Searching for UCYN-A in the MVCO dataset from the VAMPS Pipeline

0.1 Getting Started on the Poseidon Cluster

0.1.1 Accessing the Cluster

1. Get a VPN and HPC account at WHOI
2. ssh into the cluster in the terminal

```
ssh msantos@poseidon.whoi.edu
```

0.1.2 Copying Between Local Directory and the Cluster

1. copying from local to cluster

```
scp <path to object of interest> <destination>
scp filename.txt msantos@poseidon.whoi.edu:~/../../scratch/msantos/blast_search/results/
```

2. copying from cluster to local: I navigate to my local directory in the bash terminal. In Windows, this was confusing at first. I was located in root and I didn't know how to access my drive. I had to go up a directory and then go to mnt to find my c: drive. The "." indicates "here".

```
scp msantos@poseidon.whoi.edu:~/../../scratch/msantos/blast_search/results/* .
```

0.1.3 Making a SLURM Script

Slurm is the job manager on Poseidon. All jobs submitted to the cluster need to be handled by slurm. Here is an example slurm script. You can see more here `https://whoi-it.whoi.edu/creating-scripts/`

```
#!/bin/bash
#SBATCH --partition=compute         # Queue selection
#SBATCH --job-name=b_search_job       # Job name
#SBATCH --mail-type=END             # Mail events (BEGIN, END, FAIL, ALL)
#SBATCH --mail-user=msantos@whoi.edu  # Where to send mail
#SBATCH --ntasks=1                  # Run on a single CPU
#SBATCH --mem=5gb                   # Job memory request
#SBATCH --time=24:00:00             # Time limit hrs:min:sec
#SBATCH --output=b_search_job_%j.log  # Standard output/error
pwd; hostname; date
module load bio
module load blast
blastn -db UCYN_A -query data/mvco_blast_bact.fasta -out results/ r results_mvco_seq_18_Mar.out -outfmt 6
date
```

Let's walk through what each of these lines mean:

```
#SBATCH --partition=compute         # Queue selection
```

This is where one can specify which partition to use on the cluster. compute should be the main partition. One can alternatively use scavenger which "scavenges" for spare space. However, if someone has reserved a block of time to run a job on the cluster that interferes with space that you have scavenged, their job will get priority and your job will be cancelled.

0.2 Building a Local BLAST Database

1. First, I downloaded the program NCBI BLAST+. It is stored at `C:/ProgramFiles/NCBI/blast-2.10.0+`. I am currently using version 2.10.0 for Windows 64-bit from this online directory (`ftp://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/LATEST/`). The BLAST+ manual is at (`https://www.ncbi.nlm.nih.gov/books/NBK279668/#usermanual.BLAST_search_strategies`).

2. Next I will build a local BLAST database of sequences from the VAMPS pipeline and align these sequences to my reference sequence.

   (a) makeblastdb is the function for making blast databases.

   ```
   makeblastdb -in Data/UCYN-A_16S_BBig_chloroplast16S.fasta -input_type fasta -dbtype nucl
   -out UCYN_A -title "UCYN_A"
   ```

   I make use of the following arguments:

   ```
   -in (path to reference sequences)
   -input_type (input format)
   -dbtype (database type, in this case nucleotide search_
   -out  (database out name, all database files)
   -title
   ```

3. when you run this command you should get three output files in your current path. An .nin, .nan, and .nsq file.

0.3 Formating Query Files

I used this julia script to convert the sequences identified by the vamps pipeline into a fasta file.

```
## Loading in Data
using Pkg
Pkg.add("DataFrames")
Pkg.add("CSV")
Pkg.add("Bio")
Pkg.add("BioTools")
Pkg.add("FastaIO")

using DataFrames, CSV
using Bio,BioTools,FastaIO

#INPUT
#base_path = "D:/WHOI/V6_V8_Analysis/het_bac/"
base_path = "/dos/WHOI/V6_V8_Analysis/het_bac/"
source = string(base_path, "/data/mvco_seqcounts.tsv")
seqCounts = DataFrame(CSV.File(source; delim = '\t'))

#OUTPUT
fasta_path = string(base_path,"/data/fasta_data/mvco_blast_num_test_run.fasta")

fs = Dict()
for (n, f) in enumerate(seqCounts.Sequence)
   fs["$(n)"] = f
end

FastaWriter(fasta_path) do fw
    for (n,f) in fs
        writeentry(fw, n, f)
    end
end
```

