	FOOD SAFETY LAB / MILK QUALITY IMPROVEMENT PROGRAM *Standard Operating Procedure*

Title: #2 Quality control and assembly			
SOP #:	Revision: 00	Revision Date: If changed	Effective Date: Date Upload
Author: Sophia Harrand, Jingqiu Liao, Jasna Kovac	Approved by: Martin Wiedmann		

#2 Quality control and assembly

FILE NAME: Quality_control_and_assembly.doc

TABLE OF CONTENTS

1. INTRODUCTION 3
Purpose
Scope
Definitions
Safety

2. MATERIALS 4

3. PROCEDURE 5

4. TROUBLESHOOTING 6

5. REFERENCES 7

6. APPENDIX

SECTION 1 INTRODUCTION

This is protocol #2 out of 4 protocols for whole genome sequencing:

#1 Library preparation for whole genome sequencing
#2 Quality control and assembly
#3 Short read submission (SRA)
#4 Assembly submission to NCBI WGS

1.1 Purpose

The purpose of this document is to set forth **standard** guidelines for analysis (converting SRR experiments to fastq.gz files), quality control of short reads, short read trimming, short read assembly and quality control, and storage of whole genome sequencing data. This will ultimately lead to easy accession and allow for reproducible downstream analyses.

1.2 Scope

This SOP applies to the Food Safety Lab and the Milk Quality Improvement Program.

1.3 Definitions

SRA: The Sequence Read Archive (SRA) database is part of the NCBI database and stores sequence and quality data from NextGen sequencing platforms in aligned or unaligned formats.
Fastq.gz: These files contain the compressed (gzip) short sequence reads with quality scores, sequencing platform and index information.
Contigs: Contiguous sequence of DNA.
SNP: Single-nucleotide polymorphism, variable position in the sequence alignment.
Fasta: Text based DNA sequence format, which contains a greater than sign (>) followed by the sequence descriptor in the first line, and DNA sequence in the following line.
Shell: Command-line interface for access to the operating system's services. Also known as "terminal" or "bash".

1.4 Safety

This protocol is computer-based only.

SECTION 2 MATERIALS

- **PC or Mac personal computer**
- **Access to Zeus (Food Safety Lab Linux PC, also known as "Zeus")** Please talk to David Kent to help you set up your personal computer to remotely access the Linux PC.
- **PC:** Download **PuTTY** http://www.putty.org/, **WinSCP** http://winscp.net/eng/index.php and **Notepad ++** https://notepad-plus-plus.org/
- **Mac:** Mac users can access Linux computer remotely through ssh.
- If you wish to be added the FSL **BioHPC** account to use BioHPC storage an computational services, please discuss with Martin Wiedmann and David Kent prior to requesting the account through the following website: https://cbsu.tc.cornell.edu/lab/lab.aspx. Refer to the BioHPC SOP for further details.
- Programs needed for computational analyses described in this SOP (available on Linux PC):
  - SRA Toolkit
  - FastQC
  - Trimmomatic
  - SPAdes
  - BBMap
  - Samtools
  - QUAST
  - RaxML
  - kSNP

SECTION 3 PROCEDURES

3.1. Linux shell commands

Note: Pay attention to spacing and writing, including capitalize letters etc.

cd	To navigate into a folder e.g., *cd /data/intermediate/runXY*
ls	List files; shows you the content of a folder
ls –l	List files with detailed information; shows the files in a folder including their sizes, owners (individual owner and group owner), and reading, writing and executing permissions
ls | grep	Shows only what you request in search criteria (don't scroll in linux shell) e.g., *ls | grep R9* (to search for files that include R9 e.g. FSL R9-5237)
chown –R	Change the right for a directory e.g., *chown –R :fsl* (this will change the group permission to fsl group)
cd ..	To move one directory (folder) up (e.g., from */data/intermediate* to */data*)
rm	To remove (delete) a file. If you wish to delete several files with the same ending use * e.g., **.bam*
rm –r	To remove a folder
zeusctl list	Shows programs that are enabled in your shell environment. Prior to the first time you are using a program remotely from your shell, you should enable it.
zeusctl enable	Adds full path to the program into your shell environment e.g., *zeusctl enable programXY* (once enabled on your personal computer you will not have to do this again). This will allow you to run a program without providing a full path to the program. Programs you will need to enable to perform the analyses described in this SOP: - *bbmap* - *kraken* - *ksnp* - *ksnp3* - *spade* - *samtools* - *spades* - *spades* - *trimmomatic* Important note: Check the exact names and versions of the programs that need to be enabled on Zeus PC in /programs. Some programs are being regularly updated (e.g., SPAdes), therefore the names of the programs may change.
vim	To read a script in the shell text editor (e.g., *vim scriptXY.sh*) When the file (script) is opened in the text editor, use the following commands to edit, save and exit: *I* – Insert

