NIJ-ARF Metatranscriptomics batch file:

From HudsonAlpha:

Data for flowcell HTNFTDSXY have been posted to your project page on the HudsonAlpha Discovery website at <https://gslweb.discoveryls.com/projects/haib21JMD6421/view_project>.  
  
The files can be downloaded using our wget downloader, available at <https://gslweb.discoveryls.com/information/software/wget_curl_download>  
  
For each flowcell containing your samples, the following files have been posted:  
  
        **files\_HTNFTDSXY.txt** = A list of URLs pointing to your data. This file is what should be passed to the wget downloader.  
        **filenames\_HTNFTDSXY.txt** = This file will help you match the unique HudsonAlpha Discovery ID numbers to file and sample names.  
        **md5s\_HTNFTDSXY.md5** = This is a list of md5 checksums for each file to assist in validation of download integrity.  
        **Demultiplex\_Stats\_HTNFTDSXY.html** = This is generated by Illumina's demultiplexing software and provides measures of yields and data quality for each sample and lane.  
  
Libraries are typically run as large pools over many lanes for optimum quality and performance; therefore you may have several pairs of fastqs for each individual sample. These can be merged using a script that we provide at <https://gslweb.discoveryls.com/information/software/merge_fastqs_by_lib>

**From Hudson Alpha: “we demultiplex based on the dual index 8 bp barcodes and do not trim adapters”, therefore adapters will need to be trimmed.**

Downloading files from HudsonAlpha:

1. Login to HudsonAlpha and enter the project page
2. Scroll down to ‘raw data’. The four flies listed above should be present here
3. Open the files\_xxx.txt in a new window. This does not download the .txt file, only open it in a new window. To save as a .txt, copy the text, open notepad, paste text and save the file in the desired location as files\_xxx.txt
4. Go to: [HudsonAlpha Discovery (discoveryls.com)](https://gslweb.discoveryls.com/information/software/wget_curl_download)
5. Click: Download wget version (Linux) and save as a shell script in the folder you want to download the files too
6. Open the command line (ubuntu) and navigate to the directory where the shell script is saved/where you want to download the files to (to navigate to base PC, type: cd /mnt)
7. Once in the folder with the shell script and files\_xxx.txt, type the following into ubuntu:

bash hadiscovery\_gsl\_wget\_download.sh files\_xxx.txt

This may prompt dialogue to enter your hudsonalpha username and password

1. Files will begin downloading to the selected folder. *Note: this may take a long time (these 50 files took just about 3 days to download via Ubuntu on my computer)*

How files are named by Hudson Alpha: HTNFTDSXY\_s1\_1\_IDT8\_157\_i7-IDT8\_UDI\_157\_i5\_SL467049

HTNFTDSXY = flow cell name

S1 = the flow cell lane (here lane 1)

1 = read number (1 = forward read, 2 = reverse read)

IDT8\_UDI\_157\_i7-IDT8\_UDI\_157\_i5 = barcode info

SL467049 = Sequencing library ID

Transcript analysis in the Wilhelm lab:

The Wilhelm lab has 3 computers with CLC licenses (computers 1, X, and 7). Computers 7 and 8 are newer and have larger processors, meaning they are faster.

**5 Mar 2021:**

Importing files into CLC:

1. First, I need to transfer zipped fastq files (.gz) to the desktop (working on computer #7).
   1. The folder Allison\_Mason was created on computer #7
   2. Memory on the C: drive is low, so only half of the files were imported:

For this round (3.5.2021), 24 files were transferred (corresponding to 12 samples). This includes the forward and reverse read files for time points 0, 12, and 58.

1. Import files into CLC:
   1. Open CLC from desktop
   2. A new folder named ‘Allison’ was made in CLC
   3. Files were imported into CLC:
      1. Hit the ‘import’ button on the top toolbar
      2. Select ‘Illumia’ from the drop down
      3. Too add the files for upload: select ‘add files’, navigate to the folder Allison\_Mason were files were saved
      4. Select files for import and hit save
      5. Files were imported with default settings:

General options: paired reads selected

Paired read information: Paired-end(forward-reverse) selected; minimum distance = 1; maximum distance = 1000

Illuminate options: remove failed reads selected; quality scores: NCBI/Sanger to Illumina Pipeline 1.8 and later

* + 1. Hit ‘Finish’

***NOTE:*** *The HudsonAlpha naming system does not work with this upload. This method is specifically looking fo the naming format: ID\_R1\_001/ ID\_R2\_001. Adding R1/R2 or R1\_001/R2\_001 to the HusonAlpha names will not work. CLC will try and pair the first 2 alphanumeric files instead of the correct pairs. Therefore, files were renamed to the following:*

