Reference-Based Bacterial Genome Assembly on the Command Line

In this document, we have provided instructions for performing a reference-based assembly of *Vibrio cholerae*. You can perform the steps on raw sequencing data you have generated and copied onto your computer (see instructions to *set up your project directory* below).

Important notes for following this tutorial:

- Text with a gray background in `monospace font` represents commands to type in. Generally commands are one line, however, in this document, commands might wrap to the next line visually. We will add a blank line between commands to indicate when multiple commands are present.
- Bold text surrounded by **< >** is something you will have to replace with your own folder, path, or sample name.
- This tutorial assumes you have set up the bioinformatics pipeline directory on your computer Before starting the tutorial below, check if your computer has a folder called `bacpage/` in the home directory. If you do **not** have this folder on your computer, complete the Bioinformatics Pipeline Setup instructions to install the necessary files and software before proceeding.

This tutorial will take you through reference-based assembly from amplicon sequencing data. The process is comprised of five main steps, including a sequence assessment step:

1. Align paired-end FASTQ files to a reference genome, generating BAM files.
2. Call variants in the BAM file relative to the reference.
3. Generate a consensus sequence from the variants.
4. Mask known recombinant regions from the consensus sequence
5. Assess the quality of raw sequencing reads and alignment.

These steps are performed using the `snakemake` platform. Snakemake is a tool for creating reproducible and modular bioinformatic pipelines. Each task in an analysis, including those above, can be written as an individual step of a pipeline. Snakemake makes it easier to conduct analyses by reducing the number of commands you need to type., parallelizes steps across all your samples, and confirms steps were completed successfully.

Below, you'll find instructions to use snakemake to run the whole pipeline all at once, as well as instructions (in the *Appendix*) that walk you through each of the above steps one by one.

STEP 0: Project Directory Setup [*always required*]

Before starting any bioinformatics analysis, it is good practice to create a directory specifically for the dataset you are about to analyze. This directory is where you will run analyses and save outputs and is called your *project directory*. You will want to set up your project directory inside the `bacpage/` folder in your HOME directory to ensure all sequencing data and results are in the same place and can use the same pipeline tools and software.

We will refer to the path of the `bacpage/` folder as <**`sequencing-path`**>. On most machines, the <**`sequencing-path`**> will be `~/bacpage`.

1. Navigate to the bioinformatics pipeline folder as described above:

```
cd ~/bacpage
```

2. In the `bacpage/` folder there is a sub-folder called `example/`. The first step is to make a copy of this folder. You will make a new copy every time you perform a new sequencing run, thus ensuring that the files and folders inside the example directory are set up in the exact same way each time. To copy this folder and give it a project-specific name, run the command below. We recommend that you give your project directory an informative name, such as <date>_<sequencing-run-name> (for example: `20220609_cholera_run1`).

   Type the following into your terminal window and then press **Enter**:

```
cp -R example/ <project-directory-name>
```

3. Navigate to your project directory and record the absolute path using.

```
cd <project-directory-name>

pwd
```

   We will refer to the absolute path to your project directory as <**`project-path`**> You will need the absolute path for your project directory in *Step 6* below.

4. Locate the input data you want to use in the assembly pipeline. This pipeline requires demultiplexed FASTQ files (i.e., two FASTQ files per sample). Using the file finder on your computer, move this data into the `input/` directory of your project directory (i.e., the folder you just created above).

5. Within your project directory, there should be a file called `sample_data.csv`, which will hold the information about your samples. Open this file in Excel or a similar spreadsheet software, and add the information for each of your samples on an individual row.

   In the **sample** column, record the name or identifier of each sample (these must be unique and not contain unusual characters like periods "." or slashes "/"). In the **read1** column, record the absolute path of the

FASTQ file corresponding to the first set of reads for a sample (generally containing “R1” in the filename). In the **read2** column, record the absolute path of the FASTQ file corresponding to the first set of reads for a sample (generally containing “R2” in the filename).

