GUIDE FOR PROCESSING POPULATION GENOMIC DATA ON A HIGH PERFORMANCE COMPUTING CLUSTER (HPC) at TAMUCC

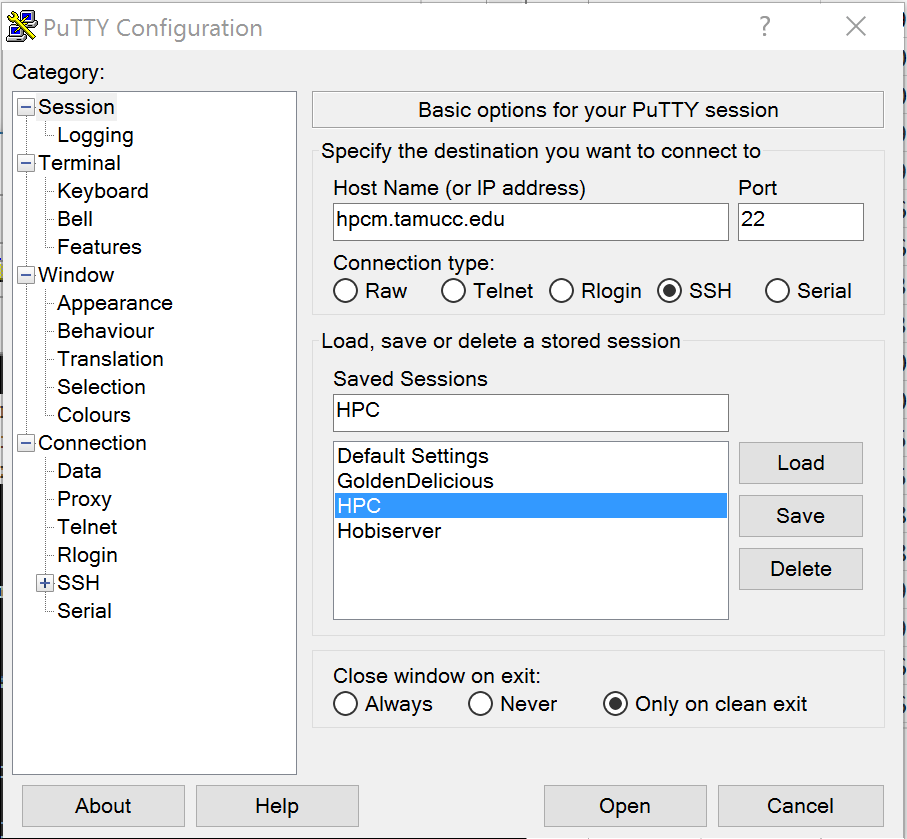
© Dr. Christopher E. Bird, *cbird@tamucc.edu*

Contributions made by: Dr. Jon Puritz, Dr. Rebecca Hamner, Dr. Amanda Ackiss, Dr. Brian Stockwell, Ms. Sharon Furiness, Ms. Luz Lopez de Mesa, Mr. Evan Krell, Mr. Martin French, Ms. Maili Allende, Ms. Cynthia Soliz, Mr. Mitch Burr

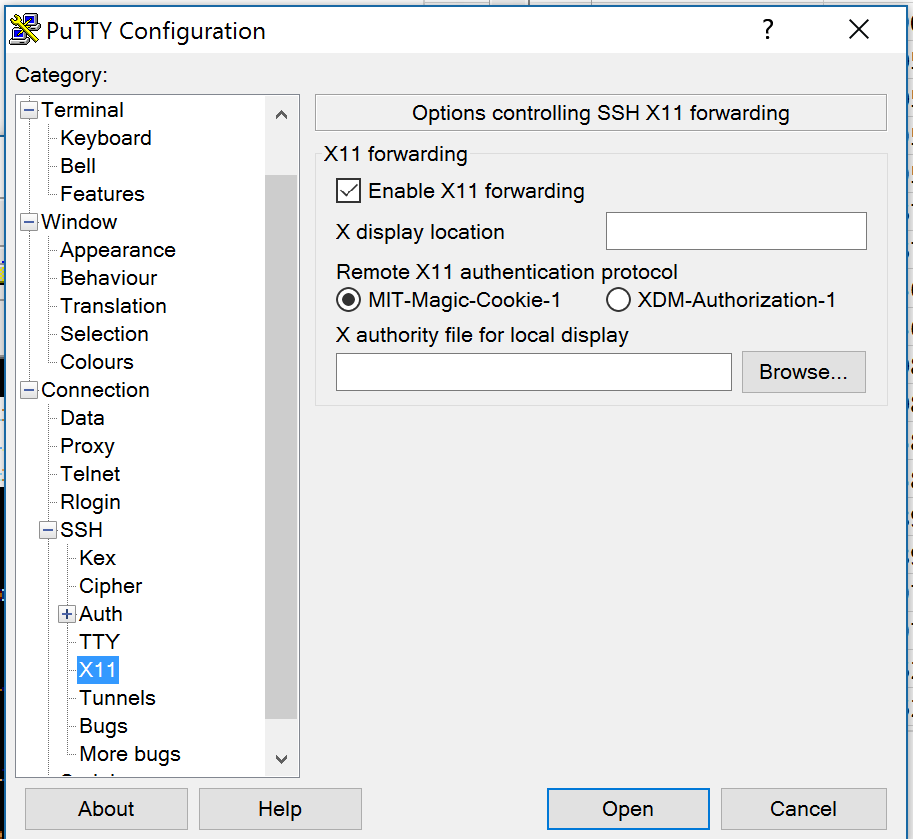
Supported in part by NSF-CNS-MRI and NOAA Saltonstall-Kennedy Award NA15NMF4270330

*Dr. Jonathan Puritz made dDocent 2.24, a BASH script that is used heavily here, but it has been modified substantially by CEB. Please contact Dr. Bird with questions about script functioning in this guide because if something doesn’t work right, it’s probably from modifications that he made.*

1. Interfacing With HPC
2. HPC Basics
3. Standardized Directory Format
4. FASTQC
5. Counting Reads
6. Demultiplex
7. dDocent Basics
8. Trim
9. Assembly I
10. Assembly II
11. Mapping
12. Genotyping
13. Filtering
14. Cleaning Up Directories and Files
15. Appendices
16. **Interfacing With HPC**
    1. Request and account from [thomas.merrick@tamucc.edu](mailto:thomas.merrick@tamucc.edu)
    2. If you are a student or off campus, you will need VPN access
       1. Ithelp can help you get started
       2. When they tell you no, come see CEB
    3. Windows
       1. Install xming to run “mouse” programs on hpc
          1. See appendix for install guide
          2. Turn xming on
       2. Install putty to interface with hpc
          1. See appendix for install guide
          2. Settings
             1. Host Name: hpcm.tamucc.edu



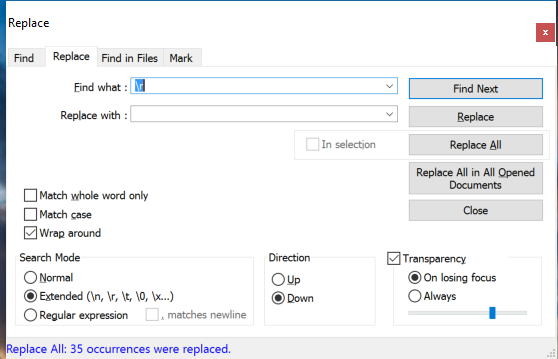
* + - * 1. Set: SSH,X11, enable



* + 1. Install ssh secure shell to interface with hpc, drag and drop between HPC and PC
       1. Install instructions
          1. <https://sils.unc.edu/it-services/servers/using-ssh>
       2. HPC address:
          1. hpcm.tamucc.edu
       3. User Name
          1. Yourislanderid
    2. Install notepad++
       1. This program is good for editing scripts and viewing the files that you create on the hpc, such as NumRead.txt files
       2. Set the Language to Shell
          1. This is a very nice way to view and edit bash scripts on your pc
       3. Set the whitespace to linux/unix
          1. Goto View, Show Symbol, End of Line

There should only be LF at the end of a line

If you have LF CR, then do a find and replace like this:



* + - * 1. If you don’t do this, scripts written on your computer probably won’t run on the hpc
        2. If you are curious as to why, see here: <http://www.cs.toronto.edu/~krueger/csc209h/tut/line-endings.html> I
  1. Mac
     1. Open terminal
     2. Type
        1. ssh –X [yourislanderid@hpcm.tamucc.edu](mailto:yourislanderid@hpcm.tamucc.edu)
     3. Use textwrangler to edit bash scripts
  2. Linux
     1. Open terminal
     2. Type
        1. ssh –X [yourislanderid@hpcm.tamucc.edu](mailto:yourislanderid@hpcm.tamucc.edu)

1. **HPC Basics**

The hpc operating system is RedHatEnterprise Linux. Most scripts are written in the bash shell scripting language. The queue system is called slurm and responds to sbatch scripting language. These terms will likely be useful when you’re searching for answers.

* 1. All questions about achieving a task should be asked via the hpc forum at hpc.tamucc.edu
     1. Please use this resource, we worked hard to make it for your benefit
     2. The forum can be searched for answers and contains tutorials
  2. Google is your friend, try googling the solution to your problem
  3. The mouse will rarely work
     1. In putty, selecting with mouse automatically copies selection
     2. In putty, right clicking the mouse pastes
     3. That’s it for mousing in a terminal window
  4. See tutorial on HPC forum at hpc.tamucc.edu
  5. Each program you run must be loaded with following command
     1. module load programname
  6. You can view programs available with following command
     1. module avail
  7. Programs should be run from bash scripts
     1. There are special sbatch commands that need to be listed in the script
     2. Run the script as follows:
        1. sbatch script.sh
     3. View running jobs with following command
        1. squeue
     4. Cancel running job with following command
        1. scancel jobid
     5. Track progress of script in the slurm\*.out file
        1. The slurm\*.out file contains all info that would normally be printed to the screen
        2. open with the following command
           1. less slurmfilename
     6. see the hpc cheat sheet at hpc.tamucc.edu
  8. See the linux command line cheat sheet at hpc.tamucc.edu
     1. Files can be viewed with less and zless
        1. Zless is for zipped files
        2. You can’t edit, only view for large files, like fq.gz files
     2. Editing scripts with nano
        1. Nano is a word processor
        2. It’s clunky, but it gets the job done
        3. Open an existing script or create a new script
           1. nano scriptname.sh

1. **Standardized Directory Format**

You are about to create a lot of files that will take up 10s to 1000s of gigabytes of space. It is imperative that a standardized directory format be followed so that we can all find the files and information we need, and so that dDocent will run properly.

* 1. Work directory
     1. All of your work should be saved in your work directory which was created when you requested an HPC account
     2. Navigate to your work directory
        1. cd $WORK
           1. $WORK is a variable that stores the path to your work directory

/work/GenomicsSamples/yourislanderid

* 1. Script directory
     1. You will be using many scripts. Generally, you will need to modify the scripts for a specific purpose, and those scripts should be stored within you project directory (see below) where they were executed to document what was done to the files. However, you should keep one example of each script in your script directory so that it is easy to find the scripts and share them.
     2. Examples of all of the scripts in this manual can be found in the following directory
        1. /work/GenomicSamples/cbird/scripts
     3. Make a directory named scripts in your work directory
        1. Mkdir scripts
     4. Copy the scripts from my script directory to your script directory
        1. cp /work/GenomicSamples/cbird/scripts/\* ./scripts
        2. mkdir scripts/ddocent
        3. cp /work/GenomicSamples/cbird/scripts/ddocent/\* ./scripts/ddocent
  2. Project directory
     1. Each project should have its own directory
     2. Subdirectories
        1. Assembly
           1. Fastqc
           2. Logfiles
           3. Trimreports
        2. Fastq
        3. Fastqc
        4. highReadNum
        5. lowReadNum
        6. mapping
           1. Fastqc
           2. Logfiles
           3. lowReadNum
           4. Trimreports
           5. unpairedreads
        7. removed\_seqs
        8. filtering

#####################################################################################

Note: the rest of the information in this chapter is only for reference. You are not expected to execute each step.

####################################################################################

* 1. Each directory will harbor particular files and particular scripts should be run in particular directories.
     1. Project Directory
        1. Raw (demultiplexed) \*F.fq.gz and \*R.fq.gz files
           1. These files should follow a std naming convention

ID\_Treatment\_PlateXPoolY.F.fq.gz

* + - * 1. The file extensions are important for dDocent.
        2. You will probably have to modify the file names to conform using the rename function
      1. The following scripts will be run from this directory or should be in this directory
         1. dDocent224tref\_hpc3.bash

config.trim

* + - * 1. cntReads.sh
        2. ddRAD\_demultiplex.sh
        3. runFASTQC.sh
        4. trimFiles.sh
      1. Demultiplexing files (ddRAD only)
         1. Demultiplex\_PlateXPoolY.txt
         2. Process\_PlateXPoolY\_radtags.log
    1. Assembly
       1. All files associated with reference genome assembly will be stored here, including the all important reference.\*.fasta\* files
       2. The subdirectories’ purpose are similar to similarly named subdirectories in the project directory
       3. You may make assemblies from different sets of files, most typically from different plate x pool combinations. In this case, there should be several assembly directories that follow this naming convention
          1. assemblyPlateXPoolY
    2. fastq (ddRAD only)
       1. this is where the original sequence files (not demultiplexed) should be stored after they are demultiplexed
    3. FASTQC
       1. Files generated by fastqc on the (demultiplexed) files should be stored here
    4. highReadNum
       1. Libraries with too many reads are stored and processed here, not in the project directory with the other .fq.gz files
    5. lowReadNum
       1. Libraries with too few reads are stored and processed here, not in the project directory with the other .fq.gz files
    6. Mapping
       1. This is where all files associated with mapping are stored
       2. \*.bam and \*.vcf files are generated here
       3. Subdirectories have similar purpose to those in project directory
    7. Removed\_seqs (ddRAD only)
       1. This is where reads that were tossed by demultiplexing are stored
    8. Filtering
       1. Post genotyping filtering and manipulation of \*.vcf files should occur here
  1. Access to directories is important. I need to be able to access your directories but you do not want a novice to inadvertently mess up your finely curated directory structure and files
     1. Navigate to /work/GenomicSamples with cd
     2. Then change permissions so that people in our lab can see your directories and files, but not change them
        1. chmod –R 750 /work/GenomicSamples/yourislanderid
     3. If I ask you to give me full access, you need to do the following
        1. chmod –R 770 /work/GenomicSamples/yourislanderid

1. **FASTQC & MultiQC**

Fastqc is run to evaluate overall quality of the base calls and other broad patterns in the sequence data. Fastqc should be run immediately upon receiving sequences. MultiQC is a tool to aggregate fastqc output from multiple files

* 1. Script: runFASTQC.sh
     1. Copy the script to the folder where files will be fastqc’d
     2. Ensure that the script, directories, and files are compatible
     3. Run: sbatch runFASTQC.sh

#!/bin/bash

#SBATCH --job-name=fqcFDA

#SBATCH --time=24:00:00

#SBATCH -p normal

#SBATCH --nodes=1

#SBATCH --mail-user=islanderid@tamucc.edu

#SBATCH --mail-type=begin # email me when the job starts

#SBATCH --mail-type=end # email me when the job finish

module load fastqc

module load parallel/20160722

#run fastqc in parallel

ls \*001.fq.gz | parallel "fastqc {}"

#make a directory called FASTQC and move the FASTQC files into the directory

mkdir FASTQC

ls \*fastqc.html | parallel "mv {} FASTQC"

ls \*fastqc.zip | parallel "mv {} FASTQC"

* 1. Multiqc can be supplied with a directory or list of files using wildcards
     1. Run multiqc on fastq data just generated above
        1. module load multiqc/0.9
        2. multiqc FASTQC/
  2. Evaluate results
     1. Are base call quality scores low?
     2. Do forward reads have lower scores than reverse?
     3. Are there odd patterns in the data?
     4. Note the with ddRAD and ezRAD there will likely be the same motif at the beginning and end of the sequences. This is expected, but fastqc doesn’t like it.

1. **Count Reads**

The purpose of counting the reads in each file is to make sure that there are an adequate number of reads and that the operations performed on the data (demultiplexing and trimming) did not result in excessive loss of data. Consequently, reads should be counted when the sequences are received and after each step that alters the fastq/fasta files. The read counts should be logged in a table, where each row represents one file, and a new column or read counts is added for each step.

* 1. Script: cntReads.sh
     1. Copy script to the directory containing the fastq files to be quantified
        1. cp
     2. Modify directories, files, and the script so that they are compatible
     3. Run script
     4. Extract data to R or Excel

#!/bin/bash

#SBATCH --job-name=cntRiginos

#SBATCH --time=24:00:00

#SBATCH -p normal

#SBATCH --nodes=1

#SBATCH --mail-user=cbird@tamucc.edu

#SBATCH --mail-type=begin # email me when the job starts

#SBATCH --mail-type=end # email me when the job finish

module load parallel/20160722

#count reads

ls \*.gz | parallel "echo -n {}' ' && zgrep -c '^\+$' {}"| sort -g >> NumPreReads.txt

ls mapping/\*.gz | parallel "echo -n {}' ' && zgrep -c '^\+$' {}" | sort -g >> mapping/NumMapReads.txt

ls assembly/\*.gz | parallel "echo -n {}' ' && zgrep -c '^\+$' {}" | sort -g >> assembly/NumAssmReads.txt

* 1. At what steps should reads be counted and how should the reads be interpreted?
     1. After FASTQC
        1. Fastq files should be quantified prior to manipulation. FASTQC only gives quality stats.
        2. The goal is to determine that an adequate number of sequence reads were obtained from the sequencing facility
        3. Read counts should be compared with the report from the sequencing facility. It is not unusual to obtain too few reads in sequenced libraries and these will need to be remade and resequenced.
           1. If this is ezRAD data,

we generally target 7mil reads per library

How even are the read counts between libraries?

