

PIEpiSeq Bioinformatic Pipelines Documentation

The document provides technical documentation for the bioinformatics processing protocols (also known as pipelines) developed in the PIEpiSeq project. These protocols are designed for the automatic analysis of SARS-CoV-2, influenza, and RSV viruses sequenced using Illumina and Nanopore technologies. It covers the architecture, installation and execution procedures, the content of the Nextflow modules, and details of the validation tests.

Authors:
Michał Kadlof
Michał Łażniewski

Table of Contents

Quickstart	2
Hardware requirements	4
Software requirements	8
Pipeline overview	9
External databases updates	17
Updateing Nextclade database	20
Updateing Pangolin database	21
Updateing Kraken database	23
Updateing Freyja database	24
Running pipeline	25
Pipeline Steps	29
Module: fastqc	32
Modules: kraken2 (illumina and nanopore versions)	34
Module: trimmomatic	36
Pipeline customization	37

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/pzh_pipeline_viral
```


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


```
license = 'YOUR_MODELLE_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 main -f Dockerfile-main -t
pzh_pipeline_viral-4.1-main .
docker build --target manta -f Dockerfile-manta -t
pzh_pipeline_viral-4.1-manta .
docker build --target updater -f Dockerfile-main -t
nf_illumina_sars-4.1-updater .
```

 You may add `--no-cache` flag to avoid caching effects.

 If you encounter a `certificate verify failed` error during the build process, it may be due to being on a corporate network that injects its own certificate. In this case, add the following flag to the build command, where you can pass the certificate provided by your administrator into the container. `--build-arg CERT_FILE="$(cat corporate-certificate.crt)"`

6. Download latest version of external databases:

In project root dir run:

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

For more details read the chapter External databases updates ([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
```

Hardware requirements

Platform

Pipeline is intended to run on any general purpose GNU/Linux computing server with x86_64 architecture.

CPU

The pipeline consists of multiple steps (processes) that run in separate containers, potentially concurrently. Each process has its own hardware requirements, which can vary significantly—from very low to very high demands. Some processes benefit from multiple cores, while others are single-threaded or fast enough to not require more than one core. The exact requirements depend on the number of samples expected to be analyzed in parallel. In general, the pipeline can run with any number of cores, and additional cores may reduce computation time, especially when analyzing multiple samples simultaneously.

Single sample mode


In case of running the pipeline in single sample mode we recommend using at least **8 cores per sample**.

Multiple samples mode

In case of running set of samples in parallel we recommended using **4 cores per sample**.

Memory

The most memory-intensive process is Kraken2, which must load its entire database into memory. The size of the database determines the overall memory requirement. By default, we use the standard database, which is approximately 80 GiB. Additional memory is required to support other processes and the operating system, resulting in a total memory requirement of at least 82 GiB per sample.

 Memory requirements can be significantly reduced by choosing a smaller database (e.g., one containing only specific branches of the Tree of Life).

A list of available databases, their contents, and sizes is provided here <https://benlangmead.github.io/aws-indexes/k2> (<https://benlangmead.github.io/aws-indexes/k2>).

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	2.01 GiB
nf_illumina_sars-3.0-updater	258 MiB
nf_illumina_sars-3.0-manta	1.72 GiB
Total	3.98 GiB

Databases sizes:

Pipeline require access to external databases. Total size of databases is 80.8 GiB.

Database	Size
pangolin	90.2 MiB
nextclade	19.5 MiB
kraken	80.5 GiB
freyja	113.4 MiB
Total	80.8 GiB

i Storage requirements of databases can be significantly reduced the same way as RAM memory requirement.

Temporary files and results

The pipeline generates a large number of temporary files, which are stored in the work directory, and the actual results data in the results directory. The majority of the space in the results directory is occupied by dehumanized FASTA files.

The results of our tests are summarized in the table below; however, please note that these data are not representative of real-life sequencing scenarios, and actual resource demands may vary significantly.

Pipeline	Temporary Storage Average size per sample.	Results Average size per sample.
ILLUMINA_SARS	4.43 GiB	355.41 MiB
ILLUMINA_INFL	2.28 GiB	???.??
ILLUMINA_RSV	1.35 GiB	117.5 MiB
NANOPORE_SARS		
NANOPORE_INFL		
NANOPORE_RSV		

Average results per sample.

GPU requirements

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

Software requirements

This pipeline is designed to run on a general purpose GNU/Linux 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:**
 - 24.04.4.5917

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

Pipeline overview

The pipeline was written using the Nextflow Framework. It has a modular structure in the sense that individual processes (tools) are located in separate files called modules, and the entire pipeline is invoked from the main file named `nf_pipeline_viral.nf`.

The main file actually contains 6 pipelines for all combinations of three viral organisms (SARS-CoV-2, Influenza, and RSV), and two technologies (Illumina and Nanopore). The pipelines can share selected modules or groups of modules, but in areas requiring specific analyses, they use dedicated modules designed only for those purposes. Which pipeline is invoked is determined by the input parameters.

