Emulating Genetic Drift in Serial Transfers, Experiment and Theory

N.Tin, J.Asencao, J.Denk, O.Hallatschek

August 2019

1 Introduction

The Wright-Fisher model is a simple discrete-time Markov chain that represents genetic drift in a population's alleles. In the model, it is assumed that the population size N is constant, and all individuals in a population will die at the end of each generation and be replaced by its offspring, such that generations can be viewed as discrete time steps. Because reproduction is a random process, some individuals' $2N$ copies of their genome will be passed on, where the remaining few may end their lineage (genetic drift). If i is the number of alleles and have a frequency of $p = m/2N$, then each allele in the continuing generation is selected with that same probability, resembling a Poisson distribution.

Serial transfer experiments have been popularized by Lenski's famous LTEE experiment, pioneering a new generation in population dynamics (1). This form of experimentation allows researchers to explore time, or as Gould put it, 'replay life's tape'. Propagating numerous replicates of the same ancestor allows scientists to understand dynamics of adaptation at a large scale. In addition, researchers may change the parameters to fit their needs; population size, fitness differences, etc. Transfer experiments resemble serial dilutions; a given time allows cell growth before being transferred to new media to repeat the same pattern. The process of sampling to inoculate into new media, however, creates a bottleneck. Traditionally, the Wright-Fisher model would be applied to understand the dynamics of growth, but new research indicates a brief and variable phase in which cells must transition from stationary to exponential phase.

In this report, we detail our inquiry into whether serial dilutions could be truly represented by the Wright-Fisher model, or if other mechanisms, such as lag time, need to be incorporated. We recreated the conditions of a serial transfer experiment both in the lab and through Python. Neutral mutants, identifiable by Kanamycin resistance, would assist in identifying frequencies within such a sub-population. Comparison between these experimental and theoretical results and identifying discrepancies lend itself to identifying new mechanisms in genetic drift.

2 Experimental Framework

Serial transfer experiments typically involve a uniform population that is divided among a number of subsets such that each subset is a replicate of the original. The subset is regularly and randomly sampled from and inoculated into new media. After each transfer, the population is sampled and plated on LB/Kan to determine the number of mutants. If the candidate had a relatively stable CFU and balanced count, that would indicate that the mutation was had neither a beneficial or deleterious effect. When the mutant is *known* to be neutral, then the fluctuations are able to be quantified, especially if it is subject to stochastic extremes.

In the setup for the experiment, we isolated four colonies from a Kanamycin-resistant Rel606 Transposon Library and staged them against their REL606 wild-type. For seven days, cultures would be grown, later mixed, plated, and stored in a 37C incubator. Because of unforeseen issues with some lab equipment, this process was executed multiple times to ensure the validity of the results. In this phase, we were interested in finding a neutral mutant such that it would be almost the same as its wild-type. When such a mutant was chosen, this process was again repeated, but with the intent of tracking the quantitative changes a population is subject to throughout its growth.

3 Experimental Results

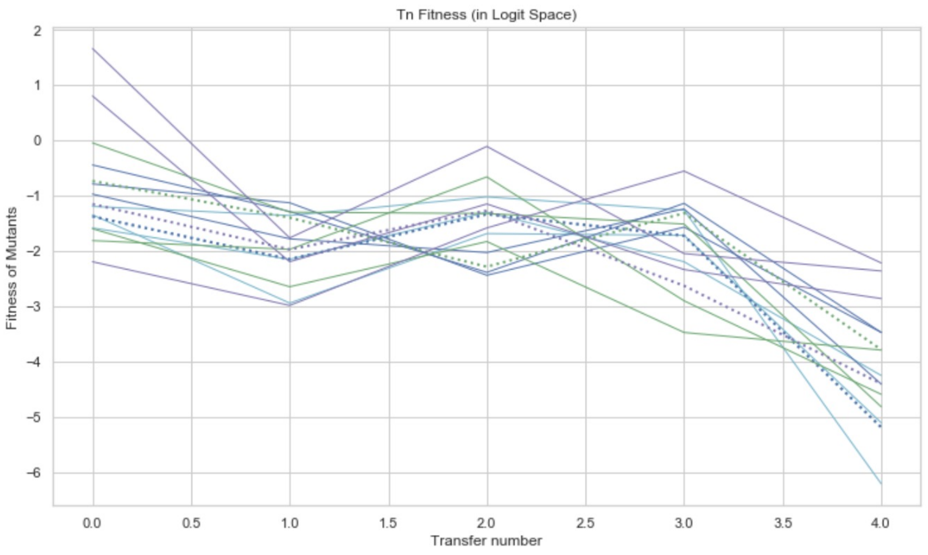


Figure 2. Candidate Fitness Differences

DM25 supports $5 * 10^7$ cells per mL in stationary phase, and we propagated with 10uL of the previous culture, or $5 * 10^5$ cells at bottleneck (2). While plating, we plated the previous growth after a 10^{-5} dilution, so we would expect to see approximately 250/500 cells on the plate. Three replicates of each clone were propagated, with unique clones being represented by a unique colored line. Dotted lines represent the mean of each replicate. The chart in logit space (Fig

2) represents the fitness of the clone relative to the wild-type, where a fitness difference of zero indicates a perfectly neutral mutant. Fitness seemed to fluctuate greatly between -1 and -2, where the purple line is the only to approach zero (at transfer 2), but is the most erratic. For the body of the experiment, we selected the mutant on the basis of its stability and neutrality (Green Clone 3, see Fig 2, 1 in *Supplementary Figures*. The primary experiment would be set up in the same manner as the first phase, with the exception of there being 12 replicates of one lineage, instead of three per each four lineages. Each was propagated for five days in 1mL DM25 and plated on LB/Kan. From this, we could see the fluctuations in growth within a uniform population.

