MiSeq Amplicon Bioinformatics Pipeline

MERGING SEQUENCES WITH **PEAR** AND CREATING OTU CLUSTERS AND REPRESENTATIVE SEQUENCES WITH **SWARM**

By Chip Sisson
Started on October, 2016

This guide will help you understand how to manipulate the raw data from a run of Illumina MiSeq High-Throughput sequencing so that you have workable data for papers, theses, etc. It is, essentially, a heavily annotated version of JD's guide, `Guide_MiSeqPipeline.txt`, which is in the `MiSeq_pipeline` folder you'll be working from.

Table of Contents

BEFORE YOU START…

How to Use this Guide

Each section is linked in the table of contents, so you shouldn't have to do too much digging! In terms of formatting, any sort of Terminal command will be formatted like this:

```
command -m parameters
```

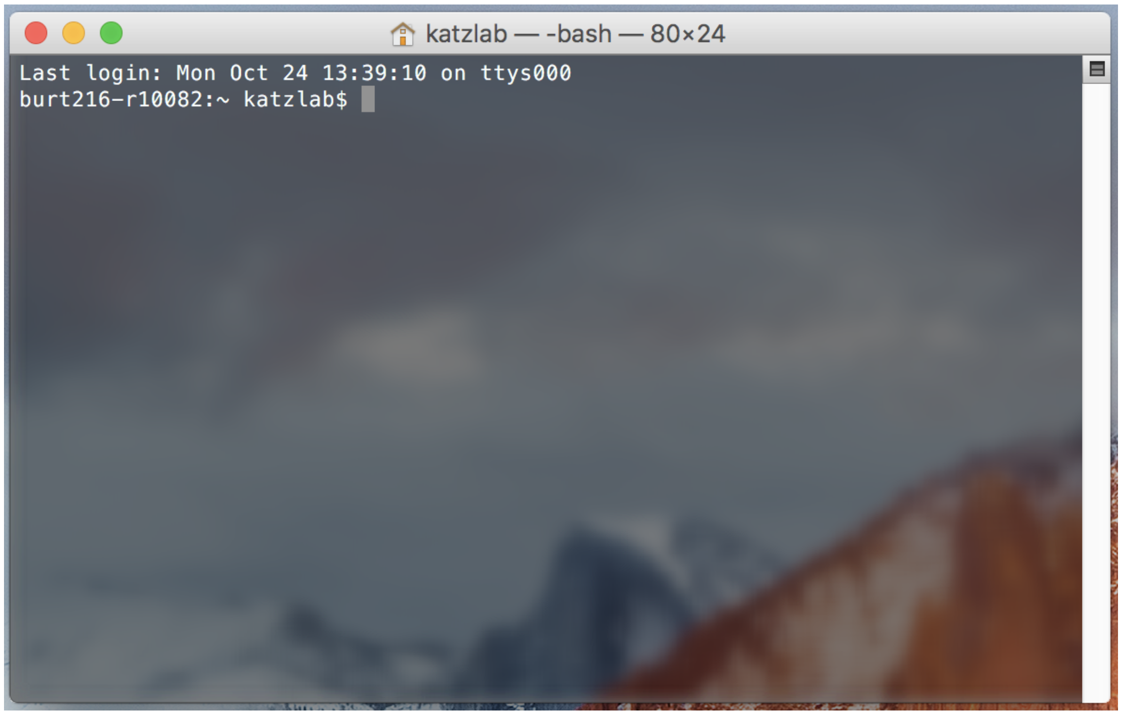
You can copy and paste these boxes directly into the Terminal. Hit enter after this unless the guide instructs you to otherwise..

Don't be intimidated by the size of this document-- it's here to help you! That being said, if anything is unclear be sure to leave a comment with what you had trouble following or any suggestions for improvement. This guide exists to assist undergrads working with Miseq data, so please, don't feel bad about voicing concerns!

Getting Familiar with the Terminal

You'll be doing all of your work in the terminal. You can access the terminal by clicking the black box icon with a small carrot and an underscore (>_) on the toolbar of your computer. If the terminal isn't on the dock, you can open it through the applications folder or through spotlight search. You'll be using it a lot, so I recommend pinning it to the dock.

When you log onto the terminal, it will look something like this….



```
Last login: Mon Oct 24 13:39:10 on ttys000
burt216-r10082:~ katzlab$
```

It will likely appear white with black text unless someone has edited the shell under Preferences. You could do this too, if you're so inclined, but it's not necessary for the any of these analyses--just aesthetics.

Some basic commands, tips & tricks...

Command	Function
Enter / Return	Runs your command line
`cd`	Move through the folders on your computer. You can also drag files from a finder window if you don't know the folder path.

Up or down arrow	Scroll through past commands. Saves time if you need to rerun something!
Left or right arrow	Move through your command line to edit/fix typos
`exit`	Ends session. Make sure to do this when you’re done!
`../`	Moves you back a folder
`/`	Moves you forward a folder
⌘+	Makes text bigger
⌘-	Makes text smaller
⌘o	Restores text to original size
⌘K	Clears terminal
Ctrl + Z	Stops current command
⌘q	Quits the terminal. Will ask before terminating background tasks.
`sudo`	Prompts password input
tab	Autocompletes folder/file names
⌘F	Search for a particular term

Using the terminal is pretty intuitive, but this guide will help you through anything that gets tricky.

Prepping Your Computer

It is vital to make sure your computer has all the programs required for the bioinformatics you're about to do. This section will include a list of the required programs, and how to download them if your computer doesn't have them.

An easy way to check for any of them is to open the terminal, type the name of the software followed by `--version`

Required Software

Below is a list of software necessary for these scripts to function. It's best to create a folder in your Documents folder (or somewhere else permanent!) and name it something like bioinformatics_software. What you name it is up to you-- just be sure there's no spaces. You can move all the source code download files to this folder.

Homebrew and git

Before you start, you'll need to download **Homebrew**, which is a program that manages packages that is **not** included standard on most macs. To install homebrew, open the terminal and paste:

```
ruby -e "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

Now, you can get started with the installation of the programs you'll *actually* be using for bioinformatics.

After installing **Homebrew**, you need to install `git`. You can paste this into the terminal:

```
brew install git
```

Now you're ready to install the required programs for bioinformatics!

Python3

Python is the programming language that was used to write all the scripts present in this guide. Make sure you're running the latest version of Python. If you don't have Python3, you can download it here. The easiest way to check is to type `python3` into the terminal and hit enter. Make sure Python3 is installed properly, because it contains all of the modules necessary to install the other software. Without this, you can't move forward!

You can follow this guide to install simultaneous versions of Python (in this case, 2.7 and 3.5).

To install Python3, you can use Homebrew. Copy and paste this into the terminal after downloading the package from the Python website:

```
brew install python3
```

After this, you can begin to install the dependencies necessary for Biopython.

Biopython

Biopython is a set of free-to-use, open-source written by and for biologists to perform work in bioinformatics using Python. You can learn more about Biopython here, and download the package here. A guide for installation troubleshooting can be found here. If your computer already has Biopython, it's a good idea to reinstall unless you're certain it's installed under Python3-- if Biopython is installed under Python2.7 or any other earlier version, it won't work.

It's important to make sure the dependencies of Biopython (`numpy`, `scipy`, `pandas`) are installed, as biopython often utilizes them to run command lines.

First, make sure `pip` (pronounced 'pipe') is up-to-date by running this command:

```
python3 -m pip install --upgrade pip
```

Then install the 3 dependencies. One line at a time, run:

```
python3 -m pip install numpy
```

```
python3 -m pip install scipy
```

```
python3 -m pip install pandas
```

Once these have been successfully installed, you can install Biopython itself.

```
python3 -m pip install biopython
```

To make sure Biopython installed correctly, run these simple commands to test:

```
python3
```

```
import Bio
```

Note: this is case sensitive

If you don’t receive an error message, it worked! You can exit Python3 now.

```
exit()
```

PEAR

PEAR stands for Paired-End reAd mergeR, and is an “ultrafast, memory-efficient and highly accurate pair-end read merger. It is fully parallelized and can run with as low as just a few kilobytes of memory. PEAR evaluates all possible paired-end read overlaps and without requiring the target fragment size as input. In addition, it implements a statistical test for minimizing false-positive results. Together with a highly optimized implementation, it can merge millions of paired end reads within a couple of minutes on a standard desktop computer.” (from their website)

To summarize, PEAR mergers your forward and reverse reads from the amplicons.

To download PEAR, open the terminal and enter these commands:

```
brew install autoconf automake libtool
```

```
git clone https://github.com/xflouris/PEAR.git
```

Note: git is a package within Python, so if you have an issue with your Python installation you won't be very successful here.

And follow the rest of the steps on the website here. For convenience's sake, the command line is pasted here as well.

```
cd PEAR
```

```
./autogen.sh
```

```
./configure
```

```
make
```

```
sudo make install
```

`sudo` will prompt you for the password of whatever account you're currently working from.

To make sure PEAR installed correctly, paste:

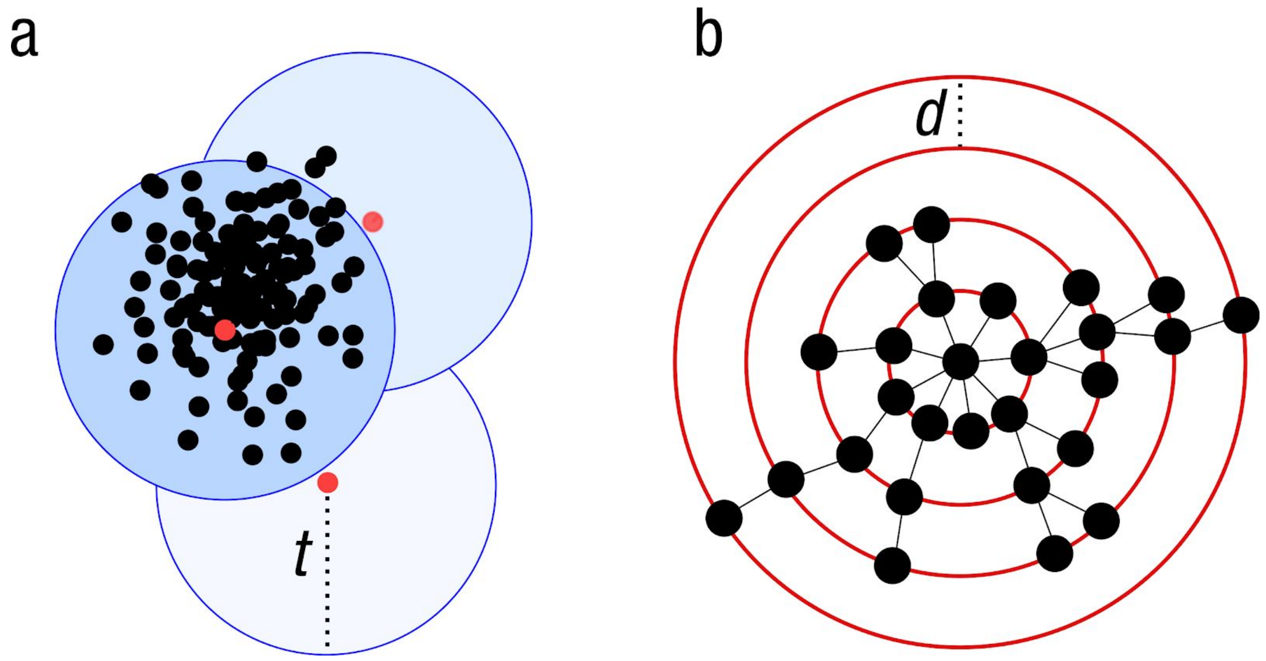
```
PEAR --version
```

As of 10/25/2016, this should be version 0.9.10

SWARM

SWARM is a clustering algorithm used to create OTUs. "The OTU picking step assigns similar sequences to operational taxonomic units, or OTUs, by clustering sequences based on a user-defined similarity threshold. Sequences which are similar at or above the threshold level are taken to represent the presence of a taxonomic unit (e.g., a genus, when the similarity threshold is set at 0.94) in the sequence collection. -- from their website

In other words, it chooses a central sequence based on a parameter you set (usually abundance) and moves outwards in concentric circles, connecting points until a robust OTU is formed.



Older methods (a) are more blunt, and less likely to accurately capture an OTU as it takes a radius *t* and circles everything within it. SWARM (b) works through concentric circles that are distance *d* away, and thus creates a network of sequences that cluster together that would not under older methods.

[You can download the SWARM package here.]

To install SWARM:

```
git clone https://github.com/torognes/swarm.git
```

```
cd swarm/src/
```

```
make
```

```
cd ../bin/
```

[Read a journal article about SWARM here]

Similar to PEAR, make sure SWARM is installed correctly by checking what version you have.

```
./swarm --version
```

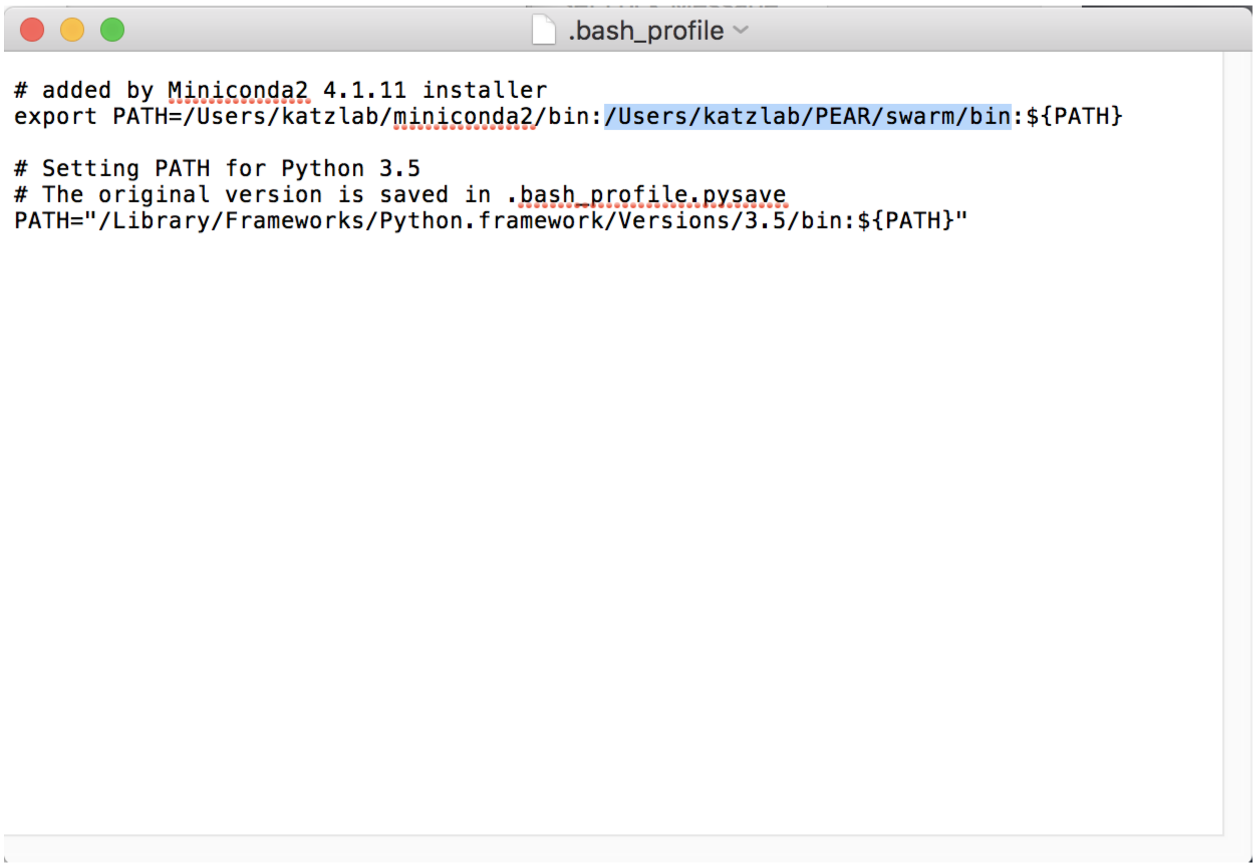
It should be 2.1.9 as of when this guide was written.

You'll need to update the path for SWARM as well, you can do this by opening the bash profile:

```
cd
```

```
open .bash_profile
```

It should look something like this:



```
# added by Miniconda2 4.1.11 installer
export PATH=/Users/katzlab/miniconda2/bin:/Users/katzlab/PEAR/swarm/bin:${PATH}

