# Service results report from the Bioinformatics Unit (BU-ISCIII)

## SRVCNM983 service information:

Here we describe the information about the service request:

- Service ID: SRVCNM983
- Request date: 2024-02-28
- Center Requested service : CNM
- Service status : in_progress
- Requested services:

Genomic Data Analysis

Next Generation Sequencing data analysis

DNAseq / cDNAseq: Exome sequencing (WES) / Genome sequencing (WGS) / Targeted sequencing

Low-frequency variants detection and annotation for whole genome or sequencing panel (e.g. retinoblastoma gene panel)

Bacteria: De novo genome assembly and annotation

Viral: Genomic reconstruction, variant calling and de novo assembly

- Service Notes : asdasda

## User information:

Here we describe the information about the user requesting the service.

- Name and first name: Admin Admin
- Username: bioinfoadm
- E-mail: bioinformatica@isciii.es

## Resolution information

Here we describe information about the requested resolution.

- Resolution: SRVCNM983.1
- Resolution full name: SRVCNM983_20240228_TEST001_bioinfoadm_S
- Resolution date: 2024-02-28
- Resolution estimated delivery date: 2024-03-08

- Resolution on Queued date: 2024-02-28
- Resolution in Progress date: 2024-06-03
- Requested services handle in this resoution:
    - Bacteria: De novo genome assembly and annotation
    - Viral: Genomic reconstruction, variant calling and de novo assembly
- Resolution Notes: asda

## Delivery information

Here we describe information about the resolution delivery.

- Resolution Delivery date: 2024-02-28
- Resolution Delivery notes: alskjdalskdj

## Samples sequenced at iSCIII:

Here we describe information about the project associated to the service:

- Run name: NextSeq_GEN_344
    - Project name: NextSeq_GEN_344_20211223_ICasas
    - Samples: 219907, 2110055, 2110065, 2110066

# Results

Here we describe the content of the RESULTS folder.

# Assembly

Here, we describe the results from the Assembly pipeline for de novo genome assembly and annotation.

- **assemblies**: a symbolic link to the raw reads associated with the resolution.
- **kmerfinder_summary.csv**: a .csv file containing the main results from kmerfinder. For each sample, you should check that both the best hit and the second hit reported by kmerfinder correspond to the species name indicated by the researcher when requesting the service. If the second hit is associated to a different species, check other metrics like %GC or %genome fraction in the MultiQC report, since this might reveal a contamination in that sample.
- *sample_name*: sample name.
- *07-kmerfinder_best_hit_# Assembly*: RefSeq assembly accession ID.
- *07-kmerfinder_best_hit_Accession Number*: accession number of entry ID in fasta file.
- *07-kmerfinder_best_hit_Depth*: is the number of matched kmers in the query sequence divided by the total number of Kmers in the template. For read files this estimates the sequencing depth.
- *07-kmerfinder_best_hit_Description*: additional descriptions available in fasta file, or in the case of organism databases the identifier lines of fasta files.
- *07-kmerfinder_best_hit_Expected*: is the expected score, i.e.the expected total number of matching Kmers between query and template (randomly selected).
- *07-kmerfinder_best_hit_Num*: is the sequence number of accession entry in the KmerFinder database.
- *07-kmerfinder_best_hit_Query_Coverage*: is the percentage of input query/reads Kmers that match the template.
- *07-kmerfinder_best_hit_Score*: is the total number of matching Kmers between the query and the template.
- *07-kmerfinder_best_hit_Species*: Species name.
- *07-kmerfinder_best_hit_TAXID*: NCBI's TaxID number of the hit.
- *07-kmerfinder_best_hit_TAXID Species*: NCBI's species TaxID number of the hit (sometimes bacterial strain or substrain TaxIDs can be given above).
- *07-kmerfinder_best_hit_Taxonomy*: complete taxonomy of the hit.
- *07-kmerfinder_best_hit_Template_Coverage*: is the template/genome coverage.
- *07-kmerfinder_best_hit_Template_length*: is the number of Kmers in the template.
- *07-kmerfinder_best_hit_p_value*: is the p-value corresponding to the obtained q_value.
- *07-kmerfinder_best_hit_q_value*: is the quantile in a standard Pearson Chi-square test, to test whether the current template is a significant hit.
- *07-kmerfinder_best_hit_tot_depth*: depth value based on all query kmers that can be found in

the template sequence.

- *07-kmerfinder_best_hit_tot_query_Coverage*: is calculated based on the ratio of the score and the number of kmers in the query sequence, where the score includes kmers matched before.
- *07-kmerfinder_best_hit_tot_template_Coverage*: is calculated based on ratio of the score and the number of unique kmers in the template sequence, where the score includes kmers matched before.
- *07-kmerfinder_second_hit_# Assembly*: RefSeq assembly accession ID.
- *07-kmerfinder_second_hit_Accession Number*: accession number of entry ID in fasta file.
- *07-kmerfinder_second_hit_Depth*: is the number of matched kmers in the query sequence divided by the total number of Kmers in the template. For read files this estimates the sequencing depth.
- *07-kmerfinder_second_hit_Description*: additional descriptions available in fasta file, or in the case of organism databases the identifier lines of fasta files.
- *07-kmerfinder_second_hit_Expected*: is the expected score, i.e.the expected total number of matching Kmers between query and template (randomly selected).
- *07-kmerfinder_second_hit_Num*: is the sequence number of accession entry in the KmerFinder database.
- *07-kmerfinder_second_hit_Query_Coverage*: is the percentage of input query Kmers that match the template.
- *07-kmerfinder_second_hit_Score*: is the total number of matching Kmers between the query and the template.
- *07-kmerfinder_second_hit_Species*: Species name.
- *07-kmerfinder_second_hit_TAXID*: NCBI's TaxID number of the hit.
- *07-kmerfinder_second_hit_TAXID Species*: NCBI's species TaxID number of the hit (sometimes bacterial strain or substrain TaxIDs can be given above).
- *07-kmerfinder_second_hit_Taxonomy*: complete taxonomy of the hit.
- *07-kmerfinder_second_hit_Template_Coverage*: is the template coverage.
- *07-kmerfinder_second_hit_Template_length*: is the number of Kmers in the template.
- *07-kmerfinder_second_hit_p_value*: is the p-value corresponding to the obtained q_value.
- *07-kmerfinder_second_hit_q_value*: is the quantile in a standard Pearson Chi-square test, to test whether the current template is a significant hit.
- *07-kmerfinder_second_hit_tot_depth*: depth value based on all query kmers that can be found in the template sequence.
- *07-kmerfinder_second_hit_tot_query_Coverage*: is calculated based on the ratio of the score and the number of kmers in the query sequence, where the score includes kmers matched before.
- *07-kmerfinder_second_hit_tot_template_Coverage*: is calculated based on ratio of the score and the number of unique kmers in the template sequence, where the score includes kmers matched before.
- *Total_hits_07_kmerfinder*: number of total hits.

- **multiqc_report.html**: an interactive report containing the results from MultiQC (kmerfinder, QUAST, quality control, etc.)
- **quast_GCF_XXXXXXXXX.X_ASMXXXXXv2_report.html**: an interactive report obtained after the execution of QUAST, providing different metrics about the assembly QC against the reference.
- **quast_global_report.html**: an interactive report obtained after the execution of QUAST, providing different metrics about the global assembly QC.
- **summary_assembly_metrics_mqc.csv**: a custom table containing most relevant assembly QC metrics.
- *Sample*: sample ID.
- *Input reads*: number of input reads for each sample.
- *Trimmed reads (fastp)*: number of trimmed reads.
- *Contigs*: number of contigs.
- *Largest contig*: length of the largest contig.
- *N50*: is the contig length such that using longer or equal length contigs produces half of the bases of the assembly. Usually there is no value that produces exactly 50%, so the technical definition is the maximum length x such that using contigs of length at least x accounts for at least 50% of the total assembly length.
- *% Genome fraction*: the total number of aligned bases in the reference, divided by the genome size.
- *Best hit (Kmerfinder)*: best hit species name.
- *Best hit assembly ID (Kmerfinder)*: best hit RefSeq assembly accession ID.
- *Best hit query coverage (Kmerfinder)*: best hit query coverage.
- *Best hit depth (Kmerfinder)*: best hit depth.
- *Second hit (Kmerfinder)*: second hit species name.
- *Second hit assembly ID (Kmerfinder)*: second hit RefSeq assembly accession ID.
- *Second hit query coverage (Kmerfinder)*: second hit query coverage.
- *Second hit depth (Kmerfinder)*: second hit depth.

[!WARNING] Software's versions used in this analysis can be obtained from the `MultiQC` report.

# Viralrecon

Here we describe the results from the Viralrecon pipeline for viral genome reconstruction.

[!WARNING] Some of the files listed here may not be in your `RESULTS` folder. It will depend on the analysis you requested.

## Mapping approach results

- mapping_illumina.xlsx: statistics for mapped reads against viral and host genomes.
- run: Run name
- user: User name

- host: Host name
- Virussequence: Reference virus used
- sample: Sample name
- totalreads: Total reads after trimming
- readshostR1: Total reads of host genome in R1
- readshost: Total reads of host genome in R1 and R2
- %readshost: Percentage of reads that correspond to the host genome
- readsvirus: Number of reference viral genome reads
- %readsvirus: Percentage of reference viral genome reads
- unmappedreads: number of reads that did not correspond to viral reference or host genome.
- %unmapedreads: Percentage of reads that did not correspond to viral reference or host genome.
- medianDPcoveragevirus: Median depth of coverage of the reference viral genome
- Coverage>10x(%): Percentage of viral reference genome coverage to more than 10X
- Variantsinconsensusx10: Number of variants included in the consensus after filtering: more than 10X and 0.75 AF
- %Ns10x: Percentage of consesus genome masked due to having less than 10X depth
- Lineage: Pangolin assigned lineage. *Warning: Only for SARS-CoV-2 sequencing data*
- Date: Analysis date. *Warning: Only for SARS-CoV-2 sequencing data*
- mapping_consensus: this folder contains the masked (<10x) genomes obtained with consensus sequences using mapping and majority variant calling
- variants_annot: table with all annotated variants. *Warning: Only when annotation .gff file was provided*
- variants_long_table.xlsx: Table with variants for all the samples in long format. *Warning: Only when annotation .gff file was provided*
- SAMPLE: sample name
- CHROM: Reference ID
- POS: Position of the variant
- REF: Ref allele
- ALT: Alt allele
- FILTER: Column indicating if the variant passed the filters. If PASS the variant passed all the filters. If not, the name of the filter that wasn't passed will appear
- DP: Position depth
- REF_DP: Ref allele depth
- ALT_DP: Alt allele depth
- AF: Allele frequency
- GENE: Gene name in annotation file
- EFFECT: Effect of the variant
- HGVS_C: Position annotation at CDS level
- HGVS_P: Position annotation at protein level
- HGVS_P_1LETTER: Position annotation at protein level with the aminoacid annotation in 1 letter format

- Caller: Variant caller used
- pangolin.xlsx: Pangolin complete results *Warning: Only for SARS-CoV-2 sequencing data*
- nextclade.xlsx: Results from Nextclade *Warning: Only for SARS-CoV-2 sequencing data*

## *de novo* assembly approach results

- assembly_stats.xlsx: Stats of the *de novo* assembly steps. This table contains the following columns:
- run: Run name
- user: User name
- host: Host name
- Virussequence: Reference virus used
- sample: Sample name
- totalreads: Total reads after trimming
- readshostR1: Total reads of host genome in R1
- readshost: Total reads of host genome in R1 and R2
- %readshost: Percentage of reads that correspond to the host genome
- Non-host-reedas: Number of reads remaining after host removal
- #Contigs: Number of contigs in the assembly
- Largest contig: Size in nucleotides of the larges contig in the assembly
- % Genome fraction: Percentage of the reference genome covered by the assembly.*Warning: Only when reference genome was provided*
- assembly_spades: Scaffolds fasta files with the spades de novo assembly.*Warning: Only when NO reference genome was provided, or reference genome didn't match*
- abacas_assembly: spades de novo assembly where contigs were contiguated using ABACAS and the reference genome. *Warning: Only when reference genome was provided*

[!WARNING] Software's versions used in this analysis can be obtained from the MultiQC report.

