

**PlEpiSeq Bioinformatic Pipelines Documentation**

The document provides technical documentation for the bioinformatics processing protocols (also known as pipelines) developed in the PlEpiSeq 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 Łaźniewski

Table of Contents

[Chapter 1. Conventions in this document 4](#__RefHeading___Toc110_3077261986)

[1.1. Paragraphs styles: 4](#__RefHeading___Toc112_3077261986)

[1.2. Character styles: 4](#__RefHeading___Toc114_3077261986)

[Chapter 2. Quick start 4](#__RefHeading___Toc106_3077261986)

[2.1. Installation and usage 4](#__RefHeading___Toc116_3077261986)

[Chapter 3. Hardware requirements 6](#__RefHeading___Toc119_3077261986)

[3.1. Platform 6](#__RefHeading___Toc121_3077261986)

[3.2. CPU 6](#__RefHeading___Toc123_3077261986)

[3.3. Memory 6](#__RefHeading___Toc125_3077261986)

[3.4. Storage requirements 6](#__RefHeading___Toc127_3077261986)

[Chapter 4. Software requirements 8](#__RefHeading___Toc154_3077261986)

[4.1. Compatibility 8](#__RefHeading___Toc156_3077261986)

[Chapter 5. Pipeline overview 8](#__RefHeading___Toc164_3077261986)

[5.1. Pipelines as DAGs 9](#__RefHeading___Toc166_3077261986)

[5.2. Flowcharts 9](#__RefHeading___Toc168_3077261986)

[Chapter 6. External Databases 16](#__RefHeading___Toc137_3077261986)

[6.1. Updates procedure 16](#__RefHeading___Toc182_3077261986)

[6.2. Updates internals 17](#__RefHeading___Toc839_1555702251)

[Chapter 7. Running pipeline 20](#__RefHeading___Toc851_1555702251)

[7.1. Pipeline parameters 20](#__RefHeading___Toc853_1555702251)

[Chapter 8. Modules 23](#__RefHeading___Toc864_1555702251)

[8.1. Module alphafold.nf 23](#__RefHeading___Toc2740_900569203)

[8.2. Module bwa.nf 23](#__RefHeading___Toc2742_900569203)

[8.3. Module coinfection\_analysis.nf 23](#__RefHeading___Toc2744_900569203)

[8.4. Module coinfection\_ivar.nf 23](#__RefHeading___Toc2746_900569203)

[8.5. Module coinfection\_varscan.nf 23](#__RefHeading___Toc2748_900569203)

[8.6. Module consensus.nf 23](#__RefHeading___Toc2750_900569203)

[8.7. Module copy\_genome\_and\_primers.nf 24](#__RefHeading___Toc2752_900569203)

[8.8. Module dehumanization.nf 24](#__RefHeading___Toc2754_900569203)

[8.9. Module detect\_subtype\_nanopore.nf 24](#__RefHeading___Toc2756_900569203)

[8.10. Module detect\_subtype.nf 24](#__RefHeading___Toc2758_900569203)

[8.11. Module detect\_type.nf 25](#__RefHeading___Toc2760_900569203)

[8.12. Module fastqc.nf 26](#__RefHeading___Toc866_1555702251)

[8.13. Module filtering\_multiple\_segments.nf 28](#__RefHeading___Toc2762_900569203)

[8.14. Module filtering\_one\_segment.nf 28](#__RefHeading___Toc2764_900569203)

[8.15. Module filter\_out\_non\_SNPs.nf 28](#__RefHeading___Toc2766_900569203)

[8.16. Module freeBayes.nf 28](#__RefHeading___Toc2768_900569203)

[8.17. Module freyja.nf 28](#__RefHeading___Toc2770_900569203)

[8.18. Module indelQual.nf 28](#__RefHeading___Toc2772_900569203)

[8.19. Module integrate\_medaka\_and\_varscan.nf 28](#__RefHeading___Toc2774_900569203)

[8.20. Module json\_aggregator.nf 28](#__RefHeading___Toc2776_900569203)

[8.21. Module kraken2.nf 28](#__RefHeading___Toc2778_900569203)

[8.22. Module lofreq.nf 29](#__RefHeading___Toc2786_900569203)

[8.23. Module lowCov.nf 29](#__RefHeading___Toc2788_900569203)

[8.24. Module make\_genome\_from\_vcf.nf 29](#__RefHeading___Toc2790_900569203)

[8.25. Module manta.nf 29](#__RefHeading___Toc2792_900569203)

[8.26. Module masking.nf 29](#__RefHeading___Toc2794_900569203)

[8.27. Module medaka.nf 29](#__RefHeading___Toc2796_900569203)

[8.28. Module merging\_nanopore.nf 29](#__RefHeading___Toc2798_900569203)

[8.29. Module merging\_one\_segment\_filtering.nf 29](#__RefHeading___Toc2800_900569203)

[8.30. Module minimap2.nf 29](#__RefHeading___Toc2802_900569203)

[8.31. Module nextalign.nf 29](#__RefHeading___Toc2804_900569203)

[8.32. Module nextclade.nf 29](#__RefHeading___Toc2806_900569203)

[8.33. Module pangolin.nf 29](#__RefHeading___Toc2808_900569203)

[8.34. Module picard.nf 29](#__RefHeading___Toc2810_900569203)

[8.35. Module picard\_wgsMetrics.nf 29](#__RefHeading___Toc2812_900569203)

[8.36. Module reassortment.nf 29](#__RefHeading___Toc2814_900569203)

[8.37. Module resistance.nf 29](#__RefHeading___Toc2816_900569203)

[8.38. Module snpEff.nf 29](#__RefHeading___Toc2818_900569203)

[8.39. Module sort\_and\_index.nf 29](#__RefHeading___Toc2820_900569203)

[8.40. Module substitute\_ref.nf 29](#__RefHeading___Toc2822_900569203)

[8.41. Module trimmomatic.nf 30](#__RefHeading___Toc2824_900569203)

[8.42. Module varscan.nf 31](#__RefHeading___Toc2826_900569203)

[8.43. Module vcf\_for\_fasta.nf 31](#__RefHeading___Toc2828_900569203)

[Chapter 9. JSON output 32](#__RefHeading___Toc2830_900569203)

# Conventions in this document

Following styles names are used in this document:

## Paragraphs styles:

Normal paragraphs are written with Text Body

Code is presented in Code Box

## Character styles:

Paths are denoted like here /usr/bin/bash with path style

Inline Code is inserted with code character style like this, --no-cache and it's pretty nice.

# Quick start

## Installation and usage

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

curl -s https://get.nextflow.io | bash

1. Put the Nextflow binary in the PATH. E.g.:

sudo mv nextflow /usr/local/bin

1. Clone the repository:

git clone --depth 1 https://github.com/mkadlof/pzh\_pipeline\_viral

1. Enter downloaded repository:

cd pzh\_pipeline\_viral

1. 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>.

1. 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)"

1. Download latest version of external databases. In project root dir run:

./update\_external\_databases.sh --database pangolin --output-path

/path/to/external\_databases

./update\_external\_databases.sh --database nextclade --output-path

/path/to/external\_databases

./update\_external\_databases.sh --database kraken2 --output-path

/path/to/external\_databases --kraken-type standard\_08gb

./update\_external\_databases.sh --database freyja --output-path

/path/to/external\_databases

kraken2 require providing database name: One of following: "standard" "standard\_08gb" "standard\_16gb" "viral" "minusb" "pluspf" "pluspf\_08gb" "pluspf\_16gb" "pluspfp" "pluspfp\_08gb" "pluspfp\_16gb" "nt" "eupathdb48"

1. For more details read the chapter External databases updates ([External databases updates](#_toc343)).
2. Copy run\_nf\_pipeline.sh.template to run\_nf\_pipeline.sh and fill in the paths to the reads and output directory.
3. 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>

## Storage requirements

Total storage requirements is **~80 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 file system 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-4.1-main | 2.01 GiB |
| nf\_illumina\_sars-4.1-updater | 258 MiB |
| nf\_illumina\_sars-4.1-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 Sizes

|  |  |
| --- | --- |
| **Database** | **Size** |
| pangolin | 90.2 MiB |
| nextclade | 19.5 MiB |
| kraken2 | 80.5 GiB |
| freyja | 113.4 MiB |
| **Total** | **80.8 GiB** |

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 |  |  |

### 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

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

### Nextflow:

24.04.4.5917

# 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 down-sampling 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:

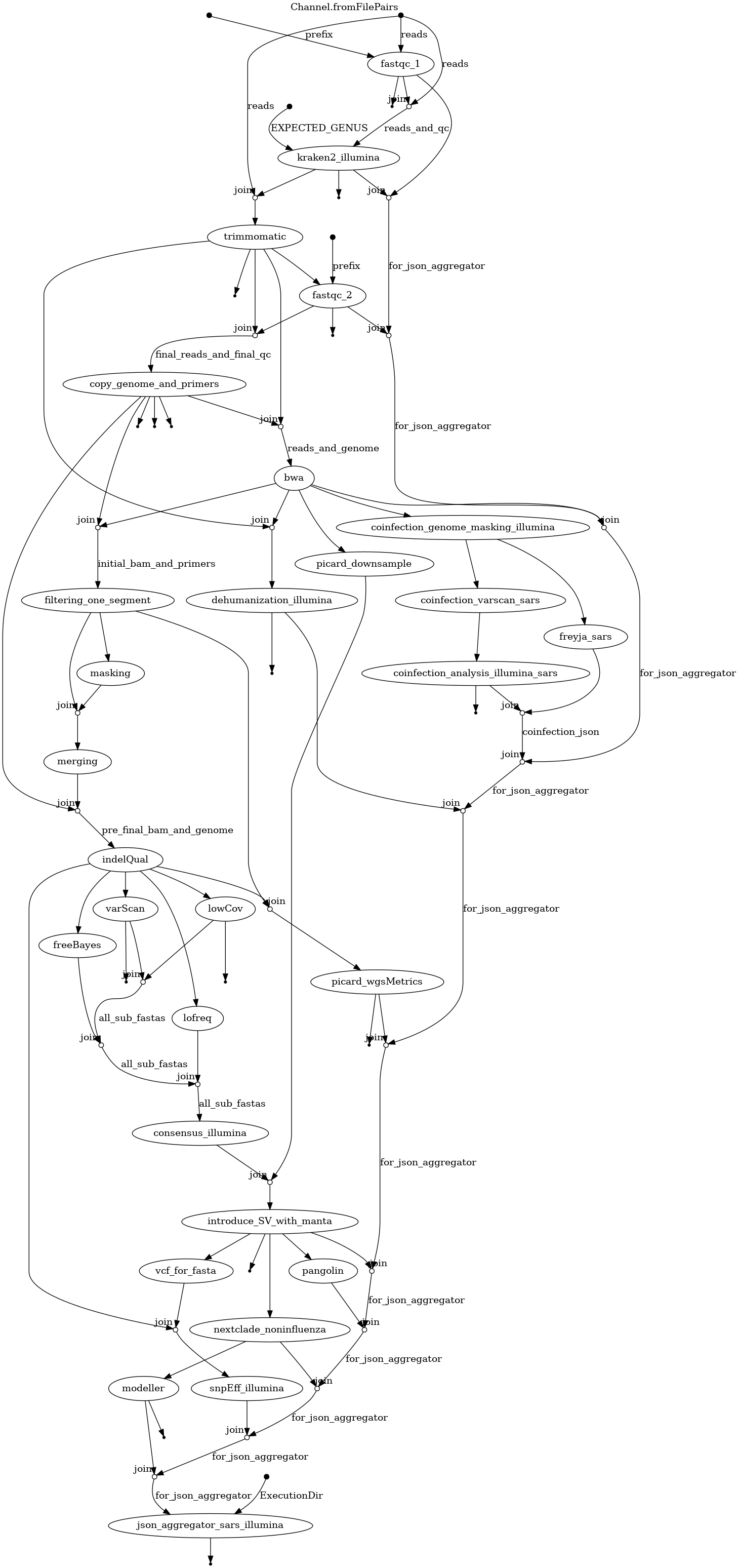
* Module settings (meta-information, as well as the method of execution and result return)
* Definition of input channels
* Definition of output channels
* Script executing the task (process)

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 pipelineów w formie DAG'ów. Na przedstawionych grafikach owale reprezentują procesy, zaś linie je łączące kanały przepływu danych.

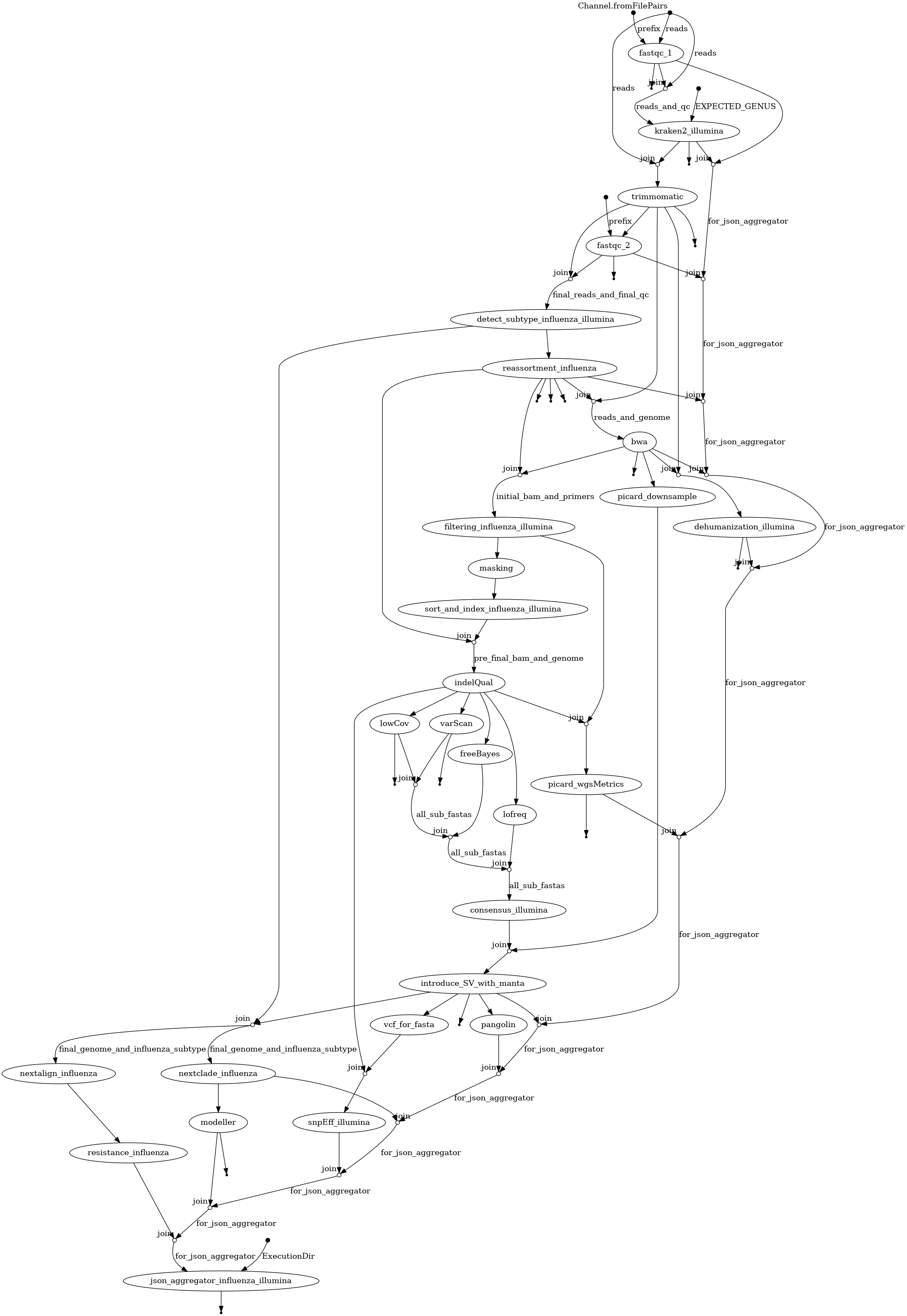
## Flowcharts

Following section contain all pipelines depicted as DAG graphs.