# Setting PATH for Python 3.5
# The original version is saved in .bash_profile.pysave
PATH="/Library/Frameworks/Python.framework/Versions/3.5/bin:${PATH}"
```

Where the blue highlighted text is wherever the SWARM folder is in your directory. You can click and drag the **bin** folder from within swarm after the colon, and add another colon. Add also /usr/local/sbin: to your PATH to be able to run swarm (sh command) as 'swarm' without './' .

Hit ⌘S and exit the terminal.

ncbi_BLAST+

You've probably used BLAST online before. To review, BLAST is a massive database of sequences to which you can compare your amplicon data. BLAST+ is a package of commands that can be input into the terminal to run BLAST. More about the package here.

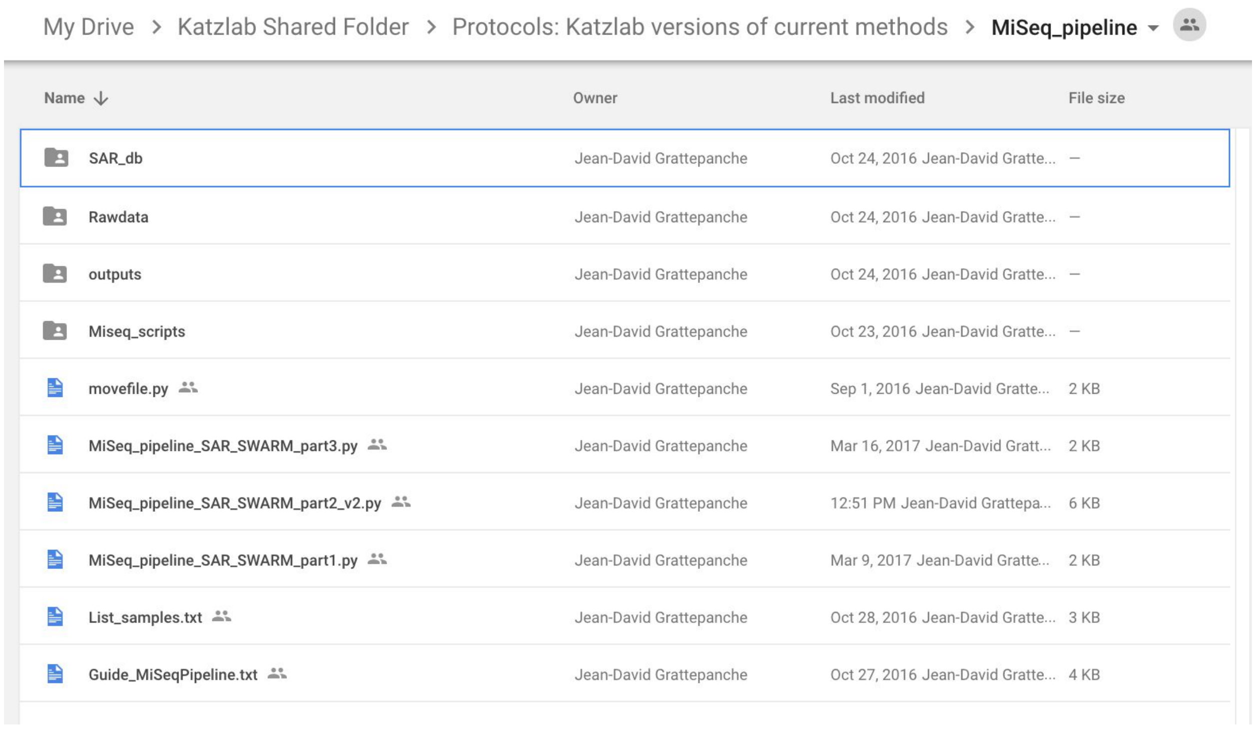
Clicking this link should download the package directly.

You don't have to install this one through the the terminal; just follow through the installation wizard.

After this step, you should have all the programs necessary to run the scripts!

Downloading your files

Perhaps the most important step of all! You can download a folder called MiSeq_pipeline from google drive (Katzlab Shared > Protocols > MiSeq Pipeline) that will have all of the python scripts and data you need. Be warned-- this folder is massive and will take some time to download, and at times it will break up into smaller fragments. Be sure to move all the files into one folder. It's best to download:



My Drive > Katzlab Shared Folder > Protocols: Katzlab versions of current methods > MiSeq_pipeline

Name	Owner	Last modified	File size
SAR_db	Jean-David Grattepanche	Oct 24, 2016 Jean-David Gratte...	—
Rawdata	Jean-David Grattepanche	Oct 24, 2016 Jean-David Gratte...	—
outputs	Jean-David Grattepanche	Oct 24, 2016 Jean-David Gratte...	—
Miseq_scripts	Jean-David Grattepanche	Oct 23, 2016 Jean-David Gratte...	—
movefile.py	Jean-David Grattepanche	Sep 1, 2016 Jean-David Gratte...	2 KB
MiSeq_pipeline_SAR_SWARM_part3.py	Jean-David Grattepanche	Mar 16, 2017 Jean-David Gratt...	2 KB
MiSeq_pipeline_SAR_SWARM_part2_v2.py	Jean-David Grattepanche	12:51 PM Jean-David Grattepa...	6 KB
MiSeq_pipeline_SAR_SWARM_part1.py	Jean-David Grattepanche	Mar 9, 2017 Jean-David Gratte...	2 KB
List_samples.txt	Jean-David Grattepanche	Oct 28, 2016 Jean-David Gratte...	3 KB
Guide_MiSeqPipeline.txt	Jean-David Grattepanche	Oct 27, 2016 Jean-David Gratte...	4 KB

- **SAR_db**
- **MiSeq_scripts**

- **movefile.py**
- **MiSeq_pipeline_SAR_SWARM_part3.py**
- **MiSeq_pipeline_SAR_SWARM_part2_v2.py**
- **MiSeq_pipeline_SAR_SWARM_part1.py**
- **Guide_MiSeqPipeline.txt**

>>> **You can download your data here** <<<

Create a new folder called `'Rawdata'` and move your files here. Also create an empty folder called `'outputs'`.

Do NOT rename anything-- this will potentially render the scripts ineffective.

>>>Be sure to download the most recent version of all scripts (part1-3 wrappers, plus all scripts in the MiSeq_scripts folder) before beginning to assure you are using the correct, most up-to-date version of the scripts. This avoids scripts crashing or having to redo the entire pipeline (very time consuming!!)<<<

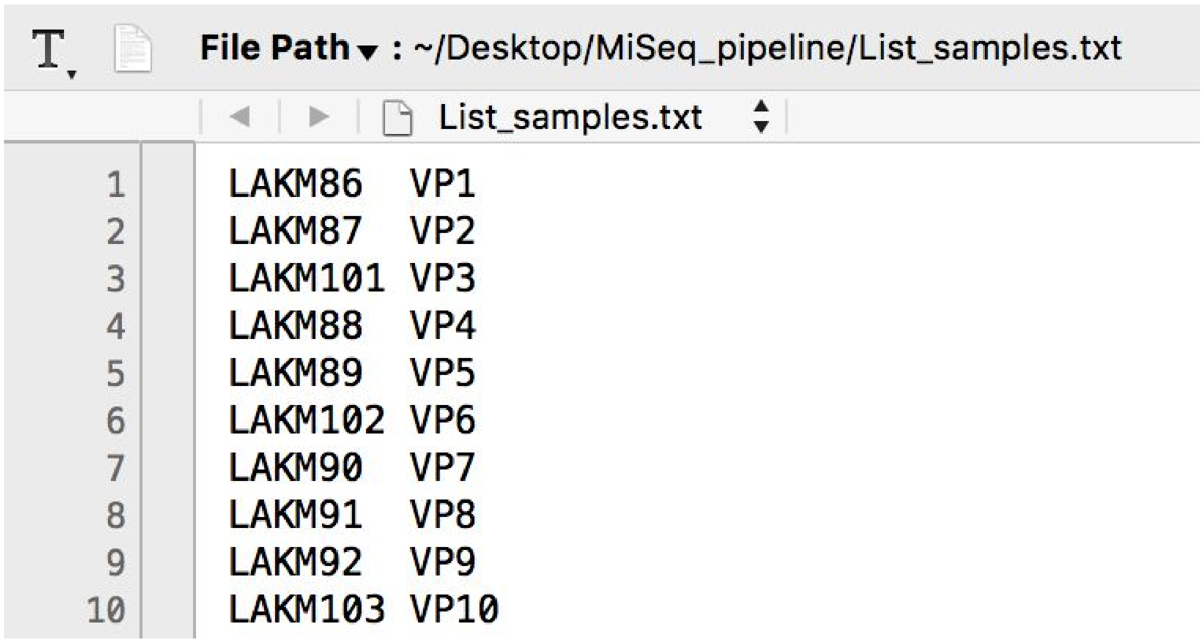
GETTING STARTED

Once you've made sure your computer has all the necessary scripts and downloads, and you have all of your files from the Miseq_pipeline folder from google drive, you're ready to begin! This section will guide you through the last of the preparatory steps before you can execute the commands.

Make sure all of your raw data files are present in the Rawdata folder within the Miseq_pipeline folder. As of 10/24/16, all of the data from the first and second MiSeq plates from URI are in the folder; you shouldn't have to worry about hunting anything down.

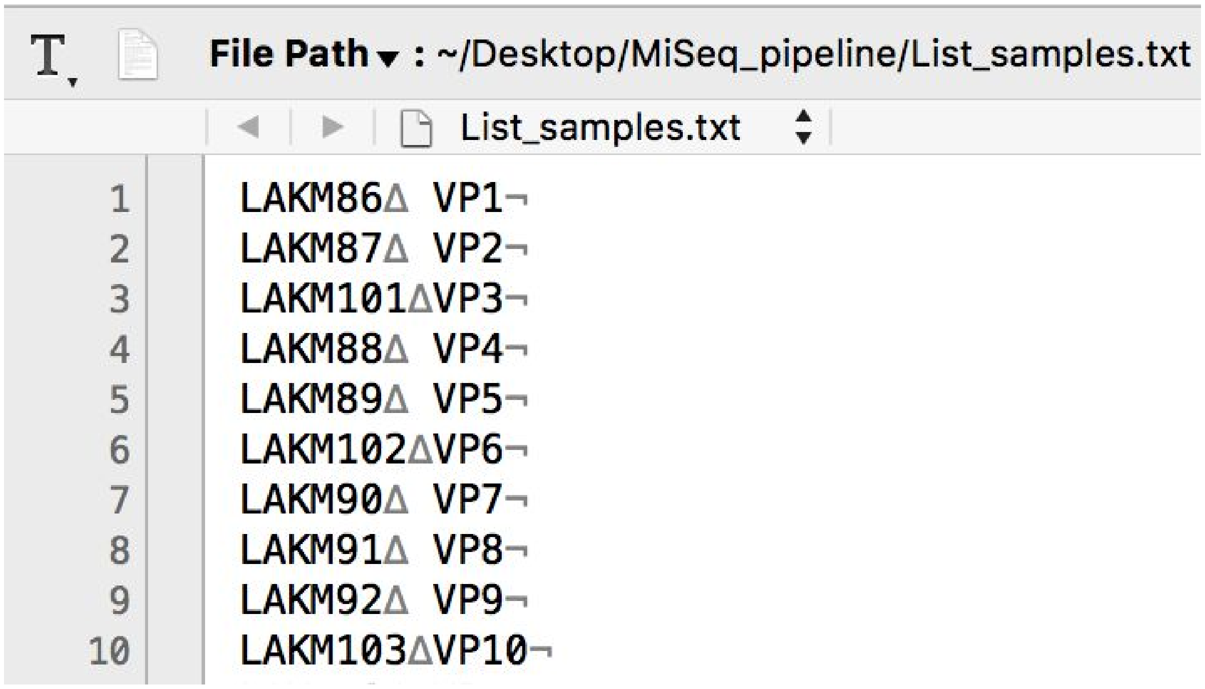
Create a list of all samples

There will be a file within the MiSeq_pipeline called `List_samples.txt.` This file contains all the LAKM numbers (used to identify your sample on the MiSeq plate) and a descriptive name that you assign all of your amplicons.



```
LAKM86  VP1
LAKM87  VP2
LAKM101 VP3
LAKM88  VP4
LAKM89  VP5
LAKM102 VP6
LAKM90  VP7
LAKM91  VP8
LAKM92  VP9
LAKM103 VP10
```

Open the file in TextWrangler and edit as needed. Separate the LAKM number from the sample identifier with `tab`. To be sure all of your samples are formatted correctly, follow the menu from `View > Text Display > Show Invisibles.` Make sure all of your samples are



```
LAKM86△ VP1¬
LAKM87△ VP2¬
LAKM101△VP3¬
LAKM88△ VP4¬
LAKM89△ VP5¬
LAKM102△VP6¬
LAKM90△ VP7¬
LAKM91△ VP8¬
LAKM92△ VP9¬
LAKM103△VP10¬
```

separated by a grey triangle-- this indicates that they're separated by a tab. If you see any periods, edit them-- those are spaces. It should look something like the image below...

Note: you can order the samples by LAKM number **or** by sample number; in this case my samples were split between two plates and I felt that ordering them chronologically rather than numerically would facilitate analysis down the line.

Periods, underscores, and dashes frequently break the script-- give the samples a shortened title and create a list with longer, more descriptive titles for your own

reference. For example, ‘VP1’ is ‘Feb10_S1_10D’. You can edit the names in your final excel sheet.

If you have and excel or google sheets page, you can download the text as a Tab Delimited Text (.txt) file and save some time.

`List_samples.txt` as it exists when you download it has all LAKM numbers from the first two URI plates. You can delete unwanted numbers and edit the document, but **DO NOT** change the name or the rest of the scripts will not work correctly. Save the file and make sure it’s still in the `MiSeq_pipeline` folder.

FIVE-STEP METHOD

JD has graciously created scripts that automate almost all of the process. This is ultimately an easier way to move through bioinformatics. These scripts will give you a few different outputs, such as a .txt file with all sequences that don't have any matches on BLAST or within the SAR database. The main file you will use for your analyses is called `OTUtable_final.txt`.

Each of the scripts calls on smaller scripts, also written by JD, that are numbered 0-10 in the `Miseq_scripts` folder. This, in part, is why not moving any of the documents is very important!

Let's get started.

STEP 1

Essentially, the first script will use PEAR to merge all of your reverse and forward reads and then convert all the files from .fastq to .fasta. It is important to have **all** your raw reads-- not just ones from a single plate-- for this step and onwards. This will help reduce the number of computer-generated artifacts that are interpreted by the program as robust OTUs. This will also help reduce the amount of false leads from PCR artifacts present in your individual amplicons. Previous methods using `QIIME` neglected the number of reads, and thus vastly misrepresented the samples.

Open the terminal, and enter:

```
cd Desktop/MiSeq_pipeline
```

```
python3 MiSeq_pipeline_SAR_SWARM_part1.py
```

You can either type in, copy-and-paste, or drag the the script onto the terminal window. You'll be using the script named `MiSeq_pipeline_SAR_SWARM_part1` for this step.

Follow the prompts as instructed.

Where is your **Raw Data** folder?	Drag `'MiSeq_pipeline'` folder into terminal window
Where is your **sample list**?	Drag `'List_samples.txt'` into terminal window
Do you need to **subsample**?	Type '**n**' for no.

This takes about 5-10 minutes per sample to run.

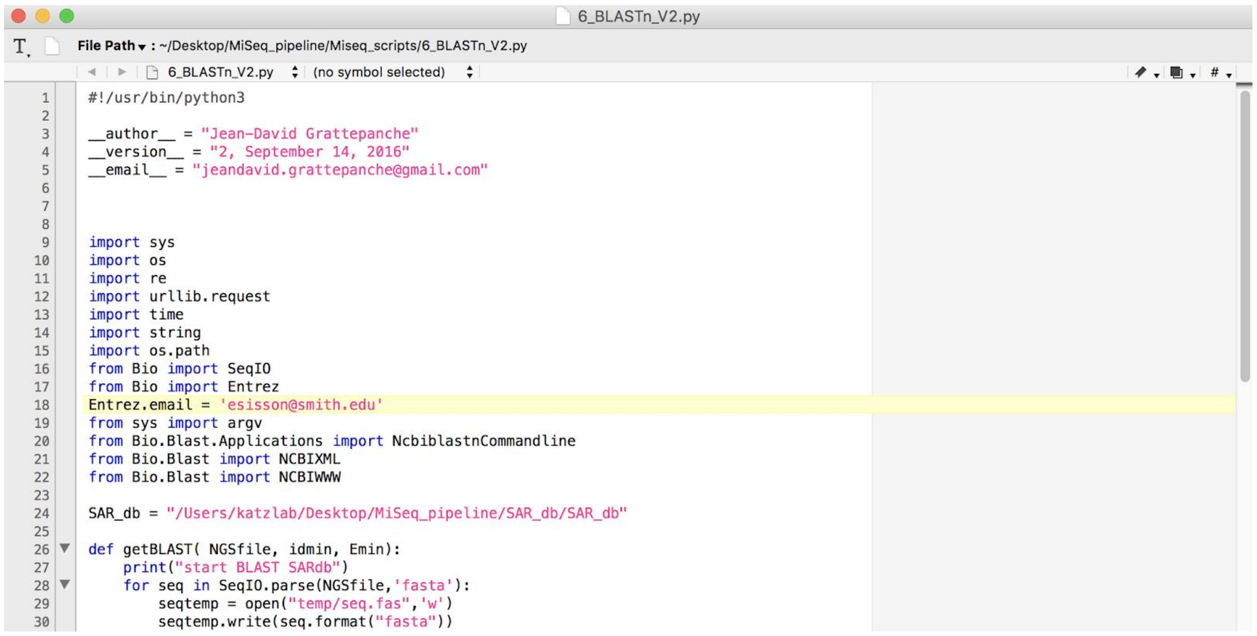
STEP 2

This script creates OTUs with SWARM, and will output your sequences as a file called `SWARM_postout.fas`. From there, the script creates an OTU table which is perhaps the most useful feature of all-- this script will list all the OTUs created by swarm, and list the number of reads that appear in each amplicon. This step is where the order of `List_samples.txt` comes into play.

The script will BLAST the SWARM OTUs that are considered "not rare" (ie, have greater than 2 reads. These are arbitrary, but work to reduce junk data.) The script will BLAST first to the SAR database included in your folder. A separate file with all the sequences that did not BLAST to something else will be created in the end.

You'll use this to create the files for outgroup removal, which will appear in a folder within `OTUs` called '`outgroup_removal`.' This will be the file you send to CIPRES to return your unmasked tree.

Before running the second script, you need to create an NCBI account and input your email into line 18 (highlighted) of the script `5_BLASTn_V3.py` under the `Miseq_scripts` folder.


6_BLASTn_V2.py
File Path ▾ : ~/Desktop/MiSeq_pipeline/Miseq_scripts/6_BLASTn_V2.py
6_BLASTn_V2.py
(no symbol selected)

```
1   #!/usr/bin/python3
2
3   __author__ = "Jean-David Grattepanche"
4   __version__ = "2, September 14, 2016"
5   __email__ = "jeandavid.grattepanche@gmail.com"
6
7
8
9   import sys
10  import os
11  import re
12  import urllib.request
13  import time
14  import string
15  import os.path
16  from Bio import SeqIO
17  from Bio import Entrez
18  Entrez.email = 'esisson@smith.edu'
19  from sys import argv
20  from Bio.Blast.Applications import NcbiblastnCommandline
21  from Bio.Blast import NCBIXML
22  from Bio.Blast import NCBIWWW
23
24  SAR_db = "/Users/katzlab/Desktop/MiSeq_pipeline/SAR_db/SAR_db"
25
26  def getBLAST( NGSfile, idmin, Emin):
27      print("start BLAST SARdb")
28      for seq in SeqIO.parse(NGSfile,'fasta'):
29          seqtemp = open("temp/seq.fas",'w')
30          seqtemp.write(seq.format("fasta"))
```

While you're here, make sure line 24 routes to the SAR database on **your** computer-- if you're using a user other than katzlab, or the `MiSeq_pipeline` folder is anywhere other than the Desktop, this script will crash.

As with STEP 1, move to the `MiSeq_pipeline` folder if you haven't already. Run the part 2 script.

```
cd Desktop/MiSeq_pipeline
```

```
python3 MiSeq_pipeline_SAR_SWARM_part2_v2.py
```

You'll be asked a series of prompts afterwards.

Prompt	Input
1. Where is your **raw data output file**?	Drag the **MiSeq_pipeline folder icon** into the terminal window, hit enter.
2. What is your **list of your samples**?	Drag **`List_samples.txt`** onto the terminal
3. At what '**percentage' you want to cluster your OTUs**? (By percentage it means *d,* distance.)	1 is standard, you can hit enter
4. Do you want to **subsample your reads**?	You don't. Type **'n'** or **'no'**
5. Do you want to **assign taxonomy**?	Type **yes** or **y**.
6. What is the **minimum identity percentage** for BLAST?	Default is 95%, we often use 97%
7. What is the **minimum e-value cutoff**?	Hit enter for standard

After this, hit enter! The script will take a few days to run, especially if you have a lot of reads to process. You can still use the internet, excel, etc on your computer without worrying about a crash.

STEP 3

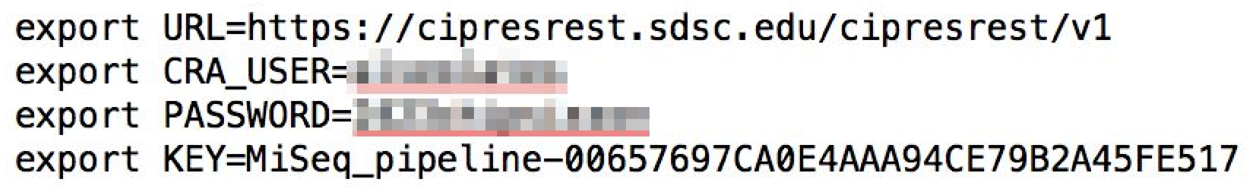
In this step, you'll use the output file `OTUseq_TA.fasta` to create an unmasked alignment with RAxML on CIPRES. CIPRES is a web service that includes a number of phylogenetic tools and programs, and is helpful because it saves us the trouble of having to install all the dependencies on our own computers.

The output will be a phylogenetic tree with all of your OTUs nested within. Exciting! You'll use this for outgroup removal later on.

Unmasked trees take a few days to run, but you'll get an email when the tree is finished running.

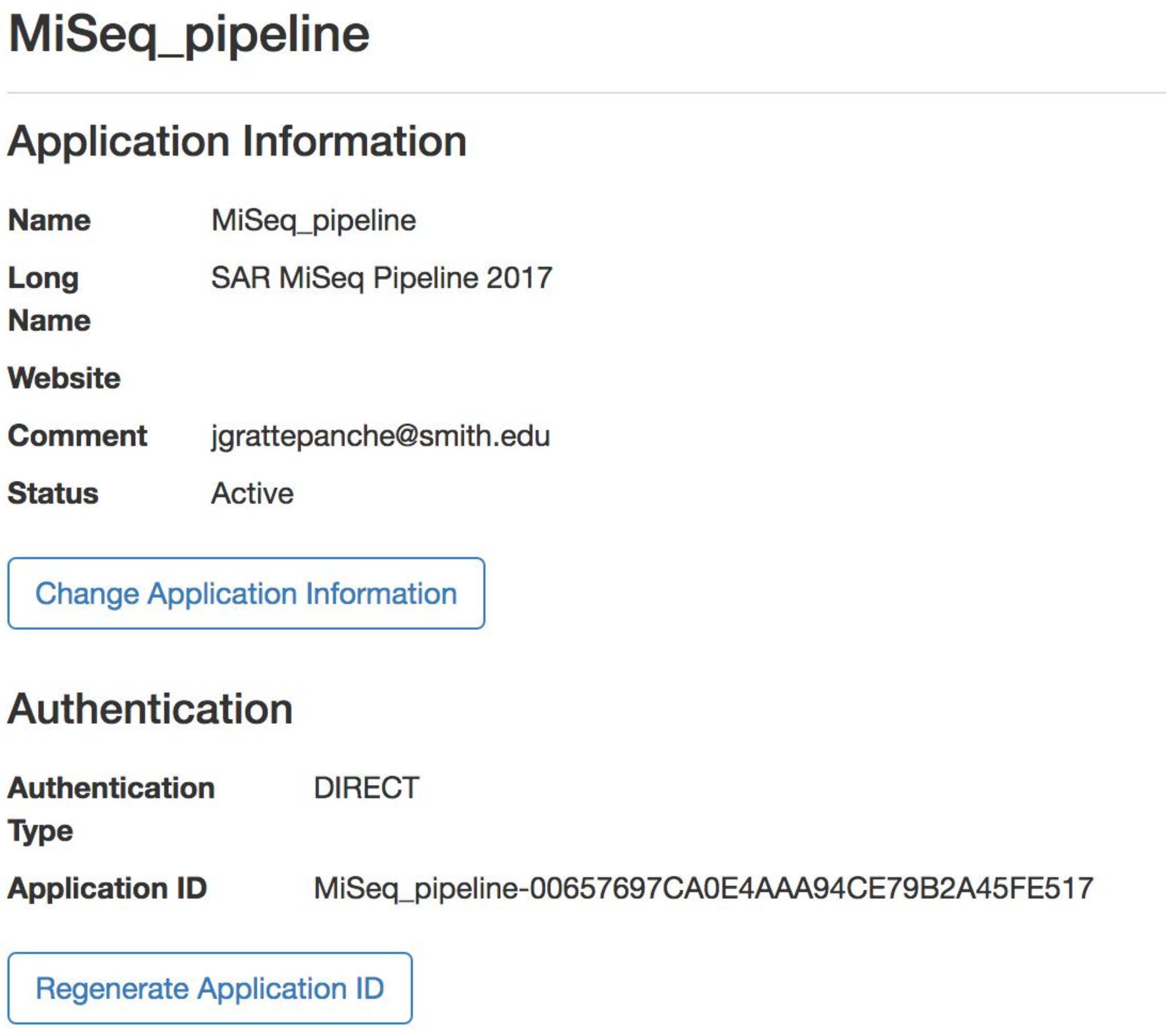
This step has been largely automated, but to do it manually:

First you'll need a CIPRES and CIPRES Rest API account. It's free and pretty quick, just be sure to write down your username and password. I created a document called '`CIPRES_info.txt`' and saved it in my `MiSeq_pipeline` folder. Save it in this format:



```
export URL=https://cipresrest.sdsc.edu/cipresrest/v1
export CRA_USER=[redacted]
export PASSWORD=[redacted]
export KEY=MiSeq_pipeline-00657697CA0E4AAA94CE79B2A45FE517
```

Where `CRA_USER` is your username and `PASSWORD` is your password. There will also be a unique key that is generated for the application you will need to set up. From the CIPRES Rest API homepage, click the drop down menu below Developer and click Application Management. From there, there will be an option to create a new application. Fill in the information accordingly (your contact can be JD) and hit enter.



MiSeq_pipeline

Application Information

Name	MiSeq_pipeline
Long Name	SAR MiSeq Pipeline 2017
Website	
Comment	jgrattepanche@smith.edu
Status	Active

Authentication

Authentication Type	DIRECT
Application ID	MiSeq_pipeline-00657697CA0E4AAA94CE79B2A45FE517

The information should look something like this. Copy your Application ID into your info file.

Open a new terminal window. Import the account information (below is the Katzlab shared account):

```
export URL=https://cipresrest.sdsc.edu/cipresrest/v1
export CRA_USER=Katzlab
export PASSWORD=ciliate123
export KEY=Tree_pipeline-C6E99C79E24545C19E78740CE389B274
```

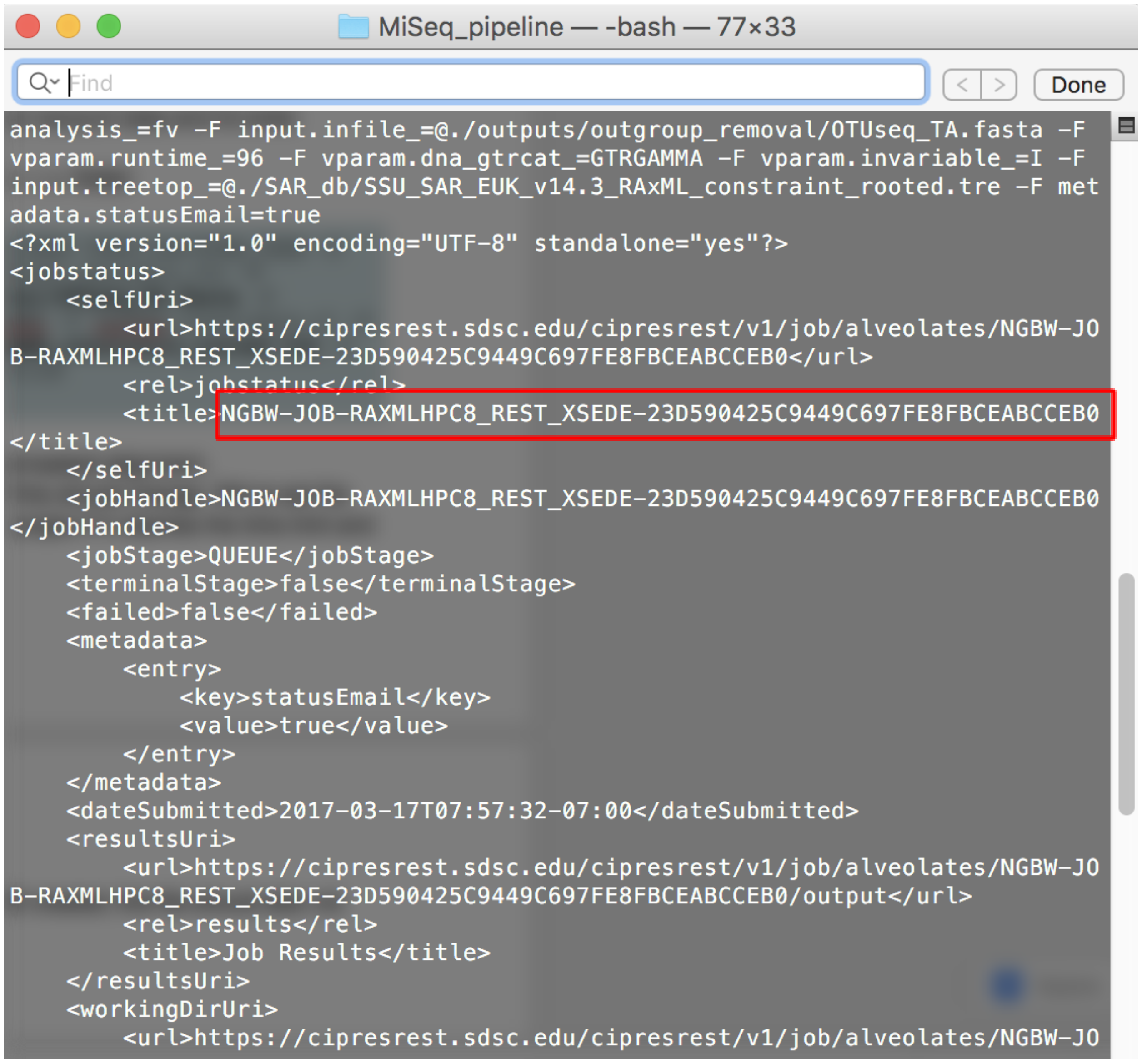
Hit enter. If something was written wrong, you'll receive an error. Make sure the username, password, and application key are all correct.

To run the tree, use this command from your `MiSeq_Pipeline` folder:

```
curl -u $CRA_USER:$PASSWORD -H cipres-appkey:$KEY $URL/job/$CRA_USER -F
tool=RAXMLHPC8_REST_XSEDE -F vparam.select_analysis_=fv -F
input.infile_=@./outputs/outgroup_removal/OTUseq_TA.fasta -F
vparam.runtime_=168 -F vparam.dna_gtrcat_=GTRGAMMA -F vparam.invariable_=I -F
input.treetop_=@./SAR_db/SSU_SAR_EUK_v14.3_RAxML_constraint_rooted.tre -F
metadata.statusEmail=true
```

In bold, I have highlighted the file you use to created the unmasked alignment, `OTUseq_TA.fasta`, and the path to that file. None of this should change, but if you've renamed any of the folders or files at any point you'll have to make sure the path is correct. We've set the runtime to 168 (the maximum time for XSEDE projects), because CIPRES will kill your project if it reaches the time limit and hasn't finished. **Unmasked alignment trees can take days to run.** The last bolded part is the database tree JD created; this file should also not change, just be aware of it.

The terminal will print something like this if the tree was sent successfully.



```
analysis_=fv -F input.infile_=@./outputs/outgroup_removal/OTUseq_TA.fasta -F
vparam.runtime_=96 -F vparam.dna_gtrcat_=GTRGAMMA -F vparam.invariable_=I -F
input.treetop_=@./SAR_db/SSU_SAR_EUK_v14.3_RAxML_constraint_rooted.tre -F met
adata.statusEmail=true
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<jobstatus>
    <selfUri>
        <url>https://cipresrest.sdsc.edu/cipresrest/v1/job/alveolates/NGBW-JO
B-RAXMLHPC8_REST_XSEDE-23D590425C9449C697FE8FBCEABCCEB0</url>
        <rel>jobstatus</rel>
        <title>NGBW-JOB-RAXMLHPC8_REST_XSEDE-23D590425C9449C697FE8FBCEABCCEB0
</title>
    </selfUri>
    <jobHandle>NGBW-JOB-RAXMLHPC8_REST_XSEDE-23D590425C9449C697FE8FBCEABCCEB0
</jobHandle>
    <jobStage>QUEUE</jobStage>
    <terminalStage>false</terminalStage>
    <failed>false</failed>
    <metadata>
        <entry>
            <key>statusEmail</key>
            <value>true</value>
        </entry>
    </metadata>
    <dateSubmitted>2017-03-17T07:57:32-07:00</dateSubmitted>
    <resultsUri>
        <url>https://cipresrest.sdsc.edu/cipresrest/v1/job/alveolates/NGBW-JO
B-RAXMLHPC8_REST_XSEDE-23D590425C9449C697FE8FBCEABCCEB0/output</url>
        <rel>results</rel>
        <title>Job Results</title>
    </resultsUri>
    <workingDirUri>
        <url>https://cipresrest.sdsc.edu/cipresrest/v1/job/alveolates/NGBW-JO
```

The NGBW-JOB_RAXMLHPC8_REST_XSEDE-[numbers] (boxed in red) is your job code.
CIPRES will email you when the job is finished. It will take a few days.
It is vital that you copy down the job code-- this is unique and the only means you have to retrieve your tree once RAxML is finished. I keep all of my information in my `CIPRES_info.txt` file, backed up on Google Drive.

Note: As of May 2017, we have been using **masked alignments exclusively**.

STEP 4

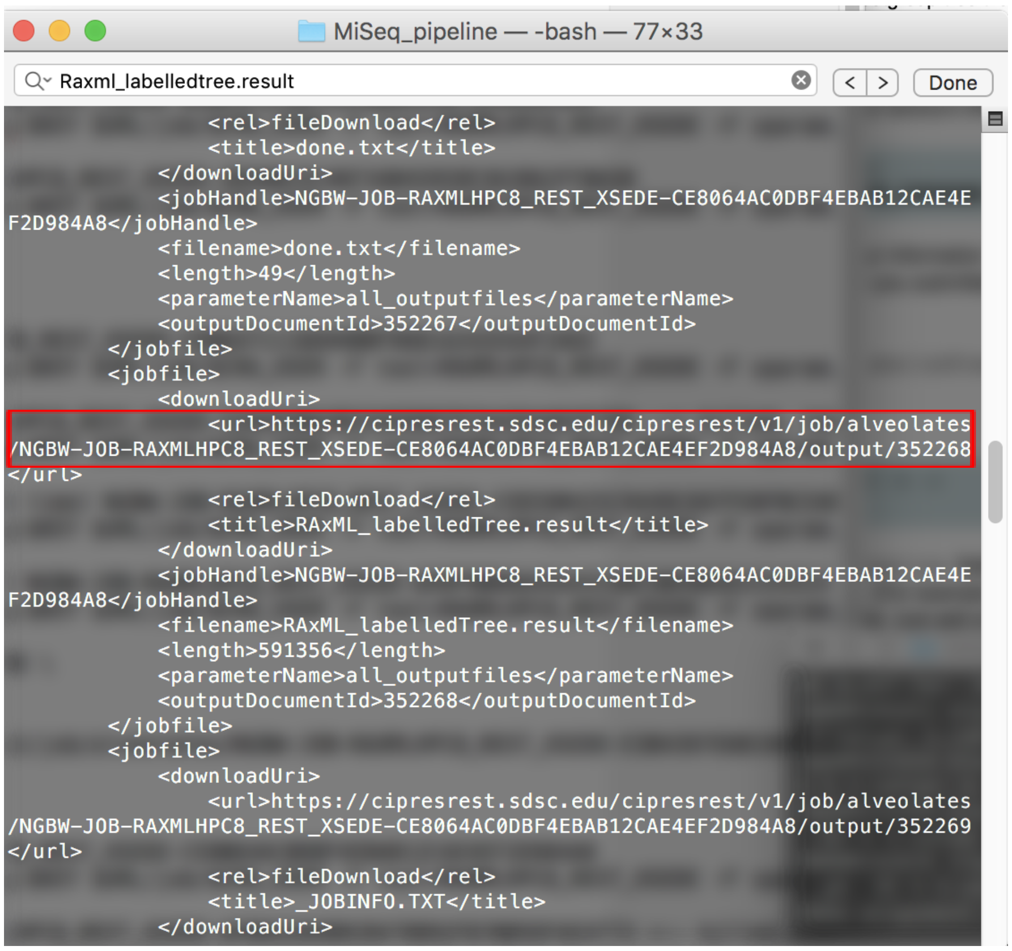
Once your tree is finished running, you'll have to download your results from the terminal and then manually root the tree on the outgroups and create an outgroup tree that will be used for outgroup removal. This takes about 30 minutes to do.

First, run this command to get a list of results. You can do this in the same window as when you submitted the job. If you've closed it, you'll have to export your account information again.

