Honors Computational Genetics

Final Exam

- This exam is open notes but all of your answers should be your OWN work
- Partial credit will be given for ALL questions, so make sure to answer everything that you can
- Email Dr. Ellison with any questions: chris.ellison@rutgers.edu

When you finish

- Sign the honor pledge at the bottom of the notebook
- Make sure you completed the Canvas quiz
- Please **save a copy of the notebook to GitHub** BEFORE 8:00am on December 17th.

IMPORTANT

- *REMEMBER TO PERIODICALLY SAVE YOUR WORK!*
- *All PYTHON code questions worth 10 or more points should have comments*
- *DON'T FORGET TO DO THE MULTIPLE CHOICE QUESTIONS ON CANVAS UNDER THE QUIZZES TAB!*

```
# This code mounts your google drive to this notebook so that you can access the files in the google drive folder we shared with
from google.colab import drive
drive.mount('/content/drive')
!ln -s /content/drive/MyDrive/gen203/final exam_data
```

```
Mounted at /content/drive
```

```
# This code installs a set of bioinformatics software packages that you will need for the exam
!pip install biopython
!pip install -q condacolab
import condacolab
condacolab.install_from_url("https://github.com/Ellison-Lab/gen203-condacolab-installer/releases/download/0.2.0/gen203_condacolab
```

```
Collecting biopython
  Downloading biopython-1.84-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (12 kB)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from biopython) (1.26.4)
Downloading biopython-1.84-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (3.2 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3.2/3.2 MB 25.8 MB/s eta 0:00:00
Installing collected packages: biopython
Successfully installed biopython-1.84
⏬ Downloading https://github.com/Ellison-Lab/gen203-condacolab-installer/releases/download/0.2.0/gen203_condacolab-0.2.0-Li
📦 Installing...
📌 Adjusting configuration...
🩹 Patching environment...
⏲ Done in 0:01:12
🔁 Restarting kernel...
```

01

14 points

Use Unix commands to do the following:

(A) make three new directories to produce the following path: `/content/A/B/C`

```
%%bash

mkdir /content/A
mkdir /content/A/B
mkdir /content/A/B/C
```

```
bash: /usr/local/lib/libtinfo.so.6: no version information available (required by bash)
```

(B) Use a single command to move into the directory `C` and then use another Unix command that prints your current location

```
%%bash

cd /content/A/B/C
pwd
```

```
/content/A/B/C
bash: /usr/local/lib/libtinfo.so.6: no version information available (required by bash)
```

(C) Use Unix commands to create a new file named `seq1.fasta`.

- The file should be in FASTA format with the sequence ID `seq1` and the DNA sequence `GATTACA`
- After making the file, use a Unix command to display the contents of the file on your screen

```
%%bash

echo '>seq1' > seq1.fasta
echo 'GATTACA' >> seq1.fasta

cat /content/seq1.fasta
```

```
>seq1
GATTACA
bash: /usr/local/lib/libtinfo.so.6: no version information available (required by bash)
```

(D) copy the file `seq1.fasta` to a new file named `seq1_copy.fasta`

```
!cp seq1.fasta seq1_copy.fasta
```

```
/bin/bash: /usr/local/lib/libtinfo.so.6: no version information available (required by /bin/bash)
```

(E) Use a single Unix command to move `seq1.fasta` and `seq1_copy.fasta` so that they are located inside the directory `C`.

```
!mv seq1.fasta seq1_copy.fasta /content/A/B/C
```

```
/bin/bash: /usr/local/lib/libtinfo.so.6: no version information available (required by /bin/bash)
```

(F) Use a single command to move into directory `C` and another command to list the contents of the directory.

```
%%bash

cd /content/A/B/C
ls
```

```
seq1_copy.fasta
seq1.fasta
bash: /usr/local/lib/libtinfo.so.6: no version information available (required by bash)
```

(G) Use a single command to move into directory `C` and then:

- Use a command to search `seq1.fasta` for the character that precedes the sequence ID in a FASTA file
- Use another command to print the contents of both FASTA files to the screen

```
%%bash

cd /content/A/B/C/
grep '>' seq1.fasta
cat seq1.fasta seq1_copy.fasta
```

```
>seq1
>seq1
GATTACA
>seq1
GATTACA
bash: /usr/local/lib/libtinfo.so.6: no version information available (required by bash)
```

02

21 points

The file `exam_data/genome_features.tab` contains the coordinates for various types of genome features in the Drosophila genome. The tab-delimited columns in the file are:

1. Chromosome ID
2. Feature start coordinate
3. Feature end coordinate
4. Source of feature (e.g. the name of the group/program that annotated the feature)
5. Type of feature (e.g. what kind of feature it is)

Note: Although it may appear that the features in this file are sorted according the their start coordinates, this may not be true for all of the features and you will need to use Unix to ensure they are properly sorted for some of the questions.

