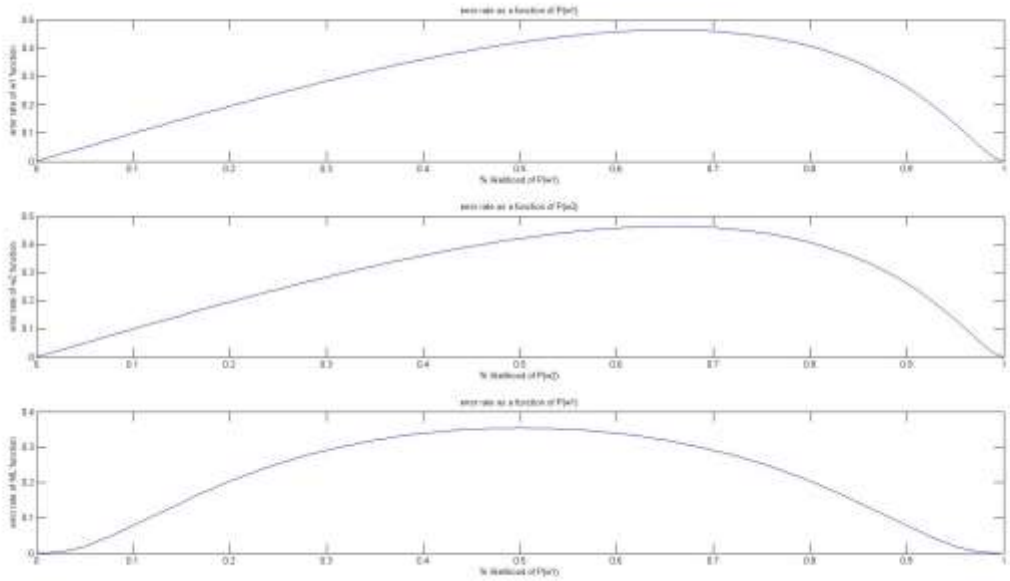
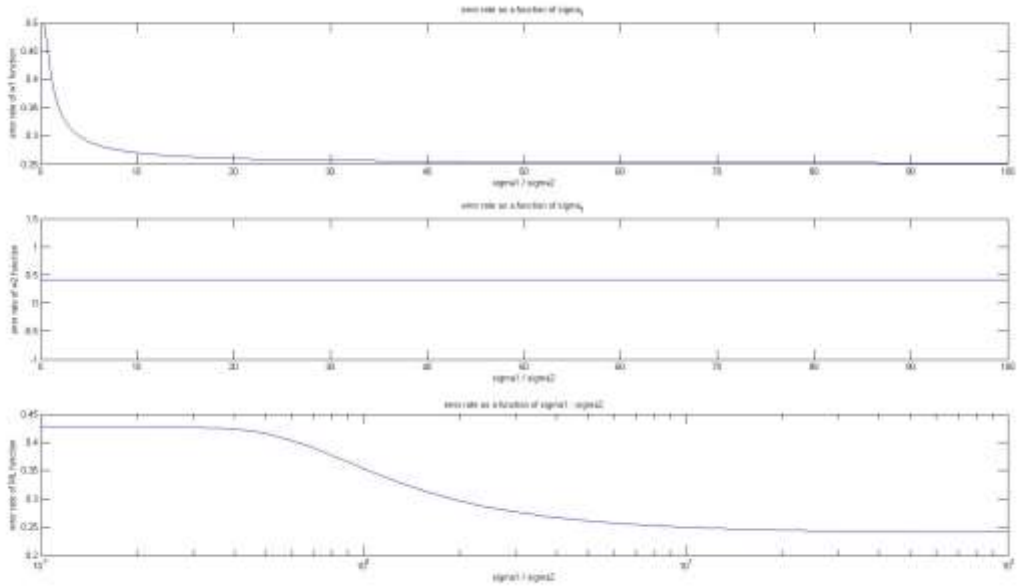
ECE 8110 Homework 2

Question 1:
Given GRVs of μ = [1 1] and μ = [-1 -1], show how variations in the prior probabilities and covariance matrices alter the probability of error for a ML classifier.