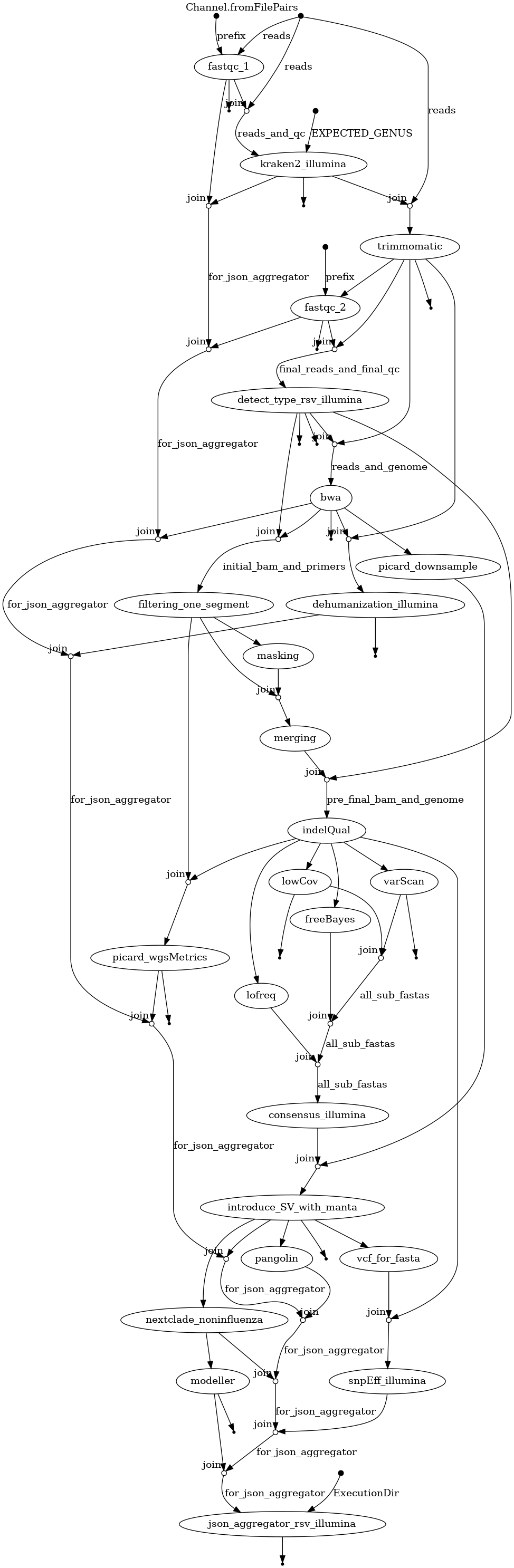
### Illumina Sars-CoV-2 Flowchart



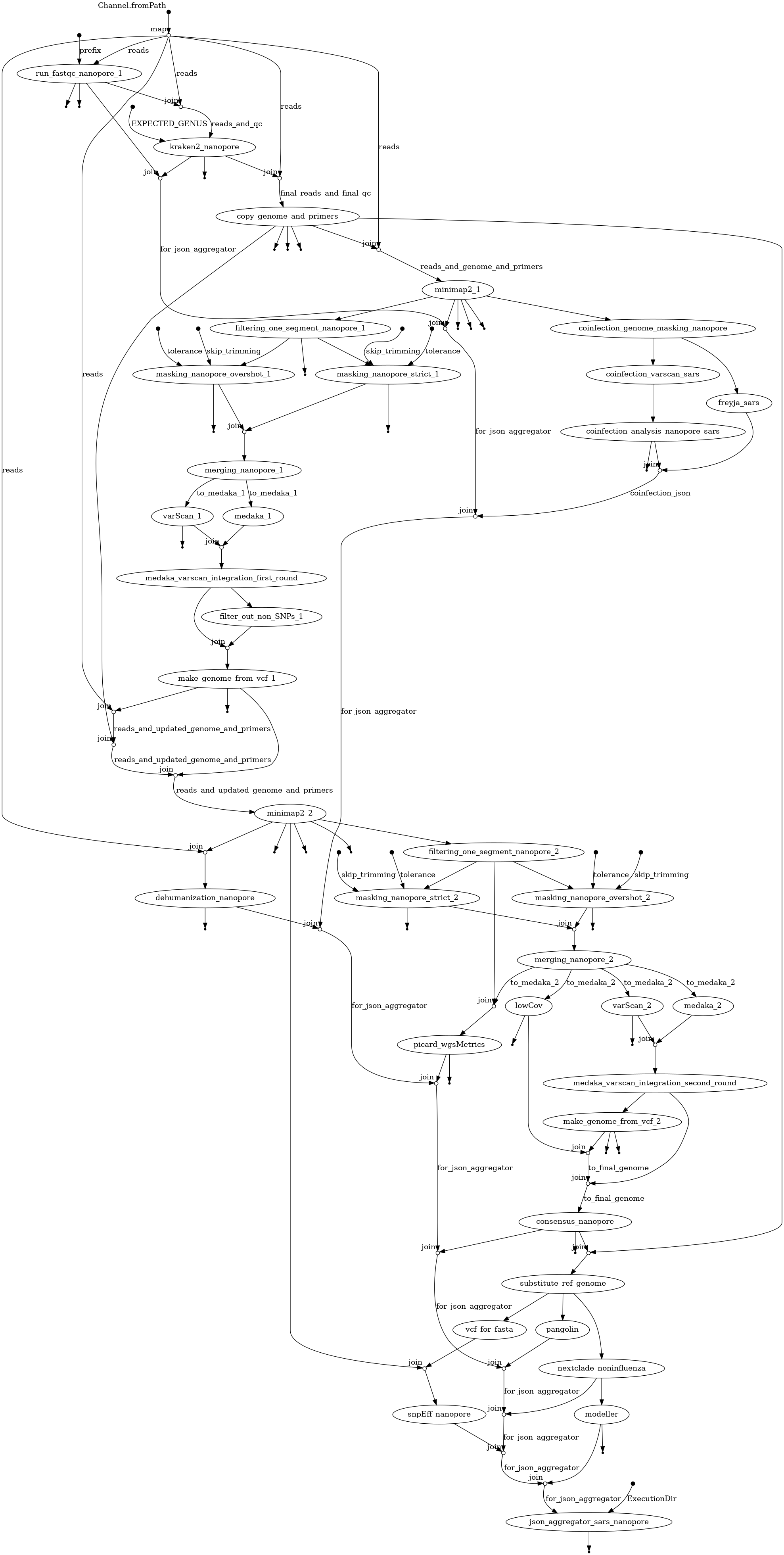
### Illumina Influenza Flowchart



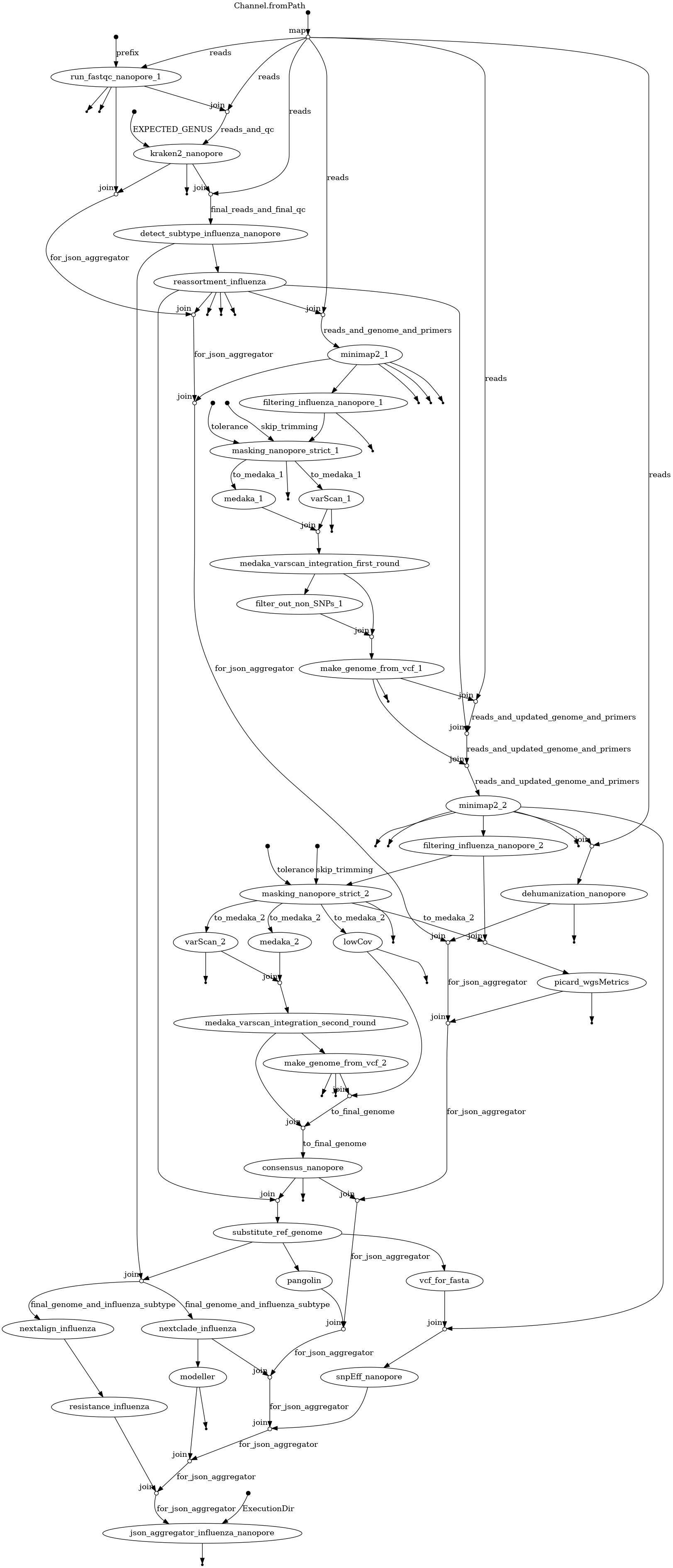
### Illumina RSV Flowchart



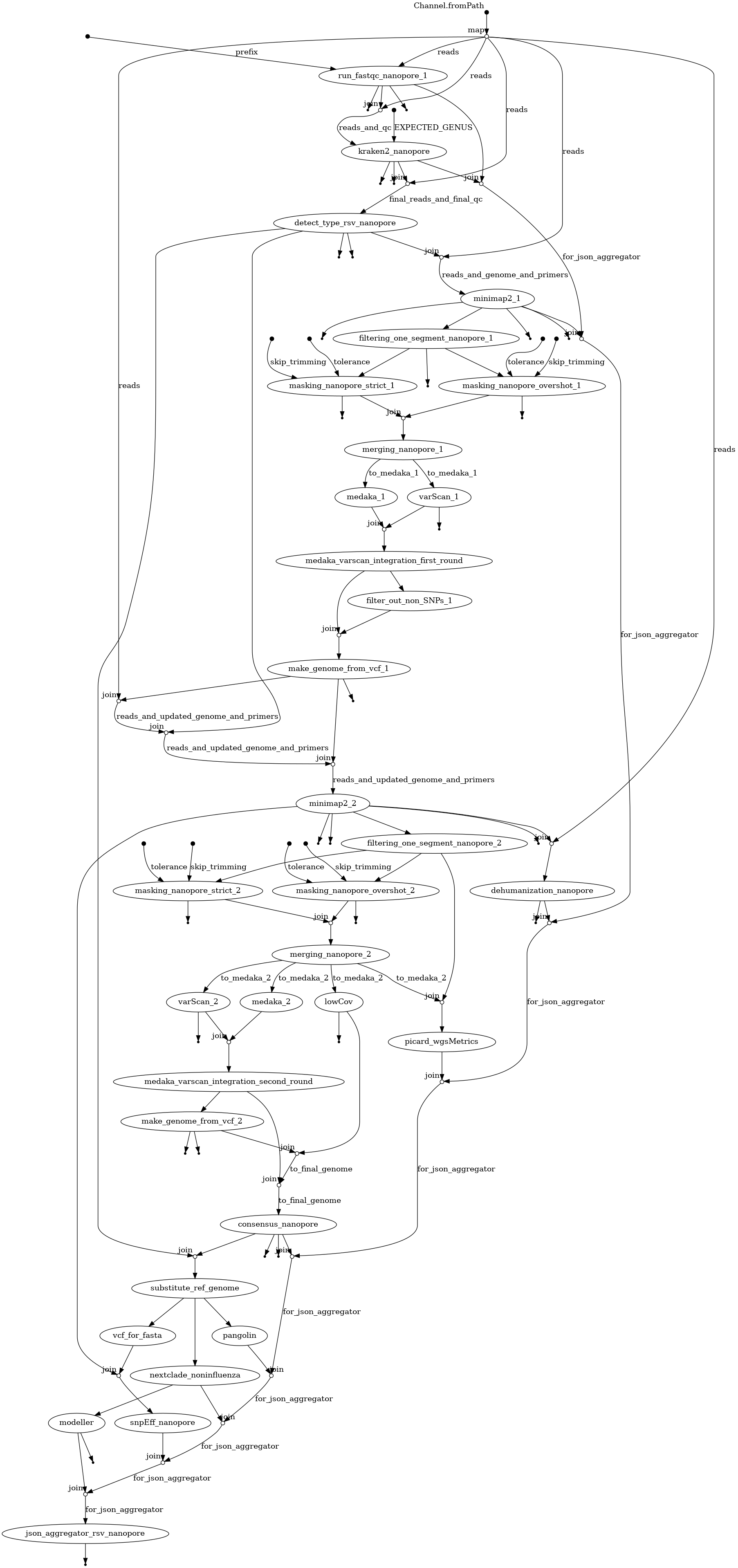
### Nanopore Sars-CoV-2 Flowchart



### Nanopore Influenza Flowchart



### Nanopore RSV Flowchart



# External Databases

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-4.1-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.