	Esc (on the keyboard) – exit the Insert (or other) mode *:q!* – quit (without saving any changes) *:wq* – write and quit (saves what you changed before exiting the script) *:%s*/what needs to be replaced/what you want to replace it with/g (g stands for "global" and will replace all searched strings in a file). It is recommended to save a copy of a file prior to modifying it, because there is no "undo" option.
cp	To copy a file or folder, provide path where you want to copy it to e.g., *cp scriptXY.sh /data/intermediate/folderXY*
scp	Secure copy – used for copying files from one device to another (e.g., from Linux PC to your PC).
mv	To move a file, provide the path where you want to move it to (e.g., *mv file.txt /data/intermediate*)
mkdir	Will create a new folder in the directory you are currently in (e.g., *mkdir newfolderXY)*
tmux	Terminal multiplexer. It enables a number of terminals (shell windows), each running a separate program, to be created, accessed, and controlled from a single screen. tmux may be detached from a screen and continue running in the background, then later reattached. Run programs/script in tmux mode to prevent the process from breaking (e.g., due to computer going into sleep mode, broken internet connection).
tmux attach	If a remote connection though a shell was interrupted, you can resume an active shell by using *tmux attach* command
exit	After a run is finished you should exit a tmux mode by typing in *exit*.
sh	To run a shell (*sh*) script *scriptXY.sh /data/intermediate/folder XY* (Run script from a folder where it is located and provide the path to the files that will be used by a program that is called in the script). Alternatively, if you are running a python script, you replace *sh* with *python*. This is changed depending on the computer language a script is written in.
.	Current directory. You can use a dot instead of providing the full path to the current directory (e.g., when copying files), however, be careful with its use when running programs (e.g., a current directory may change depending on the code in the script and may change the way the script is interpreted if you do not provide the full path).
*	Using * at beginning or end for names to replace variations e.g., FSL R9* will include all files that are named FSL R9 such as FSL R9-5252, FSL R9-5372,… This command is particularly useful for copying files with the same suffix (e.g., *cp *.fasta /data/intermediate/fasta_files*)
pwd	Print writing directory, to show you the path to the directory you are currently in.
htop	Displays the CPU and memory usage. Type in q to exit the htop.

3.2. Download raw sequencing data

Note: Make sure all programs that you call by running a script are enabled on your personal computer (*zeusctl enable*).

To be able to remotely connect to Linux PC, you need to contact David Kent to set up an account for you.

3.2.1. Download raw sequencing data (when sequencing at the BRC)

(1) BRC will provide a shell script to download fastq sequences (download.sh)
Windows user: Open PuTTY and connect to Zeus; Mac user: Open Terminal and type in *ssh yournetID@zeus.foodscience.cornell.edu* to remotely log in to Linux PC

(2) Use mkdir Command to create a new folder in */data/intermediate*, naming convention: *miseq/hiseq_Ordernumber/date*

(3) Open WinSCP (Windows user) to transfer the *download.sh* script from home computer to the newly created folder on Linux PC. Mac users can transfer files through the Terminal by navigating to the folder in which the script is saved and typing in: *scp download.sh*

yournetID@zeus.foodscience.cornell.edu:/data/intermediate/miseq/hiseq_Ordernumber/date (scp <file_name> <path_to_file_destination>)

(4) Execute *sh download.sh* to download fastq.gz sequences from BRC

3.2.2. Download from BaseSpace (when sequencing at the VetSchool/AHDC)

(1) Install BaseSpace downloader (https://basespace.illumina.com/dashboard)

(2) Download fastq.gz sequences from a run. This will create a folder for each isolate with a pair of sequences forward and reverse files.

