# Class Project

## Objective

To combine what you have learned about the command-line, regular expressions, command-line data wrangling, Bash coding and Nextflow towards the accomplishment of a high-throughput computing task.

## Due

Friday May 7th before 5pm PST.

## Background

This project uses the NCBI sequence similarity tool BLAST. If you are unfamiliar with it, you can learn more about it here: <https://en.wikipedia.org/wiki/BLAST>. BLAST is already installed on Kamiak.

## Tasks

### Step 1

Given any FASTA file (<https://en.wikipedia.org/wiki/FASTA_format>) containing 10,000 or more protein sequences, split that file into 10 equally sized files where each file contain the same number of sequences (except perhaps the last file). Using Kamiak, execute the BLAST blastp program on all 10 files. You must ***automate submission*** of those 10 files in some way. You cannot have 10 different Kamiak submission scripts. Use the *Oryza sativa* (rice) protein file found here /data/ficklin\_class/AFS505/course\_material/data/all.pep as the input ***query*** file. The file contains entries in FASTA format such as:

>LOC\_Os01g01010.1 protein|TBC domain containing protein, expressed

MSSAAGQDNGDTAGDYIKWMCGAGGRAGGAMANLQRGVGSLVRDIGDPCLNPSPVKGSKM

LKPEKWHTCFDNDGKVIGFRKALKFIVLGGVDPTIRAEVWEFLLGCYALSSTSEYRRKLR

AVRREKYQILVRQCQSMHPSIGTGELAYAVGSKLMDVRTMSKETHIAEEVSTSQQTSQNT

AGSLVEDSDYGPGGAQQSQKRESCSKSAELVGFNVHNDTSLYDSSNFIVSSTEVNNCSKD

SQDYNDMGEPRYDTETFDDYPSLPVTNFFSTDGVGSNGVDKNHCSFSVPEDRLRHRDERM

HSFQINNNIDLIIESNSCSSDVFRASNSDSAIFHSDAYKQDRWLDDNGYNREVIDSLRIS

DAPEADFVDGTKSNSVVASKDRVSEWLWTLHRIVVDVVRTDSHLDFYGESRNMARMSDIL

AVYAWVDPSTGYCQGMSDLLSPFVVLYEDDADAFWCFEMLLRRMRENFQMEGPTGVMKQL

QALWKIMEITDVELFEHLSTIGAESLHFAFRMLLVLFRRELSFEESLSMWEMMWAADFNE

DVILHLEENCLEPLLVDMRNDLSCEVKEEHRVNSYTRRKSKSRKPHHRNGEMRVACNLGM

KPNTRNPLCGLSGATIWARHQQMPHISTNVLAKNGDDDLPIFCVAAILVINRHKIIRETR

SIDDAIKMFNDNMLKINVKRCVRMAIKLRKKYIYKLLKGGSE

Each protein entry contains a definition line (that begins with a ‘>’ character, followed by multiple lines of protein sequence). Each protein sequence begins with a new definition line. This file contains over 66K *Oryza sativa* (rice) proteins. Test it out by running this command: grep -c ">" /data/ficklin\_class/AFS505/course\_material/data/all.pep . In the FASTA format, the name of the sequence (in this case a protein sequence) is the first word of the definition line. In the example above this would be “LOC\_Os01g010101.1”

You should use BLAST to compare these rice proteins with other similar proteins in the ExPASy SwissProt protein database. SwissProt is a manually curated list of proteins with experimentally verified functional details. This type of comparison is very useful to help identify the potential function of proteins because proteins with similar sequences will tend to have similar function. For newly sequenced genomes, whose function of proteins is unknown, this can be useful to help elucidate their function. A copy of the SwissProt protein database, suitable for BLAST can be found here: /data/ficklin\_class/AFS505/course\_material/data/swissprot (note, the files is already indexed in binary format) and the to test your script (). When you execute BLAST, be sure to set the output file format to tab-delimited and the e-value threshold cutoff to 1e-6. BLAST is a multi-threaded program, so to speed up the analysis, request 5 threads per job.