In each pipeline, the following stages of analysis can be distinguished:

1. Quality control of reads (including contamination detection)
2. Mapping of reads to the reference genome
3. Filtering and downsampling of reads
4. Detection of structural variants
5. Functional analysis
6. Descriptive statistics
7. Phylogenetic analysis
8. 3D Modeling of selected proteins
9. Results aggregation

All pipelines are embedded in Docker images. Containers are run by natively installed Nextflow Framework.

Pipelines as DAGs

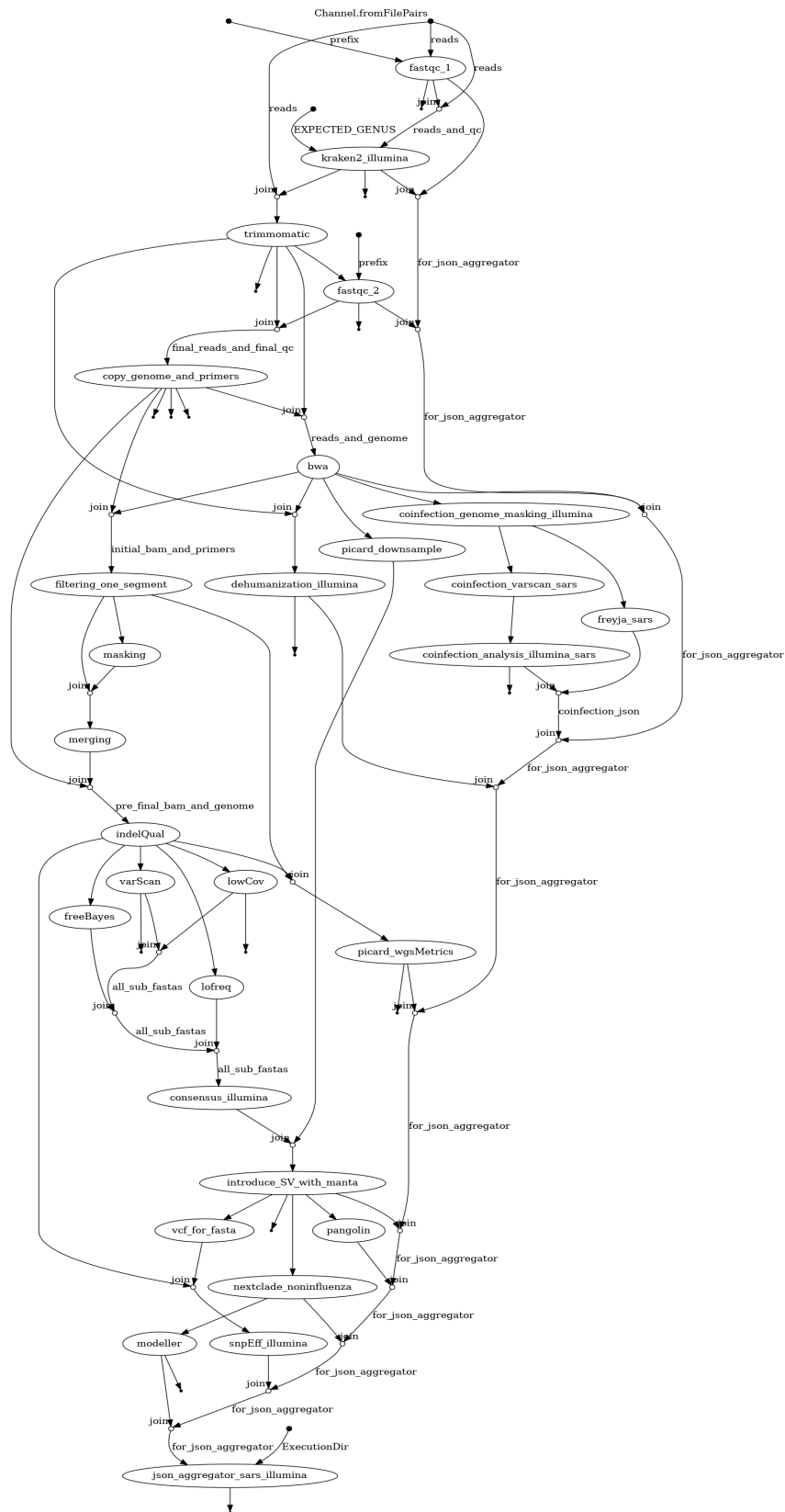
Each Nextflow module consists of four basic sections:

- Module settings (meta-information, as well as the method of execution and result