|  |  |
| --- | --- |
| Hudson Alpha name: | New CLC name: |
| HTNFTDSXY\_s1\_1\_IDT8\_UDI\_157\_i7-IDT8\_UDI\_157\_i5\_SL467049.fastq.gz | NIJ\_0\_CON\_R1\_001 |
| HTNFTDSXY\_s1\_2\_IDT8\_UDI\_157\_i7-IDT8\_UDI\_157\_i5\_SL467049.fastq.gz | NIJ\_0\_CON\_R2\_001 |
| HTNFTDSXY\_s1\_1\_IDT8\_UDI\_158\_i7-IDT8\_UDI\_158\_i5\_SL467050.fastq.gz | NIJ\_0\_SP1\_R1\_002 |
| HTNFTDSXY\_s1\_2\_IDT8\_UDI\_158\_i7-IDT8\_UDI\_158\_i5\_SL467050.fastq.gz | NIJ\_0\_SP1\_R2\_002 |
| HTNFTDSXY\_s1\_1\_IDT8\_UDI\_159\_i7-IDT8\_UDI\_159\_i5\_SL467051.fastq.gz | NIJ\_0\_SP2\_R1\_003 |
| HTNFTDSXY\_s1\_2\_IDT8\_UDI\_159\_i7-IDT8\_UDI\_159\_i5\_SL467051.fastq.gz | NIJ\_0\_SP1\_R2\_003 |
| HTNFTDSXY\_s1\_1\_IDT8\_UDI\_160\_i7-IDT8\_UDI\_160\_i5\_SL467052.fastq.gz | NIJ\_0\_SP3\_R1\_004 |
| HTNFTDSXY\_s1\_2\_IDT8\_UDI\_160\_i7-IDT8\_UDI\_160\_i5\_SL467052.fastq.gz | NIJ\_0\_SP3\_R2\_004 |
| HTNFTDSXY\_s1\_1\_IDT8\_UDI\_161\_i7-IDT8\_UDI\_161\_i5\_SL467053.fastq.gz | NIJ\_12\_CON\_R1\_005 |
| HTNFTDSXY\_s1\_2\_IDT8\_UDI\_161\_i7-IDT8\_UDI\_161\_i5\_SL467053.fastq.gz | NIJ\_12\_CON\_R2\_005 |
| HTNFTDSXY\_s1\_1\_IDT8\_UDI\_162\_i7-IDT8\_UDI\_162\_i5\_SL467054.fastq.gz | NIJ\_12\_SP1\_R1\_006 |
| HTNFTDSXY\_s1\_2\_IDT8\_UDI\_162\_i7-IDT8\_UDI\_162\_i5\_SL467054.fastq.gz | NIJ\_12\_SP1\_R2\_006 |
| HTNFTDSXY\_s1\_1\_IDT8\_UDI\_163\_i7-IDT8\_UDI\_163\_i5\_SL467055.fastq.gz | NIJ\_12\_SP2\_R1\_007 |
| HTNFTDSXY\_s1\_2\_IDT8\_UDI\_163\_i7-IDT8\_UDI\_163\_i5\_SL467055.fastq.gz | NIJ\_12\_SP2\_R2\_007 |
| HTNFTDSXY\_s1\_1\_IDT8\_UDI\_164\_i7-IDT8\_UDI\_164\_i5\_SL467056.fastq.gz | NIJ\_12\_203\_R1\_008 |
| HTNFTDSXY\_s1\_2\_IDT8\_UDI\_164\_i7-IDT8\_UDI\_164\_i5\_SL467056.fastq.gz | NIJ\_12\_SP3\_R2\_008 |
| HTNFTDSXY\_s1\_1\_IDT8\_UDI\_165\_i7-IDT8\_UDI\_165\_i5\_SL467057.fastq.gz | NIJ\_58\_CON\_R1\_009 |
| HTNFTDSXY\_s1\_2\_IDT8\_UDI\_165\_i7-IDT8\_UDI\_165\_i5\_SL467057.fastq.gz | NIJ\_58\_CON\_R2\_009 |
| HTNFTDSXY\_s1\_1\_IDT8\_UDI\_166\_i7-IDT8\_UDI\_166\_i5\_SL467058.fastq.gz | NIJ\_58\_SP1\_R1\_010 |
| HTNFTDSXY\_s1\_2\_IDT8\_UDI\_166\_i7-IDT8\_UDI\_166\_i5\_SL467058.fastq.gz | NIJ\_58\_SP1\_R2\_010 |
| HTNFTDSXY\_s1\_1\_IDT8\_UDI\_167\_i7-IDT8\_UDI\_167\_i5\_SL467059.fastq.gz | NIJ\_58\_SP2\_R1\_011 |
| HTNFTDSXY\_s1\_2\_IDT8\_UDI\_167\_i7-IDT8\_UDI\_167\_i5\_SL467059.fastq.gz | NIJ\_58\_SP2\_R2\_011 |
| HTNFTDSXY\_s1\_1\_IDT8\_UDI\_168\_i7-IDT8\_UDI\_168\_i5\_SL467060.fastq.gz | NIJ\_58\_SP3\_R1\_012 |
| HTNFTDSXY\_s1\_2\_IDT8\_UDI\_168\_i7-IDT8\_UDI\_168\_i5\_SL467060.fastq.gz | NIJ\_58\_SP3\_R2\_012 |
| HTNFTDSXY\_s1\_1\_IDT8\_UDI\_169\_i7-IDT8\_UDI\_169\_i5\_SL467061.fastq.gz | NIJ\_86\_CON\_R1\_013 |
| HTNFTDSXY\_s1\_2\_IDT8\_UDI\_169\_i7-IDT8\_UDI\_169\_i5\_SL467061.fastq.gz | NIJ\_86\_CON\_R2\_013 |
| HTNFTDSXY\_s1\_1\_IDT8\_UDI\_170\_i7-IDT8\_UDI\_170\_i5\_SL467062.fastq.gz | NIJ\_86\_SP1\_R1\_014 |
| HTNFTDSXY\_s1\_2\_IDT8\_UDI\_170\_i7-IDT8\_UDI\_170\_i5\_SL467062.fastq.gz | NIJ\_86\_SP1\_R2\_014 |
| HTNFTDSXY\_s1\_1\_IDT8\_UDI\_171\_i7-IDT8\_UDI\_171\_i5\_SL467063.fastq.gz | NIJ\_86\_SP2\_R1\_015 |
| HTNFTDSXY\_s1\_2\_IDT8\_UDI\_171\_i7-IDT8\_UDI\_171\_i5\_SL467063.fastq.gz | NIJ\_86\_SP2\_R2\_015 |
| HTNFTDSXY\_s1\_1\_IDT8\_UDI\_172\_i7-IDT8\_UDI\_172\_i5\_SL467064.fastq.gz | NIJ\_86\_SP3\_R1\_016 |
| HTNFTDSXY\_s1\_2\_IDT8\_UDI\_172\_i7-IDT8\_UDI\_172\_i5\_SL467064.fastq.gz | NIJ\_86\_SP3\_R2\_016 |
| HTNFTDSXY\_s1\_1\_IDT8\_UDI\_173\_i7-IDT8\_UDI\_173\_i5\_SL467065.fastq.gz | NIJ\_186\_CON\_R1\_017 |
| HTNFTDSXY\_s1\_2\_IDT8\_UDI\_173\_i7-IDT8\_UDI\_173\_i5\_SL467065.fastq.gz | NIJ\_186\_CON\_R2\_017 |
| HTNFTDSXY\_s1\_1\_IDT8\_UDI\_174\_i7-IDT8\_UDI\_174\_i5\_SL467066.fastq.gz | NIJ\_186\_SP1\_R1\_018 |
| HTNFTDSXY\_s1\_2\_IDT8\_UDI\_174\_i7-IDT8\_UDI\_174\_i5\_SL467066.fastq.gz | NIJ\_186\_SP1\_R2\_018 |
| HTNFTDSXY\_s1\_1\_IDT8\_UDI\_175\_i7-IDT8\_UDI\_175\_i5\_SL467067.fastq.gz | NIJ\_168\_SP2\_R1\_019 |
| HTNFTDSXY\_s1\_2\_IDT8\_UDI\_175\_i7-IDT8\_UDI\_175\_i5\_SL467067.fastq.gz | NIJ\_168\_SP2\_R2\_019 |
| HTNFTDSXY\_s1\_1\_IDT8\_UDI\_176\_i7-IDT8\_UDI\_176\_i5\_SL467068.fastq.gz | NIJ\_168\_SP3\_R1\_020 |
| HTNFTDSXY\_s1\_2\_IDT8\_UDI\_176\_i7-IDT8\_UDI\_176\_i5\_SL467068.fastq.gz | NIJ\_168\_SP3\_R2\_020 |
| HTNFTDSXY\_s1\_1\_IDT8\_UDI\_177\_i7-IDT8\_UDI\_177\_i5\_SL467069.fastq.gz | NIJ\_378\_CON\_R2\_021 |
| HTNFTDSXY\_s1\_2\_IDT8\_UDI\_177\_i7-IDT8\_UDI\_177\_i5\_SL467069.fastq.gz | NIJ\_378\_CON\_R2\_021 |
| HTNFTDSXY\_s1\_1\_IDT8\_UDI\_178\_i7-IDT8\_UDI\_178\_i5\_SL467070.fastq.gz | NIJ\_378\_SP1\_R1\_022 |
| HTNFTDSXY\_s1\_2\_IDT8\_UDI\_178\_i7-IDT8\_UDI\_178\_i5\_SL467070.fastq.gz | NIJ\_378\_SP2\_R2\_022 |
| HTNFTDSXY\_s1\_1\_IDT8\_UDI\_179\_i7-IDT8\_UDI\_179\_i5\_SL467071.fastq.gz | NIJ\_378\_SP2\_R1\_023 |
| HTNFTDSXY\_s1\_2\_IDT8\_UDI\_179\_i7-IDT8\_UDI\_179\_i5\_SL467071.fastq.gz | NIJ\_378\_SP2\_R2\_023 |
| HTNFTDSXY\_s1\_1\_IDT8\_UDI\_180\_i7-IDT8\_UDI\_180\_i5\_SL467072.fastq.gz | NIJ\_378\_SP3\_R1\_024 |
| HTNFTDSXY\_s1\_2\_IDT8\_UDI\_180\_i7-IDT8\_UDI\_180\_i5\_SL467072.fastq.gz | NIJ\_378\_SP3\_R2\_024 |

* 1. File import 3.5.2021:
     1. 48 files were imported in 2 batches (12 samples each) into CLC following the steps outlined above
     2. ***Note:*** *importing and pairing 12 samples took ~1 hr*
     3. Import log 1:

|  |  |  |
| --- | --- | --- |
| Merged file pairs | Paired file (saved to CLC) | Reads imported |
| NIJ\_0\_CON\_R1\_001 ; NIJ\_0\_CON\_R2\_001 | NIJ\_0\_CON\_R1\_001 (paired) | 63,614,592 |
| NIJ\_0\_SP1\_R1\_002 ;  NIJ\_0\_SP1\_R2\_002 | NIJ\_0\_SP1\_R1\_002 (paired) | 91,353,623 |
| NIJ\_0\_SP2\_R1\_003 ;  NIJ\_0\_SP2\_R2\_003 | NIJ\_0\_SP2\_R1\_003 (paired) | 101,006,659 |
| NIJ\_0\_SP3\_R1\_004 ;  NIJ\_0\_SP3\_R2\_005 | NIJ\_0\_SP3\_R1\_004 (paired) | 90,940,450 |
| NIJ\_12\_CON\_R1\_005 ;  NIJ\_12\_CON\_R2\_005 | NIJ\_12\_CON\_005 (paired) | 95,195,127 |
| NIJ\_12\_SP1\_R1\_006 ;  NIJ\_12\_SP1\_R2\_006 | NIJ\_12\_SP1\_006 (paired) | 97,865,170 |
| NIJ\_12\_SP2\_R1\_007 ;  NIJ\_12\_SP2\_R2\_007 | NIJ\_12\_SP2\_007 (paired) | 132,290,589 |
| NIJ\_12\_SP3\_R1\_008 ;  NIJ\_12\_SP3\_R2\_008 | NIJ\_12\_SP3\_008 (paired) | 107,436,059 |
| NIJ\_58\_CON\_R1\_009 ;  NIJ\_58\_CON\_R2\_009 | NIJ\_58\_CON\_009 (paired) | 115,123,179 |
| NIJ\_58\_SP1\_R1\_010 ;  NIJ\_58\_SP1\_R2\_010 | NIJ\_58\_SP1\_010 (paired) | 109,617,223 |
| NIJ\_58\_SP2\_R1\_011 ;  NIJ\_58\_SP2\_R2\_011 | NIJ\_58\_SP2\_011 (paired) | 121,032,180 |
| NIJ\_58\_SP3\_R1\_012 ;  NIJ\_58\_SP3\_R2\_012 | NIJ\_58\_SP3\_012 (paired) | 118,617,587 |