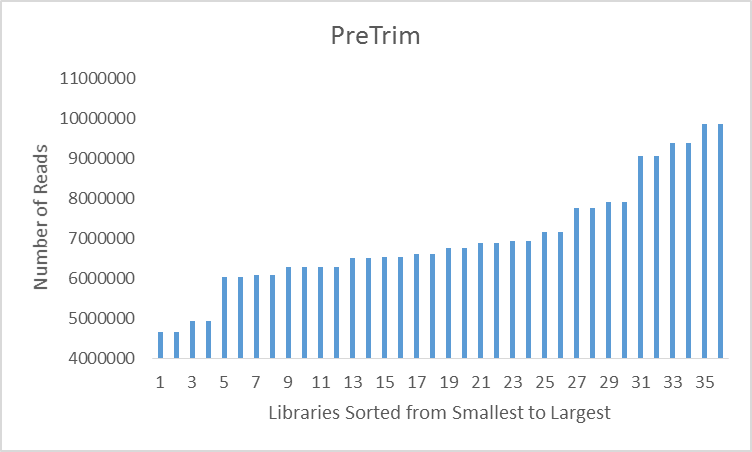
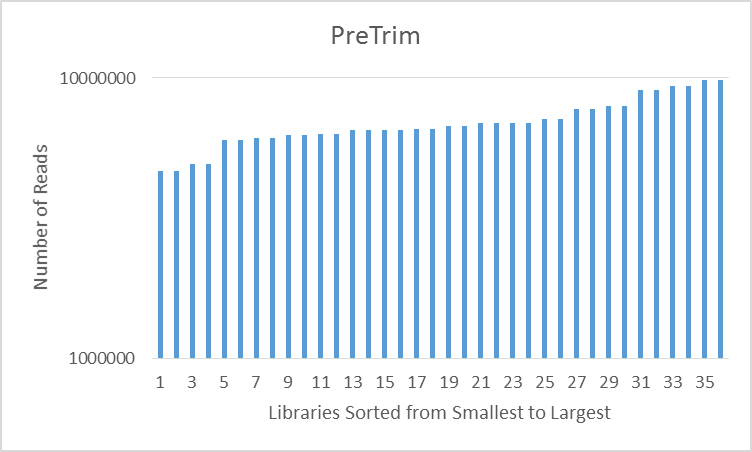
Are there too many or too few sequences?

Make graphical representations of the data

Bar plot of read count vs fastq file, sorted from least to most

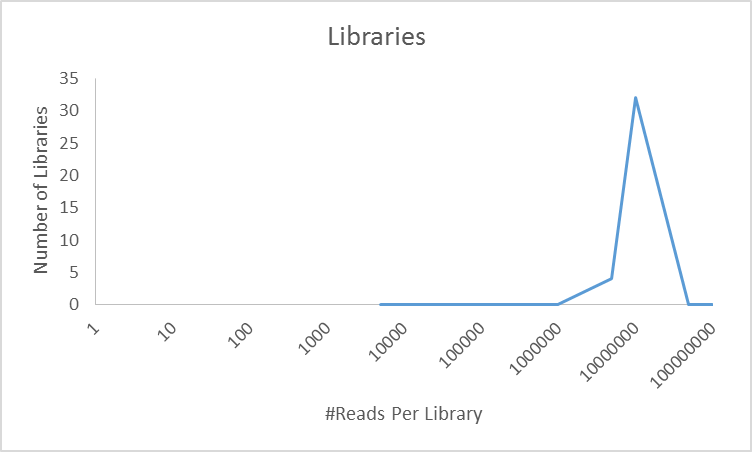
In this example, both plots are of the same data, difference is log scale (left) and linear scale (right)

Even the smallest libraries are probably adequate



Histogram of the number of reads per file

Bins to use

5000

10000

50000

100000

500000

1000000

5000000

10000000

50000000

100000000

Files that have too few reads can be moved to a directory called lowReadNum

mkdir lowReadNum

display files in order from largest to smallest

ls –lS

find files between x byte and y bytes

find . –maxdepth 1 –size +55000c –a –size -260000c

if that grabs the files you want to move, then do this:

find . -maxdepth 1 -size +55000c -a -size -260000c -exec mv {} lowReadNum/ \;

* + 1. Files that have too many reads can be moved to a directory called highReadNum
       1. mkdir highReadNum
       2. Use the commands described for moving files of a particular size range above
       3. Large files can be divided into smaller files and treated as library replicates
          1. Script: splitFqFiles.sh

#!/bin/bash

#SBATCH --job-name=splitFiles

#SBATCH --time=96:00:00

#SBATCH -p normal

#SBATCH --nodes=1

#SBATCH --mail-user=cbird@tamucc.edu

#SBATCH --mail-type=begin # email me when the job starts

#SBATCH --mail-type=end # email me when the job finish

module load parallel/20160722

echo "", echo `date` " unzip files"

ls \*.fq.gz | parallel "gunzip {}"

NumFiles=( `ls \*.fq | grep -c '^[0-9]' `)

echo "NumFiles=$NumFiles"

ZeroToNumFiles=( `seq 0 $(($NumFiles - 1)) `)

echo ${ZeroToNumFiles[\*]}

splitFiles() {

FileNames=( `ls \*.fq `) #load list of filenames

echo ${FileNames[\*]}

NumLines=( `cat "splitAmounts.txt" `) #load list of number of lines per chunk per file

echo ${NumLines[\*]}

mkdir $1 #make directory (1-Numfiles)

cd $1

split -d -l ${NumLines[$1]} ../${FileNames[$1]}

xfiles=( `ls x\* `)

echo ${xfiles[\*]}

Numxfiles=( `ls x\* | grep -c '^x' `)

echo $Numxfiles

for i in ${xfiles[\*]}; do mv $i ../$i-${FileNames[$1]};done

cd ..

rm -rf $1

}

export -f splitFiles

parallel splitFiles ::: ${ZeroToNumFiles[\*]}

ls \*.fq | parallel gzip {}

* + - * 1. Calculate the desired number of lines per file

Choose a target number of reads per file

Use the mean read count for the acceptable libraries

Then divide the number of true retained reads by 2 million

Round up to the nearest whole integer, do not round down

7612633/2000000 = 3.8 -> 4

Divide True Retained Reads by 4 and round up

7612633/4 = 1903158.3 -> 1903159

The target number of lines per split file is #reads \* 4 because there are 4 lines per read in a fastq file

1903159\*4= 7612636

* + - * 1. Make a file called splitChunks.txt

nano splitChunks.txt

This should have, in the order that files are returned by the ls command, the target number of lines per split file per library. This number will be the same for F and R files.

Example: there are two libraries with fwd and rev reads. The file will read as follows

7612636

7612636

7117024

7117024

* + - * 1. Before running the script, modify it, the filenames, and directory structure so that they are compatible.
        2. Run the script

sbatch splitFqFiles.sh

* + - 1. Lastly select one or more of the split files per library to move back with the files to be trimmed
         1. mv x00\* ../
         2. If this is ddRAD data, not much additional can be determined until demultiplexing is performed
    1. After demultiplex (ddRAD data)
       1. See demultiplexing section
       2. Confirm that the “true retained reads” counts from the process\*radtags.log
    2. After trimming
       1. Determine and plot the following for both libraries trimmed for assembly and for mappings
          1. how many reads are left per library?
          2. what proportion of reads were lost?
    3. After reference creation
       1. Use the cntContigs.sh script
          1. Same idea as cntReads.sh, but customized for the reference
          2. See Assembly Part 2, below, for more details

1. **Demultiplex (ddRAD only)**

ddRAD libraries have an internal DNA barcode that is used to identify individuals. The \*.fq.gz files you received from the sequencing facility likely have 48 individual libraries that need to be extracted.

* 1. Demultiplex decode files demultiplex\_PlateXPoolY.txt
     1. There should be one of these files for each PlateXPoolY combination

GCATG 252\_Plate1Pool1

AACCA 463\_Plate1Pool1

CGATC 244\_Plate1Pool1

TCGAT 445\_Plate1Pool1

TGCAT 443\_Plate1Pool1

CAACC 435\_Plate1Pool1

GGTTG 434\_Plate1Pool1

AAGGA 476\_Plate1Pool1

AGCTA 465\_Plate1Pool1

ACACA 464\_Plate1Pool1

AATTA 483\_Plate1Pool1

ACGGT 482\_Plate1Pool1

ACTGG 456\_Plate1Pool1

ACTTC 453\_Plate1Pool1

ATACG 228\_Plate1Pool1

ATGAG 229\_Plate1Pool1

ATTAC 262\_Plate1Pool1

CATAT 441\_Plate1Pool1

CGAAT 484\_Plate1Pool1

CGGCT 233\_Plate1Pool1

CGGTA 266\_Plate1Pool1

CGTAC 268\_Plate1Pool1

CGTCG 448\_Plate1Pool1

CTGAT 450\_Plate1Pool1

CTGCG 438\_Plate1Pool1

CTGTC 486\_Plate1Pool1

CTTGG 454\_Plate1Pool1

GACAC 452\_Plate1Pool1

GAGAT 239\_Plate1Pool1

GAGTC 256\_Plate1Pool1

GCCGT 240\_Plate1Pool1

GCTGA 241\_Plate1Pool1

GGATA 260\_Plate1Pool1

GGCCA 442\_Plate1Pool1

GGCTC 462\_Plate1Pool1

GTAGT 481\_Plate1Pool1

GTCCG 451\_Plate1Pool1

GTCGA 449\_Plate1Pool1

TACCG 234\_Plate1Pool1

TACGT 232\_Plate1Pool1

TAGTA 261\_Plate1Pool1

TATAC 458\_Plate1Pool1

TCACG 290\_Plate1Pool1

TCAGT 267\_Plate1Pool1

TCCGG 257\_Plate1Pool1

TCTGC 254\_Plate1Pool1

TGGAA 253\_Plate1Pool1

TTACC 457\_Plate1Pool1

* 1. Script: ddRAD\_demultiplex.sh
     1. Copy the script to your project directory
     2. Note, this script could be made parallel, where each demultiplex decode file is run on a different thread

#!/bin/bash

#copy this file into the pool folders

#change the job name to whatever you want it named

#SBATCH --job-name=dmpxOpihi

#SBATCH --time=96:00:00

#SBATCH -p serial

#SBATCH --nodes=1

#SBATCH --mail-user=cbird@tamucc.edu

#SBATCH --mail-type=begin # email me when the job starts

#SBATCH --mail-type=end # email me when the job finish

#Have to tell the hpc which software it need to load to run your job, the numbers after the programs are the version numbers

#for now just leave all of these programs up. probably dont need all of them

module load stacks

#create the directory where the demultiplexed sequences will go

mkdir demultiplex

process\_radtags -1 fastq/opihi\_R1P2\_CAGATC\_L001\_R1\_001.fastq.gz -2 fastq/opihi\_R1P2\_CAGATC\_L001\_R2\_001.fastq.gz -i gzfastq -b demultiplex\_Plate2Pool1\_opihi.txt -r -e ecoRI -o demultiplex -D

# -1 designates the read 1 fiel in a set of paired-end sequences. This needs to match the sequence file name.

# -2 designates the read 2 file in a set of paired-end sequences. This needs to match the sequence file name.

# -i sets the input file type. This should not be changed

# -b designates the barcode file that is used to separate the barcodes and name individual sequence files. This should be changed to match your barcode file name.

# -r tells the program to rescue barcodes that have no more than 2 mismatches. You do not need to adjust this parameter

# -e indicates the radtag that the program should search for. This is the enzyme associated with the read 1 sequences

# -o is the output path for the demultiplexed sequences. In this case it should be the demultiplexed seqs file

# -D tells the program to catch all of the discarded reads to a file

#cleanup on aisle 7

rename .1. \_Plate2Pool1.F. demultiplex/\*.1.fq.gz

rename .2. \_Plate2Pool1.R. demultiplex/\*.2.fq.gz

rename .rem\_Plate2Pool1 \_Plate2Pool1\_rem demultiplex/\*rem\*

rename \_ \_Plate2Pool1\_ demultiplex/\*radtags.log

mv demultiplex/\*rem\* removed\_seqs/

mv demultiplex/process\_\*\_radtags.log .

mv demultiplex/\*.discards removed\_seqs/

mv demultiplex/\*.fq.gz .

rm -rf demultiplex

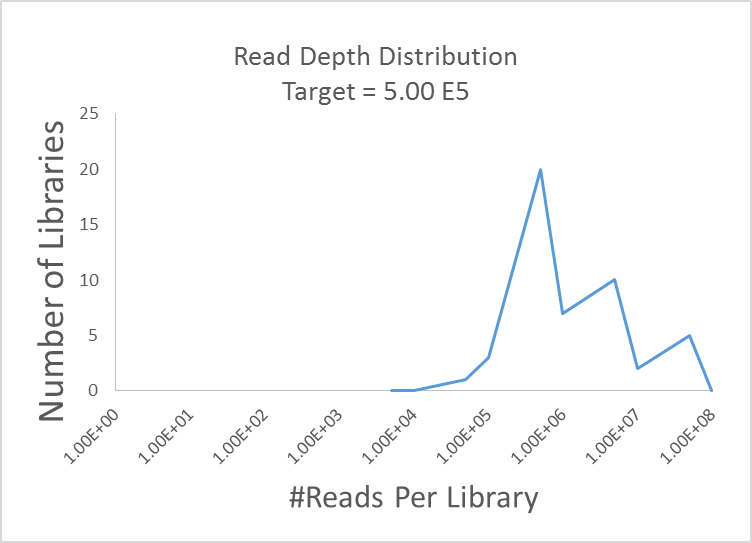
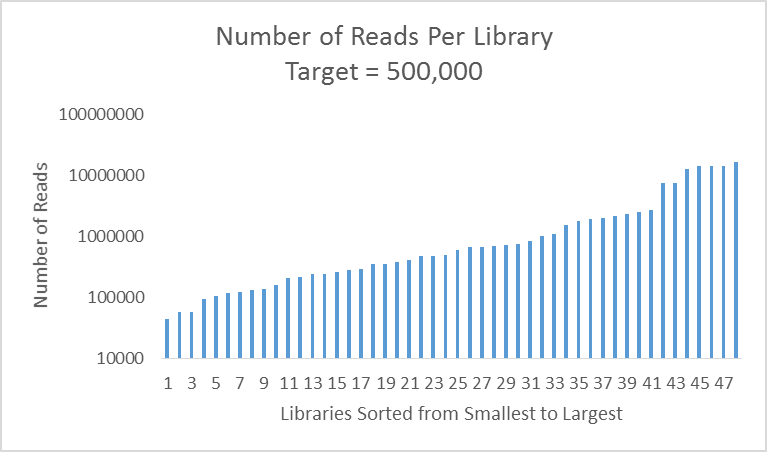
#count the reads (only do this for last demultiplex)

ls \*.gz | parallel "echo -n {}\_ && zgrep -c '^\+$' {}" | sort -g >> NumDmxReads.txt

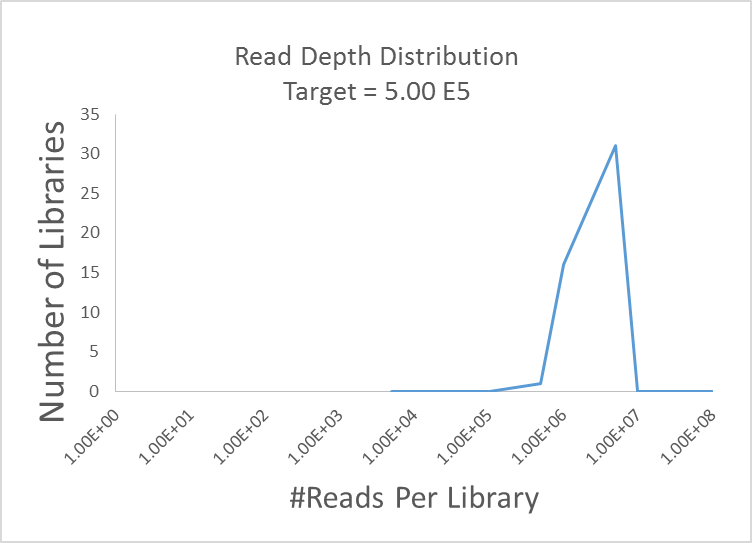
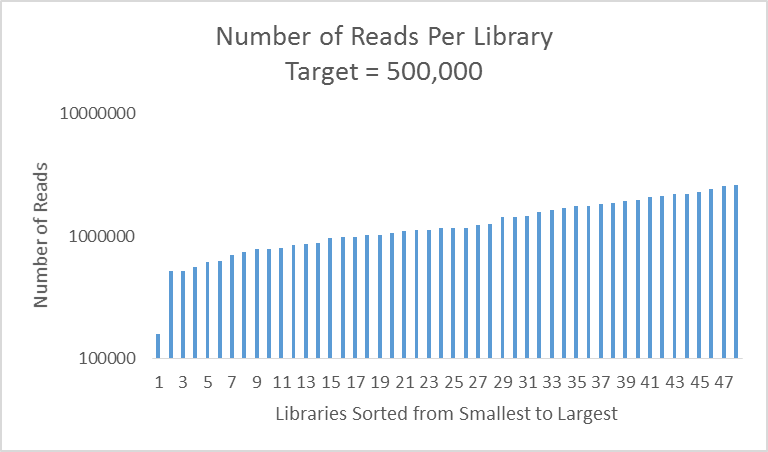
ls fastq/\*.gz | parallel "echo -n {}\_ && zgrep -c '^\+$' {}" | sort -g >> fastq/NumFqReads.txt

ls removed\_seqs/\*.gz\* | parallel "echo -n {}\_ && zgrep -c '^\+$' {}" | sort -g >> removed\_seqs/NumRemReads.txt

* + 1. Make sure the –e argument is set correctly
       1. Note, we had poor results by setting –renz\_2 but you may want to experiment
    2. \*\*\*\*note, process\_radtags is not parallel
       1. don’t use normal nodes or cbirdq on HPC, use “serial”
       2. if you want to demultiplex multiple files simultaneously, they will need to be done inside different directories so that the output files don’t overwrite each other
    3. make sure that files and directories are compatible with the script
    4. run the script
       1. sbatch ddRAD\_demultiplex.sh
  1. If necessary, rename the demultiplexed output files to reflect the name of the plate, pool, and sequencing run
     1. Sample 228 from the second sequence output for plate 2 in pool 1 should be
        1. 228\_Plate2Pool1Seq2.F.fq.gz and 228\_ Plate2Pool1Seq2.R.fq.gz
  2. Rename process\_radtags.log
     1. Process\_ Plate1Pool2Seq2\_radtags.log
  3. Move removed seqs (after renaming) to the removed\_seqs directory
  4. Check for the same individual being represented in multiple pools
     1. If yes, determine source of duplication
  5. Copy process\_radtags.log read count data to R or excel
     1. Note, the values reported in this file appear to be off if you have paired end reads, correct as follows:
        1. Total/2=TrueTotal
        2. Ambiguous|No RadTag=True Ambiguous|No RadTag
        3. (Total/2)-(Ambiguous|No RadTag)=True Retained Reads
     2. This is a bad sign:



1. In the libraries depicted above, most need to be requanted, recombined, and resequenced
2. A more even distribution on the left and a single peak on the right would be the best. Here the target was exceeded in 47 of 48 libraries, but not by too much. Very nice!