Use Unix commands to answer the questions below:

(A) How many lines are in the file?

```
!wc -l exam_data/genome_features.tab
```

```
/bin/bash: /usr/local/lib/libtinfo.so.6: no version information available (required by /bin/bash)
85740 exam_data/genome_features.tab
```

(B) Print the first 5 lines

```
!head -n 5 exam_data/genome_features.tab
```

```
/bin/bash: /usr/local/lib/libtinfo.so.6: no version information available (required by /bin/bash)
2L      7874    8318    DRSC_dsRNA      RNAi_reagent
2L      8238    8534    BKN_dsRNA       RNAi_reagent
2L      8267    8580    DRSC_dsRNA      RNAi_reagent
2L      8267    8580    HFA_dsRNA       RNAi_reagent
2L      10964   10984   TRiP_4  RNAi_reagent
```

(C) Print the last 5 lines

```
!tail -n 5 exam_data/genome_features.tab
```

```
/bin/bash: /usr/local/lib/libtinfo.so.6: no version information available (required by /bin/bash)
2R      25253314        25253440        FlyBase exon
2R      25253581        25253761        FlyBase exon
2R      25253820        25254078        FlyBase exon
2R      25254128        25254535        FlyBase exon
2R      25255009        25255318        FlyBase exon
```

(D) Use a single command to print the number of features present on each chromosome that is listed in the file.

```
!cut -f 1,5 exam_data/genome_features.tab | sort | uniq -c
```

```
/bin/bash: /usr/local/lib/libtinfo.so.6: no version information available (required by /bin/bash)
  15623 2L      exon
  24303 2L      RNAi_reagent
    919 2L      transposable_element
  17495 2R      exon
  26042 2R      RNAi_reagent
   1358 2R      transposable_element
```

(E) Use a single command to print the first feature (i.e. the feature whose start coordinate is closest to zero) on chromosome 2L

```
!cut -f 1-5 exam_data/genome_features.tab | grep 2L | sort -k2,2 -n -r | tail -1 | column -t
```

```
/bin/bash: /usr/local/lib/libtinfo.so.6: no version information available (required by /bin/bash)
2L  7529  8116  FlyBase  exon
```

(F) Use a single command to print the last feature (i.e. the feature with the largest end coordinate) on chromosome 2R

```
!cut -f 1-5 exam_data/genome_features.tab | grep 2R | sort -k2,3 -n -r | head -1 | column -t
```

```
/bin/bash: /usr/local/lib/libtinfo.so.6: no version information available (required by /bin/bash)
2R  25272400  25272919  DRSC_dsRNA  RNAi_reagent
```

(G) Use a single command to print each type of feature whose source is from `FlyBase`, without duplicates

Note: feature type is listed in column number 5.

```
!cut -f 4,5 exam_data/genome_features.tab | grep FlyBase | sort | uniq
```

```
/bin/bash: /usr/local/lib/libtinfo.so.6: no version information available (required by /bin/bash)
FlyBase exon
FlyBase transposable_element
```

(H) How many features on chromosome 2L share the exact same start and end coordinates with at least one other feature?

- Use a single Unix command to answer this question

```
!cut -f 1,2,3 exam_data/genome_features.tab | grep 2L | sort | uniq -d | wc -l
```

```
/bin/bash: /usr/local/lib/libtinfo.so.6: no version information available (required by /bin/bash)
5438
```

03

15 points

Use the multiple sequence alignment in the file `exam_data/msa.fasta` to answer the questions below

(A) use biopython to print the following information about the multiple sequence alignment:

- `There are ______ sequences and ______ columns in the alignment.`
- `______ columns contain only a single gap`
- `______ columns contain more than one gap`

```
from Bio import AlignIO

align = AlignIO.read('exam_data/msa.fasta', 'fasta')

# number of columns and sequences
alen = len(align[0])
num_seq = 0
for i in align:
    num_seq += 1
# print(num_seq)

# counters for single and multiple gaps
single_gap_col = 0
multiple_gap_col = 0

# iterate through each column
for j in range(alen):
    column = align[:,j]
    gaps = column.count('-')  # count number of gaps
    # print(gaps)

    if gaps == 1:
        single_gap_col += 1
    elif gaps > 1:
        multiple_gap_col += 1
    else:
      continue

print("There are", num_seq, "sequences and", alen, "columns in the alignment.")
print(single_gap_col, "columns contain only a single gap")
print(multiple_gap_col, "columns contain more than one gap")
```

```
There are 4 sequences and 610 columns in the alignment.
11 columns contain only a single gap
15 columns contain more than one gap
```

(B) Use biopython to make a distance matrix for this alignment

```
from Bio.Phylo.TreeConstruction import DistanceCalculator
from Bio import AlignIO

aln = AlignIO.read('exam_data/msa.fasta', 'fasta')
calculator = DistanceCalculator('identity')
dm = calculator.get_distance(aln)
print(dm)
```

```
sequence1   0.000000
sequence2   0.024590    0.000000
sequence3   0.013115    0.037705    0.000000
sequence4   0.047541    0.022951    0.060656    0.000000
    sequence1   sequence2   sequence3   sequence4
```

(C) What two sequences are the most similar?

sequence1 and sequence3

(D) What two sequences are the most dissimilar?

sequence3 and sequence4

04

10 points

Use biopython to parse the Illumina sequences in `exam_data/yeast.fastq`.

- Write the sequences whose average quality score is 35 or greater in FASTQ format to the file `my_filtered.fastq`.
- Print the total number of sequences in the original file, the number of sequences in the filtered file, and the number of sequences that were discarded:

```
Total Sequences: ______

High Quality Sequences: ______

Discarded Sequences: ______
```

```
from Bio import SeqIO
import statistics

total = 0
high_quality = 0
discarded = 0

filtered = open('my_filtered.fastq', 'w')

# iterate over the original FASTQ file
for record in SeqIO.parse('exam_data/yeast.fastq', 'fastq'):
    total += 1

    # find quality scores for each sequence
    scores = record.letter_annotations["phred_quality"]
    avg_score = statistics.mean(scores)

    # filtering based on avg score of 35
    if avg_score >= 35:
        high_quality += 1
        SeqIO.write(record, filtered, "fastq")    # writing scores to new file
    else:
        discarded += 1

print("Total Sequences:", total)
print("High Quality Sequences:", high_quality)
print("Discarded Sequences:", discarded)

filtered.close()    # I didn't forget this time :)
```

```
Total Sequences: 1410
High Quality Sequences: 1393
Discarded Sequences: 17
```

05

7 points

Use the file `exam_data/yeast.fastq` for this question

- Align these sequences to the yeast chromosome #8, which is in the file `exam_data/chr08.fasta`
- Convert your alignments to a sorted BAM file named `my_sorted_alignments.bam`

```
%%bash

bowtie2-build exam_data/chr08.fasta chr08.fasta
bowtie2 --no-unal -x chr08.fasta -U exam_data/yeast.fastq -S my_sorted_alignments.sam
samtools view -b my_sorted_alignments.sam > my_sorted_alignments.bam
```

```
    Renaming chr08.fasta.1.bt2.tmp to chr08.fasta.1.bt2
    Renaming chr08.fasta.2.bt2.tmp to chr08.fasta.2.bt2
    Renaming chr08.fasta.rev.1.bt2.tmp to chr08.fasta.rev.1.bt2
    Renaming chr08.fasta.rev.2.bt2.tmp to chr08.fasta.rev.2.bt2
    1410 reads; of these:
      1410 (100.00%) were unpaired; of these:
        0 (0.00%) aligned 0 times
        1410 (100.00%) aligned exactly 1 time
        0 (0.00%) aligned >1 times
    100.00% overall alignment rate
```

Write a short paragraph addressing the points below:

- What information is contained in the SAM file?
- How is the SAM format different from the BAM file format?
- Why would you want to convert a SAM file to a BAM file?
- What gets sorted when you sort a BAM file and in what order?