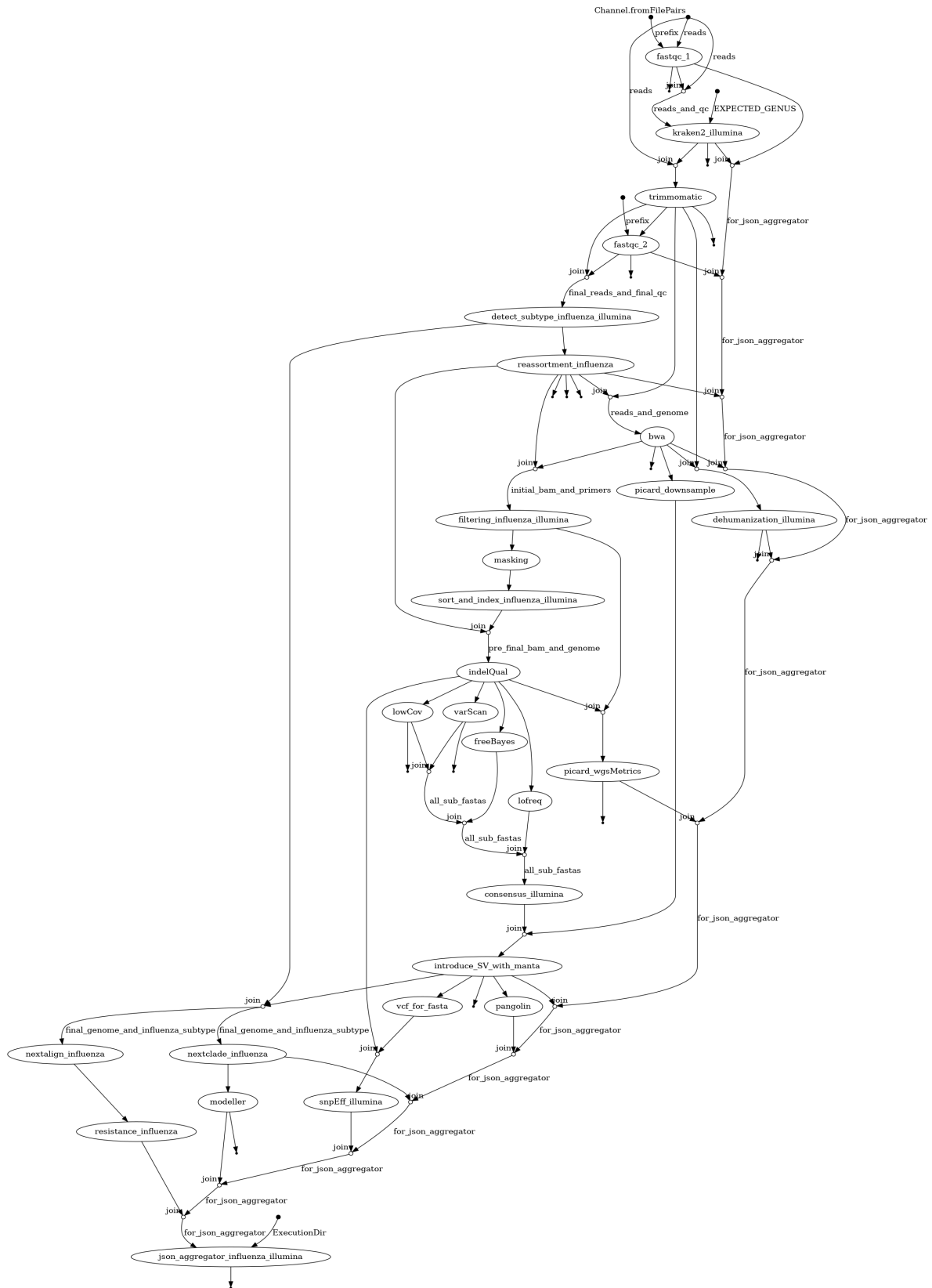
return)

- Definition of input channels
- Definition of output channels
- Script executing the task (process)

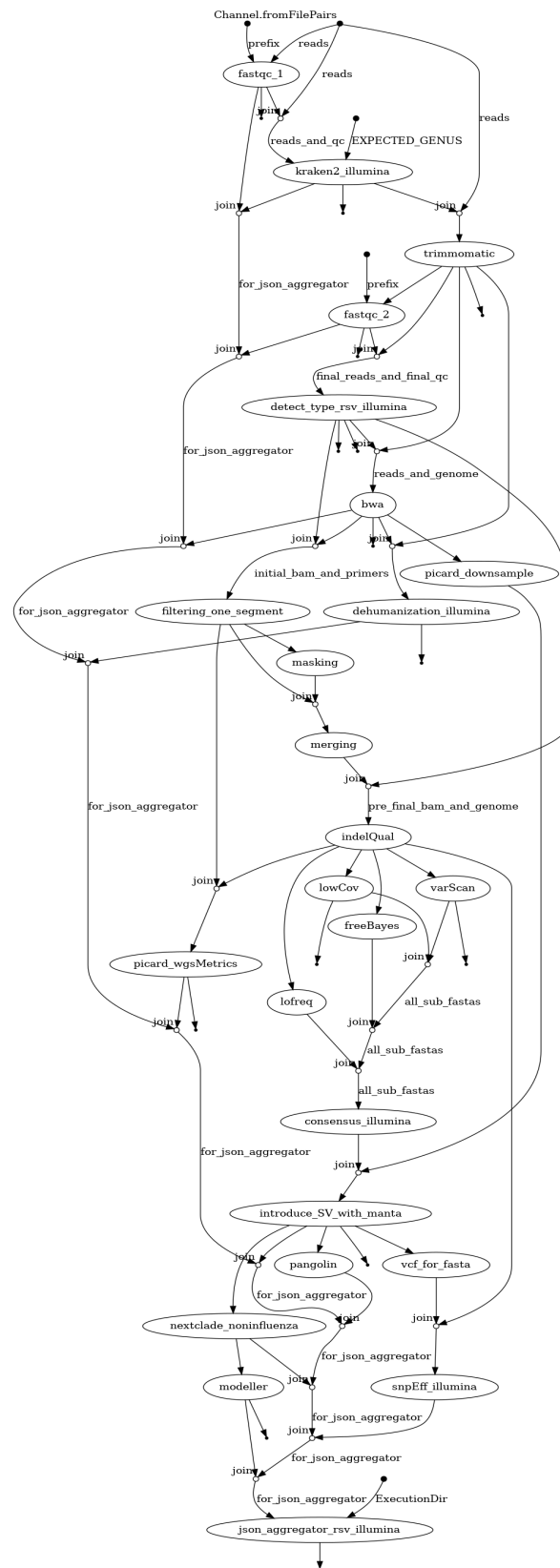
Taka organizacja pozwala na łączenie modułów w dowolnych kombinacjach, o ile wyjście jednego modułu jest kompatybilne z wyjściem drugiego modułu. Połączone moduły tworzą acykliczny graf skierowany (tzw. DAG). Grafy te wygodnie jest przedstawiać w formie graficznej, dzięki czemu można z łatwo prześledzić proces analizy danych. Poniższe 6 stron zawiera wszystkie 6 pipeline'ów w formie DAG'ów. Na przedstawionych grafikach owale reprezentują procesy, zaś linie je łączące kanały przepływu danych.