* + 1. Import log 2:

|  |  |  |
| --- | --- | --- |
| Merged file pairs | Paired file (saved to CLC) | Reads imported |
| NIJ\_168\_SP3\_R1\_020 ;  NIJ\_168\_SP3\_R2\_020 | NIJ\_168\_SP3\_R2\_020 (paired) | 118,157,609 |
| NIJ\_168\_CON\_R1\_017 ;  NIJ\_168\_CON\_R2\_017 | NIJ\_168\_CON\_R1\_017 (paired) | 128,964,387 |
| NIJ\_168\_SP1\_R1\_018 ;  NIJ\_168\_SP1\_R2\_018 | NIJ\_167\_SP1\_018 (paired) | 114,895,508 |
| NIJ\_168\_SP2\_R1\_019 ;  NIJ\_168\_SP2\_R2\_019 | NIJ\_168\_SP2\_R1\_019 (paired) | 99,627,508 |
| NIJ\_378\_CON\_R1\_021 ;  NIJ\_378\_CON\_R2\_021 | NIJ\_378\_CON\_R1\_021 (paired) | 115,709,432 |
| NIJ\_378\_SP1\_R1\_022 ;  NIJ\_378\_SP1\_R2\_022 | NIJ\_378\_SP1\_022 (paired) | 120,555,590 |
| NIJ\_378\_SP2\_R1\_023 ;  NIJ\_378\_SP2\_R2\_023 | NIJ\_378\_SP2\_R2\_023 (paired) | 113,572,964 |
| NIJ\_378\_SP3\_R1\_024 ;  NIJ\_378\_SP3\_R2\_024 | NIJ\_378\_SP3\_024 (paired) | 113,704,922 |
| NIJ\_86\_CON\_R1\_013 ;  NIJ\_86\_CON\_R2\_013 | NIJ\_86\_CON\_R1\_013 (paired) | 51,892,500 |
| NIJ\_86\_SP1\_R1\_014 ;  NIJ\_86\_SP2\_R2\_014 | NIJ\_86\_SP1\_R1\_014 (paired) | 103,783,522 |
| NIJ\_86\_SP2\_R1\_015 ;  NIJ\_86\_SP2\_R2\_015 | NIJ\_86\_SP2\_R1\_015 (paired) | 104,722,146 |
| NIJ\_86\_SP3\_R1\_016 ;  NIJ\_86\_SP3\_R2\_016 | NIJ\_86\_SP3\_R1\_016 (paired) | 107,059,839 |

From CLC manual:

Both reads of a pair will be stored in the same sequence list. Forward and reverse reads alternate so the first read is forward and the second read is the mate reverse read. The orientation of reds can be changed after import by opening the Element information view and clicking ‘edit’ next to paired status.

Paired end data can be checked by looking at the first few sequence names in the imported data object. The first 2 sequences should have the same name, except for a 1 or 2 somewhere in the read name line.

*Do I need to merge overlapping pairs prior to trimming? How do I get CLC to automatically detect adapters/indices?*

Trimming reads:

1. Click ‘Toolbox’ in the top bar —> click ‘Prepare Sequencing Data’ —> click ‘Trim Reads…’
2. Select all paired files from their location in the CLC directory and hit ‘next’

***Note:*** *if adding more than one file select ‘batch’ in the bottom left corner to tell CLC that each file is separate*

1. Quality trimming settings:
   1. Select trim using quality scores (Limit default is 0.05)
   2. Select trim ambiguous nucleotides (Maximum number of ambiguities default is 2)
2. Adapter trimming settings:
   1. Select automatic read-through adapter trimming
3. Homopolymer trimming settings:
   1. Select none
4. Sequence filtering settings:
   1. This is where you can trim a set number of bases from each end
5. Result handling:
   1. Save the output and log to a new folder named ‘trimmed reads’

**Mar 8 2021:**

Attempting read trimming on 1 file (NIJ\_0\_CON\_R1-001 (paired))

The trimming reads protocol outlined above was used with the following settings:

Quality: Trim QC limit = 0.2; max ambit = 2; Adapter; select automatic read-through adapter trimming; Homoplymer trimming: select none; sequence filtering:

***Note****: 1 file took ~3:30 min to complete*

I started trimming all of the files as a batch. This took ~1:30-2 hours.

Trimming did not seem to alter the mean read lengths very much, this makes me wonder if adapters were indeed removed in the trimming. I will work with Naomi to run FastQC on reads before and after trimming to see if adapters are present. This should appear in the FastQC report as commonly repeated sequences.

**Mar 11 2021:**

Goal: check before and after trimming files to see if adapters were remove. If they were not, strategize how to remove them (likely providing an adapter trim list).

First attempt: CLC has a version of FastQC. To implement this:

1. Navigate to toolbox
2. Select create sample report
3. Select the files you want reports for
   1. This will only allow reports for trimmed files
   2. If running more than one at a time, select ‘batch’ in the bottom left corner
4. Select what you want to include in the report
5. Finish

This did not give us what I wanted (a report with quality scores and detection of commonly recurring sequences, aka adapters. Therefore we will try and run FastQC in KBase

**Mar 30 – Apr 26 2021:**

After a 2 week break to start TOX mesocosm experiments, 2 weeks was spent importing all raw files into KBase and then uploading them to the staging area. This took forever….. next time do not use KBase, learn command line for FastQC and Trimmomatic. After finally uploading all files as paired library files into KBase, each file was individual run through Trimmomatic using the settings described below. FastQC was checked on a random subset of files, and FastQC was run on pre- and post- trimmed files for sample 001 (NIJ\_0\_CON\_R1\_001)

Uploading in KBase:

Importing files:

1. Login to KBase
2. Create a new narrative called ‘NIJ-ARF Metatrans’
3. Click ‘Add data’ in the top left-hand workspace
4. Select files and upload
   1. ***Note:*** *2 .gz files took ~10 minutes to upload*
5. Click on the arrow to the right of the uploaded file, select fastq (Do this for both files) and click the downward facing arrow
6. Import files from data to staging area:
   1. Select the R1 file as the forward file
   2. Select the R2 files as the reverse file
   3. Unselect: single genome
   4. Hit RUN
   5. ***Note****: 2 .gz files took ~ 5 min to import*

**Mar 12:** Only 2 files were uploaded to KBase: the untrimmed R1 and R2 files for NIJ\_CON\_0\_001 sample

***Note:*** only way to import files > 3.5 GB, you have to go through Globus; however, globus is not connecting to KBase shared drive. Logging in and back out did not help, nor did entering my drive as the location (/amason30/). Sent a ticket to KBase help and received he following instructions:

A couple of quick troubleshooting tips:

* First, try to manually insert your KBase username as the location in the empty field in that screenshot underneath "KBase Bulk Share" (i.e. /amason30/
* Second, confirm that your linked sign-in for Globus matches the account that's signed in. Check your linked sign in accounts tab at <https://narrative.kbase.us/#/auth2/account> and make sure that matches the Globus account.
* Third, do you have multiple Globus and/or KBase accounts? If so, try to sign out of all the accounts except for the KBase account you want to use and the linked Globus account.
* Finally, if those don't work, follow [this link](https://kbase.us/services/staging_service/add-acl) and paste the resulting text into this ticket.

**Apr 8:** Went to try the troubleshoot tips from the help desk. Upon signing back in, globus connected to the drive. I didn’t try these troubleshoot tips, so Im not sure what worked. Its possible that signing in this time linked the KBase and Globus accounts this time. Started import of remaining 46 files.

***Note****:* 46 .gz files took ~3.5-4 days to import

**Apr 13-Apr 21**: Each sample (002-024) was individually uploaded to the staging area using the forward and reverse files are imports, resulting in a paired library file for each sample.