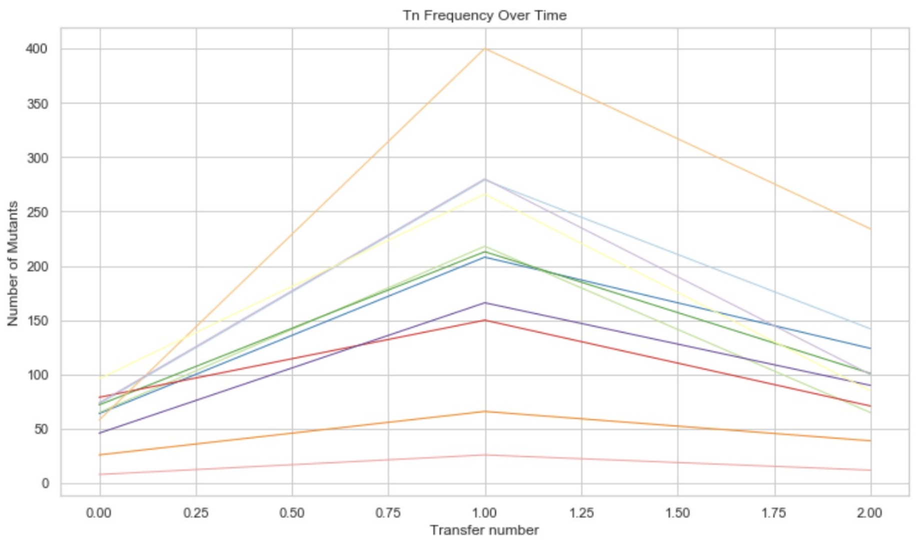


Figure 3. Number of Mutants per Replicate over Time

The experiment faced some storage difficulties, such that there were only three of five days of data harvest. From those three days, there was clear evidence of some high fluctuations in the mutant population. One lineage, represented in yellow, saw a jump in population from 59 to 400. With the dilution factors, we should expect a total of 500 cells (wild type and mutant) in each plate, so it is questionable whether one neutral mutant came to almost fix in the population after one transfer. The uniform rise in population may be due to a pipetting error, which further begs the need for more data.

4 Written Simulation

The simulation attempts, as much as possible, to rely on the basic assumptions made when conducting a serial transfer experiment.
A grandfather population is first initialized, and "*barcodes*" are assigned to each individual so that each may be tracked over time (This method informs frequency not simply quantitatively, but if one mutant proliferated to become the whole population, or if there is coexistence). Given that the population is well mixed, replicates may be sampled to propagate in its own lineage. In this step, we begin to count the number of *mutants* as well as the transfer number.

In the lab, we would then let the cultures grow up to a certain size (determined by a length of time) before sampling it to inoculate new media, and repeat the process. The simulation "tiles" the barcoded cells up to size N before sampling, so the sampling proportions is the same as the proportion of each barcode.

```
def wfisher(N, nflasks, pmutants, gens, dilfactor):
    cutoff = int(N*pmutants)
    pick = int(N*dilfactor)

    IDs = np.random.choice(np.arange(N*1.2), N, replace = False)
```

Our function *wfisher* is parameterized by N fixed number of cells, $nflasks$ number of replicates, *pmutants* decimal percent of mutant/second allele likelihood, *gens* number of transfers, and $dilfactor$ which represents the bottlenecking at each transfer. From this we can calculate a way to identify mutants and the number of cells randomly chosen to contribute to the next generation. Though the simulation was originally created to track the progress of each individual cell, we primarily track the number of mutants in each replicate as time progresses. To track the progress of each lineage, we randomly assigned a number from 1 to $N * 1.2$, where numbers less than the cutoff would be identified as mutants. *mutants* as an array of dimensions (1, $nflasks$) will be continuously used to store the number of mutants for each replicate.

```
    mutants = np.array([])

    dils = np.random.choice(IDs, pick)
    count = len(find_mutants(dils, cutoff))
    for i in np.arange(nflasks-1):
        select = np.random.choice(IDs, pick)
        dils = np.vstack([dils, select])
        count = np.append(count,
    len(find_mutants(select, cutoff)))
    all = np.array([dils])
    mutants = np.append(mutants, count)
```

The next section of the simulation draws the first bottleneck from the intial population and stores it in *dils* and does the same for $nflasks$ number of replicates. *count* takes the recently drawn population and counts the number of cells numbered less than the cutoff (or the number of mutants). This process iterates for every flask and is stored in the initial array. The array *dils* catalogues every numbered cell in a replicate to now be stored in a tensor *all*.

```
    dils = np.random.choice(IDs, pick)
    for i in np.arange(gens):
        lastdil = np.tile(all[-1], int(N/pick))
        gendil = np.random.choice(lastdil[0], pick)
        genmutants = len(find_mutants(gendil, cutoff))
        for j in np.arange(nflasks-1):
            btlneck = np.random.choice(lastdil[j+1], pick)
            gendil = np.vstack([gendil, btlneck])
            genmutants=np.append(genmutants,
    len(find_mutants(btlneck, cutoff)))
```

```
        all = np.vstack([all,[gendil]])
        mutants = np.vstack([mutants, genmutants])

return N, nflasks, pmutants, gens, dilfactor, mutants
```

In this last section, which loops to fill each of the last generations, unilaterally "*tiles*", or scales, each replicate to size N such that the probability of drawing a mutant reflects its proportions in the population. Then, *pick* number cells are drawn randomly to again simulate the bottleneck. Mutants are counted and stored in mutants again in as a new array. The process repeats for *gens* transfers. *wfisher* returns the input parameters and the resulting array of mutants such that the data may be analysed.

