Final Project - CRISPR Design and Off-Target Analysis

Your final project starts with your Module 11 assignment. For comparison I've copied the Module 11 assignment below:

Module 11:

The purpose of this lab is to build upon your previous program, adding some of the parts you'll need to complete your final project. You'll get more details on the specifics of the final project later, but the direction this is heading is to complete a subset of the programming involved in designing CRISPRs. CRISPRs provide a new way to edit specific locations within a genome.

- Make a copy of your k-mer counting program from the previous lab. Call it `uniqueKmersEndingGG.pl`.
- Open a filehandle for writing output to `uniqueKmersEndingGG.fasta`
- Change the window length from 15 to 23.
- Rather than printing out the k-mers and associated counts, go through your hash of k-mers and only print out the first 1000 that occur once and end with GG. The reason for limiting the output to the first 1000 is to keep the time required to run BLAST manageable.
- Put a FASTA header before each k-mer, assigning a number to each based on their order in the hash. If, for example, you find 3 k-mers that only occur once, print them to a file like this:

```
>crispr_1
ATGCATGCATGATTCAGTCAAGG
>crispr_2
ATTCATGCATGATTCAGTCACGG
>crispr_3
ATGCATGCATGATTAAGTCATGG
```

- Create a BLAST database using `makeblastdb` with `dmel-2L-chromosome-r5.54.fasta` as the input and `Drosophila2L` as the output and title. Use the `makeblastdb -help` command to determine the command and options to use.
- BLAST `uniqueKmersEndingGG.fasta` against BLAST DB `Drosophila2L` using a Perl script that converts the BLAST output lines with 100% identity to GFF3 format. Write the lines with less than 100% identity to `offTarget.txt`, without changing the format of the BLAST output for these lines. Write the GFF3 output to `crispr.gff3`.

Final Project:

Make a copies of your Module 11 programs, and make the following changes:

- Recent CRISPR off-target research indicates that shortening the CRISPR can actually reduce off-target effects, so change the window size from 23 to 21.
- The last 12 positions of the CRISPR must be unique in the genome. There are several ways you can do this, and any efficient method that accomplishes the task is OK, but one suggestion is to:

  Get the last 12 positions in the same regular expression where you check for the last two positions GG.

  Get the last 12 as $2 and store them in a hash.

  Get all 21 positions of the CRISPR as $1.

  The hash to check for uniqueness of the last 12 positions will have the last 12 positions (`$2`) as the key, and the entire CRISPR(`$1`) as the value.

  A separate hash will keep a count of the occurrences of the last 12 positions in the genome. One suggestion would be to use

  ```
  $last12hash{$2}++;#Increment the count of occurrences
  of the last 12 positions.
  ```

- Generate the CRISPRs from the full Drosophila genome in `/scratch/Drosophila/dmel-all-chromosome-r6.02.fasta`.
- BLAST the CRISPRs against the full Drosophila genome using the shared BLAST database `/scratch/blast/DrosophilaAllChroms`.
- For each BLAST hit with < 100% identity and 3 or fewer mismatches to the CRISPR, count it as an off-target. If the length of the match is less than 21, remember that the difference between the match length and 21 counts as mismatch positions.
- In the GFF3 output for each CRISPR, replace:
  `Note=Some info on this oligo`
  with
  `Note=<n> off-target matches`
  where `<n>` is the number of BLAST hits you found with 3 or fewer mismatches.
- Make sure you don't store anything unnecessarily. If, for example, you can process the rows of a file as you read them, you will lose points for first storing the lines in an array then reading from the array. Make sure that everything you store in a hash or array really needs to be stored. If you store

information unnecessarily, you'll lose points.
- When you run your programs, be sure to use 1>, 2> ,and & to run the programs in the background and capture STDERR and STDOUT as separate files.
- The CPU time for running your program will be one of the metrics I use for grading. If your program is grossly inefficient you'll lose points.
- You can ask for feedback on storage efficiency, CPU efficiency, etc. before final submission so you have time to make corrections. For every line of your code, think about what it's doing, and if what it's doing is actually necessary.
- I'll have expanded office hours M-F until Thursday Dec. 11 at 8 PM. See the sign-up Wiki for available times. Be sure to sign up at least two hours before the time slot so I can plan my schedule.
- Start early.
- Don't panic. Take this one step at a time and you'll be fine.