***Note:*** this took forever, why is KBase like this????

**Apr 21-Apr 27:** ran Trimmomatic on files 002-024

Trimmomatic settings in KBase:

Input: paired reads library

Enable adapter clipping:

Adapters = TruSeq3-PE-2

Seed mismatches = 2

Palindrome clip threshold = 30

Simple clip threshold = 10

Sliding window options:

Sliding window size = 4

Sliding window minimum quality = 15

Output name: name\_paired\_trimmomatic.fastq

**Apr 26**: In depth review of FastQC modules, what they say and if errors are reflective of issues or other reasons.

**FastQC report – in depth description of each module using NIJ\_0\_CON\_R1\_001\_paired\_trimmomatic-test\_paired\_85638\_8\_1.fastq FastQC report:**

Basic stats:

~5% of reads were removed during trimming.

Length = 36-101

%GC = 57

Per base sequence quality: shows an overview of the range of quality values across all bases ([Per Base Sequence Quality (babraham.ac.uk)](https://www.bioinformatics.babraham.ac.uk/projects/fastqc/Help/3%20Analysis%20Modules/2%20Per%20Base%20Sequence%20Quality.html))

Quality looks good, but I noted that no yellow boxplots were visible (this is what is typically seen in this section, however it appears this is nothing to worry about – “*Probably nothing to worry about. As technology has matured, the Q-scores for good libraries can be uniformly high across the cycles (leading to no visible yellow boxes).*” ([FastQC: box plots without boxes? (biostars.org)](https://www.biostars.org/p/208614/) )

Per sequence quality scores: Looks to see if a subset of sequences display universally low quality. ([Per Sequence Quality Scores (babraham.ac.uk)](https://www.bioinformatics.babraham.ac.uk/projects/fastqc/Help/3%20Analysis%20Modules/3%20Per%20Sequence%20Quality%20Scores.html))

Read quality looks good. Average quality per read ~36

***Note:*** warnings here typically indicate over all poor quality for the run, possibly due issues with the flow cell itself. As we did not see this in the pre- or post- trimmed reads, that would suggest the run looks ok.

Per base sequence content: Plots the proportion of each base position for which of the 4 DNA based has been called. ([Per Base Sequence Content (babraham.ac.uk)](https://www.bioinformatics.babraham.ac.uk/projects/fastqc/Help/3%20Analysis%20Modules/4%20Per%20Base%20Sequence%20Content.html))

We have a failure for this module, and it appears the issue is contained to the first ~10 bases. Important to note that biostars indicates: “*some types of library will always produce biased sequence composition, normally at the start of the read. Libraries produced by priming using random hexamers (including nearly all RNA-seq libraries) and those which were fragmented using transposases inherit an intrinsic bias in the positions at which reads start. This bias does not concern and absolute sequence, but instead provides enrichment of a number of different K-mers at the 5’ end of the reads. Whilst this is a true technical bias, it isn’t something which can be corrected and in most cases doesn’t seem to adversely affect the downstream analysis. It will produce and waring or failure in this module*” Since these are RNA libraries produced by random priming, this failure is nothing to worry about.

Per sequence GC content: measures GC content across the whole length of each sequence in a file and compares it to a modelled normal distribution of GC content. ([Per Sequence GC Content (babraham.ac.uk)](https://www.bioinformatics.babraham.ac.uk/projects/fastqc/Help/3%20Analysis%20Modules/5%20Per%20Sequence%20GC%20Content.html)). Sharp peaks usually indicate overrepresented sequences, while broad peaks may represent contamination ([Quality control: Assessing FASTQC results | Introduction to RNA-Seq using high-performance computing - ARCHIVED (hbctraining.github.io)](https://hbctraining.github.io/Intro-to-rnaseq-hpc-salmon/lessons/qc_fastqc_assessment.html)). Since this is a native microbial community contamination with another organism is hard to tell and most of these GC waring/fails (see all sample summary file) display sharp ridges.

Trimmed reads look good.

***Note:*** waring or failure here can indicate contamination or bias in a subset of reads.

Per base N content: plots the percentage of base calls at each position for which an N was called. ([Per Base N Content (babraham.ac.uk)](https://www.bioinformatics.babraham.ac.uk/projects/fastqc/Help/3%20Analysis%20Modules/6%20Per%20Base%20N%20Content.html))

Trimmed read look good.

Sequence length distribution: graph showing the distribution of fragment sizes in the file which was analyzed ([Sequence Length Distribution (babraham.ac.uk)](https://www.bioinformatics.babraham.ac.uk/projects/fastqc/Help/3%20Analysis%20Modules/7%20Sequence%20Length%20Distribution.html))

Warning: majority of read are ~100 bp but are as low as 35. “*This module will raise and error if all sequences are not the same length.”* “*For some sequencing platforms it is entirely normal to have different read lengths so warnings can be ignored*.”

***Note:*** after trimming sequencing length distribution went from good to a warning, however this makes sense as we ran the reads through Trimmomatic which will remove certain reads and alter the distribution. This was confirmed when researching on the web ([Sequence Length Distribution after trimmomatic (biostars.org)](https://www.biostars.org/p/321785/)). No need to worry about this error, as raw reads indicate a good run.

Sequence duplication levels: counts the degree of duplication for every sequence in a library can plots the relative number of sequences with different degrees of duplication. Works under the assumption that in a diverse library, most sequences will occur only once in the final set. Low duplication level suggest high coverage of target sequence, but high duplication may indicate enrichment boas such as PCR over amplification. ([Duplicate Sequences (babraham.ac.uk)](https://www.bioinformatics.babraham.ac.uk/projects/fastqc/Help/3%20Analysis%20Modules/8%20Duplicate%20Sequences.html))

Failure: 34.56% seq remain if deduplicated; ~15% >10 duplication level, ~15% >100 duplication level; “*Low complexity contaminants or enrichment of subsets will cause peaks in the right side of the plot.*” “*This module will raise a failure in non-unique sequences make up more than 50% of the total.*”

Common reasons for errors:

1. Technical duplicates arising from PCR artefacts; 2) biological duplicates which are natural collisions where different copies of the same sequence are randomly selected.

Some library types will tend to naturally over-sequence parts of the library and therefore generate duplication leading to warnings or failures. For example in RNA-seq libraries, sequences of different transcripts will be present at wildly different levels in the starting population. To observed signals from lowly expressed transcripts, it is common to drastically over-sequence highly expressed transcripts leading to a large set of duplicates.

As this is an RNA-seq library, this warning is to be expected and should not be treated as an issue.

Overrepresented sequences: lists all sequences that make up more than 0.1% of the total library. ([Overrepresented Sequences (babraham.ac.uk)](https://www.bioinformatics.babraham.ac.uk/projects/fastqc/Help/3%20Analysis%20Modules/9%20Overrepresented%20Sequences.html))

Warning: 2 sequences found at >0.1%; “*this module is often triggered in small RNA libraries where sequences are not subject to random fragmentation or is naturally present as a significant proportion of the library.*”

| **Sequence** | **Count** | **Percentage** | **Possible Source** |
| --- | --- | --- | --- |
| GTTGGATTGTCGGCCCAAAGCCGCTTGACCAGTGAGCTATTACGCTTTCT | 133488 | 0.22193278403328884 | No Hit |
| GTCTGAACATACTTAGCCTTACGAGGTGGTCCTCGCAGATTCACGCCGGT | 96518 | 0.16046767087172611 | No Hit |

BLAST searched these 2 sequences:

1: hits to uncultured genome sequences and 23S rRNA gene, indicating remaining rRNA

2: hits to multiple uncultured bacterium clone 23S rRNA gene, indicating remaining rRNA

Adapter Content: shows a cumulative percentage of the proportion of the library which has detected adapter sequences at each position. ([Adapter Content (babraham.ac.uk)](https://www.bioinformatics.babraham.ac.uk/projects/fastqc/Help/3%20Analysis%20Modules/10%20Adapter%20Content.html))

Good to go.

Kmer Content: measures the number of each 7-mer at each position and then uses a binomial test to look for deviation from even coverage at all positions. This module assumes that any small fragment sequence should not have positional bias within a diverse library. ([Kmer Content (babraham.ac.uk)](https://www.bioinformatics.babraham.ac.uk/projects/fastqc/Help/3%20Analysis%20Modules/11%20Kmer%20Content.html))

Failure: uneven in first 10 base-pairs; likely due random priming during library construction. “*Libraries which derive from random priming will nearly always show Kmer bias at the start of the library due to an incomplete sampling of the possible random primers*.”