* + 1. Files that have too few reads can be moved to a directory called lowReadNum
       1. mkdir lowReadNum
       2. display files in order from largest to smallest
          1. ls –lS
       3. find files between x byte and y bytes
          1. find . –maxdepth 1 –size +55000c –a –size -260000c
       4. if that grabs the files you want to move, then do this:
          1. find . -maxdepth 1 -size +55000c -a -size -260000c -exec mv {} lowReadNum/ \;
    2. Files that have too many reads can be moved to a directory called highReadNum
       1. Use the commands described for moving files of a particular size range
       2. Large files can be divided into smaller files and treated as library replicates
          1. Calculate the desired number of lines per subfile

Choose a target number of reads per file

Use the mean read count for the acceptable libraries

Then divide the number of true retained reads by 2 million

Round up to the nearest whole integer, do not round down

7612633/2000000 = 3.8 -> 4

Divide True Retained Reads by 4 and round up

7612633/4 = 1903158.3 -> 1903159

The target number of lines per split file is #reads \* 4 because there are 4 lines per read in a fastq file

1903159\*4= 7612636

* + - * 1. Make a file called splitChunks.txt in the highReadNum directory

This should have, the order that files are returned by the ls command, the target number of lines per split file per library. This number will be the same for F and R files.

Example: there are two libraries with fwd and rev reads. The file will read as follows

7612636

7612636

7117024

7117024

* + - * 1. Script: splitFqFiles.sh

Copy to the highReadNum directory

#!/bin/bash

#SBATCH --job-name=splitFiles

#SBATCH --time=96:00:00

#SBATCH -p normal

#SBATCH --nodes=1

#SBATCH --mail-user=cbird@tamucc.edu

#SBATCH --mail-type=begin # email me when the job starts

#SBATCH --mail-type=end # email me when the job finish

module load parallel/20160722

echo "", echo `date` " unzip files"

ls \*.fq.gz | parallel "gunzip {}"

NumFiles=( `ls \*.fq | grep -c '^[0-9]' `)

echo "NumFiles=$NumFiles"

ZeroToNumFiles=( `seq 0 $(($NumFiles - 1)) `)

echo ${ZeroToNumFiles[\*]}

splitFiles() {

FileNames=( `ls \*.fq `) #load list of filenames

echo ${FileNames[\*]}

NumLines=( `cat "splitAmounts.txt" `) #load list of number of lines per chunk per file

echo ${NumLines[\*]}

mkdir $1 #make directory (1-Numfiles)

cd $1

split -d -l ${NumLines[$1]} ../${FileNames[$1]}

xfiles=( `ls x\* `)

echo ${xfiles[\*]}

Numxfiles=( `ls x\* | grep -c '^x' `)

echo $Numxfiles

for i in ${xfiles[\*]}; do mv $i ../$i-${FileNames[$1]};done

cd ..

rm -rf $1

}

export -f splitFiles

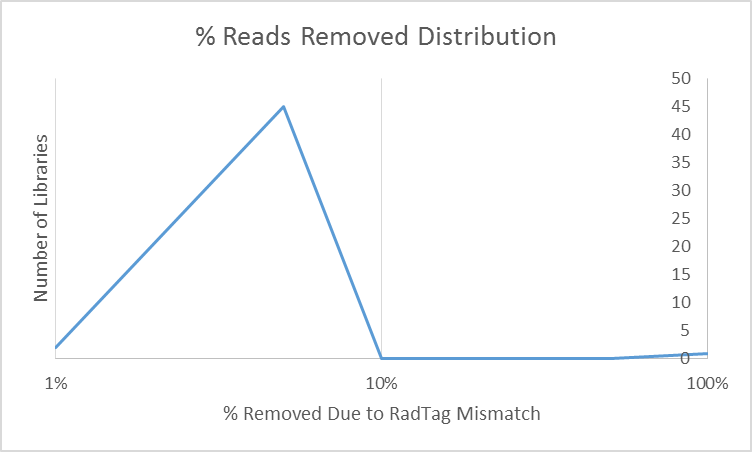
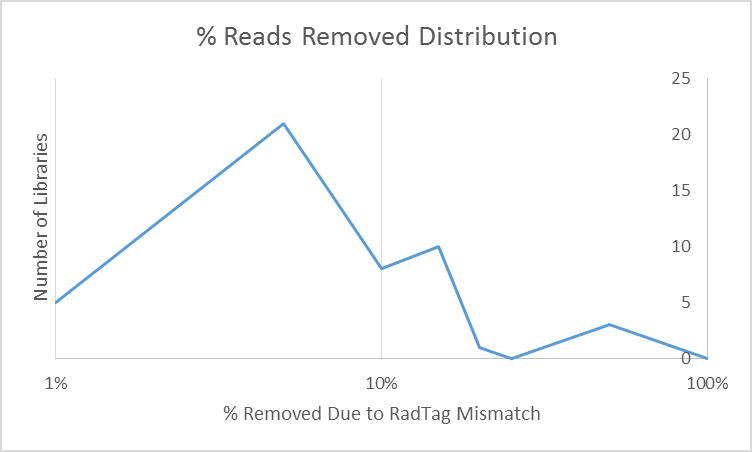
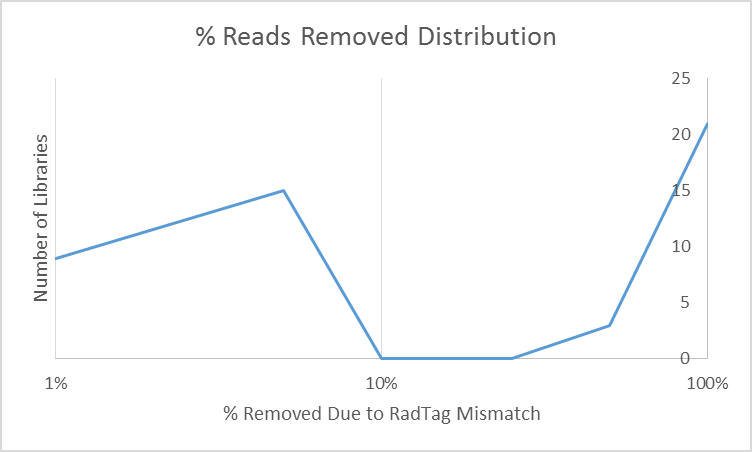
parallel splitFiles ::: ${ZeroToNumFiles[\*]}

ls \*.fq | parallel gzip {}

* + - * 1. Before running the script, modify it, the filenames, and directory structure so that they are compatible.
        2. Run the script

sbatch splitFqFiles.sh

* + - 1. Lastly select one or more of the split files per library to move back with the files to be trimmed
         1. mv x00\* ../
    1. Calculate % reads removed per library and overall due to No RadTag
       1. 100\*True No RadTag/TrueTotal
          1. From bad to good, left to right:



* + - * 1. We expect this value to be low <5%, if not then

Inspect removed seqs for low read quality

* + 1. Calculate % reads without a good barcode
       1. 100\*(True No BarCode)/(Overall True Total)
       2. We expect this value to be low <5%
    2. Assess the demultiplex.
       1. Are there an appropriate number of reads?
       2. Were a large proportion of reads lost?
          1. If yes, why?

Barcodes or radtags?

* + - * 1. Are some libraries affected more than others?

Is there something noteworthy about those libraries?

Low /high quant?

Low gel score?

???

1. **dDocent Basics**

dDocent is used to process the fastq files, both for developing a de novo reference genome and for genotyping.

*Dr. Jonathan Puritz made dDocent 2.24, but it has been modified by CEB. Please don’t bother Dr. Puritz with questions about script functioning unless CEB authorizes it. Please do cite dDocent as Dr. Purtz’s intellectual work* [*http://ddocent.com/citing/*](http://ddocent.com/citing/)

* 1. The dDocent documentation can definitely be helpful and the changes made by CEB were mainly to enable it to run on an HPC. However, there are some key incompatibilities.
     1. Direct user interface is disabled in dDocent for HPC, a config file must be used to give dDocent its settings
     2. Don’t use the standard dDocent config file
     3. dDocent for HPC is not meant to be run from beginning to end. It is meant to be run in separate steps
        1. trimming
        2. assembly
        3. mapping
           1. it’s possible to run genotyping with mapping
        4. genotyping
     4. dDocent for HPC prepares fastq files separately for assembly and mapping and saves them into directories called assembly and mapping
        1. when assembling, dDocent should be run from within the assembly folder
        2. when mapping and genotyping, dDocent should be run from within the mapping folder
  2. dDocent’s documentation will certainly enrich your understanding of how it works and what your goals are, but it is not assumed that you have read it in this manual.
     1. Website <http://ddocent.com/>
     2. Quick start <http://ddocent.com/quick/>
     3. Assembly <http://ddocent.com/assembly/>
     4. User Guide <http://ddocent.com/UserGuide/>
  3. dDocent is a bash script that mainly uses other software. As such the documentation for the different software used by dDocent will also enrich your understanding of what’s happening and how to make decisions.
  4. Script
     1. dDocent224tref\_hpc3 as of 12/27/2016
        1. note dDocent is in version 2.27, but the differences are minimal and there’s no reason to upgrade
  5. Running script
     1. While all of the scripts that run the dDocent script have been made for you, this info might be helpful in modifying or creating scripts
        1. Use module load to load ddocent/2.24
        2. Must use config.\* file on hpc
        3. bash dDocent224tref\_hpc3 configfilename
        4. remember, this should be run from a bash script
        5. see HPC Basics
  6. Functionality that is not yet supported on HPC
     1. Trimming
        1. It might work, I haven’t tried. [It works! Tested by Becca…details coming soon.]
        2. When I needed to trim some files, dDocent wasn’t loaded on the hpc and so I ripped some code from an earlier version of dDocent that used programs that were installed on the hpc. That’s why I didn’t work on the trimming portion of the script.
        3. The primary issue is that trimmomatic is a java program. I’m not sure how the hpc will deal with it

1. **Trim Files**

The raw sequence files need to be trimmed to remove low quality sequence and adapters. Files used for reference assembly are trimmed differently than those used for mapping. ezRAD files (forward reads only) are trimmed for assembly in a slightly different way than for ddRAD files.

***Prior to beginning this section, it is important to determine if your F & R reads are highly likely to overlap. You need to obtain the fragment size that was selected for the libraries (Sharon has this, but we will start a master file that also has this info). You also need to know what type of sequencing was performed (2x100, 2x125, 2x150, 2x250, 2x300). Lastly, compare the size selection to the total length of F and R reads. If they are close, your reads are highly likely to overlap. If your data is MiSeq 2 x 300, it is highly likely that your reads will overlap. If your data is 2 x 150, then is it is generally unlikely that the reads will overlap.***

* 1. Choose the script that matches your data type: trimFiles\_ezRAD.sh or trimFiles\_ddRAD.sh
     1. Modified from an older version of dDocent
     2. trimFiles\_ddRAD.sh

#!/bin/bash

#SBATCH --job-name=trim\_ddRAD

#SBATCH --time=96:00:00

#SBATCH -p normal

#SBATCH --nodes=1

#SBATCH --mail-user=USER@islander.tamucc.edu

#SBATCH --mail-type=begin # email me when the job starts

#SBATCH --mail-type=end # email me when the job finish

module load ddocent/2.24

module load parallel/20160722

module load cutadapt/20151012

module load fastqc/20151106

module load trim\_galore/0.4.0

#save file names into text file

echo -n "Getting filenames"

date

ls \*F.fq.gz | sed 's/\F.fq\.gz//' > filenames.txt

#mkdir assembly

#mkdir mapping

#Trim for assembly

seq 1 1 > loop1.txt

echo -n "Trimming F and R reads for assembly"

date

parallel "trim\_galore --paired --length 301 -q 10 -a GATCGGAAGAGCACACGTCTGAACTCCAGTCACNNNNNNATATCGTATGCCGTCTTCTGCTTG -a2 GATCGGAAGAGCGTCGTGTAGGGAAAGAGTGTAGATCTCGGTGGTCGCCG --stringency 8 -e .2 {2}F.fq.gz {2}R.fq.gz --output\_dir assembly" :::: loop1.txt filenames.txt

#echo -n "Trimming F reads for assembly, only required for ezRAD PE data"

#date

#parallel "trim\_galore --length 96 -q 0 -a GATCGGAAGAGCACACGTCTGAACTCCAGTCACNNNNNNATATCGTATGCCGTCTTCTGCTTG --stringency 8 -e 0.2 --clip\_R1 5 assembly/{2}F\_val\_1.fq.gz --output\_dir assembly" :::: loop1.txt filenames.txt

#Clean up files after trimming for assembly

mkdir assembly/trimreports

mv assembly/\*report.txt assembly/trimreports

rename F\_val\_1 r1 assembly/\*.fq.gz

rename R\_val\_2 r2 assembly/\*.fq.gz

#Trim for mapping

echo -n "Trimming F and R reads for mapping"

date

parallel "trim\_galore --paired --retain\_unpaired --length 50 -q 15 -a GATCGGAAGAGCACACGTCTGAACTCCAGTCACNNNNNNATATCGTATGCCGTCTTCTGCTTG -a2 GATCGGAAGAGCGTCGTGTAGGGAAAGAGTGTAGATCTCGGTGGTCGCCG --stringency 5 -e .2 {2}F.fq.gz {2}R.fq.gz --output\_dir mapping" :::: loop1.txt filenames.txt

mkdir mapping/trimreports

mv mapping/\*report.txt mapping/trimreports

mkdir mapping/unpairedreads

mv mapping/\*\_unpaired\* mapping/unpairedreads

rename F\_val\_1 R1 mapping/\*.fq.gz

rename R\_val\_2 R2 mapping/\*.fq.gz

* + 1. trimFiles\_ezRAD.sh

#!/bin/bash

#SBATCH --job-name=trim\_ezRAD

#SBATCH --time=96:00:00

#SBATCH -p normal

#SBATCH --nodes=1

#SBATCH --mail-user=USER@tamucc.edu

#SBATCH --mail-type=begin # email me when the job starts

#SBATCH --mail-type=end # email me when the job finish

module load ddocent/2.24

module load parallel/20160722

module load trim\_galore/0.4.0

#save file names into text file

echo -n "Getting filenames"

date

ls \*F.fq.gz | sed 's/\F.fq\.gz//' > filenames.txt

#mkdir assembly

#mkdir mapping

#Trim for assembly

seq 1 1 > loop1.txt

echo -n "Trimming F and R reads for assembly"

date

parallel "trim\_galore --paired --length 100 -q 10 -a GATCGGAAGAGCACACGTCTGAACTCCAGTCACNNNNNNATATCGTATGCCGTCTTCTGCTTG -a2 GATCGGAAGAGCGTCGTGTAGGGAAAGAGTGTAGATCTCGGTGGTCGCCG --stringency 8 -e .2 {2}F.fq.gz {2}R.fq.gz --output\_dir assembly" :::: loop1.txt filenames.txt

echo -n "Trimming F reads for assembly, only required for ezRAD PE data"

date

parallel "trim\_galore --length 95 -q 0 -a GATCGGAAGAGCACACGTCTGAACTCCAGTCACNNNNNNATATCGTATGCCGTCTTCTGCTTG --stringency 8 -e 0.2 --clip\_R1 5 assembly/{2}F\_val\_1.fq.gz --output\_dir assembly" :::: loop1.txt filenames.txt

#Clean up files after trimming for assembly

mkdir assembly/trimreports

mv assembly/\*report.txt assembly/trimreports

rm assembly/\*val\_1.fq.gz

rename F\_val\_1\_trimmed r1 assembly/\*trimmed.fq.gz

rename R\_val\_2 r2 assembly/\*\_2.fq.gz

#Trim for mapping

echo -n "Trimming F and R reads for mapping"

date

parallel "trim\_galore --paired --retain\_unpaired --length 50 -q 15 -a GATCGGAAGAGCACACGTCTGAACTCCAGTCACNNNNNNATATCGTATGCCGTCTTCTGCTTG -a2 GATCGGAAGAGCGTCGTGTAGGGAAAGAGTGTAGATCTCGGTGGTCGCCG --stringency 5 -e .2 {2}F.fq.gz {2}R.fq.gz --output\_dir mapping" :::: loop1.txt filenames.txt

mkdir mapping/trimreports

mv mapping/\*report.txt mapping/trimreports

mkdir mapping/unpairedreads

mv mapping/\*\_unpaired\* mapping/unpairedreads

rename F\_val\_1 R1 mapping/\* 1.fq.gz

rename R\_val\_2 R2 mapping/\* 2.fq.gz

* + 1. Copy trimFiles\_xxRAD.sh to the main project directory
       1. cp

***If your F & R reads are highly likely to overlap, you may want to customize the trim settings. I would decrease the min quality score to 4. I would decrease the minimum read length at least 10bp below the typical PE setting (the number of bp will be dependent on the amount of overlap)***

* + 1. Edit the script for compatibility with your files and directories
       1. mkdir assembly (if not already present)
       2. mkdir mapping (if not already present)
       3. rename files such that they follow the \*\_PlateXPoolY.F.fq.gz naming convention
       4. Under Trim for Assembly, edit --length ### so that it matches the length of your sequences.
          1. For ezRAD, there is a second trim line that clips 5bp off the forward read, set this --length 5bp shorter than the previous length.
          2. For ddRAD the F sequence is already 5bp shorter than the R, so make sure you input the length of the F reads here to avoid losing them in the trim.
       5. The quality score is probably the most important part of the trim procedure.
          1. For assembly, there is a fairly low quality threshold as low quality reads will generally have little effect on the creation of the reference
          2. For the mapping, the standard is higher.

