**Project #2**

**Objective**

Build on the basics of Python by adding tools for data science, including Numpy, Pandas, data cleaning, data wrangling and visualization,

**Due**

Friday April 6th at 5pm PST. Please create a new Project02 folder on your GitHub repository and upload your final script once completed. Remember, you should not consult with other students to work on this project unless you do so using the #project channel on Slack. This is because the communication via Slack is preserved and everyone can see it. You may ask and answer questions on Slack but do not post code. If you finish the project early, please wait to upload it to GitHub until noon on Friday at the earliest.

**Background**

Winter Dawn is a cultivar of *Fragaria x ananassa*—the garden strawberry. It is an octoploid strawberry, meaning it has 8 sets of chromosomes. Some basic information about ploidy in strawberries can be found here: <https://en.wikipedia.org/wiki/Fragaria>. It is believed that the 8 sets of chromosomes are from other strawberry cultivars, including one set from *Fragaria vesca,* a diploid strawberry. However, the exact progenitors are unknown. The Winter Dawn strawberry genome has been sequenced by the RosBREED project (<https://www.rosbreed.org/>), and the sequencing “reads” (remember the reads in the FASTQ files from project #1) have been aligned to the *Fragaria vesca* version 1.1 genome assembly*.*

There are multiple formats for storing read alignments. One format is called the pileup format. It is a tab-delimited file where each line corresponds to a single nucleotide base in the genome sequence. Each line consists of several columns in the following order:

1. The chromosome name
2. The position of the base in the sequence
3. The nucleotide base (ATCGN) in the genomic reference (*F. vesca* genome)
4. The number of reads that overlap the position in the alignment
5. The nucleotide base of the reads that overlaps the position (Winter Dawn reads) represented by the following characters. See the documentation for the meaning of each character in this string: <https://en.wikipedia.org/wiki/Pileup_format#Column_5:_The_bases_string>
6. The base quality score for reach read that overlaps the position. Simply convert to the ASCII value and subtract 33 to get the Phred quality score for the base.

**Tasks**

You will be provided with a section of the first chromosome (named LG1) of *F.vesca* with Winter Dawn read alignments represented in a pileup file. Only the first 50,000 lines of the pileup are provided to you.

Write a Python program that reads in this pileup file, examines the set of bases that are aligned to each position on the genome and identify Single Nucleotide Polymorphisms (SNPs). A SNP is a base pair difference between the reference genome (*F. vesca*) and the individual’s genome (Winter Dawn). Calling a SNP can be challenging as DNA sequencers sometimes make mistakes and incorrectly call the wrong base. Additionally, Winter Dawn has eight copies of its genome (it is an octoploid), therefore, there could be up to 8 possible variants at any given position. If we were writing SNP-calling software, we would consider several complex criteria to ensure a real SNP call. However, for this assignment we will use a simple test. To test if we should call a SNP in Winter Dawn use the following rules:

1. Skip lines with a deletion or insertion. These lines contain one of these characters: \*, + or -. We will not try to deal with insertions or deletions.
2. Skip lines with a reference skip. These lines contain one of these characters: >, or <.
3. Count the number of bases that match the reference (have a comma or a dot).
4. Count the number of bases that are variants. A variant will be any of these letters: A, T, C, G, a, t, c or g.
5. A base must have a quality score greater than 30 to be counted.
6. There must be at least 3 copies of the ***same*** variant to consider the position a SNP in Winter Dawn. Remember there are 8 sets of chromosomes, so you can potentially have 8 different variants at any given position (although this is unlikely).
7. There must be at least 10 reads aligned at a position to call a SNP in Winter Dawn.

Your program should create a tab-delimited output file that contains the following columns:

1. The chromosome name
2. The position of the SNP in the genome
3. The reference base in *F. vesca*
4. The Winter dawn base.
5. The frequency of the variant between 0 and 1.

*Note:* If a position in the genome has multiple SNPs, place each SNP on separate lines in the output file.

Finally, generate a plot using matplotlib where the x-axis is the genomic coordinates from 1 to 60881 (the last position in the file you were given). The y-axis should be a frequency value from 0 to 1. Plot the SNPs such that their position in the genome is appropriately oriented with the x-axis and the frequency of the SNP is appropriate oriented with the y-axis. Save the image with a DPI of 300.

You are free to use Numpy or Pandas or any other Python code we have discussed in class to complete the project.

**Grading**

You will be graded on the following scale

1. Your program runs without any errors: 70 points.
2. Your program generates a properly formatted output file and plot: 20 points.
3. Your script follows the criteria specified above: 10 points.
4. 5 points will be subtracted for each day late.

**Hints to help you out:**

This project builds somewhat on the previous project as you will be comparing bases and their corresponding quality scores. You do not need to implement Python Classes as we did in Project #1, but you may want to adapt some of the lines of code from your previous project.

**Best Practices**

Avoid trying to write your program in one large sitting. If you do, odds are you will spend an inordinate amount trying to figure out where bugs are occurring. Rather, program in steps.

* Plan the design of your program before you begin coding!
  1. This will help you organize your thoughts and tasks.
  2. Try to plan your program in “atomic” units that you can test individually as you go.
* Test your program by running it to make sure that one thing works before moving on to the next.
  1. Once you know that something works, then move on to the next thing.
  2. This will help tremendously because you will work out bugs as you go along.
* If you get an error, believe it!
  1. Error messages tell you exactly what is going on.
  2. Make sure you understand what the error messages mean before you start trying to apply a fix. Otherwise you may waste a lot of time trying to fix something that wasn’t really a problem…
  3. Don’t skip an error message because you don’t understand it. Deal with it.
* Don’t spend too much time on a bug.
  1. You should not spend hours trying to figure something out.
  2. Always try to resolve bugs first because this is how you learn, but don’t spend too long on a bug.
  3. Rather, send a message on the slack #project channel. You can post your error messages and ask for help--just don’t post code.