**27 Apr 20201:**

All files have now been trimmed. Today, I will run FastQC on all files, take a look and export the trimmed files for the next step.

**28 Apr 2021:**

FastQC for 007-024 did not run, so these were rerun today. All FastQC reports were annotated into a summary and saved: "C:\Users\Allison\OneDrive - University of Tennessee\DeBruyn Lab\NIJ ARF Metatranscriptomes\Trimmed FastQC\NIJ-ARF\_metaT\_fastqc\_all samples\_summary.docx". All files look ok, GC waring/failure is attributed to overrepresented sequences/highly expressed gene(s).

Next steps: export all trimmed files.

**4-6 May 2021:**

Start running SortMeRNA on the paired, trimmed FASTQ files, this was completed in the Wilhelm lab (Computer #4).

SortmeRNA requires specific versions of python, so this computer has a separate environment (this is specific to computer 4)

***Note:*** (- - h will show all the parameters possible)

***Note:*** top (will show you all of the commands that are running and how many CPUs that it is using)

All of the following can be found in SortmeRNA.txt in the NIJ-ARF-Metatranscriptomes folder. Use this file to start sortmerna.

1. Open command line
2. Navigate to Allison folder on Desktop
3. Type: source activate sortmerna.v4 (computer will change from base to sortmerna)
4. Type: mkdir SortmeRNA
5. Type:
   1. Sortmerna -ref (these are the reference files sortmerna will use; it is in FASTQ format; these are sortmerna base databases)
   2. -reads ‘file path for input file’
   3. -workdir (where the intermediate and aligned reads are going)
   4. -fastx (will output ribosomally depleted reads in fastQ)
   5. -other (we want the unaligned reads, which will go into the current folder)
   6. - - threads (tells sortmerna how many computer treads to use)

Files 004, 005, and 006 all were killed in the middle of alignment on computer 4. No error was given and there are no output files in the ‘out’ folder. Talking with Naomi, we think this is an memory issue on the computer. Our files are rather large (~47 GB) and computer 4 is low on memory and slower than the others. We decided to try a file on computer 8, as it is not open and has more available memory and CPUs. This run was started on 6 May.

Test run update: The test run was successful (file 004 was used), however I noticed that there was no \_notaligned file in the ‘out’ folder. Looking at the code there were no spaces between the workdir file name and -fastx -other, so these were not run. This file will need to be re-run.

**10 May 2021:**

Goals for week of 10 May: Continue sortmerna on computer 8.

When running 005 with - - threads 28, I got a segmentation error during the reports portion of the run. This led to no aligned fastq and 28 \_notaligned fastq files. Looking into this further, it may be an issue with running on multiple threads and this version (4.3) of sortmerna ([https://github.com/bio core/sortmerna/issues/288](https://github.com/bio%20core/sortmerna/issues/288)). There is a pre-release version of sortmerna that may elevate this, but I am not sure how to download it and this is hopefully a case by case issue. I will run 4 samples (006, 007, 008, and 009) using - - threads 6 and see what happens.

**11 May 2021:**

Good news! It seems like samples 006-009 successfully ran and provided \_notaligned files. These were copied to the TOX drive along with the .log files and the rest were deleted to make space for running the remaining files. Toady I started 8 runs (004, 010-016) without the - - threads prompt (default will use threads = 2).

**12 May 2021:**

Using only 2 threads, only 2 files have finished running by 11 am (013 and 015). The rest appear to be on the last alignment. Files 017 and 018 were started.

**27 May 2021:**

We learned that this version of sortmerna is outputting corrupt fastq files; potentially something to do with formatting interleaved files. When repair.sh from bbmap is run, an error is thrown ☹. This issue is not specific to computer 8 or our files. Brittany (computer 1) and Kristen (computer 7) are also experiencing this issue, leading us to believe it is something to do with this version of sortmerna. We don’t think it is a file issue as we used different sequencing services and data preprocessing programs. We are not sure exactly what is causing the problem and it will take a long time to interact with sortmerna help boards, so Naomi will show me how JGI typically removes rRNA with bbmap.

**28 May 2021:**

Goals: check all sortmerna files for corruption and start running bbmap on all files.

A preliminary check of a few sortmerna output files suggests our files are indeed corrupt, we will shift to bbmap instead.

To run bbmap, see bbmap script guide.txt file in the new Elements hard drive (this is the new NIJ-ARF metaT only 5T hard drive)

**3-4 Jun 2021:**

On these days, all samples were run through bbmap to remove rrna from reads files. Upon looking at the bbmap results summary and comparing (solely based on file sizes), it seems there are some differences between bbmap and SortMeRNA output. First, the output files from bbmap are all larger than the SortMeRNA output files (see example below). Additionally, the % reads mapped seem to differ; however, I am not completely sure how to read the bbmap result summary. Both the bbmap and SortMeRNA not\_aligned files are smaller than the input files, so rrna was removed using both programs. It seems SortMeRNA removes more than bbmap.

Note: when running bbmap, pairs are always kept together… therefore, read if one read is mapped and the other is unmapped, both with go to outm (this is the mapped reads output file). Therefore, outu files should not have split pairs and return ‘corrupt’ when bbrepair is run.

Sample 010:

Input file -> size = 54.8 GB

SortMeRNA -> 57% pass e-val threshold; 43% failing e-val threshold; not\_aligned file = 36.9 GB

Bbmap -> read 1 6.99% reads mapped/72% error rate; read 2 6.96% reads mapped/73% error rate; not\_aligned file = 52.2 GB

Upon discussion with Naomi, I have decided to move forward with assembly from bbmap sorted files. This is JGI standard protocol, and it will be obvious if rRNA remains if it is annotated later and I should (in theory) be able to bioinformatically be able to remove rRNA later.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Sample** | **% Mapped** | **# reads removed** |  | **Sample** | **% Mapped** | **% reads removed** |
| NIJ\_0\_CON\_001 |  |  |  | NIJ\_86\_CON\_013 |  |  |
| NIJ\_0\_SP1\_002 |  |  |  | NIJ\_86\_SP1\_014 |  |  |
| NIJ\_0\_SP2\_003 |  |  |  | NIJ\_86\_SP2\_015 |  |  |
| NIJ\_0\_SP3\_004 |  |  |  | NIJ\_86\_SP3\_016 |  |  |
| NIJ\_12\_CON\_005 |  |  |  | NIJ\_168\_CON\_017 |  |  |
| NIJ\_12\_SP1\_006 |  |  |  | NIJ\_168\_SP1\_018 |  |  |
| NIJ\_12\_SP2\_007 |  |  |  | NIJ\_168\_SP2\_019 |  |  |
| NIJ\_12\_SP3\_008 |  |  |  | NIJ\_168\_SP3\_020 |  |  |
| NIJ\_58\_CON\_009 |  |  |  | NIJ\_378\_CON\_021 |  |  |
| NIJ\_58\_SP1\_010 |  |  |  | NIJ\_378\_SP1\_022 |  |  |
| NIJ\_58\_SP2\_011 |  |  |  | NIJ\_378\_SP2\_023 |  |  |
| NIJ\_58\_SP3\_012 |  |  |  | NIJ\_278\_SP3\_024 |  |  |

**10 Jun 2021:**

Goal: begin first assembly with megathit

1. Check to see if megahit is installed on Wilhelm computer 8 (megahit –h)

No, need to install.

1. Run: conda install -c bioconda megahit

*Note: error was thrown. Comp 8 has python version 3.8, which is not compatible with megahit current version (1.2.9). Need to create an environment within the computer to install and run megahit.*

1. Create a new environment and install megahit:

conda create -n megahit-v1.2.9 python=3.4

conda activate megahit-v1.2.9

conda install -c bioconda megahit

*Note: This worked, megahit is now installed on computer 8 and is ready to run*

1. Code to run megahit:
   1. conda activate megahit-v1.2.9
   2. navigate to directory where you want the new output directories to go
   3. megahit --12 read.fastq - -o output\_name.megahit - -k-min 21 - -k-max 81 - -k-step 10 - -kmin-1pass - -continue

12 = tells megahit the input file, 12 is specific for a paired, interleaved input file

o = tells megahit the name of the output directory

k-min = tells megahit the min k-mer

k-max = tells megahit the max k-mer

k-step = tells megahit the interval between k-min and k-max to run

kmin-1pass = activates 1 pass mode to make assemblies more memory efficient, may be optimal for ultra-low depth datasets, such as soil

continue = continue a megahit run from its last available check point (prevents from having to start over if an error or some problem is thrown)

Today, the assembly for NIJ\_0\_CON\_001\_paired\_trimmed\_notaligned.fastq was started with megahit. I am expecting this to take ~1-3 days to complete.