```
curl -u $CRA_USER:$PASSWORD -H cipres-appkey:$KEY
https://cipresrest.sdsc.edu/cit/v1/jobpresres/YOUR_USERNAME/YOUR_ID/output
```

Obviously substitute the bolded parts of the command for your information. Your username is your account username, and your ID the job code from when you submitted the tree. The job code will also be in the email.

This will generate a list of results. Use ⌘F to find `'RAxML_labelledTree.result'`. Copy the entire url, then run this command:



```
                <rel>fileDownload</rel>
                <title>done.txt</title>
            </downloadUri>
            <jobHandle>NGBW-JOB-RAXMLHPC8_REST_XSEDE-CE8064AC0DBF4EBAB12CAE4E
F2D984A8</jobHandle>
            <filename>done.txt</filename>
            <length>49</length>
            <parameterName>all_outputfiles</parameterName>
            <outputDocumentId>352267</outputDocumentId>
        </jobfile>
        <jobfile>
            <downloadUri>
                <url>https://cipresrest.sdsc.edu/cipresrest/v1/job/alveolates
/NGBW-JOB-RAXMLHPC8_REST_XSEDE-CE8064AC0DBF4EBAB12CAE4EF2D984A8/output/352268
</url>
                <rel>fileDownload</rel>
                <title>RAxML_labelledTree.result</title>
            </downloadUri>
            <jobHandle>NGBW-JOB-RAXMLHPC8_REST_XSEDE-CE8064AC0DBF4EBAB12CAE4E
F2D984A8</jobHandle>
            <filename>RAxML_labelledTree.result</filename>
            <length>591356</length>
            <parameterName>all_outputfiles</parameterName>
            <outputDocumentId>352268</outputDocumentId>
        </jobfile>
        <jobfile>
            <downloadUri>
                <url>https://cipresrest.sdsc.edu/cipresrest/v1/job/alveolates
/NGBW-JOB-RAXMLHPC8_REST_XSEDE-CE8064AC0DBF4EBAB12CAE4EF2D984A8/output/352269
</url>
                <rel>fileDownload</rel>
                <title>_JOBINFO.TXT</title>
            </downloadUri>
```

```
curl -u $CRA_USER:$PASSWORD -H cipres-appkey:$KEY -O -J
[YOUR_RAxML_labelledTree.result_URL]
```

If successful, the file will download directly to your `MiSeq_Pipeline` folder. Rename the file accordingly so you are able to find it and easily distinguish it. (For example, `'RAxML_labelledTree_unmasked_Guam.result'`. Just add on the parameters after the initial part.

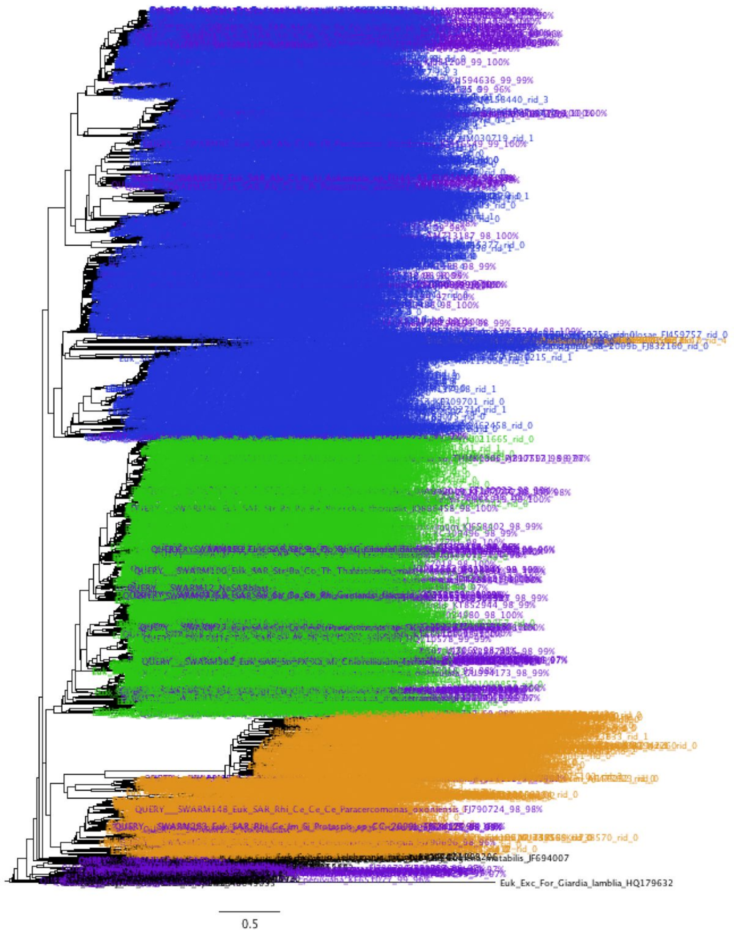
If there is no `RAxML_labelledTree.result` in this list, this means the job failed. Usually it means the job has exceeded the time limit and was killed by CIPRES.

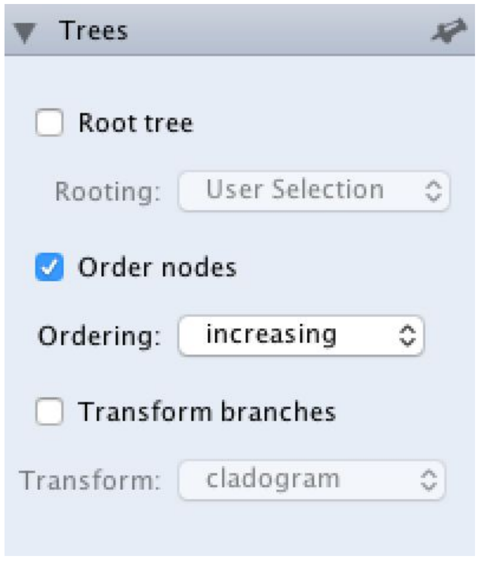
Then you'll open your tree in **Seaview.** It will look messy, but this is just a temporary step. Go to File > Save unrooted tree. Save your tree with the same filename as above, but change **`.result`** to **`.tre`**.

Then open your file in **FigTree**. To make visualization easier, I like to assign colors to the SAR groups and the OTUs. To do this, search for each clade and then use the Color wheel on the top menu bar. Typically, we assign as follows (search the term in bold to highlight all the taxa in that group):

- Alveolates (**Euk_SAR_Alv**): Blue
- Stramenopiles (**Euk_SAR_Str**): Green
- Rhizaria (**Euk_SAR_Rhi**): Orange
- SWARM OTUs (**SWARM**): Purple (hint: do this one last)

Once you have the colors in place, it will be easier to visualize the outgroup. I like to play around with the rotate function until all of the outgroup sequences and OTUs are at the bottom of the tree.





Root tree [ ]
Rooting: User Selection
Order nodes [✓]
Ordering: increasing
Transform branches [ ]
Transform: cladogram

Then, click the drop down ‘Trees’ and check ‘Order nodes.’ This will help you isolate the outgroup sequences. Find the root and click ‘reroot.’ Then, with that root selected, copy the sequences (⌘C) and open a new file (⌘N) and paste the outgroup tree (⌘V). Then go to **File > Export Trees.**

Export the tree as a **Newick** file. Save it as `‘RAxML_labelledTree_unmasked_[your project]_outgroup’`.

From here, you’re ready to move on to the final part of the pipeline.

STEP 5

This is labelled ‘part 3’ of the MiSeq scripts; this will remove any outgroup sequences, subsample, and create your final OTU table that you can use for your analyses. Similar to the previous two automated scripts, it’s pretty straightforward, though you’ll have to wait a few minutes before leaving it alone because there is a subsampling prompt after outgroup removal.

Move your outgroup tree from the previous step to the `outgroup_removal` folder.

