this implies that the rate of adaptation is still $\frac{\partial \bar{X}(t)}{\partial t} = V$, when the corresponding value for asexual populations starts to decline significantly.

Sample code for Problem Set 3

```
# Problem 1 of Problem Set 4

import pylab
import numpy
import sys
from math import exp
from scipy.stats import linregress

# Load Data from File
filename = "../data_files/levy_blundell_etal_2015_barcode_trajectories.txt"
file = open(filename,"r")
header = file.readline()
header_items = header.split(",")
ts = numpy.array([int(item.split("=")[1]) for item in header_items[1:]])
print("Loading trajectories...")
coverage_trajectories = []
for line in file:
    items = line.split(",")
    trajectory = [int(item) for item in items[1:]]
    coverage_trajectories.append(trajectory)
coverage_trajectories = numpy.array(coverage_trajectories)
depths = coverage_trajectories.sum(axis=0)
frequency_trajectories = coverage_trajectories*1.0/depths[None,:]
print("Done!")
print("Total coverage at each timepoint", depths)
print("Frequency of $R=50: ", 50*1.0/depths[0])

# Set up some figures
pylab.figure(1)
# generating function for rare lineages (R0=40)
# df/dt = s*f + sqrt(c/Rtot*f)
pylab.xlabel('x = 1/(z*f0)')
pylab.ylabel('y = 1/log(1/H(z))')
# Set up some figures
pylab.figure(2)
plotted_trajectory_example=False
# first trajectory with inferred fitness > 9%
pylab.xlabel('Time (generations)')
pylab.ylabel('Lineage frequency')
# Set up some figures
pylab.figure(3)
# DFE
pylab.xlabel('Fitness effect, $s$')
pylab.ylabel('DFE, $U \\rho(s)$')

# Infer kappas and mean fitnesses!
kappa_ts = []
mean_fitnesses = []
for drift_idx in range(0,len(ts)-1):
    dt = ts[drift_idx+1]-ts[drift_idx]
    winvs = []
    kappas = []
```

```
    for R0 in range(20,60):
        good_idxs = (coverage_trajectories[:,drift_idx]==R0)
        observed_coverages = (coverage_trajectories[:,drift_idx+1])[good_idxs]
        expected_coverage = R0*1.0/depths[drift_idx]*depths[drift_idx+1]
        zs = 1.0/(numpy.linspace(0.1,2)*expected_coverage)
        hs = numpy.exp(-zs[None,:]*observed_coverages[:,None]).mean(axis=0)
        ys = 1.0/numpy.log(1.0/hs)
        xs = 1.0/zs/expected_coverage
        slope,intercept,dummy,dummy2,dummy3 = linregress(xs,ys)
        if (drift_idx==0) and (R0==50):
            # Plot the generating function!
            pylab.figure(1)
            pylab.plot(xs,ys,'k.')
            pylab.plot(xs,xs*slope+intercept)
            pylab.xlim([0,2])
        winv = slope
        kappa = intercept*expected_coverage*winv
        winvs.append(winv)
        kappas.append(kappa)
        #print "kappa = ", kappa
    kappas = numpy.array(kappas)
    winvs = numpy.array(winvs)
    kappa_ts.append(kappas.mean())
    mean_fitnesses.append(numpy.log(winvs.mean())/dt)
kappa_ts = numpy.array(kappa_ts)
mean_fitnesses = numpy.array(mean_fitnesses)
# Other parameters
f0s = frequency_trajectories[:,0]
twoc = 3.5
Nb = 7e07
dt = 8
Ne = Nb*dt
dts = ts[1:]-ts[:-1]
mean_fitness_Ws = numpy.exp(-numpy.cumsum(mean_fitnesses*dts))
Ub0 = 1e-05
sb0 = 1e-1
ss = numpy.linspace(0,0.4,80)[1:]
ds = ss[1]-ss[0]
taus = numpy.arange(-250,100)*1.0
dtau = taus[1]-taus[0]

# Prior from original paper
#log_prior = (numpy.log(dtau/(taus[-1]-taus[0])))*numpy.ones_like(taus)[None,:]+(numpy.log(Ub0*ds/sb0

# Modified prior (flat prior in s, but taking account overall probability of producing a mutation)
log_prior = numpy.log(2/twoc*Ne*numpy.median(f0s)*(taus[-1]-taus[0])*Ub0*sb0)+(numpy.log(dtau/(taus[-

#
beneficial_fs = twoc/2/Ne*numpy.exp(ss[None,:,None]*ts[:,None,None]-ss[None,:,None]*taus[None,None,:]