(3) Transfer sequences to Linux PC. Create a new folder in */data/intermediate*, naming convention: *miseq/hiseq_Ordernumber/date*

3.2.3. Download SRA files from NCBI

(1) Open PuTTY and connect to Zeus (mac user: Open Terminal and run *ssh zeus* to remotely log in Zeus)

(2) Navigate to the folder you want to download SRA into (use *cd* command)

(3) *wget* ***http://sra-download.ncbi.nlm.nih.gov/srapub/SRR1951985*****,** replace the SRR number in the link with the SRR number for the file you would like to download. Use a script when downloading a batch.

3.3. Sequence assembly and quality control

(1) Open PuTTY and navigate to the folder where sequences are saved (*cd /data/intermediate/..)* (For mac user: Open Terminal and run *ssh* to Linux PC)

(2) Run Trimmomatic from folder with *trimmomatic.sh* script (Appendix A) and fastq files to trim the adapters. The trimmomatic.sh script uses adapters for Nextera XT library (NexteraPE-PE.fa). Make sure you change the adapter sequence file in the script if you used a different library. The program will output the trimmed reads (files ending with trimmed.fastq.gz) and trimmed adapters (trimmedS.fastq.gz). Only trimmedP.fastq.gz files are used in further steps.

Run as follows:
sh trimmomatic.sh <path to fastq.gz files>

Note: This step is needed for raw sequences downloaded from BRC; but is not necessary for sequences from MiSeq platform from VetSchool/AHDC, as the MiSeq software automatically performs trimming on the machine.

(3) Run FastQC from folder with fastqc script (Appendix B) and fastq files to check for quality of fastq files. Several output files will be given; among them will be html files, which need to be inspected in a web browser.

Run as follows:
Mkdir fastqc_out
sh fastqc.sh <path to the files> <path to the fastqc_out> fastq.gz

(4) Open WinSCP and copy html files to home computer (Mac users use *scp* command in Terminal). Open files and control for quality:
This document provides detailed instruction on how to interpret the output of FastQC: https://biof-edu.colorado.edu/videos/dowell-short-read-class/day-4/fastqc-manual

(5) Assemble short reads *de novo* using SPAdes (Appendix C). Make sure your fastq.gz files and *spadesloop.sh* script are in the same folder before running it.

Run as follows:
sh spadesloop.sh <path to the fastq.gz files>

(6) Control the quality of assemblies using QUAST by running *quast.sh* (Appendix D). Note that larger genomes, as well as more "complicated" (e.g., AT rich) genomes have more contigs.

Run as follows:
sh quast.sh <path to contigs files>

(7) Run a python script to combine the files for each sequence into one cvs file. Copy python script into quast_reports folder and run *combine_reports.py* and provide the path (Appendix F).

Run as follows:
Python combine_reports.py

Note: This will combine QUAST results for all isolates in a single file, but you will manually need to add a column with the descriptors (i.e., first column from individual files).

(8) Access the csv file through WinSCP (FileZilla for mac user) and copy to your home computer. Open quast_and coverage_TEMPLATE file from network drive (BoorWiedmannLab > WGS > quast_and coverage_TEMPLATE). In your newly created csv file insert a first new column and copy paste the content of the first column of the quast_and_coverage_TEMPLATE file. Save your newly created cvs as excel format and save on your computer. This file will be part of the report that you save on the network drive (refer to 3.3. Report). Quality indicators are:

a. N50 value (less than 50 000 is an indicator for poor quality; less than 20,000 is considered to be unacceptable).
b. Number of contigs (A very large number is an indicator for poor quality; 30 contigs per Mbp would be considered good).
c. Total length (e.g., the organism's expected genome size is 5Mb, but total length of assembled reads is 9Mb - that indicates contamination, and needs to be addressed more closely).