***Hint***: if you use Nextflow you can use the splitFasta operator (<https://www.nextflow.io/docs/latest/operator.html?#splitfasta>) to automate splitting.

Summary of requirements for step 1:

Use whatever method you want to do this step: Python, BASH, Nextflow, command-line tools, etc. Just be sure the to follow these criteria:

* Write code to divide the input query file into 10 equally sized chunks and save them in files
* Write code to execute BLAST on each “chunk” file automatically
* Use the following settings when running BLAST:
  + Query file: /data/ficklin\_class/AFS505/course\_material/data/all.pep
  + Database File: /data/ficklin\_class/AFS505/course\_material/data/swissprot
  + Output Format: Tab-delimited
  + E-value threshold: 1e-6
  + Threads: 5

### #!/usr/bin/bash

### #SBATCH --partition=ficklin\_class

### #SBATCH --account=ficklin\_class

### #SBATCH --job-name=swissprot\_blastp

### #SBATCH --output=swissprot\_vs\_all.pep.out

### #SBATCH --error=swissprot\_vs\_all.err

### #SBATCH --time=0-14:00:00

### #SBATCH --nodes=1

### #SBATCH --cpus-per-task=5

### module add blast

### blastp \

### -query /data/ficklin\_class/AFS505/Spring2020/example-

### data/all.pep \

### -db /data/ficklin\_class/AFS505/Spring2020/example- data/swissprot \

### -num\_threads 5 \

### -outfmt 6 \

### -evalue 1e-6

module add blast

blastp \

-query /data/ficklin\_class/AFS505/course\_material/data/all.pep \ -db /data/ficklin\_class/AFS505/course\_material/data/swissprot \

-num\_threads 5 \

-outfmt 6 \

-evalue 1e-6

module add blast

blastp \

-query /data/ficklin\_class/AFS505/course\_material/data/all.pep \ -db /data/ficklin\_class/AFS505/course\_material/data/swissprot \ -num\_threads 5 \

-outfmt 6 \

-evalue 1e-6

idev --partition=ficklin\_class --account=ficklin\_class -t 24:00:00

### pairs4trim = Channel.fromFilePairs(params.sample\_glob)

[joshua.freimark@login-p1n01 Project]$ cat Project\_step1.nf

#!/usr/bin/env nextflow

Channel

.fromPath('/data/ficklin\_class/AFS505/course\_material/data/all.pep')

.splitFasta(by: 5000, file:true)

.set { split\_ch }

process blast {

input:

path '/data/ficklin\_class/AFS505/course\_material/data/all.pep' from split\_ch

output:

file 'protein\_match' into match\_ch

"""

blastp \

-query /data/ficklin\_class/AFS505/course\_material/data/all.pep \

-db /data/ficklin\_class/AFS505/course\_material/data/swissprot \

-num\_threads 5 \

-outfmt 6 \

-evalue 1e-6

> protein\_match

"""

}

[joshua.freimark@login-p1n01 Project]$ cat Project\_step1.srun

#!/usr/bin/env nextflow

#SBATCH --partition=ficklin\_class

#SBATCH --account=ficklin\_class

#SBATCH --job-name=biology\_boiii\_blastp

#SBATCH --output=TRIM.out

#SBATCH --error=TRIM.err

#SBATCH --time=0-24:00:00

#SBATCH --nodes=1

#SBATCH --cpus-per-task=5

module add blast

module add nextflow

module add java

nextflow run Step1.nf

[joshua.freimark@login-p1n01 Project]$

ERROR ~ Invalid process definition -- Make sure the process ends with a script wrapped by quote characters @ line 8, column 1.

process blast {

^

1 error

Error

May-06 11:46:04.609 [main] DEBUG n.processor.LocalPollingMonitor - Creating local task monitor for executor 'local' > cpus=24; memory=251.6 GB; capacity=24; pollInterval=100ms; dumpInterval=5m

May-06 11:46:05.061 [main] DEBUG nextflow.processor.TaskProcessor - Creating operator > blast -- maxForks: 24; blocking: false