**11 Jun 2021:**

Megahit update – instead of megahit taking 1-3 days to complete, NIJ\_001 only took ~1 hour to complete. The output looks ok, the n50 ~ 400. Since the run went do fast, I compared the contig stats for 001 run both with and without kmin-1pass. The overall run time was the same and the N50s were similar, if not slightly better without kmin-1-pass. I will move forward without kmin-1pass. I also compared the script above with a script with parameters suggested by JGI for Naomi’s dataset (k-min 23, k-max 123) run with 001. The N50 was better with JGI parameters, so the following script will be run on all samples:

megahit –12 /path/input.fastq -o NIJ\_0\_SP1\_002\_paired\_trimmed\_123.megahit --k-min 23 --k-max 123 --k-step 10 --continue

**17 Jun 2021:**

All files have now finished assembly. Next steps: gene region identification and annotation.

**29 Jun 2021:**

Gene identification and annotation will be performed following this pipeline published by Pound et al.:

https://www.protocols.io/view/functional-and-taxonomic-characterization-of-seque-buvbnw2n?step=3

Prior to functional annotation of genes, we first need to identify genes on our contigs. To do this we will use MetaGeneMark. This is an open access software accessed through a webpage gui. Final contig files are submitted for GFF gene prediction.

Next, protein (aa) sequences identified by MetaGeneMark are submitted to GhostKOALA.

MetaGeneMark version 3.25 and GhostKOALA version 2.2 were used for all samples.

**1 Jul 2021:**

All samples contig files were run individually through MetaGeneMark (http://exon.gatech.edu/meta\_gmhmmp.cgi) as described in Pound et al. 2021. Each sample took 5-10 min to both run and download and save output files. Both nucleotide and protein sequence files were saved as .fasta.

**1 Jul 2021 – 9 Jul 2021:**

All protein sequence files from MetaGeneMark were uploaded individually to GhostKOALA via the site (<https://www.kegg.jp/ghostkoala/>), searching the query against genus\_prokaryotes + family\_eukaryotes + viruses KEGG GENES database. Both annotation and taxonomy files were downloaded. Annotation files were already .txt, so files were renamed for each sample. Taxonomy files are downloaded as .top files, so they were renamed and saved as .txt files.

GhostKOALA Annotation Stats:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Sample** | **% Annotated** | **% undef** |  | **Sample** | **% Annotated** | **% undef** |
| NIJ\_0\_CON\_001 | 41.8 | ~49 |  | NIJ\_86\_CON\_013 | 50.2 | ~42 |
| NIJ\_0\_SP1\_002 | 40.4 | ~49 |  | NIJ\_86\_SP1\_014 | 56.1 | ~27 |
| NIJ\_0\_SP2\_003 | 40.8 | ~40 |  | NIJ\_86\_SP2\_015 | 56.3 | ~33 |
| NIJ\_0\_SP3\_004 | 41.5 | ~40 |  | NIJ\_86\_SP3\_016 | 56.8 | ~27 |
| NIJ\_12\_CON\_005 | 42 | ~40 |  | NIJ\_168\_CON\_017 | 42.1 | ~45 |
| NIJ\_12\_SP1\_006 | 46.1 | ~35 |  | NIJ\_168\_SP1\_018 | 52.4 | ~30 |
| NIJ\_12\_SP2\_007 | 55.3 | ~30 |  | NIJ\_168\_SP2\_019 | 50.0 | ~35 |
| NIJ\_12\_SP3\_008 | 41.6 | ~40 |  | NIJ\_168\_SP3\_020 | 50.4 | ~33 |
| NIJ\_58\_CON\_009 | 45.6 | ~45 |  | NIJ\_378\_CON\_021 | 40.2 | ~48 |
| NIJ\_58\_SP1\_010 | 58 | ~25 |  | NIJ\_378\_SP1\_022 | 52.8 | ~37 |
| NIJ\_58\_SP2\_011 | 57.4 | ~22 |  | NIJ\_378\_SP2\_023 | 43.9 | ~45 |
| NIJ\_58\_SP3\_012 | 54.5 | ~30 |  | NIJ\_278\_SP3\_024 | 51.3 | ~35 |

**8 Jul 2021:**

Next, we need to combine the functional and taxonomic annotations in a single excel sheet. This can be done in excel, but I decided to do it in R (I just know how to manipulate datasets better in R than excel). The full script for combination in R can be found in the GhostKOALA folder of the NIJ-ARF elements hard drive. Briefly, both .txt files were imported using read.delim() and named ‘an’ for functional annotation and ‘tax’ for taxonomic annotation. Columns were renamed in both datasets and then joined by gene query. The combined files were then exported as .xlsx into the GhostKOALA file folder.

*NOTE: It was realized that the number of rows in each file did not add up.* *The functional annotation files had the same number listed as ‘total queries’ by GhostKOALA, while the taxonomic annotation file had less rows. After some investigation, it seems that the tax file contains queries that were KO annotated taxonomically and/or were not KO annotated but assigned taxonomy (assuming via BLAST). The difference were gene queries that were not KO tax annotated, or BLAST annotated.*

*NOTE: The taxonomy file queries have the prefix ‘user:’ in front of the gene query from the annotation dataset. In order to join by query, this prefix must be removed prior to merging. You can do this using gsub().*

**9 Jul 2021:**

Working on subsetting the nucleotide sequences for gene expression analysis in CLC in python. An initial test of the program does not work. A new directory is created, but no output files are put in the folder and the script closes.

Try running this again, but with only “gene\_#” in the gene list file. Previously, the full query was saved in the gene list, but upon the investigation into the python script the queries are no the same. The nucleotide file has gene sequence lengths (nt) whereas the protein files has the protein length in amino acid (aa). This led to no sequences being filtered by the python script.

**13 Jul 2021:**

First, I made new gene list .txt files for 001 containing only “gene\_#”. This was done in excel using the following formula: =LEFT(A1,FIND(“|”,A1,1)-1) , where A1 contains the full query name. The formula was applied to all cells in the row containing observations in column A by double clicking in the bottom right hand corner of the first cell. The gene list was then copied to another workbook and saved as .txt.

After navigating to the GOI\_subsets folder in NIJ\_ARF\_metaT, the new input gene list was tested by both running the script from subset\_fasta.py manually in python (GOI\_001\_test2 folder) and by running the script in ubuntu (GOI\_001\_testscript) by typing ‘python subset\_fasta.py’. Both test runs ran correctly and contain output folders of the same size.

Next steps: need to save new gene lists for all remaining files and run them through subset\_fasta.py

**19 Jul 2021:**

First, new gene lists were saved for all samples. I decided to save 2 gene lists per sample. The first (#\_GOI.txt) contains all queries taxonomically annotated as bacteria, fungi, or archaea, while the second (#\_GOI\_fullan.txt) contains all queries taxonomically annotated as bacteria, fungi, or archaea AND has a functional annotation.

We will run 2 subsets per sample, one using #\_GOI.txt (folder = sampleID\_allGOI) and one using #\_GOI\_fullan.txt (folder = sampleID\_fullanGOI). Gene sequences output from MetaGeneMark were subset using the subset\_fata.py provided by Pound et al. (2021) by typing ‘python subset\_fasta.py’ and entering the required information. The number of sequences in the output fasta files post-subset are listed below.