|  |  |
| --- | --- |
| **Database** | **Size** |
| pangolin | 90.2 MiB |
| nextclade | 19.5 MiB |
| kraken2 | 80.5 GiB |
| freyja | 113.4 MiB |
| **Total** | **80.8 GiB** |

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 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"

(...)

### 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

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 80 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](https://benlangmead.github.io/aws-indexes/k2) 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-4.1-updater container. For running this script simply pass kraken2 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.

### Updating 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 bar codes. 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 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\_LEFTnCoV-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 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-4.1-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.

# Modules

## Module alphafold.nf

Builds protein models*.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Platform** | **Illumina** | | | **Nanopore** |
| **Species** | SARS INFL RSV | | | SARS INFL RSV |
| **QC**  **switch** | NO | **QC**  **Criteria** |  | |
| **JSON output** | YES | **JSON key** | "structural\_data": {} | |

Proteins:

* SARS-Cov-2: Spike
* Influenza: HA and NA
* RSV: F and G

## Module bwa.nf

Module align paired-end reads to a reference genome using BWA. The alignment is filtered and converted to BAM format with samtools view, sorted with samtools sort, and indexed with samtools index. This process generates an indexed BAM file containing properly aligned and sorted reads.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Platform** | **Illumina** | | | **Nanopore** |
| **Species** | SARS INFL RSV | | | SARS INFL RSV |
| **QC**  **switch** | YES | **QC**  **Criteria** | * number of reads is less than threshold defined in ${params.min\_number\_of\_reads} | |
| **JSON output** |  | **JSON key** | * "viral\_mapping\_data": {} | |

bwa index ${reference\_fasta}

bwa mem -t ${params.threads} -T 30 ${reference\_fasta} ${reads[0]} ${reads[1]} | \

samtools view -@ ${params.threads} -Sb -f 3 -F 2048 - | \

samtools sort -@ ${params.threads} -o mapped\_reads.bam -

samtools index mapped\_reads.bam

## Module coinfection\_analysis.nf

The coinfection analysis module analyzes allele distribution and compares it with known coinfected samples using the Smirnoff-Kolmogorov test.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Platform** | **Illumina** | | | **Nanopore** |
| **Species** | SARS INFL RSV | | | SARS INFL RSV |
| **QC**  **switch** | NO | **QC**  **Criteria** |  | |
| **JSON output** | YES | **JSON key** | "sars\_data": {} | |

The coinfection analysis is based on similarity to 3 known samples from EQA2023. It is performed by custom script: predict\_coinfection\_illumina.py. The new version of the script allows specifying any number of samples that are known to be coinfected. Simply add additional files as positional arguments.

## Module coinfection\_ivar.nf

Preparatory module whose task is to trim primer sequences from the sequences.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Platform** | **Illumina** | | | **Nanopore** |
| **Species** | SARS INFL RSV | | | SARS INFL RSV |
| **QC**  **switch** | NO | **QC**  **Criteria** |  | |
| **JSON output** | NO | **JSON key** |  | |

ivar trim -i ${mapped\_reads} \

-b ${primers} \

-m ${params.length} \

-q ${params.quality\_initial} \

-e \

-p for\_contamination

(...)  
samtools mpileup --max-depth 10000 \

--fasta-ref ${ref\_genome\_with\_index[final\_index]} \

--min-BQ ${params.quality\_snp} \

for\_contamination\_sorted.bam >> for\_contamination.mpileup

## Module coinfection\_varscan.nf

Result of Module coinfection\_ivar.nf is passed to the VarScan program, which identifies mutations in unfiltered samples.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Platform** | **Illumina** | | | **Nanopore** |
| **Species** | SARS INFL RSV | | | SARS INFL RSV |
| **QC**  **switch** |  | **QC**  **Criteria** |  | |
| **JSON output** | NO | **JSON key** |  | |

varscan\_qual=`echo "${params.quality\_snp} - 1" | bc -l`

java -jar /opt/varscan/VarScan.v2.4.6.jar pileup2snp ${for\_contamination\_mpileup} \

--min-avg-qual \${varscan\_qual} \

--p-value 0.9 \

--min-var-freq 0.05 \

--min-coverage 20 \

--variants \

--min-reads2 0 > detected\_variants\_varscan\_contamination.txt

## Module consensus.nf

Consensus module predicts consensus sequence based on mutations identified by SNP Callers (Module freeBayes.nf, Module lofreq.nf, Module varscan.nf) and masks the sequence.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Platform** | **Illumina** | | | **Nanopore** |
| **Species** | SARS INFL RSV | | | SARS INFL RSV |
| **QC**  **switch** | NO | **QC**  **Criteria** |  | |
| **JSON output** | NO | **JSON key** | "genome\_files\_data": {}  "viral\_genome\_data": {}  (only for Nanopore platform - corresponding keys for illumina are returned by manta) | |

There are two custom scripts for this: one for the Illumina platform and another for the Nanopore platform.

* make\_consensus.py for illumina
* make\_consensus\_nanopore.py for nanopore

## Module copy\_genome\_and\_primers.nf

Helper modules that pass genome from within container to the nextflow work dir. There is no difference between lllumina and Nanopore

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Platform** | **Illumina** | | | **Nanopore** |
| **Species** | SARS INFL RSV | | | SARS INFL RSV |
| **QC**  **switch** | NO | **QC**  **Criteria** |  | |
| **JSON output** | NO | **JSON key** |  | |

## Module dehumanization.nf

This module removes reads that did not map to the expected organism species.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Platform** | **Illumina** | | | **Nanopore** |
| **Species** | SARS INFL RSV | | | SARS INFL RSV |
| **QC**  **switch** | NO | **QC**  **Criteria** |  | |
| **JSON output** | YES | **JSON key** | "dehumanized\_data": {} | |

The module uses samtools for identifying mapped reads. Then, using the seqkit program, FASTQ files are filtered with the list of identifiers generated in the first step.

## Module detect\_subtype\_nanopore.nf

*Short description*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Platform** | **Illumina** | | | **Nanopore** |
| **Species** | SARS INFL RSV | | | SARS INFL RSV |
| **QC**  **switch** |  | **QC**  **Criteria** |  | |
| **JSON output** | NO | **JSON key** |  | |

## Module detect\_subtype.nf

This module detects the INFL subtype by mapping reads to predefined genomes. We record the scores for each segment (creating a genome vs. segments matrix) which is then passed to the reassortment detection module, where hybrid\_genome will be constructed.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Platform** | **Illumina** | | | **Nanopore** |
| **Species** | INFL | | | INFL |
| **QC**  **switch** | NO | **QC**  **Criteria** | To be added (vide issue #18). | |
| **JSON output** | NO | **JSON key** |  | |

## Module detect\_type.nf

This module detects the RSV type by mapping reads to predefined genomes. We select the genome with the highest total mapping score.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Platform** | **Illumina** | | | **Nanopore** |
| **Species** | RSV | | | RSV |
| **QC**  **switch** | YES | **QC**  **Criteria** | * Number of reads for A and B is less than predefined threshold (vide [issue #19](https://github.com/mkadlof/pzh_pipeline_viral/issues/19)) | |
| **JSON output** | YES | **JSON key** | "rsv\_data": {} | |

## Module fastqc.nf

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.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Platform** | **Illumina** | | | **Nanopore** |
| **Species** | SARS INFL RSV | | | SARS INFL RSV |
| **QC**  **switch** | YES | **QC**  **Criteria** | * criterium 1 * criterium 2 * criterium 3 | |
| **JSON output** | YES | **JSON key** | {"output": "sequencing\_summary\_data": [...]} | |

### 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.

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").

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.

### Description

Ten moduł jest wrapperem dla skryptu python

bin/common/run\_fastqc\_and\_generate\_json.py uruchamianym w linijkach:

DANE\_FORWARD=(`run\_fastqc\_and\_generate\_json.py -i ${reads[0]} -m

${params.memory} -c ${params.threads} -x

${params.min\_number\_of\_reads} -y ${params.min\_median\_quality} -s

${QC\_STATUS} -r "\${ERROR\_MSG}" -e ${prefix} -p

"${params.results\_dir}/${sampleId}/QC" -o forward\_${prefix}.json`)

(...)

