EASY REFERENCES

Cisco connect:
`uovpn.uoregon.edu/talapas`
Talapas
```
srun --account=bgmp --partition=compute --time=4:00:00 --pty bash
```

EVENTS

7/20/2023
Computer got wiped on 7/20/2023. Was working fine with no sign of issues at the end of class, shut computer on sleep mode as usual. Opened computer at office hours at Pegasus and could not boot OS due to the hard drive being undetected. Opened Bios - showed no hard drive detected. Went home, opened case and physically disconnected/reconnected hard drive. Didn't work. Jason brought a loaner laptop from the BGMP program on 7/21/23. Set up the loaner laptop to finish work (username is ruben). Tried to recover data from the drive but it wasn't possible due to the nature of the error, which was a software partition problem (trying to assign memory sized of 1TB and 2TB despite drive size being 512GB). The hard drive is totally wiped except for some Windows process files. Lab notebook for PS6 and PS8 was deleted, as was my almost completed PS6. Looking to the future, Justin (?) recommended OneDrive but also said DropBox was free to UO students - OneDrive should work with WSL just fine? I'm not a fan of OneDrive, since I've used it in the past and have had files go missing or not update correctly. Will look into getting a DropBox set up on a new computer. I had most of my work backed up on Google Drive, but required me to back up manually since I didn't have automatic backups enabled. The last backup was either Tuesday (7/18) or Wednesday (7/19).

PROJECTS

PS6

Assembling fosmids from goldeye fish (*Hiodon alosoides*) from quality trimmed Illumina PE100 reads (paired-end, 100bp).

Environment: bgmp-velvet
Scripts: cont_tsv_dist.py, resultsmaker.sh, velvetg.srun, velveth.srun
Required modules: matplotlib, bioinfo.py v0.4, python re, python argparse, velvet 1.2.10
Collaborators: Ike, Farris, Eden

Original laboratory notebook section and PS6 deleted with loss of hard drive, redoing PS6 and re-filling out lab notebook as I go

Use this equation for genomic coverage:
```
k-mer coverage = genomic coverage * ((mean of length of reads (bp) - kmer size + 1) / mean of length of reads (bp))
```

Tested dist_tsv.py with my own unit test (Unit_test.fa) and with Farris's unit test (Unit_test_FARRIS.fa), passed both.

Installed matplotlib to base in Talapas so I can run my cont_tsv_dist.py.

Getting parameters for running velveth (assuming k-mer length of 49):
1. Our fosmid library is comprised of 50 fosmids, each (approximately) 40 kb long. How many total nt should be in this fosmid library?
50 fosmids x 40,000bp = 2,000,000bp

2. Coverage is a metric applicable to total fosmid library size just as it is applicable to genome (estimated) size. Think of all of these fosmids combined as the genome we're trying to assemble. Calculate the expected coverage.
>NOTE: Reads are varying lengths due to adapter clipping and quality trimming. You must calculate the total number of NT in the input FASTQ files in order to calculate the expected coverage.

To calculate total NT in the input FASTQ files:
Count the number of characters in the sequence lines (includes newline characters)
```
cat <filename> | grep -A1 "^@HWI" --no-group-separator | grep -v "^@HWI" | wc -m
```
```
fq1: 68868840, fq2: 67942146, fq_unmatched: 62939394
```
Then find the number of lines in the file (number of reads)
```
cat <filename> | grep -A1 "^@HWI" --no-group-separator | grep -v "^@HWI" | wc -l
```

`fq1:858989, fq2: 858989, fq_unmatched: 849803`

Subtract number of lines from the character count to get total nucleotides in the file (to get rid of newline chars from count).

`fq1: 68868840-858989=68009851`

`fq2: 67942146-858989=67083157`

`fq_unmatched: 62939394-849803=62089591`

Add together to get total nucleotide count

`Total NT = 62089591+67083157+68009851 = 197182599`

Expected coverage: C = Total nucleotides from all reads/Genome length

`Expected coverage = 197182599/2000000 = 98.591…`

3. Given the calculated coverage and total fosmid library size, calculate the k-mer coverage.

To get mean read length:

Total length of reads/Number of reads = mean read length

L_{mean} `= 197182599/(858989+858989+849803) = 76.791…`

To get k-mer coverage:

$C_k = C*(L_{mean}-k+1)/L_{mean}$

C_k `=`
`(197182599/2000000)*((197182599/(858989+858989+849803))-49+1)/(197182599/(858989+858989+849803)=36.964…`

Running velveth/velvetg

FASTQ files for fosmids are on Talapas, 2 paired and one unmatched:

`/projects/bgmp/shared/Bi621/800_3_PE5_interleaved.fq_1`
`/projects/bgmp/shared/Bi621/800_3_PE5_interleaved.fq_2`
`/projects/bgmp/shared/Bi621/800_3_PE5_interleaved.fq.unmatched`

Ran velveth with k-mer sizes of 31, 41, and 49 and made the 6 total directories for my velvetg to run on.