## MAG

Here we describe the results from the MAG pipeline for multispecies metagenomic analysis.

- krona_results.html : Final HTML report with the top 5 species most present in all samples.

[!WARNING] Software's versions used in this analysis can be obtained from the MultiQC report.

# nf-core/bacass: Output

## Introduction

This document describes the output produced by the pipeline. Most of the plots are taken from the MultiQC report, which summarises results at the end of the pipeline.

The directories listed below will be created in the results directory after the pipeline has finished. All paths are relative to the top-level results directory.

## Pipeline overview

The pipeline is built using Nextflow and processes data using the following steps:

## Quality trimming and QC

### Short Read Trimming

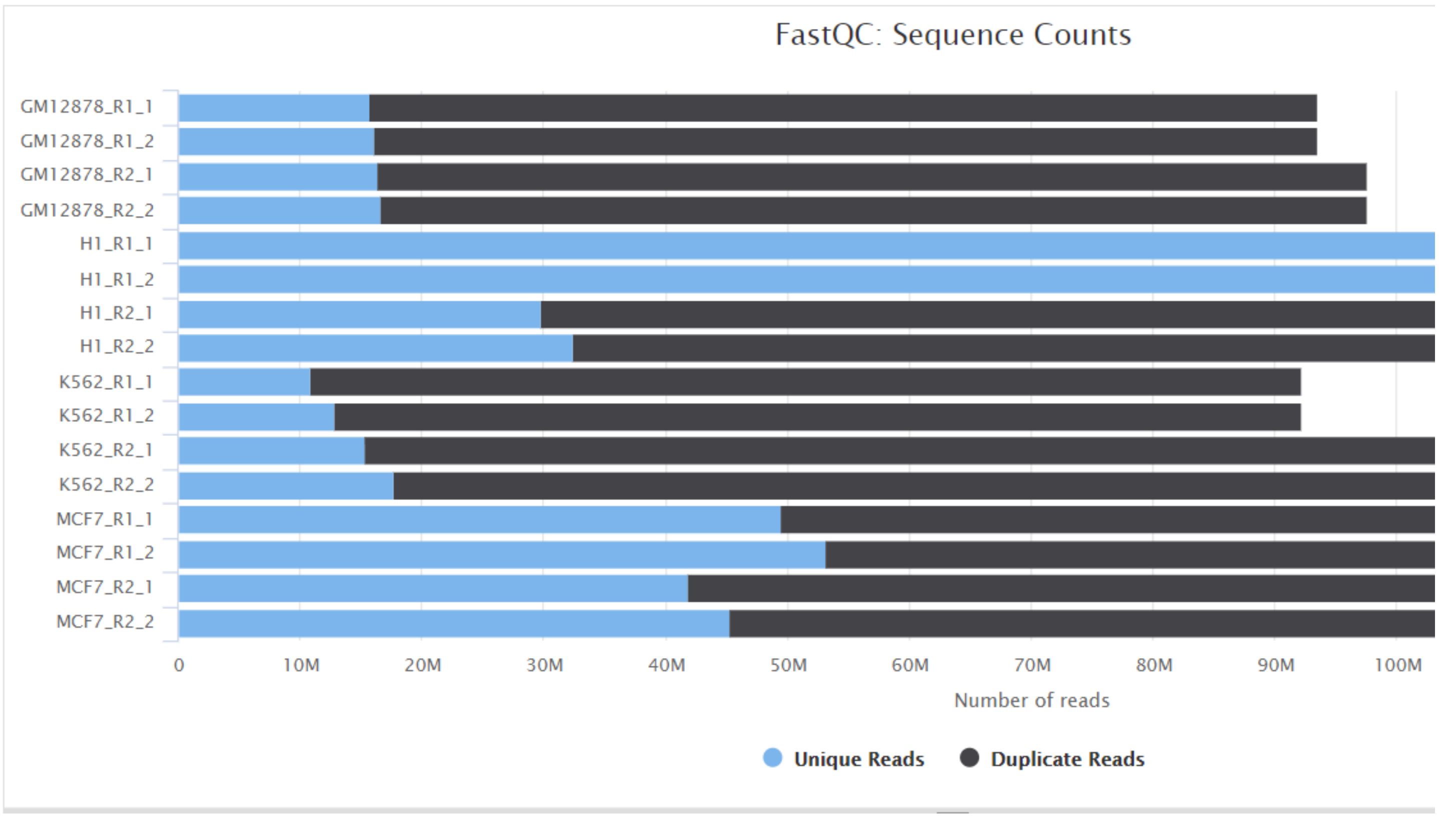
This step quality trims the end of reads, removes degenerate or too short reads and if needed, combines reads coming from multiple sequencing runs.