May-06 11:46:05.136 [main] DEBUG nextflow.Session - Session aborted -- Cause: No such property: protein\_match for class: Script\_13573c9d

May-06 11:46:05.313 [main] DEBUG nextflow.Session - The following nodes are still active:

### Step 2:

Now that you have the BLAST results, you will generate a tab delimited summary file of all the results. The file should have only two columns, the first being the gene name, the second being the total number of alignments matches for that gene. Note: use the gene name NOT the protein name. For the *Oryza sativa* genome, protein names and gene names are identical except protein names have a suffix that includes a period followed by a number. For example, for the protein named “LOC\_Os01g010101.1”, its corresponding gene name will be “LOC\_Os01g010101” (note there is no “.1” at the end). Sort the file such that the gene with highest number of matches occurs at the top of the list.

***Hint:*** If you are using Nextflow you should use the collect operator (<https://www.nextflow.io/docs/latest/operator.html?#collect>) to wait until all BLAST jobs have finished so that you can then perform step2 on all of the results.

Summary of requirements for step 2:

Use whatever method you want to do this step: Python, BASH, Nextflow, command-line tools, etc. Just be sure the to follow these criteria:

* Combine the output from all 10 BLAST jobs into a single file.
* Generate a tab-delimited summary file with two columns: the gene name, the number of matches
* Make sure the tab-delimited file is sorted with the gene having the most matches at the top.

Step 3:

Once your code is completed you should generate a **README.md** file to be include with your project code. The README.md file should describe to a reader what your package does, and how to execute each step. The instructions should be sufficient such that someone not in the class could use your code to execute a split BLAST job. You can assume that the reader knows how to use the UNIX/Linux command-line. The README.md file can contain plain text, or it can use markdown to “prettify” the text. Information about how to use markdown in your REAME.md file can be found here: <https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet>

## Working Together

You are allowed to work together via Slack: ask questions to try to solve problems you may encounter and help others*.* However, this is not a group project and each student must provide their own solution. Please feel free to share bits of code on Slack with which you may need help, but do not share large chunks of code that may give away too much of your solution to others.

## What to Turn in

Please send to Dr. Ficklin via Slack a zipped folder of your solution. It should include all scripts necessary to run the project as well as the README.md file containing the instructions. Alternatively, if you prefer you may use Git to manage your files and send the link to the GitHub project to Dr. Ficklin. The benefit to using Git is that it is easier to manage your submission and the README.md file is automatically rendered on the git repository page! Anyone who was not in Units 2 and 3 can request instructions for using Git. However, Git is not required.

## Grading

Each portion of the project will be weighted in the following way:

Step 1: (60 points)

* Your solution equally divides the input query file into 10 files (10 points)
* Your solution automates submission of BLAST for each chunk on Kamiak (20 points)
* Your solution uses these blast settings (10 points)
  + Query file: /data/ficklin\_class/AFS505/course\_material/data/all.pep
  + Database File: /data/ficklin\_class/AFS505/course\_material/data/swissprot
  + Output Format: Tab-delimited
  + E-value threshold: 1e-6
  + Threads: 5
* The solution must execute properly without any errors (20 points)

Step 2: (30 points)

* Your solution combines the output from all 10 BLAST jobs into a single file (5 points)
* Your solution generates a tab-delimited summary file
  + The file must have two columns (2 points)
  + The gene names must be in the first column (not protein names) (3 points)
  + The list must be sorted such that the gene with the most hits is at the top (5 points)
  + The counts must be correct (5 points)
* Your solution executes without errors (10 points)

Step 3: (10 points)

* Your README.md file must be present with instructions (5 points)
* Your README.md must give sufficient instructions to fully execute your solution (5 points)

### Late Policy

Please note, 5 points will be subtracted for each day late.

### Extra Credit (10 points)

Rather than split the input FASTA file into 10 equally sized files, split it into files no larger than 5000 sequences each. So, for the 66K rice protein example file you would be creating 14 files. Adjust your workflow such that it can accept any number of files rather than just 10 and someone could provide any size input file and the script will work. Update the README.md so that any changes needed by the reader to run the code is reflected.