*Note: for some samples a KeyError: ‘ ‘ was thrown when running subset\_fasta.py. It was found that when the nucleotide.fasta was opened, there were 1-2 spaces at the beginning of the files. When these were deleted, the script ran fine. Not sure why these spaces threw the script off.*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Sample** | **# GOI** | **# GOI f ann** |  | **Sample** | **# GOI** | **# GOI f ann** |
| NIJ\_0\_CON\_001 | 97099 | 44124 |  | NIJ\_86\_CON\_013 | 76152 | 40169 |
| NIJ\_0\_SP1\_002 | 144457 | 60684 |  | NIJ\_86\_SP1\_014 | 239384 | 138948 |
| NIJ\_0\_SP2\_003 | 215609 | 90895 |  | NIJ\_86\_SP2\_015 | 98272 | 58804 |
| NIJ\_0\_SP3\_004 | 199533 | 85371 |  | NIJ\_86\_SP3\_016 | 242840 | 144201 |
| NIJ\_12\_CON\_005 | 245970 | 106373 |  | NIJ\_168\_CON\_017 | 309888 | 134922 |
| NIJ\_12\_SP1\_006 | 143428 | 68757 |  | NIJ\_168\_SP1\_018 | 188002 | 102588 |
| NIJ\_12\_SP2\_007 | 241142 | 137071 |  | NIJ\_168\_SP2\_019 | 177219 | 92140 |
| NIJ\_12\_SP3\_008 | 154344 | 66991 |  | NIJ\_168\_SP3\_020 | 236980 | 124504 |
| NIJ\_58\_CON\_009 | 239293 | 113749 |  | NIJ\_378\_CON\_021 | 223652 | 93562 |
| NIJ\_58\_SP1\_010 | 379745 | 225618 |  | NIJ\_378\_SP1\_022 | 181906 | 102393 |
| NIJ\_58\_SP2\_011 | 275598 | 164587 |  | NIJ\_378\_SP2\_023 | 173775 | 79906 |
| NIJ\_58\_SP3\_012 | 296387 | 168556 |  | NIJ\_278\_SP3\_024 | 286213 | 156952 |

Next steps: need to recruit reads back to GOI sequences in CLC; specifically, I will need to use the “Map reads to reference” tool.

**7 Sep 2021:**

Today’s goal: import clean, trimmed read files back into CLC and import gene of interest (GOI) list files into CLC

We are going to search the trimmed, ribosomal removed (bbmap) reads fastq files against the GOI files for each sample. To import fastq files into CLC, click import, then Illumina. Add the fastq files to be uploaded and be sure to un-click the ‘paired reads’ box. Even though these are paired reads, the files I am importing are interleaved. After upload, reads can be specified as paired by opening the fasta file, clicking the box with a green check mark in front of white paper. Then, next to ‘paired status’ click ‘edit’ and click the ‘paired reads’ box. Now they are paired. All read files for mapping were saved in the ‘NIJ clean reads’ folder in CLC within the ‘Allison’ folder.

Next, import GOI list txt files. Click import, then standard. Ensure ‘automatic import’ is filled in the first page. This tells CLC to autodetect the file type for import. For this part, only the functionally and taxonomic annotated GOI lists were uploaded and searched. All GOI txt files were saved in the ‘GOI list\_func tx an’ file in CLC within the ‘Allison’ folder.

**8 Sep 2021:**

Start mapping reads to GOI files to get read counts in CLC using the ‘Map reads to Reference’ tool.

To open the tool, click ‘Toolbox’, then hover over ‘Resequencing Analysis’, then select the ‘Map Reads to Reference tool’. In this scenario, the reference is the GOI list and reads files are the sequences to map. Once the tool is open do the following:

1. Select sequencing reads. Navigate to the reads folder and select the sequencing reads file. Click on the file of interest and then select the the ‘->’ arrow. Ensure only your file of interest is listed in the ‘selected elements’ list. Click next.
   1. NOTE: because each sample has a different GOI list, we have to run each separately. If we wanted to search all reads files against 1 reference, we could select ‘batch’ in the bottom left corner.
2. Select the reference file to map to. Click the file with magnifying glass in the ‘references’ section and select the GOI list file in a similar manner to step 1. Click next.
3. Mapping options. Do not change any parameters, we are running default, which are suggested by Pound et al. (2021). Click next.
4. Click ‘create report’. Then next.
5. Select where we want the output file and report to save. For this project a new folder named ‘Read\_map\_GOI funtax an’ was created in CLC within the ‘Allison’ folder. This is where all the files will be saved to. Click finish.

**10 Sep 2021:**

Once the mapping is done, you can copy the mapping table from CLC into a new excel worksheet. This table contains the reference name, length, read count, and coverage. The total read count is what we want for downstream analysis.

Note: you can copy the whole table selecting row one and then hitting ctrl+A. However, only 50,000 rows can be copied to excel at one time. Not sure how to select rows in the middle to the bottom…

**20 Sep 2021:**

I noticed that the % mapped reads to the contig reference files are mostly below 10%, this is odd as these contigs were generated from the reads that are mapping to them. I chatted with Naomi and she suggested that .9 identity and similarity is stringent and recommended seeing how the mappings go at 0.85 identity and similarity. This was started today with the first 4 files in CLC.

**22 Sep 2021:**

Lowering the identity threshold did improve the % mapped values, but they are still well below 10%. It was also noted that the number of reads mapped in broken pairs is high (~40-50%). Not looking good. Will check with Naomi about how to proceed.

**27 Sep 2021:**

In checking the number of reads removed at each step thus far, as well as the reduction in total bp, one problem becomes apparent. It looks like a very small percentage of the total bases are assembled into contigs (<1%). This may explain why a small percentage is mapping.

Chatting with Naomi, she agrees this most definitely is contributing to the low read mappings. She also suspects that reads that are assembling into contigs are the more abundant constituents in the community, and to see a general snapshot/include rare things looking at the raw reads will work. Moving forward I am planning a three-pronged approach to these metaTs: (1) pull the mapping of reads to the annotated and assembled contigs and continue with analysis of this “abundant” community; (2) Upload reads and metadata to MG-rast and compare annotations from reads to annotations from contigs; (3) map reads to multiple databases such as CAzY, RDB, complete genomes from NCBI, PhyloBD, or IMG. To complete 1, I need to make a single reference of all contigs to be able to compare gene expression across all samples. MetaGenMark and GhostKOALA can then be run on this single reference to identify ORF and annotated identified genes. I can then re-map all of the samples to this single file. To complete 2, I need to get the metadata for these samples from Stacy and figure out how to upload samples and data to MG-rast. To complete 3, I need to download and format the database files to be uploaded into CLC.

**29 Sep 2021:**

Goal: merge and cluster contigs to make a single reference file.

1: Check the contig names, is there possibility for overlap that may cause issues when concatenating? (ubuntu)

***NOTE:*** when the prompt ‘permissions denied’ is returned in ubuntu, you can unlock permissions of the full folder by navigating back one folder and submitting the following code:

Chmod -R 777 foldername

1. To add the shorthand sample ID (e.g. 001) to the fasta header, navigate to the folder containing the files of interest and submit the following code (ubuntu):

Perl -p -e ‘s/^(>.\*)$/$1-001/g’ input.fa > output.fa

***NOTE:*** do not try to overwrite the input file by naming the output file the same as the input file. This does not work and will delete all contigs in the original file!

The fasta headers will now include -001 at the **end** of the header.

**30 Sep 2021:**

Goal: continue goal from 29 Sep.

2: concatenate all contig files into one, large reference (ubuntu).

1. To do this, navigate to the folder containing all of the contig files and submit the following code:

Cat file1.fa file2.fa file3.fa … > output.fa

3: cluster contigs in the master file to remove any redundancies (ubuntu).

1. To do this, we will use cd-hit in ubuntu. First, we need to install the software through conda using the following code:

Conda install -c bioconda cd-hit

1. Next, cluster contigs at 0.97 identity cutoff by submitting the following code:

cd-hit -i input.fa -o output -c 0.97

***NOTE:*** This was run locally on my laptop and returned the following error – Fatal Error: not enough memory, please set -M option greater than 4065.

I will need to try and run this in the Wilhelm lab on a computer with greater capacity. Earliest this can be done is Monday 10/4.

**4 Oct 2021:**

Goal: run cd-hit on Wilhelm lab computer 8 to cluster contigs at 0.97% identity.

Computer 8 did not have cd-hit installed in conda, so the same install code from 30 sep was submitted and cd-hit installed. Code from step 3b on 30 Sep was submitted, but same fatal memory error was returned. The following code was then submitted to increase memory:

cd-hit -i input.fa -o output -c 0.97 -M 5000

This was accepted and is currently running.