The absolute path of a file can be determined with the `pwd` command. If your demultiplexed FASTQ files were placed in the `input/` directory of your project directory as described above, you can find their absolute path by running the following command:

```
cd <project-path>/input

pwd
```

The absolute path of your files will be the output of `pwd` plus “/” plus the file name. An example completed `sample_data.csv` file is shown below:

sample	read1	read2
sample1	/home/user/bacpage/20220609_run1/input/sample1_S13_L001_R1_001.fastq.gz	/home/user/bacpage/20220609_run1/input/sample1_S13_L001_R2_001.fastq.gz
sample2	/home/user/bacpage/20220609_run1/input/sample2_S3_L001_R1_001.fastq.gz	/home/user/bacpage/20220609_run1/input/sample2_S3_L001_R2_001.fastq.gz
sample3	/home/user/bacpage/20220609_run1/input/sample3_S22_L001_R1_001.fastq.gz	/home/user/bacpage/20220609_run1/input/sample3_S22_L001_R2_001.fastq.gz

6. Save the file after entering the data for your samples.

7. Within your project directory, open the configuration file called `config.yaml` with a text editor of your choice. This file contains the parameters and options for the analysis. Parameters may be altered to suit your analysis, however the default options should be appropriate for most analysis.

   With the configuration file open in a text editor, replace <**project-path**> and <**sequencing-path**> with their *absolute paths* for the first six parameters.

```
# General parameters; point to input files.
run_type: "Illumina"
samples: "<project-path>/sample_data.csv"
output_directory: "<project-path>"
reference: "<sequencing-path>/resources/vc_reference.fasta"
reference_genes: "<sequencing-path>/resources/cholera_ref_genes/"
recombinant_mask: "<sequencing-path>/resources/cholera_mask.gff"
```

8. Lastly, determine how many processors are available on your computer, since using more processors will make the pipeline run faster. The output of the following command will indicate how many processors you have available. Run the command below and write down the result for later:

```
cat /proc/cpuinfo | grep processor
```

STEP 1: Run the Complete Pipeline

If you would like to run the entire pipeline without walking through each of the intermediate steps, you can generate consensus sequences and calculate quality metrics with a single command.

1. Navigate to the location of the bioinformatics pipeline:

```
cd ~/bacpage
```

2. Run the pipeline with the following command:

```
snakemake --configfile <project-path>/config.yaml --cores
<number-of-processors> --keep-going --until mask_consensus
generate_complete_report
```

This will generate a consensus sequence in FASTA format for each of your samples and place them in `<project-path>/results/consensus_sequences/<sample>.masked.fasta`. An HTML report containing alignment and quality metrics for your samples can be found at `<project-path>/results/reports/qc_report.html`.

If a snakemake command is successfully completed, you should see something like this on the screen:

```
[Tue Sep  5 12:09:40 2023]
Finished job 119.
253 of 253 steps (100%) done
Complete log: .snakemake/log/2023-09-05T120148.784555.snakemake.log
```

Note: The job number, step, and total steps will depend on the number of samples you have.

If a snakemake command was unsuccessful, you will see the following at the end of the output:

```
Error in rule x:
    jobid: 1
    shell:
        command -options arguments
Shutting down, this might take some time.
Exiting because a job execution failed. Look above for error message
Complete log: .snakemake/log/2023-09-05T120148.784555.snakemake.log
```

Record the failed rule, the sample being processed, and the expected output file of the failed rule. You can attempt to rerun the incomplete step by running the command again:

```
snakemake --configfile <project-path>/config.yaml --cores
<number-of-processors> --keep-going
```

STEP 2: Assess the quality of Genome Assembly

A key component of generating genome assemblies is assessing their quality. The bioinformatics pipeline generates a report to visually inspect the quality, coverage, and confidence we have in the resultant consensus sequences.

1. To review the quality metrics of your samples, open `<**project-path**>/results/reports/qc_report.html` with a web browser.

Appendix to Genome Assembly on the Command Line

The following instructions will describe the intermediate steps of generating the consensus sequences. This process can be optionally done instead of running the complete pipeline using the instructions above.