13=95%

20=99%

* + - 1. make sure the adapters are correct
         1. those below are for single indexing
         2. dual indexing requires different adapter sequences
      2. Note that the script assumes that your files will have the following format
         1. \*.F.fq.gz \*.R.fq.gz
      3. Note that the file extension will be different for reads trimmed for assembly
         1. \*.r1.fq.gz \*.r2.fq.gz
      4. And those trimmed for mapping
         1. \*.R1.fq.gz \*.R2.fq.gz
  1. After trimming, run cntReads.sh to count the reads per file
     1. Add columns for assembly and mapping to the table of read counts
     2. Calculate the proportion of reads lost
        1. If there is excessive loss of reads, the quality settings may be set too high or the quality of your sequence may be too low
           1. ***The first steps of Assembly and Mapping instruct you how to rename files that should not be considered for assembly because they have too few reads***

***If your reads are highly likely to overlap, then you should pay close attention to the NumAssmReads.txt output here. You may want to compare the results from different trim settings. If you choose to do this, I recommend that you also compare FASTQC results and the number of reads assembled that is output to the slurm file in the assembly (next section).***

* 1. After trimming, run fastqc on the trimmed data and use multiqc to visualize
     1. You want to visualize results separately for libraries from different pools and sequencing lanes for the purpose of comparison

***If your reads are highly likely to overlap, then you should pay close attention to the FASTQC output here. There should be little to no detectable adapter in the assembly \*.fq.gz files. If there is still adapter, then decrease the –stringency setting (minimum match length for removing adapter) and retrim for assembly. You can also drop the –q value to 4, which will retain more sequences by setting the trim quality score to 60% certain of the base call.***

1. **Assembly of Reference Genome 1: Determine Cutoffs**

The assembly of paired end reads is really a two step process. In the first step, we determine which reads to use in reference creation. In the second step, the reference is created. At this point, I like to generate several references then pick 3 for mapping based upon the results.

***Prior to beginning this section, it is important to determine if your F & R reads are highly likely to overlap. You need to obtain the fragment size that was selected for the libraries (Sharon has this, but we will start a master file that also has this info). You also need to know what type of sequencing was performed (2x100, 2x125, 2x150, 2x250, 2x300). Lastly, compare the size selection to the total length of F and R reads. If they are close, your reads are highly likely to overlap. If your data is MiSeq 2 x 300, it is highly likely that your reads will overlap. If your data is 2 x 150, then is it is generally unlikely that the reads will overlap.***

***If you reads are highly likely to overlap, then will have to set the OL in your config files (instead of PE). If your reads are not likely to overlap, then the config files should be set to PE.***

* 1. Optional: if you don’t want to use all of your fq.gz files for assembly, you can use the FilterFilesForAssm.sh script to remove files from consideration without deleting them
     1. You would want to do this if there are files that have too few reads to be viable.
     2. Open the NumAssmReads.txt file (in excel)
        1. Use the sort functions and anything else to isolate files that should not be used.
     3. Make a \*.txt file using nano
        1. Copy the names of files that should NOT be used in the assembly to the \*.txt file, 1 file per row, both r1 and r2 files should be included
        2. Save the \*.txt file
     4. Open the FilterFilesForAssm.sh script
        1. In the script, modify the \*.txt file name so that it matches the one you created in the last step

#!/bin/bash

# This script will change file names for assembly, effectively inactivating some libraries based upon a file

# that has a list of the libraries to be inactivated. Both r1 and r2 file names that should not be used in

# the assembly should be in the file.

#SBATCH --job-name=FltrAssm

#SBATCH --time=2:00:00

#SBATCH -p dev

#SBATCH --nodes=1

module load parallel/20160722

#for this script to run you need to set the file that contains the list of samples that WILL NOT be used in

#the assembly

FilterFile="**14FishPerSample.txt**"

#reset name of files

ls \*.fq.gz | parallel "rename .read .r {}"

#rename files for assembly

cat $FilterFile | parallel "rename .r .read {}"

* 1. Script: assmFiles.sh
     1. Copy the script to the assembly directory
        1. Make sure the script is compatible with the locations of dDocent and the config.assm file
     2. All fastq files used for assembly have to have the following format
        1. \*.r1.fq.gz
        2. \*.r2.fq.gz
     3. Initial Settings in config.assm

Number of Processors (Auto, 1, 2, 3, ..., n threads) cbirdq=40 normal=20

20

Maximum Memory (1G,2G,..., 256G) G=gigabytes

230G

Trimming (yes,no)

no

FixStacks (yes,no) Demultiplexing with stacks introduces anomalies. This removes them.

no

Assembly? (yes,no)

yes

**Type\_of\_Assembly (PE, SE, OL, RPE) PE is normal, OL if F&R reads are likely to overlap**

**PE**

Clustering\_Similarity%

0.9

Mapping\_Reads?

no

Mapping\_Match\_Value

1

Mapping\_MisMatch\_Value

3

Mapping\_GapOpen\_Penalty

5

Calling\_SNPs?

no

Email

cbird@tamucc.edu

HPC, Get graphs for cutoffs, then stop? (yes,no)

yes

Manually set cutoffs? (yes,no) If yes is selected then dDocent will prompt you to enter cutoff values.

no

Cutoff1

4

Cutoff2

4

* 1. Run dDocent from within your assembly directory
     1. sbatch assmFiles.sh
  2. squeue to make sure the job is running
  3. Open slurm\*.out file to check results, output, and error messages
     1. If there are error messages, check the dDocent script and filenames for errors
        1. If there are errors, files created in this run of ddocent should be deleted
           1. Run the following command to list files in reverse chronological order

ls -ltr

* + - * 1. Run the following command to delete erroneous files

rm –rf erroneousfiles

replace erroneousfiles with a file that you want to delete

* + - * 1. Run the following command to list files newer than a particular date

find –newermt “Dec 27 08:01:31”

* + - * 1. Run the following command to delete files newer than a particular date (be careful!)

find –newermt “Dec 27 08:01:31” -delete

* + - 1. Once ddocent is run to this point, it will skip over most of the things it did the first time through if the files are created. This increases speed.
    1. ***If you selected OL, view the PEAR output and determine if Pear is assembling a high proportion of reads.***

Fri Mar 17 15:28:24 CDT 2017 OL assembly: PEAR

\_\_\_\_ \_\_\_\_\_ \_ \_\_\_\_

| \_ \| \_\_\_\_| / \ | \_ \

| |\_) | \_| / \_ \ | |\_) |

| \_\_/| |\_\_\_ / \_\_\_ \| \_ <

|\_| |\_\_\_\_\_/\_/ \\_\\_| \\_\

PEAR v0.9.6 [January 15, 2015]

Citation - PEAR: a fast and accurate Illumina Paired-End reAd mergeR

Zhang et al (2014) Bioinformatics 30(5): 614-620 | doi:10.1093/bioinformatics/btt593

Forward reads file.................: Barnacle\_Plate1Pool1runs1and2.r1.fq.gz

Reverse reads file.................: Barnacle\_Plate1Pool1runs1and2.r2.fq.gz

PHRED..............................: 33

Using empirical frequencies........: YES

Statistical method.................: OES

Maximum assembly length............: 999999

Minimum assembly length............: 100

p-value............................: 0.010000

Quality score threshold (trimming).: 0

Minimum read size after trimming...: 1

Maximal ratio of uncalled bases....: 1.000000

Minimum overlap....................: 10

Scoring method.....................: Scaled score

Threads............................: 40

Allocating memory..................: 200,000,000 bytes

Computing empirical frequencies....: DONE

A: 0.274348

C: 0.226595

G: 0.221725

T: 0.277331

194843 uncalled bases

Assemblying reads: 0%^MAssemblying reads: 10%^MAssemblying reads: 20%^MAssemblying reads: 30%^MAssemblying reads$

**Assembled reads ...................: 714,045 / 720,278 (99.135%)**

**Discarded reads ...................: 0 / 720,278 (0.000%)**

**Not assembled reads ...............: 6,233 / 720,278 (0.865%)**

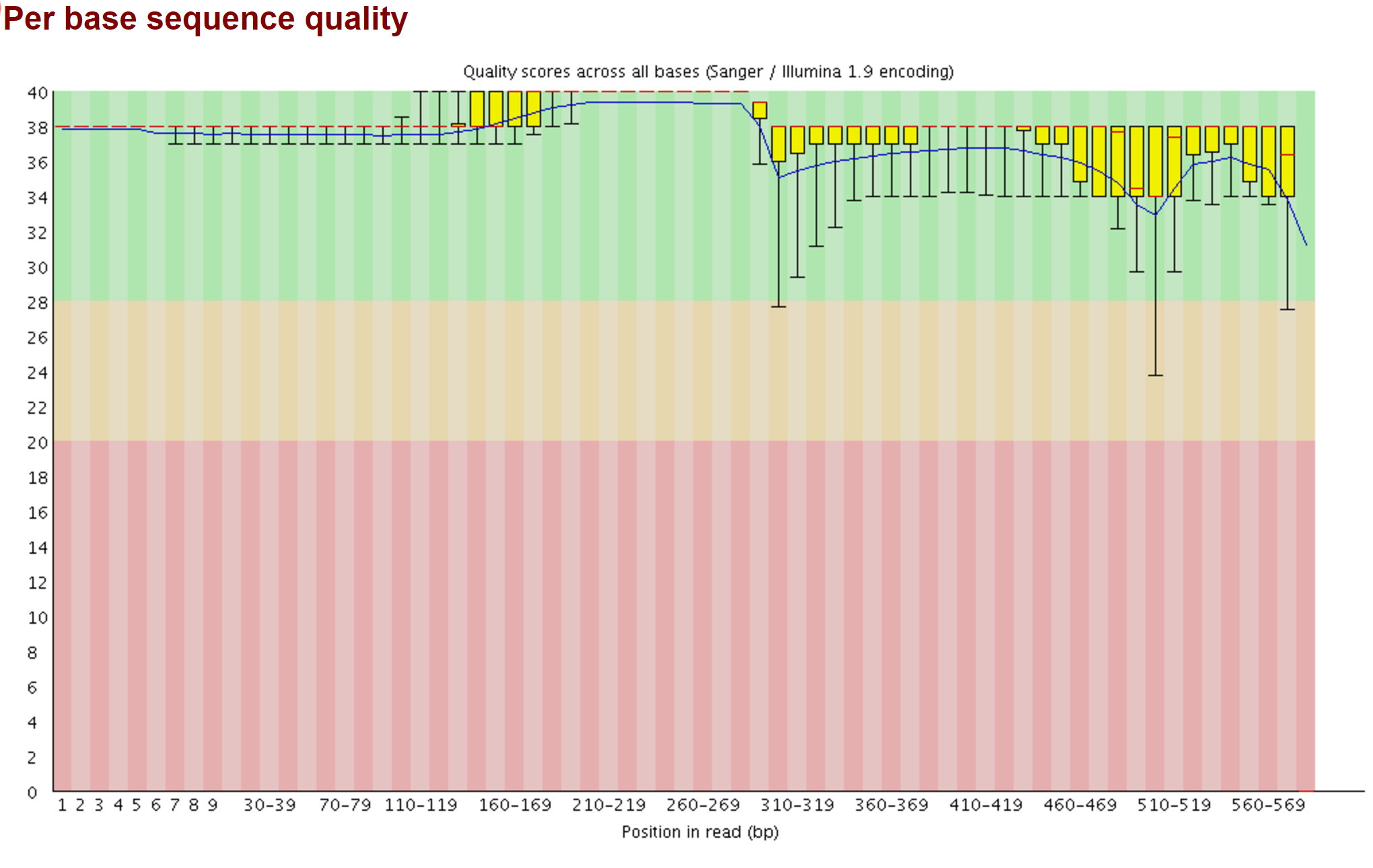
Assembled reads file...............: Barnacle\_Plate1Pool1runs1and2.assembled.fastq

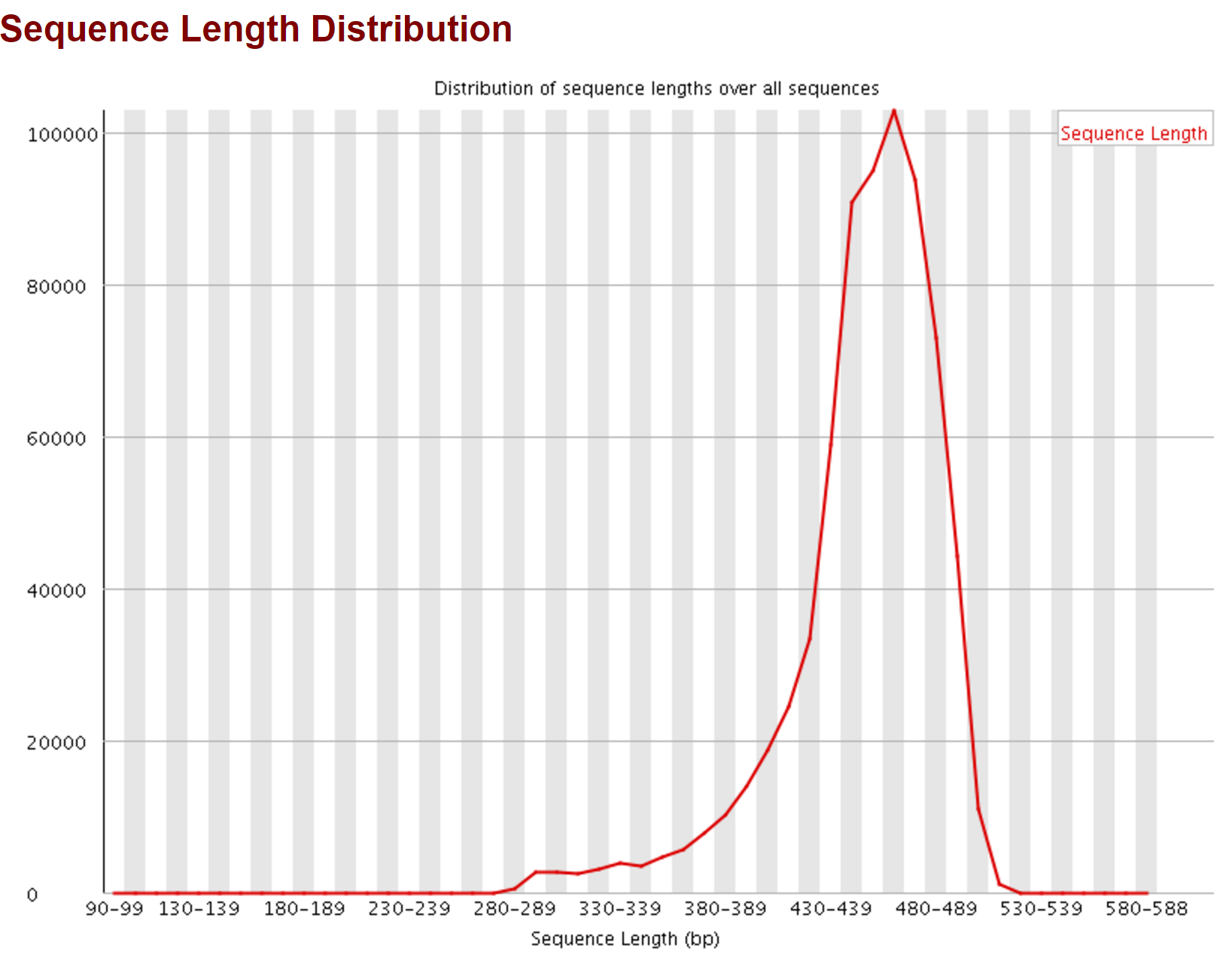
Discarded reads file...............: Barnacle\_Plate1Pool1runs1and2.discarded.fastq