DANE\_REVERSE=(`run\_fastqc\_and\_generate\_json.py -i ${reads[1]} -m

${params.memory} -c ${params.threads} -x

${params.min\_number\_of\_reads} -y ${params.min\_median\_quality} -s

${QC\_STATUS} -r "\${ERROR\_MSG}" -e ${prefix} -p

"${params.results\_dir}/${sampleId}/QC" -o reverse\_${prefix}.json`)

Opcje skryptu run\_fastqc\_and\_generate\_json.py:

Usage: run\_fastqc\_and\_generate\_json.py [OPTIONS]

Options:

-i, --input\_file PATH [INPUT] a path to a file in fastq format

-m, --memory INTEGER [INPUT] Memory available to fastqc program

-c, --cpu INTEGER [INPUT] CPUs available to fastqc program

-x, --min\_number INTEGER [INPUT] QC parameter if sample has less reads,

the json will contain blad

-y, --min\_qual INTEGER [INPUT] QC parameter if reads have median quality

less than this value the json will contain blad

-s, --status [tak|nie|blad]

[INPUT] PREDEFINED status that is transferred to

an output json. If this status was either

nie or blad fastqc will not run

[required]

-e, --stage [pre-filtering|post-filtering]

[INPUT] Stage on which data is analyzed

-p, --publishdir TEXT [INPUT] Path with fNEXTFLOW output, required to

correctly format json [required]

-r, --error TEXT [INPUT] PREDEFINED error message that is put in

json. Only used when status was set to

nie or blad

-o, --output TEXT [Output] Name of a file with json output

--help Show this message and exit.

## Module filtering\_multiple\_segments.nf

Coverage equalization for INFL.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Platform** | **Illumina** | | | **Nanopore** |
| **Species** | INFL | | | INFL |
| **QC**  **switch** |  | **QC**  **Criteria** |  | |
| **JSON output** | YES | **JSON key** | {viral\_genome\_data: {primer\_usage\_data:[]}} | |

simple\_filter\_illumina\_INFL.py

simple\_filter\_nanopore\_INFL\_ekstralayer\_EQA2024.py

## Module filtering\_one\_segment.nf

Coverage equalization for single segment organisms (SARS-CoV-2 and RSV)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Platform** | **Illumina** | | | **Nanopore** |
| **Species** | SARS RSV | | | SARS RSV |
| **QC**  **switch** |  | **QC**  **Criteria** |  | |
| **JSON output** | YES | **JSON key** | {viral\_genome\_data: {primer\_usage\_data:[]}} | |

## Module filter\_out\_non\_SNPs.nf

A module that filters out mutations from a VCF file that alter the genome length while retaining those that do not change the genome length. In Nanopore sequencing, genome generation occurs in two stages. The first is a rough draft, where only mutations that do not alter the genome length are considered.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Platform** | **Illumina** | | | **Nanopore** |
| **Species** |  | | | SARS INFL RSV |
| **QC**  **switch** | NO | **QC**  **Criteria** |  | |
| **JSON output** | NO | **JSON key** |  | |

## Module freeBayes.nf

*Short description*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Platform** | **Illumina** | | | **Nanopore** |
| **Species** | SARS INFL RSV | | | SARS INFL RSV |
| **QC**  **switch** |  | **QC**  **Criteria** |  | |
| **JSON output** | NO | **JSON key** |  | |

freebayes --limit-coverage ${params.max\_depth} \

--min-coverage ${params.min\_cov} \

--min-mapping-quality 20 \

--min-base-quality ${params.quality\_snp} \

--use-mapping-quality \

--fasta-reference ${reference\_fasta} \

--ploidy 1 \

${bam} > detected\_variants\_freebayes.vcf

Calling the freebayes program. The options used are:

* --limit-coverage ${max\_depth} – limit coverage at a position to this value when identifying variants. Please note that quasi-downsampling used in Step 4 for read filtering does not guarantee that a given position will not exceed the value specified in the max\_depth variable.
* --min-coverage ${min\_cov} – minimum coverage for a position to be considered as a variant.
* -m 20 – alignment quality of a read for its nucleotides to be considered in variant analysis.
* -q ${quality\_SNP} – minimum nucleotide quality at a position in a read to be considered in variant counting.
* -p 1 – ploidy of the analyzed organism.
* -f ${input\_genome} – path to the reference genome.
* -j – use mapping qualities when calculating variant likelihood.

cat detected\_variants\_freebayes.vcf | \

bcftools norm --check-ref w \

--rm-dup all \

--fasta-ref ${reference\_fasta} | \

bcftools norm --check-ref w \

--multiallelics -indels \

--fasta-ref ${reference\_fasta} > detected\_variants\_freebayes\_fix.vcf

Normalization of the VCF file similar to Module varscan.nf.

bcftools filter --include "QUAL >= \${qual} & INFO/DP >= ${params.min\_cov} & (SAF + SAR)/(SRF + SRR + SAF + SAR) > ${params.upper\_ambig} " \

detected\_variants\_freebayes\_fix.vcf > detected\_variants\_freebayes\_fix\_high.vcf

Freebayes cannot identify ambiguous positions, and at the position corresponding to the alternative allele in the VCF file, it always introduces a symbol: A, T, G, or C. Introducing the allele ambiguous symbol at such a position must be done manually. In the first step, we save to the detected\_variants\_freebayes\_fix\_high.vcf file all mutations in which the frequency of using the alternative allele exceeds the value provided in the ${upper\_ambig} variable and whose quality is greater than 15 (meaning p-value is less than 0.03).

bcftools filter --include "QUAL >= \${qual} & INFO/DP >= ${params.min\_cov} & (SAF + SAR)/(SRF + SRR + SAF + SAR) >= ${params.lower\_ambig} & (SAF + SAR)/(SRF + SRR + SAF + SAR) <= ${params.upper\_ambig} " \

detected\_variants\_freebayes\_fix.vcf > tmp\_low.vcf

introduce\_amb\_2\_vcf.py tmp\_low.vcf \

detected\_variants\_freebayes\_fix\_ambig.vcf

Next, we extract mutations with allele alternative frequency between the values specified in the variables `$lower\_ambig` and `$upper\_ambig`. For these positions, we apply a simple Python script where instead of introducing the alt allele symbol, we introduce the ambiguous nucleotide symbol.

bcftools concat detected\_variants\_freebayes\_fix\_high.vcf.gz \

detected\_variants\_freebayes\_fix\_ambig.vcf.gz | \

bcftools sort --output-type z > detected\_variants\_freebayes\_final.vcf.gz

## Module freyja.nf

Freyja is an alternative tool to detect coinfections in SARS-CoV-2 samples.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Platform** | **Illumina** | | | **Nanopore** |
| **Species** | SARS | | | SARS |
| **QC**  **switch** | NO | **QC**  **Criteria** |  | |
| **JSON output** | YES | **JSON key** | "sars\_data": {} | |

mkdir variants\_files depth\_files demix\_files

freyja variants mapped\_reads.bam --variants variants\_files/test.variants.tsv --depths depth\_files/test.depth --ref ${reference\_fasta}

freyja demix variants\_files/test.variants.tsv depth\_files/test.depth --output demix\_files/test.output --confirmedonly --barcodes /home/external\_databases/freyja/usher\_barcodes.csv

freyja aggregate demix\_files/ --output coinfections.tsv

*The method uses lineage-determining mutational “barcodes” derived from the UShER global phylogenetic tree as a basis set to solve the constrained (unit sum, non-negative) de-mixing problem.*

Documentation - <https://andersen-lab.github.io/Freyja/index.html>

## Module indelQual.nf

Improves mapping scores in a BAM file around indels using the Viterbi algorithm.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Platform** | **Illumina** | | | **Nanopore** |
| **Species** | SARS INFL RSV | | | SARS INFL RSV |
| **QC**  **switch** | NO | **QC**  **Criteria** |  | |
| **JSON output** | NO | **JSON key** |  | |

lofreq indelqual --ref ${reference\_fasta} \

--out forvariants.bam \

--dindel clean\_sort\_dedup\_trimmed\_sort\_viterbi.bam

samtools sort -@ ${params.threads} \

-o forvariants.bam \

forvariants.bam

samtools index forvariants.bam

This fragment is part of the procedure recommended for identifying reads using the Lofreq program following GATK recommendations. Reads are realigned and a quality score is introduced for indel quality, enabling their identification by the Lofreq program. The above procedure does not affect the results of Varscan or Freebayes.

## Module integrate\_medaka\_and\_varscan.nf

This module integrates medaka vcf and SPECIFIC parts of varscan prediction i.e. SNPs with over 80% usage (according to varscan) that are not predicted as valid SNPs according to medaka due to complex underlying genotype of a region

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Platform** | **Illumina** | | | **Nanopore** |
| **Species** |  | | | SARS INFL RSV |
| **QC**  **switch** | NO | **QC**  **Criteria** |  | |
| **JSON output** | NO | **JSON key** |  | |

Operates in two rounds - first for rough draft, and then for final genome.

## Module json\_aggregator.nf

Main module aggregating partial JSON files into a final output file.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Platform** | **Illumina** | | | **Nanopore** |
| **Species** | SARS INFL RSV | | | SARS INFL RSV |
| **QC**  **switch** | NO | **QC**  **Criteria** |  | |
| **JSON output** | YES | **JSON key** | * "timestamp": str * "output": {} - main output file | |

It contain six submodules one-per each pipeline. Apart from aggregating partial results, it also fills in several metadata fields.

## Module kraken2.nf

The kraken2 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.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Platform** | **Illumina** | | | **Nanopore** |
| **Species** | SARS INFL RSV | | | SARS INFL RSV |
| **QC**  **switch** | YES | **QC**  **Criteria** | * criterium 1 * criterium 2 * criterium 3 | |
| **JSON output** | YES | **JSON key** | {"output": "sequencing\_summary\_data": [...]} | |

kraken2.nf file contain two modules: kraken2\_illumina and kraken2\_nanopore modules. They vary with inputs.