The Sequence Alignment/Map (SAM) stores the Illumina sequence ID and quality scores of the alignment, the alignment location including the reference sequence ID and position mapping quality, the locations of mismatches and/or gaps in alignment, and the alignment strand, whether it is in forward or reverse. The SAM format is tab-delimited; the alignments have to be compressed to BAM format, which is only in binary and, therefore, not readable by humans. By compressing into a BAM format, the software can sort the alignments based on the genome positions, aided by the compressed format, which saves space.

06

9 points

The yeast chromosome 8 is over 500 kb long but the Illumina sequences in `exam_data/yeast.fastq` are from a much smaller contiguous region, located somewhere on the chromosome.

IMPORTANT: these questions purposefully do not tell you whether to use Unix or Python, nor do they tell you whether there is a specific Unix program that will accomplish the task. However, you have done all of these things previously in the homework assignments.

(A) Create a tab-delimited file that lists the per-base sequencing coverage for each position of chromosome 8

```
%%bash
samtools sort my_sorted_alignments.bam > my_really_sorted_alignments.bam
samtools depth -a my_really_sorted_alignments.bam > coverage_per_position.tab
```

```
bash: /usr/local/lib/libtinfo.so.6: no version information available (required by bash)
```

(B) Determine the start and end positions of the chromosome 8 region that was sequenced.

Print the start and end positions, assuming the first nucleotide of chromosome 8 is position 1.

```
read_coverage = open("coverage_per_position.tab")
position = []

for coverage in read_coverage:
  coverage = coverage.rstrip()
  cov_list = coverage.split("\t")

  num_pos = int(cov_list[1])
  depth = int(cov_list[2])

  if depth > 0:
    position.append(num_pos)

min_value = min(position)
max_value = max(position)

print("Read Coverage begins at position", min_value)
print("Read Coverage ends at position", max_value)

read_coverage.close()
```

```
Read Coverage begins at position 28166
Read Coverage ends at position 32936
```

(C) All of the Illumina sequences in `exam_data/yeast.fastq` are 100 bp long.

Calculate the average sequencing coverage for the region from (B) above.

Show your work.

```
num_reads = 1410    # from bowtie2
size = 32936 - 28166
length = 100

coverage = (num_reads * length)/size
print(coverage)
```

```
29.559748427672957
```

07

9 points

(A) Use `Velvet` to assemble the Illumina sequences in `exam_data/yeast.fastq` by using a Kmer size of 27

```
!velveth YEAST 27 -short -fastq exam_data/yeast.fastq
```

```
/bin/bash: /usr/local/lib/libtinfo.so.6: no version information available (required by /bin/bash)
[0.000000] Reading FastQ file exam_data/yeast.fastq;
[0.004120] 1410 sequences found
[0.004139] Done
[0.066690] Reading read set file YEAST/Sequences;
[0.067218] 1410 sequences found
[0.075104] Done
[0.075119] 1410 sequences in total.
[0.075286] Writing into roadmap file YEAST/Roadmaps...
[0.077499] Inputting sequences...
[0.077621] Inputting sequence 0 / 1410
[0.197294]  === Sequences loaded in 0.119801 s
[0.197349] Done inputting sequences
[0.197357] Destroying splay table
[0.200392] Splay table destroyed
```

```
!velvetg YEAST -exp_cov 30 -cov_cutoff auto
```

```
[0.233790] Scaffolding node 0
[0.243937]  === Nodes Scaffolded in 0.033618 s
[0.243985] Preparing to correct graph with cutoff 0.200000
[0.244062] Cleaning memory
[0.244088] Deactivating local correction settings
[0.244106] Pebble done.
[0.244112] Concatenation...
[0.244118] Renumbering nodes
[0.244123] Initial node count 1
[0.244129] Removed 0 null nodes
[0.244134] Concatenation over!
[0.244139] Removing reference contigs with coverage < 10.957038...
[0.244149] Concatenation...
[0.244155] Renumbering nodes
[0.244160] Initial node count 1
[0.244165] Removed 0 null nodes
[0.244170] Concatenation over!
[0.244385] Writing contigs into YEAST/contigs.fa...
[0.245472] Writing into stats file YEAST/stats.txt...
[0.245656] Writing into graph file YEAST/LastGraph...
[0.247669] Estimated Coverage cutoff = 10.957038
Final graph has 1 nodes and n50 of 4745, max 4745, total 4745, using 1410/1410 reads
```

