Assignment 6: Moving Along

Austin Derbique A01967241

1. Done. Look at problem assn6/problem 1 split_month.py and split_year.py

2. During these times, the period of a. June 1st, 2017 – August 31st, 2017 the eclipse seems to be prevalent. Over the times of December 1st 2016 – January 31st, I saw a drop in session data. The way of determining this is looking at the moving averages increasing.

3. a. During the period of the eclipse, we see that the higher than average session data in august move across the 365 moving day average.

b. In December of 2016 to January.

c. These cross over points mean that looking at data on a more macro scale may not be as useful as looking at the data in a more fine grained setting. Ie monthly averages vs yearly averages.

4. Done. The information appears to be similar. Although slightly off (Not every day appears to be recorded), the data is consistent with what is in the first exercise.

5. As for the dates of interest, they appear to be the same as exercises one and two.

6. Done. Found in assn6/problem_6. The functions were computed in the console and using the normalize function as defined in the previous assignment. Doing this appeared to level out the results to where there weren't such large spikes in the averages.

7. They appear to be somewhat consistent. Only a few values are much different from what was originally determined in the exercise one and two.

8. In assn6/problem8 is moreprobability.py. This code was supplied from the professor and is modified for this assignment. My testing approach consisted of generating a number between 1 and 365 several thousand times. In my exercise, I did 20,000.

9. I am not sure how to answer this question. The approach I took was to call the random function but use the same number on the first number n times. N is how many repeats you want for a specific number. If it's zero, then the number will be avoided and another random number will be gathered. These results on a large scale appear to be similar to as if it were completely random. Except for when you limit the n size to a particular value. Then everything else is roughly normal and that random number with the n values is off by an anticipated amount.

10. My computer kept segmentation faulting when trying to run my code. To better estimate, I figured out how long it could do in a 10 second interval and then multiplied by 6. For this, I determined that my computer could simulate the coin toss 269,637 times. So multiplying this by 6, the answer would be roughly 2.2 million times.

11. Using each run of 10,000 coin flips, I got widely varying results. One time resulted in 5 heads in a

row and then another time it was 11. It appears as though it is not very common considering this was out of 10,000 coin flips. Maybe the simulation is not actually random?

12. After modifying the code, the heads streak jumped up significantly to 28 in a set of 10,000 flips.

13. While doing 10,000 flips, I found that for there to be an average length of 100 flips, I needed to set the weight to be 77% for heads. I found this by trial and error. Moving originally from 60% to 80%, I settled upon 77% after a few guesses.

14. What is interesting is I only had to increase to 84% to see the increase jump to 200 in a set of 10,000 coin flips. My number was not very accurate but this number consistently produced results that met the required criteria for the question.

15. I am going to say yes, but not very well. The most similar thought to this is when you take the floor or a ceiling of a decimal. It may either round up or round down. By changing the probability of that number, you still have the possibility of rounding up or down, but there is an inherently higher accuracy in predicting which direction the coin will flip.