5 Analysis

To match the parameters used in the lab experiment, our parameters were as follow:

$N = 10^5$
$nflasks = 15$
$pmutants = 0.5$
$gens = 500$
$dilfactor = 0.01$

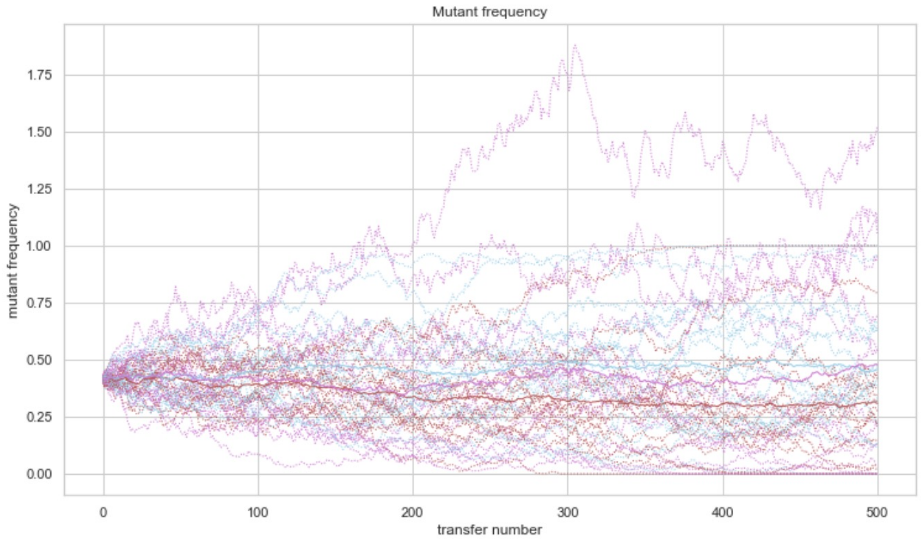


Figure 5. Simulated, Binomial, and Poisson Mutant Frequency over Time

These results were plotted in red against Wright-Fisher simulations using binomial sampling (in blue) and Poisson (in purple). The solid lines running through the center are the mean frequencies of a model's replicates. Each run of the simulation, especially given the limited number of replicates, prominently showed the great fluctuation in population over time. Many, if not all trajectories have a number of significant drops or rises in population, at no specific time point.

Due in part to the large percentage of mutants, scenarios of their extinction and fixation were equally probable. The simulation was initially built to analyze mutants in a smaller population, and in such the extinction times were not as telling. Again, the high number of neutral mutants gave rise to much longer lasting lineages, a majority of which exceeded 500 transfers. As it stood from averaging the extinction times from ten runs with 15 replicates, the extinction time/transfer for the simulation was 486.5, binomial model was 490, and 465.75 for Poisson. To convert this to reach generation times, we use $gens = log2(dilfactor) * transfer$ (this yields negative numbers, so I took the absolute value; 3229 generations for the simulation, 3255.5 for binomial, and 3089 generations with Poisson. (Note that there is an unresolved error which allows some Poisson replicates to reach a frequency greater than 1. The figure showing the negative binomial distribution is attached at the end.)

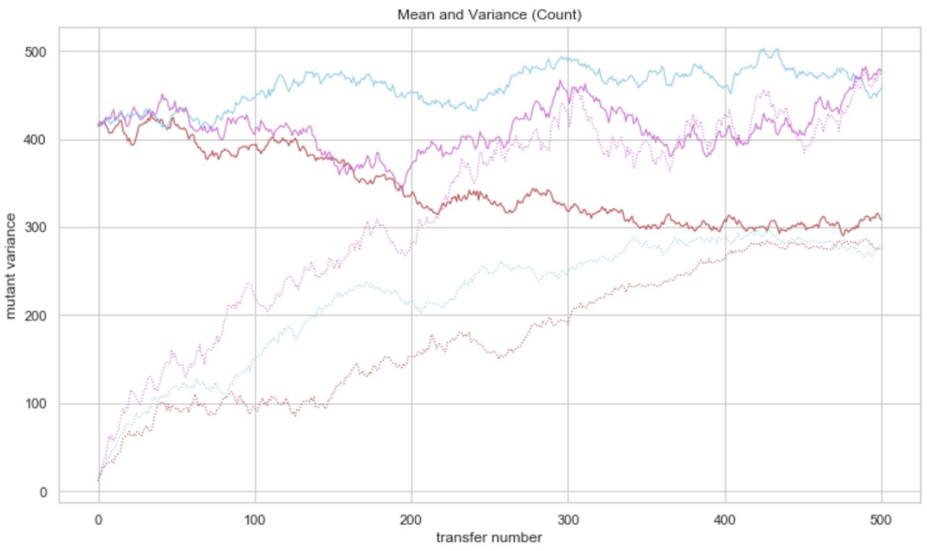


Figure 6. Mean and Variance of Mutant Frequency

The mean frequencies, represented by a bold line, were mildly more erratic than expected. Because populations rise and decline almost equally, we expect to see a mean that is relatively unchanging; however for the simulation, the mean fell 100 mutants and rose almost 100 for binomial. It is notable that, as we expect, the mean is not subject to the same sharp changes in frequency. This follows the assumption described, leading me to make the assumption that declines or increases are due to chance in drawing a mere 15 replicates. This lends greater faith to the Wright-Fisher code that we wrote. However, it is difficult to immediately discern if there is a significant difference in the behaviours of the different models.