(B) Use Unix to print your assembled sequence to the screen

```
!cat YEAST/contigs.fa
```

```
AATATAAAACAAAAACGATAACGGCGACTGTCAAAAAGCGAGGAGTTATTACGGTAATGA
GAAATATAATTGATGCGCATTGAATTAGGCAAAATATAGTTACTTCTAAGTATGGAATCA
ATTCTTGATCAACACCTTCGATGTCCTTTGAAAAGCGATTCATGATTCTACCAACCGGCG
TCACGTCGAAAAATCGTATTTGGGCATGTAGAACTAGATCTAGCAGATTATTAAAGATCT
TTCTGGAGGCTCGCATACCGGATAAAAACGTCATCATTGTTTTAAAACCACCTAGCATTG
CCTGAATGATACCAATAAGAAAATATACGGTTAAGTAATAAAATGCATTATGTTTATTTT
TCGAAGAGTCGGTCATCCCTTTTAATGGCAGCGTGTCCATCGCAAAACCTGGAGCATTTA
TTCGTACATTGGTATCGTTGACCCAATGTCGTATCCACCAAGACTGACTGATGAACAAAA
TTTGAGCTGTGATATAAAGAGCGAACAGGGCTGTTAAAGCTTTGAAGCCTCCAAAAAATT
TCAGGTACCATTTATAAACATCGGGGCTTATGGCACCGTTTGATTTTTCTTCCTCTTCTA
TTAGCTGGCCATCATTGACAAAATTTGCATCAAAATTTATGTTCTCGGTGACAGGTTCGA
TTTTCTGAGAGTCATTTTTTCTGGGAGCTTTTAATCTATTAGCGTTCTTTTCATTAATGC
TATCTCGAGAAGAAAGTTGAACATATTTTTCCTTAAAAAGCCCTTTGCTTTGTAATTCTG
TAATAGTTCCTTGATTCTTCACTTTGCCATTTTCCAACACAATCGCGAAATGGGCATTTC
TAAGTGTTAATGAAACATTGTGCGTAACTAAAATGCAGGTTCTATTTTTCATTAGTGGAC
CTGTGATGCAATTTTCATAGATCCATACAGCAGTATGTGAATCGACAGCGCTCAAACAAT
CATCTAGTAAGACATGCTTAGCACTCGAATAAACAGCTCTCGCCAAGGAAATTCTCTGTT
TCTGCCCTCCTGATAAAGTTATACCCTTTTCACCAATATCTGTTAGGTCACCTGCTGGTA
AAATCTCCAGGTCTCTTTTCAGCCCACATGCATCAATTACTTTGTTGTACCTATCCTCGT
TATAGAAGTTATCAAAGATAATATTGTTTTTTACCGTGTCATTTAATAGCCACGCACTTT
GTGAACAATATGCGAAGGAATTGGTTAAACCTTCGCAGTCGGGAATTAAATCATGCTTTG
GTTCTAAGCTCGGAACAATGATAGAGCCACTAATTAGATTTAGTTCACCCAGTAAACCCA
GCAGCAATGCACTTTTACCAGATCCTGTAGAACCCAAAATCAAATTTAACTTACCAATTT
GAAATTTAATATTCAAACCACATAATTTGAATGCATTCATATCGCTGTCATTTTCATTCC
AGGTTAAAGTCGCATTTTTAAATTCAATTTTATTTTTGTCTGGAGATATGGTTAGTTGAT
TATATTTTTCTGTATCGTCCATCCTTAAAAAATCGCTTATTCTTTTTAGAGAGACTTTTG
ATTGATTTATGAAACTTAGCATATTTGATAATTGATCCAGGGGTGTCTTTAACAAAGTGA
AGAGTGACAAAGTAGTGAAAGCAAGCGGGGCATTCAAATCTTCATGTTGAACAAATGTAC
AGATGGCGAAAGTGACACCTGTCACCAAGGTCGGTGTCACGAACCAAAGAAAAGAAGTTA
CGGACCACACCAAAGATTTTTTTAATAAGGATCTTAATTCCTTTTGCCTTATTGATTTGA
TTTCATTTATAATATTCCTTTCCCAAGCAAAATATTTGACAATTCTTATGTTCTGTAAGC
ACTCGTTCAATTTTGAGATTCTTTGGTCAGTACATTTCAGTGTTTGCTTTTGAAACTTAC
```

```
AATAGTAGCTCAATAGCTGAGTTCTTCCTAAACGAGTAAAATCATCAACTTCCCAAAATG
AACCATTATTTCGGATAATAAGGGGATCCgt
```

(C) Is your assembled sequence a contig or a scaffold? Explain how you know

This is a single contig because there is one reported contig and the lack of N's indictative of scaffolds.

