Step 2 - Evaluate your sample quality

For Step 2, we need to evaluate the quality of your sequencing samples. We'll be using AMR++ and one of it's sub-workflows, "eval_qc", to create html reports using fastQC and multiQC.

Table of contents

- Step 2 - Evaluate your sample quality
- Confirm the AMR++ environment is installed on your server
  - Running conda on TAMU's HPRC
- Find the location of your reads
- Download the AMR++ github repository
- Use AMR++ to run "eval_qc"
  - AMR++ demonstration
  - The "--reads" flag
  - The "--pipeline" flag
  - The "--output" flag
  - Putting it all together for the final command
- Evaluate the output
- Next steps

Confirm the AMR++ environment is installed on your server

If you already have "miniconda" or "conda" installed and know how to access the AMR++_env on the server, skip to the next step.

There are various ways to install AMR++ and more information can be found in the AMR++ github repository.

If you're using TAMU's HPRC, we have local installations you can use but we recommend that you install "miniconda" for easy handling of AMR++ and other software. Run the following commands to install miniconda:

```
# Download miniconda
wget https://repo.anaconda.com/miniconda/Miniconda3-py310_23.11.0-2-Linux-x86_64.sh

# The next command will initiate the installation.
# You'll click "ENTER" to review the license.
# Hold "ENTER" until you reach the bottom (or read it carefully!)
# Answer "yes" if you accept the license terms.
# The final question asks about the location of the installation.
# We recommended inputing the path to somewhere that is not your home directory and clicking "ENTER"(i.e. scratch space).
# Finally, click "yes" to initiate the installation and allow for bash to automatically load the "base" conda environment.
```

```
bash Miniconda3-py310_23.11.0-2-Linux-x86_64.sh
```

Next time you login, you'll notice the word "(base)" to the left of the prompt in the command. This means that a conda environment is loaded and you can now search for any tool you want to still by searching "conda install fastQC". Usually, you'll find a single command that you can use to easily install software. To activate this without having to log back in, run the following command which runs the file that automatically sets up your environment everytime you log in.

```
source ~/.bashrc
```

Now you'll the the word "(base)" to the left of your name. This means conda is loaded and you can now run the command below to check what environments are available. Check this page for more information on using conda.

```
conda env list
```

The other useful thing about conda and miniconda, is that you can have multiple environments with different tools/versions installed. This can become a challenge with older software which can have specific version requirements for it to function properly.

Running conda on TAMU's HPRC

On TAMU's HPRC, we installed an environment with all of the AMR++ tools installed.

- Grace
  - `/scratch/group/vero_research/conda_envs/AMR++_env`
- Terra
  - `/scratch/group/morley_group/bin/AMR++_env`

For example, to activate the environment on Grace you can run this:

```
conda activate /scratch/group/vero_research/conda_envs/AMR++_env
```

You'll notice that the left of your prompt now has the name of the AMR++ environment. You can now run all of the tools needed for the AMR++ pipeline.

Find the location of your reads

Once you've downloaded your fastq samples to the server, navigate to this location in the terminal and run `pwd` to get the absolute path to your reads.

```
cd /path/to/reads # based on your directory structure
pwd
```

We'll use this absolute path with the `--reads` flag in the next step to tell AMR++ what samples to analyze.

Download the AMR++ github repository

The AMR++ environment we discussed above is different than the actual code that is needed to run AMR++. Above, we set up our computing environment to have the multitude of software that AMR++ needs to run your samples through the pipeline (e.g. fastQC, bwa, kraken2, etc.).

However, this is different than having access to the AMR++ code. The two main ways to do this is by: 1) download the repository to a single location, or 2) download the repository for each project that you are going to analyze. Because AMR++ can be updated on the github, you would have to ensure that you update it before running it for a new project using `git pull origin master`. This can be difficult to keep track of, so I like to download the repository for each project, this way I have a "snapshot" of the current version of AMR++ that was used for each project.

Either way, on the terminal you can use this command to download AMR++:

```
git clone https://github.com/Microbial-Ecology-Group/AMRplusplus
```

Use AMR++ to run "eval_qc"

You'll eventually learn how to use AMR++ and all it's various components as shown in this document. For now, here are major points you need to understand for this tutorial.

