# Genome Assembly Tutorial: Batch Processing

## Learning Objectives

In this tutorial you will learn

1. What a workflow manager is and why they are used
2. How Nextflow works as a workflow manager
3. How to run Nextflow to assemble multiple samples

## Tutorial

### What is a workflow manager and why do you need one?

A workflow manager is a method to run a series of predefined steps on data. It has the benefits of

- Handling the process of taking a batch of samples and applying several bioinformatics tools to them in a specific order
- Processing the steps in parallel where possible.

It also enables consistency and reproducibility so that it can be assured that the same data set processed using the same workflow will give the same results. This also means that a workflow can easily be shared between collaborators or made public.

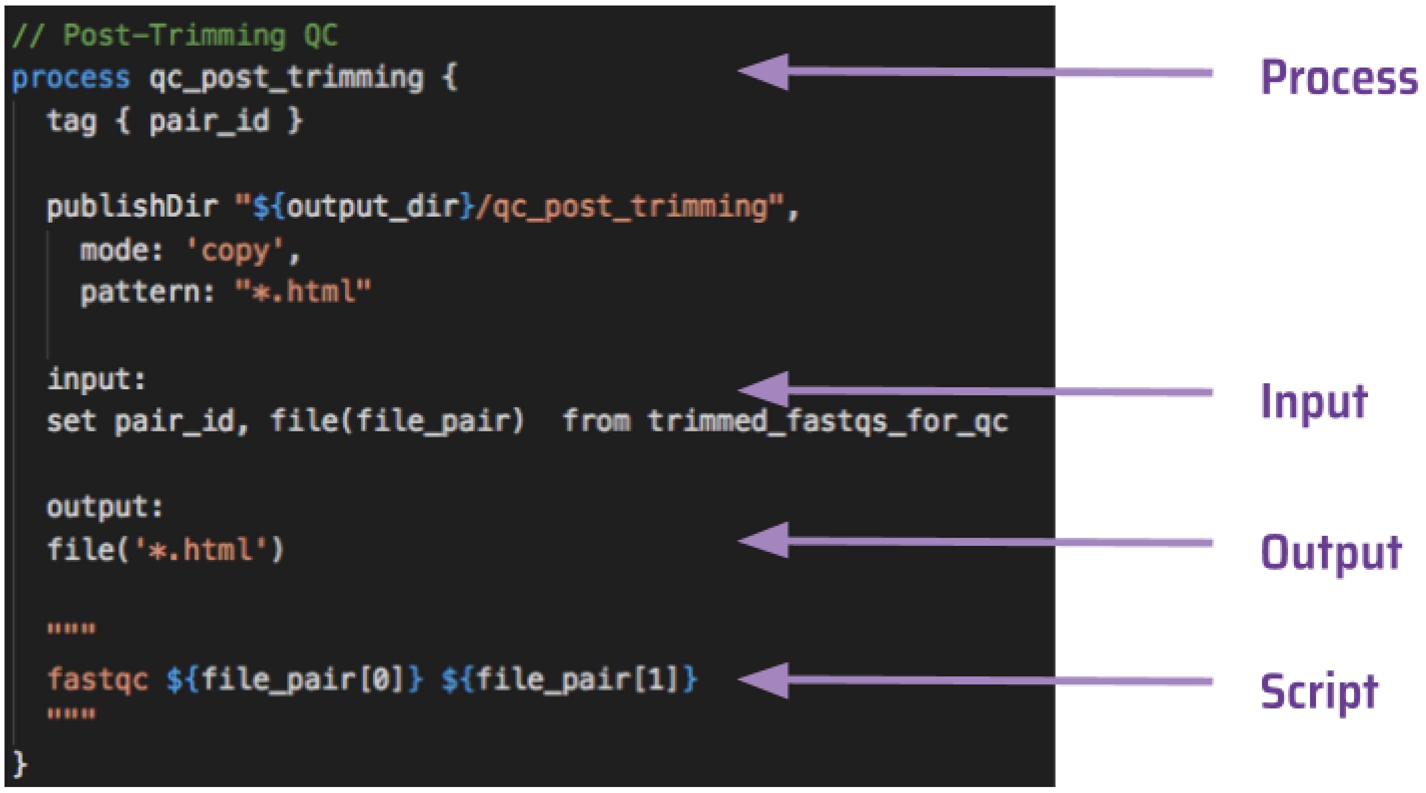
### Nextflow

Nextflow is one example of workflow manager software. It uses a text file to describe the steps (or processes as it terms them) within a workflow. This workflow can be executed on several different computing environments such as locally on a laptop, using a high performance computing cluster (HPC), using containers such as Docker or remotely in the cloud for example with Amazon's Amazon Web Services (AWS).
More details can be found here in its published paper and in the documentation.

When you run Nextflow you specify a workflow file (usually ending in the suffix .nf) that describes the steps that will be executed throughout the workflow. In the workflow file each step is called a process. The structure of a process is shown below. Some of the details you observe are beyond the scope of this tutorial and therefore will not be described here but the essentials are labelled with yellow arrows.
Within each process are the following sections

- input: determines from which previous process the required files for this step are derived. In this case they are from a trimming step
- output: determines which output files produced in this process will be passed onto the next process or will be kept for future use
- lines wrapped in tripe quotes (""")  are the script itself and you may recognise the fastqc command from the previous command line tutorial and observe that two items from a file pair are used within the script.

These are the read 1 and read 2 files that were produced from a previous trimming step using Trimmomatic



```
// Post-Trimming QC
process qc_post_trimming {
  tag { pair_id }

  publishDir "${output_dir}/qc_post_trimming",
    mode: 'copy',
    pattern: "*.html"

  input:
  set pair_id, file(file_pair)  from trimmed_fastqs_for_qc

  output:
  file('*.html')

  """
  fastqc ${file_pair[0]} ${file_pair[1]}
  """
}
```