```
cd Desktop/MiSeq_pipeline
```

```
python3 MiSeq_pipeline_SAR_SWARM_part3.py
```

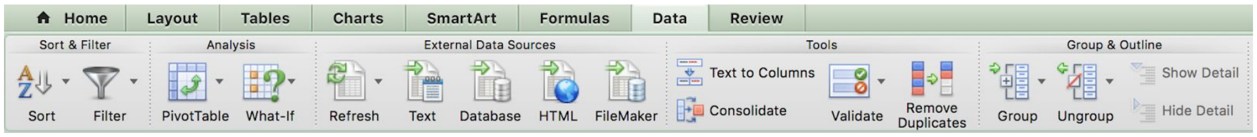
You’ll be asked a series of prompts afterwards.

Prompt	Input
Where is your **raw data output file**?	Drag the **MiSeq_pipeline folder icon** into the terminal window, hit enter.
What is your **list of your samples**?	Drag **`List_samples.txt`** onto the terminal
What is the **name of your outgroup tree**?	**Manually type** the name of the tree from **STEP 4** *****(RAxML_labelledTree_[project]_outgroup)
Do you want to **subsample your reads**?**	Type **yes**
How many **reads**?***	Typically 70,000****

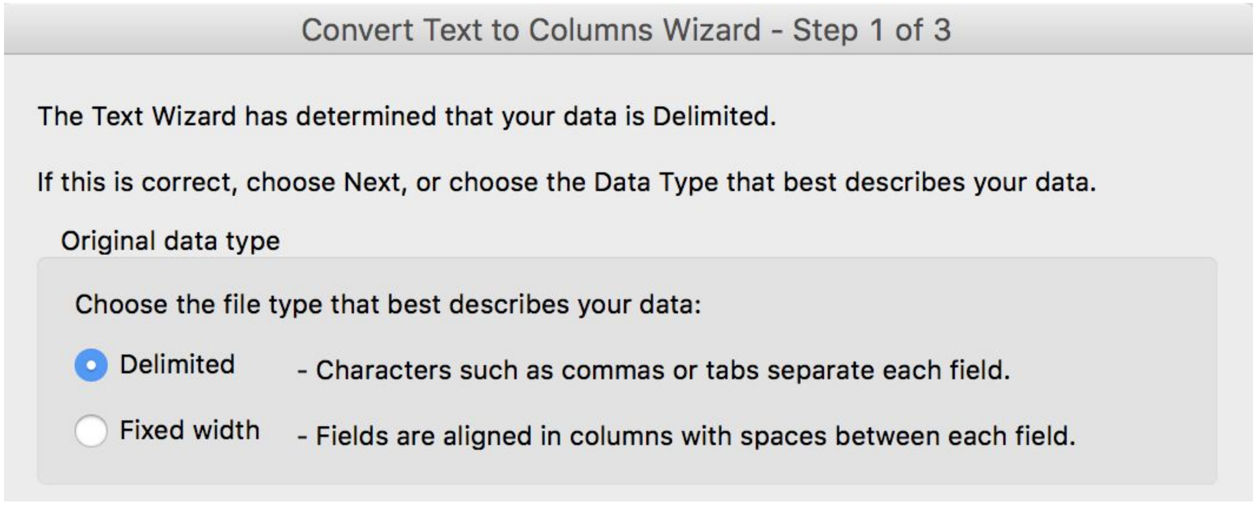
* Do not drag the file; typing the entire path will crash the script.

** The script will print the number of remaining reads for each of your samples. Use this to determine the number of reads you’ll subsample.

*** The easiest way to quickly assess how many reads you should set the subsample threshold at is to copy and paste the “[sample] has [#####] reads.” from the terminal into excel. You can easily create a manipulatable spreadsheet by choosing **Text to Columns** under the **Data** tab.



From there, choose **Delimited** and hit **Next >**



Convert Text to Columns Wizard - Step 1 of 3

The Text Wizard has determined that your data is Delimited.

If this is correct, choose Next, or choose the Data Type that best describes your data.

Original data type

Choose the file type that best describes your data:

[✓ Delimited] - Characters such as commas or tabs separate each field.

Fixed width - Fields are aligned in columns with spaces between each field.

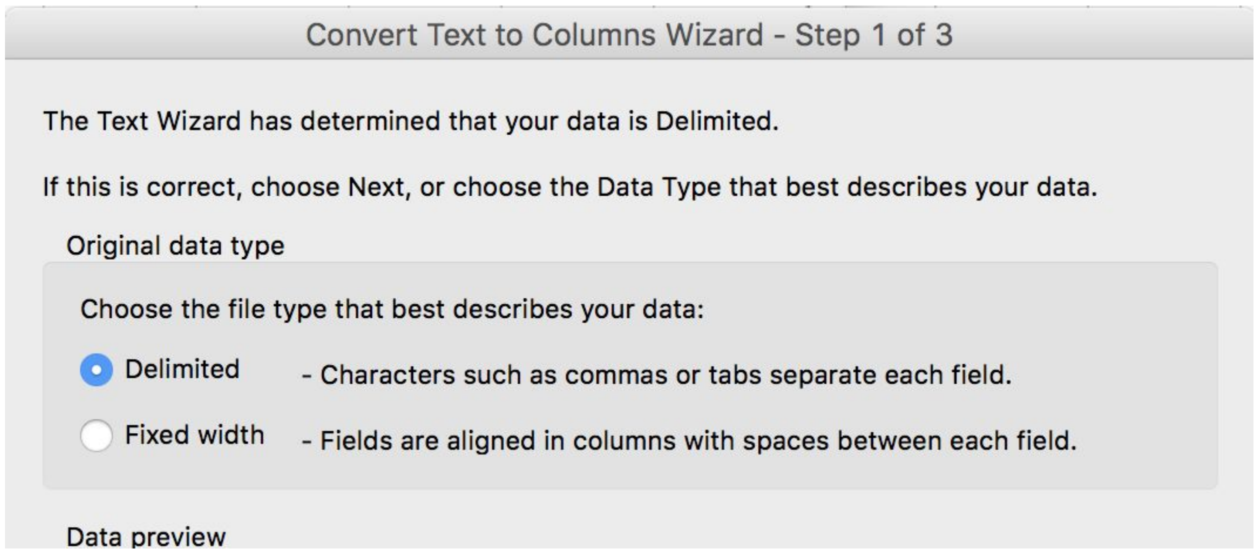
For this input, you'll want to select **space** as your delimiter. Click finish; your data will now appear in 4 separate columns. From here you can delete the columns with 'has' and 'reads.' so you can easily sort the table by read number **(⌘+Shift+R)**

**** Anything under the threshold you input **will be considered in its entirety**. Keep this in mind-- if you have two samples with 30,000 reads but the rest have 120,000, it's worth it to set the subsampling to capture a greater number of reads. In your manuscript, figures, and tables, you'll have to denote which samples are below the threshold. Talk to Chip, JD, or Laura if you have questions.

The script will take a few days to run, but at the end you'll be rewarded with a finalized OTU table. Remember the database hits are not necessarily remotely close to the true identity of the organism-- double check the most abundant OTUs and any unique samples by seeing where the OTU falls in your tree. And get to work knowing thy data!

Data Analysis: Extracting monophyletic clades from phylogenetic trees

1. Open your tree in FigTree and root on the group you intend to extract (ie, Ciliates)
2. Copy and paste your data in a new tree.
3. Export as NEWICK file
4. Open NEWICK in TextWrangler
5. Find (⌘F) and Replace all:
    a. ‘(‘ (forward parenthesis) with nothing
    b. ‘)’ (reverse parenthesis) with nothing
    c. ‘’’ (apostrophe) with nothing
    d. ‘,’ (comma) with ‘\n’ (backslash n)
    e. ‘QUERY___’ (QUERY followed by 3 underscores) with nothing
6. Copy info and paste in Excel
7. In Excel: select the ‘Data’ tab
8. Select ‘Text to Column’
9. Choose ‘Delimited’


Convert Text to Columns Wizard - Step 1 of 3

The Text Wizard has determined that your data is Delimited.

If this is correct, choose Next, or choose the Data Type that best describes your data.

Original data type

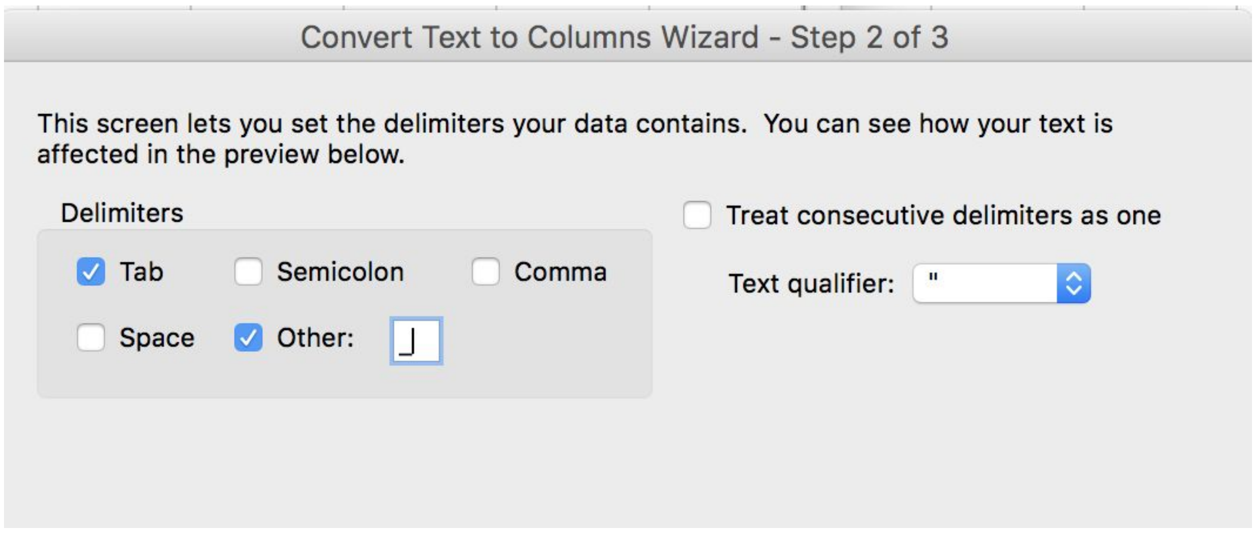
Choose the file type that best describes your data:

[✓ Delimited] - Characters such as commas or tabs separate each field.

Fixed width - Fields are aligned in columns with spaces between each field.

Data preview

10. Hit ‘Next >’


Convert Text to Columns Wizard - Step 2 of 3

This screen lets you set the delimiters your data contains. You can see how your text is affected in the preview below.

Delimiters

[✓] Tab
[ ] Semicolon
[ ] Comma
[ ] Space
[✓] Other: _

[ ] Treat consecutive delimiters as one

Text qualifier: "

11. Select ‘Tab’ and ‘Other’, input ‘_’ into the other field and click finish.
12. Sort the spreadsheet A-Z, remove any entries beginning with ‘Euk’
13. Delete extraneous data
14. Go to final spreadsheet to extract matching OTU information

Some useful websites/manuscripts for background:

PEAR stands for Paired-End reAd mergeR:

- Look here:
- Read this paper:
-

Know thy data-- launching your analyses

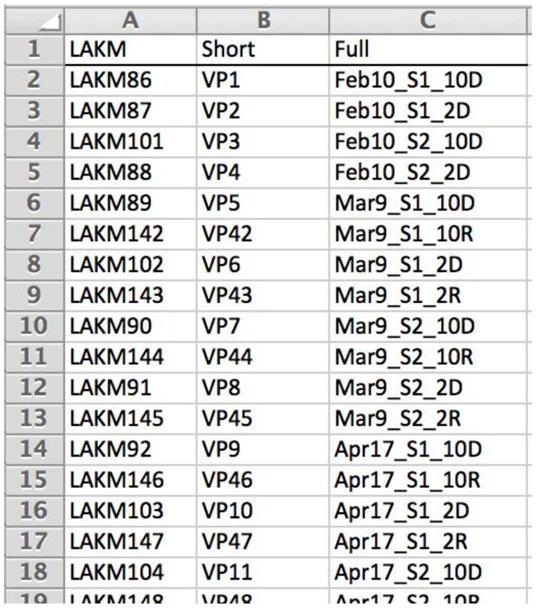
Once the pipeline is finished, you'll be rewarded with a hefty OTU table. Here are some launching points to get you started on your analyses.

Renaming

Use the find and replace function (⌘F) to rename your samples. If you followed earlier instructions, you should have a list pairing the short titles to the long ones.

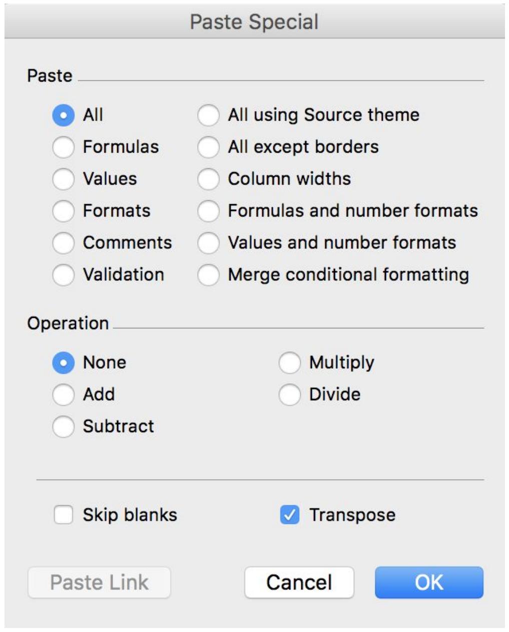
I strongly recommend immediately creating a copy of your OTUtable_final to manipulate and play around with, that way in case anything is irrevocably damaged or unintentionally lost there will be an intact file to use as a clean slate.

You can use this table to quickly rename your samples. First, select all the sample names in the 'Full column' and copy (⌘C).



	A	B	C
1	LAKM	Short	Full