(9) Run average coverage script, which will align reads in fastq.gz files to assembled contig files and output a text file: average_coverage.txt. Make sure you run *average_coverage.sh* from a location where the script and both file formats are located, and provide full path to the directory with these input files (Appendix E). Text file can be transferred to home computer using WinSCP (FileZilla for mac user) and opened.

Run as follows:
sh average_coverage.sh <path to input contigs.fasta and fastq.gz files>

(10) Open the file WGS_Submission_ordernumber_mmddyyyy_TEMPLATE from network drive (BoorWiedmannLab > WGS > WGS_Submission_ordernumber_mmddyyyy_TEMPLATE) and save on your computer. Add all the requested information using your previously created quast_and_coverage file, as well as Nanodrop and Qubit results.

(11) Remove contigs that are smaller than 200bp by running a shell script that will loop through all contigs.fasta files and execute a python script on them. Make sure both, the shell and *remove_short_contigs.py* and *loop_remove_short_contigs.sh* scripts are in your contigs folder and execute shell script *loop_remove_short_contigs.sh* (Appendix G and Appendix H). This will output logng.fasta files.

Run as follows:
sh loop_remove_short_contigs.sh <path to contigs files and remove_short_contigs.py>

(12) Create a new directory and move long.fasta files and kraken_mul-files.sh script to the new directory (e.g., *mkdir long_fasta*). Make sure the path to kraken database database is still correct, also double-check endings of long.fasta files.

Run as follows:
sh kraken_mul-files.sh <path to the long.fasta files>

Verify that the top hit organism matched the sequenced organism and add the top hit to the WGS_Submission_ordernumber_mmddyyyy spreadsheet (you will have to scroll through the file, longest sequence is not necessarily on the top). If you encounter mismatches between your expected organism and organism identified by Kraken please refer to the trouble shooting section.

(13) For *Salmonella* isolates run seqseroloop.sh to confirm serotypes.

Run as follows:
sh seqseroloop.sh <path to the directory with trimmed.fastq.gz> files

(14) Submit the Whole genome sequence information to the Food Microbe Tracker (FMT; http://www.foodmicrobetracker.com) by filling in the FSM_submission_form.xclx, which is available on the BoorWiedmannLab > WGS.

The submission spreadsheet needs to be sent to Qi Sun (qs24@cornell.edu; cc to jk2739@cornell.edu and martin.wiedmann@cornell.edu), who will batch upload the metadata to the FMT database. The email needs to provide following information:

- Are listed isolates already deposited in the FMT and if not should the new FSL records be created;

- Is there any existing WGS information for the submitted isolates in the FMT and if so, should these data be overwritten;

  Whether you want to be notified in case there are any conflicts or any data could be overwritten.

(15) Submit the sequences to Sequence Read Archive (SRA; NCBI). See #3 Short Read Submission (SRA) SOP for detailed instructions. Make sure you update the FMT records with BioProject and BioSample numbers after submitting sequences to SRA.

(15) Submit the assembled genomes to Whole Genome Shotgun database (WGS; NCBI). See #4 Assembly submission to NCBI SOP for detailed instructions.

3.4. Whole genome sequence quality control report

(1) Create a new folder on the network drive: BoorWiedmannLab > WGS > Order_#Ordernumber_mmddyyyy.

(2) Save the completed WGS_Submission_ordernumber_mmddyyyy file and the quast_and_coverage file.

(3) If a sequencing run was carried out at the Vetschool/AHDC, save the MiSeq Appendix File with library normalization data in the same folder. The Appendix_file_TEMPLATE.xclx is available on the BoorWiedmannLab > WGS.

(4) Report any unusual observation within the WGS_Submission file and if possible provide explanation in trouble shooting tab

3.5. Storage of sequence data on BioHPC

The sequence data are normally stored on the Linux PC while being analyzed. Once the data is submitted to the SRA and/or WGS and the including these sequences is published, all the data can be deleted. You are strongly encouraged to make tree SNP and tree files publicly available in supplemental materials of scientific papers. Since the Linux PC has limited storage capacity, there is a possibility of temporary storage (i.e., during the process of the sequence analysis, prior to publication) of the data on BioHPC (https://cbsu.tc.cornell.edu/lab/lab.aspx).

