NextFlow pipeline for SARS-CoV-2 Illumina data

Table of contents

Quickstart

Installation and usage

1. Install Docker (https://docs.docker.com/desktop/install/linux-install/)

2. Install NextFlow

```
curl -s https://get.nextflow.io | bash
mv nextflow ~/bin
```

3. Clone the repository:

```
git clone https://github.com/mkadlof/nf_illumina_sars
```

4. Copy `third-party/modeller/config.py.template` to `third-party/modeller/config.py` and replace the line

`license = 'YOUR_MODELLER_KEY'`

with the actual Modeller key you own. If you don't have one, you can get a free academic license here (https://salilab.org/modeller/registration.html).

5. Build three containers:

```
docker build --target production -f Dockerfile-main -t
nf_illumina_sars-3.0-main .
docker build --target prodcution -f Dockerfile-manta -t
nf_illumina_sars-3.0-manta .
docker build --target updater -f Dockerfile-main -t nf_illumina_sars-
3.0-updater:latest .
```

6. Download latest version of external databases:

In project root dir run:

```
./update_external_databases.sh
```

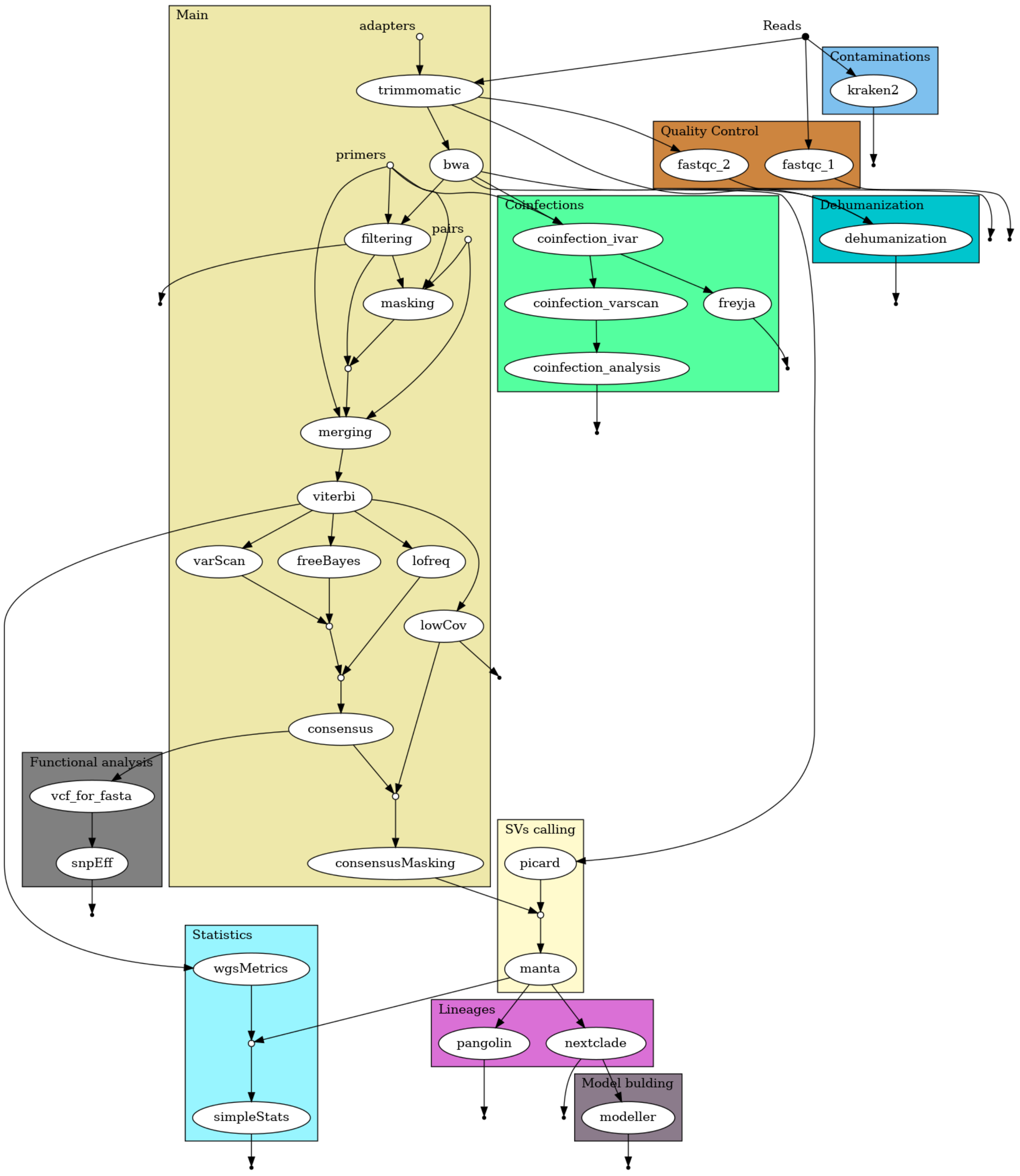
This should fill directories in `data/pangolin` and `data/nextclade`. For more details read the chapter External databases updates.

7. Copy `run_nf_pipeline.sh.template` to `run_nf_pipeline.sh` and fill in the paths to the reads and output directory.

8. Run the pipeline:

```
./run_nf_pipeline.sh
```

Pipeline overview



Overview of the pipeline.

NextFlow pipeline for SARS-CoV-2 Illumina data consist of 30 NextFlow modules. The steps are depicted on image in Pipeline overview (`IndexGenome` module which gives input to many other modules was hidden for clarity). Modules are grouped into logical sections, by their function.

Main section

Goal of **Main** section is to perform mapping with bwa (https://bio-bwa.sourceforge.net/) aligner, filtering, small indel calling by three callers (varScan (https://varscan.sourceforge.net/), freeBayes (https://github.com/freebayes/freebayes) and lofreq (https://csb5.github.io/lofreq/)), identify low quality / coverage regions and finally obtain consensus sequence.

Quality control

This section executes FastQC (https://www.bioinformatics.babraham.ac.uk/projects/fastqc/). Test is run on raw dataset and after trimming adapters with Trimmomatic (http://www.usadellab.org/cms/?page=trimmomatic).

Contamination detection

This section test if sample is nt contaminated with other species than SarsCov-2. Kraken2 (https://ccb.jhu.edu/software/kraken2/) is used.

Dehumanization

This section is used to create a subset of reads in FASTQ format without any reads from organisms other than SARS-CoV-2. Own python scripts are used.

Coinfections

This section is to detect infections with more than one variant of Sars-Cov-2 virus. Coinfections are detected with both own python scripts and Freya (https://andersen-lab.github.io/Freyja/index.html).

Functional analysis

This module returns the effects of nucleotide mutations onto protein sequence. We employ SnpEff (http://pcingola.github.io/SnpEff/).

SVs Calling

This module is capable to detect large genome rearrangements (*Structural variants*). This is done by down sampling reads with Picard (https://broadinstitute.github.io/picard/) and running Manta (https://github.com/Illumina/manta).

Statistics

Some simple stats ara calculated with both - own python script and tools from Picard tools.

Lineages

After SVs calling pango (https://cov-lineages.org/resources/pangolin.html) line and nextclade (https://clades.nextstrain.org/) lineages are identified.

Building Spike protein model

As a last step Spike protein model is built with Modeller (https://salilab.org/modeller/)

Hardware requirements

Platform

Pipeline is intended to run on Unix-like computing server with x86_64 architecture.

Memory and CPU

Pipeline consist of multiple steps (processes) that are run in separate containers possibly concurrently. Each process has its own hardware requirements. Proceses vary highly in their demands, from very low to very high. Some may benefit from multiple cores while others are single-threaded or fast enough to not require more than one core. Exact requirements depend on expected number of samples analyzed in parallel.

During our tests we run the pipeline for 32 samples in parallel on a machine with 96 cores (Intel(R) Xeon(R) Gold 6240R CPU @ 2.40GHz) and 503 GiB of RAM, and we faced out-of-memory issues. We introduced limits on memory-intensive processes (BWA and Kraken2) to maximum 5 concurrent instances, to avoid OOM killer.

Single sample mode

In case of running the pipeline in single sample mode we recommend using at least 16 cores and 64 GiB of RAM.

Multiple samples mode

In case of running set of samples in parallel we recommended using 4 cores and ~100 GiB of RAM per sample.

Storage requirements

Total storage requirements is **~60 GiB** of constant data, and further **~3.3 GiB per sample**.

Performance

Many processes in the perform a lot of I/O operations, thus pipeline definitely can benefit from fast storage. We recommend store external databases, and temporary files in fast storage like NVMe SSD in RAID 0. It also may be beneficial to store it on in-memory filesystem like `tmpfs`, however no extensive tests were performed.

Docker images sizes:

Pipeline consist of three docker images two for computations and one is wrapper for external databases updates.

Image	Size
`nf_illumina_sars-3.0-main`	1.82 GiB
`nf_illumina_sars-3.0-updater`	257 MiB
`nf_illumina_sars-3.0-manta`	1.26 GiB
Total	**3.33 GiB**

Databases sizes:

Pipeline require access to external databases. Total size of databases is **~56 GiB**.

Database	Size
pangolin	~90 MiB
nextclade	~1.3 MiB
kraken	~55 GiB
freyja	~100 MiB
Total	**~56 GiB**

Temporary files and results

Pipeline generates a lot of temporary files, which are stored in `work` directory. According to our tests on EQA2023 dataset (32 samples) took ~105 GiB of disk space **~3.3 GiB per sample**. Please note that those tests are not representative, and real life data may vary significantly.

GPU requirements

Pipeline does not exploit GPU acceleration, so no GPU is required.

Software requirements

This pipeline is designed to run on a Unix-like operating system. The pipeline is written in Nextflow (https://www.nextflow.io/docs/latest/index.html), which is a language for writing bioinformatics pipelines. It is designed to be portable and scalable, and can be run on a variety of platforms, including local machines, clusters, and cloud computing environments.

Our pipeline is containerized using Docker (https://www.docker.com/), which is a platform for developing, shipping, and running applications in containers. Containers allow a developer to package up an application with all the parts it needs, such as libraries and other dependencies, and ship it all out as one package. This makes it easy to deploy the application on any machine that supports Docker.

However, our containers are run differently than usual docker workflow. Containers are run by nextflow, instead of manual execution of docker. Nextflow take care of mounting volumes and deciding which container should be run and when.

Compatibility

Pipeline was tested with following software versions:

- **Operating system**:
  - Ubuntu 20.04.06 LTS
  - Debian 12.5
- **Docker**:
  - 24.0.7
  - 26.0.2
- **Nextflow**:
  - 23.10.1.5891

It is known that pipeline will not work with docker 20.10.5.

External databases updates

Some components of the pipeline require access to their two databases, which are updated roughly once every two weeks. Different software pieces need to be updated in different ways. To make this process as smooth and painless as possible, we prepared a dedicated Docker container exactly for this task, along with a bash script for running it with appropriate volume mounts. The script should be placed in either the cron or systemd timer and run on a weekly basis.

Updates procedure

Build the dedicated container:

```
docker build --target updater -f Dockerfile-main -t nf_illumina_sars-3.0-updater:latest .
```

Run the updater script. The working dir must be in project root directory.

```
update_external_databases.sh nextclade
update_external_databases.sh pangolin
update_external_databases.sh kraken
update_external_databases.sh freyja
```

Total size of downloads is ~55 GiB.

Database	Size
pangolin	~90 MiB
nextclade	~1.3 MiB
kraken	~55 GiB
freyja	~100 MiB

If everything work fine in directories `data\pangolin` and `data\nextclade` you should see downloaded content like below:

```
data/nextclade/
└── sars-cov-2.zip

data/pangolin/
├── bin
├── pangolin_data
└── pangolin_data-1.25.1.dist-info

data/kraken
└── k2_standard_20240112.tar.gz

data/freyja/
├── curated_lineages.json
├── lineages.yml
└── usher_barcodes.csv
```

It is recommended to put the following in crontab or equivalently systemd timer.

```
0 3 * * 6 cd /path/to/sars-illumina &&
bin/update_external_databases.sh nextclade
5 3 * * 6 cd /path/to/sars-illumina &&
bin/update_external_databases.sh pangolin
10 3 * * 6 cd /path/to/sars-illumina &&
bin/update_external_databases.sh freyja
15 3 1 */3 * cd /path/to/sars-illumina &&
bin/update_external_databases.sh kraken
```

Updates internals

The following section contain information what and how is updated. Unless you need to debug or refactor the code, and you followed guides in chapter "Updates procedure" in "External databases updates" you can safely skip it.

List of components that require updates

- Nextclade
- Pangolin

- Kraken
- Freyja

Updateing Nextclade database

Nextclade (https://docs.nextstrain.org/projects/nextclade/) is software for assigning evolutionary lineage to SARS-Cov2. To make it work properly, it requires a database which is updated roughly once every two weeks.

The recommended way of downloading dataset is using `nxtclade` tool.

```
nextclade dataset get --name sars-cov-2 --output-zip sars-cov-2.zip
```

Detailed manual is available https://docs.nextstrain.org/projects/nextclade/en/stable/user/datasets.html. Nextclade is downloading `index.json` from site: https://data.clades.nextstrain.org/v3/index.json, and based on that files it decide what to download and from where. Probably the same data are available directly on GitHub: https://github.com/nextstrain/nextclade_data/tree/master/data/nextstrain/sars-cov-2/wuhan-hu-1/orfs.

The command above will download a `sars-cov-2.zip` file in desired destination (default: `data/nextclade`). That directory have to be mounted inside `main` container. It is done by Nextflow in the `modules/nextclade.nf` module.

```
process nextclade {
    (...)
    containerOptions "--volume
${params.nextclade_db_absolute_path_on_host}:/home/SARS-
CoV2/nextclade_db"
    (...)
```

Updateing Pangolin database

Pangolin (https://github.com/cov-lineages/pangolin) (**P**hylogenetic **A**ssignment of **N**amed **G**lobal **O**utbreak **LIN**eages) is alternative to Nextclade software for assigning evolutionary lineage to SARS-Cov2.

To make it work properly, it requires a database that is stored in the Git repository pangolin-data (https://github.com/cov-lineages/pangolin-data).

Pangolin-data is actually a regular python package. Normal update procedure is via command: `pangolin --update-data`. It also can be installed by `pip` command. Keeping it inside main container is slightly tricky. We don't want to rebuild entire container just to update the database. We also don't want to keep the database inside the container, because it would force us to run the update before every pipeline run, which is stupid. The best solution is to mount the database from the host.

To achieve this goal we install the package externally to the container in designated path using host native `pip`.

```
pip install \
    --target data/pangolin \
    --upgrade \
    git+https://github.com/cov-lineages/pangolin-data.git@v1.25.1
```

Make sure you entered proper version in the end of git url. The version number is also git tag. List of available tags with their release dates is here (https://github.com/cov-lineages/pangolin-data/tags).

Then that dir is mounted as docker volume inside the container (which is done automagically in the Nextflow module file):

```
process variantIdentification {
    containerOptions "--volume
${params.pangolin_db_absolute_path_on_host}:/home/SARS-CoV2/pangolin"
    (...)
```

During container build the `$PYTHONPATH` environment variable is set to indicate proper dir.

```
(...)
ENV PYTHONPATH="/home/SARS-CoV2/pangolin"
(...)
```

So the manual download consist of two steps:

1. Install the desired version of `pangolin-data` package in `data/pangolin` directory.
2. Provide absolute path to that dir during starting pipeline

   `--pangolin_db_absolute_path_on_host /absolute/path/to/data/pangolin`

Updateing Kraken database

Kraken 2 (https://ccb.jhu.edu/software/kraken2/) is a taxonomic classification system using exact k-mer matches. The pipeline utilizes it to detect contamination in samples. It requires a database of approximately 55 GiB, which could potentially be reduced to 16 GiB or 8 GiB, albeit at the cost of sensitivity and accuracy. It updated roughly quarterly. Skipping updates may result in skipping newer taxa.

Database may be built for user own (not recommended) or be downloaded from aws s3. DB is maintained by Kraken 2 maintainers. Here you can find more here (https://github.com/BenLangmead/aws-indexes), and here (https://benlangmead.github.io/aws-indexes/).

Downloading is via aws cli (`apt install awscli`), python library `boto3` or HTTP protocol. There are several types of databases, that differ with set of organism. We use and recommend using `standard` db, which is quite complete. There is also `nt` db which contain all RefSeq and GenBank sequences, but it's size and processing time is too high for routine surveillance.

Links for http downloads are available here (https://benlangmead.github.io/aws-indexes/k2). They are in form of: `https://genome-idx.s3.amazonaws.com/kraken/k2_DBNAME_YYYMMDD.tar.gz`

where DB name is one of following: `standard`, `standard_08gb`, `standard_16gb`, `viral`, `minusb`, `pluspf`, `pluspf_08gb`, `pluspf_16gb`, `pluspfp`, `pluspfp_08gb`, `pluspfp_16gb`, `nt`, `eupathdb48`.

Refer to official docs for more details.

In case of our pipeline we use python script that is using `boto3` library. It is part of `nf_illumina_sars-3.0-updater` container. For running this script simply pass `kraken` to the container during running.

We recommend using for this dedicated script: `update_externeal_databases.sh`.

Kraken DB will not be updated if local path already contain the file with the same name.

Updateing Freyja database

Freyja (https://andersen-lab.github.io/Freyja/index.html) is a tool to recover relative lineage abundances from mixed SARS-CoV-2 samples from a sequencing dataset.

Natively Freyja updates require installing Freyja python package (by default with conda) and running `freyja update` command. This command will download the latest version of curated lineages and usher barcodes. However, in our pipeline we do it simpler way. We download the files directly from the dedicated GitHub repository andersen-lab/Freyja-data (https://github.com/andersen-lab/Freyja-data) using regular `wget`, which is implemented in the `update_external_databases.sh` script, and `updater` container.

Running pipeline

Pipeline parameters

There are three types of flags that we use explicit, implicit and nextflow flags. Explicit MUST be provided during starting pipeline - they are paths for input files. Explicit parameters are mostly numeric values for various modules. They have set reasonable default values and usually there is no need to modify them. NextFlow flags aplay to the way how NextFlow is executed rather than to pipeline itself.

By convention pipeline params starts with two dash `--param_name`, while NextFlow flags starts with single dash `-param-name`.

Explicit pipeline parameters

```
./run_pipeline.sh \
--ref_genome 'path/to/reference/genome.fasta' \
--reads 'path/to/reads/sample_id_{1,2}.fastq.gz' \
--primers 'path/to/primers.bed' \
--pairs 'path/to/pairs.tsv' \
--adapters 'path/to/adapters.fa' \
--pangolin_db_absolute_path_on_host '/home/user/path/to/pangolin_db' \
--nextclade_db_absolute_path_on_host '/home/user/path/to/nextclade_db'
\
--kraken2_db_absolute_path_on_host '/home/user/path/to/kraken2_db' \
--freyja_db_absolute_path_on_host '/home/user/path/to/freyja'
```

`ref_genome` - path to reference genome fasta file

`reads` - path to reads in fastqc format. Must be gzipped and be in form: `sample_id_{1,2}.fastq.gz`. Name must be resorvable by shell into two different files. One for forward reads, and second fo reverse reads.

`primers` - primers in bed file format. Example is below.

```
MN908947.3   2826    2850    nCoV-2019_10_LEFT    1    +
TGAGAAGTGCTCTGCCTATACAGT
MN908947.3   3183    3210    nCoV-2019_10_RIGHT   1    -
```

```
TCATCTAACCAATCTTCTTCTTGCTCT
(...)
```

Common primers sets are included in data/generic/primers directory and include following:

SARS1_partmerge_exp, SARS2_partmerge_exp, V1, V2, V3, V4, V4.1, V1200, V1201

pairs - definition of primers identifiers in two column tab separated file. This file is included in corresponding every primers set in data/generic/primers. Structure of primer identifier is meaningful. Must match regexp nCoV-2019_[1,2]_(LEFT,RIGHT). Example:

```
nCoV-2019_1_LEFT    nCoV-2019_1_RIGHT
nCoV-2019_2_LEFT    nCoV-2019_2_RIGHT
(...)
```

adapters - path to fasta file with adapters. Common adapters are included in data/generic/primers. Example:

```
>PrefixPE/1
TACACTCTTTCCCTACACGACGCTCTTCCGATCT
>PrefixPE/2
GTGACTGGAGTTCAGACGTGTGCTCTTCCGATCT
```

Implicit pipeline parameters

All implicit parameters are listed in main pipeline file nf_pipeline.nf with their reasonable defaults.

```
params.threads = 5
params.memory = 2024
params.quality_initial = 5
params.length = 90
params.max_number_for_SV = 200000
params.max_depth = 6000
params.min_cov = 20
params.mask = 20
params.quality_snp = 15
params.pval = 0.05
params.lower_ambig = 0.45
params.upper_ambig = 0.55
params.ref_genome_id = "MN908947.3"
```

To run pipeline with modified parameter simply add appropriate flag:

```
./run_nf_pipeline.sh --threads 10
```

`--threads`, positive integer, number of threads used by couple of different modules. In single sample mode should be set to 1/3 of available CPUS. In multisample mode should be adjusted empirically. Recommended value for decent server: 5.

`--memory`, positive integer in MiB, similar to above. Default: 2024

`--quality_initial`, positive integer in PHRED scale, per base quality threshold used in various filtering and reporting modules. Default: 5.

`--length`, positive integer - number of base pairs, minimum length of a read. Default: 90

`--max_number_for_SV` - positive integer, maximum number of reads in bam file for manta module, down sampled by Picard, Default: 200000

`--max_depth` - positive integer, number of base pairs, threshold for short indel callers, reads above this value will be discarded. This is used for speedup indel calling. Default: 6000

`--min_cov` - positive integer, number of base pairs, threshold below which mutation will not be called. Default: 20

`--mask` - positive integer, number of base pairs, below this coverage value, genome will be masked with N. Should be the same as `min_cov`. Default: 20

`--quality_snp` - positive integer, PHRED scale, minimum quality of a base for INDEL calling, Default: 15

`--pval` - float from range [0; 1], minimal probability for INDEL calling, Default: 0.05

`--lower_ambig` and `--upper_ambig` - float from range [0, 1], if fraction of reads introducing alternative allel, fall within this range the position will be classified as *ambiguity*. `upper_ambig` must be greater than `lower_ambig`. Default: [0.45; 0,55]

`--ref_genome_id` - string, identifier of reference genome. Do not change unless you know what you are doing. Default: `MN908947.3`

Nextflow parameters

This parameters comes with Nextflow and should not be modified without solid reason.

`-config` path to nextflow config file. Default file `nextflow.config` is provided with repo. `-with-report` path to report from pipeline execution. May be safely disabled. Default: `report.html` `-with-dag` path tu file with pipeline graph. May be safely disabled Default: `flowchart-raw.png` `-with-docker` Docker image used for execution processes. Strictly required. Default: `nf_illumina_sars-3.0-main:latest` `-resume` Control if restarted pipeline should use cached results or not. Irrelevant in production environment, since every sample will be always run exactly once. In development or during debug may significantly speed up things.

Pipeline Steps

Main section

Module Trimmmomatic

```
java -jar /opt/trimmomatic/trimmomatic.jar PE ${reads[0]} ${reads[1]}
\
                                           forward_paired.fastq.gz \
                                           forward_unpaired.fastq.gz \
                                           reverse_paired.fastq.gz \
                                           reverse_unpaired.fastq.gz \

ILLUMINACLIP:${adapters}:2:30:10:8:True \

LEADING:${params.quality_initial} \

TRAILING:${params.quality_initial} \
                                           SLIDINGWINDOW:4:4 \
                                           MINLEN:${params.length}
```

At this stage, we remove nucleotides of low quality from the 5' and 3' ends of the reads as well as adapter sequences from the reads. Then, we filter out reads that do not meet the length criterion. Those pairs in which both reads meet the length criterion are saved to appropriate files named "paired", and those pairs in which one of the reads does not meet the length criterion after filtering are saved to files named "unpaired". Adapter sequences are provided in the form of .fasta files. In the pipeline, we use adapter sequences provided by the authors of the Trimmomatic program (e.g., default sequences from the TruSeq3-PE-2.fa file), which have been placed in the container in the /SARS-CoV2/adapters/ directory.

When setting the quality threshold, it is recommended to use low qualities (default is just 5), which may seem "unorthodox" at first glance. However, it should be remembered that we are dealing with amplicon-based sequencing, and the crucial information here is from which amplicon the read originates and whether it maps to any of the primers. Removing a fragment of the read from its 5' and 3' ends may lead to its incorrect classification, which will affect further analysis. On the other hand, not removing reads of very poor quality may result in incorrect read alignment. Below is an example of what excessive zeal in removing nucleotides from the 5’/3’ end can lead to.

a. Original situation (X - any nucleotide; P – primer position in the reference genome; M - masked position in the read, i.e., not used for variant identification or coverage counting)

```
Read                       XXXXXXXXXXXXXXXXXXXXXXX
Reference genome XXXXXXXXXXXXXPPPPPPPPPXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

b. Mapping after rigorous removal of nucleotides from the 5’/3’ end of the read.

```
Read                            XXXXXXXXXXXXXXXXXX
Reference genome XXXXXXXXXXXXXPPPPPPPPPXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

c. Final effect after masking primers for the read after rigorous removal of the 5’/3’ end (described in Step 5)

```
Read                              MMMMXXXXXXXXXXX
Reference genome XXXXXXXXXXXXXPPPPPPPPPXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
```

In the above situation, it can be seen that the read does not originate from amplification using the shown primer (it maps before its start). However, after removing the initial nucleotides (which tend to have lower quality), we will assume that the read was indeed generated using this primer. Consequently, the entire read region that maps to this primer will be masked, and we will not use this information in further analysis. Although this problem may seem insignificant due to excessive sequencing of SARS-CoV-2 samples with coverages exceeding tens of thousands, it has serious consequences in regions where the use of a given amplicon is small and the information conveyed by individual reads is very valuable. Additionally, please note that the variable $length is set to 90. This is related to the analysis of one of the EQA test sequences, where short reads (length ~50) were artificially boosting coverages near the 5’ end of the genome.

Alphabetic list of modules

- bwa.nf
- coinfection_analysis.nf
- coinfection_ivar.nf
- coinfection_varscan.nf
- consensus.nf
- consensusAnalysis.nf
- consensusMasking.nf
- dehumanization.nf
- fastqc.nf
- filtering.nf
- freeBayes.nf
- freyja.nf
- functionalAnalysis.nf
- indexGenome.nf
- kraken2.nf
- lofreq.nf
- lowCov.nf
- manta.nf
- masking.nf
- merging.nf
- modeller.nf
- nextclade.nf

- pangolin.nf
- picard.nf
- simpleStats.nf
- trimmomatic.nf
- varscan.nf
- vcf_for_fasta.nf
- viterbi.nf
- wgsMetrics.nf