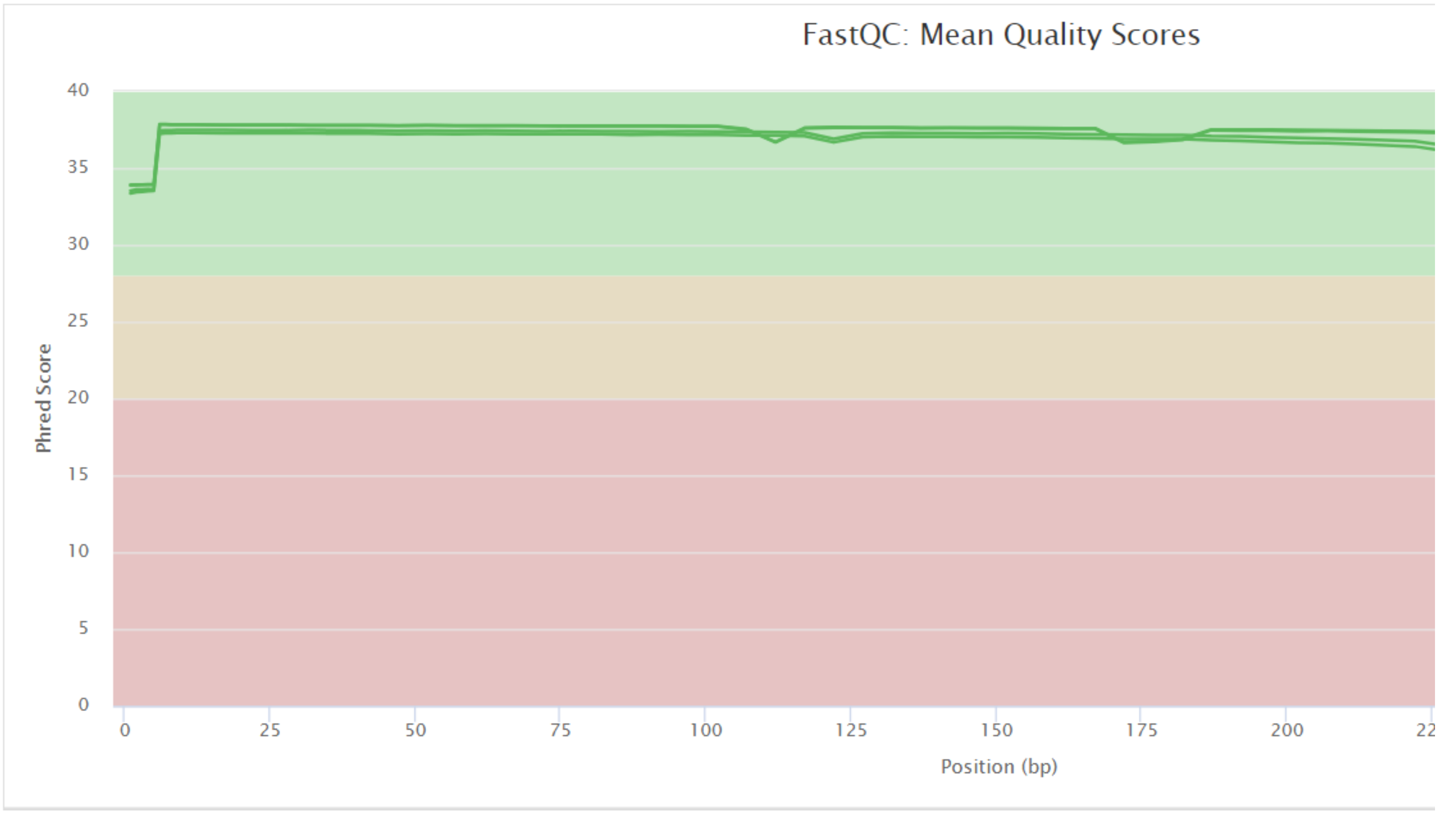
▼ Output files

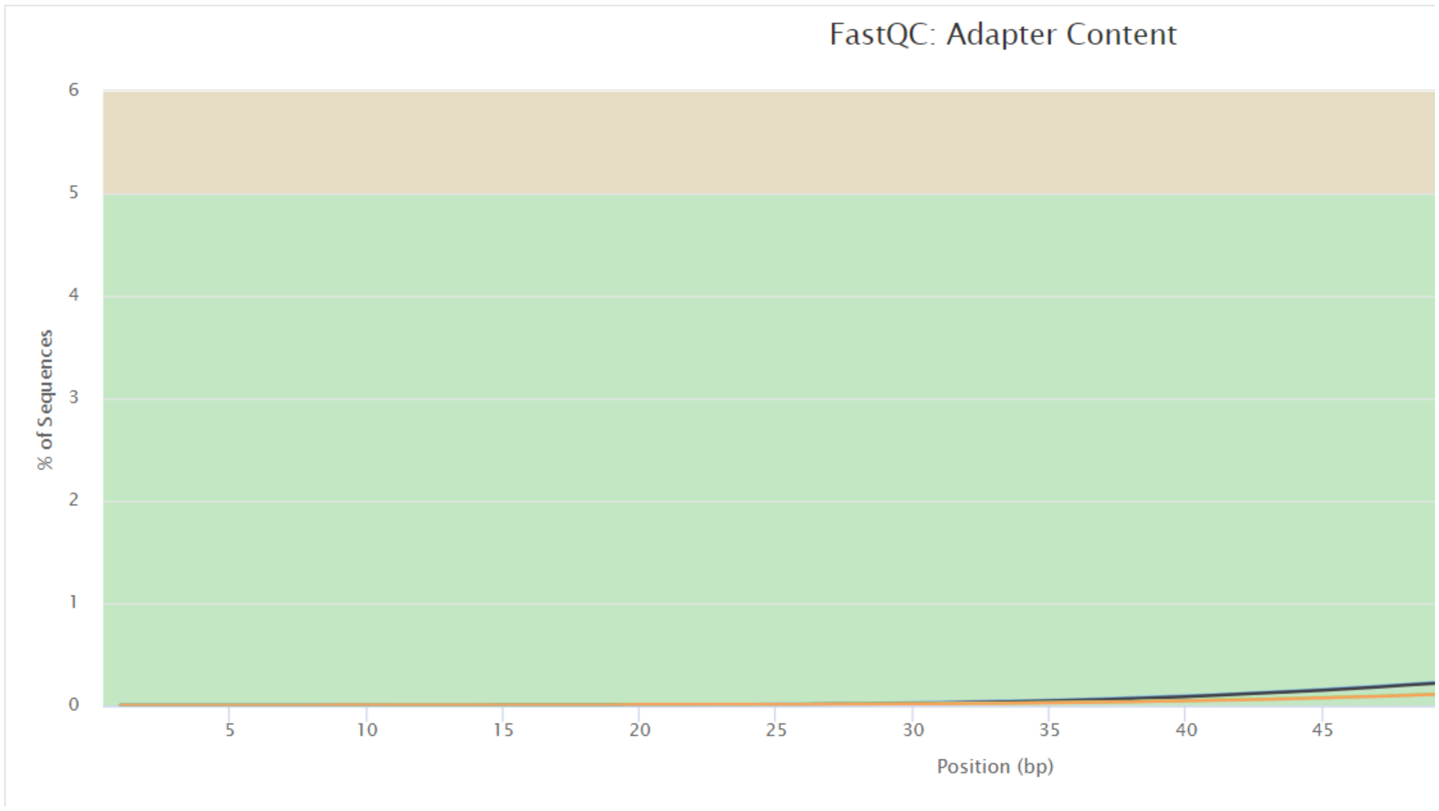
- `trimming/shortreads/`
- `*.fastp.fastq.gz`: The trimmed/modified/unmerged fastq reads

### Short Read RAW QC

FastQC gives general quality metrics about your sequenced reads. It provides information about the quality score distribution across your reads, per base sequence content (%A/T/G/C), adapter contamination and overrepresented sequences. For further reading and documentation see the FastQC help pages.



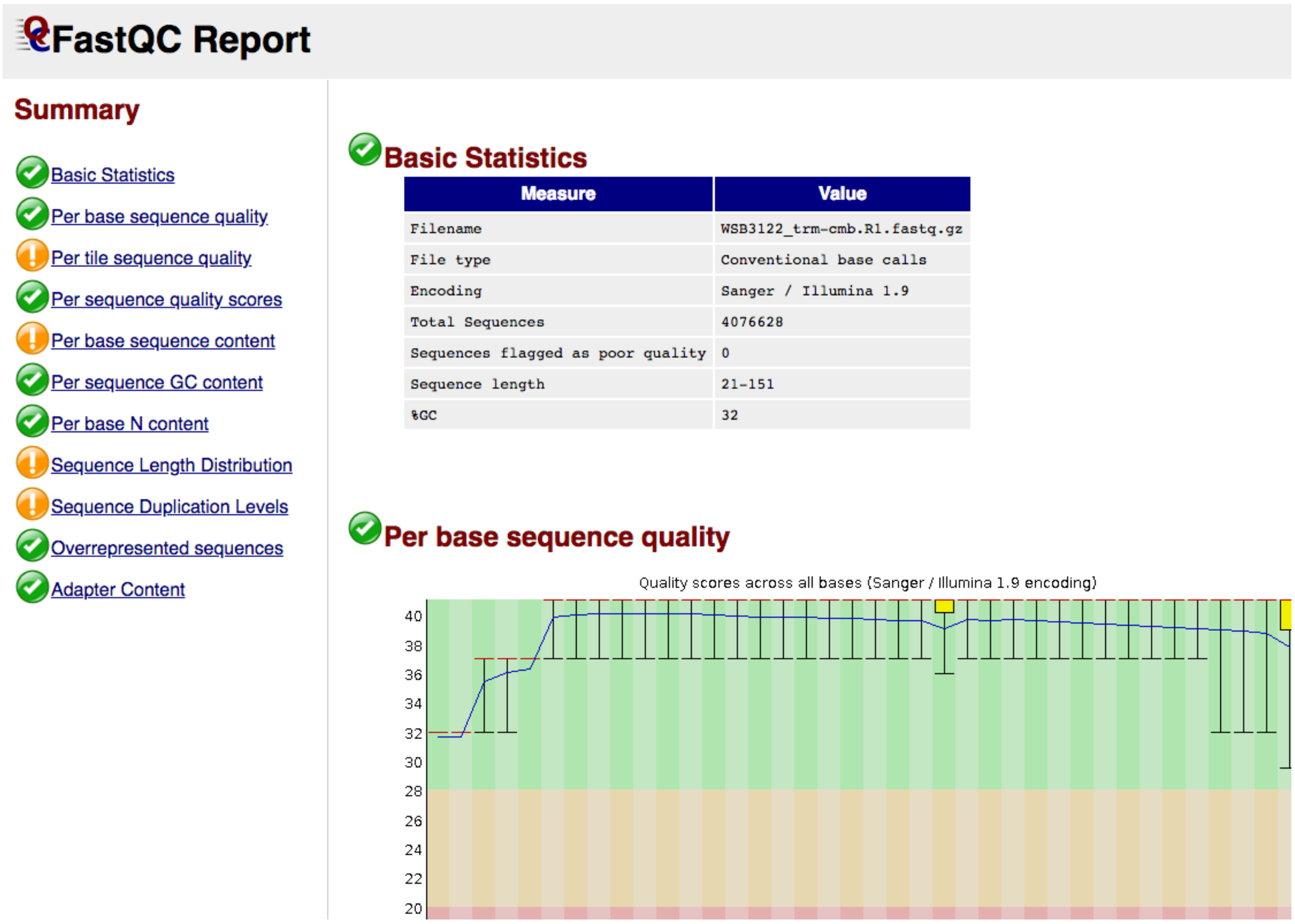




:::note The FastQC plots displayed in the MultiQC report shows *untrimmed* reads. They may contain adapter sequence and potentially regions with low quality. :::

▼ Output files

- FastQC/
- *.html: FastQC report containing quality metrics.
- *.zip: Zip archive containing the FastQC report, tab-delimited data file and plot images.



FastQC Report

Summary

- ✓ Basic Statistics
- ✓ Per base sequence quality
- ! Per tile sequence quality
- ✓ Per sequence quality scores
- ! Per base sequence content
- ✓ Per sequence GC content
- ✓ Per base N content
- ! Sequence Length Distribution
- ! Sequence Duplication Levels
- ✓ Overrepresented sequences
- ✓ Adapter Content

✓ Basic Statistics

| Measure | Value |
|---|---|
| Filename | WSB3122_trm-cmb.R1.fastq.gz |
| File type | Conventional base calls |
| Encoding | Sanger / Illumina 1.9 |
| Total Sequences | 4076628 |
| Sequences flagged as poor quality | 0 |
| Sequence length | 21-151 |
| %GC | 32 |

✓ Per base sequence quality

Quality scores across all bases (Sanger / Illumina 1.9 encoding)
40
38
36
34
32
30
28
26
24
22
20

## Long Read Trimming

This step performs long read trimming on Nanopore input (if provided).

▼ Output files

- trimming/longreads/
- *.fastq.gz: The trimmed FASTQ file
- *.log*: Log file

## Long Read RAW QC

These steps perform long read QC for input data (if provided).

Please refer to the documentation of NanoPlot and PycoQC if you want to know more about the plots created by these tools.