If you would like to use the BioHPC computational and/or storage resources, please consult with Martin Wiedmann, Jasna Kovac or David Kent. David Kent is an administrator of the FLS BioHPC account and will need to add you into the group to be able to use the resources. Prior to that you will need to create your own BioHPC account as detailed below.

3.5.1. BioHPC set up account

(1) Create a User account https://cbsu.tc.cornell.edu/NewUserRequest.aspx.

(2) Lab ID is your Cornell ID.

(3) David Kent has to add your ID to the FSL BioHPC group to have access to the storage machine (@cbsulogin.tc.cornell.edu) or computational machines (general memory, medium memory or large memory).

(4) Login to the BioHPC through the PuTTY (Windows users: cbsulogin.tc.cornell.edu) or through the *ssh* (Mac user: ssh netID@cbsulogin.tc.cornell.edu).

(5) Access sequence data: *cd /home/dk657_0001/WGS*

Note: Never perform computational work on the BioHPC login (storage) machine; login machine is intended only for data storage, Never modify files stored on the BioHPC login machine; transfer data to Zeus if you would like to run analyses on the files. If you need to use BioHPC computational resources, you need to reserve a computational machine on the following website: https://cbsu.tc.cornell.edu//lab/labres.aspx

There is an option to reserve general memory machine (8 cores, 16GB RAM), medium memory (24 cores, 128GB RAM), large memory machine (64 cores, 512GB RAM). Usually medium memory machine will suffice for our computational needs.

3.4.2. Transfer data from BioHPC to Linux PC or personal computer

(1) Log in to Linux PC with PuTTY (Windows users) or through Terminal (Mac users). In case you are transferring files to your personal computer, there is no need to connect remotely to Linux PC.

(2) Create a new folder or navigate into the folder you would like to transfer the data in using the shell.

(3) Open another shell window and use your NetID to remotely log into BioHPC login (storage) machine (*ssh* netID@*cbsulogin.tc.cornell.edu*). To transfer the files, you need to provide the path to the files on the BioHPC, followed by the path to the destination to which you would like to transfer the files:

```
scp –r netID@cbsulogin.tc.cornell.edu:/home/dk657_0001/WGS/file_or_directory
<path to the destination directory>
```

3.4.3. Transfer data from Linux PC or personal computer to BioHPC

To transfer files to BioHPC, follow the procedure (1) – (2) from 3.4.2, and modify the command from the (3):

```
scp -r <path to the file_or_directory>
netID@cbsulogin.tc.cornell.edu:/home/dk657_0001/WGS/file_or_directory
```

SECTION 4 TROUBLESHOOTING

Script does not run:
If the script does not run properly open the script (*vim)* and check the following:

a. Make sure the file names in the script are correct and match the files in the folder. Especially endings such as ., _, _1, R1 (e.g., fastq.gz files will sometimes have _1.fastq.gz, other times _R1.fastq.gz indicating forward sequence).
b. Make sure the programs that are called in the script are enabled in your shell environment.
c. Make sure the path to the programs is correct and the names and versions of the programs are correct.
d. Make sure you are providing path to the files and not only path to the directory, when necessary.

Kraken maps contigs to unexpected organisms:

a. Sample could have been mixed up on the 96-well plate when preparing library. Check whether more than one sample is not matching? Are two samples switched on the plate?
b. Sample could have been contaminated with an adjacent sample. Check, whether Kraken maps contigs to unexpected organisms also in sequences from other samples on the plate?
c. Kraken could not map a sequence to an organism, because of the database does not contain this organism.
d. The wrong DNA sample could have been submitted.
e. The organism is truly another genus/species than expected.

When Kraken maps a contig to an unexpected organism verify the result by blasting full or partial contigs on NCBI BLAST website:

(1) Choose a few sequences (NODEs) from results file (you can open it with WinSCP for Windows user or FileZilla for Mac user) and transfer to your computer.