(D) What is the best match for the sequence above in the Genbank refseq_protein database? Write a sentence listing the best match and explaining what you did to figure this out.

putative ATP-binding cassette multidrug transporter VMR1 [Saccharomyces cerevisiae S288C]

By using the NCBI BLASTX tool for a nucleotide sequence to protein, the results of 99.94% percent identity with an e-value of nearly 0 and a query coverage of 99% suggests this is the best match.

08

15 points

This question involves a different yeast gene that is 186 bp long

(A) Use BLAST to search the DNA sequence of the gene in `exam_data/yeast_gene.fasta` against the yeast chromosome 8 (`exam_data/chr08.fasta`).

- Use an e-value threshold of 1e-10
- Output the results in a tab-delimited format and let them print to the screen

```
%%bash

cp exam_data/chr08.fasta /content
makeblastdb -dbtype nucl -in chr08.fasta
blastn -query exam_data/yeast_gene.fasta -db chr08.fasta -outfmt 6 -evalue 1e-10 > output.tab
# cat output.tab
```

```
Building a new DB, current time: 12/17/2024 03:19:46
New DB name:   /content/chr08.fasta
New DB title:  chr08.fasta
Sequence type: Nucleotide
Keep MBits: T
Maximum file size: 1000000000B
Adding sequences from FASTA; added 1 sequences in 0.0115981 seconds.
bash: /usr/local/lib/libtinfo.so.6: no version information available (required by bash)
```

```
!cat output.tab
```

```
/bin/bash: /usr/local/lib/libtinfo.so.6: no version information available (required by /bin/bash)
YHR055C chrVIII 100.000 186     0       0       1       186     215064  214879  1.74e-96        344
YHR055C chrVIII 97.849  186     4       0       1       186     5601    5786    8.17e-90        322
```

(B) How many copies of this gene are on chromosome 8?

Explain how you were able to tell how many copies there are and whether the copy/copies are likely to be real copies of the gene or just spurious results.

There are likely 2 real copies of the gene on chromosome 8 based on the high percent identity (100.00% and 97.849%) in consideration of a very low e-value.

(C) Looking at your BLAST results, fill in the table below that lists the start and end coordinates on chrVIII where each copy of the gene is located.

Note that the start coordinate should refer to where the gene begins (i.e. its start codon) and the end coordinate should refer to where the gene ends (i.e. its stop codon).

Be sure to fill in the column in your table that lists whether the gene copy is on the forward or reverse strand of chromosome 8

Copy	Start	End	Strand
YHR055C	215064	214879	reverse
YHR055C	5601	5786	forward

(D) Write python code to produce a file in FASTA format named `blast_copies.fasta` containing the sequence for each gene copy listed in (C)

```
from Bio import SeqIO

output = open('blast_copies.fasta', 'w')

blast_result = [
    ('YHR055C', 215064, 214879, '-'), ('YHR055C', 5601, 5786, '+')
]
# sequences = []

for record in SeqIO.parse('exam_data/chr08.fasta', 'fasta'):
    for name, start, end, strand in blast_result:
        subseq = record.seq[start - 1:end]
        print(subseq)
        # SeqIO.write(subseq, output, 'fasta')

        if strand == "-":
            subseq = subseq.reverse_complement()
            print(subseq)
            # SeqIO.write(subseq, output, 'fasta')
```

```
ATGTTCAGCGAATTAATTAACTTCCAAAATGAAGGTCATGAGTGCCAATGCCAATGTGGTAGCTGCAAAAATAATGAACAATGCCAAAAATCATGTAGCAGCCCAACGCGGTGTAACACCGACG
```

(E) Create a multiple sequence alignment from the file you produced in (D).

Note: if you were unable to complete (D), you can still get credit for this question by listing the command you WOULD have run

```
!mafft —auto /content/blast_copies.fasta > blast_copies.MAFFT.fasta
```

SIGN HONOR PLEDGE BELOW

On my honor, I have neither received nor given any unauthorized assistance on this examination

KRYSTAL KUANG