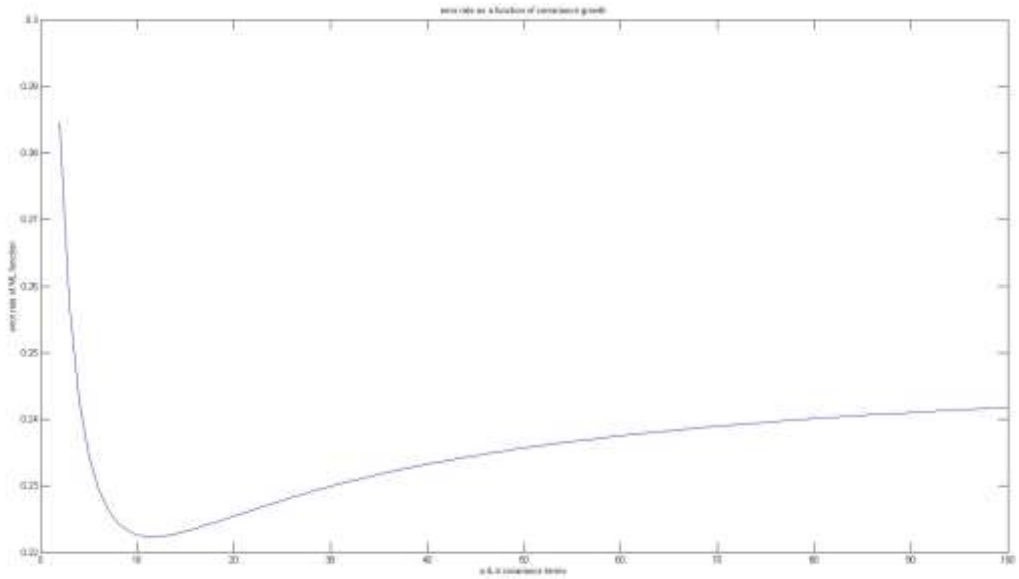


With both covariance matrices set for [1 0; 0 1] the algorithm performs best when the odds of either function are between 33% and 66%. The error for each individual function looks to peak near 65% likelihood because shortly after that the competing function's likelihood is already trailing and soon approaches zero. This is why the overall error of the function approaches zero as the prior likelihood approaches both 0 and 1.

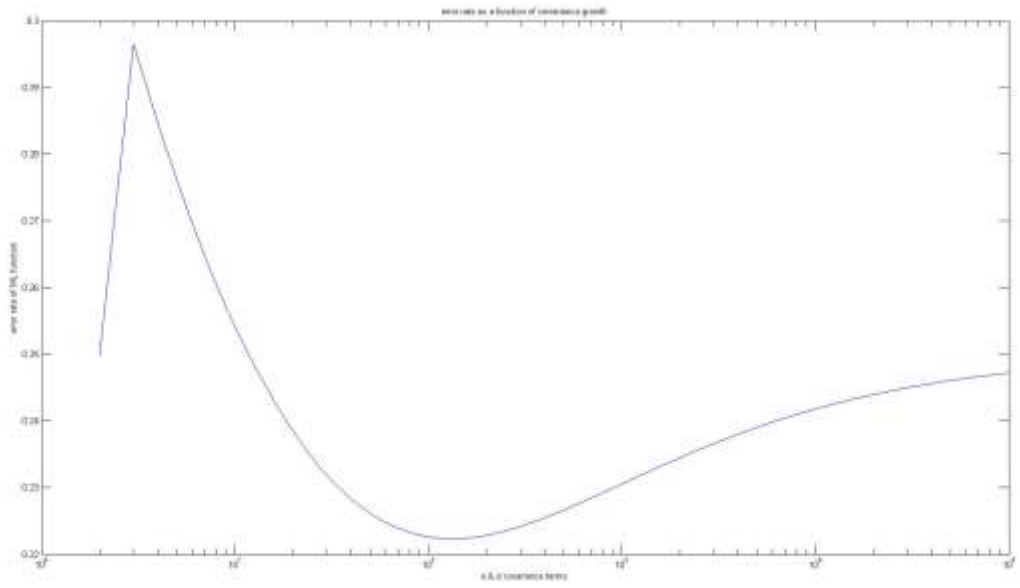
Another degree of freedom lies in the sigma of the given functions being evaluated. Since the case of changing priors has been explored, the priors will now be kept equal for future testing. Instead, the value of sigma_1 will be shifted to discern the theoretical repercussions on the error rates. Driving sigma_1 from 0.1 to 100 provides a large span to capture the shift, but is most noticeable when plotted logarithmically against the varying sigma. As the ratio between the two sigmas grows, the error rate lists towards zero as one function begins to overpower the other.