0.4 Running a BLAST Search

The blastn function is for running blast searches. The arguments are:

```
-db (database name)
-query (path to query files)
-out (path to output files)
-outfmt (output format)
-evalue (for finetuning search)
-bitscore (for finetuning search)
```

These are somethings to keep in mind:

1. These are the options for the output format.

```
-outfmt <String>
 alignment view options:
    0 = pairwise,
    1 = query-anchored showing identities,
    2 = query-anchored no identities,
    3 = flat query-anchored, show identities,
    4 = flat query-anchored, no identities,
    5 = XML Blast output,
    6 = tabular,
   7 = tabular with comment lines,
    8 = Text ASN.1,
    9 = Binary ASN.1,
   10 = Comma-separated values,
   11 = BLAST archive format (ASN.1)
```

2. (from metagenomics wiki) The E-value is the number of expected hits of similar quality (score) that could be found just by chance. E-value of 10 means that up to 10 hits can be expected to be found just by chance, given the same size of a random database. E-value can be used as a first quality filter for the BLAST search result, to obtain only results equal to or better than the number given by the -evalue option. Blast results are sorted by E-value by default (best hit in first line).

   ```
   blastn -query genes.ffn -subject genome.fna -evalue 1e-10
   ```

   The smaller the E-value, the better the match.

   ```
   -evalue 1e-50
   ```

   - small E-value: low number of hits, but of high quality
     Blast hits with an E-value smaller than 1e-50 includes database matches of very high quality.

     ```
     -evalue 0.01
     ```

     Blast hits with E-value smaller than 0.01 can still be considered as good hit for homology matches.

     ```
     -evalue 10   (default)
     ```

   - large E-value: *many hits, partly of low quality*
     E-value smaller than 10 will include hits that cannot be considered as significant, but may give an idea of potential relations. The E-value (expectation value) is a corrected bit-score adjusted to the sequence database size. The E-value therefore depends on the size of the used sequence database. Since large databases increase the chance of false positive hits, the E-value corrects for the higher chance. It's a correction for multiple comparisons. This means that a sequence hit would get a better E-value when present in a smaller database. E = $\frac{m*n}{2*bit-score}$ where m = query sequence length and n = total database length (sum of all sequences)

3. (from metagenomics wiki) he higher the bit-score, the better the sequence similarity Bit-score does not depend on database size. The bit-score gives the same value for hits in databases of different sizes and hence can be used for searching in an constantly increasing database.

0.5 Parsing the Output of a BLAST Search

You should get an output file ending with .out. You can read this file as a tab-delimited file. It does not come with header columns, so I had to add them in this julia script.

header	**meaning**
qseqid	query sequence id
sseqid	subjects sequence id
pident	percent identity
length	alignment length
mismatch	number of mismatches
gapopen	number of gap openings
qstart	start of alignment in the query
qend	end of alignment in the query
sstart	start of alignment in subject
send	end of alignment in subject
evalue	expected value
bitscore	bit score

Here is Julia script i used to read in the tabular data.

```
blast_path = string(base_path, "/results/results_mvco_seq_22_mar.out")
blast_result = DataFrame(CSV.File(blast_path,delim ='\t',header = false))
blast_col = ["qseqid", "sseqid" ,"pident" ,"length", "mismatch","gapopen", "qstart", "qend",
 "sstart", "send", "evalue", "bitscore"];
names!(blast_result, Symbol.(blast_col))
```

I then used the findall() command in Julia to filter through the output and filter out matches weren't atleast 97 percent similar to the reference sequences.

```
sim_threshold = 97.0;

ucyn_match = findall(blast_result.sseqid .=="MH807559.1")#UCYNA
big_match = findall(blast_result.sseqid .=="AB847985.2") #Bigelowii chloroplast
sim_match = findall((blast_result.pident .>= sim_threshold))

usim_match = intersect(ucyn_match,sim_match)
bsim_match =  intersect(big_match,sim_match)

usim = blast_result.qseqid[usim_match];
bsim= blast_result.qseqid[bsim_match]; #row indices of matches
```