**3 Nov 2021:**

It was recommended by Steve to try a mega-assembly (also co-assembly?) to help improve the number of reads assembled into contigs (it was low, and seemed to only be dominant reads as the reads mapped was low (< 10%). To mega-assemble, I first need to merge all sample files into one reference. To do this, paired, trimmed, ribosomal reduced reads were concatenated in Linux using the following command:

cat input\_001.fastq, input\_002.fastq, … input\_024.fastq > NIJ\_all\_paired\_trimmed\_notaligned.fastq

**5 Nov 2021:**

All 24 interleaved reads files from each sample were successfully merged into one 1.2 tb file containing all reads from all samples. Next, is to use this master file as the input for assembly. I will first try doing this in Megahit using the same command from 11 Jun, using the NIJ\_all\_paired\_trimmed\_notaligned.fastq file as input. For this, I was working on computer 7 in the Wilhelm lab. Megahit needed to be installed via bioconda as described on 10 Jun. Note no error was thrown during install, so a new environment was not created. Megahit v1.2.9 was used. The follow code was used to start the Megahit program:

Megahit - -12 /media/wilhelmlab/Elements/NIJ\_ARF\_metaT/bbmap/NIJ\_all\_paired\_trimmed\_notaligned.fastq -o NIJ\_all\_contigs.megahit - -k-min 23 - -k-max 123 - -k-step 10 - -continue

I am expecting this process to take 3-4 days total. I will check progress on Monday 8 Nov. In the mean time I will try uploading paired, trimmed, ribosomal read files to MG-rast using API. To see how this was done, refer to the MGrast\_API.txt file in the NIJ-ARF elements drive. Briefly, the base command below was used for each individual file:

curl -X POST -H “auth: auth\_key: -F “upload:@input.fastq” <https://api.mg-rast.org/1/inbox>

**8 Nov 2021:**

Update on Megahit: to complete the mega-assembly, Megahit took ~2.25 days to run through all sequences and k-mer lengths. Overall, the mega-assembly seemed to perform better than individual assemblies, resulting in more total contigs (3969878 to 5042547; +1072669) and more basepairs assembled (2303279612 to 3080406514; +777126902). The average contig length an N50 also increased (avg contig: 610; N50: 644). Next step is to upload contigs to identify genes (ORF). Will try MetaGenMark following Pound et al. (2021) protocol.

Update on MG-rast: The API worked beautifully. All 23 files submitted were uploaded within a day. Need to wait to on MG-rast to complete their sequence analysis so all files can be submitted.

**9 Nov 2021:**

Goal for today: detect genes on contigs. To do this, the mega-assembly contigs file was submitted to the MetaGenMark server as described on **29 Jun 2021.** Unfortunately, it seems the mega-assembly file is too large to work with MetaGenMark. The job submits, but after ~30 minutes an error is thrown. I was able to start one job that didn’t throw an error; however no output was given after ~3 days of waiting. I chatted with Naomi about the JGI pipeline and what programs they use for gene detection and annotation. She pointed me to the JPI SOP publication:  <https://pubmed.ncbi.nlm.nih.gov/26512311/>. It seems that they use Prodigal for ORF detection. This program can output gff nucleotide and protein files and is available via conda, meaning it can be run in linux (ubuntu). I will try this for gene detection.

Install prodigal in linux (ubuntu):

conda install -c bioconda prodigal

Run prodigal on a metatranscriptome (after navigating to the folder that contains the mega-assembly file):

prodigal -i NIJ\_all\_contigs.fa -o NIJ\_all\_genes -d NIJ\_all\_nuc.fa -a NIJ\_all\_proteins.fa -p meta -f gff

Prodigal version V2.6.3: February, 2016 was used

**Note**: this took ~9 hours for the large mega-assembly file.

**10 Nov 2021:**

Goal: with genes identified, the next step is to annotate those genes. I will look into GhostKOALA ([GhostKOALA - Query Data Input (kegg.jp)](https://www.kegg.jp/ghostkoala/)) first, to keep things consistent with individual assemblies. I will also try submitting to EggNOG ([eggNOG-mapper (embl.de)](http://eggnog-mapper.embl.de/)), available of EMBL server and used by Naomi for gene annotation.

Upon looking at GhostKOALA, the maximum file size is 300 MB (equating to 1 million sequences with an average of 300). Unfortunately, our file is 1625 MB and thus too large to submit to GhostKOALA and there is no command line version of this software.

Similarly, EggNOG mapper only allows 100,000 proteins in FASTA format. I will chat with Naomi to see what she suggests. Can I split the file in bits and run then in chunks and then merge when all are done? Is there a program on command line that will handle the full file?

**11 Nov 2021:**

Naomi said that I could separate the large protein file into smaller files and run them in chunks – this shouldn’t impact the annotations. She also said EggNOG mapper has a command line version that can be accessed via conda and should handle the whole file. She will check and see if anyone is using computer 1 in the Wilhelm lab next week; if clear, she will set me up to run on Monday.

**15 Nov 2021:**

Goal: start eggNOG mapper for gene annotations on computer 1 in the Wilhelm lab. Github page for reference: [eggNOG mapper v2.1.5 to v2.1.6 · eggnogdb/eggnog-mapper Wiki · GitHub](https://github.com/eggnogdb/eggnog-mapper/wiki/eggNOG-mapper-v2.1.5-to-v2.1.6)

1: install eggnog-mapper in linux (ubuntu)

conda install -c bioconda eggnog-mapper

2: run basic command:

emapper.py -i NIJ\_all\_proteins.fa -o NIJ\_all\_an - -excel

This will run a diamond blastp search, and for those queries with hits to eggNOG proteins, will carry out functional annotation. Result will be exported as an excel file.

Note: the step 2 command was submitted on computer 1, but the error - -excel not found. Looking into what was installed, it looks like version v2.1.3 was installed. This explains why - -excel was not found as this was added in version v.2.1.4. I had to leave for class, Naomi said she would look into it tomorrow.

**16 Nov 2021:**

EggNOG update: Naomi said she ran ‘conda update eggnog mapper’ and this updated to version V2.1.6. Will submit the command again.

The following was run:

emapper.py -i NIJ\_all\_proteins.fa -o NIJ\_all\_an - -excel

Note: another error was thrown. This was stated the database was not present and to run download\_eggnog\_data.py. As a result, download\_eggnog\_data.py was run however it did not download the databases. It appears that the folder the software wants to unpack the databases into within conda does not exist. Naomi navigated to the root folder and made a new folder (mkdir) under the name the software wants. This worked out as the folder did not exist previously. download\_eggnog\_data.py was rerun and databases were successfully installed. The command above was rerun and started with no errors. I am expecting this to take ~2-3 days.

**19 Nov 2021:**

EggNOG mapper finished running! Now I need to determine how I want to subset genes of interest for the reference file to submit for read mapping.

Originally, all bacterial, archaeal, and fungal genes were kept as genes of interest and filtered directly in the merged GhostKOALA output. Looking at eggNOG, the taxonomy information is sort of merged all in one column, making it difficult to filter in excel. Instead, I imported the excel file in R and filtered for terms ‘Bacteria’, ‘Archaea’, and ‘Fungi’. For more information, see the ‘Annotation file processing’ r file found in the Eggnog folder ("D:\NIJ\_ARF\_metaT\Eggnog\Annotation file processing.R") of the Elements drive. The list of gene names for GOI were saved as an excel file (NIJ\_all\_GOI) and .txt file.

Next, I pulled over the subset\_fasta.py that was used to search the nucleotide gene file from GhostKOLA (Pound et al. 2021) and ran that script in linux. The NIJ\_all\_GOI.txt and NIJ\_all\_nuc.fa were used as the inputs. Unfortunately, this did not work as no sequences were pulled into the subset file. Suggests to me that the output from MetaGenMark and Prodigal differ enough that the script doesn’t detect it. I will have to find an alternative to filter or update the script.

**22 Nov 2021:**

In searching google for how to filter a fasta file by matching a part of header to a list, I found a conda package called seqkit that looks promising.

1: install seqkit:

conda install -c bioconda seqkit

First I used seqkit to see what is part of the header it considers the “ID”

seqkit seq NIJ\_all\_nuc\_sub.fa -n -i

From this, it seems that the eggnog mapper query matches the ID values (e.g., k123\_2696563\_1). I will look to see if I can filter the nucleotide file by the IDs of GOI found in the list file (NIJ\_all\_GOI.txt).

Looks like grep within seqkit can be used to filter sequences from a list input. The following string was run:

seqkit grep -f NIJ\_all\_GOI.txt NIJ\_all\_nuc\_sub.fa -o NIJ\_all\_GOI\_nuc.fa

This results in a successful subset of the sequences. This was confirmed by looking at the file size and the first few lines using the head command. Next step is to map reads from each individual file to this reference (NIJ\_all\_GOI\_nuc.fa).

A new folder called NIJ mega assembly was created under the Allison folder in CLC on computer 3. Map reads to reference was used as described above, however this time batch (bottom left corner) was clicked to run each sample file in succession. For this the input files are each of the 24 trimmed, clean, and bbmap rRNA removed reads files, which will be individually mapped to the GOI reference file (NIJ\_all\_GOI\_nuc.fa). This was started ~10:30 and is expected to take anywhere from 3-5 days to run through all 24 files.