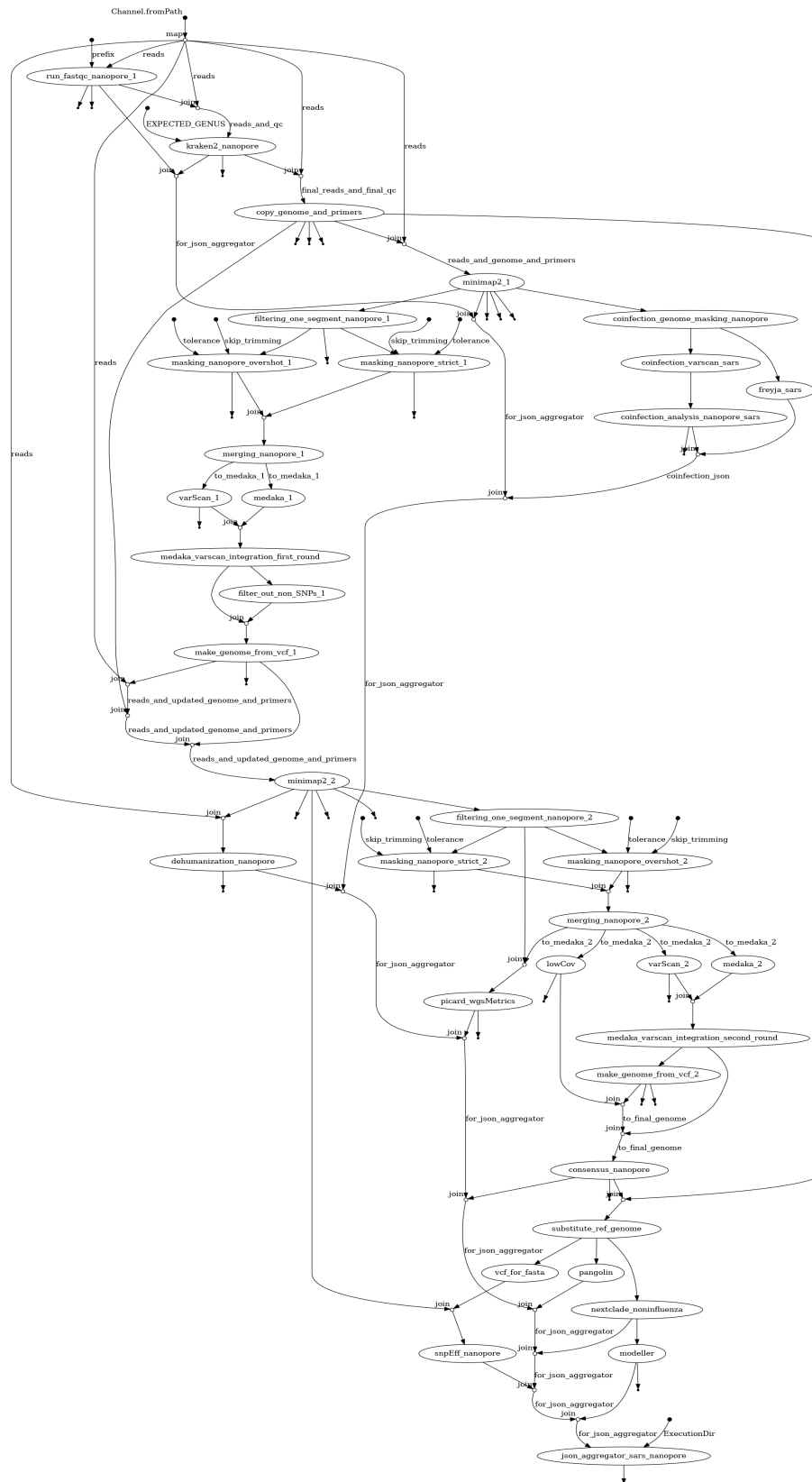


Illumina Sars-CoV-2 Flowchart

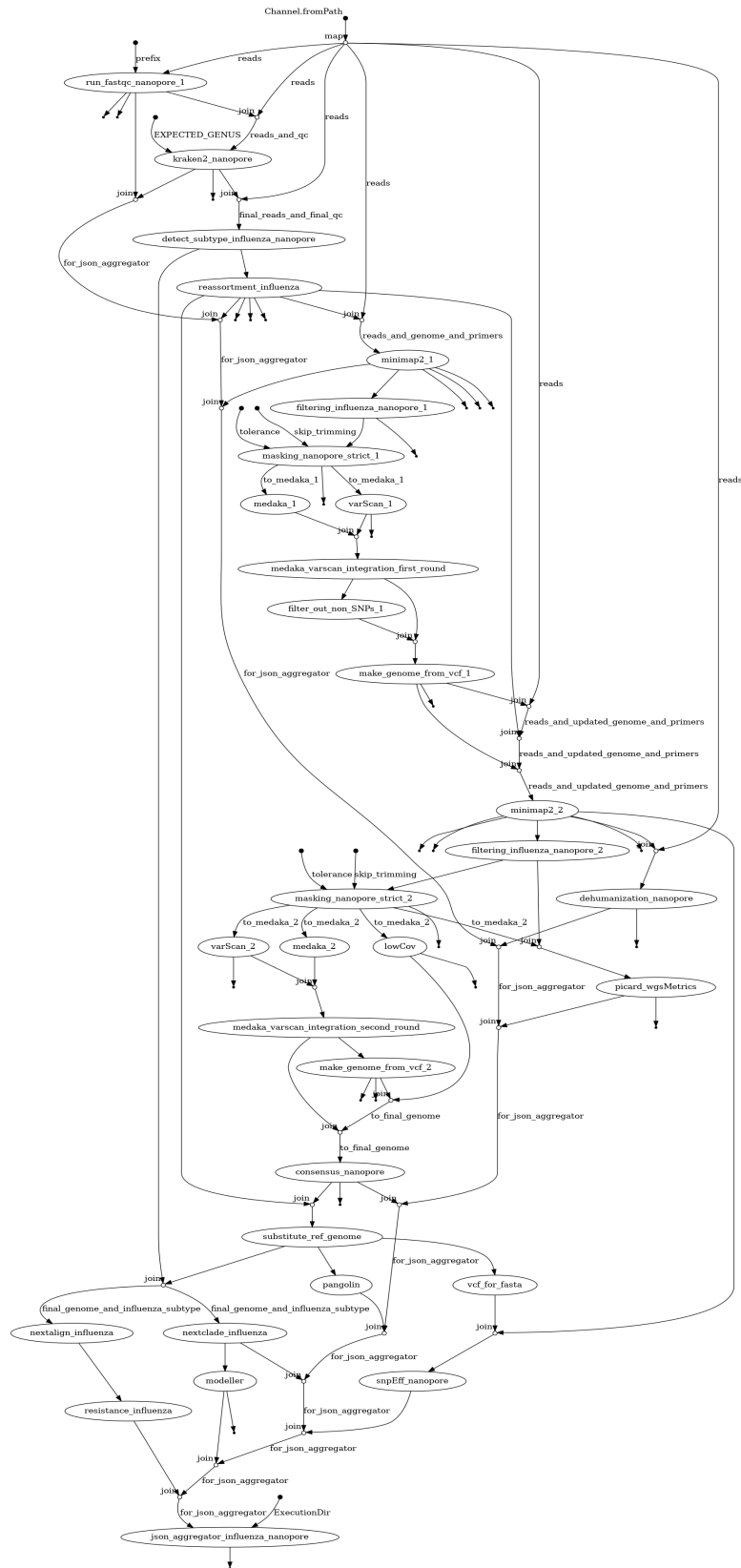


Illumina Influenza Flowchart

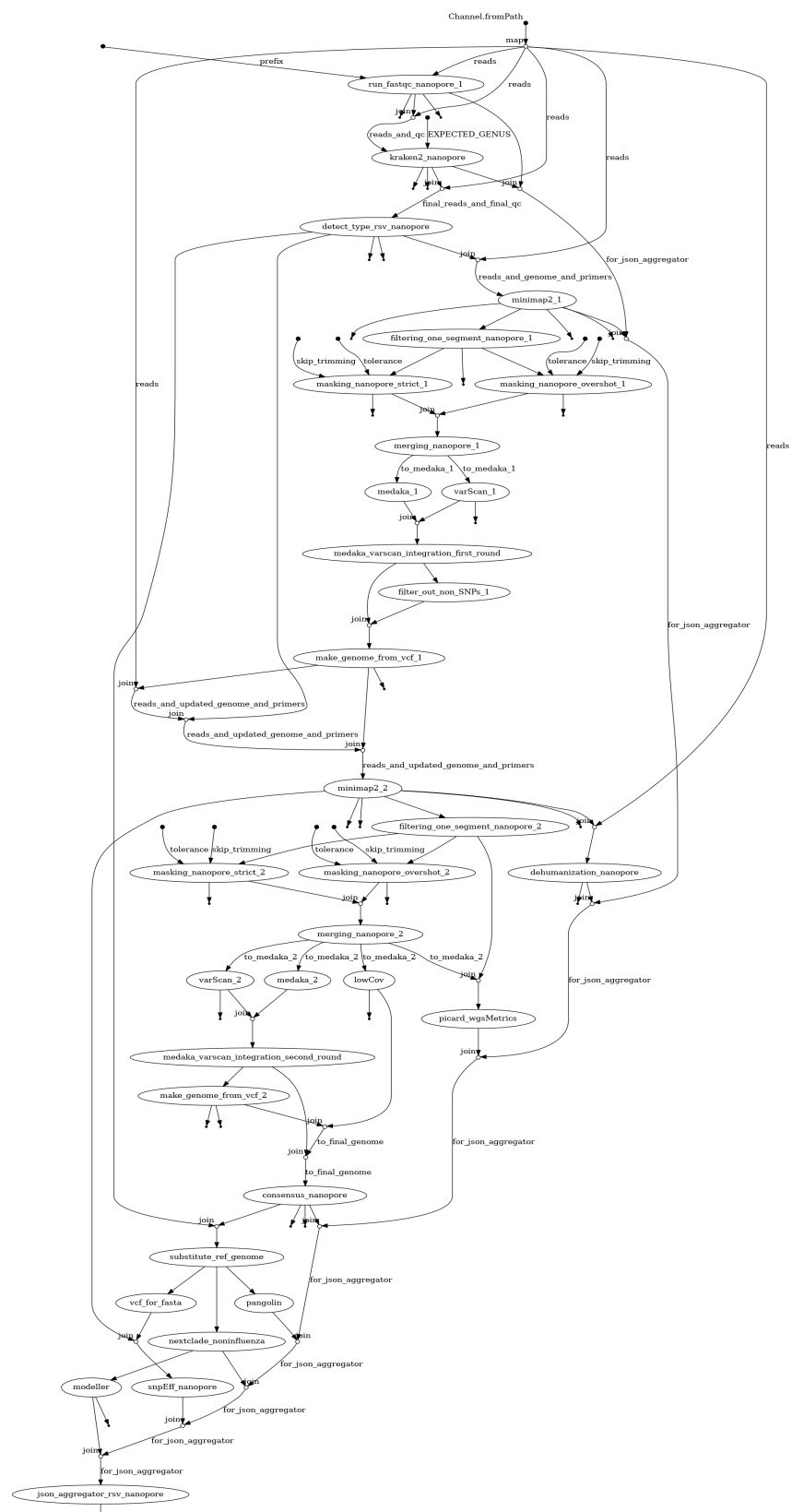




Nanopore Sars-CoV-2 Flowchart



Nanopore Influenza Flowchart



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 80.8 GiB GiB.

Database	Size
pangolin	90.2 MiB
nextclade	19.5 MiB
kraken	80.5 GiB
freyja	113.4 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 (["Updates procedure" in "External databases updates"](#)) you can safely skip it.

List of modules that require regular updates

- Nextclade
- Pangolin
- Kraken
- Freyja

Updating 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 `nextclade` 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>
(<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>
(<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
(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"  
    (...)
```

Updating Pangolin database

Pangolin (<https://github.com/cov-lineages/pangolin>) (Phylogenetic Assignment of Named Global Outbreak LINEages) 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
```