(2) Open the transferred sequences with SeqBuilder, it will give you a list and you can choose the number of NODE that you would like to open. Alternatively, open a file with a regular text editor and search for the target NODE.

(3) Copy the sequence and paste into BLAST at NCBI website and BLAST against their nucleotide database. Check for identity percentage and query coverage percentage.

(4) Document your results in your WGS_Submission report excel.

(5) If you do not get any good hits with the BLAST, try to identify an organism by extracting and BLAST-ing the 16S rRNA sequence. Run *sh rnammer.sh <path to contigs files>* to get 16s fasta sequence (Appendix J).

(6) Repeat procedure from step (2) by using BLAST on NCBI website. Document the results in WGS_Submission excel spreadsheet.

(7) Check if sample was contaminated with adjacent samples while preparing the library by checking the locations on 96 well plate – Plate Overview tab in Excel WGS_Submission sheet).

(8) Go back to the glycerol stock, streak out a plate and run a 16s PCR (see SOP for amplifying full length and internal region of 16S) and send for sequencing. This will allow you to check if your glycerol stock is contaminated.

SECTION 5 REFERENCES

FastQC. http://www.bioinformatics.babraham.ac.uk/projects/fastqc/

Trimmomatic. http://www.usadellab.org/cms/?page=trimmomatic

SPAdes. http://bioinf.spbau.ru/spades

QUAST. http://bioinf.spbau.ru/quast

Kraken. https://ccb.jhu.edu/software/kraken/

SECTION 6 Appendix

Appendix A - **trimmomatic.sh**

```
#!/bin/bash
#Matt Stasiewicz 7-1-14
#Use Trimmomatic to trim the raw reads
#sh trimmomatic2.sh <inpath>

#$1=/media/drive2/NYSDOH_ENV_SRA1/MJS

#O: *.trimmed[S/P].fastq.gz files in $gp
#deletes .fastq.gz

#loops through the output from fastq, the _[1/2].fastq.gz and does read trimming
#Raw read trimming with Trimmomatic, ref below.  MJS comments each step of the loop with documentation from:
#http://www.usadellab.org/cms/?page=trimmomatic
#It appears Henk used the default parameters settings for all steps, adjusting file paths and names as appropriate

cd $1
echo | pwd
for f in *_R1.fastq.gz
        do
                if [ -f "${f%_R1.fastq.gz}_R1.trimmedP.fastq.gz" ]
                then
                echo 'skip'${f}
                continue
                fi
        echo 'trim' ${f}
        java -jar /programs/trimmomatic/trimmomatic-0.33.jar PE -threads 7 -phred33 -trimlog log $f
${f%_R1.fastq.gz}_R2.fastq.gz ${f%_R1.fastq.gz}_R1.trimmedP.fastq.gz ${f%_R1.fastq.gz}_R1.trimmedS.fastq.gz
${f%_R1.fastq.gz}_R2.trimmedP.fastq.gz ${f%_R1.fastq.gz}_R2.trimmedS.fastq.gz
ILLUMINACLIP:/programs/trimmomatic/adapters/NexteraPE-PE.fa:2:30:10 LEADING:3 TRAILING:3
SLIDINGWINDOW:4:15 MINLEN:36;
done;

#rm *_1.fastq.gz
#rm *_2.fastq.gz
```

Appendix B - **fastqc.sh**

```
#!/bin/bash
#Matt Stasiewicz 3-7-14
#Use fastq to perform intial qc and save results
# sh <inpath> <outpath> <suff>

#$1=/media/drive2/NYSDOH_ENV_SRA1/MJS
#$2=/media/drive2/NYSDOH_ENV_SRA1/MJS/fastqc_res
#$3=[1-2].fastq.gz

#O: fastqc html in $gp/fastqc_res

cd $1
for file in *$3
    do
        cd $2
        if [ -d "${file%.fastq.gz}_fastqc" ]
        then
        echo 'skip'${file}
        continue
        fi
        cd $1
    echo 'fastqc 1'${file}
    /usr/bin/fastqc $file -o $2;
done
```