- how to run the AMR++ demonstration
- how to add certain "flags" or "arguments" to change how AMR++ functions.
    - `--reads` to point to your sample sequences
    - `--pipeline` to change what analysis AMR++ runs
    - `--output` to name the output folder
- download results for evaluation

AMR++ demonstration

To run the AMR++ pipeline demonstration, you just need to navigate into the repository and run the following command (with the conda environment activated):

```
cd AMRplusplus/
nextflow run main_AMR++.nf
```

This works because AMR++ looks for the default settings in the "params.config" file which is in the main directory.

However, we need to change a few things to make this work with your data. We could modify these flags directly in the "params.config" file or add them to the command above.

The "--reads" flag

Now we need to add the `--reads` flags so that your samples are analyzed. Often, samples are sequenced as paired reads (forward and reverse). So that AMR++ can handle paired reads, we need to use something called "regular expressions", which consist of letters (strings) that have a different meaning programmatically, to provide the pattern matching your reads. For example, if your samples are named like this:

```
Sample_1_S23_L004_R1_001.fastq.gz
Sample_1_S23_L004_R2_001.fastq.gz
Sample_2_S46_L004_R1_001.fastq.gz
Sample_2_S46_L004_R2_001.fastq.gz
```

One pattern that you can use to match it is "*fastq.gz", like this:

```
# Assuming you are in the same directory as your samples
ls *fastq.gz
```

However, we have paired samples here and we want to tell AMR++ how to keep them together. To do this, you can use brackets to specify that at a certain location in the name, you can expect only a limited option of characters. Such as for forward and reverse reads, like this:

```
ls *R{1,2}_001.fastq.gz
```

Finally, you need to add the absolute path to the location of your reads like this:

```
ls /path/to/reads/*R{1,2}_001.fastq.gz
```

If that works, then you're ready for the next part as that location and regular expression will be passed to the `--reads` flag so that AMR++ can find your samples like this (notice the quotes):

```
--reads "/path/to/reads/*R{1,2}_001.fastq.gz"
```

The "--pipeline" flag

Now, AMR++ needs to know what type of analysis to run on your samples and we can do that by specifying the "--pipeline" flag. Read the AMR++ documents for more information, but here we'll need the "eval_qc" pipeline. This will use fastQC and multiQC to create a html report for you to visualize the sample quality on a browser.

So, you'll add this to the AMR++ command:

```
--pipeline eval_qc
```

The "--output" flag

Finally, AMR++ needs to know where to put the output. Since you'll be running the pipeline from the AMRplusplus folder you downloaded, you can just use a new name for the output folder like this:

```
--output AMR++_results_folder
```

Putting it all together for the final command

Now let's combine all those pieces into a single command in an sbatch script. Below is example of what would work on TAMU's Grace HPRC.

It assumes you are running the script from the AMRplusplus directory.

Here's an sbatch script that you can copy and paste into the server, remember to fill out the command to match the location of your reads. Submit is as usual with `sbatch YOUR_script_name.sbatch`

```
#!/bin/bash
#SBATCH -J AMR++ -o AMR++.out -t 6:00:00 --nodes=1 --ntasks-per-node=1 --cpus-per-task=7 --mem=50G
# Remember to change the "queueSize" parameter in the config/*_slurm.config file that you are using. This corresponds with "--ntasks" to control how many jobs are submitted at once.
# This script works on TAMU's HPRC, but you need to follow the instructions on the Github to get the right conda environment installed on your computing environment
source activate AMR++_env

nextflow run main_AMR++.nf --reads "/path/to/reads/*R{1,2}_001.fastq.gz" --pipeline eval_qc --output AMR++_results_folder
```

Evaluate the output

You can see the progress by looking at the hidden file, ".nextflow.log" to see how it's doing or just check your queue with `squeue --me` to confirm it's still running. When it's completed, look at the results within your output folder, for the example above it's "AMR++_results_folder". Download the two files in the directory

"QC_analysis/MultiQC_stats/". This will include the "AMR-Bioinformatic-pipeline_multiqc_report.html" file which has interactive figures you can explore in a browser. The second file, "multiqc_general_stats.txt", has results in a table format which can be opened in excel.

Next steps

Now that we know the quality scores of our data, we have to pick our trimming parameters and actually perform the QC trimming using AMR++ and the `--pipeline trim_qc`.