It is also difficult to relate the experimental data to the theoretical sets given the small batch, however it is promising to see the similarities in the great fluctuations between replicates. Functions controlling the seed between the different distributions, visualizing changes in N, creating a probability density function, and other data analysis methods were later omitted, but hopefully could be reincorporated with greater functionality later.

6 More Data

In hopes to better see genetic drift in effect, we changed the percentage of mutants in the population such that $pmutants = 0.01$ rather than the experimental condition 0.5. This allows us to play with scenarios in which new mutations arise, rather than simply see the fluctuations in 0.5 of a population. We also increased $nflasks = 50$ for increased replicates, and $gen = 250$ because lineages, on average, had smaller extinction times. These were the original numbers for the simulation, and yielded results more reflective of mutations that arise randomly in a population.

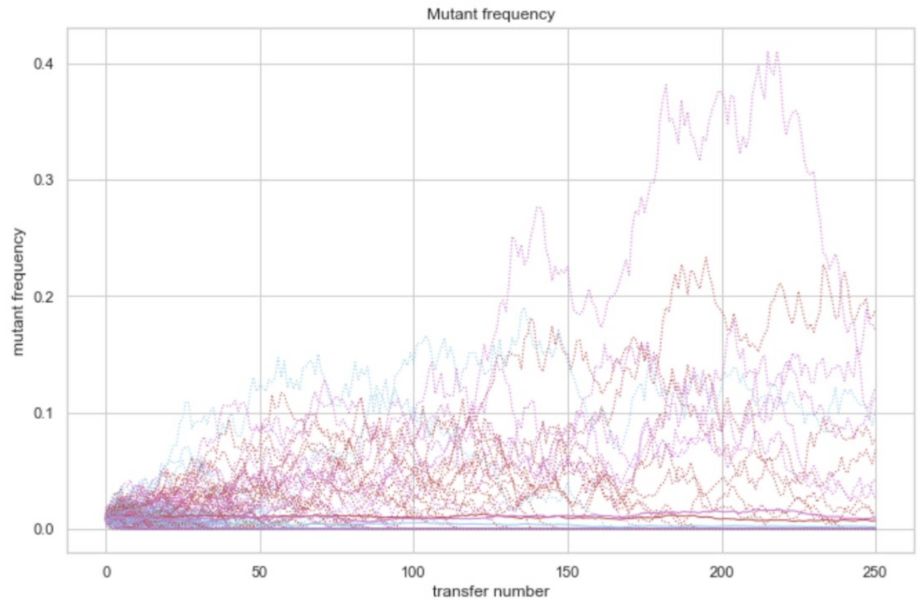


Figure 7. Simulated, Binomial, and Poisson Mutant Frequency over Time

In these simulations, it is clear that a majority of the lineages have an extinction time less than 100 transfers, but often one or two replicates the propagate further. From this we can tell its not *unlikely* for mutants to rise in the population. In this run of the simulation, the extinction time was 51, 51, 47, for the simulation, binomial, and Poisson distributions, respectively.

More runs of this simulation are in Supplementary Figures (Figs 9, 10, 11). From the statistics the function returns now, it is not clear if any one model consistently over or underestimates compared to the other models. Mean extinction times all hover around 50-60 transfers, but a future step will be creating a histogram for these dates.

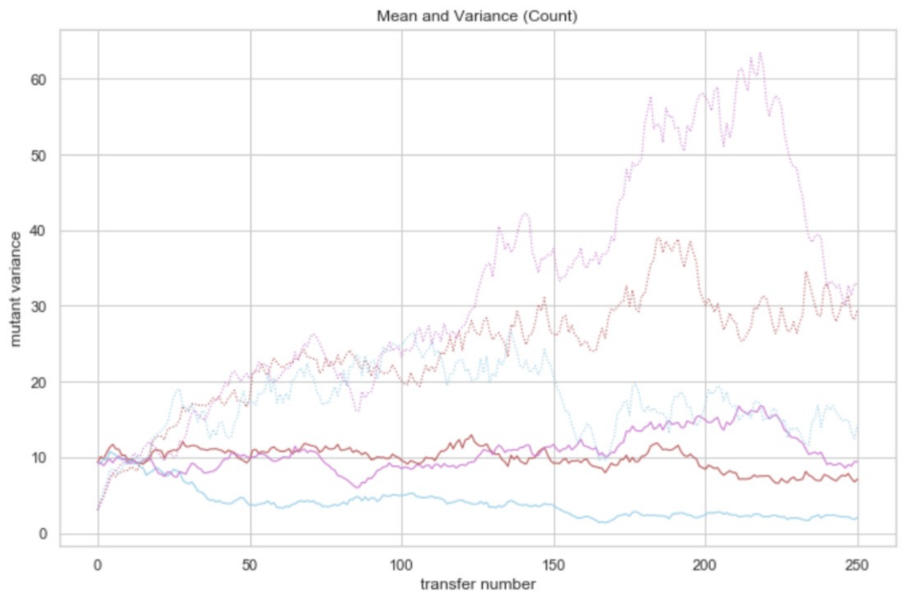


Figure 8. Simulated, Binomial, and Poisson Mutant Frequency over Time

The mean and variance of the data set behave closer to how we would expect; the mean is relatively more constant, though the binomial mean falls significantly below. The variance increases over time as lineages diverge.