i 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
```


Updating 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_YYMMDD.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_external_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 apply 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 resolvable by shell into two different files. One for forward reads, and second for 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
TACTCTTTCCCTACACGACGCTCTTCCGATCT
>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 allele, 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 to 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

- trimmomatic - removing adapters
- bwa - mapping reads
- filtering - filtering bad quality reads
- masking - masking primers
- merging - intermediate step
- viterbi - improving alignment
- lowCov - detecting low coverage regions
- varScan - small INDEL caller
- freeBayes - small INDEL caller
- lofreq - small INDEL caller
- consensus - consensus sequence of three callers
- consensusMasking - masking low coverage regions

Quality Check section

- fastqc_1 - check raw reads
- fastqc_2 - check reads after trimmomatic

Contaminations section

- Kraken2 - detecting other reads from other organisms

Coinfections section

- coinfections_ivar - intermediate step
- coinfections_varscan - intermediate step
- coinfection_analysis - detecting coinfections
- freyja - detecting coinfections - alternative method

Dehumanization section

- dehumanization - removing non viral reads

Functional analysis

- vcfForFasta - intermediate step
- snpEff - detecting effects of mutations on protein sequence

SV Calling section

- picard - intermediate step
- manta - detecting large SVs

Lineages section

- pangolin - detecting pango line
- nextclade - detecting nextclade line

Model building section

- modeller - building Spike protein model

Module: fastqc

Description

The `fastqc` module performs quality control (QC) on paired-end sequencing reads. It generates statistical summaries and visualizations of read quality, length, and position-based metrics. It also evaluates if the input reads pass predefined QC thresholds, producing JSON files with detailed results and an overall QC status.

Input

- **sampleId**: Identifier for the sample being analyzed.
- **reads**: Path to paired-end sequencing read files (forward and reverse).
- **QC_STATUS**: Initial QC status from previous module (default: "tak").
- **prefix**: Prefix for output files.

i The **prefix** field is used to identify module calls. The `fastqc` module is invoked twice: before and after Trimmomatic, and the quality of the reads is assessed twice. The same module is imported twice. To allow Nextflow to distinguish between them, a prefix field is added with an arbitrary label: "pre-filtering" or "post-filtering".

Output

- CSV files with read quality metrics and histograms (e.g., read quality histogram, read length histogram, position-based quality plot).
- JSON files (`forward_<prefix>.json`, `reverse_<prefix>.json`) containing detailed QC results.
- Environment variable **QC_STATUS_EXIT** indicating the overall QC status ("tak", "nie", or "blad").

i The QC_STATUS_EXIT field acts as a quality control flag. It is passed to each subsequent module. If an error is detected at any stage that prevents further analysis, or if data quality issues are identified, the flag is set to "error" or "no." In such cases, subsequent modules will not perform analyses but will instead pass the results obtained so far to the result-aggregating module.

Modules: kraken2 (illumina and nanopore versions)

kraken2_illumina

Description

The `kraken2_illumina` module combines Kraken2 with a species identification process for Illumina reads. It evaluates contamination and updates the QC status based on the presence of reads from the expected genus.

Input

- **sampleId**: Sample identifier.
- **reads**: Path to paired-end sequencing read files.
- **QC_STATUS**: QC status from the upstream module.
- **EXPECTED_GENUS**: Expected genus for the sample.

Output

- JSON file (`contaminations.json`) with contamination results.
- **QC_status_contaminations**: Updated QC status as an environment variable.
- **FINAL_GENUS**: Most likely genus as an environment variable.

kraken2_nanopore

Description

The `kraken2_nanopore` module is similar to `kraken2_illumina` but processes single-end reads from Nanopore sequencing.

Input

- **sampleId**: Sample identifier.
- **reads**: Path to single-end sequencing read file.
- **QC_STATUS**: QC status from the upstream module.
- **EXPECTED_GENUS**: Expected genus for the sample.

Output

- JSON file (`contaminations.json`) with contamination results.
- **QC_status_contaminations**: Updated QC status as an environment variable.
- **FINAL_GENUS**: Most likely genus as an environment variable.

Module: trimmomatic

Description

This module processes raw sequencing reads by trimming adapters and low-quality bases. It uses the Trimmomatic tool to clean the input reads, producing paired and unpaired output files. If the quality control (QC) status is "no," the module creates dummy output files.

Input

- tuple val(sampleId), path(reads), val(QC_status)

Output

- tuple val(sampleId), path('*_paired.fastq.gz'), val(QC_status), emit: proper_reads_and_qc
- tuple val(sampleId), path('*_paired.fastq.gz'), emit: proper_reads
- tuple val(sampleId), path('_paired.fastq.gz'), path('_unpaired.fastq.gz'), val(QC_status), emit: all_reads

This module performs quality trimming on sequencing reads based on the QC status and produces cleaned paired and unpaired reads. It emits three types of output: proper reads with QC status, proper reads without QC status, and all reads (paired and unpaired).

Pipeline customization

All files required by the module for SNP/INDEL identification are located in the directory `data/generic` if the installation was conducted exactly as described in Quickstart ([Quickstart](#)). During the image building process, its contents are copied into the image. To use custom files, you need to appropriately modify the contents of the subdirectories "adapters", "contaminations", "genome", "modeller", "primers" and "vcf_template" before building the image.

Adapters file

The adapter sequences are located in the subdirectory `adapters`. In cases where the length of the insert (the DNA fragment to be sequenced) is shorter than the number of cycles in sequencing, it may happen that residues of adapter sequences are present at the 3' or 5' end next to the actual insert sequence. This unnecessary part of the read should be trimmed, which requires specifying the adapter sequences used during sequencing. Besides the adapters currently distributed with the trimmomatic program, you can create your own adapter sequence files provided they are in fasta format. Path to `adapters` dir is passed as a one of a parameter. The pipeline assumes the use of adapters placed in the file `TruSeq3-PE-2.fa`.

Indexed genome

The indexed genome of SARS-CoV-2 is located in the subdirectory `data/generic/genome/SarsCov2`. After building the image, files from this directory are available inside the container in the directory `/SARS-CoV2/genome/SarsCov2/`. To propose your own version of the genome, follow these steps:

1. Download the genome sequences in fasta format (for example, for the SARS-CoV-2 virus, the reference sequence is available on the website <https://www.ncbi.nlm.nih.gov/sars-cov-2/>). Save the downloaded file with the name `sarscov2.fasta` in any directory.
2. Index the genome using the program BWA. Navigate to the directory where you saved the `sarscov2.fasta` file and execute the following command. Assuming the `bwa` program is in the `$PATH`.

```
bwa index sarscov2.fasta
```

3. Index the genome using the program faidx. Navigate to the directory where you saved the sarscov2.fasta file and execute the following command. Assuming the samtools program is in the \$PATH.