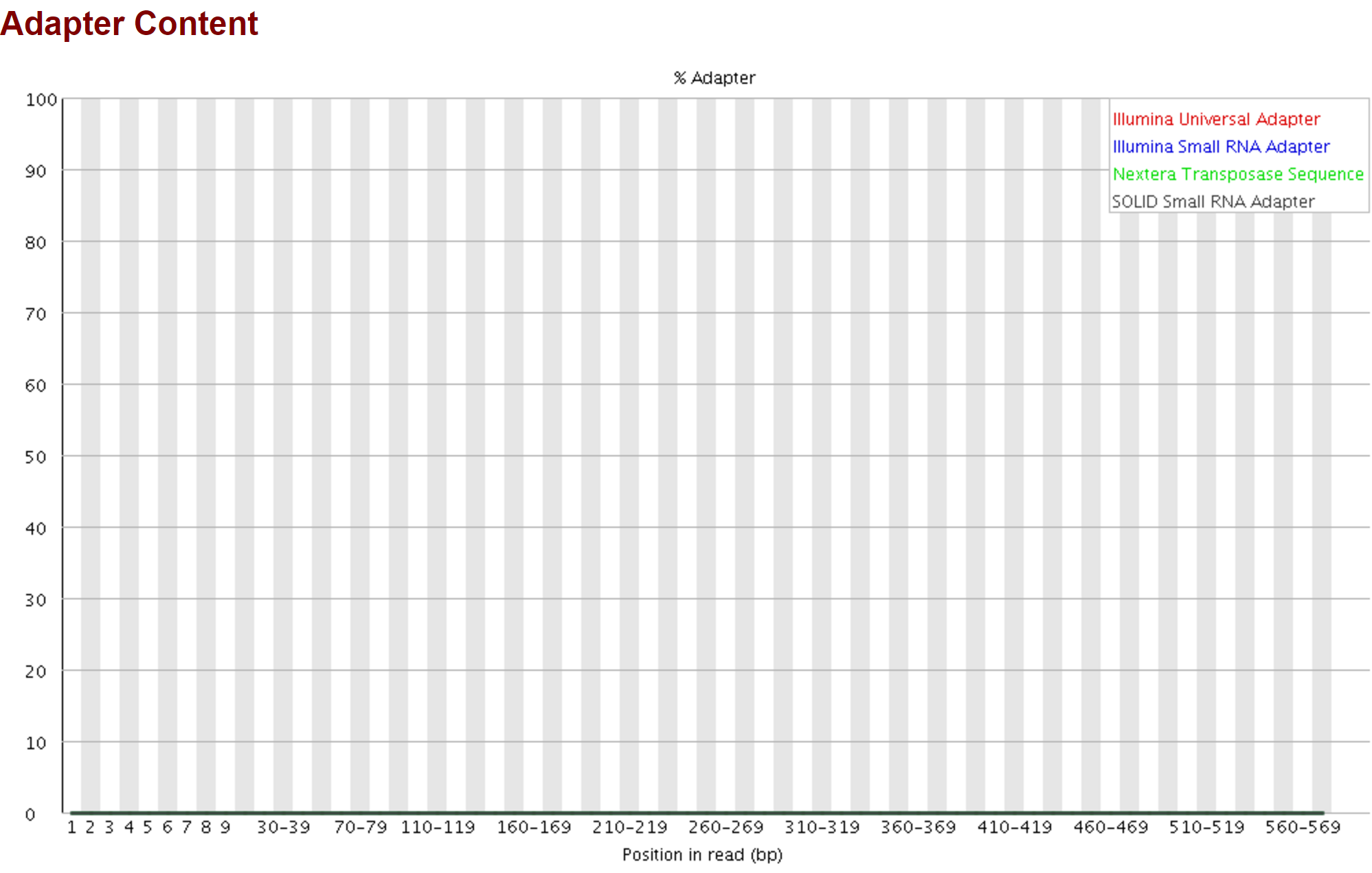
Unassembled forward reads file.....: Barnacle\_Plate1Pool1runs1and2.unassembled.forward.fastq

Unassembled reverse reads file.....: Barnacle\_Plate1Pool1runs1and2.unassembled.reverse.fastq

* + - 1. If a high proportion of reads are being assembled, then you’ll want to be sure that you paid close attention to the FASTQC output for the trimmed assembly \*.fq.gz files at the end of the previous section (Trimming)
      2. you can run FASTQC on your \*.assembled.fastq and \*unassembled\*fastq files if you want to be sure that the results are good.
         1. Indications of a problem would be an unusually unstable q scored in the overlap region (middle, remember that the middle should have the lowest quality because that where the and of read1 and 2 overlap) or discernable adapter content in the middle.
         2. Here is an example of acceptable FASTQC output from a \*assembled.fastq file where r1 and r2 overlapped substantially







* + 1. If all is well, view the first graph to select first cutoff: #uniq seqs with at least X coverage

Number of Unique Sequences with More than X Coverage (Counted within individuals) $

$

3.5e+06 +-+---+----------+----------+----------+----------+----------+-----------+----------+----------+------$

| \* + + + + + + + + $

| \* $

| \* $

3e+06 +-+ \* $

| \* $

| \* $

| \* $

2.5e+06 +-+ \* $

| \* $

| \* $

2e+06 +-+ \* $

| \* $

| \*\*\*\*\* $

| \*\* $

1.5e+06 +-+ \*\*\* $

| \* $

| \*\*\*\*\*\*\*\*\*\*\* $

| \*\*\*\*\*\*\*\*\* $

1e+06 +-+ \*\* $

| \*\*\*\*\*\*\*\*\*\*\* $

| \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* $

| + + + + + + + \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*$

500000 +-+---+----------+----------+----------+----------+----------+-----------+----------+----------+------$

2 4 6 8 10 12 14 16 18 $

Coverage $

* + 1. Data from first graph is stored in
       1. uniqseq.data
  1. Select 3 cutoffs, realizing that cutoff values will vary depending upon the number of libraries
     1. Here, I will select 4, 7, 10, 16
  2. Use the second graph to guestimate an appropriate range of second cutoffs

$

Number of Unique Sequences present in more than X Individuals $

$

300000 +-+------------------+---------------------+--------------------+---------------------+-----------------$

+ + + + + $

| \* $

| \* $

250000 +-+ \* $

| \* $

| \* $

| \* $

200000 +-+ \* $

| \* $

| \* $

150000 +-+ \*\* $

| \*\* $

| $

| \*\*\* $

100000 +-+ \* $

| \*\*\*\* $

| \* $

| \*\*\*\* $

50000 +-+ \*\*\*\*\*\*\*\*\* $

| \*\*\*\*\*\*\*\*\*\*\*\* $

| \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* $

+ + + + \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*$

0 +-+------------------+---------------------+--------------------+---------------------+-----------------$

0 5 10 15 20 $

Number of Individuals $

* + 1. The first cutoff used to make this graph was 3. Higher values for cutoff1 will cause this graph to decline more precipitously. The lower the value on the y axis of graph2, the fewer sequences will be used to make the reference. In this example, we targeted 10,000 loci, so we expect about 10,000 contigs in the reference. Realize that the values on the y axis are so low because several individuals did not have adequate sequencing depth.
    2. Ideally, you’d want to select cutoff2 based upon the asymptote, but number of unique sequences must also be considered and balanced
    3. Here, I will select 5, 10, 15 if cutoff1 was 3
    4. Because I selected 4,7,10,16 for cutoff 1 I will try the following combos next
       1. Cutoff1: 4, Cutoff 2: 5
       2. Cutoff1: 7, Cutoff 2: 4
       3. Cutoff1: 10, Cutoff 2: 3
       4. Cutoff1: 16, Cutoff 2: 3

1. **Assembly 2: Make Reference**

The cutoffs have been selected, now you can simultaneously run assmFiles.sh for different values of cutoff1. You cannot yet run two different sets of cutoffs where the cutoff1 values are the same. But, after the second graph is made in the output files, you can run multiple jobs with the same cutoff1

* 1. Change the config file as follows:

HPC, Get graphs for cutoffs, then stop? (yes,no)

no

Cutoff1

4

Cutoff2

5

* 1. Run 4 of the proposed cutoff combos, **one per cutoff 1**, each combo in a different job (these will vary depending upon your data set).
     1. 4,5
     2. 7,4
     3. 10,3
     4. 16,3
     5. If I were to use cutoff1=3, then I could run all of the combos now because the required files have been made previously
        1. uniqCperindiv.CUTOFF1
     6. If the assembly ends prematurely due to an error during Pear (see your slurm file)…

Check for overlap in paired end reads with PearFri Mar 17 09:34:46 CDT 2017

/work/GenomicSamples/llopezdemesa/Terrapin2017runs1and2/dDocent224tref\_hpc3.bash: line 1011: \* 5 / 4: syntax error: operand expected (error token is "\* 5 / 4")

* + - 1. Check the files that have been created
         1. If rbasm\*, rainbow\*, ref\* have a size of zero but rbdiv\* and rcluster\* do not then you probably need to change PE to OL in the config.assm file. See the dDocent documentation at ddocent.com
      2. If you have some other error, post it on the HPC forum under the dDocent HPC discussion.
  1. When the second cutoff graphs are created in the slurm\*.out files, the remaining combos can be chosen and run
     1. check the slurm\*.out file after 30 minutes
        1. here are the second graphs in order from 4,5 through 16,3

4

I’m going to try 9,12,18 for cutoff2

Number of Unique Sequences present in more than X Individuals $

$

250000 +-+------------------+---------------------+--------------------+---------------------+----------------$

+ + + + + $

| $

| \* $

| \* $

200000 +-+ \* $

| \* $

| \* $

| \* $

150000 +-+ \* $

| \* $

| \*\* $

| \*\* $

| $

100000 +-+ \*\*\* $

| \* $

| \*\*\*\*\* $

| \*\*\*\* $

50000 +-+ \*\*\*\* $

| \*\*\*\*\* $

| \*\*\*\*\*\*\*\*\*\*\*\* $

| \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* $

+ + + + \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*$

0 +-+------------------+---------------------+--------------------+---------------------+----------------$

0 5 10 15 20 $

Number of Individuals $

7

I’m going to try 7,11, 16 for cutoff2

Number of Unique Sequences present in more than X Individuals $

$

160000 +-+------------------+---------------------+--------------------+---------------------+----------------$

+ + + + + $

| \* $

140000 +-+ \* $

| \* $

| \* $

120000 +-+ \* $

| \* $

| \* $

100000 +-+ \* $

| \*\* $

80000 +-+ \*\* $

| $

| \*\*\* $

60000 +-+ \* $

| \*\*\*\* $

| \* $

40000 +-+ \*\*\*\* $

| \*\*\*\*\*\*\*\*\* $

| \*\*\*\*\*\*\*\* $

20000 +-+ \*\*\*\*\*\*\*\*\*\*\*\*\* $

| \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* $

+ + + + \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*$

0 +-+------------------+---------------------+--------------------+---------------------+----------------$

0 5 10 15 20 $

Number of Individuals $

10

I’m going to try 7,10, 15 for cutoff2

Number of Unique Sequences present in more than X Individuals $

$

120000 +-+------------------+---------------------+--------------------+---------------------+----------------$

+ \* + + + + $

| \* $

| \* $

100000 +-+ \* $

| \* $

| \* $

| \* $

80000 +-+ \* $

| \*\* $

| \*\* $

60000 +-+ $

| \*\*\* $

| \* $

| \*\*\*\* $

40000 +-+ \* $

| \*\*\*\* $

| \*\*\*\* $

| \*\*\*\*\*\*\*\*\* $

20000 +-+ \*\*\*\*\*\*\*\* $

| \*\*\*\*\*\*\*\*\*\*\*\*\* $

| \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* $

+ + + + \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*$

0 +-+------------------+---------------------+--------------------+---------------------+----------------$

0 5 10 15 20 $

Number of Individuals $

16

I’m going to try 6,10, 15 for cutoff2

Number of Unique Sequences present in more than X Individuals $

$

90000 +-+-------------------+--------------------+---------------------+--------------------+-----------------$

+ \* + + + + $

| \* $

80000 +-+ \* $

| \* $

70000 +-+ \* $

| \* $

| \* $

60000 +-+ \* $

| \*\* $

50000 +-+ \*\* $

| $

| \*\*\*\* $

40000 +-+ \* $

| \*\*\* $

30000 +-+ \* $

| \*\*\*\* $

| \*\*\*\*\* $

20000 +-+ \*\*\*\*\*\*\*\* $

| \*\*\*\*\* $

10000 +-+ \*\*\*\*\*\*\*\*\*\*\*\* $

| \*\*\*\*\*\*\*\*\* $

+ + + + \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*$

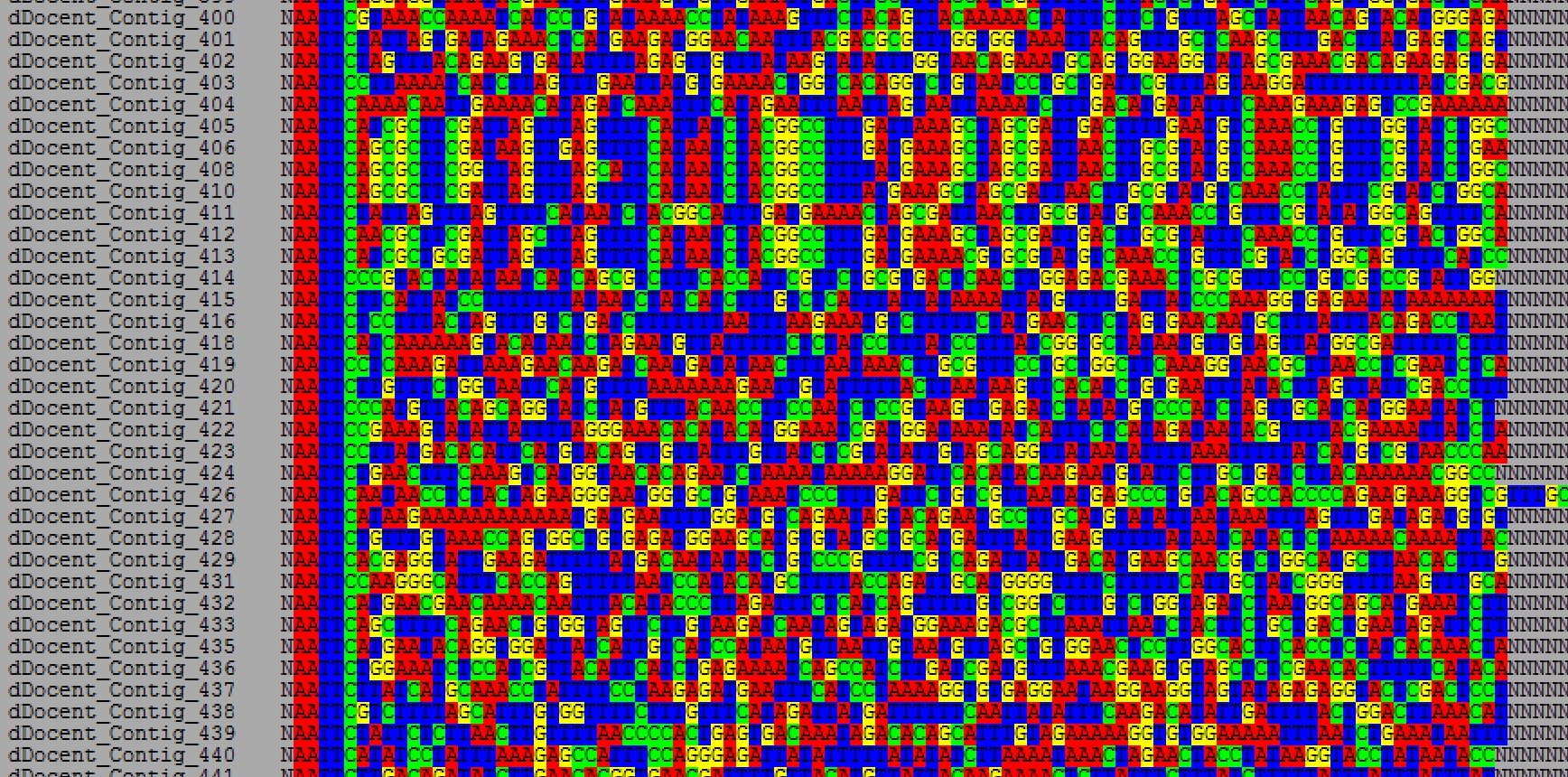
0 +-+-------------------+--------------------+---------------------+--------------------+-----------------$

0 5 10 15 20 $

Number of Individuals $

$

* + 1. If you want to go for it without viewing the second graph, the uniqCperindiv.CUTOFF1 file must be complete before running more simultaneous assemblies with the same CUTOFF1
       1. the file will be made in between the first and second graphs
    2. Make sure you check the completed slurm\*.out files for errors
  1. Run the additional cutoff combinations through ddocent
     1. As long as the cutoff1 has been run to this point, there is no limit on how many different combinations of cutoffs can be run simultaneously in different jobs from the same directory
     2. This process should complete very quickly now
     3. When all references are made, run srtContigs.sh if you assembled in OL mode. This will sort the reference\*fasta files and save them as sortedreference\*fasta
        1. If you used the PE setting, then don’t run the srtContigs.sh
  2. Finally, you can evaluate the reference.fasta or sortedreference.fasta sequences
     1. Open the reference.\*.fasta files with a graphical alignment viewer
        1. I use seaview but there are many. The important part is that each base should be highlighted in a different color so that it is easy to see similar sequences
        2. You want to note the prevalence of contigs with duplication in read1



* + - * 1. Count how many unique blocks of duplicates you see in first 1000 contigs and record the score (I would score the above screen as having 1 unique block of contigs)