# Don't take last timepoint
beneficial_fs = beneficial_fs[0:-1,:,:]
beneficial_fs *= mean_fitness_Ws[:,None,None]
# Now go through and infer things per site
#plotted_example
```

```
beneficial_mutation_idxs = []
beneficial_mutation_ss = []
beneficial_mutation_taus = []
import time
start_time = time.time()
#desired_idxs = numpy.arange(0,1000)
desired_idxs = numpy.arange(0,coverage_trajectories.shape[0])
for i in desired_idxs:
    freqs = frequency_trajectories[i,0:-1]
    safe_freqs = (freqs+(freqs==0))
    beneficial_subfreqs = numpy.clip(beneficial_fs/safe_freqs[:,None,None],0,1)
    # Calculate effective s as a function of tau
    effective_ss = ss[None,:,None]*beneficial_subfreqs-(mean_fitnesses)[:,None,None]
    effective_Ws = numpy.exp(effective_ss*dts[:,None,None])
    neutral_expected_reads = freqs*depths[1:]
    selected_expected_reads = neutral_expected_reads[:,None,None]*effective_Ws
    sqrt_neutral_expected_reads = numpy.sqrt(neutral_expected_reads)
    sqrt_selected_expected_reads = numpy.sqrt(selected_expected_reads)
    sqrt_observed_reads = numpy.sqrt(coverage_trajectories[i,1:]*1.0)
    log_likelihood = 1/4*numpy.log(effective_Ws).sum(axis=0)
    log_likelihood += -(numpy.square(sqrt_selected_expected_reads-sqrt_observed_reads[:,None,None])/k
    log_likelihood += +(numpy.square(sqrt_neutral_expected_reads-sqrt_observed_reads)/kappa_ts).sum()
    log_bayes_factor = log_prior + log_likelihood
    max_b = log_bayes_factor.max()
    if max_b < 0:
        continue
    max_idxs = (log_bayes_factor==max_b)
    max_ss = (ss[:,None]*numpy.ones_like(taus)[None,:])[max_idxs]
    max_taus = (taus[None,:]*numpy.ones_like(ss)[:,None])[max_idxs]
    s = max_ss[0]
    tau = max_taus[0]
    beneficial_mutation_idxs.append(i)
    beneficial_mutation_ss.append(s)
    beneficial_mutation_taus.append(tau)
    if i in [14,15]:
        print "Estimated fitness", s, "for lineage", i, "(0-based)"
    if s>0.08 and not plotted_trajectory_example:
        print("Plotting example:", i, s, tau, max_b)
        # Try to plot it
        fs = frequency_trajectories[i,:]
        ff = fs[-1]
        # build it from reverse
        reversed_fitted_fs = [ff]
        for t in reversed(range(0,len(dts))):
            f = reversed_fitted_fs[-1]*exp(mean_fitnesses[t]*dts[t]-s*dts[t])
            reversed_fitted_fs.append(f)
        fitted_fs = numpy.array(reversed_fitted_fs)[::-1]
        pylab.figure(2)
        pylab.plot(ts,fs,'k.')
        pylab.semilogy(ts,fitted_fs,'-')
        #pylab.ylim(1,1e08)
        pylab.ylim(1e-08,1)
        plotted_trajectory_example=True
print("Done!")
```

```
print("Found", len(beneficial_mutation_idxs), "beneficial mutations out of", i, "(%d total)" % covera
end_time = time.time()
print("Took", end_time-start_time, "seconds to run")
# Calculate contribution for each one:

print("Calculating DFE")
mus = []
for s in ss:
    tmaxs = numpy.log(f0s*2*Ne*s/twoc)/s
    mutation_weight = Ne*2*s/twoc*(f0s[desired_idxs,None]*(mean_fitness_Ws*dts)[None,:]*(ts[None,1:]<
    if mutation_weight == 0:
        mus.append(-1)
    else:
        num_mutations = (beneficial_mutation_ss==s).sum()
        mus.append(num_mutations/mutation_weight)
mus = numpy.array(mus)
pylab.figure(3)
pylab.plot(ss[mus>0],mus[mus>0],'k.-')
# Problem set output
pylab.figure(1)
pylab.savefig('levy_blundell_fig1.pdf',bbox_inches='tight')
pylab.figure(2)
pylab.savefig('levy_blundell_fig2.pdf',bbox_inches='tight')
pylab.figure(3)
pylab.savefig('levy_blundell_fig3.pdf',bbox_inches='tight')
```