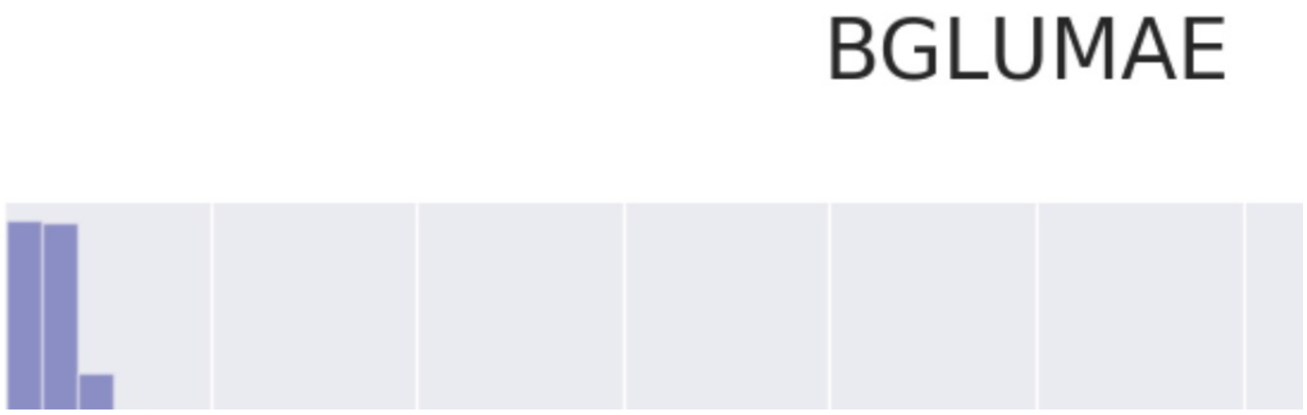
▼ Output files

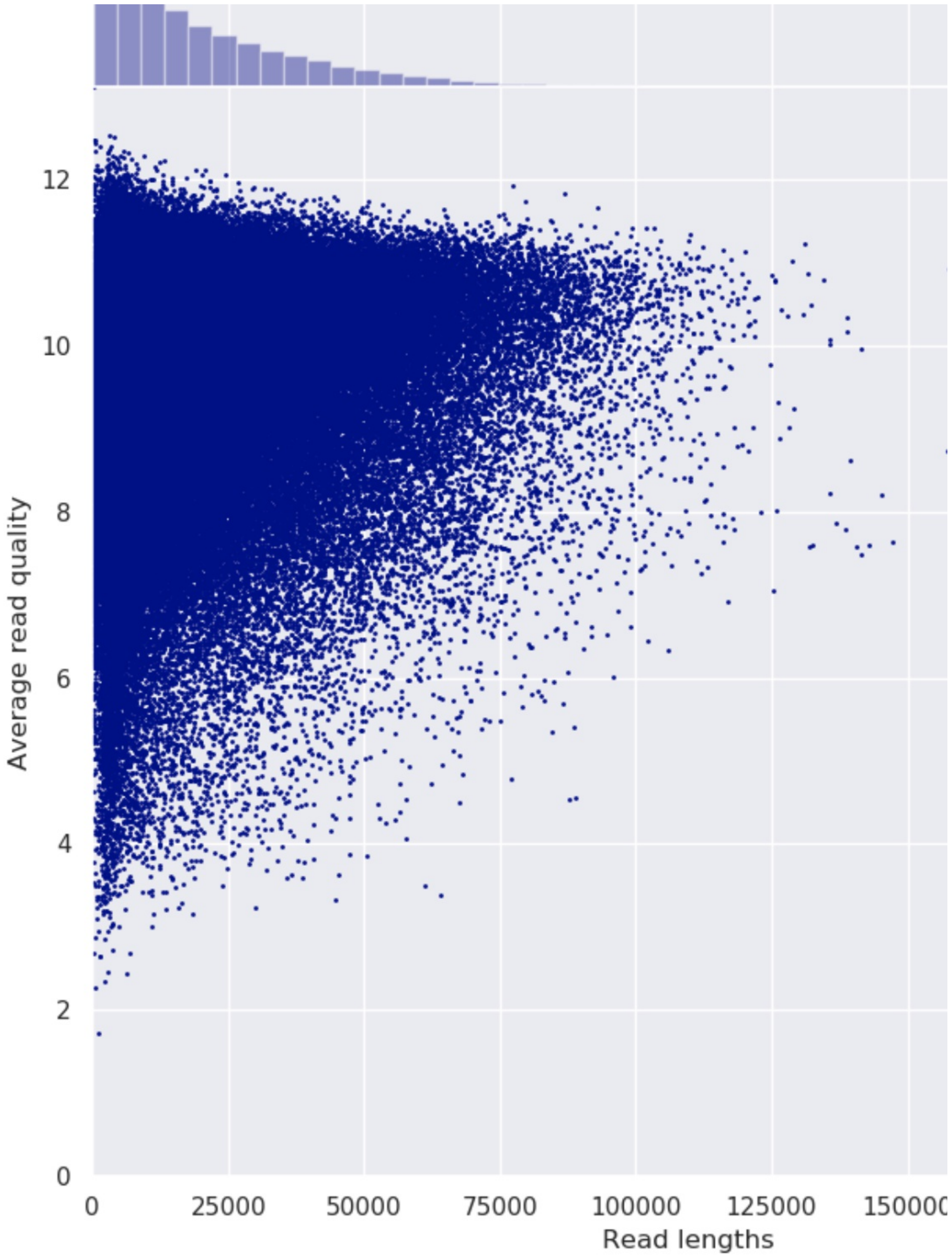
- QC_Longreads/NanoPlot: Various plots in HTML and PNG format

- QC_Longreads/PycoQC

- *_pycoqc.html: QC report in HTML format
- *_pycoqc.json: QC report in JSON format

Example plot from Nanoplot:





## Taxonomic classification

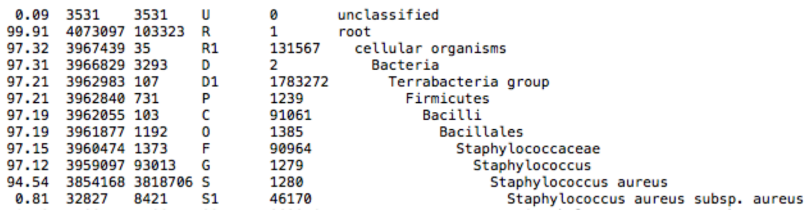
This QC step classifies your reads usingKraken2 a k-mer based approach. This helps to identify samples that have purity issues. Ideally you will not want to assemble reads from samples that are contaminated or contain multiple species. If you like to visualize the report, try Pavian or Krakey.

▼ Output files

- Kraken2/
- *.kraken2.report.txt: Classification of short reads in the Kraken(1) report format.
- *_longreads.kraken2.report.txt: Classification of long reads in the Kraken(1) report format.

See webpage for more details.

Exemplary Kraken2 report screenshot:



```
 0.09  3531    3531    U       0       unclassified
99.91  4073097 103323  R       1       root
97.32  3967439 35      R1      131567    cellular organisms
97.31  3966829 3293    D       2           Bacteria
97.21  3962983 107     D1      1783272       Terrabacteria group
97.21  3962840 731     P       1239            Firmicutes
97.19  3962055 103     C       91061             Bacilli
97.19  3961877 1192    O       1385                Bacillales
97.15  3960474 1373    F       90964                 Staphylococcaceae
97.12  3959097 93013   G       1279                    Staphylococcus
94.54  3854168 3818706 S       1280                      Staphylococcus aureus
 0.81  32827   8421    S1      46170                       Staphylococcus aureus subsp. aureus
```

## Reads QC and Sample purity

The pipeline includes a dedicated step for short and long reads QC as well as contamination analysis usingKmerfinder. This process helps assess the quality and purity of the samples.

▼ Output files

- Kmerfinder/{ID}/
- *_results.txt: Kmerfinder report table containing reads QC results and taxonomic information.

- Kmerfinder/:

- kmerfinder_summary.csv: A CSV file containing the most relevant results of all samples analyzed with Kmerfinder.

## Assembly Output

Trimmed reads are assembled with Unicycler in short or hybrid assembly modes. For long-read assembly, there are alsocanu and miniasm available. Unicycler is a pipeline on its own, which at least for Illumina reads mainly acts as a frontend to Spades with added polishing steps.

▼ Output files

- Unicycler/
- *.scaffolds.fa: Final assembly in fasta format
- *.assembly.gfa: Final assembly in Graphical Fragment Assembly (GFA) format
- *.unicycler.log: Log file summarizing steps and intermediate results on the Unicycler execution

Check out the Unicycler documentation for more information on Unicycler output.

- Canu/
- *.contigs.fasta.gz: Final assembly in fasta format
- *.report: Log file summarizing steps and intermediate results

Check out the Canu documentation for more information on Canu output.

- Miniasm/
- *.fasta.gz: Assembly in Fasta format
- *_assembly_consensus.fasta.gz: Consensus assembly in fasta format (polished by Racon)

Check out the Miniasm documentation for more information on Miniasm output.

- Dragonflye/
- *.contigs.fa: Assembly in Fasta format
- *.dragonflye.log: Log file containing the report of the dragonflye process

Checkout the Dragonflye documentation for more information of the Dragonflye output.

### Polished assemblies

Long reads assemblies can be polished using Medaka or NanoPolish with Fast5 files.

▼ Output files

- Medaka/*_polished_genome.fa

- *_polished_genome.fa: Polished consensus assembly in fasta format

- calls_to_draft.bam: Alignment in bam format
- calls_to_draft.bam.bai: Index of alignment
- consensus.fasta.gaps_in_draft_coords.bed
- consensus_probs.hdf

- Nanopolish/

- polished_genome.fa: Polished consensus assembly in fasta format

## Assembly QC with QUAST

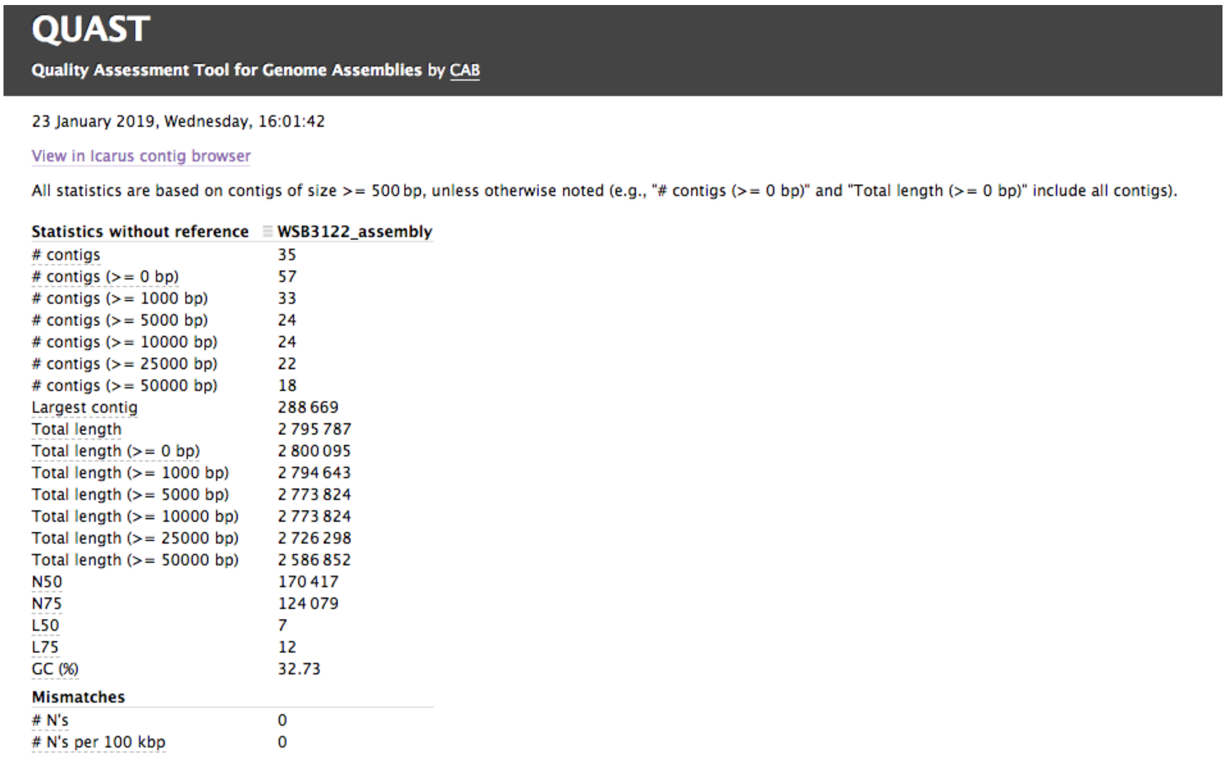
The assembly QC is performed with [QUAST](#) for all assemblies in one report. It reports multiple metrics including number of contigs, N50, lengths etc in form of an html report. It further creates an HTML file with integrated contig viewer (Icarus).