K-mer size	Coverage cutoff	Insert length
31	auto	auto
31	20	auto
31	60	auto
31	auto	500
41	auto	auto
49	auto	auto

Made multiple slurm scripts for each so I could get multiple sbatches submitted at the same time. Got an OOM kill error for 49; upped memory from 16GB to 32GB and reran.

For directories for lengths 31, 41, 49.

```
/usr/bin/time -v velveth velveth_${kmer}/ $kmer -fastq -shortPaired -separate $fq1 $fq2 -fastq -short $fqunmatch
```

For further directories for 31:

```
/usr/bin/time -v velveth velveth_${kmer}_500/ $kmer -fastq -shortPaired -separate $fq1 $fq2 -fastq -short $fqunmatch
/usr/bin/time -v velveth velveth_${kmer}_60x/ $kmer -fastq -shortPaired -separate $fq1 $fq2 -fastq -short $fqunmatch
/usr/bin/time -v velveth velveth_${kmer}_20x/ $kmer -fastq -shortPaired -separate $fq1 $fq2 -fastq -short $fqunmatch
```

Ran velvetg with coverage cutoff set at auto and insert length set at auto for the different k-mer lengths. k-mer lengths 31 and 49 originally failed for velvetg, did not create contigs.fa file, but ran with exit status 0. Reran until contigs file was successfully created.

```
/usr/bin/time -v velvetg $d1 -exp_cov auto -cov_cutoff auto -ins_length auto
/usr/bin/time -v velvetg $d2 -exp_cov auto -cov_cutoff auto -ins_length auto
/usr/bin/time -v velvetg $d3 -exp_cov auto -cov_cutoff auto -ins_length auto
```

Ran velveth with k-mer size of 31 to create 3 directories: one for cutoff 20, one for cutoff 60, one for insert length of 500.

```
/usr/bin/time -v velvetg $d4 -exp_cov auto -cov_cutoff auto -ins_length 500
/usr/bin/time -v velvetg $d5 -exp_cov auto -cov_cutoff 20 -ins_length auto
/usr/bin/time -v velvetg $d6 -exp_cov auto -cov_cutoff 60 -ins_length auto
```

Resources summary

velveth typical run
CPU: ~250%
Max resident set size: ~1100000 kb
Elapsed time: ~16 seconds

velvetg typical run
CPU: ~230%
Max resident set size: ~560000 kb
Elapsed time: ~35 seconds

All ran with an exit status of 0 despite issues creating contigs.fa files.

Because of having to rerun velveth/velvetg multiple times (*with the same parameters each time -what?*) to get it to generate contigs.fa files, I want to compare my contigs.fa files to Leslie's provided on Talapas in the following directory:

```
/projects/bgmp/shared/Bi621/VELVET_IS_A_JERK

>> diff -q velveth_31/contigs.fa VELVET_IS_A_JERK/results_31/contigs.fa
Files velveth_31/contigs.fa and VELVET_IS_A_JERK/results_31/contigs.fa differ
>> diff -q velveth_31_20x/contigs.fa VELVET_IS_A_JERK/results_31_20x/contigs.fa
Files velveth_31_20x/contigs.fa and VELVET_IS_A_JERK/results_31_20x/contigs.fa differ
>> diff -q ./velveth_41/contigs.fa ./VELVET_IS_A_JERK/results_41/contigs.fa
Files ./velveth_41/contigs.fa and ./VELVET_IS_A_JERK/results_41/contigs.fa differ
```

