



QUANTIFYING THE DIFFERENT COVID-19 VARIANTS PRESENT IN WASTEWATER WITHIN SOUTH WALES

AUTHOR: ARLYN MILES

Supervisor: Dr Bailin Deng

Client: Professor Peter Kille

QUANTIFYING THE DIFFERENT COVID-19 VARIANTS PRESENT IN WASTEWATER WITHIN SOUTH WALES

ARLYN MILES

MAY 15, 2021

ABSTRACT

Monitoring the detection of SARS-CoV-2 and changes to its circulating strains within the population is an ongoing challenge key to controlling the global coronavirus pandemic. Wastewater analysis provides a solution to population-scale monitoring of the variants present within local communities.

The focus of this report is on the creation of a bioinformatics pipeline and accompanying CLI tool to identify and display results on the variants of the SARS-CoV-2 genome present in sequenced wastewater samples.

The result of this project a report showing the allelic frequency of each variant in a wastewater sample and graphs detailing the coverage of sequenced wastewater over a SARS-CoV-2 genome. These can be used in further research to quantify the exact strains circulating within a population and approximate the individuals infected.

ACKNOWLEDGEMENTS

I would like to thank my supervisor Dr Bailin Deng for his continued support and guidance throughout this project.

I would also like to thank my client Professor Peter Kille for taking on a mentoring role introducing me to the field of Bioinformatics and providing me with learning materials and research data.

Finally, I would like to thank Dr Daniel Pass, Toby Brann, and Elliot Gibbons from the School of Biosciences for sharing relevant research papers and offering insight on the scientific interpretation of my results.

TABLE OF CONTENTS

Abstract	2
Acknowledgements.....	3
1. Introduction	7
1.1 Preface	7
1.2 Project Aims and Scope	8
1.2 Project Limitations and Constraints.....	8
1.3 Intended Audience	8
1.5 Document Layout	9
2. Background and Literature Review	9
2.1 Research references	9
2.2 Relevance of wastewater analysis	9
2.3 Sequencing Types- Nanopore and Illumina	9
2.4 Genetics Background.....	10
3. Specification and Design.....	12
3.1 Brief	12
3.2 Deliverables adjusted since initial plan	12
3.3 Functional requirements	12
3.4 Non-functional requirements	14
3.5 Environment setup	15
3.6 Pipeline overview	16
3.7 Tools chosen	17
3.7.1 Downloading data	17
3.7.2 Trimming raw reads.....	18
3.7.3 Quality checking data	18
3.7.4 Fishing to create SAM	19
3.7.5 Assembly / reference sequence-mapping	19
3.7.6 Variant Calling	19
3.7.7 Variant annotating.....	19
4. Implementation	20
4.1 implementation overview.....	20
4.2 Command Line Interface.....	20
4.2.1 Help.....	21
4.2.2 Config.....	22
4.2.3 Run.....	22

4.2.1 Results.....	23
4.3 Trimming	25
4.4 Data Quality Assurance	26
4.5 2-process.sh	27
4.6 Isolation of covid genomes	28
4.7 Alignment and Assembly of genomes and Variant Calling	29
4.7.1 Recreating original genome	29
4.7.2 De Novo Assembly	29
4.7.3 Reference-based alignment	31
4.8 Identification of variants	32
4.8.1 Samtools variant calling	32
4.8.2 GATK Variant calling	32
4.8.3 Snippy /SNP-sites variant calling	32
4.9 Annotation of variants	33
4.9.1 SnpEff.....	33
4.9.2 IVAR	33
5. Results and Evaluation	34
5.1 Overall Results.....	34
5.2 HTML Report	34
5.2.1 Filtered SNPs/variants:	34
5.2.2 SNPs on spike protein:	35
5.2.3 Coverage summary against reference	35
5.2.4 SNP Position and Total Read Depth.....	36
5.3 Other Results	37
5.4 Requirements met.....	37
5.3 Testing	39
5.3.1 Real data	39
5.3.2 Simulated data	40
5.3 Evaluation	40
6. Future work	41
6.1 Statistical Analysis of Strains.....	41
6.2 Pipeline Improvements.....	41
7. Conclusions	41
8. Reflection.....	42
Glossary.....	44
List of Abbreviations	45

Appendices	46
Appendix 1: External Tools/Software Used	46
Appendix 2: Data used	46
References.....	47

1. INTRODUCTION

1.1 PREFACE

The global coronavirus pandemic has shaken the world, and a year on it is important to manage and combat the ever-mutating virus strains to ensure infections rates can be kept low long term, and our current countermeasures remain effective. Further monitoring will be key in evaluating the role and importance of current and future mutations (Vilar and Isom 2020).

Screening services currently utilise pharyngeal (throat and nasal) swab samples which can be costly and ultimately inaccessible for developing nations (Napit et al. 2021). Therefore, an alternative method of population-wide monitoring of viral strains would be beneficial in the ongoing management of COVID-19.

Despite being a respiratory disease, COVID-19 genomes are shed in faeces (Pan et al. 2020), which later reaches wastewater through sewage systems (Medema et al. 2020). Detection of COVID-19 genomes has been proven to be equally detectable in faecal matter as it is in throat swab specimens ((Zhang et al. 2020) informing wastewater analysis as a viable method of SARS-CoV-2 surveillance (Chavarria-Miró et al. [no date]).

Wastewater samples throughout South Wales collected by researchers at Cardiff University School of Biosciences were sequenced using Oxford Nanopore sequencing machine MinION (MinION | Oxford Nanopore Technologies. [no date]). I have created a pipeline that processes this sequenced RNA data from local wastewater, identifying and quantifying the different COVID-19 variants present in the South Wales population.

Identifying COVID-19 variants present within South Wales is important as it allows us to monitor the spread of novel strains, such as the “South African variant” 501.V2. which may increase infectivity or otherwise resist immunisation efforts.

By quantifying this, it is possible to approximate the exact number of infected people present in the city with estimates for how many individuals are associated with each variant of the virus.

1.2 PROJECT AIMS AND SCOPE

The main aims of this project are to process and analyse sequenced wastewater samples from South Wales in a way that allows for identifying all COVID-19 viral variants present in the data. These variants can then be compared to the total reads at that part of the genome to obtain an allelic frequency of each variant. This information can be used by my client in further research to derive an estimated quantification of the strains circulating within a population.

This project should be able to produce the above results from raw FASTQ sequenced wastewater samples and display the results. It is aimed at bioinformatics researchers so its configuration and usability should reflect that.

The scope of this project involves building a pipeline that can perform quality checks on the sequenced wastewater data, perform possible corrections on this data and remove adapters leftover from sequencing, separate Covid genomes from bacterial and animal DNA, align present Covid genomes to a full Covid sequence, and finally identifying the variants present in these aligned genomes.

For this project, existing bioinformatics tools that are available freely for researchers will be utilised as well as self-produced scripts.

1.2 PROJECT LIMITATIONS AND CONSTRAINTS

This project will produce allelic frequencies for the Covid variants present in wastewater samples. However, to then identify the strains circulating from these variants and approximate a specific number of individuals infected with each strain at the time the sample was taken would require a bespoke statistical framework (Hillary et al. 2021) which is beyond the scope of this project.

Existing tools for bioinformatics research will be used to perform common operations on the data within my pipeline.

1.3 INTENDED AUDIENCE

The intended audience for this report is the researchers, individuals, and investigators interested in the distribution of COVID-19 strains within the South Wales population and the methods by which these can be derived from sequenced wastewater.

As an interdisciplinary project, this report has been written so that individuals with either a Computer Science or Bioinformatics background can understand the language and methods used throughout.

1.5 DOCUMENT LAYOUT

This report is structured in the following way: Section 2 details the necessary background information on COVID-19, bioinformatics, and the project background to understand the methods and reasoning used in the implementation of my solution. Next, section 3 discusses the specifications of the project and design decisions. Section 4 then covers the implementation of the solution created for this project, followed by the results of this and an evaluation in section 5. Section 6 addresses the future work arising from this project. Section 7 concludes on the findings of the project, which is followed by a reflection on the learning process of the author in Section 8. Finally, the glossary, list of abbreviations, appendices, references, and bibliography are presented.

2. BACKGROUND AND LITERATURE REVIEW

2.1 RESEARCH REFERENCES

The background knowledge learnt for this project has been obtained from a variety of sources. My client Professor Peter Kille, Director of Technology and Bio-Initiatives explained much of the theory to me and provided me with Bioinformatics Masters course lecture materials. A large number of research papers on Covid-19 and bioinformatics educated me on the background of this subject. The rest of my sources which can be found in the Bibliography were software manuals and online websites.

2.2 RELEVANCE OF WASTEWATER ANALYSIS

Testing for Covid-19 is mostly done on an individual basis, through pharyngeal or saliva samples. Testing done in this manner is costly so is not easily accessible globally. Monitoring of Covid-19 through metagenomic (environmental) samples such as wastewater allows an overview of the infection levels of local communities.

Covid-19 is detectable in faecal matter and ultimately leads to detection in wastewater treatment plants. Research is currently ongoing in Cardiff University's School of Bioscience to analyse strains present within local wastewater samples, mimicking research across the world trying to achieve similar analysis into population-wide monitoring.

Monitoring of wastewater can enable a sentinel surveillance of SARS-CoV-2 (Chavarria-Miró et al. [no date]) which can lend to an early warning system of outbreaks within local communities. By comparing variant analysis of wastewater across a region, the movement of strains can be detected. Tracking variants of concern is of utmost importance in our continued fight against Covid-19 to ensure low transmission rates and ensure our treatments remain effective.

2.3 SEQUENCING TYPES- NANOPORE AND ILLUMINA

There are many methods for sequencing biological sequences, from low-throughput Sanger sequencing to NGS (Next Generation Sequencing) methods Illumina and

Nanopore. The scope of this project involves creating a pipeline that can process both Illumina and Nanopore generated sequences.

The aim of sequencing is to take a biological sample and produce digital sequences (often in FASTQ format) that represent each nucleotide base of the DNA/RNA present. The FASTQ format is a text file of nucleotide bases e.g. (ACGGCTA) with a quality score stating the sequencing confidence encoded in ASCII for each base. An alternative data format that may be seen is FASTA format. This is a text file of nucleotide bases without any quality scores.

Illumina sequencing is a popular NGS technology that involves large sequencing platforms and can perform sequencing with a high throughput. It typically produces longer reads, meaning each sequence has a greater length (Sequencing | Key methods and uses. [no date]). Nanopore is a more recent technology that is also able to sequence with a high throughput. The Oxford Nanopore MinION is much more portable than an Illumina platform and can be held in your hand. Nanopore sequenced data produces shorter reads by measuring the electrical current of genetic material passing through channel proteins (How it works. [no date]).

Illumina sequenced data produces paired-end reads, one forwards across the DNA strands, and one backwards. This produces two different sequence files to process downstream. Data sequenced by Nanopore produces single-end reads in just one file. This difference between them requires processing Nanopore and Illumina data differently and software that works with both single and paired-end data must be used.

As part of the sequencing process, chemically synthesized sequences called adapters are mixed in with the original genetic sample to accurately sequence the DNA/RNA. However, these adapters pollute the resulting sequenced data so must be removed, also known as trimming. These adapters vary depending on if the data was sequenced with Nanopore or Illumina, therefore it is necessary to find trimming software that can remove this for both types of data.

2.4 GENETICS BACKGROUND

DNA and RNA are the building blocks of organic life and contain the genetic code for any organism. A genome refers to the complete set of genetic information that codes for a particular organism. For example, the SARS-CoV-2 genome is the full set of RNA that Covid-19 is composed of.

Within individuals of any organism, genetic variation occurs. The genome of any two organisms of the same species is seldom identical. The slight differences in the sequences of genetic information are called variants. Variants represent a change in the original nucleotide base. One type of variant is a single nucleotide polymorphism (SNP). This is the most common form of genetic diversity and represents a single base swap i.e., from a "C" nucleotide base to a "T" base (Variant Calling part 2 (Galaxy) - Bioinformatics Documentation. [no date]). Insertions and deletions (indels) are two other types of variants that can occur, where extra bases are inserted into the code or bases usually present are deleted.

Not all variants will cause a noticeable change in function for that organism. Some variants will code for the same protein regardless, while some variants will cause a

different amino acid to be produced which can result in a protein that changes the function of that part of the organism. This is called a mutation and is especially important to monitor in viruses as their high reproduction rate allows for greater genetic diversity and therefore fast mutation rates.

For Covid-19, it is important to monitor the SARS-CoV-2 variants circulating within the population to keep an eye on any 'variants of concern'. These variants of concern tend to be changes to the genome that result in changes that increase transmission of the virus or how it could be treated.

For coronavirus, the most important location on the genome to monitor these variants is on the spike protein. The spike protein allows the virus to enter host cells, so is directly related to the transmissibility of the virus (Vilar and Isom 2020). Mutations on the spike protein such as on the South African strain are worrying as circulations of this strain can increase the spread of the virus (Trevor Bedford on Twitter: "Since their recognition in the UK, South Africa and Brazil in Dec 2020 and Jan 2021, the variant of concern lineages B.1.1.7, B.1.351 and P.1 have continued to spread throughout the world with B.1.1.7 so far the most successful of the three. 1/15 <https://t.co/RtVLQPRUiV>" / Twitter. [no date]).

If the number of variants present in a given sample is compared to the number of reads (sequenced data) at the same position on the genome, a ratio of variant to non-variant can be calculated to derive allelic frequencies for each variant. An allele is a version of a nucleotide base, so allelic frequencies represent how often each variant occurs in a sample. Retrieving this data from wastewater samples is the primary aim of this project.

3. SPECIFICATION AND DESIGN

3.1 BRIEF

This project was requested by my client Professor Peter Kille, Director of Technology and Bio-Initiatives from the Cardiff University School of Biosciences. The desired outcome was a bioinformatics pipeline that can derive allelic frequencies (the frequency of a variant at a particular gene) across the SARS-CoV-2 (Covid-19) genome from sequenced wastewater samples. These allelic frequencies represent the mutations occurring in the strains circulating within local communities the wastewater is collected from.

The expected input for this system is raw FASTA format sequenced wastewater. The expected output is a list of SARS-CoV-2 variants present in the wastewater, and thus an indication of the variants circulating within the population of the area at the time the wastewater sample was taken.

The allelic frequencies identified from the sequenced wastewater can then be used by my client in further research to perform statistical analysis on the exact strains circulating within local communities.

3.2 DELIVERABLES ADJUSTED SINCE INITIAL PLAN

The plan for this project has changed since the initial planning stage. At the time of the initial plan, my client Professor Peter Kille had explained the overall results desired from this project, however, the exact design and implementation required was not clear at this point. The main reason for this was the large amount of research necessary into existing solutions, bioinformatics procedures, and the scientific background before a solution could be designed.

The use of Docker to containerise the project was no longer necessary due to being able to install all necessary modules onto the Trinity cluster.

No detail was included in the initial plan as to the steps the pipeline would involve or how it would be created. This was due to the lack of knowledge of a typical bioinformatics workflow and was designed later in the project after extensive research.

Originally, I had planned to develop a first iteration of the pipeline, and then test and evaluate it before developing a second iteration. Instead, the development of each step of the pipeline had multiple iterations which I evaluated, and the final pipeline connected each optimised step.

From initial conversations with my client, potential secondary aims for measuring Covid-19 diversity over time within an area and comparing variants with samples from the Cardiff University Screening Service were discussed. However, it was later decided this was beyond the scope of this project.

3.3 FUNCTIONAL REQUIREMENTS

Based on conversations with my client Professor Peter Kille about the data I would be working on and the analysis results desired from it, I have defined key functional requirements for the pipeline. These were agreed on by my client and align with the overall project aims.

3.3.1 Requirement 1

Title: Quality check data and trim raw reads to remove adapters

Description: The raw FASTA wastewater sequences have certain content leftover from the sequencing process called “adapters” that must be removed before the data can be processed. The data must also be quality checked to ensure the read quality is high enough for the downstream analysis to produce results.

Acceptance criteria:

- Each raw read file must be trimmed to remove any adapter content or bad quality reads.
- The resulting trimmed read files must be quality checked and only acceptable data analysed downstream.
- Each quality check will produce a report that can be manually examined.

3.3.2 Requirement 2

Title: Align sequences to Covid-19 reference

Description: Trimmed sequences that pass the quality checks should be aligned against a reference FASTA file for the first sequenced Wuhan Covid-19 genome and sorted into the right order, producing a sorted SAM/BAM (Sequence Alignment Map format and Binary Alignment Map format) file.

Acceptance Criteria:

- Trimmed sequences are aligned using a reference sequence to create a SAM file.
- SAM file converted into BAM file for further analysis.
- BAM file is then sorted into the correct order for the reference genome producing a sorted BAM file.

3.3.3 Requirement 3

Title: Identify variants that differ from the reference

Description: Areas in the mapped sequence that do not match with the reference genome should be identified, these areas can represent mutations in the sequence. These can be SNPs (single nucleotide polymorphisms, a single swap of a nucleotide e.g. A to T), or insertions/deletions, but also may be sequencing errors. The SNPs identified must be filtered to eliminate false positives.

Acceptance criteria:

- Variants are identified from each sorted BAM file and this list is exported.
- SNPs are annotated using a GenBank file to describe which mutations encode for certain proteins and estimate the relevance of SNPs.
- Only variants that pass certain tests for eliminating false positives are outputted.

3.3.4 Requirement 4

Title: Display relevant results and statistics as a report

Description: Many files and information are generated throughout the pipeline. The most relevant results from the analysis should be displayed to the user in a HTML report for ease of viewing.

Acceptance criteria:

- The pipeline should store analysis results in TSV (tab-separated values) files.
- Data from these files should be read, and relevant tables and graphs created.
- A HTML report file should be generated for each sequence to display the tables and graphs, and resultant variants identified from the wastewater.

3.4 NON-FUNCTIONAL REQUIREMENTS

My client did not request any specific design requirements outside of the overall aims of the project. Therefore, the requirements for my project were mostly self-defined based on the result my client desired.

3.4.1 Requirement 1

Title: User interface intuitive and in line with similar tools

Description: Target demographic for this tool is bioinformatics researchers with high technical experience, so the interface should feel familiar compared to similar tools used in bioinformatics.

Acceptance criteria:

- Command-line interface designed with help commands/prompts to explain the usage of the tool.
- Interface created in-line with similar tools, such as FASTP. E.g. reports generated in HTML for easy viewing, command-line interface with help messages

3.4.2 Requirement 2

Title: Ability to reproduce results with different datasets

Description: Sequenced data can come in many forms depending on the lab the data originated from and the machine the samples were sequenced with. The pipeline should be adaptable to a variety of input data.

Acceptance criteria:

- The pipeline can produce results with data sequenced by ILLUMINA machines.
- The pipeline can produce results with data sequenced by NANOPORE machines.
- The pipeline can produce results with data originating from non-metagenomic samples e.g., saliva sample from an individual.

3.4.3 Requirement 3

Title: Maximised full potential of computational power available

Description: The environment I have access to has a lot of computational power available, which is important to fully utilise as the data I will be working with is large and the computations performed can be complex.

Acceptance criteria:

- Jobs are scheduled onto the Slurm queue and not run on the head node.
- Multithreading is used when possible.
- Software that can be run on multiple CPUs concurrently will do so.

3.4.4 Requirement 4

Title: Computationally efficient, to minimise drain on resources.

Description: The pipeline should be optimised to be computationally efficient to minimise the time the analysis takes when large and numerous datasets are used as an input

Acceptance criteria:

- Pipeline code runs efficiently in an order that does not repeat steps unnecessarily or require excess waiting for previous steps to finish.
- Existing bioinformatics software used is chosen with efficiency in mind and multiple tools have been tested before deciding on the most efficient.

3.5 ENVIRONMENT SETUP

For this project I had access to the Trinity cluster, a large-memory Slurm cluster for the School of Biosciences available for staff and post-graduate students (HPC Services - School of Biosciences. [no date]). This can only be accessed while on the Cardiff University Virtual Private Network (VPN). The main access point to this cluster is the iago.bios.cf.ac.uk head node which I access by SSH connection from my terminal. This is a command-line interfaced Linux environment. On the iago head node I have access to data storage and can schedule jobs.

There are three storage areas I have access to:

- /mnt/data/GROUP-[groupCode] (quota'd group storage - keep all important data here)
- /mnt/scratch (non-quota'd ephemeral storage)
- \$HOME (quota'd storage, used only for Linux and application configuration files)

There are three queues on the Trinity cluster:

- Defq- For lower resource jobs, up to 64 CPUs and 128GB RAM
- Jumbo- Development jobs

- Mammoth- For large jobs with high RAM requirements

All programs must run as jobs scheduled onto one of these three queues using the Slurm scheduler. Parameters for the job are set at the top of a bash script and define the queue and resources to use.

Most of my scripting was done using bash scripts written using nano while on the Trinity cluster. A Python environment was set up on both iago and my local PC. Required Python modules for the CLI tool I created can be installed using a requirements.txt file I generated. Python scripts were first developed on my local Windows PC and then transferred to iago using WinSCP (WinSCP :: Official Site :: Free SFTP and FTP client for Windows. [no date])

The pipeline created requires several modules which had been pre-installed on the Trinity cluster and can be accessed by using [module load] in a bash script. Please see Appendix 1 for a full list of software packages and libraries used.

3.6 PIPELINE OVERVIEW

My project is composed of Python scripts for the user interface and report generation, and bash scripts for processing the data and running existing bioinformatics tools.

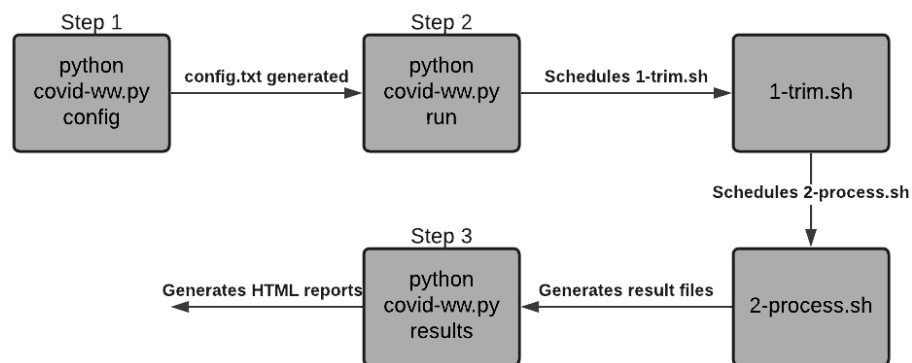


FIGURE 1: PIPELINE OVERVIEW DIAGRAM

Each step in this diagram represents a command the user of the pipeline must run in the terminal. Each stage (grey box) represents sections of code that is run. The arrows in this diagram represent the result of the previous stage and which stage occurs next.

The pipeline is run through the covid-ww.py file. This is a CLI tool I have created that allows you to set up the configuration of the pipeline, run the pipeline, and then generate a report for viewing the results. Running [python covid-ww.py] will display the available commands you can run.

Firstly, [python covid-ww.py config] is run. This will walk you through the setup of the config file interactively and is necessary to prepare the pipeline for the data you will be using.

Secondly, [python covid-ww.py run] is run. This will schedule the pipeline bash script 1-trim.sh onto the Slurm queue to start trimming and performing quality checks on the data.

After 1-trim.sh has finished running, 2-process.sh will automatically be scheduled onto the Slurm queue. This script finishes the processing of the trimmed data obtained in 1-trim.sh and generates all results such as identifying variants.

Finally, after 2-process.sh has finished running, you will optionally be able to run the command [python covid-ww.py results]. This Python script analyses the resulting files from 2-process.sh, creates tables and graphs and generates a HTML report for each input sample for viewing the results of the pipeline easily.

3.7 TOOLS CHOSEN

There are many existing bioinformatics tools available freely for research. In choosing the tools most suitable for manipulating my data and producing desirable results I have tested out many tools that perform similar functions. Decisions on which tools to use were made with a focus primarily on functional relevance, speed of performance, and utility.

For each stage of performing operations on my data I have provided a list of software explored and justifications for tool decisions made. Links to external software used can be found in Appendix 1.

3.7.1 DOWNLOADING DATA

For testing the pipeline, it was important to have access to a variety of data. Individual Covid-19 positive saliva samples, sequenced wastewater, data sequenced by Oxford Nanopore and Illumina etc. An easy way to transfer this data to my scratch data space on the Trinity cluster was needed.

Methods considered:

- Create a text file of all URLs of sequences to be downloaded. Text file must be manually created. Bash script will iterate through each URL and download it using **wget**.
- Use **curl** in the terminal to manually download sequences one by one using the URL found on NCBI, Covid-19 Data Portal, GISAID.
- Using **fastqdump** from **SRAtoolkit** (ncbi/sra-tools: SRA Tools. [no date]) to download data from a list/text file of accession IDs (e.g. ERR4971212) with **sem** from **GNU Parallel** (GNU Parallel - GNU Project - Free Software Foundation. [no date]) to parallelise the downloads across multiple tasks to decrease time taken.

Decision:

For downloading large datasets, I combined using **curl** with **GNU Parallel**. **GNU Parallel** allowed me to download multiple files at the same time using multithreading. Using **curl** was useful as it displayed the time remaining for the download and was also quicker than **SRAToolkit**'s **fastqdump** which took much longer to download the same datasets and from investigating the source code was found to be inefficient. This combination meets functional requirement 4 (*Title: Display relevant results and statistics as a report*) as it is the most computationally efficient solution. For one-off sequence downloads I used **curl** directly.

3.7.2 TRIMMING RAW READS

The data my pipeline takes as an input has adapter content leftover from the sequencing process that must be removed and may also contain poor quality data that must be removed. The process of removing these is called 'trimming'.

Methods considered:

- **Fastp** (Chen et al. 2018) to trim data. Works with both single and paired end data, so compatible with Illumina and Nanopore sequenced data. Automatically produces a HTML report for each trimmed file.
- **Trimmomatic** (USADLLAB.org - Trimmomatic: A flexible read trimming tool for Illumina NGS data. [no date]). Also allows for single and paired end data. Does not generate a report.

Decision:

Fastp was found to be 2-5 times faster than **Trimmomatic** and generated a report. Since both tools worked with Illumina and Nanopore sequenced data, I chose **Fastp** to perform the trimming of my data in the pipeline. This met the non-functional requirement 2 (*Title: Ability to reproduce results with different datasets*), as it enabled my pipeline to reproduce results with different datasets. It also meets non-functional requirement 4 (*Title: Computationally efficient, to minimise drain on resources*) as it is the most computationally efficient solution.

3.7.3 QUALITY CHECKING DATA

The data the pipeline receives as input must be checked to ensure it is of high enough quality for the downstream analysis to produce accurate results. If the data is not of sufficient quality, it should not be processed further.

Methods considered:

- **FastQC** (Babraham Bioinformatics - FastQC A Quality Control tool for High Throughput Sequence Data. [no date]) quality checks the data and produces a html report.
- **MultiQC** (Ewels et al. 2016) summarises the output of existing quality check reports and generates an overall report.

Decision:

- I decided to use both **FastQC** and **MultiQC** to produce a fuller picture of the quality of the data across multiple samples. This meets my functional requirement 4 as I can display relevant results and statistics as a report.

3.7.4 FISHING TO CREATE SAM

The sequenced wastewater data contains DNA and RNA from many other organisms such as bacteria and other viruses. It is necessary to extract the Covid-19 RNA from this large sample and line up the extracted RNA to a template Covid-19 genome. This will produce a Sequence Alignment Map (SAM) file.

Decision:

- BMAP was chosen as Macpacbio is forgiving for allowing mismatches in alignment, which is important as wastewater data tends to have low coverage of the target genome.

3.7.5 ASSEMBLY / REFERENCE SEQUENCE-MAPPING

Methods considered:

- De Novo Assembly with **SPAdes**
- **MetaSPAdes** for metagenomic data
- Mapping/Aligning sequence to a reference genome with **samtools**.

Decision:

Reference alignment using a reference sequence with **samtools** was computationally the fastest solution addressing non-functional requirement 4 (*Title: Computationally efficient, to minimise drain on resources*).

3.7.6 VARIANT CALLING

Methods considered:

- **GATK's HaplotypeCaller**. This did not work well with wastewater sequenced data; it did not produce suitable results so was not included in the final iteration of the pipeline.
- Samtools **mpileup** produced variant files quickly and integrated well with other tools and operations.
- Snippy pipeline created utilising tools **snippy** and **snp-sites** to perform variant calling and generation of a phylogenetic tree.

Decision:

Samtools mpileup was sufficient for the identification of variants, Snippy pipeline was unnecessary as generating the phylogenetic tree between samples would not be relevant for wastewater data containing many variants from multiple strains.

3.7.7 VARIANT ANNOTATING

Methods considered:

- SnpEff- No Covid-19 database existed for this tool, upon building a database from GenBank files the tool did not produce accurate results.
- Ivar- Annotates the variants using the GenBank feature format file downloaded from NCBI. Produced results as a detailed TSV file.

Decision:

Ivar was chosen as the only tool that produced successful annotations of the Covid-19 variants identified.

4. IMPLEMENTATION

4.1 IMPLEMENTATION OVERVIEW

The implementation created for this project is a bioinformatics pipeline composed of bash scripts and freely available external tools (listed in appendix 1). The user interface for the configuration, running, and viewing results of this pipeline is handled by a Python command-line interface. There are four main script files, however, this represents only a small fraction of the code created for determining the final steps of the pipeline. The final four scripts:

1. covid-ww.py
This is the command-line interface for the program. All interaction with the pipeline should be done by running the commands in this script.
2. report.py
This file reads the output files generated by the running of the pipeline and generates a HTML report with graphs and tables of the data for each sample.
3. 1-trim.sh
This is the first step of the pipeline which trims the inputted data and generates a report on the result of the removal of adapter and poor-quality content.
4. 2-process.sh
This is the next step of the pipeline which aligns the 'trimmed' reads with a reference Covid-19 genome to create a sequence alignment map (SAM) file which is used to identify variants and provide an analysis of the wastewater data.

4.2 COMMAND LINE INTERFACE

User interaction with the pipeline is done entirely with the covid-ww.py Python command line interface (CLI). A command line interface was chosen to meet non-functional requirement 1 (*Title: User interface intuitive and in line with similar tools*). Many bioinformatics tools operate with a command line interface, so this was developed in line and with inspiration from existing tools such as FastQC, fastp,

MultiQC, and Samtools.

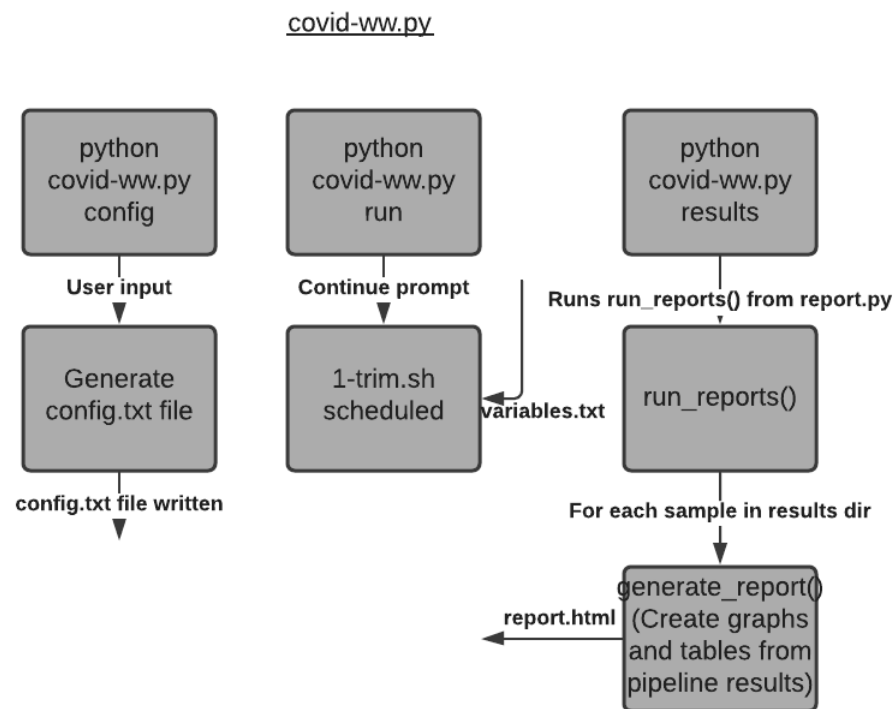


FIGURE 2: COMMAND LINE INTERFACE

The command line interface also meets functional requirement 4 (*Title: Display relevant results and statistics as a report*) through the results command, which creates tables and graphs from the data outputted from the pipeline and generates a HTML report to display them.

The CLI was created using the Python package Click(Welcome to Click — Click Documentation (8.0.x). [no date]) as it provided the benefit of automatically generating help pages with familiar appearances to existing bioinformatics tools.

There are four commands available: help, config, run, results.

4.2.1 HELP

This is an automatically generated help command run using [`python covid-ww.py --help`]. This provides a list of the other three commands that can be run using this tool. This message will also be displayed if running the CLI without a command.

```
C:\Users\Snore\Desktop\Diss\Filter>python covid-ww.py --help
Usage: covid-ww.py [OPTIONS] COMMAND [ARGS]...

  CLI for Covid Variant Wastewater pipeline

Options:
  --help  Show this message and exit.

Commands:
  config
  results
  run
```

FIGURE 3: HELP COMMAND

4.2.2 CONFIG

This command is run using [python covid-ww.py config]. The purpose of this command is to interactively create a config.txt file needed to run the pipeline. It will prompt the user to set up the correct directories, name the project, specify the sequencing type and the naming conventions of input data.

```

This will walk you through the config of the pipeline
Enter the full path of your working directory. A new directory 'covid-ww' will
/trinity/home/cl803625/scratch/
What's the name of your project? This will be the directory name created.
Wastewater
What is the full path of the directory containing your raw reads?
/trinity/home/cl803625/scratch/downloaded/illumina
Are you using Illumina or Nanopore data?
Illumina
What do your forwards read files end in? (Example: Manchester_1.fastq.gz ends
with '_1.fastq.gz')
_1.fastq.gz
What do your backwards read files end in? (Example: Manchester_2.fastq.gz end
s with '_2.fastq.gz')
_2.fastq.gz
Generation of config.txt done!
You may now run/queue the pipeline with [covid-ww.py run]
(base) [cl803625@iago covid-ww]$

```

FIGURE 4: CONFIG COMMAND

User input will be saved and sanitised to ensure spacing mistakes will not interfere with the running of the pipeline. After running this command, a config.txt will be written to the same directory as the pipeline scripts which will be used in the pipeline.

4.2.3 RUN

This command is run using `[python covid-ww.py run]` and is responsible for running the pipeline. It is necessary to have set up the `config.txt` file first, so the user will be prompted for whether they have run the config command prior. If yes, the first pipeline script `1-trim.sh` will be scheduled using **sbatch** onto the Slurm queue.

1-trim.sh must be scheduled using **sbatch** to prevent running the pipeline on the head node and utilise the computational power available on the Trinity cluster. The first script is scheduled onto the 'jumbo' queue as it is a significant sized job that requires a larger amount of RAM than the 'defq' but is not taxing enough for the 'mammoth' queue. After the first job is finished, it will automatically schedule the next job 2-process.sh afterwards.

4.2.1 RESULTS

This command is run using [python covid-ww.py run]. It takes in the results from running the pipeline from the previous command and creates tables and graphs from this data. A HTML report is then generated for each sample the pipeline was run with which displays a list of filtered variants, highlights relevant mutations on the spike protein, visualises the coverage depth per variant and throughout the genome.

Upon running the run command through covid-ww.py, the **run_reports()** function is called from reports.py.

```
def run_reports():
    #Get directory
    with open('config.txt') as f:
        results_dir = f.readline()
        #strip
        results_dir = results_dir.replace('#', '')
        results_dir = results_dir.replace('\n', '')
    #For each data points
    for file in os.listdir(results_dir):
        if file.endswith("_variants.tsv"):
            file = file.replace("_variants.tsv", "")
            generate_report(results_dir, file)
```

FIGURE 5: RUN_REPORTS()

First, the directory in which the pipeline results files are stored is retrieved from the first line of the config.txt file. Each file in this directory is then iterated over to find the name of each sample inputted. For each sample results were generated for, the **generate_report()** function is run, to create a separate report for every sample.

For every sample the **generate_report()** function runs for, it will retrieve data from TSV (tab separated values) files and store them in DataFrames using the Python library Pandas (pandas - Python Data Analysis Library. [no date]). Three files are stored this way- a list of the variants, a summary of the general coverage of the covid genome, and the depth of coverage per position on the genome.

```
#for each data
def generate_report(results_dir, name):

    variant_file = results_dir + "/" + name + "_variants.tsv"
    data = pd.read_csv (variant_file, sep = '\t')
    #Only export data which eliminated false positives
    filtered = data.loc[data['PASS'] == True]
    #Select columns from table
    export = filtered[['POS', 'ALT', 'REF_DP', "TOTAL_DP", 'PVAL']]
```

FIGURE 6: GENERATE_REPORT() – IMPORTING DATA

The Python library Plotly is then used for the creation of interactive graphs from the data (Plotly Python Graphing Library | Python | Plotly. [no date]). The DataFrames are adapted into tables with the most relevant columns selected, and graphs with correct axes labelled. These are then converted to HTML code and classes replaced with Bootstrap styled classes.

```
spike = export.loc[(export['POS'] >= spike_start) & (export['POS'] <= spike_end)]

snp_graph = go.Figure(go.Scatter(x=export.POS, y=export.TOTAL_DP, text=export.ALT, mode='markers')

snp_graph.update_xaxes(title_text='Position in genome')
snp_graph.update_yaxes(title_text='Total depth')
snp_graph = snp_graph.to_html(full_html=False, include_plotlyjs='cdn')

snp_table = export\
    .to_html()\
    .replace('<table border="1" class="dataframe">', '<table class="table table-striped">') # use
```

FIGURE 7: GENERATE_REPORT() - CREATING GRAPH

After all graphs and tables have been created and converted into HTML code, a HTML string combines HTML code and the generated figures and is written to a HTML file for the sample. This process is repeated for every data sample the pipeline has run for.

```
html_string = '''
<html>
  <head>
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css">
    <style>body{ margin:0 100; background:whitesmoke}</style>
  </head>
  <body>
    <h1>Filtered SNPs</h1>

    <h3>Filtered SNPs</h3>
    ''' + snp_table + '''
    <h3>SNPs on Spike protein</h3>
    ''' + snp_spike_table + '''
    <h3>Coverage</h3>
    ''' + coverage_table + '''
    <h3>SNP Position and Total Read Depth</h3>
    ''' + snp_graph + '''
    <h3>Coverage Graph</h3>
    ''' + coverage_graph + '''
  </body>
</html>'''
filename = results_dir + "/" + name + "_report.html"
f = open(filename, 'w')
f.write(html_string)
```

FIGURE 8: GENERATE_REPORT() - WRITING TO HTML

4.3 TRIMMING

The input data for the pipeline is raw sequenced wastewater data. This data has not been priorly treated to examine the quality or remove artefacts. As part of the sequencing process, chemically synthesized sequences called adapters are mixed in with the original genetic sample to accurately sequence the DNA/RNA. However, these adapters pollute the resulting sequenced data so must be removed, also known as trimming. Adapter trimming is a prerequisite step for analysing sequenced data. Additionally, the raw sequenced data has not been checked for any areas that are too poor quality to process further in the downstream analysis. Poor quality underneath a certain threshold should also be removed during this trimming stage.

During the development of the pipeline, I investigated two tools (listed in appendix 1) to perform the trimming of the adapters and poor-quality data. The first tool I used was **Trimmomatic**.

```
module load fastqc/v0.11.7
module load Trimmomatic/0.39

srafile=(ERR4971212)
softwaredir="/mnt/data/GROUP-smbpk/PROJECTS/shared_scripts/sratool

for i in {0..1}; do
  trimmomatic PE ${srafile[i]}_1.fastq.gz ${srafile[i]}_2.fastq.gz
  fastqc -t 2 ${srafile[i]}_1.fastq.gz ${srafile[i]}_2.fastq.gz
done;
```

FIGURE 9: OLD_TRIM.SH USING TRIMMOMATIC

I first experimenting by trimming Covid-19 positive saliva Illumina sequenced samples downloaded from the NCBI (The National Center for Biotechnology Information) database. I found this software to be effective for paired end Illumina data using the PE parameter as well as with single end Nanopore data.

However, after investigating another tool called **fastp**, I switched to **fastp** after discovering it was 2-5 times faster than Trimmomatic (Chen et al. 2018) to adhere to my non-functional requirement 4 (*Title: Computationally efficient, to minimise drain on resources*).

It was time consuming to manually list the samples to trim in a list, so I created a script to iterate through every file in a given directory, retrieve the base name without the file extension, and then trim each file and rename the output trimmed files accordingly.

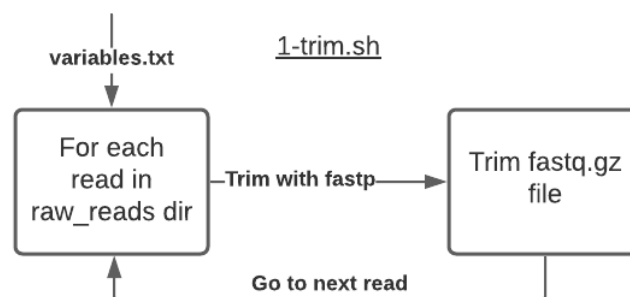


FIGURE 10: 1-TRIM.SH DIAGRAM

The final trimming script receives a source of variables from the variables.txt file which specifies the desired output directory names. The variables.txt file will further source variables from the config.txt file generated by the covid-ww.py CLI tool including the working directory and directory of the input data.

```
module load fastp/v0.20

source variables.txt
#Make dirs if not existing
mkdir -p $dir
mkdir "${dir}/${name}"
echo "${dir}/${name}"
mkdir -p $report_dir
mkdir -p $trim_dir
mkdir -p $trimming_report_dir
#should be no need to make raw_dir because otherwise there would be no input data
software_dir="/mnt/data/GROUP-smbpk/PROJECTS/shared_scripts/sratoolkit.2.9.0-cento

echo $all_raw

if [ $sequencing_type == 'ILLUMINA' ]
then
  for f in $all_raw
  do
    filepath="${f%.fastq.gz}" #Get filepath without _1.fastq.gz or _2.fastq.gz
    base="$(basename $filepath)" #Get just base name like: ERR407592
    echo "Trimming $base"
    fastp -i "${filepath}_1.fastq.gz" -I "${filepath}_2.fastq.gz" -o "${trim_dir}/${base}_1_trim.fastq.gz"
    echo "Output to: ${trim_dir}/${base}_1_trim.fastq.gz"
  done
else
  for f in $all_raw
  do
    filepath="${f%.fastq.gz}"
    base="$(basename $filepath)" #Get just base name like: ERR407592
    echo "Trimming $base"
    fastp -i "${filepath}.fastq.gz" -o "${trim_dir}/${base}_trim.fastq.gz" -w ${SL
    echo "Output to: ${trim_dir}/${base}_1_trim.fastq.gz"
  done
fi
echo "Done trimming files"
```

FIGURE 11: FINAL TRIMMING SCRIPT 2-TRIM.SH USING FASTP

Depending on the machine used to sequence the input fastq data, the fastp command was run with different parameters. For Illumina sequenced data, fastp was ran in paired end mode taking in a forward read and backward read (often labelled appended with _1 and _2 respectively) as an input. For Nanopore sequenced data only one read is taken as an input.

The output of this script is a directory of trimmed fastq files labelled with “_trim” appended to the original sample name. This directory will be in the location specified in the config.txt and variables.txt configuration files.

4.4 DATA QUALITY ASSURANCE

The data inputted into the pipeline comes in FASTQ format. FASTQ is a text format used for storing biological sequences, in this case sequenced DNA and RNA extracted from wastewater. It stores both the nucleotide bases e.g. (ACCGTGA) and a quality score for how confident the sequencing is for each base encoding with an

```
(base) [c1803625@iag_subset]$ head Liverpool_1.fastq
@MISEQ:59:000000000-J33NG:1:1101:14696:1401:1:N:0:TATCACTCTCAGAACAAGGT
GGTGTACTCTGCTATTGTACTTACGTTCGTTTGTACTACAGCATAAACAGATAAATTCGGTTAAGTGGTGGTCTAG
AGTTTAAATGTCTCCTCAGTAGCTTTGAGCGTTTTCTGCTGCAAAAAGCTTGAGTCTTTCAGTACAGGTGTTAGCTAAAATGTA
+
>>>>BF3@CFCCFFLEGGGGFGHHEDAGHDGCEGCBBABA3EAFAFDCEAB1F2AGCFGFFGADFA/FFEDAB21AA/ED1E1A
HB1>GHG1>BGFDDGGGHHFHFB2GHH2FFC?EC<FD?GCF<F<.0<=0HHGGGFFFFGHFGDGC0<<AGB:G0;CGCG:0
@MISEQ:59:000000000-J33NG:1:1101:16998:1427:1:N:0:TATCACTCTCAGAACAAGGT
CACAACTTCGGGTGGAGGTTAATGTGTCTACTGTGTGAACACCTTAATAGTCTCACTTCTCTCAAAGAAAGAAGTGTCF
TCACCTCTCTTAAGAAATCTTATACCTAGTTGTGTAGATTGTCCAGAATAGGACCAATCTTTATAGGAACCAGCAAGTGAGAT
+
BBBBBBBFFFFFFBAGGGGGGGFGGHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHGH5FGHH
HHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHHGH5FGHH
@MISEQ:59:000000000-J33NG:1:1101:16531:1432:1:N:0:TATCACTCTCAGAACAAGGT
CCACAGTACGTCTACAAGCTGTAATGAACAAGAAGTGCCTGCCAATTCAACTGTATTATCTTTCTGTGCGCTTTGCTGTAGAT
ACACACTGGTACTGGTCAGGCAATAACAGTTACACCGGAAGCCAATATGGATCAAGAATCCTTTGGTGGTGCATCGTGTGTC
```

Quality checking the FASTQ files involves analysing the quality scores of the data and producing a report on the overall quality of each file. This is important as poorly sequenced data will not produce a useful downstream analysis.

```
module load fastqc/v0.11.9
module load multiqc/1.9
source variables.txt

mkdir "${qc_report_dir}"
mkdir "${multiqc_report_dir}"

for f in $all_raw
do
    filepath="${f%_1.fastq.gz}" #Get filepath without _1.fastq.gz or _2.fastq.gz
    base="$(basename $filepath)" #Get just base name like: ERR407592
    echo "Generating 4 reports for $base"
    fastqc -t 4 "${filepath}_1.fastq.gz" "${filepath}_2.fastq.gz" "${trim_dir}/${base}"
    echo "Output to: ${qc_report_dir}"
done

#Generate report for all the trimmed data
multiqc ${qc_report_dir} -o ${multiqc_report_dir}
echo "Output to: ${multiqc_report_dir}"
```

If a sample is inputted to the pipeline with a subpar quality score, it is wise to disregard some or all of the results and cease further downstream analysis.

4.5 2-PROCESS.SH

The rest of the pipeline steps detailed after this included in the final iteration of the pipeline are run from the 2-process.sh file.

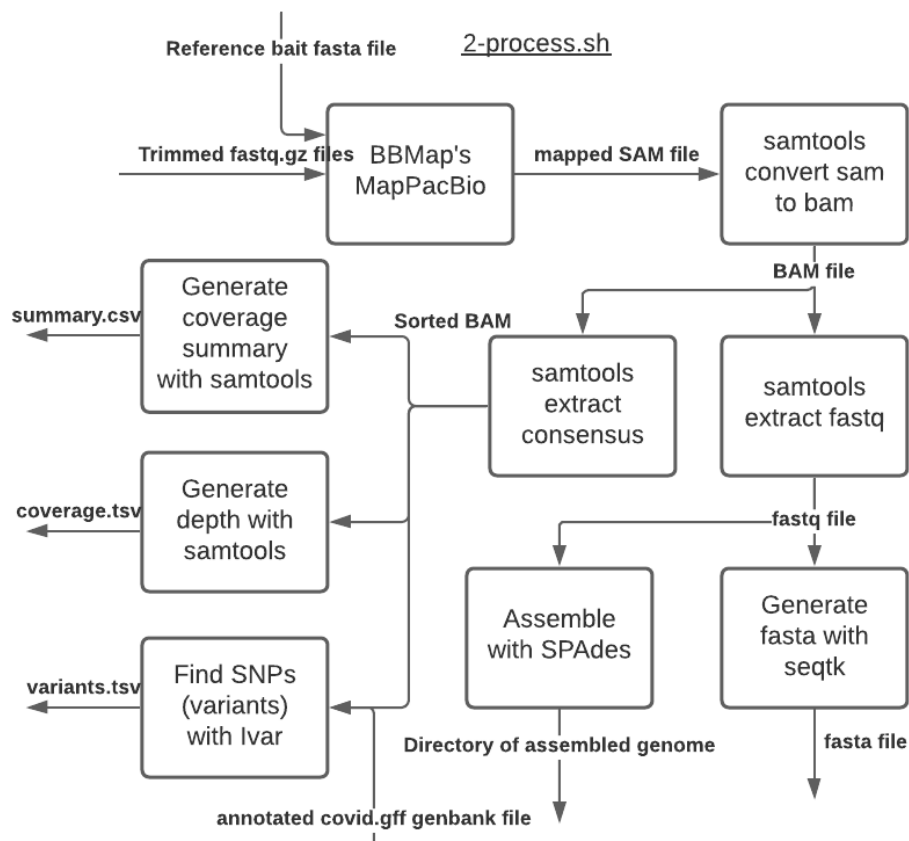


FIGURE 14: 2-PROCESS.SH DIAGRAM

Here each rectangle represents a process being done to the data, with the arrows representing the output of each process and the next process that occurs with the outputted data.

4.6 ISOLATION OF COVID GENOMES

The nature of wastewater samples results in these samples containing genetic information from many different organisms. As this project looks only to identify variants from SARS-CoV-2, the genetic information originating from the virus must be extracted from the rest of the sequence, commonly referred to as 'fishing'.

To extract the data belonging to the Covid-19 genome, a reference also known as bait file must be used as a template to align the reads to. Sequences that map to a known Covid genome can be identified as belonging to the Covid genome.

To map the trimmed FASTQ sequences to a Covid reference, the tool **BBMap** (see appendix 1) was used. It is capable of aligning reads sequenced with both Illumina and Nanopore machines. For each set of paired or single ended trimmed reads in the trimming directory, **BBMap** will align the sequence to a specified 'bait' reference file. 16 threads were used simultaneously to meet non-functional requirement 3 (Title: Maximised full potential of computational power available).

The reference file used in the pipeline was the first sequenced Covid sample Wuhan-Hu-1 NCBI reference NC_045512.2 downloaded from NCBI (Appendix 2).

```
mkdir -p "$fish_output_dir"

for f in $fish_input_dir
do
  if [ $sequencing_type == 'ILLUMINA' ]
  then
    filepath="${f%_1_trim.fastq.gz}"
    base=$(basename $filepath)
    echo "Fishing in ${base}"
    /mnt/data/GROUP-smbpk/PROJECTS/shared_scripts/bbmap/mapPacBio.sh \
    ref=$bait in="${filepath}_1_trim.fastq.gz" \
    in2="${filepath}_2_trim.fastq.gz" slow=f \
    outm=$fish_output_dir/${base}_${fish_prepend}.sam nodisk threads=16
  fi
done
```

A SAM (Sequence alignment map) file is created for each sample, containing only sequences belonging to SARS-CoV-2. This SAM file is then processed further into a BAM (Binary alignment map) file, the encoded version of a SAM file. This BAM file can be processed faster than a SAM file, so meets non-functional requirement 4 (*Title: Computationally efficient, to minimise drain on resources*).

Samtools is used to convert the SAM file to a BAM file and extract a FASTQ file. This file contains the extracted 'fished' out Covid-19 sequences only.

```
#convert to bam
samtools view -bS $fish_output_dir/${base}_${fish_prepend}.sam > \
$fish_output_dir/${base}_${fish_prepend}.bam

#extract fastq reads and pairs
samtools bam2fq $fish_output_dir/${base}_${fish_prepend}.bam > \
$fish_output_dir/${base}_${fish_prepend}.fq
```

FIGURE 15: PROCESS SAM FILE TO OTHER FORMATS

4.7 ALIGNMENT AND ASSEMBLY OF GENOMES AND VARIANT CALLING

4.7.1 RECREATING ORIGINAL GENOME

The sequenced wastewater data is unsorted, and it is not clear which sequences correspond to each part of the genome. There are two main methods for recreating the original genome(s): De novo sequence assembly, and reference-based mapping/alignment.

During the development of the pipeline, I created scripts to carry out both methods using numerous tools. I compared these methods to discover the ideal solution for the final pipeline.

4.7.2 DE NOVO ASSEMBLY

In De novo sequence assembly, you assume no prior knowledge of the original genome(s) in the sample that you are attempting to assemble. Instead, the aim is to recreate the original genome sequence through joining together overlapping sequenced reads.

The first script created to experiment with the assembly of the Covid genome was **SPAdes** (appendix 1).

```

module load SPAdes/3.14.1
#call variables
source variables.txt

mkdir -p "$fish_output_dir"

for f in $fish_input_dir
    spades.py \
        -t 16 \
        -m 32 \
        --12 $fish_output_dir/${base}_${fish_prepend}.fq \
        -o "$fish_output_dir/${base}_${fish_prepend}"

```

FIGURE 16: ASSEMBLING GENOME WITH SPAdES

This utilises the FASTQ file containing the already extracted Covid-19 sequences, and then attempts to assemble them into a genome blindly. The output of this script will be a directory per assembled sequence that contains all the output files. Two important output files are the contigs and scaffolds FASTA format file. FASTA format represents the nucleotide bases of the sequences like FASTQ except without the confidence quality scores for each base.

```

(base) [c1803625@iago Manchester_covid]$ head contigs.fasta
>NODE_1_length_1538_cov_1190.261517
ATGCACATCAGTAGTCTTACTCTCAGTTTTGCAACAACCTCAGAGTAGAATCATCATCTAA
ATTGTGGGCTCAATGTGTCCAGTTACACAATGACATTCTCTTAGCTAAAGATACTACTGA
AGCCTTTGAAAAAATGGTTTCACTACTTTCTGTTTTGCTTTCCATGCAGGGTGCTGTAGA
CATAACAAGCTTTGTGAAGAAATGCTGGACAACAGGGCAACCTTACAAGCTATAGCCTC

```

FIGURE 17: CONTIGS.FASTA

Contigs.fasta stores every contig from the original sequence. A contig is a large fragment of subsequent DNA from a sequence. The tool **SPAdes** is used to join these overlapping contigs together with the goal of completing the original genome. Scaffolds.fasta contains the resulting scaffolds sequence, composed of the contigs joined together into an assembled sequence.

There are two techniques used by SPAdes for joining contigs into scaffolds. First by examining read pairs and estimating the size of the gap separating contigs. Secondly by examining the assembly graph: e.g., if two contigs are separated by a complex tandem repeat, that cannot be resolved exactly, contigs are joined into the scaffold with a fixed gap size of 100 nucleotide base pairs (Bankevich et al. 2012).

SPAdes assembly successfully met the project aim for assembling an original genome, however, was deemed insufficient for the final iteration of the pipeline. It is completed using a greedy algorithm, which is a resource intensive process and takes a long time.

In addition to this, typical assemblers do not produce good results with metagenomic samples (genetic material obtained from the environment, e.g., wastewater) due to the lower read count and coverage across the genome than standard samples from individuals such as pharyngeal swab samples.

Specific metagenomic assemblers were investigated for completing the assembly of the Covid-19 genome. Software dedicated to assembly metagenomic samples accounts for low read count and lower coverage. The tool **metaSPAdes** (Nurk et al. 2017) was tested and considered for the pipeline, however ultimately the better decision to meet non-functional requirement 4 (*Title: Computationally efficient, to minimise drain on resources*) was to implement reference-based alignment for retrieving the genome instead.

4.7.3 REFERENCE-BASED ALIGNMENT

An alternative method of recreating the original genome from a mixed and unsorted set of sequences is mapping/aligning the sequences to a previously assembled reference genome, in this case the first sequenced SARS-CoV-2 Wuhan strain in FASTA format.

Every read from the sequence is aligned against the reference sequence and placed at the most likely position in the genome. The reference file must first be indexed to create a lookup table using **samtools** enabling faster comparisons between the input sequence and reference file. One complication to this method is variants naturally occur throughout a genome so the sequence will never line up perfectly with the reference file.

```
#make index for reference
samtools faidx $bait

#extract consensus
samtools sort -o $fish_output_dir/${base}_${fish_prepend}_sort.bam \
-T sort_temp -@ 16 $fish_output_dir/${base}_${fish_prepend}.bam
samtools index $fish_output_dir/${base}_${fish_prepend}_sort.bam

samtools mpileup -C50 -uf $bait $fish_output_dir/${base}_${fish_prepend}
| bcftools call -c | vcfutils.pl vcf2fq | gzip > \
$fish_output_dir/${base}_${fish_prepend}.fq.gz
```

FIGURE 18: SORT BAM FILE AND EXTRACT CONSENSUS

A consensus sequence is the calculated order of the most frequent nucleotide bases found at each location in the sequence alignment. This is extracted from the BAM file and sorted into the correct order for the Covid reference genome.

Samtools mpileup (See appendix 1) command is used to create a TSV (tab separated values) file containing all the sequenced reads at a single genomic position. The wastewater contains many sets of incomplete Covid genomes, so the identical parts of the genome are piled up onto the same position. This is a prerequisite step to calling variants present.

Reference-based mapping was implemented in the final pipeline as it produces more fitting results for the aim of identifying variants, as well as being more computationally efficient than De Novo assembly therefore meeting non-functional requirement 4 (*Title: Computationally efficient, to minimise drain on resources*).

4.8 IDENTIFICATION OF VARIANTS

The primary aim of this project is to identify the SARS-CoV-2 variants present in wastewater samples. Variants are naturally occurring modifications in the genetic code for an organism (genome). If a nucleotide base at position 3007 of the genome is usually a “T” but was instead found to be a “C” base, this new “C” base is said to be a variant. Variants come in many forms and sometimes may or may not have a resulting effect on the resulting protein and therefore function. This is important for tracking viral mutations that may affect the transmissivity of Covid-19 and for identifying viral strains.

4.8.1 SAMTOOLS VARIANT CALLING

The process of identifying variants from an aligned sequence is called variant calling. This requires a sequence mapped to the genome through either assembly or sequence-mapping described in the prior step. A sorted BAM file by the tool **Samtools** is taken as input with the reads lined up to each position in the Covid-19 genome. This is used by **BCftools** (see Appendix 1) to call the variants by identifying the differences between reads in the same position on the genome.

```
samtools mpileup -C50 -uf $bait $fish_output_dir/${base}_${fish_prepend}
| bcftools call -c | vcfutils.pl vcf2fq | gzip > \
$fish_output_dir/${base}_${fish_prepend}.fq.gz
```

FIGURE 19: VARIANT CALLING WITH SAMTOOLS AND BCFTOOLS

4.8.2 GATK VARIANT CALLING

Another software investigated for calling variants was GATK. This was not compatible with the sorted BAM files produced with the wastewater data so was not included in the final pipeline.

```
GNU nano 2.9.8 gatk.sh Modified
module load gatk-4.1.0.0-gcc-8.3.1-2fiyw2h
for f in $input
do
  filepath="${f%*_covid_sort.bam}" #Get filepath without _1.fastq.gz or _2.fastq$
  base="$(basename $filepath)" #Get just base name like: ERR407592
  echo "Fixing $f"
  new_name="${base}_covid_sort_fixed.bam"
  gatk --java-options "-Xmx4g" AddOrReplaceReadGroups -I "$f" -O \
  "${inpath}/${new_name}" --RGLB lib1 --RGPL illumina --RGPU unit1 --RGSM 20
  echo "New file created as ${inpath}/${new_name}"
  gatk --java-options "-Xmx4g" HaplotypeCaller -R \
  "${inpath}/NC_045512_DNA.fasta" -I "${inpath}/${new_name}" -O \
  "${outpath}/${base}.vcf"
  echo "Output to: ${outpath}/${base}.vcf"
done
```

FIGURE 20: VARIANT CALLING WITH GATK

4.8.3 SNIPPY /SNP-SITES VARIANT CALLING

I also created a pipeline for calling variants using the tools **Snippy** and **snp-sites**.

The first step of the script finds sections of data that align with the reference file and ‘snips’ them using Snippy. A core genome alignment is then generated using the Covid-19 Wuhan reference from NCBI. This then generates a variant alignment for each sample, which is used to generate a phylogenetic tree using **FastTreeMP** (see

Appendix 1) for a visual representation of how closely related each virus sample is to each other.

Phylogenetic trees are redundant for wastewater samples as no single strain or set of variants is present in the data. There are many samples from many individuals within the wastewater sequences so any phylogenetic trees drawn would be taking an average of every strain present.

```
GNU nano 2.9.8          snippy_pipeline.sh          Modified
mkdir "${output}" #Creates output folder if not already existing

#Run snippy on every forward and backwards reads in input folder
for f in $input
do
    filepath="${f%_1_trim.fastq.gz}"
    base=$(basename $filepath) #get the name of the sequence, e.g. SRR13297091
    #Get filepath for forward and backward read, e.g. SRR13297091_1_trim.fastq.gz
    file1="${trimming}/${base}${forward_read_extension}"
    file2="${trimming}/${base}${backward_read_extension}"
    echo "Will snip ${file1} and ${file2} using ${reference}"
    snippy --cpus ${SLURM_CPUS_PER_TASK} --ref "${reference}" --outdir \
    "${output}/${snip_pre}${base}" --R1 "${file1}" --R2 "${file2}" --force
done

#Generate core genome alignment from mapping runs
snippy-core --ref "${reference}" --prefix "${output}/output-alignment" \
${output}/${snip_pre}*
#Generate variant alignment
snp-sites -cb -o "${output}/snp-alignment.aln" "${output}/output-alignment.aln"

#Draw phylogenetic tree
FastTreeMP -nt -gtr < "${output}/snp-alignment.aln" > "${output}/tree.tre"
```

FIGURE 21: VARIANT CALLING WITH SNIPPY

4.9 ANNOTATION OF VARIANTS

To make the identified variant information more useful, it is possible to annotate these variants with the resulting proteins they encode for and predict the resulting function a mutation might have.

4.9.1 SnpEff

SnpEff ran from a Java file making it a slower solution than Ivar. There was no pre-existing database for the SARS-CoV-2 genome annotations, so I built this using the GenBank annotation files located on the NCBI website. Despite building a database for the annotations, the tool did not produce successful annotations on my data.

4.9.2 IVAR

Ivar takes the piped result of the variants called using **samtools mpileup** and annotates them using the covid.gff (feature format file of the annotated Covid-19 proteins) downloaded from GenBank. This produces a TSV file that is later read in the report.py script and generates a HTML table.

```
tsv_output="${fish_output_dir}/${base}"
echo "${tsv_output}"
samtools mpileup -aa -A -d 0 -B -Q 0 --reference \
$ bait $fish_output_dir/${base}_${fish_prepend}_sort.bam | \
ivar variants -p $tsv_output -r $bait -g covid.gff
done
```

FIGURE 22: IVAR ANNOTATING VARIANTS

Ivar is used in the final iteration of the pipeline in 2-process.sh to identify and label the variants within wastewater and suggest possible effects the variants have on the resultant mutations.

5. RESULTS AND EVALUATION

5.1 OVERALL RESULTS

The project was successful in achieving its primary aim in identifying and displaying the variants present in the wastewater. The pipeline successfully processed the input data of varying formats and produced reports displaying the relevant information.

5.2 HTML REPORT

After running the pipeline, you can generate a HTML report of relevant tables and graphs using the CLI tool. Each sample the pipeline is run for will generate a report which can be opened in the browser offline. This is the primary method for viewing the results and includes interactive graphs. The following results were obtained from running the pipeline on Illumina sequenced UK wastewater data (See appendix 2).

5.2.1 FILTERED SNPs/VARIANTS:

Filtered SNPs							
Filtered SNPs							
	POS	REF	ALT	ALT_FREQ	GFF_FEATURE	TOTAL_DP	PVAL
2	490	T	C	0.358232	cds-YP_009724389.1	656	4.604250e-127
3	490	T	C	0.358232	cds-YP_009725295.1	656	4.604250e-127
4	556	C	T	0.215596	cds-YP_009724389.1	654	8.870000e-73
5	556	C	T	0.215596	cds-YP_009725295.1	654	8.870000e-73
6	1271	T	C	0.350282	cds-YP_009724389.1	531	5.661160e-66

FIGURE 23: FILTERED VARIANTS

This table represents all the variants present in the wastewater sample that pass all checks for eliminating false positives. The “POS” column states the position in the genome the variant occurs in. “REF” and “ALT” state the original base that is expected, and the variant base(s) that replace it, respectively. “ALT_FREQ” is the allele frequency of the variant proportional to non-variant alleles found in the sample sequence. 0.1 would represent 1 variant allele found at this location for every 9 non-variant alleles. This is the most important result as it completes the overarching project aim for quantifying the variants from wastewater samples.

The “GFF_FEATURE” column is an annotation of which part of the Covid-19 genome this base position encodes for. “PVAL” is a quality score indicating the confidence of the variant calling.

For a fuller picture of these results with more columns, you may view the [sample_name]_variants.tsv file located in the /covid-ww/[project_name]/results folder.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	REGION	POS	REF	ALT	REF_DP	REF_RV	REF_QUAL	ALT_DP	ALT_RV	ALT_QUAL	ALT_FREQ	TOTAL_DF	PVAL	PASS	GFF_FEAT	REF_CODC	REF_AA	ALT_CODC	ALT_AA
2	NC_04551	320	C	T	0	0	0	1	0	33	1	1	1	FALSE	cds-YP_00	CCT	P	TCT	S
3	NC_04551	320	C	T	0	0	0	1	0	33	1	1	1	FALSE	cds-YP_00	CCT	P	TCT	S
4	NC_04551	490	T	C	421	2	73	235	0	73	0.358232	656	#####	TRUE	cds-YP_00	GAT	D	GAC	D
5	NC_04551	490	T	C	421	2	73	235	0	73	0.358232	656	#####	TRUE	cds-YP_00	GAT	D	GAC	D
6	NC_04551	556	C	T	513	2	74	141	0	74	0.215596	654	8.87E-73	TRUE	cds-YP_00	TAC	Y	TAT	Y
7	NC_04551	556	C	T	513	2	74	141	0	74	0.215596	654	8.87E-73	TRUE	cds-YP_00	TAC	Y	TAT	Y
8	NC_04551	1271	T	C	345	345	38	186	183	39	0.350282	531	5.66E-66	TRUE	cds-YP_00	TTT	F	CTT	L
9	NC_04551	1271	T	C	345	345	38	186	183	39	0.350282	531	5.66E-66	TRUE	cds-YP_00	TTT	F	CTT	L
10	NC_04551	1573	A	T	0	0	0	1	0	32	0.333333	3	0.333333	FALSE	cds-YP_00	ACA	T	ACT	T
11	NC_04551	1573	A	T	0	0	0	1	0	32	0.333333	3	0.333333	FALSE	cds-YP_00	ACA	T	ACT	T

FIGURE 24: IVar OUTPUT TSV ANNOTATED VARIANTS

The meanings of each column can be found on the Ivar manual (iVar: Manual. [no date]).

5.2.2 SNPs ON SPIKE PROTEIN:

The second table is a list of variants that specifically occur on the spike protein of the SARS-CoV-2 genome. Mutations that occur on the spike protein can be responsible for increased transmissivity of the virus, so variants that occur within this location are of special interest for monitoring.

SNPs on Spike protein							
	POS	REF	ALT	ALT_FREQ	REF_AA	ALT_AA	PVAL
327	23549	G	A	0.474000	D	N	3.044940e-88
328	23706	C	-CA	0.471173	NaN	NaN	2.877020e-54
329	23731	C	T	1.000000	T	T	4.633930e-301
348	24729	C	-A	0.252319	NaN	NaN	3.519940e-31
349	25333	T	-GA	0.262850	NaN	NaN	1.592130e-139

FIGURE 25: VARIANTS LOCATED ON THE SPIKE PROTEIN

REF_AA states the original amino acid that is encoded for by the reference base at this position. ALT_AA states the alternate amino acid that is encoded for by the variant base at this position. If these are different, it means the variant encodes for a different protein that may have a different function i.e., increase transmissivity.

5.2.3 COVERAGE SUMMARY AGAINST REFERENCE

Coverage								
#name	startpos	endpos	numreads	covbases	coverage	meandepth	meanbaseq	meanmapq
0 NC_045512	1	29903	139617	23491	78.5573	1161.02	38.1	45.6

FIGURE 26: COVERAGE STATISTICS FOR SAMPLE AGAINST REFERENCE

This table provides information about the general coverage of the sample across the SARS-CoV-2 genome. In this example, 78% of the genome is covered, so 22% of the genome positions do not have reads that align with those positions. The Covid genome is roughly 29,000 bases long dictated by the “startpos” and “endpos”. “Numreads” represents the total number of bases in the aligned sequence.

5.2.4 SNP POSITION AND TOTAL READ DEPTH

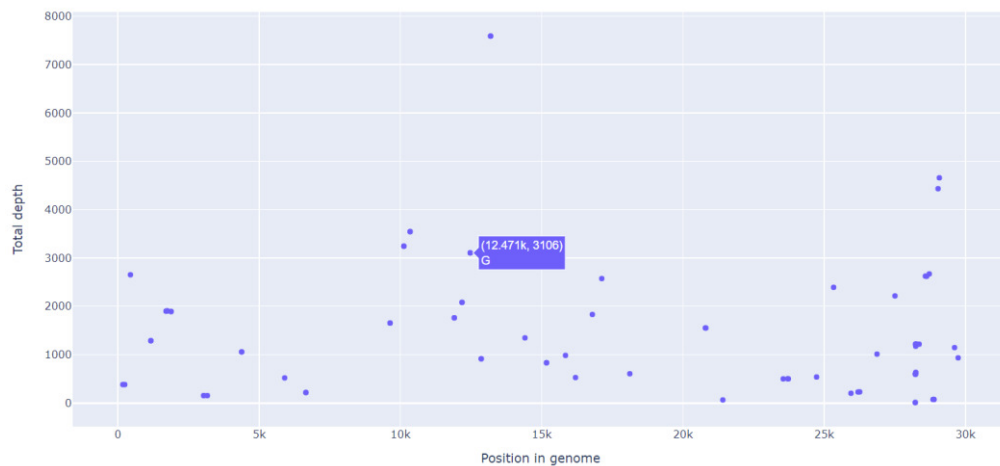


FIGURE 27: READ DEPTH FOR EACH VARIANT

This graph displays each variant against the other reads at the same position in the genome. From this graph, you can interpret which variants are more common. Variants present in the genome with a higher total read depth may indicate higher confidence of this variant being prevalent in the community the wastewater was sampled from.

Coverage graph:

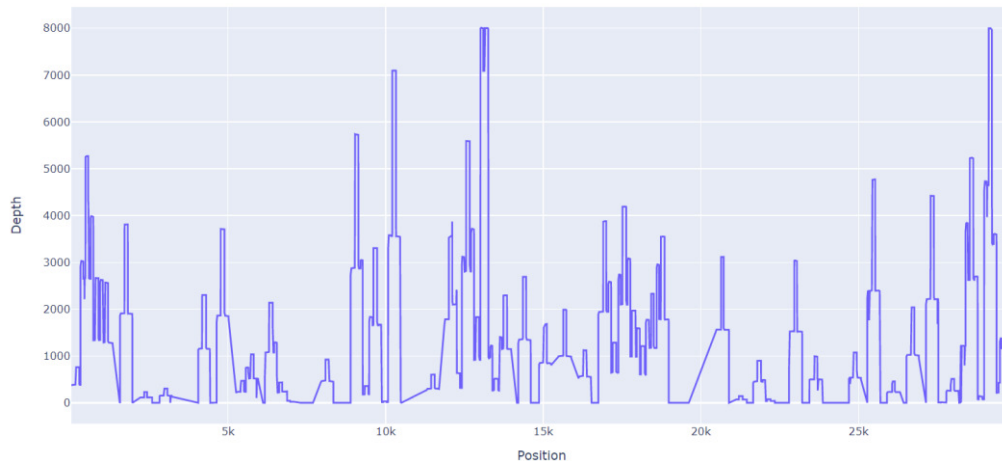


FIGURE 28: COVERAGE GRAPH FOR EACH POSITION IN THE COVID GENOME

This coverage graph represents which parts of the Covid-19 genome have been sequenced the most. Coverage is never even as wastewater contains many contaminants that may degrade the RNA and dilute the samples, and the sequencing process may amplify certain sections of RNA more than others. From this graph, it is visible that there is low coverage across the location for the spike protein, between 22k and 25k bases. This may be the reason only 1 variant present on the spike protein has been identified from the wastewater. The quality of the data across that section of the genome may not have been high enough to accurately identify variants due to low coverage.

5.3 OTHER RESULTS

Only key results are included in the HTML report, there are many other files generated by the pipeline that can be examined for a more thorough understanding of the results.

On creation of the project through the CLI tool, a directory is created with the specified project name. All files created are stored inside subdirectories in this folder, default configuration structures them as following:

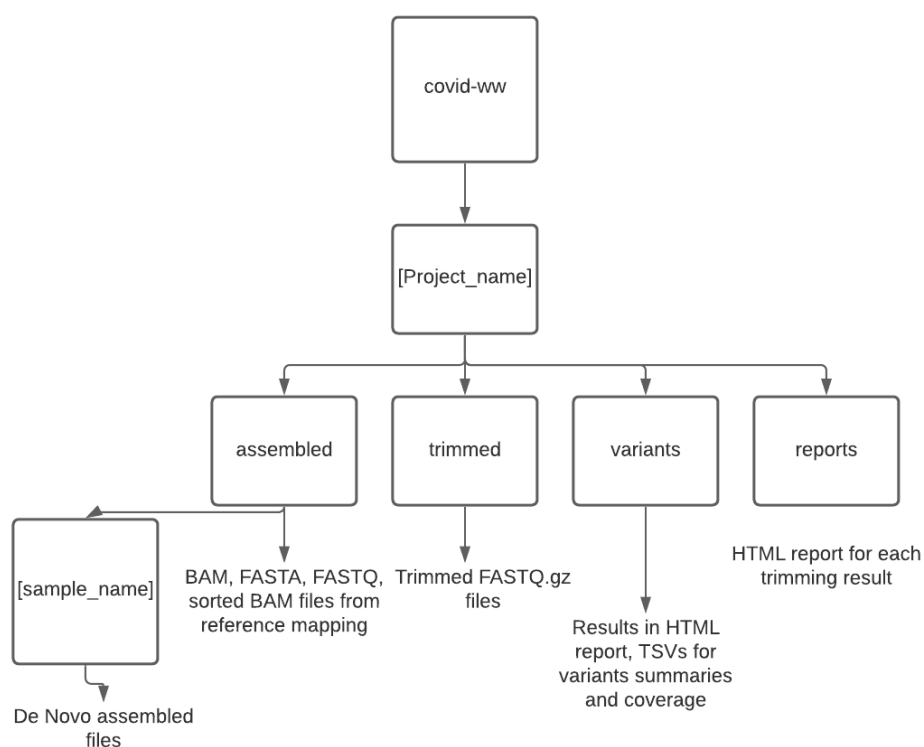


FIGURE 29- OUTPUT DIRECTORY STRUCTURE

Each rectangle represents a directory, arrows indicate hierarchy, and text outside a shape represents files present inside a directory.

5.4 REQUIREMENTS MET

For a full description of each requirement please refer back to section 3.3.3 and 3.3.4. All functional and non-functional requirements have been met.

5.4.1 Functional Requirements

1	Title: Quality check data and trim raw reads to remove adapters
Acceptance criteria:	
<ul style="list-style-type: none">Each raw read file must be trimmed to remove any adapter content or bad quality reads.Resulting trimmed read files must be quality checked and only acceptable data analysed downstream.Each quality check will produce a report that can be manually examined.	

PASS- Data is trimmed during the running of the pipeline and outputted to /trimmed directory in FASTQ.gz format. Reports are created and outputted to /reports in HTML format.

2 | Title: Align sequences to Covid-19 reference

Acceptance Criteria:

- Trimmed sequences are aligned using a reference sequence to create a SAM file.
- SAM file converted into BAM file for further analysis.
- BAM file is then sorted into the correct order for the reference genome producing a sorted BAM file.

PASS- Sequences are aligned using samtools and SAM, BAM, and sorted BAM files are outputted to /assembled directory.

3 | Title: Identify variants that differ from reference

Acceptance criteria:

- Variants are identified from each sorted BAM file and this list is exported.
- SNPs are annotated using a GenBank file to describe which mutations encode for certain proteins and estimate relevance of SNPs.
- Only variants that pass certain tests for eliminating false positives are outputted.

PASS- Variants are called using samtools and BCFtools. These variants are then annotated using iVar and exported as a TSV to the /variants directory. Variants exported have a column for if they pass the quality checks to remove false positives or not, only those that pass are displayed in the final reports generated.

4 | Title: Display relevant results and statistics as a report

Acceptance criteria:

- Pipeline should store analysis results in TSV (tab separated values) files.
- Data from these files should be read, and relevant tables and graphs created.
- A HTML report file should be generated for each sequence to display the tables and graphs, and resultant variants identified from the wastewater.

PASS- Coverage, variants, and summary information is saved as TSV files in the /variants directory. The reports.py script reads these files, creates tables and graphs, then generates a HTML report for each sample.

5.4.2 Non-functional requirements

1 | Title: User interface intuitive and in line with similar tools

Acceptance criteria:

- Command line interface designed with help commands/prompts to explain the usage of the tool.
- Interface created in-line with similar tools, such as FASTP. E.g. reports generated in HTML for easy viewing, command line interface with help messages

PASS- All pipeline configuration, running, and displaying results can be executed through the CLI tool. The interface is created using the Python click library so is

standardised and inspiration has been taken from tools such as FASTP and MultiQC.

2	Title: Ability to reproduce results with different datasets
Acceptance criteria: <ul style="list-style-type: none"> • Pipeline can produce results with data sequenced by ILLUMINA machines. • Pipeline can produce results with data sequenced by NANOPORE machines. • Pipeline can produce results with data originating from non-metagenomic samples e.g., saliva sample from an individual. 	
PASS- Pipeline has been tested successfully with ILLUMINA and NANOPORE sequenced data and produced results. Pipeline also tested and obtains results with non-metagenomic samples.	

3	Title: Maximised full potential of computational power available
Acceptance criteria: <ul style="list-style-type: none"> • Jobs are scheduled onto the Slurm queue and not run on the head node. • Multithreading is used when possible. • Software that can be ran on multiple CPUs concurrently will do so. 	
PASS- Software has been specifically chosen which allows for multiple CPUs to be utilised at once and all jobs take advantage of the computational power of the Trinity cluster by being schedule on the jumbo and mammoth queues.	

4	Title: Computationally efficient, to minimise drain on resources.
Acceptance criteria: <ul style="list-style-type: none"> • Pipeline code runs efficiently in an order that does not repeat steps unnecessarily or require excess waiting for previous steps to finish. • Existing bioinformatics software used is chosen with efficiency in mind and multiple tools have been tested before deciding on the most efficient. 	
PASS- Multiple iterations of each stage of the pipeline were tested before finalising each stage to ensure the software and scripts created were the most efficient solution. The pipeline code does not hang waiting for previous steps.	

5.3 TESTING

For testing the accuracy of my results, I chose two main methods- using real data obtained from the European Nucleotide Archive (ENA) and using simulated wastewater data.

5.3.1 REAL DATA

There is a wealth of online databases providing free access to raw Covid-19 sequences in FASTQ format. Throughout the project, data from NCBI, EBI, and GISAID (three open-access databases of genomic data) has been tested through various stages of the pipeline. Data sequenced by both Illumina and Nanopore sequencing machines has been tested by the pipeline, with successful results produced for both.

For testing the correct calling of variants, I chose a set of wastewater FASTQ files from EBI (See Appendix 2) and compared the results I obtained from my pipeline with the results of the research paper examining those samples found (Hillary et al. 2021). By comparing my results, I was able to determine my pipeline was successful in accurately calling the SNPs (variants) from those wastewater samples.

5.3.2 SIMULATED DATA

There are several existing tools for generating simulated biological sequences, however few for generating the wide mix of genomes present in wastewater.

The first software I considered was **ARTillumina**, a commonly used tool for generating simulated reads recommended by my client. However, this was not ideal for generating data that mimicked metagenomic samples like wastewater. The first metagenomic read simulation software I considered was **MetaSim** (Richter et al. 2008). After reading its paper this seemed like a promising solution, however, the link to access the software was no longer available.

I chose InSilicoSeq (Gourlé et al. 2019) to perform the simulation of wastewater data as it was able to combine samples from multiple genomes as well as generate variations of one set genome. I installed this tool in my local environment and downloaded bacteria and virus genomes to mix into the generated wastewater sample. I created a multi FASTA file with 10 bacteria genomes, 4 virus genomes, and 1 SARS-CoV-2 genome, then created a coverage.txt file listing the desired coverage of each genome in my resultant simulated wastewater data.

After attempting to use my pipeline on this simulated data, false-positive SNPs were generated. I continually adjusted the variant calling filter until these would not show up using simulated data but would show up for real data.

5.3 EVALUATION

Overall, the primary aim of the project to identify and quantify the variants present in wastewater samples was successfully met. All functional and non-functional requirements were passed, however, some original aims decided in the initial report had since been reconsidered due to a revaluation of the scope of the project (See Section 3.2 for deliverables adjusted from the initial plan).

I have a moderate degree of confidence in my results as these have been verified against real data and simulated data. To improve my confidence in my results I would like to produce simulated data for specific mixes of Covid-19 strains to test the exact proportions of variants identified by my pipeline.

The usability of the pipeline is high due to the CLI tool created that can configure, run, and display results. But this is only usable in the working environment of the Trinity Cluster currently and would be difficult to port to other environments without creating a Docker image. With more time I would have liked to have published the pipeline as a fully contained Docker image with installation instructions.

The pipeline performs full analysis on 5 samples in under 15 minutes on the jumbo queue of the Trinity cluster, which is a good speed for the amount of computation

required by the software used. The tools I have developed would struggle to run on a local PC environment due to a reduced amount of RAM and CPUs.

My client was pleased with my presented results and finished pipeline, and it met his main expectation of quantifying the variants present in the wastewater. Further aims to quantify the individuals infected with given strains was not met, as this was decided after the initial plan to be outside the scope of the project. The possibility of quantitative strain identification is discussed further in section 6. Future work.

6. FUTURE WORK

The analysis of SARS-CoV-2 in wastewater is an ongoing project with research carried out by the Cardiff University School of Biosciences and other researchers around the world. I currently plan to continue my work with my client, Professor Peter Kille, over the summer and provide insight from a computer science perspective for the ongoing tracking of strains within South Wales.

6.1 STATISTICAL ANALYSIS OF STRAINS

The results of this pipeline include the allelic frequency of each variant, meaning a proportion of the Covid-positive population who had each variant can be derived. Each Coronavirus strain has multiple variants that result in different protein mutations, so it is difficult to assign viral lineages to wastewater samples as the variants come from many individuals.

However, if a rough model of the proportions of each strain circulating at the time the sample was taken is known, it would be possible to use combinatorial mathematics to calculate the frequencies of each strain in a population and therefore approximate the number of individuals infected with a given strain.

6.2 PIPELINE IMPROVEMENTS

With more time the I would have liked to allow my pipeline to have greater configuration by allowing the reference 'bait' file to be changed, and the steps of the pipeline to be modified depending on the operations the user wants to perform.

The packaging of the software into an easily installable Docker container would also be beneficial for distribution of the project to researchers globally that are not working on the Trinity cluster.

7. CONCLUSIONS

The aim of this project was to develop a pipeline to quantify the variants of SARS-CoV-2 present in sequenced wastewater samples. This pipeline has been developed to operate via a CLI tool that accepts multiple FASTQ sequences in single or paired-end format.

The result of the pipeline generates a HTML report detailing the variants present in the sample, suggests the relevance of each variant, and provides statistics on the coverage of the data over the SARS-CoV-2 genome. Output files with more details are also created in a nested project directory.

Monitoring of viral RNA in wastewater has the potential to enable an early warning system of viral outbreaks within local communities as the wastewater represents the local population. The identification and quantification of variants present in wastewater obtained in this pipeline can also enable the tracking of variants of concern across time and locations.

The results of this project could also be analysed with combinatorial mathematics to establish community ratios of circulating viral strains and approximate the number of infected individuals per strain.

In conclusion, the project produced successful results which carry a high degree of importance for the ongoing management of the coronavirus pandemic.

8. REFLECTION

Undertaking the project has been a great first experience in the field of Bioinformatics, and I have learnt a great deal of technical and theoretical knowledge from this interdisciplinary project.

It was initially overwhelming to begin a project with an unfamiliar working environment, interacting with a cluster intended for researchers through a solely command-line interface. While also having to learn the background knowledge from just an A-Level Biology background.

The highly technical nature of the project required a lot of research into my working environment, bioinformatics, and Biological background to sufficiently understand the meaning of my work and to interpret my results. The field of bioinformatics is littered with acronyms, abbreviations, and technical terms which took a lot of research and time to become familiar with their meaning.

Thankfully, I was grateful to have access to resources for Bioinformatics Masters students, with lecture recordings and slides shared with me by my client. I was also lucky to have my client Professor Peter Kille take on a mentoring role for me who taught me the basics of bioinformatics workflows and how to get started on the Trinity cluster environment.

The vast majority of my time working on this project has been research-based. The project required a lot of background knowledge and research so as a result, I found my time was often consumed by this significantly more than the coding aspect.

On reflection, this severely limited and reduced the time I could dedicate to my individual code for the project. I would have liked to experiment with developing my own tools for manipulating the data formats had I had more time. I would have also liked to create a better-automated testing system for the pipeline to both verify the results and measure the computational efficiency. I wish I could have made a greater focus on fully utilising the computational power of the cluster by finding the ideal way to multithread and maximise CPUs and RAM across jobs.

If I could do things differently, I would have attempted to narrow down the focus and scope of my project to maximise a small part of the pipeline instead of a large

part of it. However, I am grateful for the wider knowledge I have gained from creating the full project.

I believe this has opened the door for me to continue research in this field and I am hoping to take this forward by continuing to work with my client over the summer.

GLOSSARY

- Pharyngeal – Relating to the swabs commonly taken at Covid screenings from the throat at nasal cavity.
- FASTA- File format for biological sequences
- SARS-CoV-2- Covid-19, coronavirus
- K-mer: Short stretch of sequence
- Contigs: big fragments of DNA from sequences
- N50: contig in the middle of contigs lined up by size, the medium sized fragment.
- Variant – a change in the usual genetic code for a species
- Strain- a named classified variation on a virus often with multiple variants present that result in mutations

LIST OF ABBREVIATIONS

- NCBI - The National Center for Biotechnology Information. Hosts an online database including sequenced Covid-19 positive data.
- ENA- European Nucleotide Archive. Another online database with Covid-19 sequences available in FASTQ format.
- SAM- Sequence Alignment Map file
- BAM- Binary Alignment Map file. The binary equivalent of a SAM file.
- RNA- Ribonucleic acid
- DNA- Deoxyribonucleic acid
- VCF- Variant call format, designed for SNPs and short INDELs
- BCF- binary variant call format, is the binary version of VCF.

APPENDICES

APPENDIX 1: EXTERNAL TOOLS/SOFTWARE USED

- GNU Parallel <https://www.gnu.org/software/parallel/>
- SRAToolkit <https://github.com/ncbi/sra-tools>
- Fastp <https://github.com/OpenGene/fastp>
- FastQC
<https://www.bioinformatics.babraham.ac.uk/projects/fastqc/>
- Trimmomatic <http://www.usadellab.org/cms/?page=trimmomatic>
- MultiQC <https://multiqc.info/>
- InSilicoSeq <https://insilicoseq.readthedocs.io/en/latest/index.html>
- BCFTools <https://github.com/samtools/bcftools>
- SnpEff <https://pcingola.github.io/SnpEff/>
- FastTreeMP <http://www.microbesonline.org/fasttree/>

Python requirements:

- click==7.1.2 <https://click.palletsprojects.com/en/8.0.x/>
- Pandas <https://pandas.pydata.org/>
- colorama==0.4.4
- pyfiglet==0.8.post1
- termcolor==1.1.0
- Plotly <https://plotly.com/python/>
- scipy
- chart_studio
- IPython
- matplotlib

APPENDIX 2: DATA USED

Covid Wuhan-Hu-1 reference file

<https://www.ncbi.nlm.nih.gov/nuccore/1798174254>

Illumina UK wastewater dataset

<https://www.ebi.ac.uk/ena/browser/view/PRJEB42191>

GenBank Annotated SARS-CoV-2 Genome

<https://www.ncbi.nlm.nih.gov/nuccore/1798174254>

REFERENCES

- Babraham Bioinformatics - FastQC A Quality Control tool for High Throughput Sequence Data. [no date]. Available at: <https://www.bioinformatics.babraham.ac.uk/projects/fastqc/> [Accessed: 27 May 2021].
- Bankevich, A. et al. 2012. SPAdes: A new genome assembly algorithm and its applications to single-cell sequencing. *Journal of Computational Biology* 19(5), pp. 455–477. doi: 10.1089/cmb.2012.0021.
- Chavarria-Miró, G. et al. [no date]. Sentinel surveillance of SARS-CoV-2 in wastewater anticipates the occurrence of COVID-19 cases Running Title: Sentinel surveillance of SARS-CoV-2 in wastewater. Available at: <https://doi.org/10.1101/2020.06.13.20129627> [Accessed: 7 May 2021].
- Chen, S. et al. 2018. Fastp: An ultra-fast all-in-one FASTQ preprocessor. In: *Bioinformatics*. Oxford University Press, pp. i884–i890. Available at: [/pmc/articles/PMC6129281/](https://pmc/articles/PMC6129281/) [Accessed: 27 May 2021].
- Ewels, P. et al. 2016. MultiQC: Summarize analysis results for multiple tools and samples in a single report. *Bioinformatics* 32(19), pp. 3047–3048. doi: 10.1093/bioinformatics/btw354.
- GNU Parallel - GNU Project - Free Software Foundation. [no date]. Available at: <https://www.gnu.org/software/parallel/> [Accessed: 27 May 2021].
- Gourlé, H. et al. 2019. Simulating Illumina metagenomic data with InSilicoSeq. *Bioinformatics* 35(3), pp. 521–522. doi: 10.1093/bioinformatics/bty630.
- Hillary, L.S. et al. 2021. Monitoring SARS-CoV-2 in municipal wastewater to evaluate the success of lockdown measures for controlling COVID-19 in the UK. *Water Research*, p. 117214. doi: 10.1016/j.watres.2021.117214.
- How it works. [no date]. Available at: <https://nanoporetech.com/how-it-works> [Accessed: 28 May 2021].
- HPC Services - School of Biosciences. [no date]. Available at: <http://hpc.bios.cf.ac.uk/trinity-hpc/> [Accessed: 26 May 2021].
- iVar: Manual. [no date]. Available at: <https://andersen-lab.github.io/ivar/html/manualpage.html> [Accessed: 28 May 2021].
- Medema, G. et al. 2020. Presence of SARS-Coronavirus-2 in sewage. *medRxiv*, p. 2020.03.29.20045880. Available at: <https://doi.org/10.1101/2020.03.29.20045880> [Accessed: 7 May 2021].
- MinION | Oxford Nanopore Technologies. [no date]. Available at: <https://nanoporetech.com/products/minion> [Accessed: 9 May 2021].
- Napit, R. et al. 2021. Rapid genomic surveillance of SARS-CoV-2 in a dense urban community using 1 environmental (sewage) samples Genomic environmental surveillance of SARS-CoV-2 using sewage samples. *medRxiv*, p.

2021.03.29.21254053. Available at: <https://doi.org/10.1101/2021.03.29.21254053> [Accessed: 9 May 2021].

ncbi/sra-tools: SRA Tools. [no date]. Available at: <https://github.com/ncbi/sra-tools> [Accessed: 27 May 2021].

Nurk, S. et al. 2017. MetaSPAdes: A new versatile metagenomic assembler. *Genome Research* 27(5), pp. 824–834. Available at: </pmc/articles/PMC5411777/> [Accessed: 28 May 2021].

Pan, Y. et al. 2020. Viral load of SARS-CoV-2 in clinical samples. *The Lancet Infectious Diseases* 20(4), pp. 411–412. Available at: <https://doi.org/> [Accessed: 7 May 2021].

pandas - Python Data Analysis Library. [no date]. Available at: <https://pandas.pydata.org/> [Accessed: 27 May 2021].

Plotly Python Graphing Library | Python | Plotly. [no date]. Available at: <https://plotly.com/python/> [Accessed: 27 May 2021].

Sequencing | Key methods and uses. [no date]. Available at: <https://emea.illumina.com/techniques/sequencing.html> [Accessed: 28 May 2021].

Trevor Bedford on Twitter: “Since their recognition in the UK, South Africa and Brazil in Dec 2020 and Jan 2021, the variant of concern lineages B.1.1.7, B.1.351 and P.1 have continued to spread throughout the world with B.1.1.7 so far the most successful of the three. 1/15 <https://t.co/RtVLQPRUiV>” / Twitter. [no date]. Available at: <https://twitter.com/trvrb/status/1382365894106509319/photo/1> [Accessed: 7 May 2021].

USADELLAB.org - Trimmomatic: A flexible read trimming tool for Illumina NGS data. [no date]. Available at: <http://www.usadellab.org/cms/?page=trimmomatic> [Accessed: 27 May 2021].

Variant Calling part 2 (Galaxy) - Bioinformatics Documentation. [no date]. Available at: https://www.melbournebioinformatics.org.au/tutorials/tutorials/var_detect_advanced/var_detect_advanced/ [Accessed: 18 May 2021].

Vilar, S. and Isom, D.G. 2020. One year of SARS-CoV-2: How much has the virus changed? *bioRxiv*, p. 2020.12.16.423071. Available at: <https://doi.org/10.1101/2020.12.16.423071> [Accessed: 10 May 2021].

Welcome to Click — Click Documentation (8.0.x). [no date]. Available at: <https://click.palletsprojects.com/en/8.0.x/> [Accessed: 27 May 2021].

WinSCP :: Official Site :: Free SFTP and FTP client for Windows. [no date]. Available at: <https://winscp.net/eng/index.php> [Accessed: 26 May 2021].

Zhang, J.C. et al. 2020. Fecal specimen diagnosis 2019 novel coronavirus–infected pneumonia. *Journal of Medical Virology* 92(6), pp. 680–682. Available at: <https://onlinelibrary.wiley.com/doi/full/10.1002/jmv.25742> [Accessed: 7 May 2021].