7 Future Measures

The experimental side of the project faced difficulties pertinent to timing in both length of the experiment and nearing school starting. Ideally, the experiment would run longer than five days, which can easily be restarted come the beginning of the school year. Additionally, it seems that the spinning 37C incubator was not an ideal setting for the mutants, as there were difficulties managing the shared space (resulting in loss of data). The data, from the three days recorded, seemed to unilaterally rise and consequently fall across wells. This could be due in part to pipetting error in propagating the second day, where maybe the population was over-sampled, and fixed the next day.

Much of the time was invested in creating different methods to approach the simulation, creating and removing functions, and ensuring perfect conditions for the experiment. Once the academic year settles in, we are looking to finally incorporating lag times, experimental fitness differences, and other complexities, which I am excited about! Running the experiment again, I would hope to replicate the experiment we set up for a longer duration, and possibly concurrently run with a greater diversity of parameters to compare this data to the simulation.

References

[1] http://myxo.css.msu.edu/ecoli/

[2] http://myxo.css.msu.edu/ecoli/dm25liquid.html

[3] https://docs.google.com/document/d/1IRSOiwDaFZgX3l99jWYsdb9xjpMFmbhdV5bhThwwBYk/edit?

8 Supplementary Figures

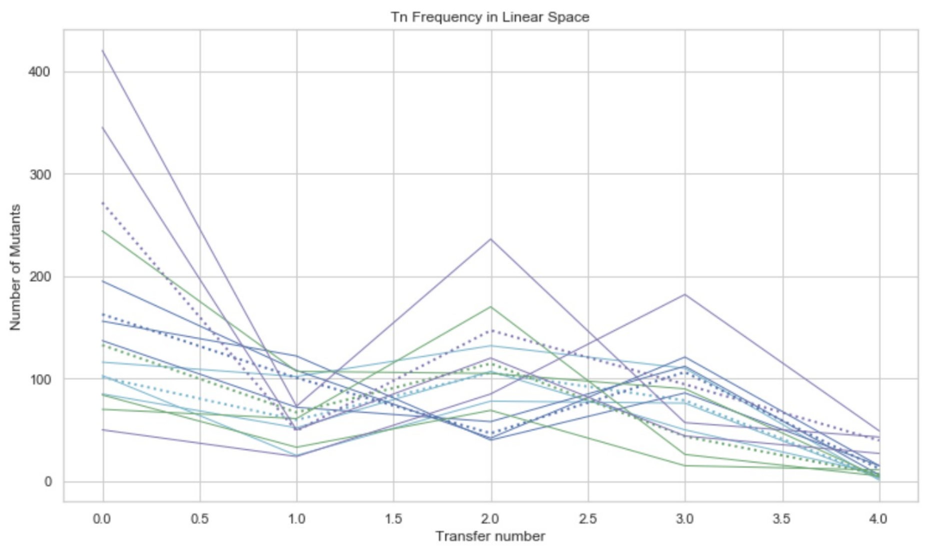


Figure 1. Linear Representation of Neutral Clone Candidates

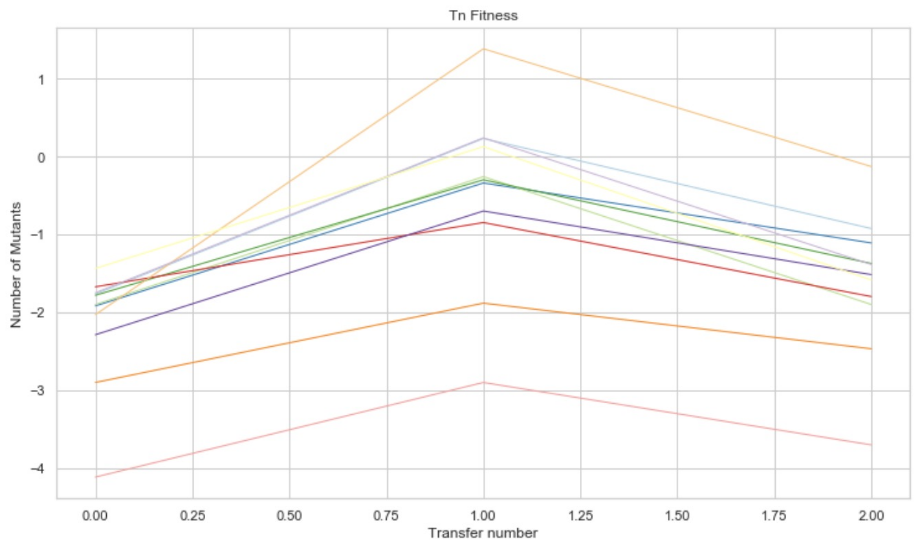


Figure 4. Fitness of Each Replicate over Time

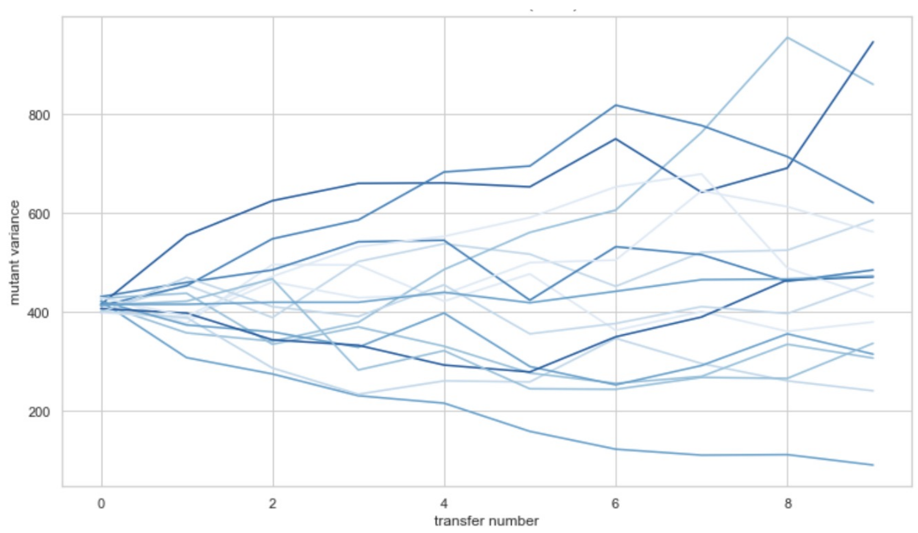
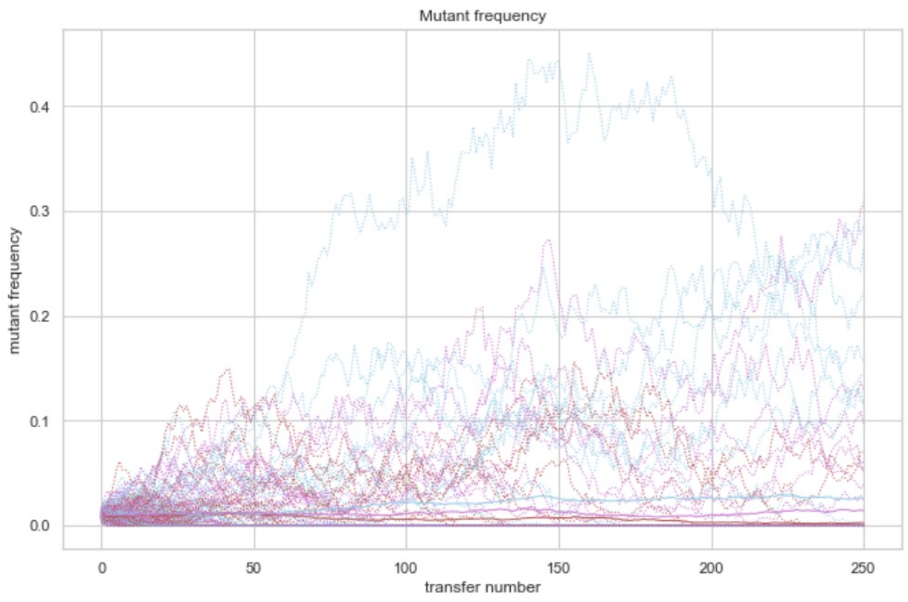


Figure 5b. Trajectory Using a Negative Binomial Distribution

```
mean extinction time for simulated values: 49.94 binomial dist: 67.2
poisson dist: 63.94
```



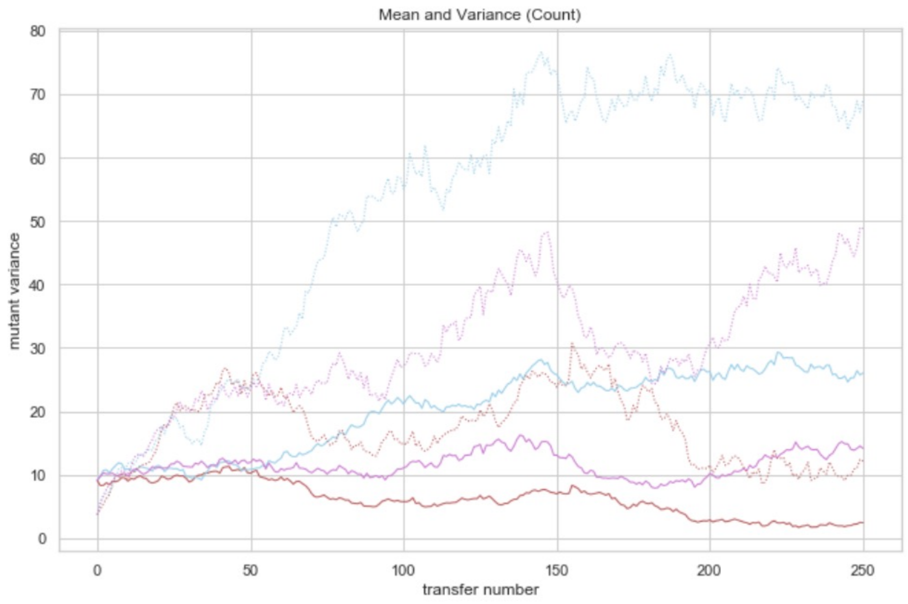
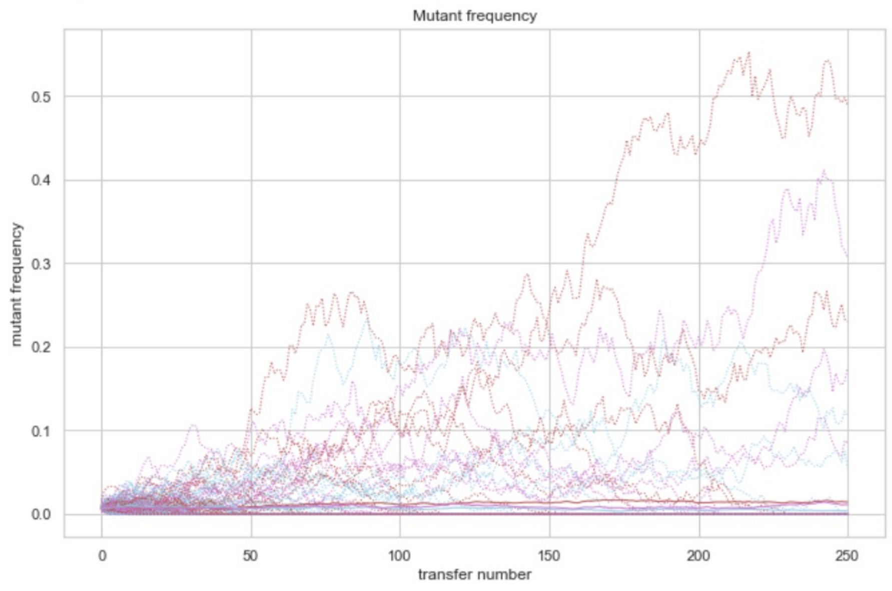