Appendix C - **spadesloop.sh**

```
#!/bin/bash

#Matt Stasiewicz 7-1-14 modified by LC Carroll 12-18-14 by S Harrand 10-12-16 by jk2739 10-26-16
#Use SPades to assemble the genome
#nohup sh spades.sh <inpath>

cd $1
#run spades
for f in *_R1_001.fastq.gz
#insert an exit if matches in dir
do
if [ -d "${f%_R1_001.fastq.gz}" ]
then
echo 'skip '${f}
continue
fi
echo 'assemble' ${f%_R1_001.fastq.gz}
python /programs/spades-3.6.2/bin/spades.py -k 21,33,55,77,99,127 --careful -1 $f -2
${f%_R1_001.fastq.gz}_R2_001.fastq.gz -o ${f%_R1_001.fastq.gz} -t 7 -m 20;
done
#check the created log file for any issues

#collect contigs files and rename them
mkdir contigs
for f in *_R1_001.fastq.gz
do
	cd ${f%_R1_001.fastq.gz}
	cat contigs.fasta > ${f%_R1_001.fastq.gz}_contigs.fasta
	cp ${f%_R1_001.fastq.gz}_contigs.fasta ../contigs
	cd ..;
done

mkdir scaffolds
for f in *_R1_001.fastq.gz
do
	cd ${f%_R1_001.fastq.gz}
	cat scaffolds.fasta > ${f%_R1_001.fastq.gz}_scaffolds.fasta
	cp ${f%_R1_001.fastq.gz}_scaffolds.fasta ../scaffolds
```

Appendix D - **quast.sh**

```
#!/bin/bash
#QUAST - assembly quality control
#jk2739
#112415

cd $1
mkdir quast_results

for f in *.fasta
do
python /programs/quast/quast.py -o ./quast_results/quast_${f%_contigs.fasta} --min-contig 1 $f
done

#collect report txt files
mkdir quast_reports
for f in *.fasta
do cd $1/quast_results/quast_${f%_contigs.fasta}
cat report.txt > ${f%_contigs.fasta}_report.txt
cp ${f%_contigs.fasta}_report.txt $1/quast_reports
done
```

Appendix E – **average_coverage.sh**

```
#!/bin/bash
# average_coverage.sh <path to directory with contigs and reads>
# written November 2, 2015 by LC Carroll
# Shout out to Matt S. for giving me the bam_coverage.sh script

cd $1
# BBMap to determine coverage
for f in *__contigs.fasta
do
echo "Indexing $f with BBMap..."
/programs/bbmap/bbmap.sh ref=$f
echo "Mapping reads to $f with BBMap..."
/programs/bbmap/bbmap.sh in=${f%__contigs.fasta}_R1_001.fastq.gz in2=${f%__contigs.fasta}_R2_001.fastq.gz
out=${f%__contigs.fasta}.sam
echo "SAM file created.  BBMap finished."
mv ref/ ${f%__contigs.fasta}_ref/

# Now let's use samtools to covert, sort, and index
echo "Converting SAM to BAM with samtools..."
/programs/samtools-1.3.1/bin/samtools view -Sb ${f%__contigs.fasta}.sam > ${f%__contigs.fasta}.bam
echo "BAM file created."
echo "Removing sam file..."
rm -r *.sam
echo "Sorting BAM file with samtools..."
/programs/samtools-1.3.1/bin/samtools sort ${f%__contigs.fasta}.bam -o ${f%__contigs.fasta}_sorted.bam
echo "Finished sorting."
echo "Indexing sorted BAM file..."
/programs/samtools-1.3.1/bin/samtools  index ${f%__contigs.fasta}_sorted.bam
echo "Index complete."
echo "Using samtools depth to obtain average genome coverage..."
X=$(/programs/samtools-1.3.1/bin/samtools depth ${f%__contigs.fasta}_sorted.bam | awk '{sum+=$3} END { print
sum/NR}');
echo "${f%__contigs.fasta}_sorted.bam";
echo "$X";
echo "${f%__contigs.fasta}_sorted.bam $X">> average_coverage.txt;
done
```