### 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.

## Module lofreq.nf

One of three SNP callers that uses a probabilistic model to detect low-frequency variants from high-throughput sequencing data, optimizing for sensitivity in the presence of sequencing errors.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Platform** | **Illumina** | | | **Nanopore** |
| **Species** | SARS INFL RSV | | | SARS INFL RSV |
| **QC**  **switch** |  | **QC**  **Criteria** |  | |
| **JSON output** | NO | **JSON key** |  | |

lofreq call-parallel --pp-threads ${params.threads} \

--ref ${reference\_fasta} \

--max-depth ${params.max\_depth} \

--min-cov ${params.min\_cov} \

--call-indels \

--out detected\_variants\_lofreq.vcf \

${bam}

* --pp-threads – number of CPU threads used for computations
* -f – path to the reference genome
* --max-depth – cutoff for the maximum coverage considered for each position
* -C – minimum coverage required for a position to be considered carrying a variant
* --call-indels – also identify INDELs

In the case of the lofreq program, we do not use parameters related to nucleotide quality in the read and mapping quality to the reference genome. The issue with counting correct coverage in such situations by the lofreq program is described in the following GitHub issue: [https://github.com/CSB5/lofreq/issues/80](./uses%20a%20probabilistic%20model%20to%20detect%20low-frequency%20variants%20from%20high-throughput%20sequencing%20data,%20optimizing%20for%20sensitivity%20in%20the%20presence%20of%20sequencing%20errors.). To calculate the proportions of the reference and alternative alleles, we use the DP4 field from the VCF file. It should be noted that it does not always sum up to the value in the DP field. For example, position 19,985 for Sample 10 from the EQA test. This is a region where the mutation is just behind a deletion, and moreover, the sequence being deleted is a repeated element in a palindrome. Additionally, this field is erroneous in the case of INDELs, so they continue to be analyzed if they meet the coverage, quality, and reference allele frequency criteria defined by the variable `$lower\_ambig`.

Then, the steps are identical to those for the freebayes program. The output\_detected\_variants\_lofreq.vcf file is split into two: one with mutations with high usage of the alternative allele, and the other with similar usage of the reference and alternative alleles. The files are appropriately filtered, merged, and used to identify the final list of mutations. Intermediate files leading to the creation of a fasta file with the sample genome are created as with the other programs.

## Module lowCov.nf

Predicts regions with low coverage.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Platform** | **Illumina** | | | **Nanopore** |
| **Species** | SARS INFL RSV | | | SARS INFL RSV |
| **QC**  **switch** | NO | **QC**  **Criteria** |  | |
| **JSON output** | NO | **JSON key** |  | |

pysam\_quality\_mask\_final.py ${bam} 10 ${params.mask}

To identify regions with low coverage, we utilize the "count coverage" function available within the pysam package. This function has an advantage over a similar function in the `bedtools` package because it allows counting coverage while considering nucleotide quality. The Python script takes 3 arguments. The first is the bam file from which coverage is calculated, the second is the quality threshold, only nucleotides mapping to a region with at least this value are considered. The last argument is the coverage value, only positions in the genome with coverage below this value are returned. The output is a file named `quality\_mask.bed`. In this file, each position with coverage below the threshold is returned as a separate entry. The file has standard 3 columns. The first is the chromosome name, the second is the start of the region (0-indexed), and the third column is the end of the region (0-indexed). Like any bed file, the regions are closed on the left and open on the right. An example content of the file is:

MN908947.3 0 1

MN908947.3 1 2

MN908947.3 2 3

MN908947.3 2 4

MN908947.3 100 200

cat quality\_mask.bed | bedtools merge -d 2 | \

awk 'BEGIN {OFS = "\t"}; {if (\$3-\$2 > 3) print \$1,\$2,\$3}' >> low\_coverage.bed

In the next step, using bedtools, regions in the quality\_mask.bed file are merged if the ranges they cover are separated by no more than 2 nucleotides. If, after merging, a low coverage region is at least 4 nucleotides long, it is saved to the low\_coverage.bed file; otherwise, the region is ignored. This empirical criterion helps avoid situations where coverage in a region oscillates around the value given in the $mask argument, and the region is alternately masked and unmasked by very short segments. The example content of the quality\_mask.bed file shown above, after this step, would look like this:

MN908947.3 0 4

This information is stored in the low\_coverage.bed file.

bedtools maskfasta -fi ${reference\_fasta} \

-bed low\_coverage.bed \

-fo lowcoverage\_masked.fa

The final step is to create a fasta file with the reference genome where positions with low coverage are masked with Ns. For this purpose, we use the appropriate function of the`bedtools program. The fasta file with such a genome is named lowcoverage\_masked.fa.

## Module make\_genome\_from\_vcf.nf

Creates genome from VCF file (nanopore only). Equivalent of Module consensus.nf.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Platform** | **Illumina** | | | **Nanopore** |
| **Species** |  | | | SARS INFL RSV |
| **QC**  **switch** |  | **QC**  **Criteria** |  | |
| **JSON output** | NO | **JSON key** |  | |

cat genome.fasta | bcftools consensus --mark-del X input.vcf.gz > sample\_genome.fa

* --mark-del X - very important option, that makes alignments easier.

## Module manta.nf

The purpose of the Manta module is to identify structural variants. Manta utilizes a separate container named nf\_illumina\_sars-4.1-manta. Manta is a standalone pipeline described in detail on the [GitHub page](<https://github.com/Illumina/manta>).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Platform** | **Illumina** | | | **Nanopore** |
| **Species** | SARS INFL RSV | | |  |
| **QC**  **switch** | YES | **QC**  **Criteria** | * The proportion of Ns to the genome length exceeds 90%. | |
| **JSON output** | YES | **JSON key** | "genome\_files\_data": {} | |

## Module masking.nf

*Primer masking (There are two separate modules for Illumina and Nanopore.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Platform** | **Illumina** | | | **Nanopore** |
| **Species** | SARS INFL RSV | | | SARS INFL RSV |
| **QC**  **switch** |  | **QC**  **Criteria** |  | |
| **JSON output** | NO | **JSON key** |  | |

### Illumina

length=`echo "${params.length} - 40" | bc -l`

ivar trim -i ${bam} \

-b ${primers} \

-m \${length} \

-f ${pairs} \

-q ${params.quality\_initial} \

-e \

-p ivar\_trimmed\_all

Standard ivar usage. Here, we invoke it only on reads that map to a single amplicon (reads from Step 4 points 1 a,b,c). These reads are stored in a file named reads\_inneramplicon\_sort.bam. At this stage, we DO NOT use reads that come from amplicon fusions, as those are removed in the Python script call in Step 4.

The flags for ivar signify:

* -b path to the amplicon scheme file
* -m minimum read length after masking. Symbolically, it indicates that a read can be 40 bases shorter than the length selected during script invocation (the minimum length read in Step 2 covering the longest EQA test primer would have this length after primer masking)
* -q additional trimming of nucleotides based on quality. This option cannot be turned off, so to prevent ivar from removing nucleotides, we set the flag to the same value as that for Trimmomatic in Step 2
* -e Keep all reads in the output file, including those not mapping to any primer. By default, ivar retains only those reads (not pairs, but individual reads from pairs) that map to any primer. In our case, all we expect from the program is to mask the regions in reads that come from the primer among the pool of predefined read pairs. We want to further analyze reads that are within the amplicon but do not cover the primer.
* -p prefix, the program will generate a bam file with this name

### Nanopore

if [ ${skip\_trimming} -eq 1 ]; then

samtools sort -o trimmed.bam $bam

samtools index trimmed.bam

else

samtools ampliconclip --filter-len 1 --both-ends -b ${primers} --tolerance ${tolerance} -o forvariants\_initial.bam -O bam ${bam}

samtools sort -o trimmed.bam forvariants\_initial.bam

samtools index trimmed.bam

fi

## Module medaka.nf

Base mustation caller for Nanopore data (SNPs, INDELS and SVs).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Platform** | **Illumina** | | | **Nanopore** |
| **Species** |  | | | SARS INFL RSV |
| **QC**  **switch** | NO | **QC**  **Criteria** |  | |
| **JSON output** | NO | **JSON key** |  | |

Medaka is a tool designed for consensus sequence generation and variant calling specifically tailored for Nanopore sequencing data. It works by leveraging a machine learning model to refine the draft consensus sequence generated by basecallers or assemblers, improving accuracy by detecting subtle errors. The process involves aligning reads to a draft reference sequence, followed by applying Medaka's neural network model to correct base-level errors and generate a highly accurate consensus sequence. This step is essential for reducing the inherent noise and systematic biases characteristic of Nanopore data.

In addition to consensus polishing, Medaka provides functionality for mutation calling by comparing the polished sequence to a reference genome. It identifies single nucleotide variants (SNVs) and small insertions or deletions (indels) with high sensitivity and precision, ensuring reliable detection even for challenging regions. Medaka is particularly valuable in genomic studies requiring high-resolution variant detection, such as microbial genomics or cancer research, where the ability to handle complex datasets and generate high-quality results is crucial.