▼ Output files

- `QUAST/report/`
- `icarus.html`: QUAST's contig browser as HTML
- `report.html`: QUAST assembly QC as HTML report
- `report.pdf`: QUAST assembly QC as pdf

- `QUAST/runs_per_reference/{reference_assembly}/`

- `icarus.html`: QUAST's contig browser as HTML
- `report.html`: QUAST assembly QC as HTML report
- `report.pdf`: QUAST assembly QC as pdf



**QUAST**

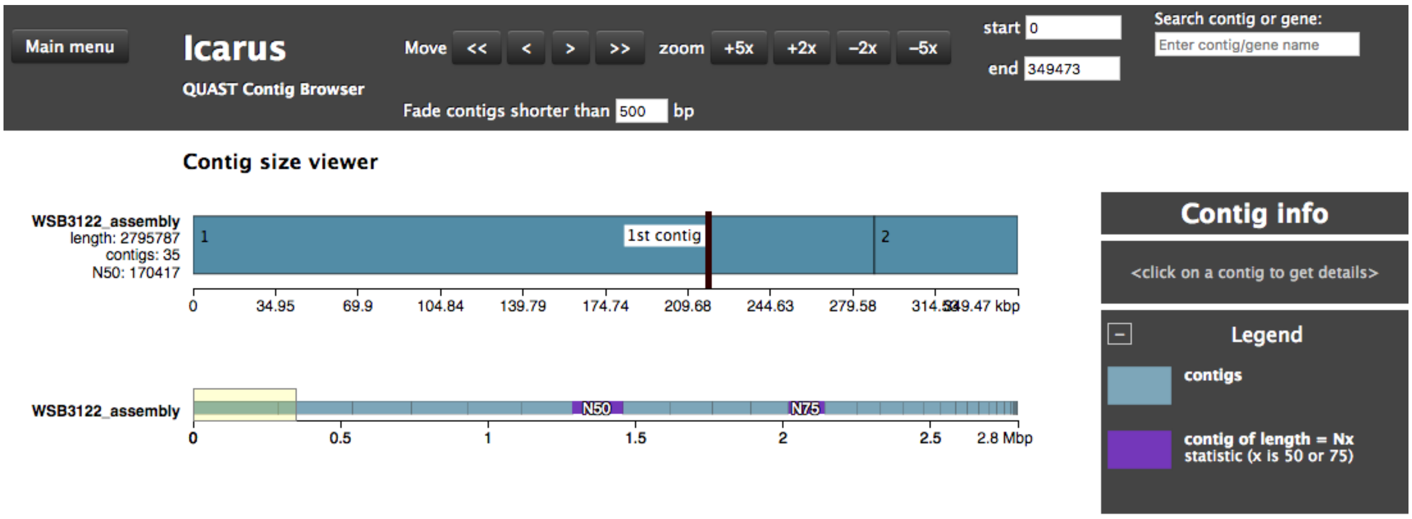
**Quality Assessment Tool for Genome Assemblies by CAB**

23 January 2019, Wednesday, 16:01:42

View in Icarus contig browser

All statistics are based on contigs of size >= 500 bp, unless otherwise noted (e.g., "# contigs (>= 0 bp)" and "Total length (>= 0 bp)" include all contigs).

| Statistics without reference | WSB3122_assembly |
|---|---|
| # contigs | 35 |
| # contigs (>= 0 bp) | 57 |
| # contigs (>= 1000 bp) | 33 |
| # contigs (>= 5000 bp) | 24 |
| # contigs (>= 10000 bp) | 24 |
| # contigs (>= 25000 bp) | 22 |
| # contigs (>= 50000 bp) | 18 |
| Largest contig | 288 669 |
| Total length | 2 795 787 |
| Total length (>= 0 bp) | 2 800 095 |
| Total length (>= 1000 bp) | 2 794 643 |
| Total length (>= 5000 bp) | 2 773 824 |
| Total length (>= 10000 bp) | 2 773 824 |
| Total length (>= 25000 bp) | 2 726 298 |
| Total length (>= 50000 bp) | 2 586 852 |
| N50 | 170 417 |
| N75 | 124 079 |
| L50 | 7 |
| L75 | 12 |
| GC (%) | 32.73 |
| **Mismatches** | |
| # N's | 0 |
| # N's per 100 kbp | 0 |


Main menu
Icarus
QUAST Contig Browser
Move << < > >> zoom +5x +2x −2x −5x
start 0
end 349473
Search contig or gene:
Enter contig/gene name
Fade contigs shorter than 500 bp
Contig size viewer
WSB3122_assembly
length: 2795787
contigs: 35
N50: 170417
1
1st contig
2
0 34.95 69.9 104.84 139.79 174.74 209.68 244.63 279.58 314.53 349.47 kbp
WSB3122_assembly
N50
N75
0 0.5 1 1.5 2 2.5 2.8 Mbp
Contig info
<click on a contig to get details>
Legend
contigs
contig of length = Nx statistic (x is 50 or 75)

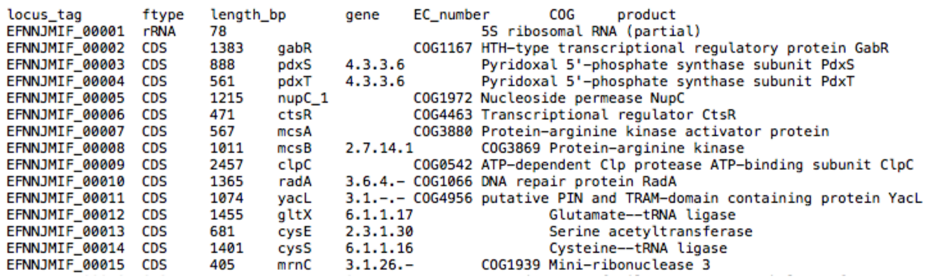
## Annotation

By default, the assembly is annotated with [Prokka](#) which acts as frontend for several annotation tools and includes rRNA and ORF predictions. Alternatively, on request, the assembly is annotated with [Bakta](#) or [DFAST](#).

▼ Output files

- `Prokka/{ID}/`
- `*.gff`: Annotation in gff format
- `*.txt`: Annotation in text format
- `*.faa`: Protein sequences in fasta format

See Prokka's documentation for a full description of all output files.



```
locus_tag       ftype   length_bp       gene    EC_number       COG     product
EFNNJMIF_00001  rRNA    78                              5S ribosomal RNA (partial)
EFNNJMIF_00002  CDS     1383    gabR            COG1167 HTH-type transcriptional regulatory protein GabR
EFNNJMIF_00003  CDS     888     pdxS    4.3.3.6         Pyridoxal 5'-phosphate synthase subunit PdxS
EFNNJMIF_00004  CDS     561     pdxT    4.3.3.6         Pyridoxal 5'-phosphate synthase subunit PdxT
EFNNJMIF_00005  CDS     1215    nupC_1          COG1972 Nucleoside permease NupC
EFNNJMIF_00006  CDS     471     ctsR            COG4463 Transcriptional regulator CtsR
EFNNJMIF_00007  CDS     567     mcsA            COG3880 Protein-arginine kinase activator protein
EFNNJMIF_00008  CDS     1011    mcsB    2.7.14.1        COG3869 Protein-arginine kinase
EFNNJMIF_00009  CDS     2457    clpC            COG0542 ATP-dependent Clp protease ATP-binding subunit ClpC
EFNNJMIF_00010  CDS     1365    radA    3.6.4.- COG1066 DNA repair protein RadA
EFNNJMIF_00011  CDS     1074    yacL    3.1.-.- COG4956 putative PIN and TRAM-domain containing protein YacL
EFNNJMIF_00012  CDS     1455    gltX    6.1.1.17                Glutamate--tRNA ligase
EFNNJMIF_00013  CDS     681     cysE    2.3.1.30                Serine acetyltransferase
EFNNJMIF_00014  CDS     1401    cysS    6.1.1.16                Cysteine--tRNA ligase
EFNNJMIF_00015  CDS     405     mrnC    3.1.26.-        COG1939 Mini-ribonuclease 3
```

- Bakta/{ID}/
- *.gff3: Annotations in gff3 format
- *.txt: Summary in txt format
- *.faa: CDS/sORF amino acid sequences in fasta format

See Baktas's documentation for a full description of all output files.

- DFAST/{ID}_results/
- genome.gff: Annotation in gff format
- statistics.txt: Annotation statistics in text format
- protein.faa: Protein sequences in fasta format

## Report

Some pipeline results are visualised by MultiQC, which is a visualisation tool that generates a single HTML report summarising all samples in your project. Further statistics are available in within the report data directory.

MultiQC is a visualization tool that generates a single HTML report summarising all samples in your project. Most of the pipeline QC results are visualised in the report and further statistics are available in the report data directory.

The pipeline has special steps which also allow the software versions to be reported in the MultiQC output for future traceability.

Results generated by MultiQC collate pipeline QC from supported tools e.g. FastQC. The pipeline has special steps which also allow the software versions to be reported in the MultiQC output for future traceability. For more information about how to use MultiQC reports, see http://multiqc.info.

▼ Output files

- multiqc/
- multiqc_report.html: a standalone HTML file that can be viewed in your web browser.
- multiqc_data/: directory containing parsed statistics from the different tools used in the pipeline.
- multiqc_plots/: directory containing static images from the report in various formats.
- summary_assembly_metrics_mqc.csv: custom table containing most relevant assembly QC metrics.

### Pipeline information

Nextflow provides excellent functionality for generating various reports relevant to the running and execution of the pipeline. This will allow you to troubleshoot errors with the running of the pipeline, and also provide you with other information such as launch commands, run times and resource usage.

▼ Output files

- pipeline_info/
- Reports generated by Nextflow: execution_report.html, execution_timeline.html, execution_trace.txt and pipeline_dag.dot/pipeline_dag.svg.
- Reports generated by the pipeline: pipeline_report.html, pipeline_report.txt and software_versions.yml. The pipeline_report* files will only be present if the --email / --email_on_fail parameter's are used when running the pipeline.
- Reformatted samplesheet files used as input to the pipeline: samplesheet.valid.csv.
- Parameters used by the pipeline run: params.json.

# nf-core/viralrecon Description

This document describes the output produced by the pipeline. Most of the plots are taken from the MultiQC report, which summarises results at the end of the pipeline.

The directories listed below will be created in the results directory after the pipeline has finished. All paths are relative to the top-level results directory.

# Illumina: Pipeline overview

## Illumina: Preprocessing

### cat

▼ Output files

- `fastq/`
- `*.merged.fastq.gz`: These files are not saved by default but can be via a custom config file such as the one below.

```
params {
    modules {
        'illumina_cat_fastq' {
            publish_files = null
        }
    }
}
```

If multiple libraries/runs have been provided for the same sample in the input samplesheet (e.g. to increase sequencing depth) then these will be merged at the very beginning of the pipeline in order to have consistent sample naming throughout the pipeline. Please refer to the usage documentation to see how to specify these samples in the input samplesheet.