```
samtools faidx sarscov2.fasta
```

4. Copy the contents of the directory with your own indexed genome to data/generic/genome/SarsCov2. This way, you overwrite the existing files there.
5. Build the image. The built image will only contain the new genome. The originally used version of the genome will not be available to the container.
6. Note that functional mutation effect predictions will only work if the genome sequence has the header MN908947.3. To change this, modify the Dockerfile in the section starting with "#SnpEFF". Add at the end of this section, after the line RUN java -jar snpEff.jar download MN908947.3, your own identical command but replacing the name MN908947.3 with the header used by your new genome. The snpEFF database is regularly updated, but it is possible that our sequence is not included in this database.
7. Your own genome also requires creating your own file with the location of primers if the genome has a different header than MN908947.3. Changing the genome sequence may require updating the location of primer sequences in the genome.

Primers

1. For the SARS-CoV-2 virus, two protocols based on amplicons are currently used: the dominant ARTIC protocol and the long-read Midnight protocol. Over time, new versions and modifications related to emerging virus variants appear. The primers available in May 2022 are located in the "primers" subdirectory and are accessible in the container at the path data/generic/primers. Each of these directories contains subdirectories V1, V2, V3, V4, V4.1 (primers used in subsequent versions of the ARTIC protocol), and V1200 (primers used in the Midnight protocol). Additionally, primers used in the EQA test from April 2023 are added (SARS1_partmerge_exp for sequencing from the "SARS1" test and SARS2_partmerge_exp for sequencing samples

in the "SARS2" test). In each of these directories, there are two files: `nCoV-2019.scheme.bed` and `pairs.tsv`. `nCoV-2019.scheme.bed` is a file containing information about the primer locations in the reference genome, and `pairs.tsv` contains information about which primer pairs flank each of the resulting amplicons. **When invoking the pipeline, the path to the selected bed file with primers must be provided. There is no "default" version.**

2. When creating custom .bed files, remember that (I) the primer positions in the bed file are indexed from 0, not from 1. (II) The ranges are half-open, meaning the start position is treated as the first position in the genome that contains the primer, and the END position is treated as the first position in the genome that the primer does not cover. Below is an example of how a properly formatted bed file should look:

```
MN908947.3 29 54 nCoV-2019_1_LEFT 1 +
MN908947.3 385 411 nCoV-2019_1_RIGHT 1 -
MN908947.3 319 342 nCoV-2019_2_LEFT 2 +
MN908947.3 322 333 nCoV-2019_2_LEFT_alt 2 +
```

3. The .bed file must contain the following columns separated by tabs:
 1. Reference genome name. In case of using a genome other than MN908947.3 (point B.2), always create a primer scheme from scratch.
 2. Start position for the primer
 3. End position for the primer
 4. Primer name. The name is built according to the scheme in which consecutive elements are separated by " ": *nCoV-2019* (amplikon number) (*from which side of the amplikon the primer is located, possible expressions are LEFT or RIGHT*) (optional field where we provide the expression 'alt' if a given amplikon has more than one primer flanking it from the same side). For example, `nCoV-2019_2_LEFT_alt` means it is the second primer flanking from the 5' side of amplikon number 2.
 5. Pool identifier from which the amplikon originates. It can appear as a single digit, e.g., "1" or "2", or as a unique text, e.g., `nCoV-2019_1` or `nCoV-2019_2`.
 6. The direction of the strand to which the primer hybridizes.

Publicly available files often have column 7 with the primer sequence, but it is not required. "Basic" and "alt" primers are NOT combined unless we understand the implications for further analysis. Primers can be combined if they are duplicates, meaning they have identical values in columns 2 and 3.

4. The pairs.tsv file should contain only two columns separated by tabs: i. Name of the primer flanking the amplicon from the 5' side. The name is identical to column 4 of the .bed file. ii. Name of the primer flanking the amplicon from the 3' side. The name is identical to column 4 of the .bed file. If there are more than one primer flanking the amplicon from the same side, provide the primer that generates the longer amplicon.
5. After creating both files, place them in a directory with a unique name inside `data/generic/primers`.
6. Primers used in the Midnight protocol (referred to as `V1200`) are available as an archive on the website <https://zenodo.org/record/3897530#.Xv5EFpMzadY> (<https://zenodo.org/record/3897530#.Xv5EFpMzadY>). These files were prepared based on the protocol described at <https://www.protocols.io/view/sars-cov2-genome-sequencing-protocol-1200bp-amplic-rm7vz8q64vx1/v6?step=20> (<https://www.protocols.io/view/sars-cov2-genome-sequencing-protocol-1200bp-amplic-rm7vz8q64vx1/v6?step=20>). Download these files by executing the following commands:

```
cd ${HOME}/my_primers/nCoV-2019
wget
https://zenodo.org/record/3897530/files/1200bp_amplicon_bed.tar.
gz
tar -zxf 1200bp_amplicon_bed.tar.gz
```

7. Primers used in the ARTIC scheme are available in the repository <https://github.com/artic-network/fieldbioinformatics.git> (<https://github.com/artic-network/fieldbioinformatics.git>)
8. Due to the operation of the primer masking program, it is recommended to extend the amplicon ranges by 1bp in the 3' and 5' directions. Practically, this means that in the primer file, the START field of the LEFT primer will be one less than implied by the sequence, and the END field of the RIGHT primer will be one more.