I use the pg down feature and spend about a second per screen

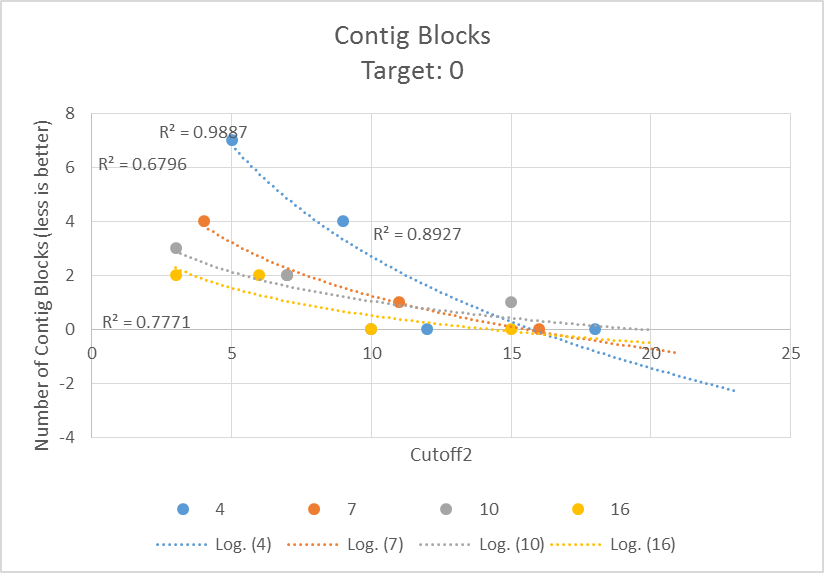
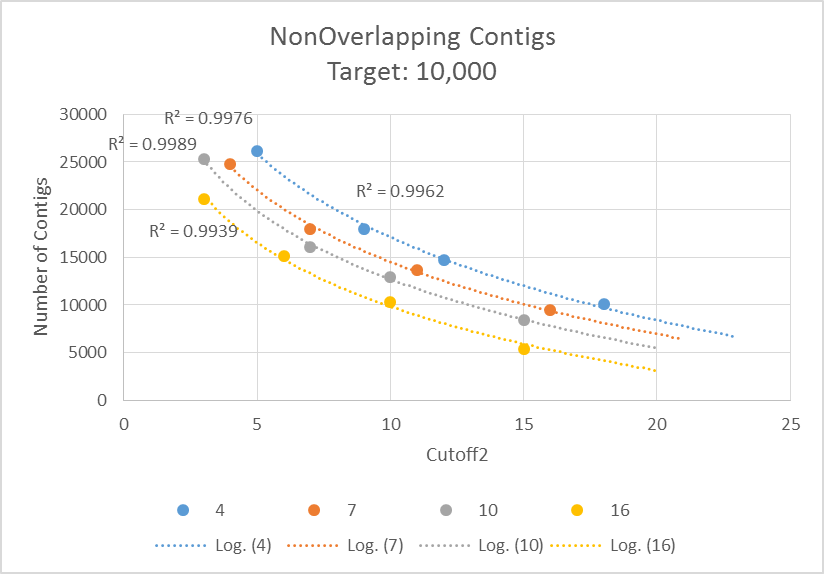
* + - * 1. They tend to be most concentrated at the beginning
        2. You don’t want them in your reference
        3. Higher cutoffs tend to eliminate these
      1. Record the number contigs with NNNNNNNNNN in the middle
         1. grep –c ‘NNNNNNNNNN’ reference.cutoff1.cutoff2.fasta
         2. To count for all references at once

grep –c ‘NNNNNNNNNN’ reference.\*.fasta

* + - 1. Record the total number of contigs
         1. grep –c ‘Contig’ reference.cutoff1.cutoff2.fasta
         2. To count for all references at once

grep –c ‘Contig’ reference.\*.fasta

* + - 1. Calculate the number of contigs without NNNNNNNNNN
         1. These fall at the end after the contigs with NNNNNNNNNN
         2. Total contigs – non-overlapping contigs = overlapping contigs



* + - 1. Select 3 references to move forward with based on graphs similar to those above: small, medium, and large
         1. The graph on the right indicates that a cutoff2 of >=10 is acceptable because there are a low number of duplicate blocks
         2. The graph on the left indicates that

cutoff1 = 4 results in a ref with the most contigs

4,12

cutoff1 = 16 results in a ref with the least contigs

16,15

cutoff1 = 10 falls in the middle

10,10

1. **Mapping**

Once you’ve settled on one or a few reference genomes, the next step is to map your reads to the reference. Note, if you’ve confirmed that mapping is working properly, mapping and genotyping can be run in 1 step. If this is your first time, then do mapping and genotyping separately.

* 1. Optional: if you don’t want to use all of your fq.gz files for mapping, you can use the FilterFilesForMap.sh script to remove files from consideration without deleting them
     1. You would want to do this if there are files that have too few reads to be viable.
     2. Open the cntMappedReads.txt file (in excel)
        1. Use the sort functions and anything else to isolate files that should not be used.
     3. Make a \*.txt file using nano
        1. Copy the names of files that should NOT be used in the assembly to the \*.txt file, 1 file per row, both r1 and r2 files should be included
        2. Save the \*.txt file
        3. These samples should be resequenced
     4. Open the FilterFilesForMap.sh script
        1. In the script, modify the \*.txt file name so that it matches the one you created in the last step

#!/bin/bash

# This script will change file names for assembly, effectively inactivating some libraries based upon a file

# that has a list of the libraries to be inactivated. Both r1 and r2 file names that should not be used in

# the assembly should be in the file.

#SBATCH --job-name=FltrAssm

#SBATCH --time=2:00:00

#SBATCH -p dev

#SBATCH --nodes=1

module load parallel/20160722

#for this script to run you need to set the file that contains the list of samples that WILL NOT be used in

#the assembly

FilterFile="**14FishPerSample.txt**"

#reset name of files

ls \*.fq.gz | parallel "rename .READ .R {}"

#rename files for assembly

cat $FilterFile | parallel "rename .R .READ {}"

* 1. Review the fastqc results for the files trimmed for mapping to ensure they meet your quality standards.
  2. ~~Review the NumMapReads.txt file. If there are libraries with too few reads(libraries with too many reads should have been dealt with prior to trimming), then~~
     1. ~~In the mapping directory~~
        1. ~~mkdir lowReadNum~~
        2. ~~move the files with too few reads to the lowReadNum directory~~
           1. ~~first list files in order of size~~

~~ls –lS~~

* + - * 1. ~~then try to list the files to be moved on the screen, make sure you only get the files that you want to move~~

~~find -maxdepth 1 -size +1600000c -a -size -10800000c~~

* + - * 1. ~~if that grabs the files you want, then move them~~

~~find -maxdepth 1 -size +1600000c -a -size -10800000c -exec mv {} lowReadNum/ \;~~

* + 1. ~~these libraries need to be resequenced~~
       1. ~~only libraries that are too small should be resequenced~~
  1. In order to run the mapping function, some files produced by the assembly process are required
     1. uniqCperindv.\*
     2. reference.\*.fasta\*
     3. use cp to copy the required files from the assembly directory into the mapping directory
  2. The config.map file should copied into the mapping folder
     1. cp
     2. adjust settings accordingly

Number of Processors (Auto, 1, 2, 3, ..., n threads) cbirdq=40 normal=20

20

Maximum Memory (1G,2G,..., 256G) G=gigabytes

230G

Trimming (yes,no)

no

FixStacks (yes,no) Demultiplexing with stacks introduces anomolies. This removes them.

no

Assembly? (yes,no)

no

Type\_of\_Assembly (PE, SE, OL, RPE)

PE

Clustering\_Similarity% (0-1)

0.9

Mapping\_Reads? (yes,no)

yes

Mapping\_Match\_Value

1

Mapping\_MisMatch\_Value

3

Mapping\_GapOpen\_Penalty

5

Calling\_SNPs?

no

Email

cbird@tamucc.edu

HPC, Get graphs for cutoffs, then stop? (yes,no)

no

Manually set cutoffs? (yes,no) If yes is selected then dDocent will prompt you to enter cutoff values.

no

Cutoff1

7

Cutoff2

10

* 1. Script: mapFiles.sh
     1. Edit the path to dDocent and the config.map file

#!/bin/bash

#rename files for ddocent

#rename R1\_001 F \*R1\_001.fastq.gz

#rename R2\_001 R \*R2\_001.fastq.gz

#SBATCH --job-name=mapOpihi

#SBATCH --time=96:00:00

#SBATCH -p normal

#SBATCH --nodes=1

#SBATCH --mail-user=cbird@tamucc.edu

#SBATCH --mail-type=begin # email me when the job starts

#SBATCH --mail-type=end # email me when the job finish

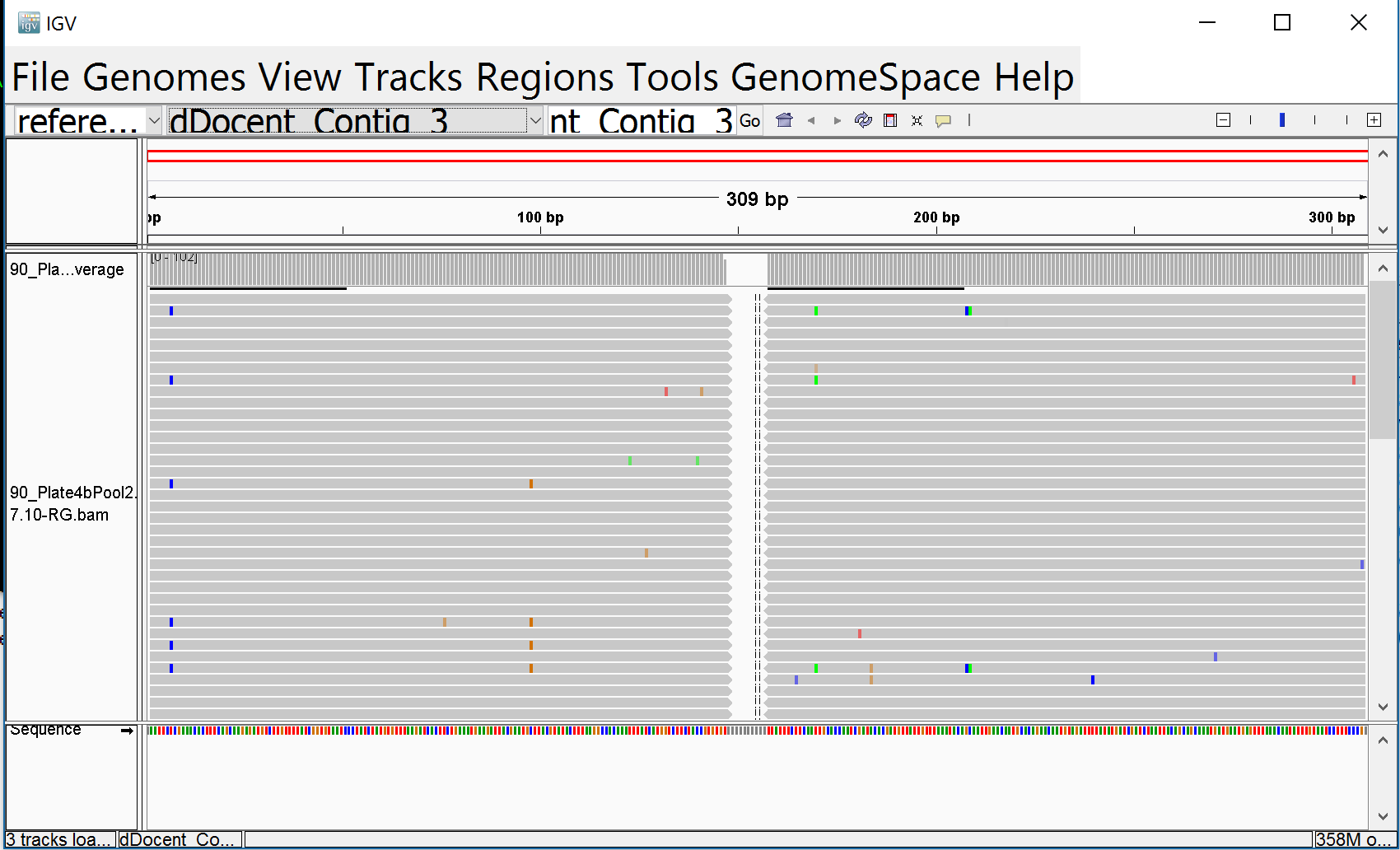
module load ddocent/2.24

module load bwa/0.7.15

module load vcflib/1.0

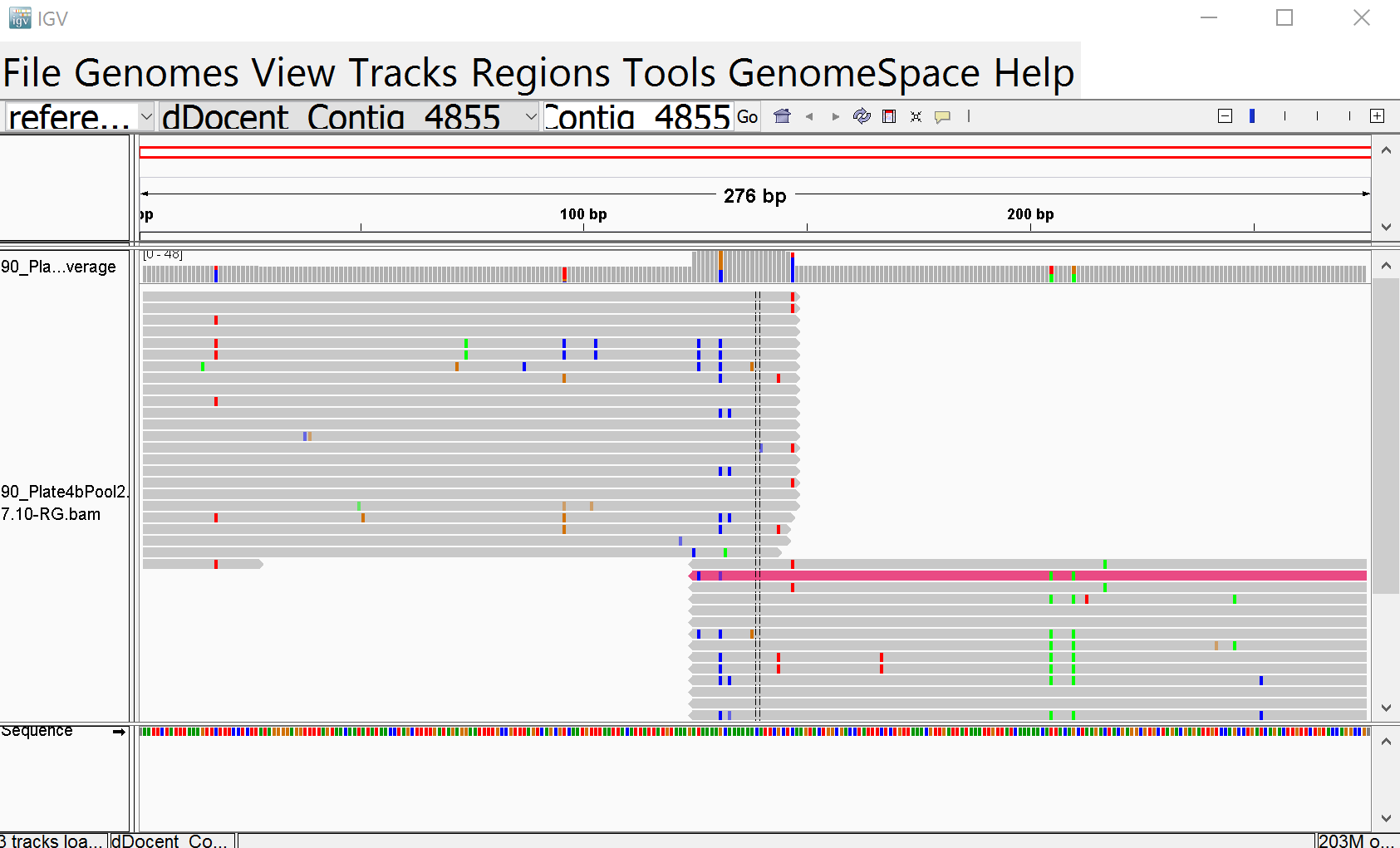
bash /work/GenomicSamples/cbird/opihiSK/dDocent224tref\_hpc3.bash /work/GenomicSamples/cbird/opihiSK/mapping/config.map

* + 1. Run it
       1. Sbatch mapFiles.sh
    2. Make sure it’s running
       1. Squeue
    3. Check the slurm\*.out file for errors
       1. nano
    4. repeat last two steps a few times
  1. Set two more mapping runs for the two other references
  2. When mapping is complete
     1. Review the slurm\*.out files for errors
     2. View contents of mapping directory in chronological order
        1. ls –ltr
        2. the run was probably successful if you have the following files with non-zero sizes
           1. samplename.\*-RG.bam
           2. bamlist.\*.list
           3. cat\*-RRG.bam\*
           4. mapped.\*.bed
     3. You should view some of your sample files using igv
        1. Either download the bam files and view on your personal computer or
        2. Open igv through the hpc
           1. module load igv
           2. igv –genome=reference.\*.fasta samplename.\*.RG.bam