Maintaining one covariance matrix as [1 0; 0 1], the other matrix was manipulated to vary [a 2; 1 d] to measure impact on error rates. In this instance the classifier is updating on each iteration based upon changes in the second covariance matrix. The error naturally grows as the second covariance matrix increases the dispersion of the data.



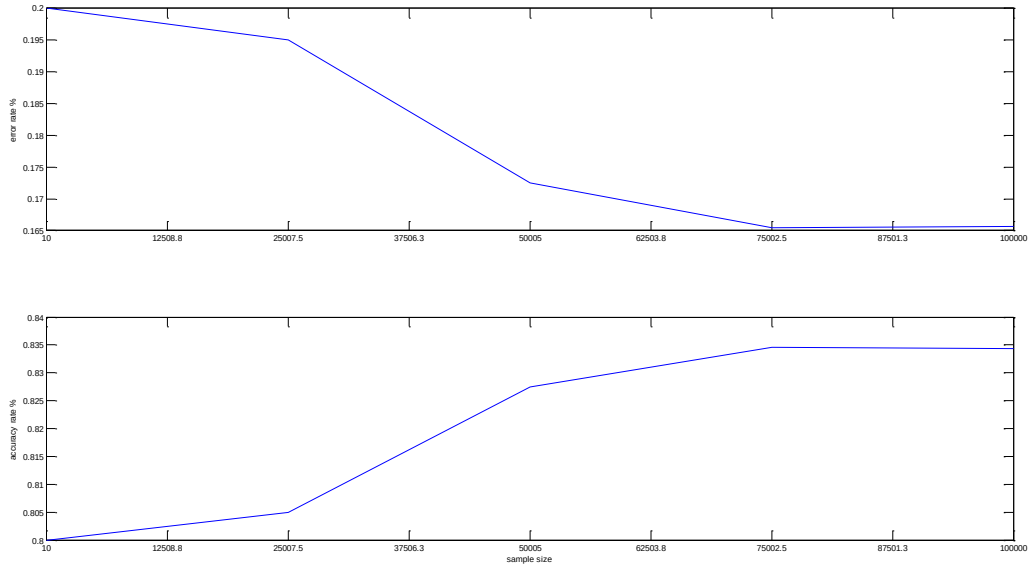
Adjusting only one of the diagonals of the covariance matrix results in a similar plot to that seen above. In both cases the error rate asymptotically approaches 25%. Even as the growing function takes over more area, it is unable to overcome the error associated with the more dense second distribution.



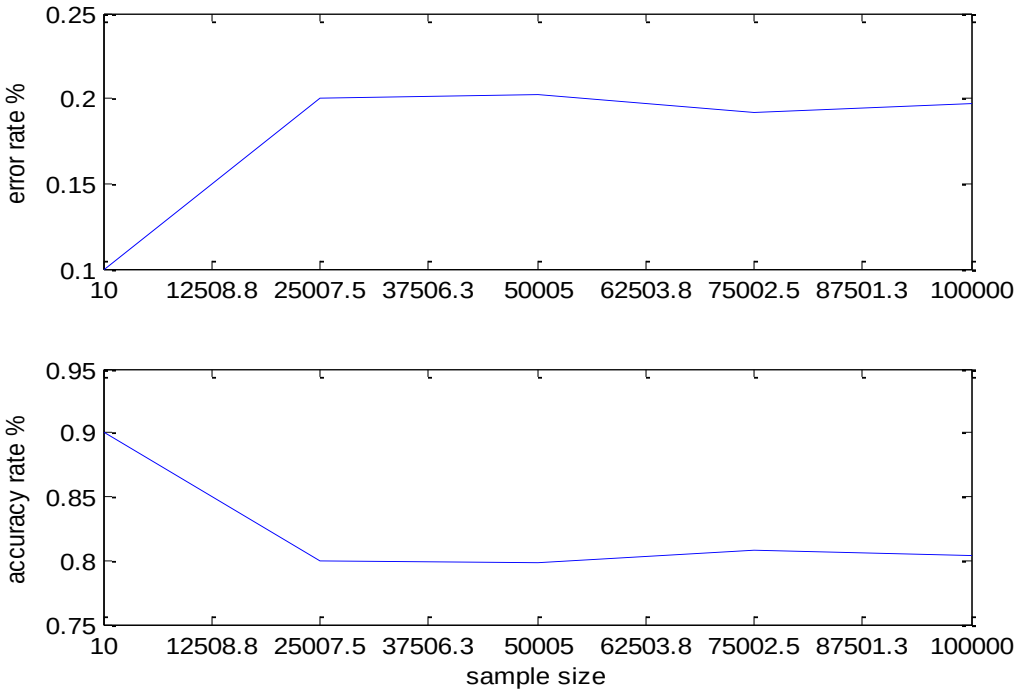
Question 2:

To compare to the theoretical limits seen in Question 1, the data sets are generated with mvnrnd in Matlab, but then the generated data is mined to find mu and sigma for each instance. Those calculated values are used for all equations to determine the analytical results of the error rates.

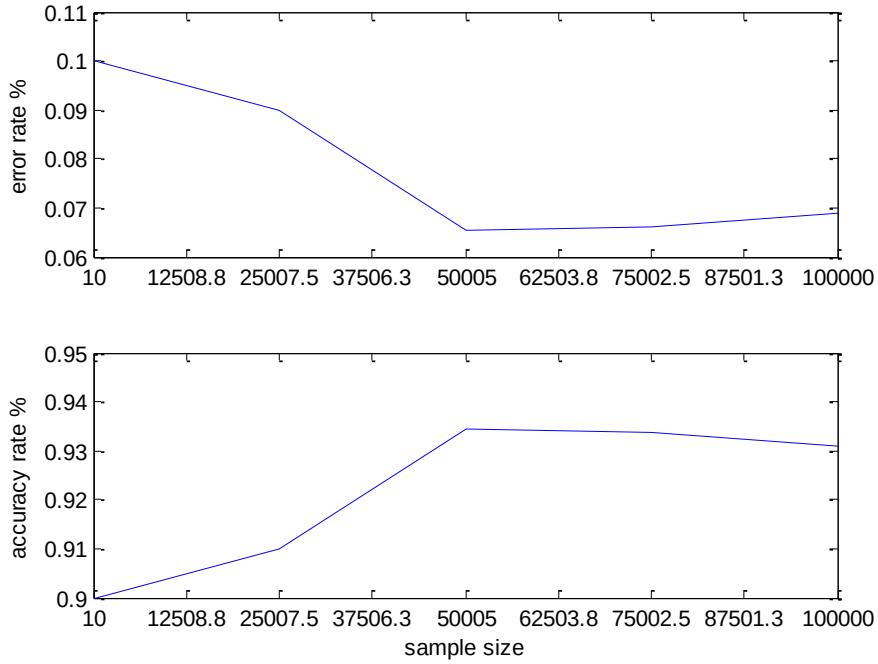
Keeping the prior probabilities equal, both set at 0.50, a variety of covariance matrices are tested to provide error rates as the number of variables in each Gaussian are increased. The constant covariance matrix is set as [1 0; 0 1] while the test covariance matrix is set at [2 1.9; 1.9 2] to provide insight into error rates as a matrix becomes less linearly dependent. These error rates are less than those that appeared in question 1.



The next matrix tested [1 0; 0 8] presented the following pattern of interference, with an exceptionally large deviation in the Y-direction. Tested error rates return is around 20% which is not close to the 30% seen when the covariance was altered in question 1. This will be due to questions one algorithms updating their threshold based upon values of sigma and question 2 having sigma values set to 50%. The artificial threshold is return more false positives as it believes everything is equally likely.