In the workflow file used for this tutorial there are a series of processes that match each step that was run manually as part of the command line assembly tutorial. Nextflow will work out which processes are able to be run in parallel and by default will perform as many parallel processes as the computer has processor/CPU cores.

When running Nextflow the command takes the format

**`nextflow run <WORKFILE FILE> [-with-docker <DOCKER IMAGE>] -resume <WORKFLOW OPTIONS>`**

## Containers

The option `-with-docker` parameter species that nextflow will use a docker container to supply the dependencies needed to run the pipeline which in this case is software such as fastqc, trimmomatic, SPAdes etc

Why is it often a good idea to use containers? Installing all the necessary software for a workflow can be a complicated process and challenging for those beginning bioinformatics.
So although each software tool can be installed on an individual basis it is often best to use a container if one that includes all the software dependencies exists. For this tutorial a container specifically for the purpose of running the workflow has been made and can be found here but since all the software has been pre-installed in the VM we won't use it in this case.

What is a container? In a nutshell, a container bundles up an operating system and software dependencies into a package that can be run in any environment allowing deployment of all the software necessary to run a pipeline or application in  a very straightforward way.

Here are two articles that do a good job of explaining this in more detail
Introduction to virtual machines and Docker containers
Explanation of container technology

The `-resume` option will allow the workflow to be restarted at the point where it stopped if for any reason it has crashed, rather than having to start the workflow from the beginning.

The workflow options will be specific to the workflow being employed. In the case of this assembly workflow there are 4 parameters

1) `--input_dir` This is the path to the directory containing the fastq files to be assembled
2) `--fastq_pattern` This is the fuzzy match pattern that defines how to group the pairs of fastq files. So for files named sample1_R1.fastq.gz/sample1_R2.fastq.gz this would have the pattern `‘*_R{1,2}.fastq.gz’` where `*` means any character and `{1,2}` means either 1 or 2.
3) `--adapter_file` This is the path to the file containing the adapter sequences
4) `--output_dir` This is the path to the output directory

## Running Nextflow on multiple samples

### Setting up data for the tutorial

1. SSH into the VM using PuTTY and create a directory to contain the input fastq files. Name this for example `fastq_data`
2. Use WinSCP to copy the 2 pairs of fastqs used in the CLI tutorial (samples ERR668456 and ERR586796) into this directory