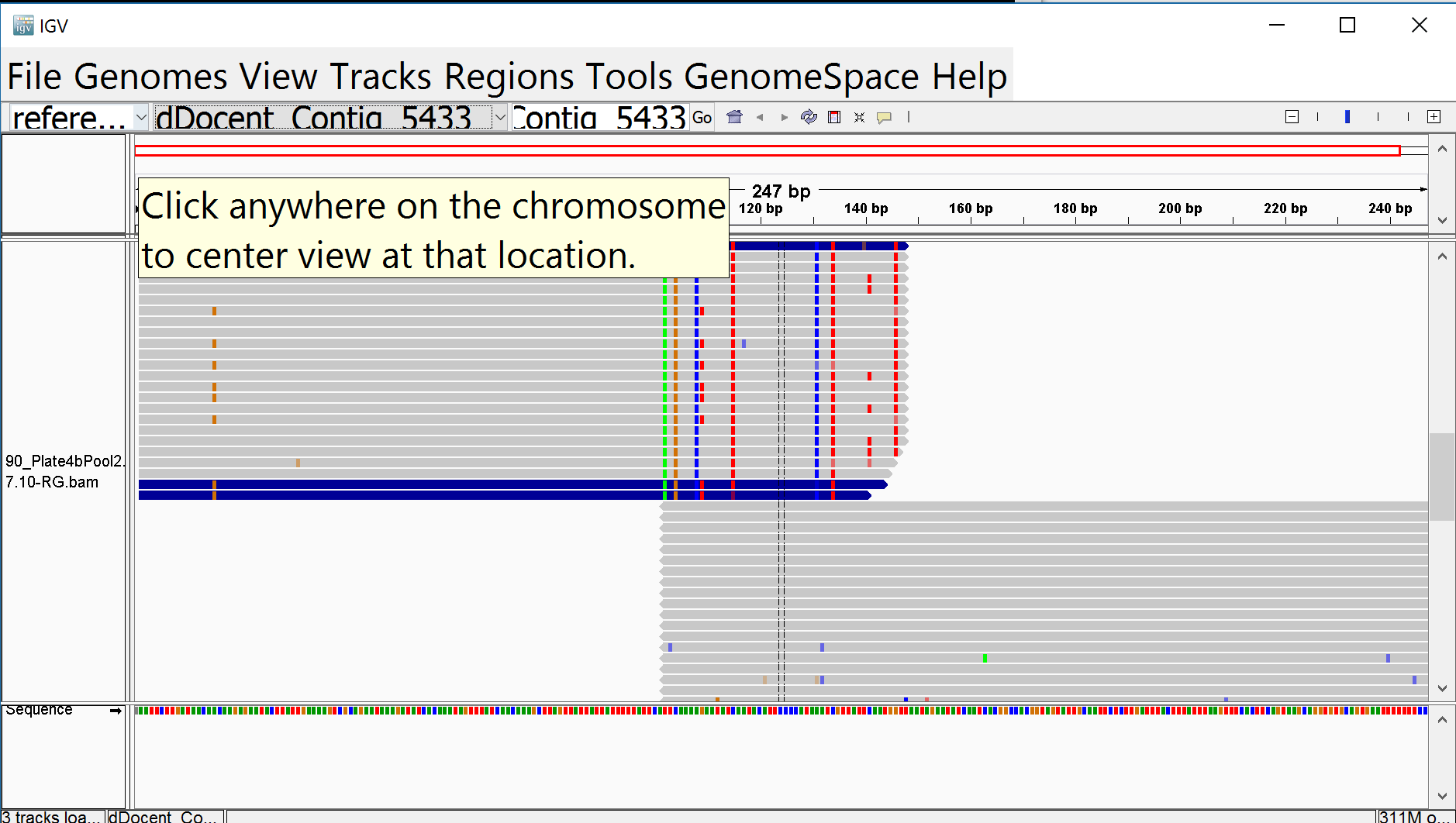


* + - * 1. We’ve noticed that a lot of contigs where reads 1 and 2 overlap have problems with strand bias. This means that they should not have been made to overlap. The program, pear, in the dDocent script controls this behavior. I’ve decided, generally, to just throw out the contigs with overlapping reads. The pear settings in the dDocent script could be adjusted to try to eliminate this problem.

Here’s a good one



Here’s a bad one, note that the fwd reads all have alternate alleles. These should not be overlapping.



* 1. Count the total number of reads mapped per individual (or pool) and the number of reads for which the F and R are not mapped to the same contig (i.e., mate mapped to different chromosome); graph to compare among references.
     1. Update the CountMappedReadsX.X.sh script, by finding and replacing all X.X with the cutoff1.cutoff2 of the reference for which you want the counts. Repeat this process for each set of reference cutoff results you want to count.
        1. To find and replace in nano, type ctrl + \ then follow prompts at bottom of screen.
     2. sbatch CountMappedReadsX.X.sh

#!/bin/bash

#SBATCH --job-name=cntX.Xmapped

#SBATCH --time=96:00:00

#SBATCH -p normal

#SBATCH --nodes=1

#SBATCH --mail-user=USER@tamucc.edu

#SBATCH --mail-type=begin # email me when the job starts

#SBATCH --mail-type=end # email me when the job finish

# UPDATE cutoff1 and cutoff2 numbers indicated by X.X

# to find and replace in nano, press ctrl + \ and follow prompts at bottom of screen

module load samtools/1.2

for i in $( ls | grep 'X.X-RG.bam$' ); do

echo item: $i

echo item: $i >> ReadCounts\_MappedX.X.txt

samtools flagstat $i >> ReadCounts\_MappedX.X.txt

done

# Count total mapped

echo Total Mapped >> ReadCounts\_MappedX.X\_Summary.txt

grep -E 'in total' ReadCounts\_MappedX.X.txt >> ReadCounts\_MappedX.X\_Summary.txt

# Count mate mapped to dif chromosome

echo Mate Mapped to Different Chromosome >> ReadCounts\_MappedX.X\_Summary.txt

grep -E 'chr$' ReadCounts\_MappedX.X.txt >> ReadCounts\_MappedX.X\_Summary.txt

1. **Genotyping**
   1. Turn off mapping and turn on genotyping in config.map

Number of Processors (Auto, 1, 2, 3, ..., n threads) cbirdq=40 normal=20

20

Maximum Memory (1G,2G,..., 256G) G=gigabytes

230G

Trimming (yes,no)

no

FixStacks (yes,no) Demultiplexing with stacks introduces anomolies. This removes them.

no

Assembly? (yes,no)

no

Type\_of\_Assembly (PE, SE, OL, RPE)

PE

Clustering\_Similarity% (0-1)

0.9

Mapping\_Reads? (yes,no)

no

Mapping\_Match\_Value

1

Mapping\_MisMatch\_Value

3

Mapping\_GapOpen\_Penalty

5

Calling\_SNPs?

yes

Email

cbird@tamucc.edu

HPC, Get graphs for cutoffs, then stop? (yes,no)

no

Manually set cutoffs? (yes,no) If yes is selected then dDocent will prompt you to enter cutoff values.

no

Cutoff1

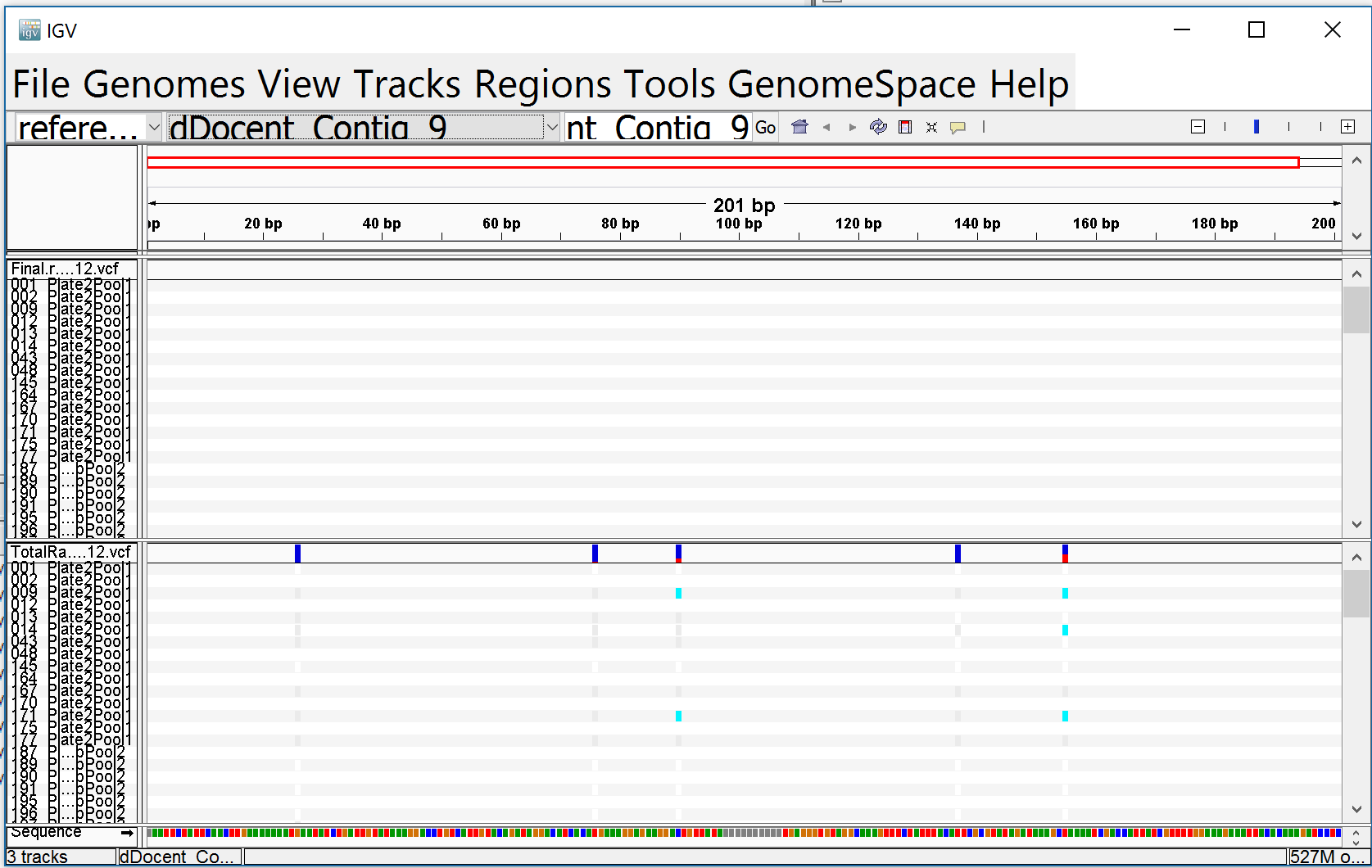
7

Cutoff2

10

* 1. run mapFiles.sh just like for mapping
     1. sbatch mapFiles.sh
  2. check that program is running
     1. squeue
  3. check slurm\*.out file for error messages
  4. log the sizes of the Raw and Final vcf files in the same table with the contig counts created during the assembly procedure
     1. Calculate the percent reduction in file size from Raw to Final for each cutoff combo
        1. Do these values seem excessive?
     2. Use igv, the same program you used to view the \*.bam files, to view the \*.vcf files to investigate any abnormalities
        1. I’ve found that you need to install igv on your local computer and view the files there. Igv is very slow using the hpc
        2. Realize that the \*.vcf files can be hundreds of mb to several gb

In the following screenshot of igv, the final vcf is in the top panel and the raw in the bottom. Only 5% of the contig file was retained. The igv view revealed that the libraries with 150 bp reads have no data. They were aligned to a reference made with 100bp reads. If you are trying to map longer reads to a reference made with smaller reads then you have to trim the longer reads down to the proper length.



* 1. Select good vcf files and gzip them
     1. gzip \*.vcf

1. **Filtering**

If you’ve made it this far, congratulations! You’ve probably come a long way on the linux learning curve. You now have a \*.vcf file loaded with snps and other genetic markers. Your mission, should you choose to accept it, is to isolate SNPs and remove erroneous data in a process termed filtering.

* 1. While dDocent does not perform filtering, Dr. Puritz provides a nice tutorial for filtering individual-based data using vcftools and vcflib.
     1. <http://ddocent.com/filtering/>
     2. Dr. Puritz’s scripts for filtering can be found here
        1. <https://github.com/jpuritz/dDocent/tree/master/scripts>
  2. I strongly encourage you to consult the manuals for the programs used
  3. Copy your vcf file(s) to the filtering folder
     1. Cp
  4. If you have individual-based data, then I suggest following Dr. Puritz’s tutorial.
  5. If you can’t decipher among individuals, then your filtering will need to be modified.
  6. Setting up filterHPC.sh
     1. Open the script and modify the following variables prior to running
        1. MAPdir
           1. Path to the mapping directory where the vcf was created
        2. CutoffCode
           1. The cutoffs you used during assembly “cutoff1.cutoff2”
        3. DataName
           1. A brief descriptor of your data. This name will be attached to many output files
     2. Other noteworthy locations to adjust variables based upon output from the script (slurm\*out file)
        1. In filterHPC.sh, the lines where settings can be adjusted are bookended by:

##!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

##!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

* + 1. You will probably have to edit your popmap file in your mapping directory. It is supposed to have two columns.
       1. First column is the individual name
       2. Second column is the pop sample name
    2. filter\_hwe\_by\_pop\_HPC.pl script
       1. if you are using the HPC version of dDocent, you need to use the HPC version of the hwe filtering script
          1. it is assumed that the popmap file is in your mapping directory
          2. you must set the –p, –d and –co options

–p the path and name of popmap file

–d the dataname, should be already set in your filteringHPC.sh bash script, at the top

–co the cutoffs, should be already set in your filteringHPC.sh bash script, at the top

* + 1. rad\_haplotyper115HPC.pl script
       1. if you are using the HPC version of dDocent, you need to use the HPC version of the rad haplotyper script
       2. you must set the –bp and –co options
          1. –bp enter the absolute path to the bam file dir
          2. –co enter the cutoff1 and cutoff2 values in this format: 3.4
       3. You must specify the correct number of threads or processors available to rad\_haplotyper
          1. Cbirdq 40
          2. Normal 20
          3. Dev 20
          4. Gpu 20
          5. If you are running on a workstation, then check using system monitor or equivalent

1. **Cleaning Up**
   1. Assess how much space your files are taking up
      1. du –s directorypath
      2. du directorypath

[cbird@hpcm cbird]$ du opihiSK/

15399240 opihiSK/highReadNum

130930 opihiSK/FASTQC

714 opihiSK/assembly/logfiles

604 opihiSK/assembly/trimreports

60004 opihiSK/assembly/FASTQC

18878834 opihiSK/assembly

127418 opihiSK/mapping/lowReadNum

196448 opihiSK/mapping/unpairedreads/unpaired

226604 opihiSK/mapping/unpairedreads

117076 opihiSK/mapping/FASTQC

14708388 opihiSK/mapping/highReadNum

250 opihiSK/mapping/logfiles

1208 opihiSK/mapping/trimreports

56463032 opihiSK/mapping

28553584 opihiSK/fastq

128664 opihiSK/lowReadNum

2077786 opihiSK/removed\_seqs

410 opihiSK/assembly4b2/trimreports

624 opihiSK/assembly4b2/logfiles

60078 opihiSK/assembly4b2/FASTQC

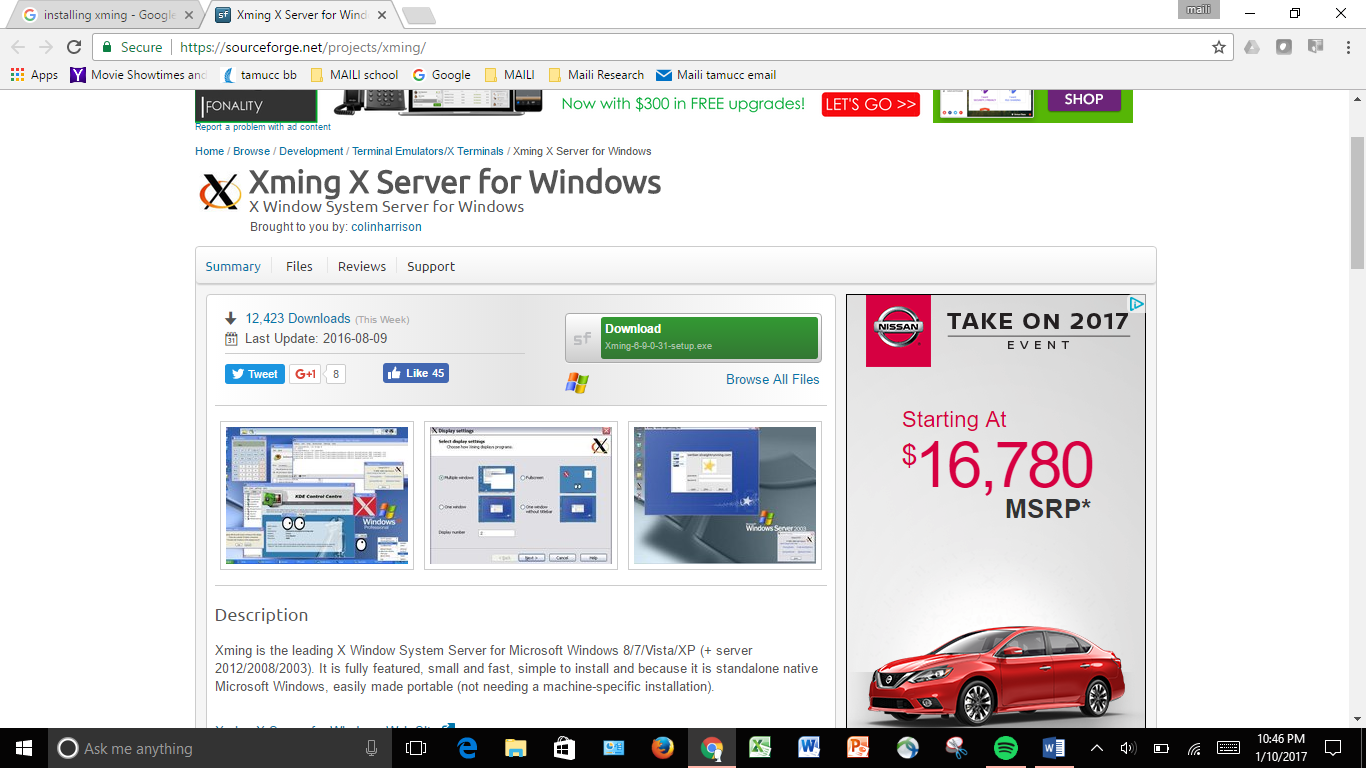
26805346 opihiSK/assembly4b2

165039576 opihiSK/

* 1. Make sure there are no unnecessary directories or files
     1. You should keep everything generated by following the instructions above
     2. But if you freestyle, make sure that you clean it up
  2. While you may want to wait until you are done working on a directory, when you are ready
     1. large files can be gzipped
        1. gzip
     2. the whole directory can be tarballed and gzipped
        1. tar
        2. if this is done, the tarball should be deposited in the lab museum (location tbd), where it can be accessed by posterity if need be

Appendix: step c-I (installing xming to run “mouse” programs on hpc.)

