Noah M. Jorgenson

Bioinformatics

Lab 4

A1. The probability vector after 15 generations turned out to look like this:

*[[ 0.23246634 0.26753366 0.25 0.25 ]].T*

This suggests that the probability for the first index increased, while the second decreased. The other two remained the same. It appears that the values appear to be approaching 0.25 in all the indeces of the probability vector. This is most likely because our original Jukes matrix was consistent and summed to 1.

A2.

For the mutation rates a=1e-1, 1e-2, 1e-5, and 1e-6 the generations to converge and resulting probability vectors were:

Resultant probability vector(transposed) [Generation 108]:

[[ 0.24999997]

[ 0.25000003]

[ 0.25 ]

[ 0.25 ]]

Resultant probability vector(transposed) [Generation 962]:

[[ 0.24999963]

[ 0.25000037]

[ 0.25 ]

[ 0.25 ]]

Resultant probability vector(transposed) [Generation 449358]:

[[ 0.24962501]

[ 0.25037499]

[ 0.25 ]

[ 0.25 ]]

Resultant probability vector(transposed) [Generation 2766659]:

[[ 0.24625001]

[ 0.25374999]

[ 0.25 ]

[ 0.25 ]]

As the mutation rate decreases the number of generations to reach convergence in the probability vector the nucleotides increases drastically. It makes sense biologically that when you are less likely to deviate from this initial probability vector (via mutation) that you will take more generations to mutate into a converged probability vector. This is why normal evolutionary models include crossover and selection to help influence diversity in its population’s children.

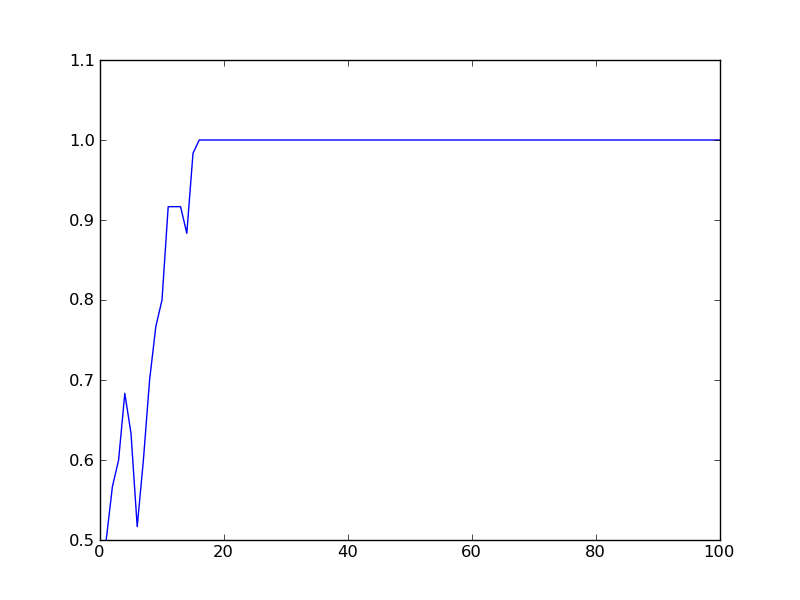
A3.

The eigenvalue-eigenvector pair that we are interested in is the eigenvalue with lamba equal to 1, which corresponds to the vector will all .5 values (which should be .25). This basically shows that our brute force way of getting the resulting probability vector (after a certain number of generations) is the same as the resulting vector calculated by linear algebra. Which is expected as proven by the Perron-Frobeni Theorem. So basically, it’s just confirming what we just calculated above in our iterative solution via brute force.

B.

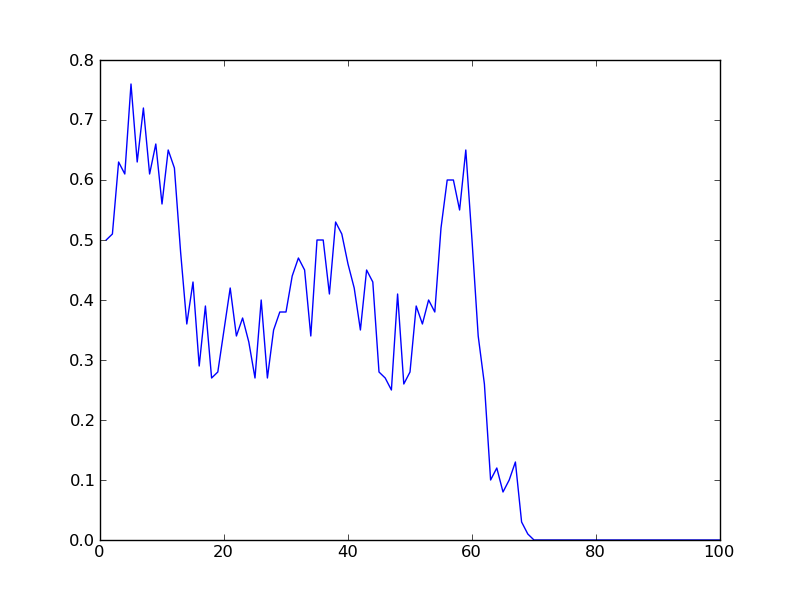
The population tends to randomly converge on either 1s or zeros, really it’s just chance as to whether which one randomly gets picked more as it becomes more and more likely to pick them in subsequent generations as the proportion goes up. Small population sizes tend to converge more quickly while larger populations tend to take longer, this is expected. Really though with a random selection function we are getting results which are not very consistent.