Not promising - I don't know why they wouldn't match. I double checked all my parameters and can't see anything wrong, no typos, using the correct input files, etc. All of the output files ran with an exit code of 0. (Saved in velvet_out_error dir). I'm going to use Leslie's contigs.fa files for the analysis and results portion of the assignment. I would normally do further investigation but I am out of time for turning in the assignment and I already know that velvet was having issues.
Ran Leslie's contigs.fa files through my cont_tsv_dist.py as a shell script called resultsmaker.sh

```
./cont_tsv_dist.py -f $f1 -o k31.cov_auto.len_default -d k31.cov_auto.len_default_DIST -k 31
./cont_tsv_dist.py -f $f2 -o k31.cov_20x.len_default -d k31.cov_20x.len_default_DIST -k 31
./cont_tsv_dist.py -f $f3 -o k31.cov_60x.len_default -d k31.cov_60x.len_default_DIST -k 31
./cont_tsv_dist.py -f $f4 -o k31.cov_auto.len_500 -d k31.cov_auto.len_500_DIST -k 31
./cont_tsv_dist.py -f $f5 -o k41.cov_auto.len_default -d k41.cov_auto.len_default_DIST -k 41
./cont_tsv_dist.py -f $f6 -o k49.cov_auto.len_default -d k49.cov_auto.len_default_DIST -k 49
```

I feel like there is a better way to do naming conventions automatically to avoid mis-naming, but I don't know how yet.

Results and analysis:
Each result generates distribution quick stats. I need to concatenate the files into a table for easy reading.

```
cat k*DIST > raw_dist.tsv
sed -n '3~2!p' raw_dist.tsv > Distributions.tsv
```

Best viewed in Excel or something since the long file name wrecks the .tsv in the terminal

Filename	N50	Number of contigs	Max contig length	Mean contig length	Genome length	Mean genomic coverage
/projects/bgmp/rubenl/bioinfo/Bi621/PS/PS6/VELVET_IS_A_JERK/results_31_20x/contigs.fa	3157	1586	11661	820.203657	1300843	88.45457844
/projects/bgmp/rubenl/bioinfo/Bi621/PS/PS6/VELVET_IS_A_JERK/results_31_60x/contigs.fa	3557	479	15443	2453.275574	1175119	78.64005934
/projects/bgmp/rubenl/bioinfo/Bi621/PS/PS	3563	481	10984	2449.690229	1178301	78.47105864

6/VELVET_IS_A_JERK/results_31_500/contigs.fa						
/projects/bgmp/rubenl/bioinfo/Bi621/PS/PS6/VELVET_IS_A_JERK/results_31/contigs.fa	3130	1669	10984	807.8004793	1348219	86.17016583
/projects/bgmp/rubenl/bioinfo/Bi621/PS/PS6/VELVET_IS_A_JERK/results_41/contigs.fa	2243	1302	10546	925.3701997	1204832	72.03253156
/projects/bgmp/rubenl/bioinfo/Bi621/PS/PS6/VELVET_IS_A_JERK/results_49/contigs.fa	1822	1263	10467	944.6389549	1193079	58.36418461

PS7

7-20-2023/7-21-2023

Environment: bgmp-blast
Scripts: pep_parse.py, crossblast_q_Danio_db_Homo.srun, crossblast_q_Homo_db_Danio.srun
Required modules: blast (Protein-Protein BLAST 2.14.0+), bioinfo.py v0.4, python re, python argparse
Collaborators: Farris, Dove, Kyra

Downloaded fasta files from Ensembl, release **109**, onto my PS7 dir in Talapas.
`Danio_rerio.GRCz11.pep.all.fa.gz`
`Homo_sapiens.GRCh38.pep.all.fa.gz`

Downloaded table of Gene Stable IDs, Gene names, and Protein Stable IDs from Biomart to Biomart_Exports/ dir in PS7.
`human_mart_export_7-13-2023.tsv`
`zebrafish_mart_export-7-13-2023.txt`

Wrote pep_parse.py python script to retain the longest protein record, uses one_line_fasta function from bioinfo.py.

To confirm my pep_parse.py is working:

```
cat longest_Danio.fa | grep "^>" | wc -l
```

30,313 longest protein fasta records for zebrafish ✔️
23,512 longest protein fasta records for human ✔️

On Talapas, created conda environment and installed blast

```
conda create -n bgmp-blast blast
```

NCBI BLAST documentation: https://www.ncbi.nlm.nih.gov/books/NBK279690/, https://www.ncbi.nlm.nih.gov/books/NBK279684/ (Appendix with Smith-Waterman explanation)
APCB BLAST walkthrough: https://open.oregonstate.education/computationalbiology/chapter/command-line-blast/

From APCB:
“Running makeblastdb on a FASTA file is fairly simple:
makeblastdb -in <fasta file> -out <database name> -dbtype <type> -title <title> -parse_seqids
where <type> is one of prot or nucl <title> is a human-readable title (enclosed in quotes if necessary). The -parse_seqids flag indicates that the sequence IDs from the FASTA file should be included in the database so that they can be used in outputs as well as by other tools like blastdbcmd (discussed below).”