1. Installing Xming
   1. You can either go to this link <https://sourceforge.net/projects/xming/>



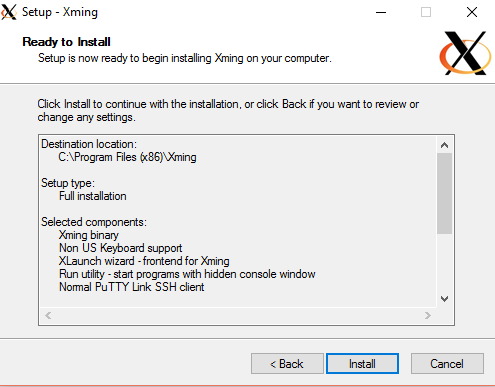
CLICK

OR you can google xming.

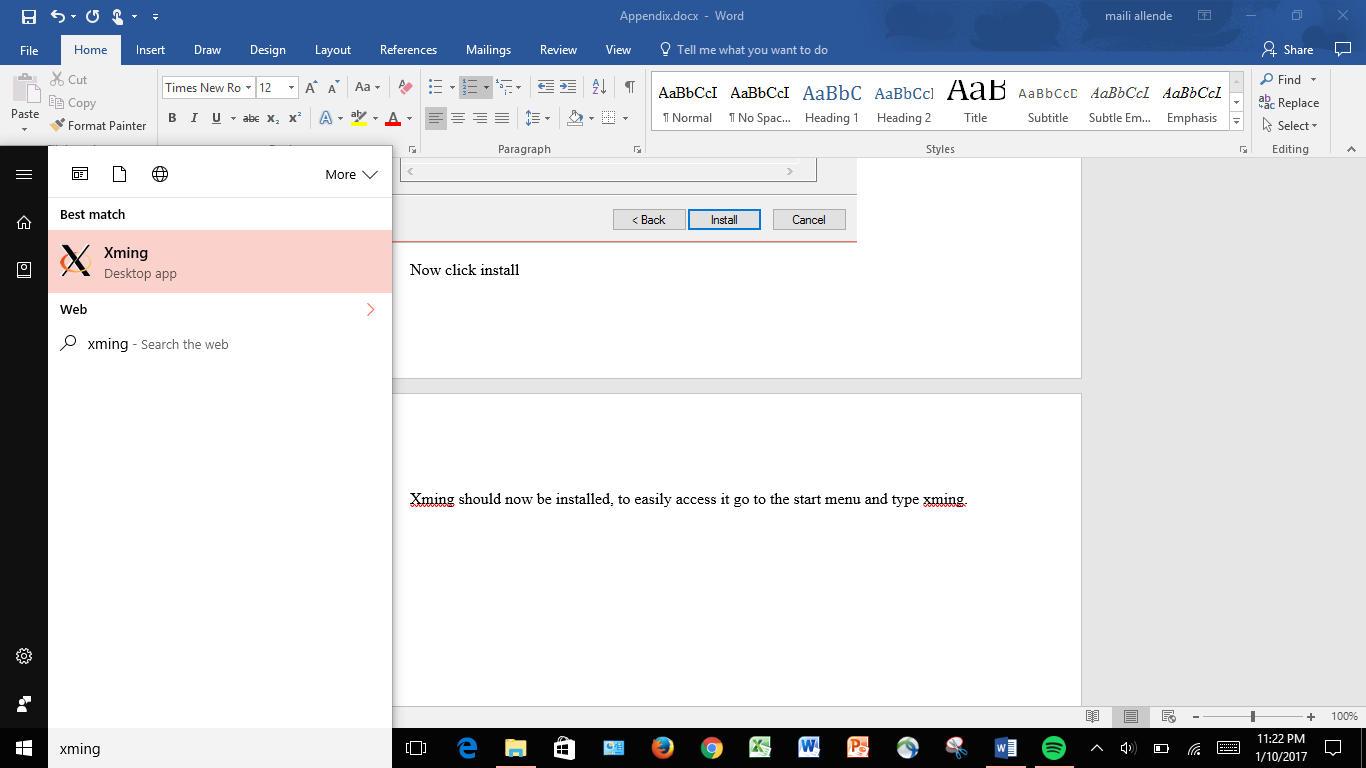
1. When you open xming it should look like this



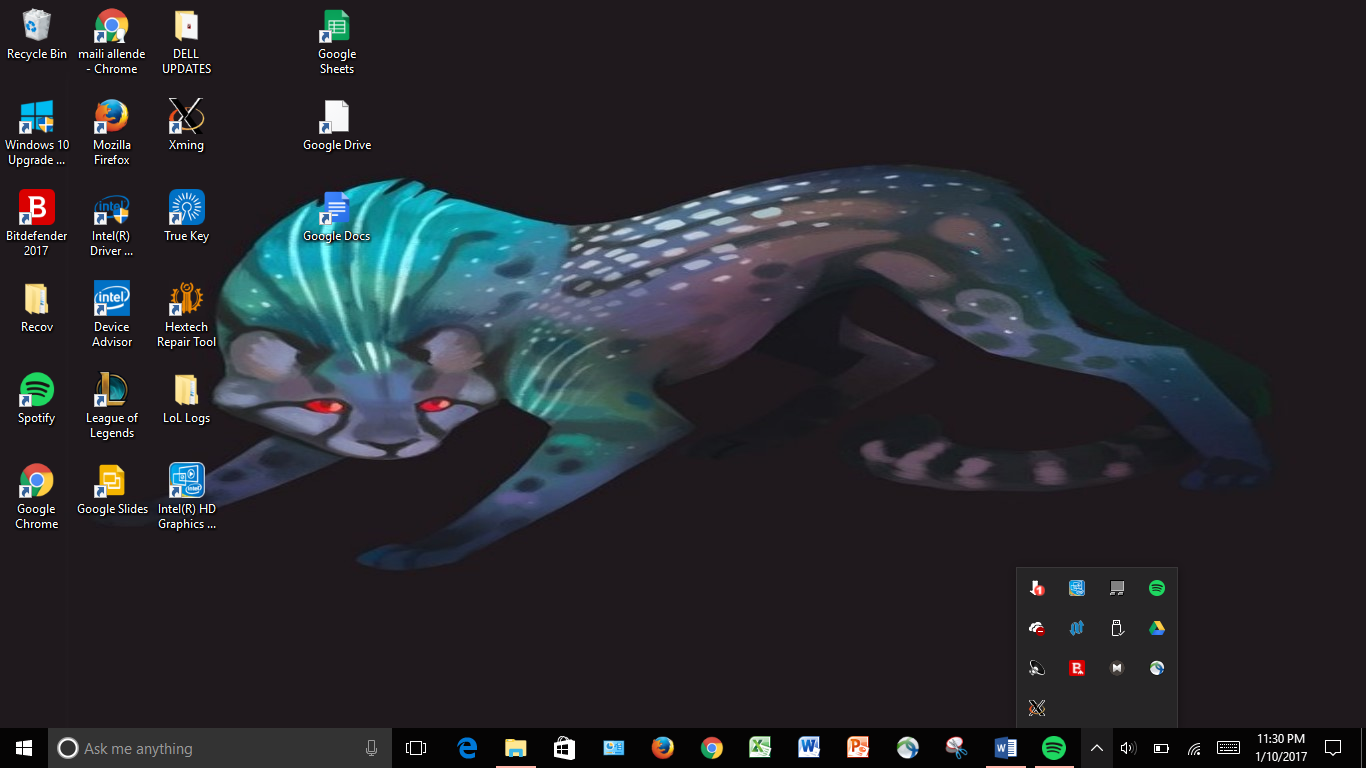
1. Keep clicking next till you reach this page



1. Now click install
2. Xming should now be installed, to easily access it go to the start menu and type xming.

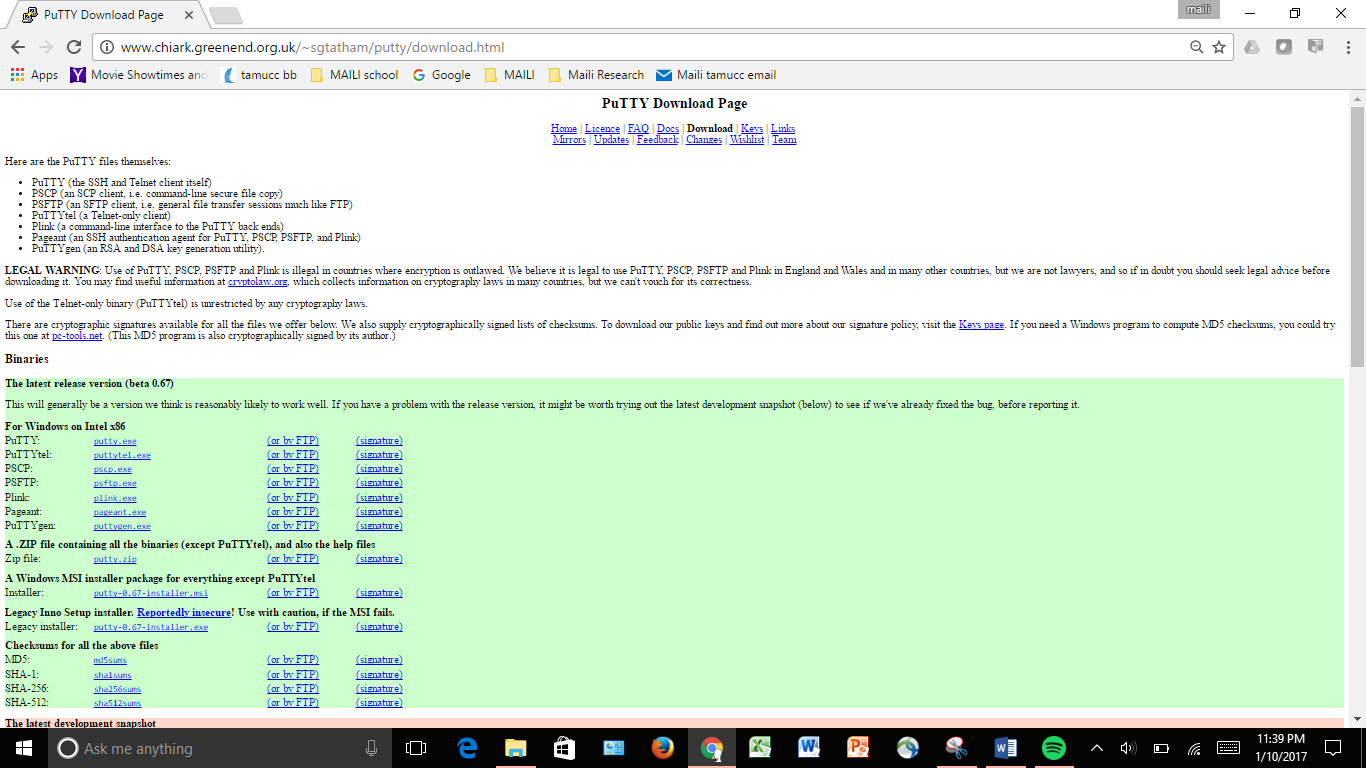


1. Then right click and pin to the start menu for easy access.
2. Once you click xming it will show that it is running in the task bar, like so.



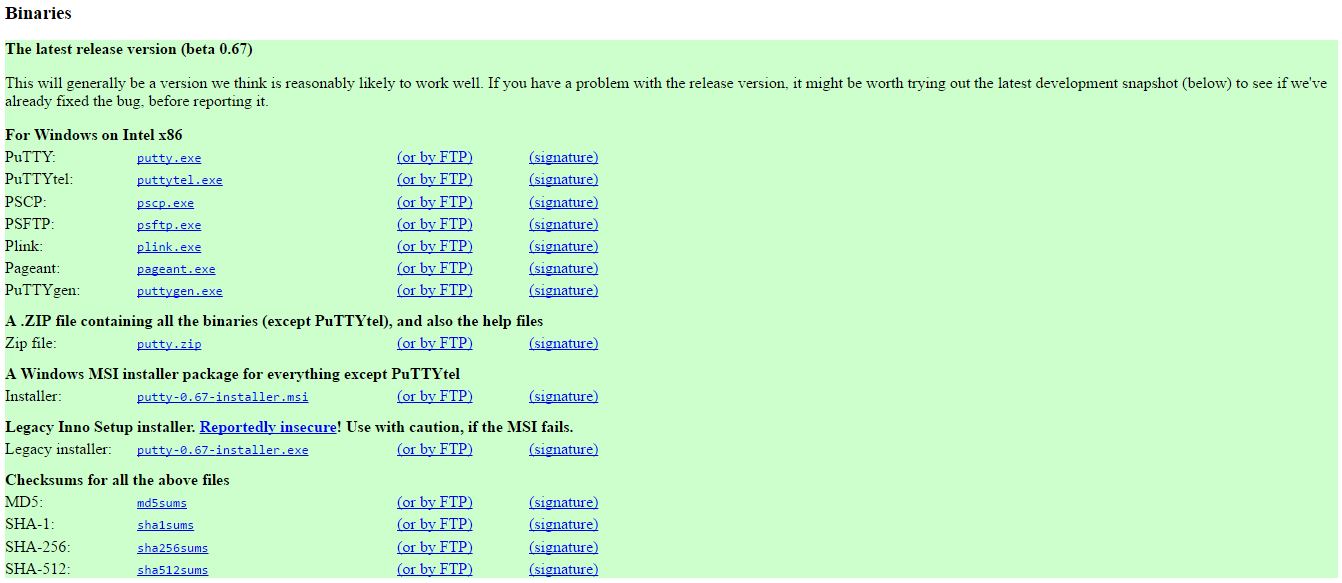
Appendix: c-ii (install putty to interface with hpc)

1. Google install putty, or use this link <http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html>.



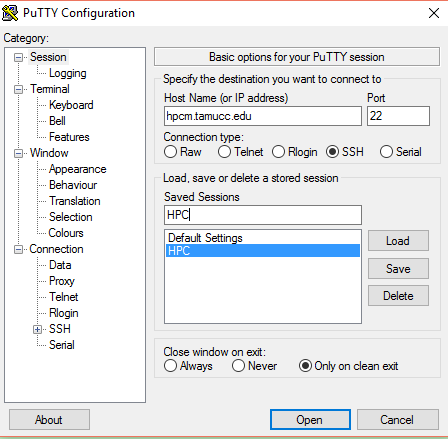
The page should look like this.

1. Click on the first link that says “PUTTY”.

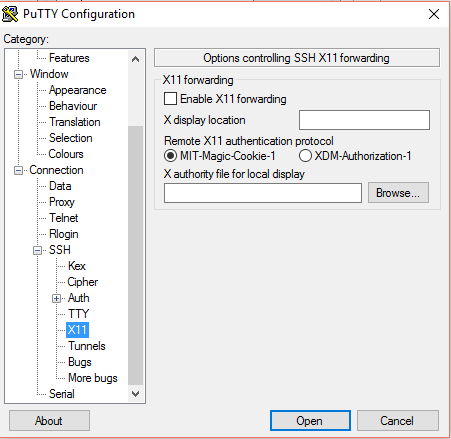


CLICK

1. Once the download is complete, this screen should pop up.
   1. In the “Host Name” type **hpcm.tamucc.edu**
   2. In the “Saved Sessions” type HPC, Then click the save button. After you click the save button “HPC” (highlighted in blue) should appear in the big box underneath.
   3. The rest should be the same as in the picture below (skip steps d-f), if not, follow the directions d-f.
   4. In the “Port” box it should say “22”
   5. In the “Connection type” box it should say “SSH”.
   6. In the “Close Window on Exit” box it should say “Only on Clean Exit”.
   7. DO NOT CLICK “OPEN” YET.



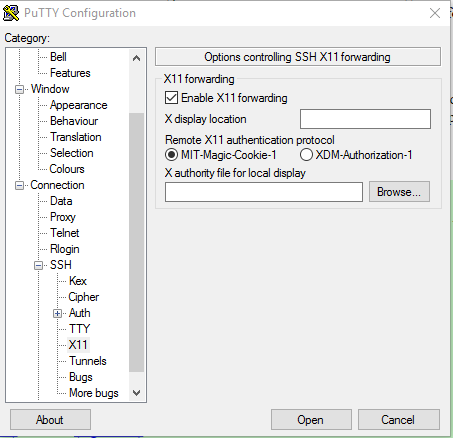
1. In the “Category” box 1st click SSH, then click X11.



2nd click

1st Click

1. Now click “Enable X11 forwarding”



1. Then go back to the “category” box and click “Session”
2. You should be back at the first screen you saw in putty, then click “HPC” and click “save” again.
3. To start putty, click “HPC” then click “load” then click “open”.
4. You can pin Putty to the start menu like we did to xming for easy access.

Appendix: Rename function

I got the trimFiles.sh to work on our computer, but I needed to modify the renaming commands from something like this:

rename F\_val\_1\_trimmed r1 assembly/\*trimmed.fq.gz

to

rename 's/F\_val\_1\_trimmed/r1/' assembly/\*trimmed.fq.gz

Consult the rename manual on your system by t

**Appendix. Comparison of Representation for Same Library Sequenced Twice**

We sequenced the Terrapin job with 22 turtles on a MiSeq 2x300bp two times. The relative representation of each sample was very similar from sequencing run to sequencing run.