### Run Nextflow

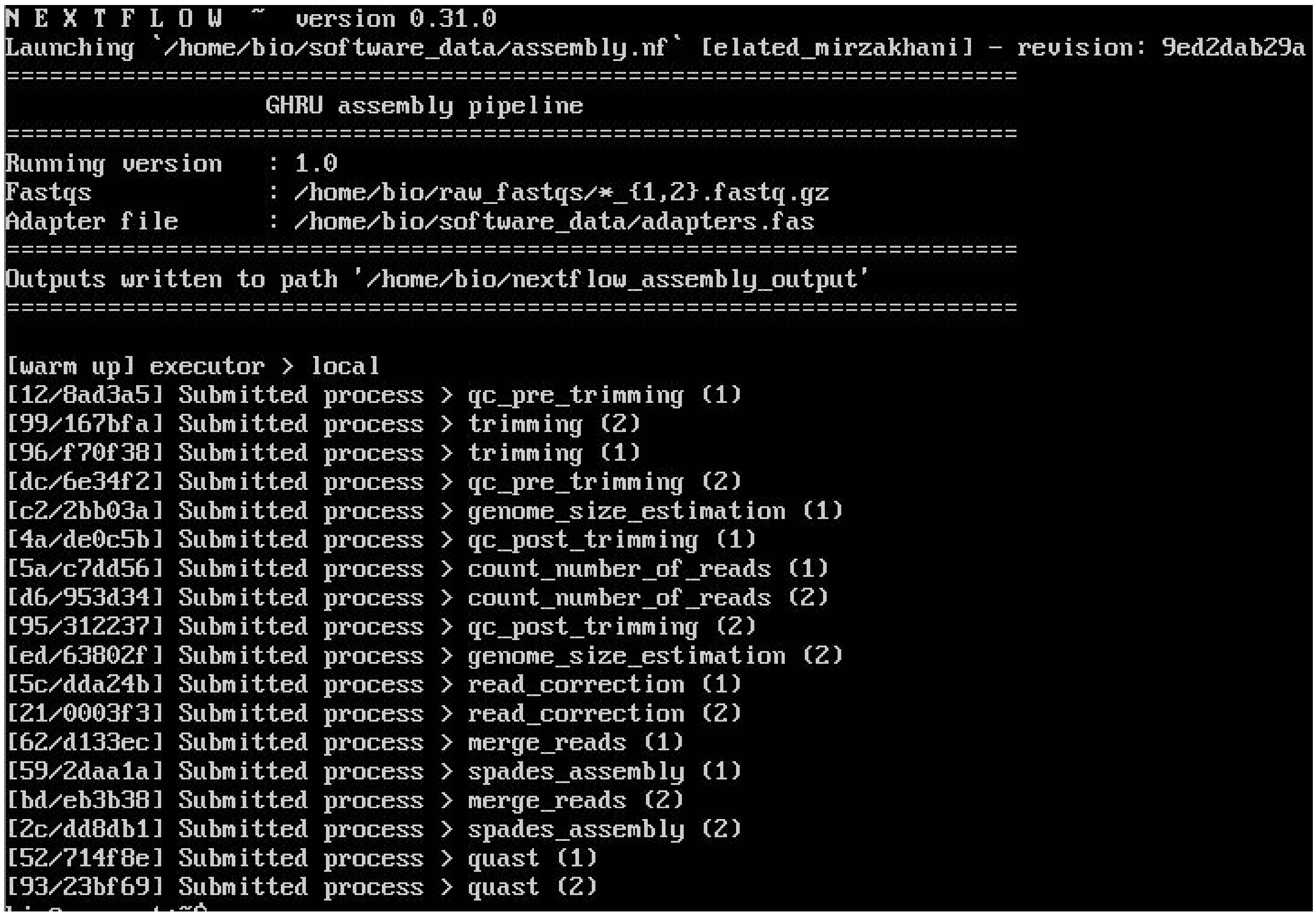
To run Nextflow you should use the following command. In this case we won’t be using Docker since all the software needed has been installed directly on the VM, so we do not need the `-with-docker` parameter

```
software/nextflow run /home/bio/software_data/assembly.nf --input_dir
/home/bio/fastq_data --fastq_pattern '*_{1,2}.fastq.gz' --adapter_file
/home/bio/software_data/adapters.fas --output_dir /home/bio/nextflow_output
```