# A=30, N=60, G=100



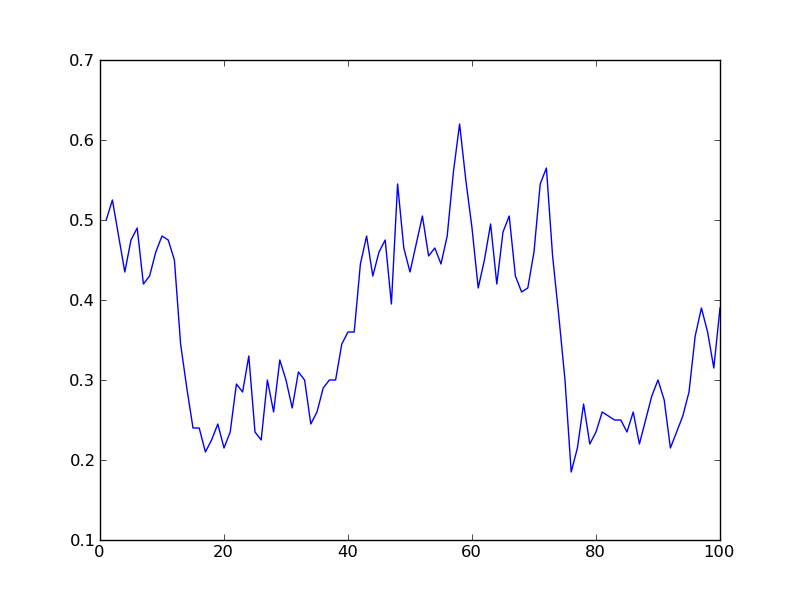
Population of 60, converges around generation 20.

# A=50, N=100, G=100



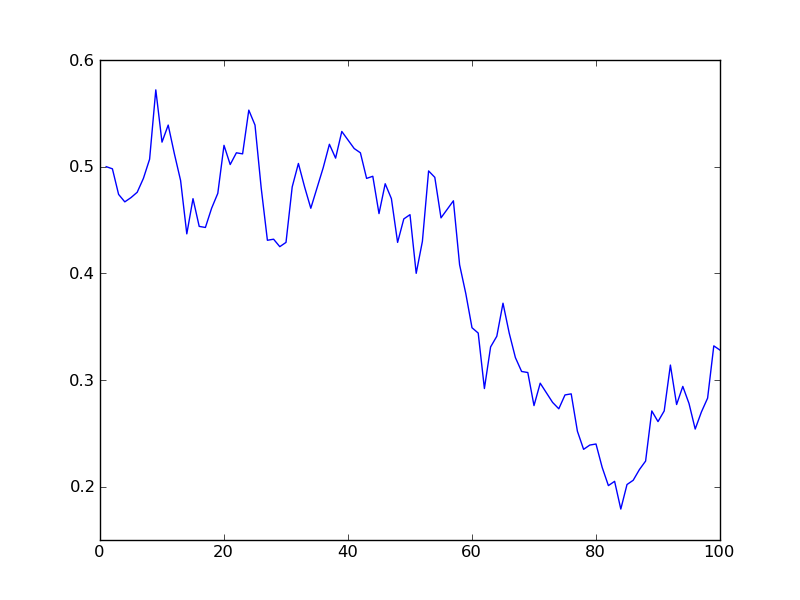
Populatoin of 100, converages around generation 70.

# A=100, N=200, G=100



Population of 200 so large it does not converge at all.

# A=500, N=1000, G=100



Once again, population of 1000, does not converge at all, too large.

If we adjust the initial proportion of A alleles we ‘tip the scales’ in favor of a particular outcome of the generations convergent population. If we increase the proportion of A to say 75% the population is much more likely to converge at 1.0 instead of 0 and perhaps converge for large population which would be less able to converge at a more equal proportion. The same holds true for if the proportion of A is smaller at say 25%.

The reason we converge towards larger amounts of 1 or 0 (A or a) in the allele is because of the Founder effect, since we are randomly selecting from the population ot create the next population, then we can sometimes pick more 1s or 0s as I mentioned earlier, and so the resulting population is predisposed to be made up of the same or more proportion of 1s thanks to the random selection. In this sense we are bottle necking out model by the random selection method, all our results are dependent upon it.