CGR-Microbiome QIIME2 Pipeline – Many Flowcells

Shalabh Suman

Contents

Data being used to test this pipeline is from: NP0452-MB5

Step 0: Preparing the Global Configuration File, Project Manifest and Environments

Step 0.1: Setting up the Global Configuration File, Minicondia and QIIME-2 environments

```
#!/bin/bash

# QIIME version
qiime_version=2017.11

# Initiate QIIME
module load miniconda/3
source activate qiime2-${qiime_version}


# Parameters

QUEUE=queue.q

SAMPLE_PREFIX=SC

Phred_score=33

# Directories

PROJECT_DIR=Project_Qiime2

TEMP_DIR=${PROJECT_DIR}/Input/tmp
```

```
LOG_DIR=${PROJECT_DIR}/Input/Log

SCRIPT_DIR=ss_scripts_microbiome_analysis/ss_scripts_qiime2_pipeline_V1

RESOURCES_DIR=${SCRIPT_DIR}/resources

FASTA_DIR=${PROJECT_DIR}/Input/Fasta

FASTA_DIR_TOTAL=${PROJECT_DIR}/Input/Fasta_Total

MANIFEST_FILE_SPLIT_PARTS_DIR=${PROJECT_DIR}/Input/manifest_file_split_parts

MANIFEST_FILE_SPLIT_PARTS_FASTQ_IMPORT_DIR=${PROJECT_DIR}/Input/
manifest_file_split_parts_fastq_import

QZA_RESULTS_DIR=${PROJECT_DIR}/Input/qza_results
QZV_RESULTS_DIR=${PROJECT_DIR}/Input/qzv_results

# Here the Directories are generated

mkdir -p $TEMP_DIR 2>/dev/null
mkdir -p $LOG_DIR 2>/dev/null
mkdir -p $MANIFEST_FILE_SPLIT_PARTS_DIR 2>/dev/null
mkdir -p $MANIFEST_FILE_SPLIT_PARTS_FASTQ_IMPORT_DIR 2>/dev/null
mkdir -p $FASTA_DIR 2>/dev/null
mkdir -p $FASTA_DIR_TOTAL 2>/dev/null
mkdir -p $QZA_RESULTS_DIR 2>/dev/null
mkdir -p $QZV_RESULTS_DIR 2>/dev/null

# Files

MANIFEST_FILE_XLSX=${PROJECT_DIR}/NP0452-MB5-manifest.xlsx

MANIFEST_FILE_TXT_WITH_HEADER=${PROJECT_DIR}/manifest.txt

MANIFEST_FILE=${PROJECT_DIR}/manifest_no_header.txt

MANIFEST_FILE_qiime2_format=${PROJECT_DIR}/Input/manifest_qiime2.tsv

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

## Stage 3

# Stage 3 Directories

demux_qza_split_parts_dir=${QZA_RESULTS_DIR}/demux_qza_split_parts

demux_qzv_split_parts_dir=${QZV_RESULTS_DIR}/demux_qzv_split_parts

table_dada2_qza_split_parts_dir=${QZA_RESULTS_DIR}/table_dada2_qza_split_parts

repseqs_dada2_qza_split_parts_dir=${QZA_RESULTS_DIR}/repseqs_dada2_qza_split_parts
```

```
log_dir_stage_3=${LOG_DIR}/stage3_qiime2

# Stage 3 Parameters

demux_param=paired_end_demux

table_dada2_param=table_dada2

repseqs_dada2_param=repseqs_dada2


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

## Stage 4: Merging

# Stage 4 Directories

table_dada2_qza_merged_parts_tmp_dir=${QZA_RESULTS_DIR}/table_dada2_qza_merged_parts_tmp

table_dada2_qza_merged_parts_final_dir=${QZA_RESULTS_DIR}/
table_dada2_qza_merged_parts_final

repseqs_dada2_qza_merged_parts_tmp_dir=${QZA_RESULTS_DIR}/
repseqs_dada2_qza_merged_parts_tmp

repseqs_dada2_qza_merged_parts_final_dir=${QZA_RESULTS_DIR}/
repseqs_dada2_qza_merged_parts_final

log_dir_stage_4=${LOG_DIR}/stage4_qiime2

# Stage 4 Parameters

table_dada2_merged_temp_param=table_dada2_merged_temp

table_dada2_merged_final_param=table_dada2_merged_final

repseqs_dada2_merged_temp_param=repseqs_dada2_merged_temp

repseqs_dada2_merged_final_param=repseqs_dada2_merged_final

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

## Stage 5: Feature-Table & Seqs Summary

# Stage 5 Directories

table_dada2_qzv_merged_parts_final_dir=${QZV_RESULTS_DIR}/
table_dada2_qzv_merged_parts_final

repseqs_dada2_qzv_merged_parts_final_dir=${QZV_RESULTS_DIR}/
repseqs_dada2_qzv_merged_parts_final

log_dir_stage_5=${LOG_DIR}/stage5_qiime2
```

```
# Stage 5 Parameters

table_dada2_merged_final_param=table_dada2_merged_final

repseqs_dada2_merged_final_param=repseqs_dada2_merged_final



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

## Stage 6: Phylogenetic Tree Analysis

# Stage 6 Directories

phylogeny_qza_dir=${QZA_RESULTS_DIR}/phylogeny_qza_results

log_dir_stage_6=${LOG_DIR}/stage6_qiime2

# Stage 6 Parameters

output1_param=aligned_rep_seqs
output2_param=masked_aligned_rep_seqs
output3_param=unrooted_tree
output4_param=rooted_tree



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

## Stage 7: Alpha-Beta Diversity Analysis

# Stage 7 Directories

core_metrics_output_dir=${QZA_RESULTS_DIR}/core_metrics_results

log_dir_stage_7=${LOG_DIR}/stage7_qiime2

# Stage 7 Parameters

sampling_depth=10000


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


## Stage 8: Rarefaction Analysis

# Stage 8 Directories

rarefaction_qzv_dir=${QZV_RESULTS_DIR}/rarefaction_qzv_results

log_dir_stage_8=${LOG_DIR}/stage8_qiime2
```

```
# Stage 8 Parameters


rarefaction_param=rarefaction

max_depth=62000



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


## Stage 9: Taxonomic Analysis

# Stage 9 Directories

taxonomy_qza_dir=${QZA_RESULTS_DIR}/taxonomy_qza_results
taxonomy_qzv_dir=${QZV_RESULTS_DIR}/taxonomy_qzv_results

log_dir_stage_9=${LOG_DIR}/stage9_qiime2

# Stage 9 Parameters

refernce_1=${RESOURCES_DIR}/gg-13-8-99-nb-classifier.qza
refernce_2=${RESOURCES_DIR}/silva-119-99-nb-classifier.qza

taxonomy_1=taxonomy_greengenes
taxonomy_2=taxonomy_silva
```