Appendix F – **combine_reports.py**

```
#!/usr/bin/python
import glob

# Create list to store information before we write it to a file. Each item will
# be a list containing the information from one file
quast_list = list()

# Loop over all text files in the current directory
for file_name in glob.glob('*.txt'):
	file_handle = open(file_name)
	file_contents = file_handle.readlines()
	file_handle.close()
	file_data = list()
	# Loop through lines 3 to the end, split on whitespace, and grab the last item
	for line in file_contents[2:]:
		file_data.append(line.split()[-1])
	quast_list.append(file_data)

combined_reports = ''
for line in range(0,len(quast_list[0])):
	combined_reports += ','.join([ fsl[line] for fsl in quast_list]) + '\n'

handle = open('combined.csv','w')
handle.write(combined_reports)
handle.close()
```

Appendix G – **loop_remove_short_contigs.sh**

```
#!/bin/bash
#jk2739
#Loop through the contig files and remove contigs shorter than 200 bp
#Usage: Run the script from the directory with the loop_remove_short_contigs.sh, remove_short_contigs.py, contigs fasta
files
#Usage: sh loop_remove_short_contigs.sh .


for f in *_contigs.fasta
do
python remove_short_contigs.py $f ${f%.fasta}_long.fasta
done
```

Appendix H – **remove_short_contigs.py**

```
#!/bin/python
#Remove contigs shorter than 200 bp
#jk2739
#092716
#Usage: run from a directory with contigs file and a script
#Usage: python remove_short_contigs.py <infile.fasta> <outfile.fasta>

import sys
from Bio import SeqIO

infile = sys.argv[1]
parsed_infile = SeqIO.parse(open(infile,"rU"), "fasta")
remove_short = (contig for contig in parsed_infile if len(contig.seq) > 200)

outfile= sys.argv[2]
output = open(outfile, "w")
SeqIO.write(remove_short, output, "fasta")
output.close()
```

Appendix I – **kraken_mul-files.sh**

```
#!/bin/sh

#  kraken for many fasta files.sh
#
#
#  Created by Jingqiu Liao on 7/16/16.
#

for f in *_contigs.fasta_long.fas
do
kraken --preload --db /data/intermediate/ho85_database/minikraken_20141208 --fasta-input $f --classified-out
${f%_contigs.fasta_long.fas}_c.fas --unclassified-out ${f%_contigs.fasta_long.fas}_u.fas >
${f%_contigs.fasta_long.fas}.kraken
done

mkdir kraken_result
for f in *.kraken
do
kraken-translate --db /data/intermediate/ho85_database/minikraken_20141208 $f > ./kraken_result/${f%.kraken}.labels
done
```

Appendix J – **rnammer.sh**

```
#!/bin/bash
#batch RNAmmer search and extraction of 16s rDNA sequences
#jk2739
#102715

mkdir rnammer_results
for f in *.fasta
do
/programs/rnammer/rnammer -S bac -m ssu -gff ./rnammer_results/${f%.fasta}_16s.gff -f ./rnammer_results/${f%.fasta}_16s.fasta < $f
awk '/^>/{if(N)exit;++N;} {print;}' ./rnammer_results/${f%.fasta}_16s.fasta > ./rnammer_results/${f%.fasta}_1st_16s.fasta
cat ./rnammer_results/*_1st_16s.fasta > ./rnammer_results/16s_cat.fasta
done
```

Appendix K – **seqseroloop.sh**

```
#!/bin/bash
# sh seqseroloop.sh <inpath to directory with .trimmedP.fastq.gz paired end reads>
# Runs SeqSero on trimmedP.fastq.gz reads and organizes output
# SeqSero is used to do in silico serotyping on Salmonella
# LC Carroll June 19th, 2015

cd $1

mkdir SeqSero

for f in *_R1.trimmedP.fastq.gz
do
        SeqSero -m 2 -i ./$f ./${f%_R1.trimmedP.fastq.gz}_R2.trimmedP.fastq.gz
        echo 'SeqSero is done'
        echo 'Moving files...'
        path=$(find $1 -name 'Seqsero_result.txt')
        mv $path SeqSero/"${f%_R1.trimmedP.fastq.gz}_seqsero_results.txt"
done
```