Characterizing Nucleotide Frequencies In Bacteriophage Lambda

Cameron Mine

When considering the significance of the information concealed within DNA sequences we find it invaluable, especially concerning sequence variation studies and comparative genomics. Solutions to uncovering the information concealed within a sequence must not be time consuming or labor intensive which is why we have enlisted the power of computational analysis. Our goal is to quickly determine the frequency of base nucleotides A, C, G, T present in the sequence and present a visual representation of our findings. The visual representation aids in identifying the areas with higher concentrations of each base nucleotide and the locations of the genome composition switches. By parsing through the DNA sequence and calculating the moving average of each nucleotide for a given window size we began the detection process so we can create a detailed plot that ultimately leads to noteworthy findings.

I. Introduction

Enterobacteria phage lambda, a type of bacterial virus that infects species of bacteria, was discovered in 1950 by Esther Lederberg an American microbiologist who was conducting experiments on E.coli mixtures. While the housing location of the virus is unknown it is believed that it lives in the genome of its host. Given the raw sequence data of the Enterobacteria phage lambda we desire to create a method of reading the genome and plotting the frequency of nucleotides in overlapping windows. In doing so we are able to paint a more detailed picture of the genome and identify areas rich with certain nucleotides.

II. Methods

I. File Parsing

The biological data required to answer our question regarding the nucleotide frequency in Enterobacteria Phage Lambda is held inside a file which we gathered from the National Center for Biotechnology Information. While these files are extremely helpful in compacting all the information we need in one place we still must parse it to access the sequence data. Doing so is often considered a challenge as there are a multitude of format types and the formats themselves often change quite frequently. Nevertheless with the help of Bio.SeqIO for Python and/or the getgenbank function for Matlab we can utilize their interfaces to select our desired format and begin working with the sequence directly. Below we have included how to do so in both languages:

Python

```
#Opening file & storing data as a record object
handle = open("lambda.fasta.txt", "r")
records = list(SeqIO.parse(handle, "fasta"))
handle.close()
```

Matlab
```
# Opening file & storing info as an object
Lambda = getgenbank('NC_001416','SequenceOnly',true);
```

II. Frequency Counting

Determining the frequency of the base nucleotides was done my counting the number of occurrences of a particular nucleotide in a sequence. This number was then divided by the total size of the sequence to give us a decimal value presenting the likelihood of seeing the nucleotide in the sequence. One simply would need to multiply this decimal with the number 100 to determine what percentage of the sequence in question is composed of that nucleotide. A tangible example of this is the number of times we find Thymine in the following sequence: AGT. First we determine the number of Thymine appearances: one and did divide this by the sequence length which gives us 0.33. From this we can tell that Thymine makes up a third of the sequence which is exactly 33%.

Python
```
"""Get_Base_Frequencies
Desc: determines the frequency of the base nucleotides in the given sequence
@requires: sequence
@returns: dictionary of base frequencies """
def get_base_frequencies(sequence):
        return {base: sequence.count(base)/float(len(sequence))
                for base in 'ATGC'}
```

III. Moving Average

Once the raw DNA sequence has been obtained from the FASTA file and converted into data we can analysis it is often too noisy. In order to smooth out the data so that we can better represent the trend we proceed to calculate the moving average. There are two types of moving averages: Simple Moving Average and the Exponential Moving Average; our solution employs the simple moving average which is formed by evaluating the average frequency of a base occurrence over a specified number of periods or windows.

To illustrate how the moving average calculation works we will take a look at Figure 1 which has a window size of 500. Here the window size helps us begin the break apart the DNA sequence into chucks of 500 nucleotides. The first chuck of the sequence starts with the nucleotide in position 0 and ends with the nucleotide at position 499. This first chuck of the sequence covers the first 500 nucleotides giving us a narrowed view of only this window. We then call the get_base_frequencies function we explained in paragraph 1, passing in this chuck of the sequence. In return we receive a dictionary holding the frequency of the bases within the sequence. The values in this dictionary are appended to the base's corresponding list where all the frequency values will be held. As the name of the function suggests in order for this to be considered a moving average we must make another chuck of the original sequence by moving our start and end point. During the second iteration the first nucleotide is popped and replaced with the nucleotide directly to the right. A similar process occurs for the ending point which instead moves over one position to the right adding new data. This movement continues until we have reached the end of the sequence. Python code:

Python
```
"""Moving_Average
```

```
Desc: calculates moving averages of bases for window size
@requires: sequence, window size, container final product
@returns: none """
def moving_average(records, window, moving_averages):
        for x in range(0, len(records[0].seq)-window):
            freq = get_base_frequencies(records[0].seq[x:window+x])
            for base in freq:
                moving_averages[base].append(freq[base])
```

Matlab is kind enough to have a built in function which does this for us and can be accessed by typing:

```
Matlab
# Finding frequency & plotting window=len(seq)/20=2425
ntdensity(Lambda,'window',2425)
```

IV. Window Size Selection

Outward appearances may lead one to believe that the window size is an arbitrary detail however this is far from true. The window size plays a large effect on the figure produced and trend analysis preformed on our results. Small window sizes provide better insight on short term trends, meaning they are best suited for predictions that will happen within a short time frame or close position. Large window sizes extend over a much greater amount of the sequence giving us a long term or broader picture of the sequence and the nucleotides it is composed of.

III. Plots

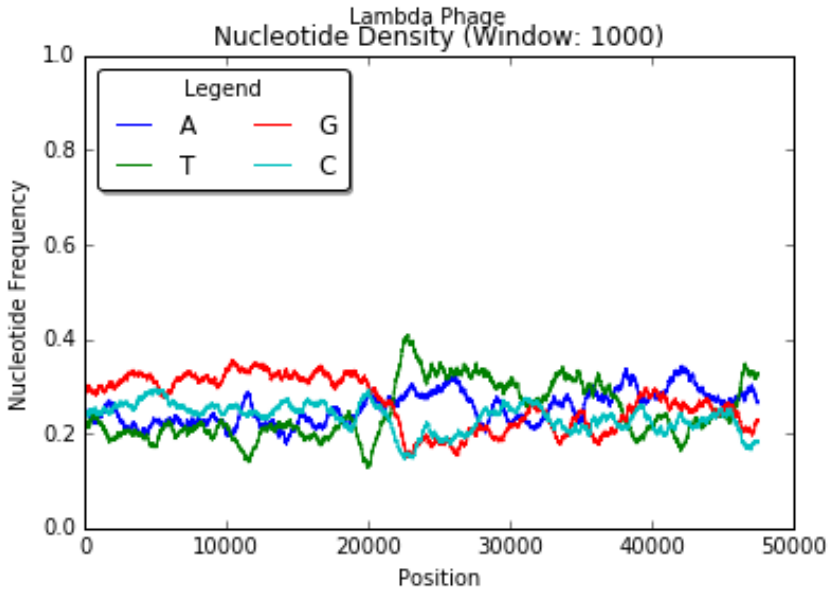

Figure 1: Python Produced

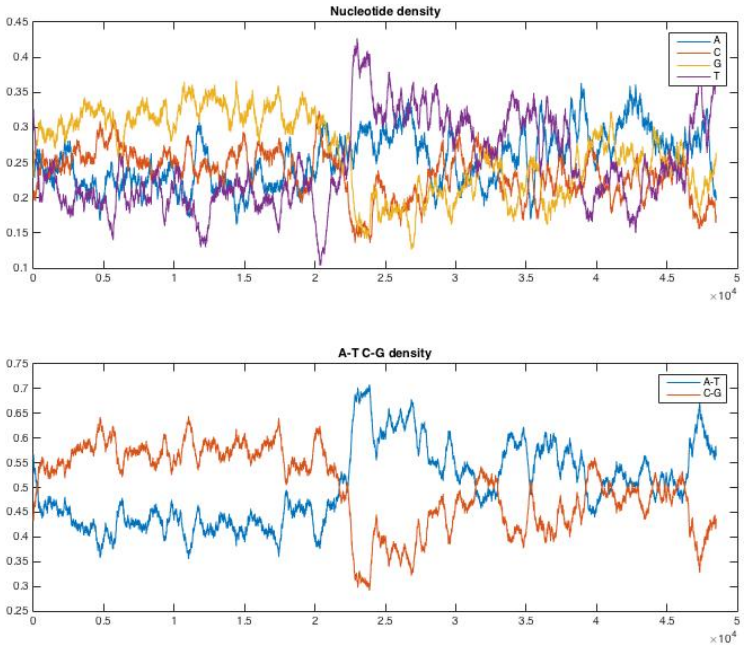

Figure 2: Window Size = 500

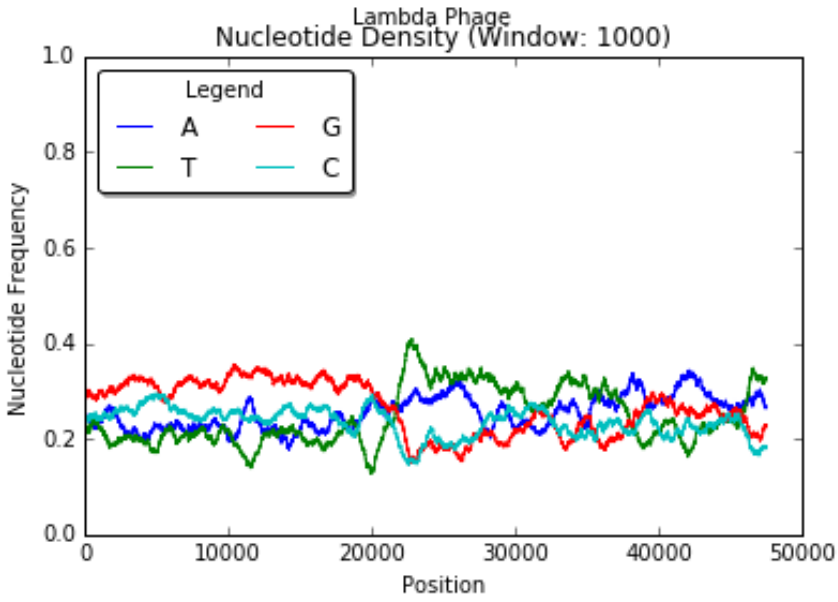

Figure 3: Python Produced

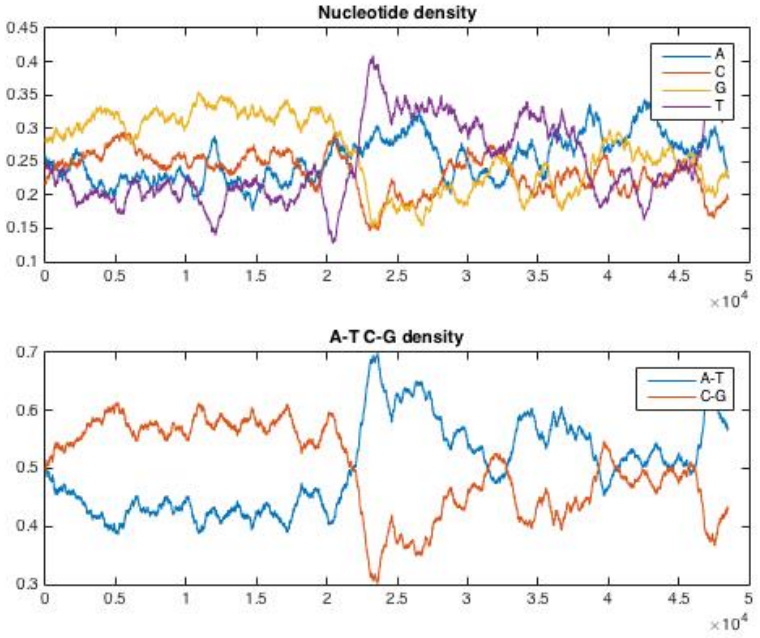

Figure 4: Window Size = 1000

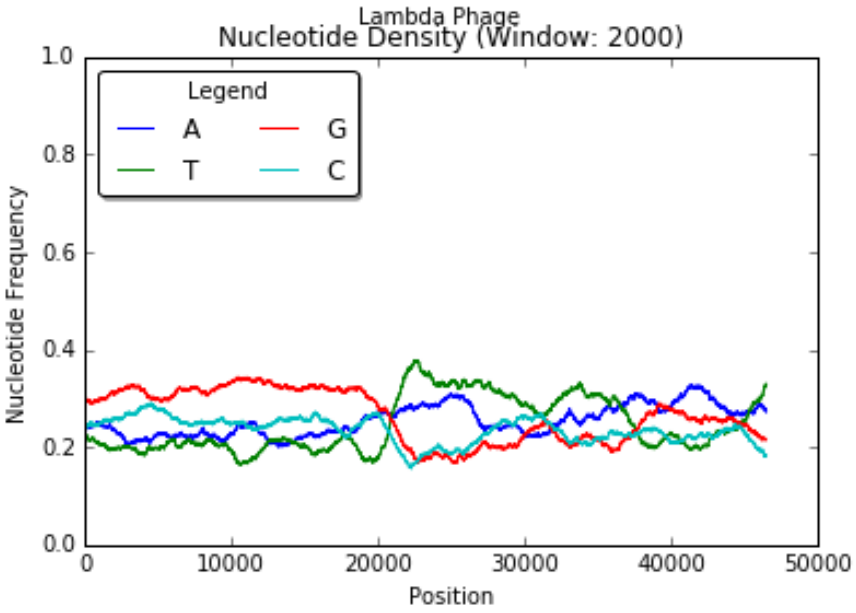