Step 0.2: Processing the Manifest File Provided by the Lab to check for errors and converting into Unix Fomart TXT

```
sh script_xlsx2txt.sh
```

Code: script_xlsx2txt.sh

```
#!/bin/bash

. ./global_config_bash.rc


echo "INPUT=$MANIFEST_FILE_XLSX"
echo "Output1=$MANIFEST_FILE_TXT_WITH_HEADER"
echo "Output2=$MANIFEST_FILE"
echo "Output3=$MANIFEST_FILE_qiime2_format"

module load python3

input_file=$1
cmd="python3 ./common_xlsx2txt.pl $MANIFEST_FILE_XLSX $MANIFEST_FILE_TXT_WITH_HEADER"
echo $cmd
```

```
eval $cmd

tail -n +2 $MANIFEST_FILE_TXT_WITH_HEADER > ${MANIFEST_FILE}

#Here we convert the TXT-Manifest into QIIME-version of Manifest
cmd="sh ${SCRIPT_DIR}/prepare_second_mapping_file.sh"
echo $cmd
eval $cmd
```

Code: common_xlsx2txt.pl

```
import sys
import os
import openpyxl


workbook=openpyxl.load_workbook(sys.argv[1])
InputFile=sys.argv[1]
print ('\n')
print ('Hey There!')
print ('\n')
print ('Your Input Excel File Name is = %s' %InputFile)
print ('\n')
#OutputFile=InputFile.split('.')[0] + '.txt'
OutputFile=sys.argv[2]
print ('Your Output Tab-Delimited Text File Name is = %s' %OutputFile)
print ('\n')
#print type(workbook)
#print (workbook.get_sheet_names())

sheet1=workbook.get_sheet_names()[0]
#print (sheet1)

worksheet=workbook.get_sheet_by_name(sheet1)
#print worksheet

tarray=[[] for i in range(worksheet.max_row)]

#print worksheet.cell(row=25,column=1).value
#print type(worksheet.max_column)
#print type(worksheet.max_row)

for i in range(worksheet.max_row):
    if worksheet.cell(row=i+1,column=1).value is not None:
        for j in range(worksheet.max_column):
            #print worksheet.cell(row=i+1,column=j+1).value
            tarray[i].append(worksheet.cell(row=i+1,column=j+1).value)

#print tarray

tarray_clean= [x for x in tarray if x != []]
#print tarray_clean
#print tarray[1][2]
```

```
numrows = len(tarray_clean)
numcols = len(tarray_clean[0])
#print numrows
#print numcols

text_file = open(OutputFile, "w")

for b in range(numrows):
    for c in range(numcols):
        if tarray_clean[b][c] is None:
            print ('', end='\t', file=text_file)
        else:
            print (tarray_clean[b][c], end='\t', file=text_file)
    print ('', file=text_file)

print ('Its all done!')
print ('\n')
```

Code: prepare_second_mapping_file.sh

```
#!/bin/bash

. ./global_config_bash.rc


echo " Here we will process the Original Manifest File and generate the corresponding
      'QIIME-compatible' version"

echo "#SampleID" > ${MANIFEST_FILE_qiime2_format}
for i in $(cat $MANIFEST_FILE| awk -F "\t" -v spx=${SAMPLE_PREFIX} '
{print $1"#"$8"#"$2"_"$7"#"$10}'); do
SN=$(echo ${i}|cut -f1 -d'#');
FN=$(echo $i | cut -f2 -d'#');
ID=$(echo ${i}|cut -f3 -d'#' | sed s/-/_/g);
PN=$(echo $i | cut -f4 -d'#');
#echo $ID;
#echo $PN;
echo -e $ID;
done >> ${MANIFEST_FILE_qiime2_format}


echo "DONE"
echo "****************"
echo
```

Step 1: Dividing the Manifest File into Flowcell-Level Many-Manifest Files

```
sh divide_samples_by_flowcell_from_manifest.sh
```

Code: divide_samples_by_flowcell_from_manifest.sh

```
# Code: divide_samples_by_flowcell_from_manifest.sh
#!/bin/bash

echo
echo "Author: Shalabh Suman"

. ./global_config_bash.rc

    echo
    echo "Here we will process the Manifest File to divide the samples by flowcell"
    echo "Based on the template of Manifest file that was decided, we are assuming RUN ID
          field is Column 8"
    echo
    echo "Original Mapping File=${MANIFEST_FILE}"

    count_for_flowcells=$(cat $MANIFEST_FILE | awk -F "\t" '{print $8}' | sort | uniq |
    wc -l)
    echo "Total number of Unique Flowcells = $count_for_flowcells"
    echo

    names_for_flowcells=$TEMP_DIR/$(basename $MANIFEST_FILE .txt)_names_for_flowcells.txt
    cat $MANIFEST_FILE | awk -F "\t" '{print $8}' | sort | uniq > $names_for_flowcells
    echo "File with List of Unique Flowcells = $names_for_flowcells"
    echo
    #echo $names_for_flowcells
    echo "List of Unique Flowcells:"
    cat $names_for_flowcells;
    echo
    echo


    count=0;
    for i in $(cat $names_for_flowcells | awk -F "\t" '{print $1}'); do

        count=$(( count + 1 ))

        run_id=$i;
        #echo $run_id;

        echo "Part $count RUN ID = $run_id"

        manifest_file_split_part=$MANIFEST_FILE_SPLIT_PARTS_DIR/$(basename $MANIFEST_FILE
        .txt)_split_part_${count}.txt

        echo "Part $count Manifest File = $manifest_file_split_part"

        cat $MANIFEST_FILE | awk -F "\t" -OF "\t" -v spx=$run_id '{if ($8 == spx)
        {print $0}}' > $manifest_file_split_part

        echo

        #exit 1;
```

```
    done

    echo "All done"
```

Step 2: Generate Fastq Folders, Consolidate into TOTAL-FASTQ Folder & Generate the Fastq-import Sample-sheet files

- Generating the Fastq Folders for each of the Flowcell-level Samples
- Consolidating all the Fastq Files into one TOTAL-FASTQ Folder
- & Generating the Fastq-import Sample-sheet files from Flowcell-Level Many-Manifest Files

```
sh prepare_sample_sheet_for_fastq_import.sh
```

Code: prepare_sample_sheet_for_fastq_import.sh

```
#!/bin/bash

. ./global_config_bash.rc



rm -rf ${FASTA_DIR}/*
rm -rf ${FASTA_DIR_TOTAL}/*
rm -rf ${MANIFEST_FILE_SPLIT_PARTS_FASTQ_IMPORT_DIR}/*

count=0

for manifest_file_split_part in $(ls -v ${MANIFEST_FILE_SPLIT_PARTS_DIR}/*txt); do

        count=$(( count + 1 ))


        # Directories
        fasta_dir_split_part=${FASTA_DIR}/fasta_dir_split_part_${count}
        mkdir -p $fasta_dir_split_part 2>/dev/null

        manifest_file_split_parts_fastq_import=${MANIFEST_FILE_SPLIT_PARTS_FASTQ_IMPORT_DIR}/
        manifest_file_split_parts_fastq_import_${count}.txt



        echo "Analysis for Part $count"
        echo "Input Part Manifest file = $manifest_file_split_part"
        echo "Output Part Fasta Directory = $fasta_dir_split_part"
        echo "Output Part Fastq-Demuliplexed-Sample-Sheet =
        $manifest_file_split_parts_fastq_import"


        # Step 1: FastQ Folder Generation Starts
        echo
        echo "Step 1: Here we will process the manifest file and locate the sample level
              Fasta files from the flowcell directory, then make a soft link for those
```

```
      files in the project directory"


for i in $(cat $manifest_file_split_part | awk -F "\t" -v spx=${SAMPLE_PREFIX} '
{print $1"#"$8"#"$10}'); do
    SN=$(echo $i|cut -f1 -d'#');
    FN=$(echo $i | cut -f2 -d'#');
    PN=$(echo $i | cut -f3 -d'#');
    #echo $PN;
    #echo "OK";
    FRP=$(find /DCEG/CGF/Sequencing/Illumina/MiSeq/PostRun_Analysis/Data/${FN}/
    CASAVA/L1/Project_${PN}/Sample_${SN}/${SN}*R1_001.fastq.gz);
    RRP=$(find /DCEG/CGF/Sequencing/Illumina/MiSeq/PostRun_Analysis/Data/${FN}/
    CASAVA/L1/Project_${PN}/Sample_${SN}/${SN}*R2_001.fastq.gz);
    #echo $FRP
    FRN=$(basename ${FRP});
    #echo $FRN
    CFR="ln -fs ${FRP} ${FRN}";
    RRN=$(basename ${RRP});
    CRR="ln -fs ${RRP} ${RRN}";
    cd $fasta_dir_split_part;
    #echo $CFR;
    eval $CFR;
    #echo $CRR;
    eval $CRR;
    #exit 1;
done


# Step 2: FastQ Files Collection Starts
echo
echo "Step 2: Here we will make a Copy of the FastQ files (links) inside
      Production Data Directory"
cmd="cp -P ${fasta_dir_split_part}/*fastq.gz ${FASTA_DIR_TOTAL}"
echo $cmd
eval $cmd


# Step 3: Fastq-Demuliplexed-Sample-Sheet Generation Starts
echo
echo "Step 3: Here we will process the Split Manifest File and generate the
      corresponding 'Fastq-Demuliplexed-Sample-Sheet' that will be utilized for
      generation of QIIME artifacts"

echo "sample-id,absolute-filepath,direction" >
${manifest_file_split_parts_fastq_import}
for i in $(cat $manifest_file_split_part| awk -F "\t" -v spx=${SAMPLE_PREFIX} '
{print $1"#"$8"#"$2"_"$7"#"$10}'); do
    SN=$(echo ${i}|cut -f1 -d'#');
    FN=$(echo $i | cut -f2 -d'#');
    ID=$(echo ${i}|cut -f3 -d'#' | sed s/-/_/g);
    PN=$(echo $i | cut -f4 -d'#');
    #echo $ID;
```

```
            #echo $PN;
            FRP=$(find /DCEG/CGF/Sequencing/Illumina/MiSeq/PostRun_Analysis/Data/${FN}/
            CASAVA/L1/Project_${PN}/Sample_${SN}/${SN}*R1_001.fastq.gz);
            RRP=$(find /DCEG/CGF/Sequencing/Illumina/MiSeq/PostRun_Analysis/Data/${FN}/
            CASAVA/L1/Project_${PN}/Sample_${SN}/${SN}*R2_001.fastq.gz);
            echo -e $ID","$FRP",forward";
            echo -e $ID","$RRP",reverse";
            #exit 1;
        done >> ${manifest_file_split_parts_fastq_import}

        #exit 1;

        echo "Part $count DONE"
        echo "****************"
        echo
done
```

Step 3: Import fastq, Generate demultiplex Visualization & table construction with DADA2

- Import Casava 1.8 paired-end demultiplexed fastq
- Generate a Visualization for the demultiplexing results
- & Sequence quality control and feature table construction using DADA2

```
sh run_qiime2_by_flowcell_stage_1_on_cluster.sh
```

Code: run_qiime2_by_flowcell_stage_1.sh

```
#!/bin/bash

. ./global_config_bash.rc


demux_qza_split_part=$1
#demux_qza_split_part=${demux_qza_split_parts_dir}/paired_end_demux_1.qza
shift

demux_qzv_split_part=$1
#demux_qzv_split_part=${demux_qzv_split_parts_dir}/paired_end_demux_1.qzv
shift

table_dada2_split_part=$1
#table_dada2_split_part=${table_dada2_qza_split_parts_dir}/table_dada2_1.qza
shift

repseqs_dada2_split_part=$1
#repseqs_dada2_split_part=${repseqs_dada2_qza_split_parts_dir}/repseqs_dada2_1.qza
shift

pe_manifest=$1
shift
```

```
# Importing Casava 1.8 paired-end demultiplexed fastq
date
echo "Here we import Casava 1.8 paired-end demultiplexed fastq "
echo "INPUT = $pe_manifest"
echo "OUTPUT = $demux_qza_split_part"
echo
cmd="qiime tools import \
    --type 'SampleData[PairedEndSequencesWithQuality]' \
    --input-path ${pe_manifest} \
    --output-path ${demux_qza_split_part} \
    --source-format PairedEndFastqManifestPhred${Phred_score}"


echo $cmd
eval $cmd
echo
date
echo "Done"
echo
echo


# Generating a Visualization for the demultiplexing results

date
echo "Here we generate a Visualization for the demultiplexing results"
echo "INPUT = $demux_qza_split_part"
echo "OUTPUT = $demux_qzv_split_part"
echo
cmd="qiime demux summarize \
    --i-data ${demux_qza_split_part} \
    --o-visualization ${demux_qzv_split_part}"

echo $cmd
eval $cmd
echo
date
echo "Done"
echo
echo


# Sequence quality control and feature table construction: DADA2

date
echo "Here we perform sequence quality control and feature table construction using
      DADA2 Plugin"
echo "INPUT = $demux_qza_split_part"
echo "OUTPUT 1= $table_dada2_split_part"
echo "OUTPUT 2= $repseqs_dada2_split_part"
echo

cmd="qiime dada2 denoise-paired \
```

```
    --i-demultiplexed-seqs ${demux_qza_split_part} \
    --o-table ${table_dada2_split_part} \
    --o-representative-sequences ${repseqs_dada2_split_part} \
    --p-trim-left-f 0 \
    --p-trim-left-r 0 \
    --p-trunc-len-f 0 \
    --p-trunc-len-r 0"


echo $cmd
eval $cmd

echo
date
echo "Done"
echo
echo
```

More information about importing other file formats can be found here

Step 4: Merge Frequency and Sequence Artifacts

- Merge the Flowcell-Level Many FeatureTable[Frequency] artifacts
- Merge the Flowcell-Level Many FeatureData[Sequence] artifacts

```
sh merge_table_seq_for_many_flowcells_on_cluster.sh
```

Code: merge_table_seq_for_many_flowcells.sh

```
#!/bin/bash

. ./global_config_bash.rc



TOTAL_RUNS=$(ls -v $MANIFEST_FILE_SPLIT_PARTS_FASTQ_IMPORT_DIR/* | wc -l)
echo $TOTAL_RUNS

# Tables Merging

count=1

while [ $count -le 1 ]

do
    echo $count

    part1=$count
    part2=$(( count + 1 ))

    input_table_qza_1=${table_dada2_qza_split_parts_dir}/${table_dada2_param}_${part1}.qza
    input_table_qza_2=${table_dada2_qza_split_parts_dir}/${table_dada2_param}_${part2}.qza
```

```
    output_table_temp_qza=${table_dada2_qza_merged_parts_tmp_dir}/
    ${table_dada2_merged_temp_param}_${part2}.qza

    cmd="qiime feature-table merge \
        --i-table1 $input_table_qza_1 \
        --i-table2 $input_table_qza_2 \
        --o-merged-table $output_table_temp_qza"

    echo $cmd
    eval $cmd


    count=$(( count + 1 ))

done



while [ $count -gt 1 ] && [ $count -lt $TOTAL_RUNS ]

do
    echo $count

    part1=$count
    part2=$(( count + 1 ))

    input_table_qza_1=${table_dada2_qza_merged_parts_tmp_dir}/
    ${table_dada2_merged_temp_param}_${part1}.qza
    input_table_qza_2=${table_dada2_qza_split_parts_dir}/${table_dada2_param}_${part2}.qza
    output_table_temp_qza=${table_dada2_qza_merged_parts_tmp_dir}/
    ${table_dada2_merged_temp_param}_${part2}.qza

    cmd="qiime feature-table merge \
        --i-table1 $input_table_qza_1 \
        --i-table2 $input_table_qza_2 \
        --o-merged-table $output_table_temp_qza"

    echo $cmd
    eval $cmd


    count=$(( count + 1 ))

done

last_true_part=$count
echo $last_true_part
output_table_merged_temp_qza=${table_dada2_qza_merged_parts_tmp_dir}/
${table_dada2_merged_temp_param}_${last_true_part}.qza
output_table_merged_final_qza=${table_dada2_qza_merged_parts_final_dir}/
${table_dada2_merged_final_param}.qza
cmd="cp ${output_table_merged_temp_qza} ${output_table_merged_final_qza}"
echo $cmd
```

```
eval $cmd



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

# Rep-Seqs Merging


count=1

while [ $count -le 1 ]

do
    echo $count

    part1=$count
    part2=$(( count + 1 ))

    input_repseqs_qza_1=${repseqs_dada2_qza_split_parts_dir}/
    ${repseqs_dada2_param}_${part1}.qza
    input_repseqs_qza_2=${repseqs_dada2_qza_split_parts_dir}/
    ${repseqs_dada2_param}_${part2}.qza
    output_repseqs_temp_qza=${repseqs_dada2_qza_merged_parts_tmp_dir}/
    ${repseqs_dada2_merged_temp_param}_${part2}.qza

    cmd="qiime feature-table merge-seq-data \
        --i-data1 $input_repseqs_qza_1 \
        --i-data2 $input_repseqs_qza_2 \
        --o-merged-data $output_repseqs_temp_qza"

    echo $cmd
    eval $cmd


    count=$(( count + 1 ))

done



while [ $count -gt 1 ] && [ $count -lt $TOTAL_RUNS ]

do
    echo $count

    part1=$count
    part2=$(( count + 1 ))

    input_repseqs_qza_1=${repseqs_dada2_qza_merged_parts_tmp_dir}/
    ${repseqs_dada2_merged_temp_param}_${part1}.qza
    input_repseqs_qza_2=${repseqs_dada2_qza_split_parts_dir}/
    ${repseqs_dada2_param}_${part2}.qza
```

```
    output_repseqs_temp_qza=${repseqs_dada2_qza_merged_parts_tmp_dir}/
    ${repseqs_dada2_merged_temp_param}_${part2}.qza

    cmd="qiime feature-table merge-seq-data \
        --i-data1 $input_repseqs_qza_1 \
        --i-data2 $input_repseqs_qza_2 \
        --o-merged-data $output_repseqs_temp_qza"


    echo $cmd
    eval $cmd


    count=$(( count + 1 ))

done

last_true_part=$count
echo $last_true_part
output_repseqs_merged_temp_qza=${repseqs_dada2_qza_merged_parts_tmp_dir}/
${repseqs_dada2_merged_temp_param}_${last_true_part}.qza
output_repseqs_merged_final_qza=${repseqs_dada2_qza_merged_parts_final_dir}/
${repseqs_dada2_merged_final_param}.qza
cmd="cp ${output_repseqs_merged_temp_qza} ${output_repseqs_merged_final_qza}"
echo $cmd
eval $cmd
```

Step 5: Generate Visualizations and Summaries for Merged Frequency and Sequence Artifacts

- Generate information on how many sequences are associated with each sample and with each feature, histograms of those distributions, and some related summary statistics
- Generate a mapping of feature IDs to sequences, and provide links to easily BLAST each sequence against the NCBI nt database

```
sh generate_table_seq_summary_on_cluster.sh
```

Code: merge_table_seq_for_many_flowcells.sh

```
#!/bin/bash

. ./global_config_bash.rc


input_table_merged_final_qza=$1
shift
output_table_merged_final_qzv=$1
shift

input_repseqs_merged_final_qza=$1
shift
```

```
output_repseqs_merged_final_qzv=$1
shift

Manifest_File=$1
shift

#Generate information on how many sequences are associated with each sample and with each
#feature, histograms of those distributions, and some related summary statistics
date
echo "Here we Generate information on how many sequences are associated with each sample
      and with each feature, histograms of those distributions, and some related summary
      statistics "
echo "INPUT1 = ${input_table_merged_final_qza}"
echo "INPUT2 = ${Manifest_File}"
echo "OUTPUT = ${output_table_merged_final_qzv}"
echo

cmd="qiime feature-table summarize \
    --i-table ${input_table_merged_final_qza} \
    --o-visualization ${output_table_merged_final_qzv}"
#   --m-sample-metadata-file ${Manifest_File}

echo $cmd
eval $cmd
echo
date
echo "Done"
echo
echo


# Generate a mapping of feature IDs to sequences, and provide links to easily BLAST each
# sequence against the NCBI nt database
date
echo "Here we generate a mapping of feature IDs to sequences, and provide links to easily
      BLAST each sequence against the NCBI nt database "
echo "INPUT = ${input_repseqs_merged_final_qza}"
echo "OUTPUT = ${output_repseqs_merged_final_qzv}"
echo

cmd="qiime feature-table tabulate-seqs \
    --i-data ${input_repseqs_merged_final_qza} \
    --o-visualization ${output_repseqs_merged_final_qzv}"

echo $cmd
eval $cmd
echo
date
echo "Done"
echo
```

Step 6: Generate a tree for phylogenetic diversity analyses

```
sh generate_phylogenetic_analysis_on_cluster.sh
```

Code: generate_phylogenetic_analysis.sh

```
#!/bin/bash

. ./global_config_bash.rc


#mkdir -p ${phylogeny_qza_dir} 2>/dev/null

#mkdir -p $log_dir_stage_6 2>/dev/null

input_repseqs_merged_final_qza=$1
shift
output1_qza=$1
shift
output2_qza=$1
shift
output3_qza=$1
shift
output4_qza=$1
shift

#Step 1: First, we perform a multiple sequence alignment of the sequences
date
echo "Here we perform a multiple sequence alignment of the sequences"
echo "INPUT = ${input_repseqs_merged_final_qza}"
echo "OUTPUT = ${output1_qza}"
echo
cmd="qiime alignment mafft \
    --i-sequences ${input_repseqs_merged_final_qza} \
    --o-alignment ${output1_qza}"
echo $cmd
eval $cmd
echo
date
echo "Done"
echo
echo

#Step 2: Here, we mask (or filter) the alignment to remove positions that are highly
# variable
date
echo "Here we mask (or filter) the alignment to remove positions that are highly variable"
echo "INPUT = ${output1_qza}"
echo "OUTPUT = ${output2_qza}"
echo
cmd="qiime alignment mask \
  --i-alignment ${output1_qza} \
```

```
  --o-masked-alignment ${output2_qza}"
echo $cmd
eval $cmd
echo
date
echo "Done"
echo
echo


#Step 3: Here, we'll apply FastTree to generate a phylogenetic tree from the masked
# alignment
date
echo "Here we'll apply FastTree to generate a phylogenetic tree from the masked alignment"
echo "INPUT = ${output2_qza}"
echo "OUTPUT = ${output3_qza}"
echo
cmd="qiime phylogeny fasttree \
  --i-alignment ${output2_qza} \
  --o-tree ${output3_qza}"
echo $cmd
eval $cmd
echo
date
echo "Done"
echo
echo


#Step 4: Here, we apply midpoint rooting to place the root of the tree at the midpoint of
# the longest tip-to-tip distance in the unrooted tree
date
echo "Here  we apply midpoint rooting to place the root of the tree at the
      midpoint of the longest tip-to-tip distance in the unrooted tree"
echo "INPUT = ${output3_qza}"
echo "OUTPUT = ${output4_qza}"
echo
cmd="qiime phylogeny midpoint-root \
  --i-tree ${output3_qza} \
  --o-rooted-tree ${output4_qza}"

echo $cmd
eval $cmd
echo
date
echo "Done"
echo
echo
```

Step 7: Perform Alpha and Beta Diversity Analysis

```
sh generate_alpha_beta_diversity.sh
```