So in this case

- The workflow file can be found `/home/bio/software_data/assembly.nf`
- The input_dir is `/home/bio/fastq_data`
- The fastq pattern is `'*_{1,2}.fastq.gz'` since the fastq files end in either `_1.fastq.gz` or `_2.fastq.gz`
- The adapter file can be found `/home/bio/software_data/adapters.fas`
- The output will be written to the directory `/home/bio/nextflow_output`

As Nextflow runs you will see each step executed in sequence and a line appear on the screen as each process is submitted for processing. The output should be something similar to the picture below:



```
N E X T F L O W  ~  version 0.31.0
Launching `/home/bio/software_data/assembly.nf` [elated_mirzakhani] - revision: 9ed2dab29a
======================================================================
                  GHRU assembly pipeline
======================================================================
Running version    : 1.0
Fastqs             : /home/bio/raw_fastqs/*_{1,2}.fastq.gz
Adapter file       : /home/bio/software_data/adapters.fas
======================================================================
Outputs written to path '/home/bio/nextflow_assembly_output'
======================================================================

[warm up] executor > local
[12/8ad3a5] Submitted process > qc_pre_trimming (1)
[99/167bfa] Submitted process > trimming (2)
[96/f70f38] Submitted process > trimming (1)
[dc/6e34f2] Submitted process > qc_pre_trimming (2)
[c2/2bb03a] Submitted process > genome_size_estimation (1)
[4a/de0c5b] Submitted process > qc_post_trimming (1)
[5a/c7dd56] Submitted process > count_number_of_reads (1)
[d6/953d34] Submitted process > count_number_of_reads (2)
[95/312237] Submitted process > qc_post_trimming (2)
[ed/63802f] Submitted process > genome_size_estimation (2)
[5c/dda24b] Submitted process > read_correction (1)
[21/0003f3] Submitted process > read_correction (2)
[62/d133ec] Submitted process > merge_reads (1)
[59/2daa1a] Submitted process > spades_assembly (1)
[bd/eb3b38] Submitted process > merge_reads (2)
[2c/dd8db1] Submitted process > spades_assembly (2)
[52/714f8e] Submitted process > quast (1)
[93/23bf69] Submitted process > quast (2)
```

When it has been completed the outputs will appear in the output directory at `/home/bio/nextflow_output`

Check which outputs you can see in this directory

## Congratulations

If you list the files in the output directory and see four directories (assembly, corrected_fastqs, qc_post_trimming and qc_pre_trimming) each containing outputs from the two samples, and two reports, one for each sample, you will have successfully performed batch *de novo* assembly using Nextflow.

## Thought Exercise

Imagine you were processing a batch of samples with the following settings

- The fastqs are in the directory `data/sequencing_run_50` in the home directory of the bio user
- The fastqs are labelled `sample_1_R1_001.fastq.gz`, `sample_1_R2_001.fastq.gz`, `sample_2_R1_001.fastq.gz`, `sample_2_R2_001.fastq.gz` etc.
- You want to write the output to a directory `assembly/sequencing_run_50` in the home directory of the bio user

What would the command to run Nextflow look like

## Version Control Table

| | | | | |
|---|---|---|---|---|
| **Title** | Genome Assembly Tutorial: Batch Processing | | | |
| **Description** | A document describing how to perform *de novo* genome assembly for multiple samples starting with a pair of fastq files for each sample. The document describes what batch processing is and how workflow managers and containers can be used to achieve this. | | | |
| **Created By** | Anthony Underwood | | | |
| **Date Created** | 14th September 2018 | | | |
| **Maintained By** | Anthony Underwood | | | |
| **Version Number** | **Modified By** | **Modifications Made** | **Date Modified** | **Status** |
| 1.0 | Anthony Underwood | First version | 14th September 2018 | First Live Version |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |