This assignment shows how random number generation is an effective way to analyze data. Whenever you need an average case or an odds computation, random sampling is a great way to capture that data. Over a large sampling the average random number generated should be around half of the sampling, therefore with a good random number generator and a high enough sample size, the random number generator should get well rounded numbers as shown as the accuracy graph shown below.

That being said if used poorly, a random number generator could skew the data. For example if bad constants are chosen, or very small number of samples was used, or a poor number generator (such as the Poor_Random_Generator that we implemented) is used, then the data that is provided is not reliable. Very good example of this is the casino that used a poor random number generator lost a lot of money because of it.

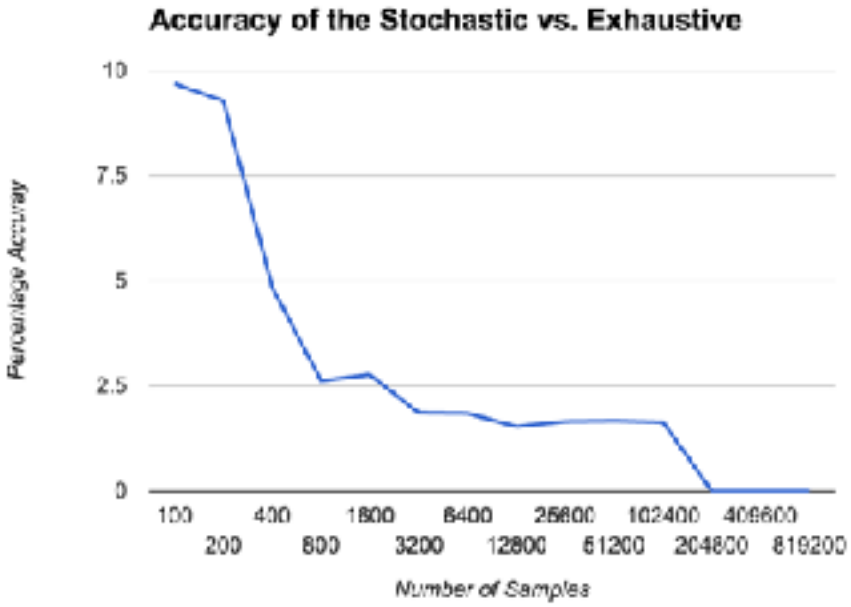

Figure 1: It is 10% at 100 samples and gets to 1% accuracy at 100,000 samples.