Code: generate_alpha_beta_diversity.sh

```
#!/bin/bash

. ./global_config_bash.rc


input_table_merged_final_qza=$1
shift
input_rooted_tree_qza=$1
shift
output_dir=$1
shift
Manifest_File=$1
shift
sampling_depth=$1
shift

#perform Alpha and beta diversity analysis
date
echo "Here we perform Alpha and beta diversity analysis "
echo "INPUT1:Tree = ${input_rooted_tree_qza} "
echo "INPUT2:Table = ${input_table_merged_final_qza}"
echo "INPUT3:Manifest-File = ${Manifest_File}"
echo "Sampling-Depth = ${sampling_depth}"
echo "OUTPUT-DIR = ${output_dir}"
echo

cmd="qiime diversity core-metrics-phylogenetic \
      --i-phylogeny ${input_rooted_tree_qza} \
      --i-table ${input_table_merged_final_qza} \
      --p-sampling-depth ${sampling_depth} \
      --m-metadata-file ${Manifest_File} \
      --output-dir ${output_dir}"

echo $cmd
eval $cmd
echo
date
echo "Done"
echo
echo

## Here we Export the Alpha-Diversity Artifacts as an integrated Visualization Table"

cmd="qiime metadata tabulate \
    --m-input-file ${output_dir}/observed_otus_vector.qza \
    --m-input-file ${output_dir}/shannon_vector.qza \
```

```
    --m-input-file ${output_dir}/evenness_vector.qza \
    --m-input-file ${output_dir}/faith_pd_vector.qza \
    --o-visualization ${output_dir}/alpha-table.qzv"

echo $cmd
eval $cmd
echo
date
echo "Done"
echo
echo
```

Step 8: Perform Rarefaction Analysis

```
sh perform_rarefaction_analysis_on_cluster.sh
```

Code: perform_rarefaction_analysis.sh

```
#!/bin/bash

. ./global_config_bash.rc

input_table_merged_final_qza=$1
shift
input_rooted_tree_qza=$1
shift
Manifest_File=$1
shift
alpha_rarefaction_qzv=$1
shift
max_depth=$1
shift

cmd="qiime diversity alpha-rarefaction \
    --i-table ${input_table_merged_final_qza} \
    --i-phylogeny ${input_rooted_tree_qza} \
    --p-max-depth ${max_depth} \
    --m-metadata-file ${Manifest_File} \
    --o-visualization ${alpha_rarefaction_qzv}"

echo $cmd
eval $cmd
echo
date
echo "Done"
echo
echo
```

Step 9: Perform Taxonomic Analysis

```
sh perform_taxonomic_analysis_on_cluster.sh
```

Code: perform_taxonomic_analysis.sh

```
#!/bin/bash

. ./global_config_bash.rc


input_table_merged_final_qza=$1
shift
input_repseqs_merged_final_qza=$1
shift
Manifest_File=$1
shift
reference_classifier_1=$1
shift
reference_classifier_2=$1
shift
taxonomy_qza_1=$1
shift
taxonomy_qza_2=$1
shift
taxonomy_qzv_1=$1
shift
taxonomy_qzv_2=$1
shift
taxa_bar_plots_qzv_1=$1
shift
taxa_bar_plots_qzv_2=$1
shift


# Taxonomic Classification for GreenGenes Refernce

cmd="qiime feature-classifier classify-sklearn \
    --i-classifier ${reference_classifier_1} \
    --i-reads ${input_repseqs_merged_final_qza} \
    --o-classification ${taxonomy_qza_1}"

echo $cmd
eval $cmd
echo
date
echo "Done"
echo
echo

cmd="qiime metadata tabulate \
    --m-input-file ${taxonomy_qza_1}\
```

```
    --o-visualization ${taxonomy_qzv_1}"

echo $cmd
eval $cmd
echo
date
echo "Done"
echo
echo

cmd="qiime taxa barplot \
    --i-table ${input_table_merged_final_qza} \
    --i-taxonomy ${taxonomy_qza_1} \
    --m-metadata-file ${Manifest_File} \
    --o-visualization ${taxa_bar_plots_qzv_1}"

echo $cmd
eval $cmd
echo
date
echo "Done"
echo
echo



# Taxonomic Classification for Silva Refernce

cmd="qiime feature-classifier classify-sklearn \
    --i-classifier ${reference_classifier_2} \
    --i-reads ${input_repseqs_merged_final_qza} \
    --o-classification ${taxonomy_qza_2}"

echo $cmd
eval $cmd
echo
date
echo "Done"
echo
echo

cmd="qiime metadata tabulate \
    --m-input-file ${taxonomy_qza_2}\
    --o-visualization ${taxonomy_qzv_2}"

echo $cmd
eval $cmd
echo
date
echo "Done"
echo
echo
```

```
cmd="qiime taxa barplot \
    --i-table ${input_table_merged_final_qza} \
    --i-taxonomy ${taxonomy_qza_2} \
    --m-metadata-file ${Manifest_File} \
    --o-visualization ${taxa_bar_plots_qzv_2}"

echo $cmd
eval $cmd
echo
date
echo "Done"
echo
echo
```

This document was processed on: 2018-03-04