I made the two databases with bash commands:

```
 makeblastdb -in ./Longest_Protein_FASTAs/longest_Danio.fa -out Danio_DB/
-dbtype prot -parse_seqids
 makeblastdb -in ./Longest_Protein_FASTAs/longest_Homo.fa -out Homo_DB/ -dbtype
prot -parse_seqids
```

Now, cross-blasting Homo sapiens prot vs Danio rerio prot. This is in a slurm script (both .srun files associated with PS8).

```
blastp -query <query fasta> -db <blast database> -evalue 1e-6 -use_sw_tback
-outfmt 6 -out <outfile.tsv> -num_threads <number of threads>
```

I am running with 8 threads and an out format of 6 (standard, tab separated).
BLAST out format options: https://www.metagenomics.wiki/tools/blast/blastn-output-format-6

(See resources summary at bottom)

To generate results:
1. Number of hits in the file:
```
>> wc -l <filename>
1967367 qHomodbDaniocrossblast.tsv
2553560 qDaniodbHomocrossblast.tsv
```

2. 10 hits with the highest bitscores:
```
>> sort -r -n -k12 <filename> | head -10
```
There are 12 columns, the bitscore is the last column (12). Sort by bitscore highest to lowest.
3. All hits with the lowest evalue, sorted by bitscore (highest to lowest), saved into files named `HLE.txt` and `ZLE.txt`
`>> sort -r -k11 -g <filename>` to find the lowest value. e-value is in column 11. Lowest value is 0.
```
gawk '$11==0 {print $0}' $HLE | sort -r -k12 -n >> ${HLE}.tsv
gawk '$11==0 {print $0}' $ZLE | sort -r -k12 -n >> ${ZLE}.tsv
```
4. Bit score seems to be a much more sensitive measurement than e-value, since the same lowest e-value score of 0.0 spans the range of bit scores 30359 to 498 (HLE) and 30322 to 498 (ZLE).

BLAST scoring parameters:
http://mmb.irbbarcelona.org/gitlab/MMBData/mirrorMMB/raw/c4c9fab8837f3b1af6c09b7af55013e65f4358ad/blast-2.2.18/doc/scoring.pdf
Looking at e-values and bitscores: https://www.metagenomics.wiki/tools/blast/evalue
https://www.biostars.org/p/332096/ interesting comments on RBH

OOPS-
I realized that my pep_parse.py code required that the Biomart exports have columns formatted in a certain way (since I split lines by tabs and used position in the line list to get values). The

columns for protein and gene symbol were flipped between the zebrafish export and the human export, and I used the same code on both. I used awk to switch the columns so they were the correct way around and confirmed by eye.

```
awk -F "\t" '{ print $1"\t"$3"\t"$2 }' human_mart_copy > human_mart_copy_2.tsv
```

I think I'm going to have to remake my databases and rerun my BLAST since I don't know how this would affect my BLAST results.

Rerun pep_parse.py

```
./pep_parse.py -f ./Pep_All_Fa/Danio_rerio.GRCz11.pep.all.fa -w
Longest_Danio_7-22.fa -ref
./Biomart_Exports/zebrafish_mart_export-7-13-2023.txt -o oneline_Danio.fa

./pep_parse.py -f ./Pep_All_Fa/Homo_sapiens.GRCh38.pep.all.fa -w
Longest_Homo_7-22.fa -ref ./Biomart_Exports/human_mart_export_7-13-2023.tsv -o
oneline_Homo.fa

diff Longest_Homo_7-22.fa ./Longest_Protein_FASTAs/longest_Homo.fa
```

These are different files. I'm going to use the current one generated with the corrected Biomart file to remake the human database. The zebrafish files were unaffected so the database is correct. Remake human database with:

```
 makeblastdb -in ./Longest_Protein_FASTAs/longest_Homo.fa -out Homo_DB/ -dbtype
prot -parse_seqids
```

Moved all my old BLAST result files to the OLDSTUFF folder.
Now going to rerun my blast slurm scripts.

Looks like all the results I've gotten are the same. Not sure why it worked the first time, unless header information isn't position-based for BLAST? Weird!