## Module merging\_nanopore.nf

Helper module - only for Nanopore

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Platform** | **Illumina** | | | **Nanopore** |
| **Species** |  | | | SARS INFL RSV |
| **QC**  **switch** | NO | **QC**  **Criteria** |  | |
| **JSON output** | NO | **JSON key** |  | |

Auxiliary module for merging filtering results of reads from other modules.

samtools merge -o merged\_initial.bam trimmed\_first.bam trimmed\_second.bam

samtools sort -@ ${task.cpus} -o merged.bam merged\_initial.bam

samtools index merged.bam

## Module merging\_one\_segment\_filtering.nf

Helper module - only for Illumina SARS and RSV

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Platform** | **Illumina** | | | **Nanopore** |
| **Species** | SARS RSV | | |  |
| **QC**  **switch** | NO | **QC**  **Criteria** |  | |
| **JSON output** | NO | **JSON key** |  | |

samtools merge -o clean\_sort\_dedup\_trimmed\_sort\_tmp.bam ${filtering\_bam} ${ivar\_bam}  
samtools sort -@ ${task.cpus} -o clean\_sort\_dedup\_trimmed\_sort.bam clean\_sort\_dedup\_trimmed\_sort\_tmp.bam  
samtools index clean\_sort\_dedup\_trimmed\_sort.bam

## Module minimap2.nf

Main mapping utility for Nanopore.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Platform** | **Illumina** | | | **Nanopore** |
| **Species** |  | | | SARS INFL RSV |
| **QC**  **switch** | YES | **QC**  **Criteria** | * number of mapped reads is lower than threshold defined in parameter ${params.min\_number\_of\_reads} | |
| **JSON output** | YES | **JSON key** | * "viral\_mapping\_data": {} | |

minimap2 -a -x map-ont -t ${params.threads} -o tmp.sam ref\_genome.fasta ${reads}  
samtools view -@ ${params.threads} -Sb -F 2052 tmp.sam | \  
samtools sort -@ ${params.threads} -o mapped\_reads.bam -  
samtools index mapped\_reads.bam  
rm tmp.sam  
NO\_READS=`samtools view mapped\_reads.bam | wc -l`

## Module nextalign.nf

Predict all aminoacid sequences of all proteins in predicted viral genome.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Platform** | **Illumina** | | | **Nanopore** |
| **Species** | SARS INFL RSV | | | SARS INFL RSV |
| **QC**  **switch** | NO | **QC**  **Criteria** |  | |
| **JSON output** | NO | **JSON key** |  | |

### SARS-CoV-2

nextalign run -r /home/data/common/nextalign/MN908947.3/sequences.fa \

-m /home/data/common/nextalign/MN908947.3/genes.gff3 \

-g "E,M,N,ORF10,orf1a,orf1b,ORF3a,ORF6,ORF7a,ORF8,S" \

-O . \

output\_consensus\_masked\_SV.fa

### RSV

nextalign run -r /home/data/common/nextalign/hRSV\_${SAMPLE\_SUBTYPE}/sequences.fa \  
 -m /home/data/common/nextalign/hRSV\_${SAMPLE\_SUBTYPE}/genes.gff3 \  
 -g "NS1,NS2,N,P,M,SH,G,F,M2,M2,L" \  
 -O . \  
 output\_consensus\_masked\_SV.fa

### Influenza

nextalign run -r \${NEXTALIGN\_DB}/${SAMPLE\_SUBTYPE}/\${SEGMENT}.fasta \  
 -m \${NEXTALIGN\_DB}/${SAMPLE\_SUBTYPE}/${SAMPLE\_SUBTYPE}.gff \  
 -g \${SEGMENT} \  
 -O . \${FILE}

and for MP protein

sed -i s"/chr7\_MP\_SV/MP/"g consensus\_MP.fasta  
 sed -i s"/chr7\_MP/MP/"g consensus\_MP.fasta  
 prep\_M2.py ${SAMPLE\_SUBTYPE} sample\_M2.fasta \${NEXTALIGN\_DB} . .  
 nextalign run -r M2.fasta \  
 -m \${NEXTALIGN\_DB}/${SAMPLE\_SUBTYPE}/${SAMPLE\_SUBTYPE}.gff \  
 -g M2 \  
 -O . consensus\_M2.fasta  
cp nextalign\_gene\_M2.translation.fasta sample\_M2.fasta

## Module nextclade.nf

Nextclade is module for determination evolutionary lineage.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Platform** | **Illumina** | | | **Nanopore** |
| **Species** | SARS INFL RSV | | | SARS INFL RSV |
| **QC**  **switch** |  | **QC**  **Criteria** |  | |
| **JSON output** | NO | **JSON key** |  | |

nextclade run --input-dataset /home/external\_databases/nextclade\_db/sars-cov-2.zip \  
 --output-csv nextstrain\_lineage.csv \  
 --output-all nextclade\_lineages \  
 ${consensus\_masked\_sv\_fa}

After obtaining the genome sequence of the sampled individual, each of the variants (individual sequences from each program as well as the consensus) is analyzed for classification into specific variants. For this purpose, we use the pango and nextclade programs. Their invocation does not require special arguments, as their configuration is done at the container creation level.

## Module pangolin.nf

Pangolin is module for determination evolutionary lineage.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Platform** | **Illumina** | | | **Nanopore** |
| **Species** | SARS INFL RSV | | | SARS INFL RSV |
| **QC**  **switch** | NO | **QC**  **Criteria** |  | |
| **JSON output** | YES | **JSON key** | * "viral\_classification\_data": [] | |

pangolin --outfile pangolin\_lineage.csv \  
 --threads ${task.cpus} \  
 output\_consensus\_masked\_SV.fa  
parse\_pangolin\_output\_csv2json.py pangolin\_lineage.csv pangolin\_sars.json  
jq -s "." pangolin\_sars.json > pangolin.json

This module is part of all pipelines, although it is actually run only for SARS. This is a remnant from the time when the concept of a single unified pipeline for all organisms was being considered.

## Module picard.nf

Down-sampling module

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Platform** | **Illumina** | | | **Nanopore** |
| **Species** | SARS INFL RSV | | |  |
| **QC**  **switch** | YES | **QC**  **Criteria** | SARS i RSV   * Number of Reads lower than hardcoded threshold at level 10'000 of reads   INFL   * The median coverage of the segment is less than the hardcoded value of 50. | |
| **JSON output** | NO | **JSON key** |  | |

java -jar /opt/picard/picard.jar PositionBasedDownsampleSam \  
 --INPUT ${bam} \  
 --OUTPUT downsample.bam -F \${NORM}

## Module picard\_wgsMetrics.nf

This module collects some standard metrics from WGS and and expose them as JSON file.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Platform** | **Illumina** | | | **Nanopore** |
| **Species** | SARS INFL RSV | | | SARS INFL RSV |
| **QC**  **switch** | NO | **QC**  **Criteria** |  | |
| **JSON output** | NO | **JSON key** | * "viral\_genome\_data": {} | |

if [ ${params.machine} == 'Illumina' ]; then  
 java -jar /opt/picard/picard.jar CollectWgsMetrics --REFERENCE\_SEQUENCE ${ref\_genome} \  
 --MINIMUM\_BASE\_QUALITY ${params.quality\_initial} \  
 --MINIMUM\_MAPPING\_QUALITY ${params.min\_mapq} \  
 --INPUT ${bam} \  
 --OUTPUT picard\_statistics.txt  
elif [ ${params.machine} == 'Nanopore' ]; then  
  
 java -jar /opt/picard/picard.jar CollectWgsMetrics --REFERENCE\_SEQUENCE ${ref\_genome} \  
 --MINIMUM\_BASE\_QUALITY ${params.quality\_initial} \  
 --MINIMUM\_MAPPING\_QUALITY ${params.min\_mapq} \  
 --INPUT ${bam} \  
 --COUNT\_UNPAIRED TRUE \  
 --OUTPUT picard\_statistics.txt  
 fi

(...)

picard\_parser.py --input\_file\_picard picard\_statistics.txt \  
 --input\_file\_primers Primer\_usage.txt \  
 --input\_file\_bedgraph \${summary\_coverage\_file} \  
 --output\_path "${params.results\_dir}/${sampleId}/" \  
 --status "tak" \  
 --output viral\_genome\_data.json

## Module reassortment.nf

The module creates a reference genome for a given sample.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Platform** | **Illumina** | | | **Nanopore** |
| **Species** | INFL | | | INFL |
| **QC**  **switch** | NO | **QC**  **Criteria** |  | |
| **JSON output** | NO | **JSON key** | * "infl\_data": {} | |

We start by selecting the HX NY subtypes, then we examine the mapping matrix for each of the subtypes. If the mapping score is below 0.7 and the sequence of the segment with a better score than your sequence (meaning the sequences are indeed different), we consider it to be a better sequence for the mappings (and an indication that it is actually reassortment). The score comes from the Needleman-Wunsch algorithm.

## Module resistance.nf

The module predicts drug resistance for influenza strains.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Platform** | **Illumina** | | | **Nanopore** |
| **Species** | INFL | | | INFL |
| **QC**  **switch** | NO | **QC**  **Criteria** |  | |
| **JSON output** | YES | **JSON key** | * "infl\_data": {} | |

There is a WHO PDF document listing mutations and drugs.

* <https://www.who.int/publications/m/item/summary-of-neuraminidase-(na)-amino-acid-substitutions-associated-with-reduced-inhibition-by-neuraminidase-inhibitors-(nais)>
* <https://www.who.int/publications/m/item/summary-of-polymerase-acidic-(pa)-protein-amino-acid-substitutions-analysed-for-their-effects-on-baloxavir-susceptibility>

The table has been manually parsed and is used for comparison with the sequence obtained from the reassortment module.