### FastQC

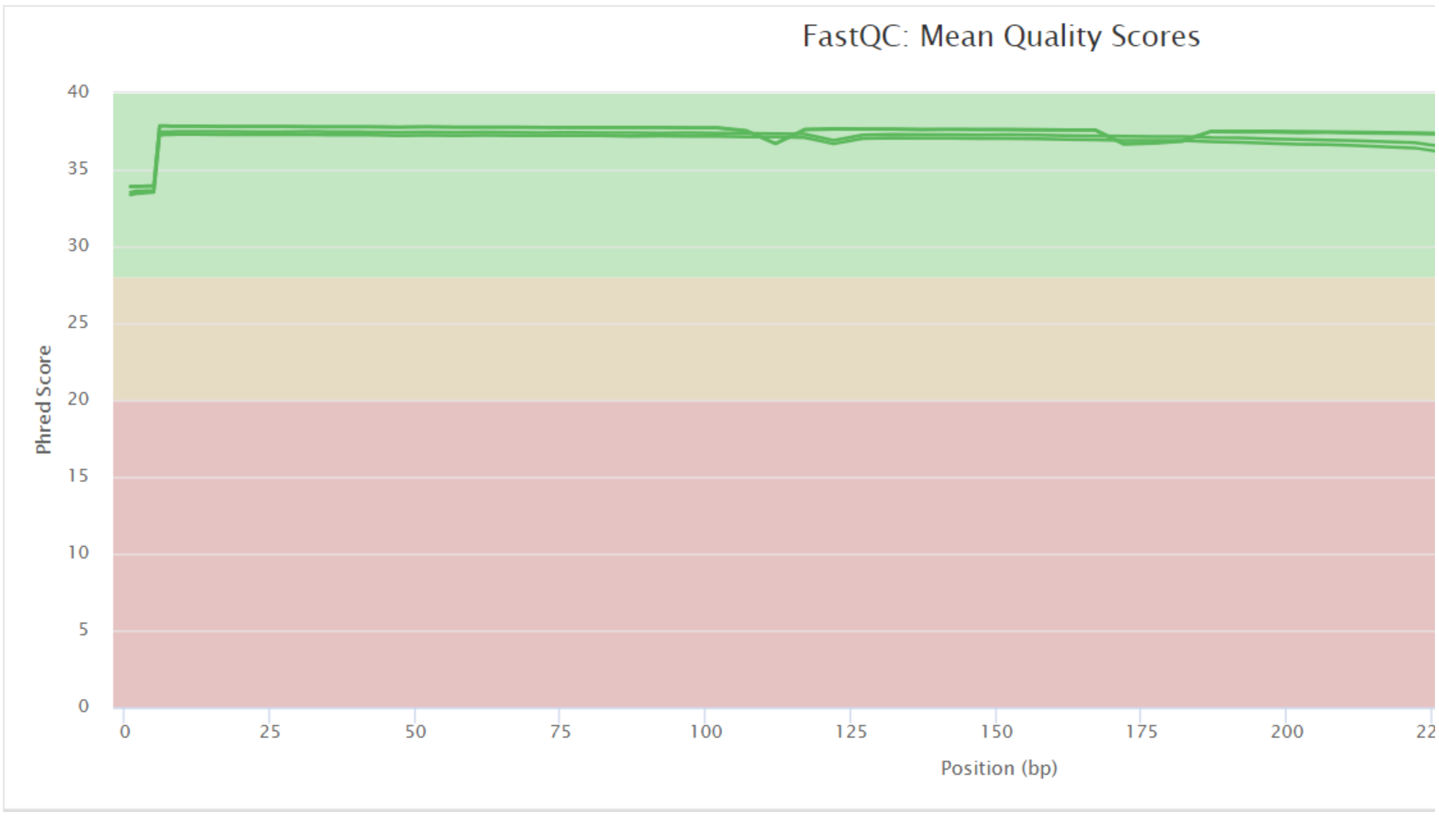
▼ Output files

- `fastqc/raw/`
- `*_fastqc.html`: FastQC report containing quality metrics.

- *_fastqc.zip: Zip archive containing the FastQC report, tab-delimited data file and plot images.

**NB:** The FastQC plots in this directory are generated relative to the raw, input reads. They may contain adapter sequence and regions of low quality. To see how your reads look after trimming please refer to the FastQC reports in the fastqc/trim/ directory.

FastQC gives general quality metrics about your sequenced reads. It provides information about the quality score distribution across your reads, per base sequence content (%A/T/G/C), adapter contamination and overrepresented sequences. For further reading and documentation see the FastQC help pages.

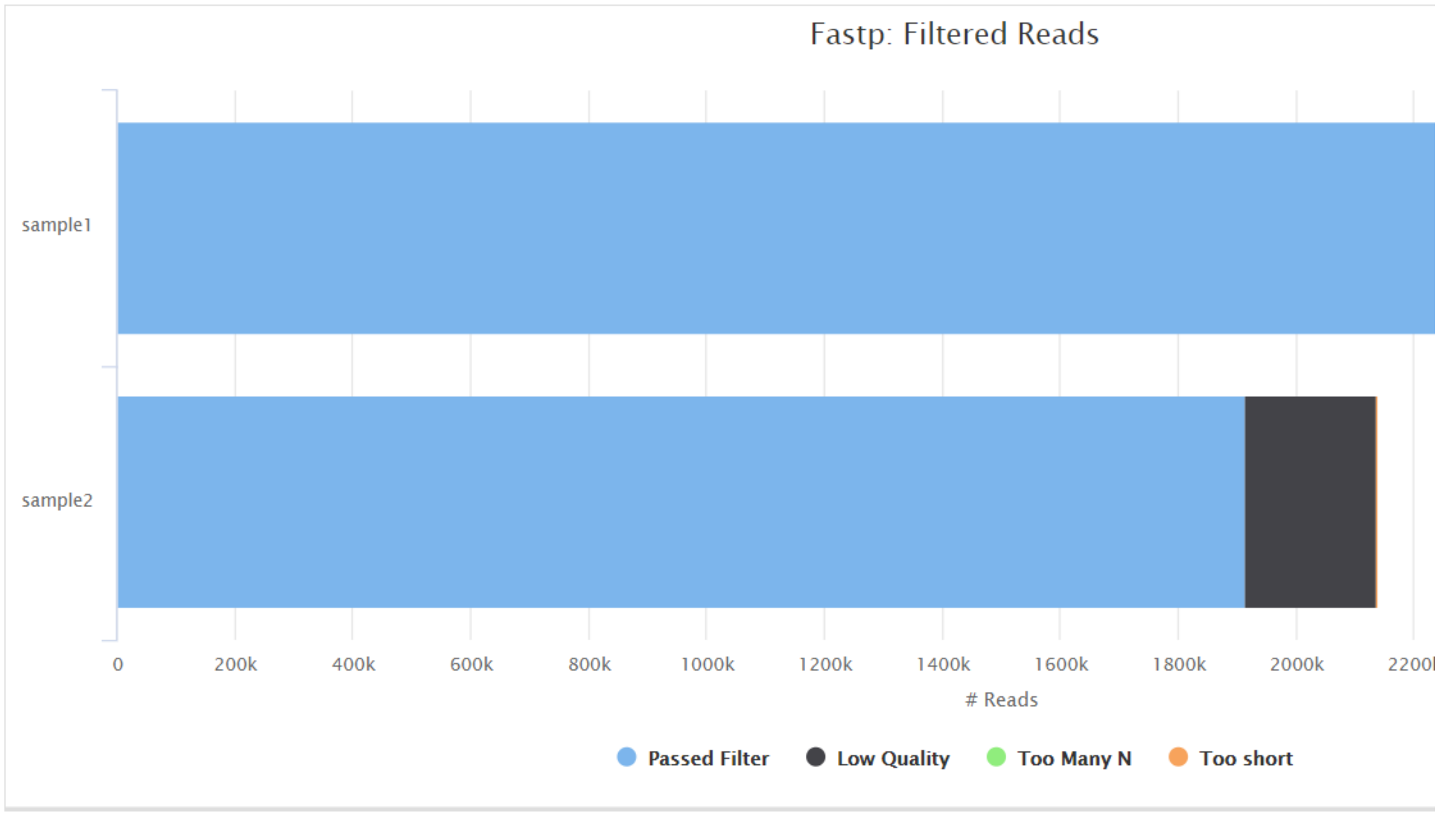


## fastp

▼ Output files

- fastp/
- *.fastp.html: Trimming report in html format.
- *.fastp.json: Trimming report in json format.
- fastp/log/
- *.fastp.log: Trimming log file.
- fastqc/trim/
- *_fastqc.html: FastQC report of the trimmed reads.
- *_fastqc.zip: Zip archive containing the FastQC report, tab-delimited data file and plot images.

fastp is a tool designed to provide fast, all-in-one preprocessing for FastQ files. It has been developed in C++ with multithreading support to achieve higher performance. fastp is used in this pipeline for standard adapter trimming and quality filtering.