2	LAKM86	VP1	Feb10_S1_10D
3	LAKM87	VP2	Feb10_S1_2D
4	LAKM101	VP3	Feb10_S2_10D
5	LAKM88	VP4	Feb10_S2_2D
6	LAKM89	VP5	Mar9_S1_10D
7	LAKM142	VP42	Mar9_S1_10R
8	LAKM102	VP6	Mar9_S1_2D
9	LAKM143	VP43	Mar9_S1_2R
10	LAKM90	VP7	Mar9_S2_10D
11	LAKM144	VP44	Mar9_S2_10R
12	LAKM91	VP8	Mar9_S2_2D
13	LAKM145	VP45	Mar9_S2_2R
14	LAKM92	VP9	Apr17_S1_10D
15	LAKM146	VP46	Apr17_S1_10R
16	LAKM103	VP10	Apr17_S1_2D
17	LAKM147	VP47	Apr17_S1_2R
18	LAKM104	VP11	Apr17_S2_10D
19	LAKM148	VP48	Apr17_S2_10R

Then, in your copy of your OTUtable_final excel sheet, right click on the first sample name and choose **Paste special**. At the bottom right of the popup box, there will be a box next to 'Transpose'. Make sure this is selected.



Paste Special

Paste
[✓ All] All using Source theme
Formulas All except borders
Values Column widths
Formats Formulas and number formats
Comments Values and number formats
Validation Merge conditional formatting

Operation
[✓ None] Multiply
Add Divide
Subtract

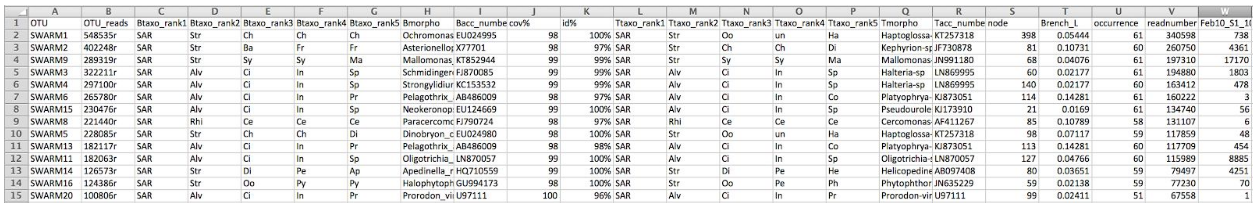
Skip blanks [✓ Transpose]

Hit ok, and your samples will now have their full names.

Additionally, per Laura's preference, use the find and replace function (⌘F) to rename every instance of 'SWARM' with 'OTU', as 'OTU' is a defined term recognizable by the wider biologist community. 'SWARM' only means something to those that use that particular clustering method.

Understanding the table

Now that your samples are renamed, you can dive into the table and start creating graphs to aid your initial analyses. Remember as you're getting to know your data, do not look to confirm suspicions or hypotheses. Nearly every peer-reviewed scientific paper edits their hypotheses in post to match what the data presents, there's no shame in doing this. Try to let the data 'tell you' what the stories are.



	A	B	C	D	E	F	G	H	I	J	K
1	OTU	OTU_reads	Btaxo_rank1	Btaxo_rank2	Btaxo_rank3	Btaxo_rank4	Btaxo_rank5	Bmorpho	Bacc_numbe	cov%	id%
2	SWARM1	548535r	SAR	Str	Ch	Ch	Ch	Ochromonas	EU024995	98	100%
3	SWARM2	402248r	SAR	Str	Ba	Fr	Fr	Asterionello	X77701	98	97%
4	SWARM9	289319r	SAR	Str	Sy	Sy	Ma	Mallomonas_	KT852944	99	99%
5	SWARM3	322211r	SAR	Alv	Ci	In	Sp	Schmidinger	FJ870085	99	99%
6	SWARM4	297100r	SAR	Alv	Ci	In	Sp	Strongylidiur	KC153532	99	99%
7	SWARM6	265780r	SAR	Alv	Ci	In	Pr	Pelagothrix_	AB486009	98	97%
8	SWARM15	230476r	SAR	Alv	Ci	In	Sp	Neokeronop	EU124669	99	100%
9	SWARM8	221440r	SAR	Rhi	Ce	Ce	Ce	Paracercomc	FJ790724	98	97%
10	SWARM5	228085r	SAR	Str	Ch	Ch	Di	Dinobryon_c	EU024980	98	100%
11	SWARM13	182117r	SAR	Alv	Ci	In	Pr	Pelagothrix_	AB486009	98	98%
12	SWARM11	182063r	SAR	Alv	Ci	In	Sp	Oligotrichia_	LN870057	99	100%
13	SWARM14	126573r	SAR	Str	Di	Pe	Ap	Apedinella_r	HQ710559	99	100%
14	SWARM16	124386r	SAR	Str	Oo	Py	Py	Halophytoph	GU994173	98	100%
15	SWARM20	100806r	SAR	Alv	Ci	In	Pr	Prorodon_vi	U97111	100	96%

	L	M	N	O	P	Q	R	S	T	U	V	W
1	Ttaxo_rank1	Ttaxo_rank2	Ttaxo_rank3	Ttaxo_rank4	Ttaxo_rank5	Tmorpho	Tacc_numbe	node	Brench_L	occurrence	readnumber	Feb10_S1_1
2	SAR	Str	Oo	un	Ha	Haptoglossa-	KT257318	398	0.05444	61	340598	738
3	SAR	Str	Ch	Ch	Di	Kephyrion-sp	JF730878	81	0.10731	60	260750	4361
4	SAR	Str	Sy	Sy	Ma	Mallomonas-	JN991180	68	0.04076	61	197310	17170
5	SAR	Alv	Ci	In	Sp	Halteria-sp	LN869995	60	0.02177	61	194880	1803
6	SAR	Alv	Ci	In	Sp	Halteria-sp	LN869995	140	0.02177	60	163412	478
7	SAR	Alv	Ci	In	Co	Platyophrya-	KJ873051	114	0.14281	61	160222	3
8	SAR	Alv	Ci	In	Sp	Pseudourole	KJ173910	21	0.0169	61	134740	56
9	SAR	Rhi	Ce	Ce	Ce	Cercomonas-	AF411267	85	0.10789	58	131107	6
10	SAR	Str	Oo	un	Ha	Haptoglossa-	KT257318	98	0.07117	59	117859	48
11	SAR	Alv	Ci	In	Co	Platyophrya-	KJ873051	113	0.14281	60	117709	454
12	SAR	Alv	Ci	In	Sp	Oligotrichia-s	LN870057	127	0.04766	60	115989	8885
13	SAR	Str	Di	Pe	He	Helicopedine	AB097408	80	0.03651	59	79497	4251
14	SAR	Str	Oo	Pe	Ph	Phytophthor	JN635229	59	0.02138	59	77230	70
15	SAR	Alv	Ci	In	Pr	Prorodon-vir	U97111	99	0.02411	51	67558	1

Your table will look something like this, with thousands and thousands of rows (hopefully!) In this section I'll provide a brief explanation as to what each of the columns mean.

Position	Title	Description
A1	SWARM (renamed to OTU)	The name of each OTU generated by the SWARM algorithm.
B1	OTU_reads	The number of reads in each SWARM OTU over the entire project
C1-G1	Btaxo_rankX	Taxonomy by rank based on best BLAST hit
H1	Bmorpho	Best BLAST hit
I1	Bacc_number	GenBank accession number of the best BLAST hit
J1	cov%	Coverage of best BLAST hit
K1	id%	Percent identity of best BLAST hit
L1-P1	Ttaxo_rankX	Taxonomy by rank based on closest sister within gene tree generated by RAxML
Q1	Tmorpho	Closest sister morphospecies on tree
R1	Tacc_number	GenBank accession number of closest sister (in SAR database)
S1, T1	Node, Brench_L	Position information on tree (rarely used for your analyses)

U1	occurrence	The number of your samples this OTU appears in
V1	readnumber	The total number of reads of this OTU, subsampled
W1-XX1	[sample name]	Read counts of each OTU by sample

Generally, we use the RAxML tree to assign taxonomy, so you can hide columns B-K to make the spreadsheet a bit easier to digest. Additionally, you can hide S-T if you'd like.

A note about taxonomy: remember, protist taxonomy is messy and convoluted. Taxonomic 'ranks' may not be equivalent across groups, so take caution when comparing different clades.