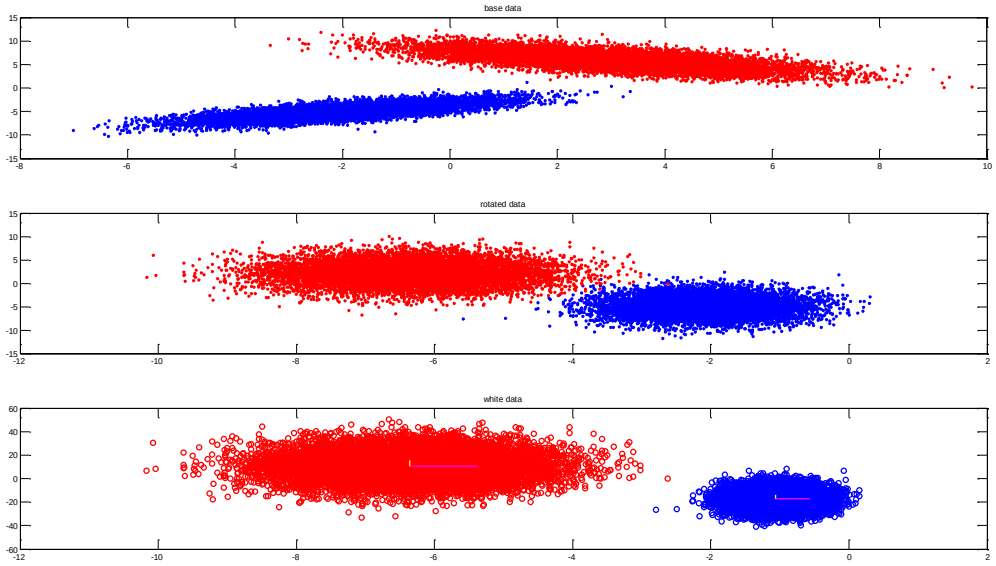


Lastly, a covariance matrix that sends the tested data set away from the control was used where the matrix was -0.9 -2; -2 5].



As with previous results, the increase in data points does not strongly impact the error rates. This is deeply troubling, because I feel these results are erroneous. They fail to fall in-line with should have followed from question one. However, I am at a loss about why and it must be something with the code I have chosen to use for implementation.

Question 3:

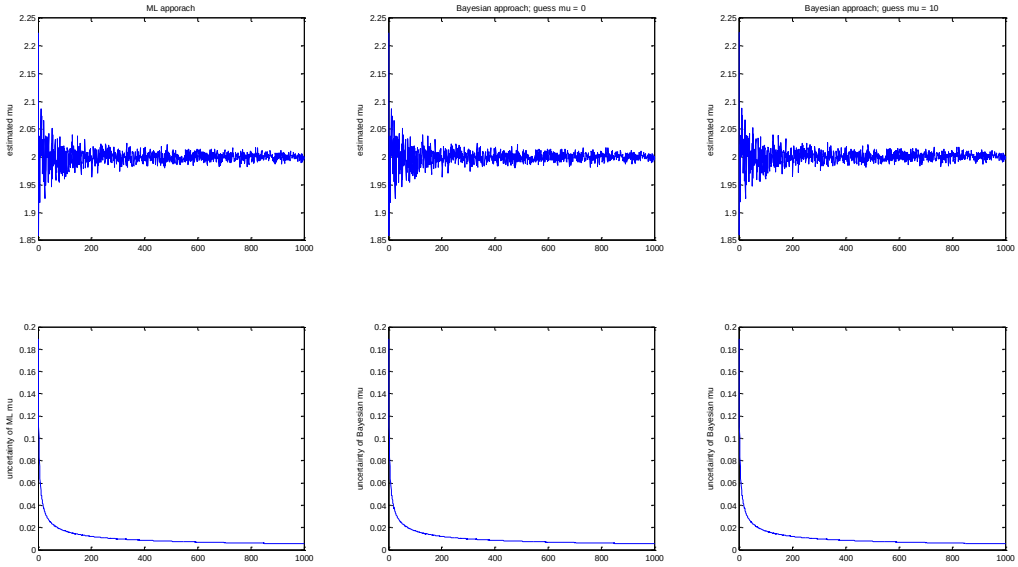


The first plot shows the two GRVs. The second shows them rotated with their eigenvalue matrix. The final plot shows the normalized (whitened) data with X-vector (magenta) and Y-vector (yellow) indicators for the eigenvectors of the resultant PCA analysis. Running ML on the initial data set provides

an error rate of 0.8% and running it again on the whitened data provides error of 5.92%. The PCA results appear worse than what is achieved with just a ML classifier, but 5% seems a reasonably small amount of error for a system handling 20,000 points. Or, again, this could be a deeper issue with incorrect instantiation of the chosen classifier schemes.