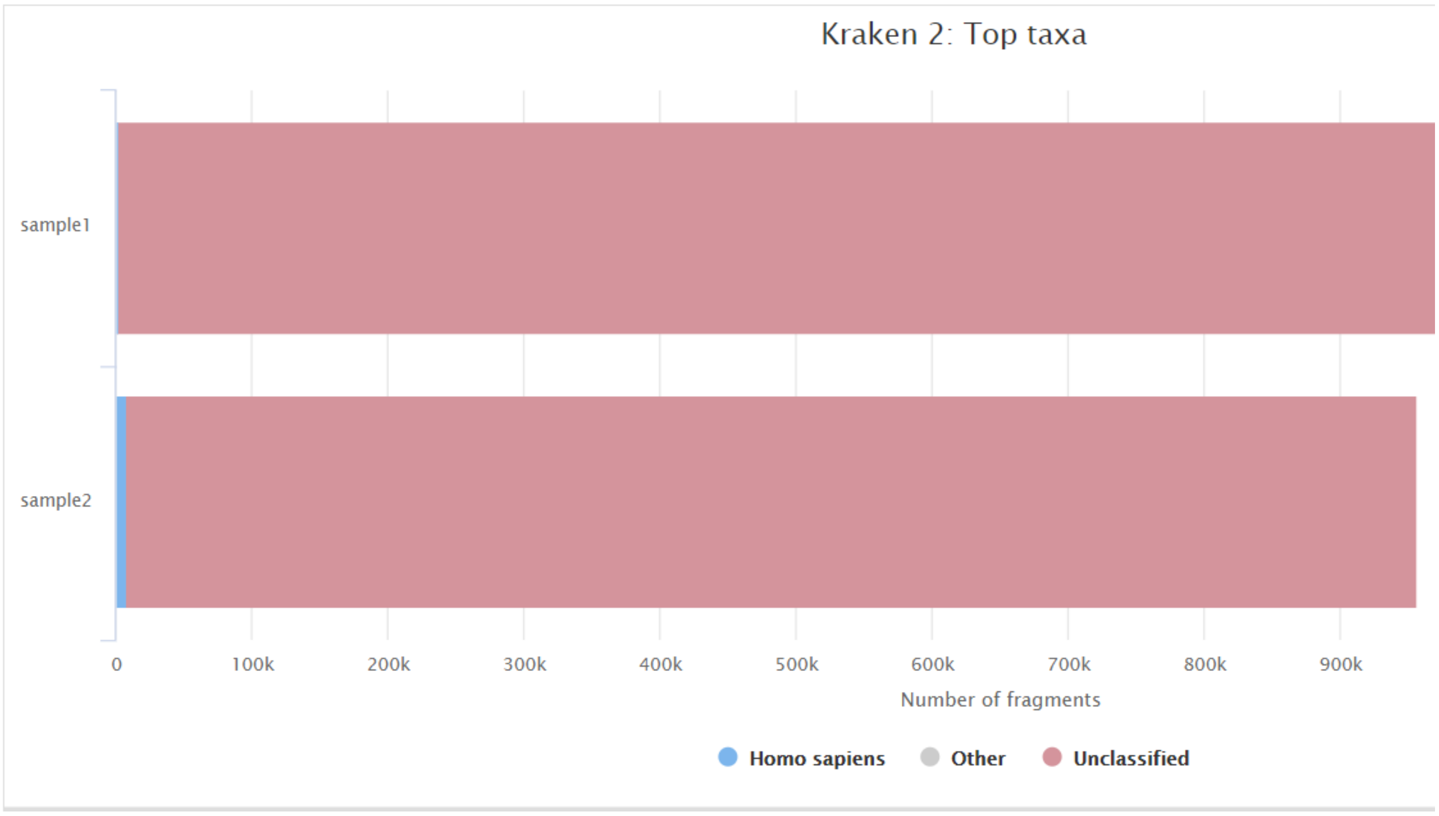
## Kraken 2

▼ Output files

- `kraken2/`
- `*.kraken2.report.txt`: Kraken 2 taxonomic report. See [here](#) for a detailed description of the format.

[Kraken 2](#) is a sequence classifier that assigns taxonomic labels to DNA sequences. Kraken 2 examines the k-mers within a query sequence and uses the information within those k-mers to query a database. That database maps k-mers to the lowest common ancestor (LCA) of all genomes known to contain a given k-mer.

We use a Kraken 2 database in this workflow to filter out reads specific to the host genome before performing the *de novo* assembly steps in the pipeline. This filtering is not performed in the variant calling arm of the pipeline by default but Kraken 2 is still run to obtain an estimate of host reads, however, the filtering can be amended via the `--kraken2_variants_host_filter` parameter.



## Illumina: Variant calling

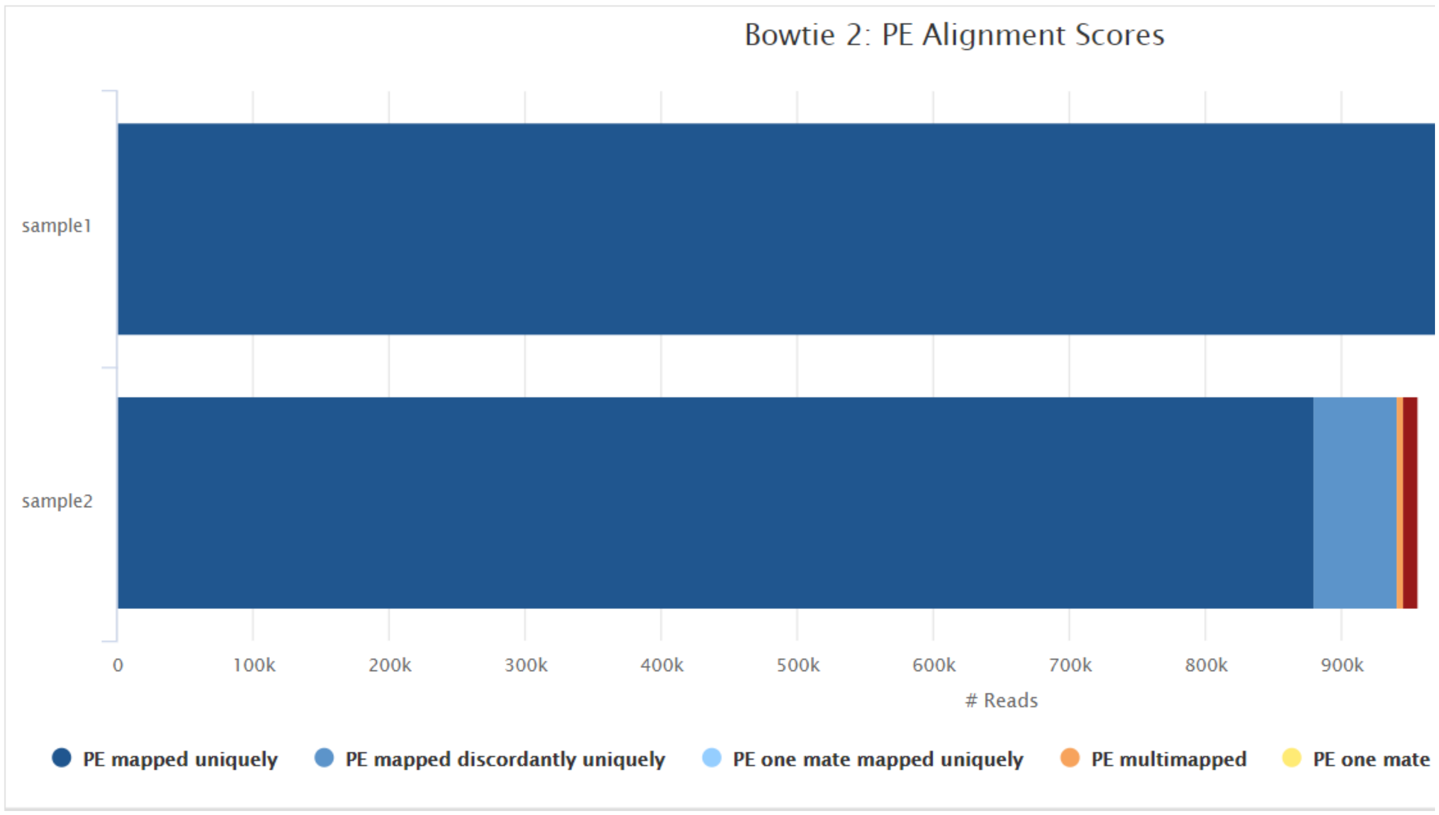
A file called summary_variants_metrics_mqc.csv containing a selection of read alignment and variant calling metrics will be saved in themultiqc/ results directory. The same metrics will also be added to the top of the MultiQC report.

## Bowtie 2

▼ Output files

- variants/bowtie2/log/
- *.bowtie2.log: Bowtie 2 mapping log file.

Bowtie 2 is an ultrafast and memory-efficient tool for aligning sequencing reads to long reference sequences. Bowtie 2 supports gapped, local, and paired-end alignment modes.

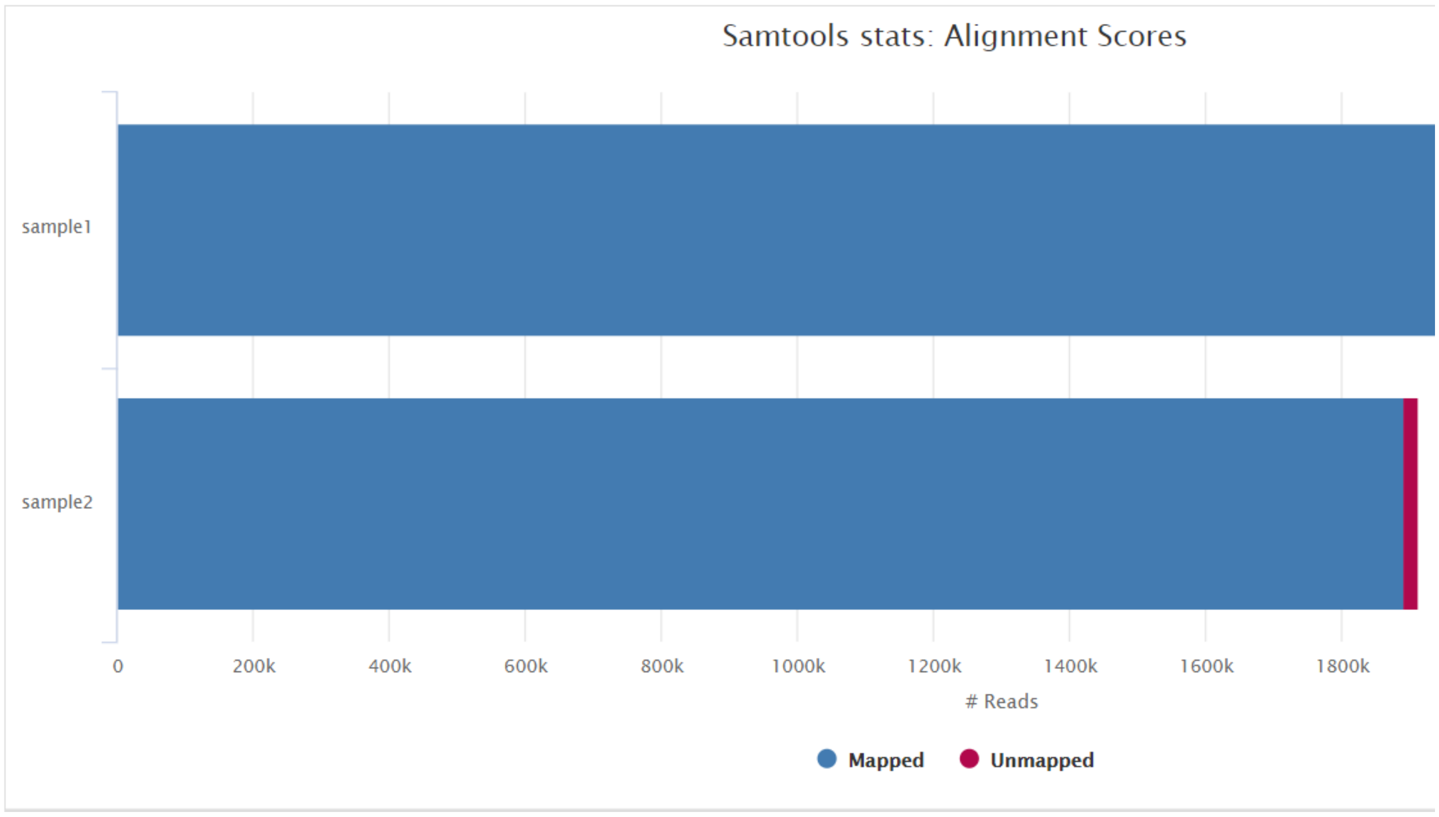


## SAMtools

▼ Output files

- variants/bowtie2/
- <SAMPLE>.sorted.bam: Coordinate sorted BAM file containing read alignment information.
- <SAMPLE>.sorted.bam.bai: Index file for coordinate sorted BAM file.
- variants/bowtie2/samtools_stats/
- SAMtools <SAMPLE>.sorted.bam.flagstat, <SAMPLE>.sorted.bam.idxstats and <SAMPLE>.sorted.bam.stats files generated from the alignment files.