Figure 9. More Simulated Results with Written, Binomial, and Poisson

Written Simulation is represented in Red, Binomial in Blue, and Poisson in Purple. Mean extinctions are on the top, followed by trajectories and their means and variance.

```
mean extinction time for simulated values: 52.58 binomial dist: 4
9.16 poisson dist: 46.38
```



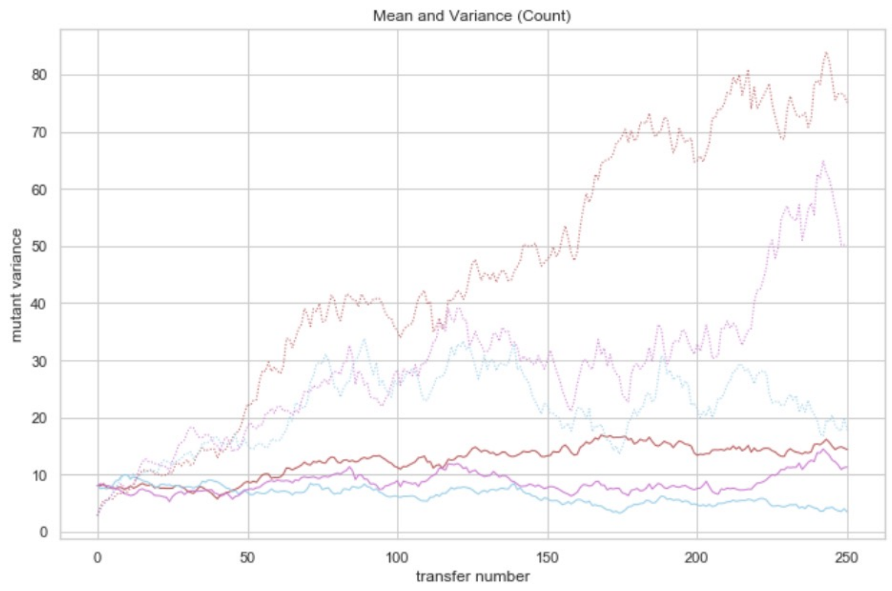
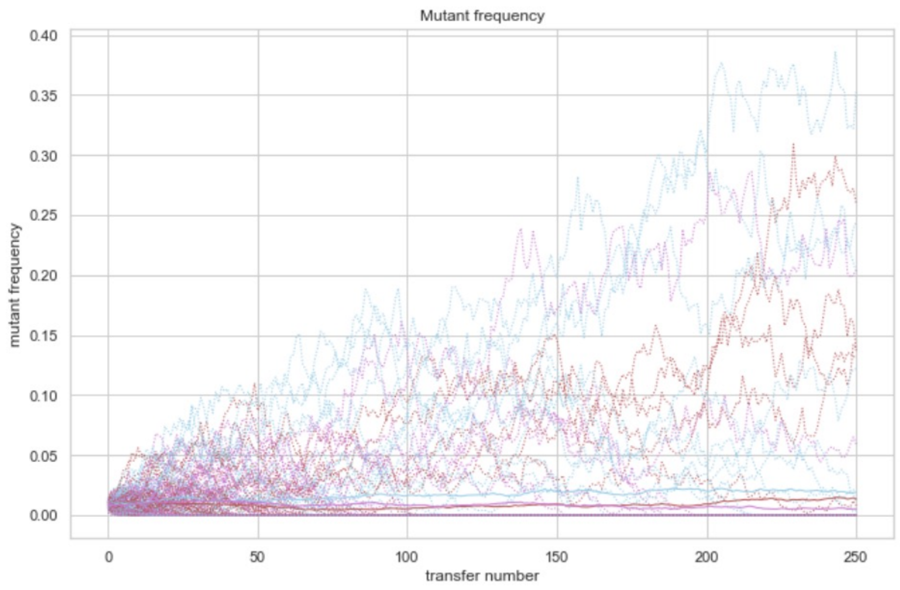


Figure 10. More Simulated Results

```
mean extinction time for simulated values: 54.14 binomial dist: 6
9.94 poisson dist: 55.68
```