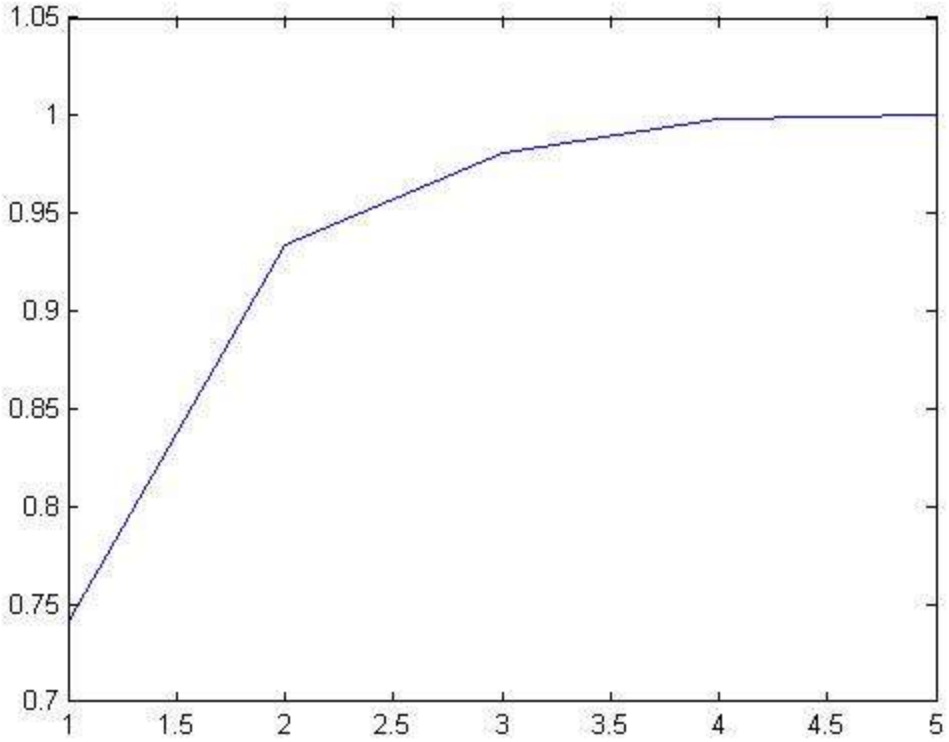
Question 4:

Tracking the variance of the correctness of the estimated mean with both an ML and Bayesian approach returned the following results. As the generated function had a μ of 2 and sigma of 1.7, all functions correctly resolved to the right value of μ. Two different initial estimates were provided to the Bayesian algorithm, but the variation does not appear to have impacted the number of samples it takes to resolve to an answer. The x-axis is from 1 to 1000, where each step accounts for 100 additional samples in the algorithm.