STEP 1: Align Paired-End FASTQ Files to a Reference Genome

The first genome assembly step is to take the paired-end FASTQ files produced by a sequencing machine and align them against a reference genome. This will generate a BAM file for each sample that is needed for future steps of the pipeline.

Note: by default, the pipeline will align reads against a *Vibrio cholerae* reference. If you are studying another pathogen, you can change the "*reference*" parameters in <**project-path**>`/config.yaml` to another reference sequence.

1. Navigate your current working directory to the location of the bioinformatics pipeline (generally `~/bacpage`). All of the following steps should be conducted in this directory. Type the following into your terminal window and then press **Enter**:

```
cd ~/bacpage
```

2. Run the alignment_bwa step of the pipeline using the following command.

```
snakemake --configfile <project-path>/config.yaml --cores
<number-of-processors> --keep-going --until alignment_bwa
```

   This command individually aligns each sample indicated in the `sample_data.csv` file, and will take a few minutes depending on the number of samples.

   A BAM file will be generated for each sample with the format <**sample**>`.sorted.bam`. BAM files for every sample can be found in <**project-path**>`/intermediates/illumina/alignments/`.

STEP 2: Call Variants Relative to Reference Genome

The next step in genome assembly is to determine differences between the sequenced reads and the reference genome. These differences are called variants and describe evolution of the sample from the reference genome (typically the earliest genome for an organism).

1. Run the variant calling step of the pipeline using the following command:

```
snakemake --configfile <project-path>/config.yaml --cores
<number-of-processors> --keep-going --until
align_and_normalize_variants
```

This command calls variants for each sample indicated in the `sample_data.csv` file, filters low-quality and unsupported variants, and normalizes insertions and deletions. A Variant Call Format file (VCF file) will be generated for each sample with the format <**sample**>`.filt.norm.vcf.gz` (see https://en.wikipedia.org/wiki/Variant_Call_Format for a description of the VCF format). VCF files for every sample can be found in <**project-path**>`/intermediates/illumina/variants/`

STEP 3: Generate a Consensus Sequence

We will now generate a genome sequence for each sample by applying the variants to our reference sequence. Because this genome is a summary of many sequencing reads, we call this sequence a consensus sequence.

1. Run the consensus calling step of the pipeline using the following command:

```
snakemake --configfile <project-path>/config.yaml --cores
<number-of-processors> --keep-going --until call_consensus
```

This command generates a consensus sequence for each sample indicated in the `sample_data.csv` file. A FASTA file containing the consensus sequence will be generated for each sample with the format <**sample**>`.consensus.fasta`. FASTA files for every sample can be found in <**project-path**>`/intermediates/illumina/consensus/`.

STEP 4: Mask the Consensus Sequence

We recommend masking regions of the genome that have low coverage and/or are wholly recombinant. Masking these regions prevents erroneous results from downstream analyses including typing and phylogenetic inference. Low coverage regions can be determined directly from the BAM file generated for each sample, while wholly recombinant regions require prior knowledge of the organism being studied. As part of the pipeline, we have provided a file indicating the wholly-recombinant regions of the cholera genome. If you are studying another organism, you will need to update the `recombinant_mask` parameter in the config file to another organism-specific mask.

1. To run the masking step of the pipeline, use the following command:

```
snakemake --configfile <project-path>/config.yaml --cores
<number-of-processors> --keep-going --until mask_consensus
```

This command masks the consensus sequence for each of your samples. A FASTA file containing the masked consensus sequence will be generated for each sample with the format <**sample**>`.masked.fasta`. Masked FASTA files for every sample can be found in:
<**project-path**>`/intermediates/illumina/consensus/`

STEP 5: Assess Assembly Quality

A key component of generating genome assemblies is assessing their quality. We will generate reports to visually inspect the quality, coverage, and confidence we have in the resultant consensus sequences.

1. Generate the quality control reports using the following command:

```
snakemake --configfile <project-path>/config.yaml --cores
<number-of-processors> --keep-going --until generate_complete_report
```

   This command generates a single HTML report containing quality metrics for all samples. The HTML report can be found in <**project-path**>`/results/reports/qc_report.html`.

2. Open this file with a web browser.