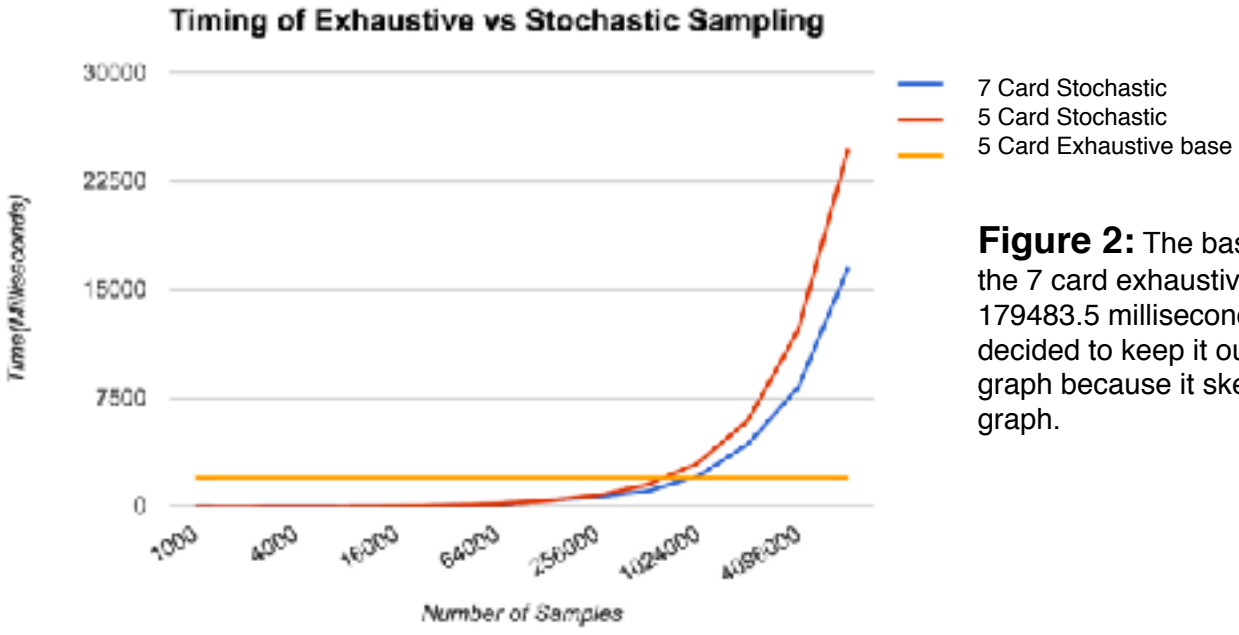

Figure 2: The base time for the 7 card exhaustive was 179483.5 milliseconds but I decided to keep it out of the graph because it skewed the graph.

What is the trade off in time vs accuracy? What would be a good trade off between accuracy and the time to compute the stochastic sampling?

The relationship is obviously more time equals more accuracy however after a point where it gets to 1% at 100,000 samples it is worth it to keep it at that time because 1% accuracy is relatively close.

How long would you expect it to exhaustively evaluate all 9 card hands? Why?

The base time for 5 card exhaustive was 1999.7ms and the 7 card was 179483.5ms. So I would expect 9 cards to be around 20 million ms or more because the 7 card exhaustive was about 90 times bigger than the 7 card and I know that factor is even more when you add more cards.

How much time did you spend on this project?

I would say that I spent around 10 hours doing this project.

What was the most time consuming part of the programming? What can you do better in the future? Did you plan enough? Did you allocate enough time from the start of the week or did you wait until the last minute to get things done?

The most consuming part I would say was making the hand class. However I did finish it in time and started the analysis earlier than I usually do so I believe I allocated enough time.

What problems came up that took disproportionate amounts of time?

Testing the accuracy of the stochastic sampling because it was a very long run time and I would get bugs almost every time.

Did writing the tests first help your development? In what way?

Yes, especially with the get rank method specifically because it made it easy to debug and run many times speeding up the process.

What insight did you get about Texas Hold'em based on your empirical analysis?

I can see why programs like this could really improve your game because looking at the percentages for every pair there are certain pairs where you would bet every time ranging to a pair of cards where you would never bet so if you know that range well it would be a huge advantage.

What are the top 10 best 2 card hands you can be dealt in Texas Hold'em? Rank them by their winning percentages.

1. Ace-Ace 2. King-King 3. Queen-Queen 4. Ace-King(same suit) 5. Jack-Jack 6. Ten-Ten 7. Ace-Queen(same suit) 8. Ace-King 9. Ace-Jack(same suit) 10. King-Queen(same suit)

How often does Ace,Ace beat King,King. How often does Ace, Ace beat Ace, King. How often does Two, Eight beat Ace, Ace?

75%, 88%, 19%

How many samples did you have to make (how many hands where dealt for each 2 card combination) to compute a valid probability of winning for those 2 cards? Justify this number of samples! (A graph would be useful here.)

I did 100000 samples because as figure 1 shows the accuracy at this number of samples is less than 1% so it is an accurate answer.

How important is the random number generator to a stochastic experiment?

It is pretty much the most crucial thing in a stochastic experiment or the data is not reliable and very uneven as we saw using the poor generator vs the better and the java one.

How well/poorly did your best and worst generator methods do compared with Java's when using the Check_Random tester.

It was very on par with it as the Check Random tester showed. The only discrepancy was that it took a little more tries to get every number in the range of numbers.

Can you think of any other tests that should be part of the Check Random Class?

Maybe the biggest occurrence of a number to see if there is a bug that favors a specific type of number.

When running your stochastic methods, what difference did you see between using Java's random generator and your random number generators?

I saw that Java's was better right of the bat because the numbers were very linear compared to the slightly skewed numbers that the other gave.