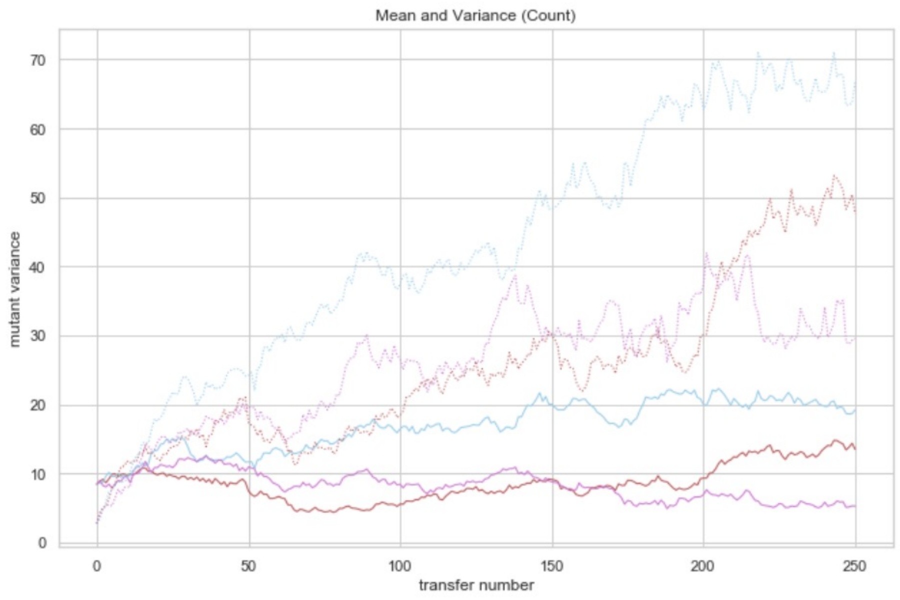


Figure 11. More Simulated Results

9 Code for Data Analysis

```
def STATS(N, nflasks, pmutants, gens, dilfactor, mutants, binom, poi):
    # EXTINCTION TIMES
    mext= np.array([])
    bext= np.array([])
    pext= np.array([])
    N, nflasks, pmutants, gens, dilfactor, mutants, binom, poi =
        wfisher(N, nflasks, pmutants, gens, dilfactor)
    for j in range(nflasks):
        mext = np.append(mext, np.count_nonzero(mutants[:,j]))
        bext = np.append(bext, np.count_nonzero(binom[:,j]))
        pext = np.append(pext, np.count_nonzero(poi[:,j]))
    print('mean extinction time for simulated
        values:',np.mean(mext),'binomial dist:',np.mean(bext),'poisson
        dist:', np.mean(pext))

    # VISUALIZE DATA
    pick = N*dilfactor
    fig = plt.figure(figsize=(11,7)) #initialize a figure
    fig.subplots_adjust(wspace=.7,hspace=0.4)
    plt.subplot(111)
    xrange = range(gens+1)
    for j in range(nflasks):
        plt.plot(xrange, mutants[:,j]/pick, ':r', linewidth=1)
        plt.plot(xrange, binom[:,j]/pick, ':', c='skyblue', linewidth=1)
        plt.plot(xrange, poi[:,j]/pick, ':', c='orchid', linewidth=1)
    #plot MEAN f
    plt.plot(xrange, np.mean(mutants/pick,axis=1), '-r', linewidth=1)
    plt.plot(xrange, np.mean(binom/pick,axis=1), c='skyblue',
        linewidth=1)
    plt.plot(xrange, np.mean(poi/pick,axis=1), 'orchid', linewidth=1)
    plt.title('Mutant frequency')
    plt.xlabel('transfer number')
    plt.ylabel('mutant frequency')
    plt.show()

    # plot MEAN and VARIANCE n
    # mean should be constant and variance changing
    fig = plt.figure(figsize=(11,7)) #initialize a figure
    fig.subplots_adjust(wspace=.7,hspace=0.4)
    plt.subplot(111)
    xrange = range(gens+1)
    plt.plot(xrange, np.mean(mutants,axis=1), '-r', linewidth=1)
    plt.plot(xrange, np.mean(binom,axis=1), 'skyblue', linewidth=1)
    plt.plot(xrange, np.mean(poi,axis=1), 'orchid', linewidth=1)

    plt.plot(xrange, np.sqrt(np.var(mutants,axis=1)), ':r', linewidth=1)
    plt.plot(xrange, np.sqrt(np.var(binom,axis=1)), ':', c='skyblue',
        linewidth=1)
    plt.plot(xrange, np.sqrt(np.var(poi,axis=1)),':', c= 'orchid',
        linewidth=1)
    plt.title('Mean and Variance (Count)')
    plt.xlabel('transfer number')
    plt.ylabel('mutant variance')
    plt.show()

    # DRAW WITH NEGATIVE BINOMIAL
    t=10 #time points
    r=nflasks #replicates
    phi=50 #dispersion factor
    sim=np.zeros((t,r))
    sim[0,:]= mutants[0]

    for j in range(1,t):
        p=phi/(phi+sim[j-1,:])
        sim[j,:]=np.random.negative_binomial(phi,p)

    fig = plt.figure(figsize=(11,7))
    fig.subplots_adjust(wspace=.7,hspace=0.4)
    plt.subplot(111)
    for i in range(r):
        plt.plot(range(t), sim[:,i])
    plt.plot(range(t), np.mean(sim, axis=1))
    plt.title('Trajectory of Mutants using a Negative Binomial Model')
    plt.xlabel('transfer number')
    plt.ylabel('mutant variance')
    plt.show()


# NOT INCLUDED IN REPORT
def increaseN(nflasks, pmutants, gens, dilfactor):
    tens = [30, 75, 150, 300, 600, 1200, 2400, 5000]
    pick = []
    for ten in tens:
        picked = ten*dilfactor
        pick.append(picked)

    #plot each loop
    fig = plt.figure(figsize=(11,7)) #initialize a figure
    fig.subplots_adjust(wspace=.7,hspace=0.4)
    plt.subplot(111)
    xrange = range(gens+1)
    for n in tens:
        Nn, nf, pm, gen, dilf, m, b, p = wfisher(n, nflasks, pmutants,
             gens, dilfactor)
        for j in range(nflasks):
            plt.plot(xrange, m[:,j], ':r', linewidth=1)
        plt.plot(xrange, np.mean(m,axis=1), '-r', linewidth=1)
    plt.show()

# increaseN(nflasks, pmutants, gens, dilfactor)
```