## Module snpEff.nf

Predicts fenotypic effects of a mutation.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Platform** | **Illumina** | | | **Nanopore** |
| **Species** | SARS INFL RSV | | | SARS INFL RSV |
| **QC**  **switch** |  | **QC**  **Criteria** |  | |
| **JSON output** | NO | **JSON key** |  | |

java -jar /opt/snpEff/snpEff.jar ann -noStats ${params.ref\_genome\_id} \  
 ${consensus\_vcf\_gz} > detected\_variants\_consensus\_annotated.vcf

The analysis was conducted using the snpEFF program. The only required input, besides specifying the genome name present in the database, is a VCF file containing mutations. This file is then parsed into a text file using bcftools. At this stage, a consensus VCF file is created. Note that this file is only used for functional analysis, not for creating the genome sequence.

bcftools query --format '%REF%POS%ALT| %ANN \n' \  
 detected\_variants\_consensus\_annotated.vcf.gz | \  
 cut -d "|" -f1,3,5,12 | \  
 tr "|" "\t" | \  
 awk 'BEGIN {OFS = "\t"} {if ( \$2 == "upstream\_gene\_variant" || \$2 == "downstream\_gene\_variant") {gene="."; aa="."} else {gene=\$3; aa=\$4}; print gene, \$1, aa, \$2}' > detected\_variants\_consensus\_annotated.txt

The `-n+2` option ensures that only mutations occurring in at least two partial files are reported, while `-c` all defines how positions in different files are treated, with "all" indicating that entries in different files covering the same position are treated as identical.

Subsequently, the snpEFF program utilizes the files `0000.vcf` and `0001.vcf` created in the dir directory, which contain VCF files from the lofreq and freebayes programs filtered to include only positions present in at least 2 programs. These files are parsed in the script to obtain a nicely formatted table.

Note: In the case of complicated variants (deletions of different lengths), there is no guarantee that such a position will be found in the annotated VCF file, which is a limitation of the isec function.

## Module sort\_and\_index.nf

Auxiliary module for Influenza that replaces merging from other pipelines. It only sorts and indexes BAM files.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Platform** | **Illumina** | | | **Nanopore** |
| **Species** | INFL | | |  |
| **QC**  **switch** | NO | **QC**  **Criteria** |  | |
| **JSON output** | NO | **JSON key** |  | |

samtools sort -o ${newBam} ${bam}  
samtools index ${newBam}

## Module substitute\_ref.nf

Auxiliary module in Nanopore to facilitate the double-round process.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Platform** | **Illumina** | | | **Nanopore** |
| **Species** | SARS INFL RSV | | | SARS INFL RSV |
| **QC**  **switch** | YES | **QC**  **Criteria** | * The proportion of Ns to the genome length exceeds 90%. | |
| **JSON output** | NO | **JSON key** |  | |

This process substitutes the reference genome from the second step of genome generation in Nanopore with the originally used reference genome and integrates Nanopore and Illumina I/O. It also functions as a QC switch to replicate the behavior of the introduce\_SV\_with\_manta module from Illumina.

## Module trimmomatic.nf

Trimmomatic is a tool for processing and filtering NGS reads. It performs adapter removal, quality trimming (leading, trailing, and using a sliding window), and discards reads below a specified minimum length. Additionally, it separates paired-end reads into paired and unpaired outputs based on the trimming results.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Platform** | **Illumina** | | | **Nanopore** |
| **Species** | SARS INFL RSV | | | SARS INFL RSV |
| **QC**  **switch** |  | **QC**  **Criteria** |  | |
| **JSON output** | NO | **JSON key** |  | |

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.

1. 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

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

Read XXXXXXXXXXXXXXXXXX

Reference genome XXXXXXXXXXXXXPPPPPPPPPXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

3. 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.

## Module varscan.nf

One of INDEL and SNP Callers.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Platform** | **Illumina** | | | **Nanopore** |
| **Species** | SARS INFL RSV | | | SARS INFL RSV |
| **QC**  **switch** | NO | **QC**  **Criteria** |  | |
| **JSON output** | NO | **JSON key** |  | |

Below is the procedure on how we use and parse the varscan program:

samtools mpileup --max-depth ${params.max\_depth} \

--fasta-ref ${reference\_fasta} \

--min-BQ ${params.quality\_snp} \

${bam} >> ${bam}.mpileup

Creating the mpileup file required by the program. The option `-d` (note: properly functioning from `samtools` version 1.09 onwards) ignores further reads mapping to a given position if the coverage exceeds the assumed value.

varscan\_qual=`echo "${params.quality\_snp} - 1" | bc -l`

java -jar /opt/varscan/VarScan.v2.4.6.jar pileup2cns ${bam}.mpileup \

--min-avg-qual \${varscan\_qual} \

--p-value ${params.pval} \

--min-var-freq ${params.lower\_ambig} \

--min-coverage ${params.min\_cov} \

--variants \

--min-reads2 0 > detected\_variants\_varscan.txt

Calling varscan, the above options signify:

* --min-coverage – minimum coverage at a given position required for the program to identify a variant.
* --min-reads2 - minimum number of reads supporting the variant, set to 0.
* --min-avg-qual – minimum read quality at this position required to include the read in the analysis.
* --p-value – threshold p-value the variant must achieve to be reported.
* --min-var-freq – minimum percentage of reads with a non-reference allele required to report a variant.
* --variants – besides SNPs, the program should also identify short INDELs.`

parse\_vcf\_output\_final.py detected\_variants\_varscan.txt ${params.upper\_ambig} ${params.pval}

A parser converting the text file `detected\_variants\_varscan.txt` to a vcf file format. Compared to the txt file, the vcf file introduces the following changes:

1. Positions reported as deletions/insertions relative to the reference genome, e.g.,

MN908947.3 22204 T +GAGCCAGAA

Containing symbols "+" or "-" are corrected to:

MN908947.3 22204 . T TGAGCCAGAA

2. Heterozygous positions, e.g.,

MN908947.3 21766 . ACATGTC A

3. Positions where the frequency of using the alternative allele is greater than the value provided in the variable $upper\_ambig are converted from ambiguous to the alternative allele version. Varscan returns ambiguous positions even with a reference allele frequency of 0.75, e.g.,

MN908947. 25324 C M (where the frequency of allele A is 60%)

are corrected to

MN908947.3 25324 . C A

4. The QUAL field, which does not have a clear representation in the varscan output, is changed to 30 in the vcf file.

bcftools norm --check-ref w \

--rm-dup all \

--fasta-ref ${reference\_fasta}\

detected\_variants\_varscan.vcf.gz | \

bcftools norm --check-ref w \

--multiallelics -indels \

--fasta-ref ${reference\_fasta} | \

bcftools filter \

--include "QUAL >= \${qual} && AF >= ${params.lower\_ambig} && DP >= ${params.min\_cov}" > detected\_variants\_varscan\_final.vcf

"Sorting" the vcf file using the norm function from the bcftools package. The `-c w` option causes the program to return a warning instead of an error if ambiguous positions are present in the file, and further process the vcf file. The "-d all" removes duplicates, the `-m -indels` option splits multiallelic positions into individual entries. Generally, vcf files show the richness of changes identified by programs. Sometimes overlapping changes occur, sometimes programs return unusual mutation notations, multiallelic positions, etc. To organize such a vcf file, we call the above command. Then we ensure that the resulting records still meet the quality and coverage criteria. In the case of varscan, this is not as important because when parsing its output, we set the quality to 1 or 30. Simple examples of changes identified in sample 1 from EQA for the freebayes program.

Before:

MN908947.3 22204 . TGA TGAGCCAGAAGA

After:

MN908947.3 22204 . T TGAGCCAGAA

Or splitting a complex mutation

Before:

MN908947.3 25334 . GAAG CACG,CACC,GACC,GAAC

After, splitting the mutation into components

MN908947.3 25334 . GAA CAC

MN908947.3 25334 . GAAG CACC

MN908947.3 25336 . AG CC

MN908947.3 25337 . G C

cat ${reference\_fasta} | bcftools consensus --samples - detected\_variants\_varscan\_final.vcf.gz > varscan.fa

The above command incorporates all mutations present in the vcf file into the reference genome. The -s - option means that we ignore the genotype for the sample (we do not have multiple samples in the vcf file), and we introduce all mutations present in the vcf file. The result is a fasta file format with the genome.

## Module vcf\_from\_fasta.nf

The module generates a VCF file based on FASTA files.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Platform** | **Illumina** | | | **Nanopore** |
| **Species** | SARS INFL RSV | | | SARS INFL RSV |
| **QC**  **switch** | NO | **QC**  **Criteria** |  | |
| **JSON output** | NO | **JSON key** |  | |

# JSON output

This table combines information about which module is responsible for filling in which part of the JSON file.

|  |  |  |
| --- | --- | --- |
| **JSON keys** | **Illumina** | **Nanopore**  **(if empty identical as illumina)** |
|  |  |  |
| { |  |  |
| "output": { |  |  |
| "pathogen": str | nf\_pipeline\_viral.nf |  |
| "pipeline\_version": str | nf\_pipeline\_viral.nf |  |
| "timestamp": str | json\_aggegator.nf |  |
| "dehumanized\_data": {} | dehumanization.nf |  |
| "genome\_files\_data": {} | manta.nf | consensus.nf |
| "sequencing\_summary\_data": [] | fastqc.nf |  |
| "contamination\_data": [] | kraken2.nf |  |
| "viral\_genome\_data": {} | picard\_wgsMetrics.nf manta.nf | picard\_wgsMetrics.nf consensus.nf |
| "viral\_classification\_data": [] | pangolin.nf  nextclade.nf |  |
| "viral\_mapping\_data": {} | bwa.nf | minimap2.nf |
| "viral\_mutation\_data" [] | snpEff.nf |  |
| "structural\_data": {} | alphafold.nf |  |
| "sars\_data": {} | coinfection\_analysis.nf  freyja.nf |  |
| "infl\_data": {} | reassortment.nf  resistance.nf |  |
| "rsv\_data": {} | detect\_type\_rsv.nf |  |
| } |  |  |
| } |  |  |