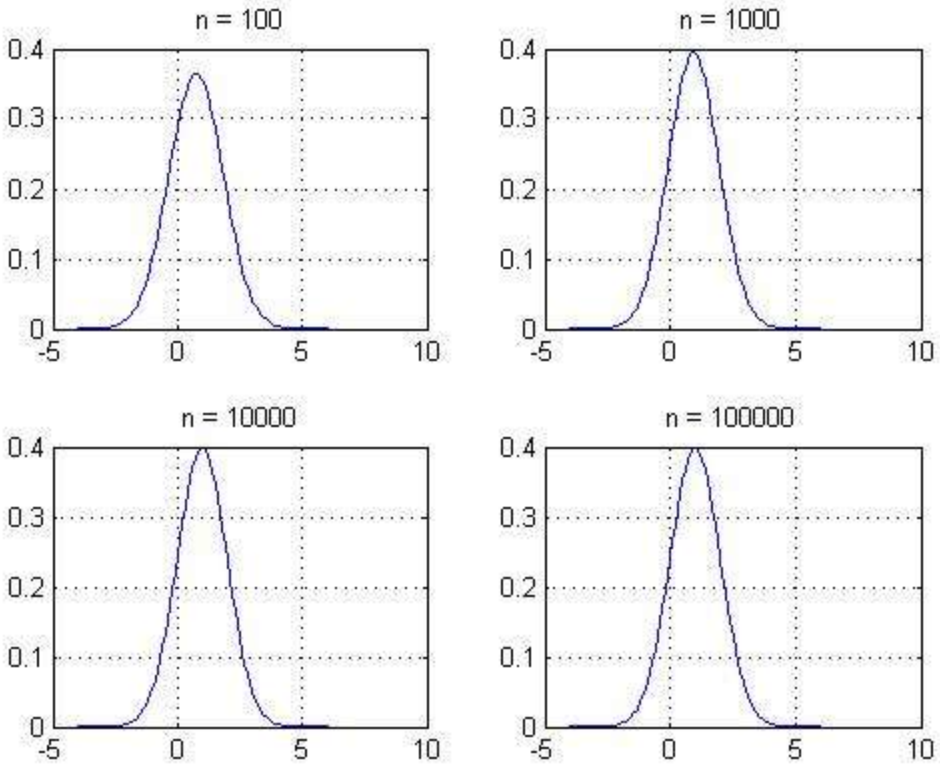


Question 5:

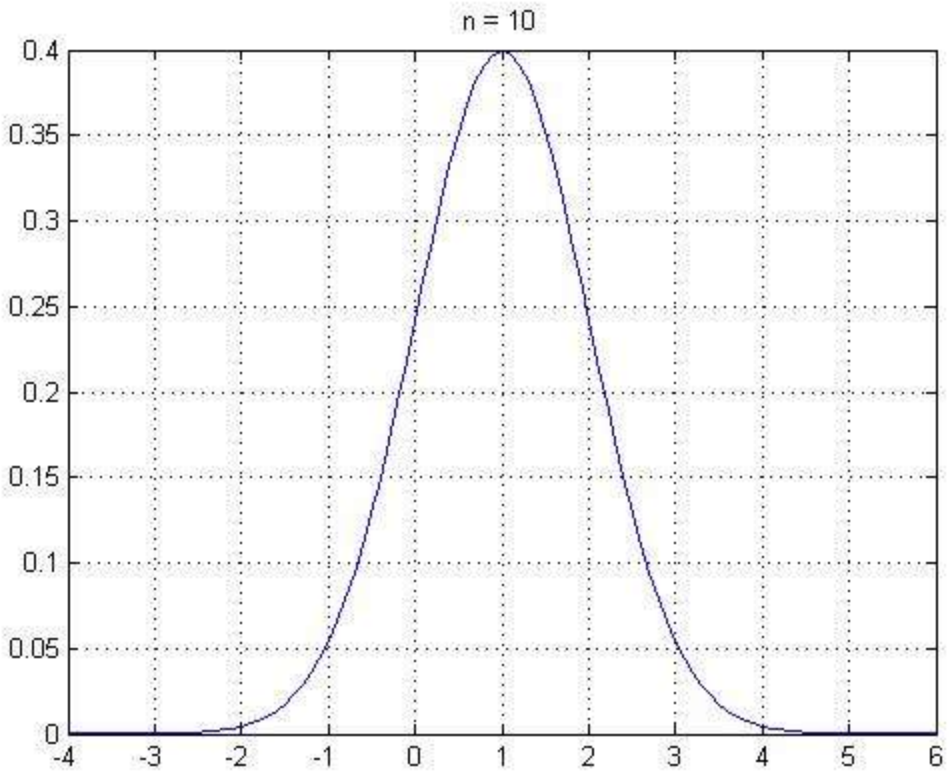
Plotting the Bayesian estimate of the data and μ as a function of the number of data points returned interesting results. The determination of μ does not appear to be as fluid as the ability to resolve a strong confidence of the area in which the data resides. The below plot of μ is working towards the correct mean of 1, but it fails to reach it within the limits of the data set.



Initially, the plots of the convergence region looked incorrect as they were nearly identical. However, upon review this highlights the beauty of Bayesian estimation's ability to provide a strong sense of how the data should appear from a very small portion of data.



While n=100, 1,000 the area under the curve is slightly off-center from one. It quickly converges for n=10,000, 100,000 and there is no apparently difference between 10,000 and 100,000. This interested me, so I also generated a plot for when n = 10. The results were most excellent.



Even with only 10 data points and an incorrect guess for μ, it still manages to converge with only knowing sigma.