Figure 5: Python Produced

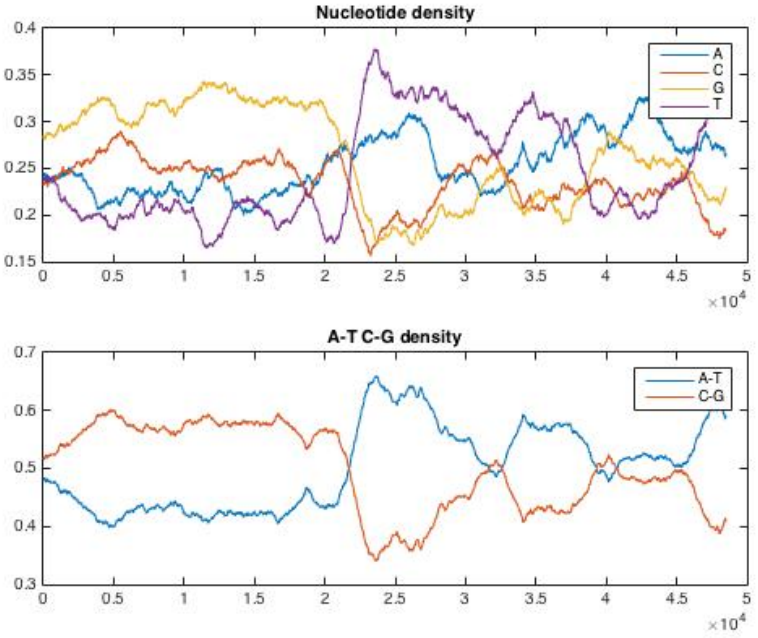

Figure 6: Window Size = 2000

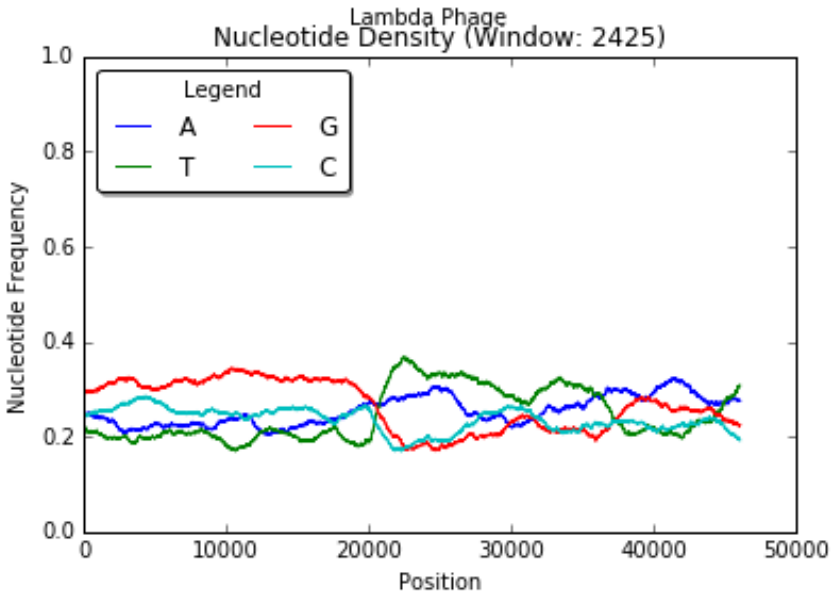

Figure 7: Python Produced

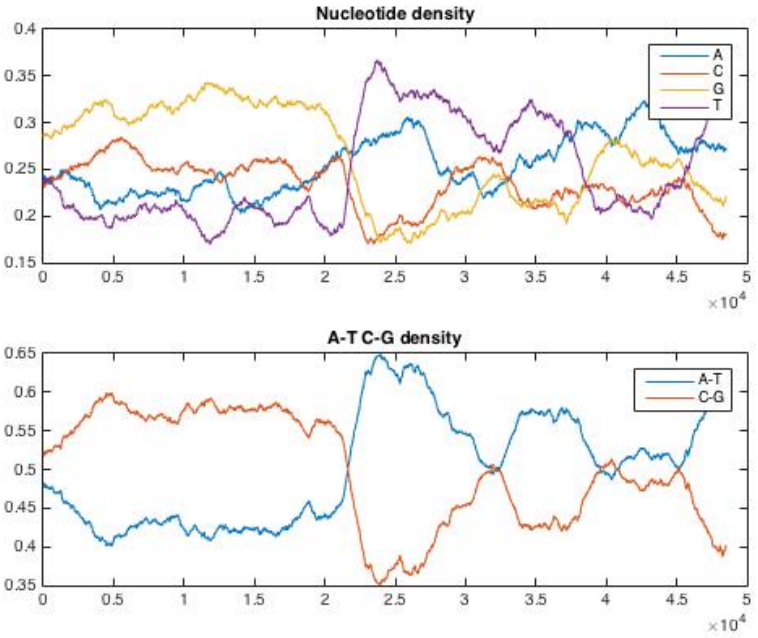

Figure 8: Window Size = 2425

IV. Conclusions

We were able to achieve a successful conclusion at the end of our experiment. Original we set out to establish a method the determine the frequency of base nucleotides A, C, G, T present in the sequence and present a visual representation of our findings. The figures that we created by following the detailed method above reveal exciting information about the genome sequence. Overlapping the window sizes enables us to pinpoint roughly where the genome composition switches between A-T rich and G-C rich. If we return to figure 8, where we have the most detailed account which results from it being the largest window, we can see that at approximately 2.2 or 2.3 from G-C rich to A-T. Additionally we see another transition from G-C to A-T at 3.3.

References

[1] Langtangen, Hans Petter, and Geir Sandve Kjetil. *"Illustrating Python via Bioinformatics Examples¶."* Center for Biomedical Computing. N.p., 22 Mar. 2015. Web. 19 Sept. 2016.

[2] *"Enterobacteria Phage Lambda."* National Center Biotechnology Information. National Center Biotechnology Information, n.d. Web. 19 Sept. 2016.

[3] *"Lambda Phage."* Wikipedia. Wikimedia Foundation, n.d. Web. 19 Sept. 2016.

[4] *"Locate CpG Islands in DNA Sequence."* Mathworks Documentation. Mathworks, n.d. Web. 19 Sept. 2016.