<u>Resources summary:</u>
blastp Homo query Danio db
CPU: 297%
Max res set size: 166484kb
Elapsed time: 1:57:23
Ran with exit status 0.

blastp Danio query Homo db
CPU: 268%
Max res set size: 153136kb
Elapsed time: 2:18:10
Ran with exit status 0.

PS8

Original laboratory notebook section deleted with loss of hard drive on 7/20/2023, recreating what I remember here

7/19/2023?

Environment: bgmp_star
Scripts: samparse.py, star_align.srun, star_dre.srun
Required modules: STAR v2.7.10b, samtools 1.9 (using htslib 1.9)
Collaborators: Ike

Referencing these fastq files (unzip before running :
`/projects/bgmp/shared/Bi621/dre_WT_ovar12_R1.qtrim.fq.gz`
`/projects/bgmp/shared/Bi621/dre_WT_ovar12_R2.qtrim.fq.gz`

Downloaded FASTA and GTF files from Ensembl (version 109 - be careful, since has been updated to a new version. You can access the old version by manually typing in the version into the url).

`Danio_rerio.GRCz11.dna.primary_assembly.fa.gz`
`Danio_rerio.GRCz11.109.gtf.gz`

Ran STAR database creation and alignment using `star_dre.srun` and `star_align.srun` and with the Talapas queuing system.
Initially had an OOM kill issue, upped the memory to 32GB and scripts ran fine.

Converted SAM to BAM with bash commands:

1. Convert SAM file to BAM file using “samtools view -S -b <filename.sam> > <fileout.bam>
2. Sort BAM file “samtools sort <filename.bam> -o <fileout.bam>”
3. Index BAM file to make .bai “samtools index <filename>”
4. Extract chromosome 1 “samtools view <filename> 1 > <outfile.sam>

Cout alignments with “wc -l”. 921737 alignments.

Python program for parsing: samparse.py
Filter out secondary alignments first, then count mapped.

Bitwise flag 4 checks if sequence IS mapped:

```
if((flag & 4) != 4):
    mapped = True
```

Bitwise flag 256 checks if sequence IS a secondary alignment:

```
if((flag & 256) != 256):
    mapped = True
```

Worked with Ike on this one. The question of whether discarded secondary alignments should be counted as unmapped came up. Answer: they shouldn't be counted at all in either mapped or unmapped, just skipped.

Resource summary

star_dre (database creation)
CPU: 181%
Max res set size: 28177960kb
Elapsed time: 20:32.07
Exit status 0.

star_align (alignment)
CPU: 512%
Max res set size: 15960648kb
Elapsed time: 4:03.68
Exit status 0.

Demultiplexing: Assignment the First

7-26-2023

We will be de-multiplexing sequences with dual-matched indexes (same index on both ends of the read).
N=unknown and put the read into “unknown” category for the simplest version of quality filtering.
Index hopping has occurred when index reverse complement does not match but is actually another index from the list of provided indexes known to be in the multiplexed samples.

Notes:
With open files as fhread1, fhread2, fhindex1, fhindex2
If startswith (@) in fh:
Read1 = fhread1.readline
Read 2 = fhread2.readline
Index 1 = fhindex1.readline
Index 2 = fhindex2.readline
Compare index to list of indexes
Append to header: “index - RC of index”
Eg, TCTTCGAC-TTCTCGAC (index seq and reverse complement of index seq: easy to look at and see that they match).

Bash commands for initial file exploration:

```
>>zcat <filename> | head -4 | grep -A1 "^@" --no-group-separator |
grep -v "^@" | wc -m
102
```

102 -1(newline character) = 101nt per read.
9-1(newline character)=8nt per index.

Determine the encoding (and Q-score binning):
Illumina 1.8+ uses Phred+33 encoding for quality scores (J scores are present).

Made test files

```
>> zcat <filename> | head -40 > <testfilename>
```

Installed matplotlib to my bgmp_py311 conda environment so I can run matplotlib on it.

First sbatch didn’t work because I didn’t unzip the files. Added gzip open to .py
Second sbatch didn’t work because I was trying to write to the bgmp shared folder. Oops.
Ran a third time and it worked.

<u>Resource summary:</u>
Maximum resident set size (kbytes): 73136

Percent of CPU this job got: 99%
Elapsed (wall clock) time (h:mm:ss or m:ss): 15:31.04
Exit status: 0