Bowtie 2 BAM files are further processed withSAMtools to sort them by coordinate, for indexing, as well as to generate read mapping statistics.



### iVar trim

▼ Output files

- variants/bowtie2/
- *.ivar_trim.sorted.bam: Coordinate sorted BAM file after primer trimming.
- *.ivar_trim.sorted.bam.bai: Index file for coordinate sorted BAM file after primer trimming.
- variants/bowtie2/samtools_stats/
- SAMtools *.ivar_trim.sorted.bam.flagstat, *.ivar_trim.sorted.bam.idxstats and *.ivar_trim.sorted.bam.stats files generated from the primer trimmed alignment files.
- variants/bowtie2/log/
- *.ivar_trim.ivar.log: iVar trim log file obtained from stdout.

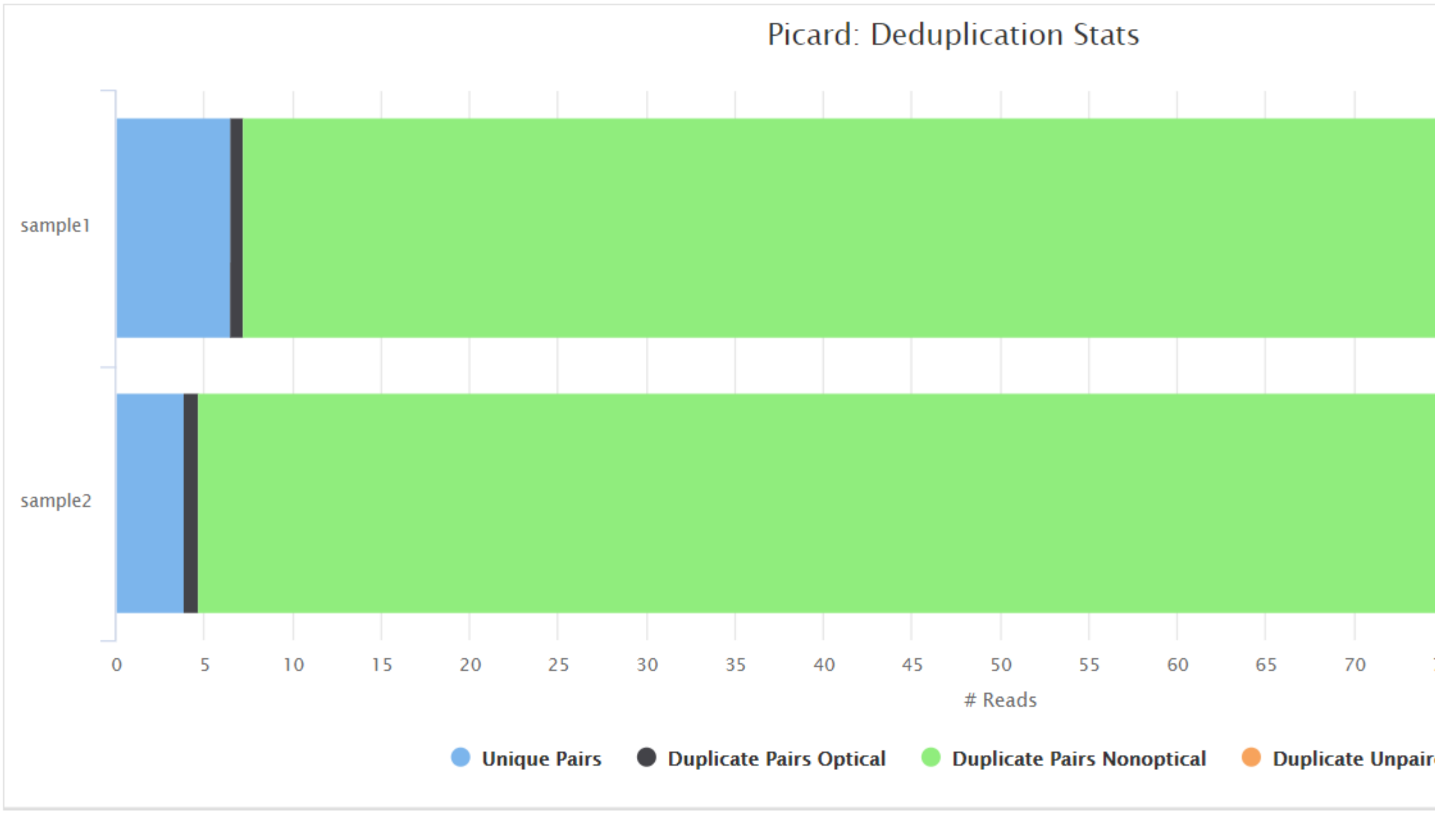
If the --protocol amplicon parameter is provided then iVar is used to trim amplicon primer sequences from the aligned reads. iVar uses the primer positions supplied in --primer_bed to soft clip primer sequences from a coordinate sorted BAM file.

### picard MarkDuplicates

▼ Output files

- variants/bowtie2/
- *.markduplicates.sorted.bam: Coordinate sorted BAM file after duplicate marking.
- *.markduplicates.sorted.bam.bai: Index file for coordinate sorted BAM file after duplicate marking.
- variants/bowtie2/samtools_stats/
- SAMtools *.markduplicates.sorted.bam.flagstat, *.markduplicates.sorted.bam.idxstats and *.markduplicates.sorted.bam.stats files generated from the duplicate marked alignment files.
- variants/bowtie2/picard_metrics/
- *.markduplicates.sorted.MarkDuplicates.metrics.txt: Metrics file from MarkDuplicates.

Unless you are using UMIs it is not possible to establish whether the fragments you have sequenced from your sample were derived via true biological duplication (i.e. sequencing independent template fragments) or as a result of PCR biases introduced during the library preparation. picard MarkDuplicates isn't run by default because you anticipate high levels of duplication with viral data due to the size of the genome, however, you can activate it by adding --skip_markduplicates false to the command you use to run the pipeline. This will only*mark* the duplicate reads identified amongst the alignments to allow you to guage the overall level of duplication in your samples. You can also choose to remove any reads identified as duplicates via the --filter_duplicates parameter.

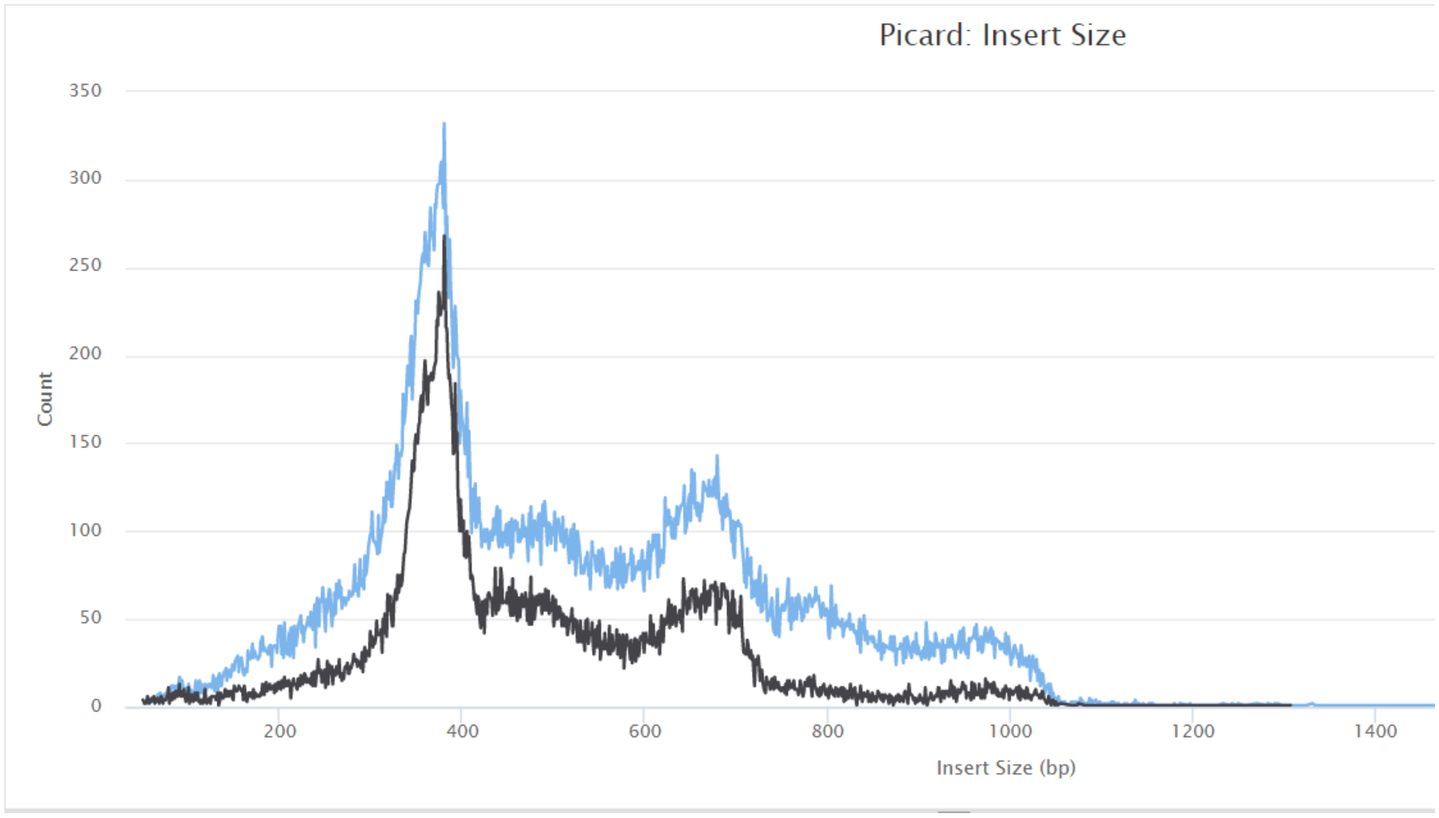


### picard CollectMultipleMetrics

▼ Output files

- variants/bowtie2/picard_metrics/
- *.CollectMultipleMetrics.*: Alignment QC files from picard CollectMultipleMetrics in*_metrics textual format.
- variants/bowtie2/picard_metrics/pdf/
- *.pdf plots for metrics obtained from CollectMultipleMetrics.

[picard-tools](picard-tools) is a set of command-line tools for manipulating high-throughput sequencing data. We use picard-tools in this pipeline to obtain mapping and coverage metrics.



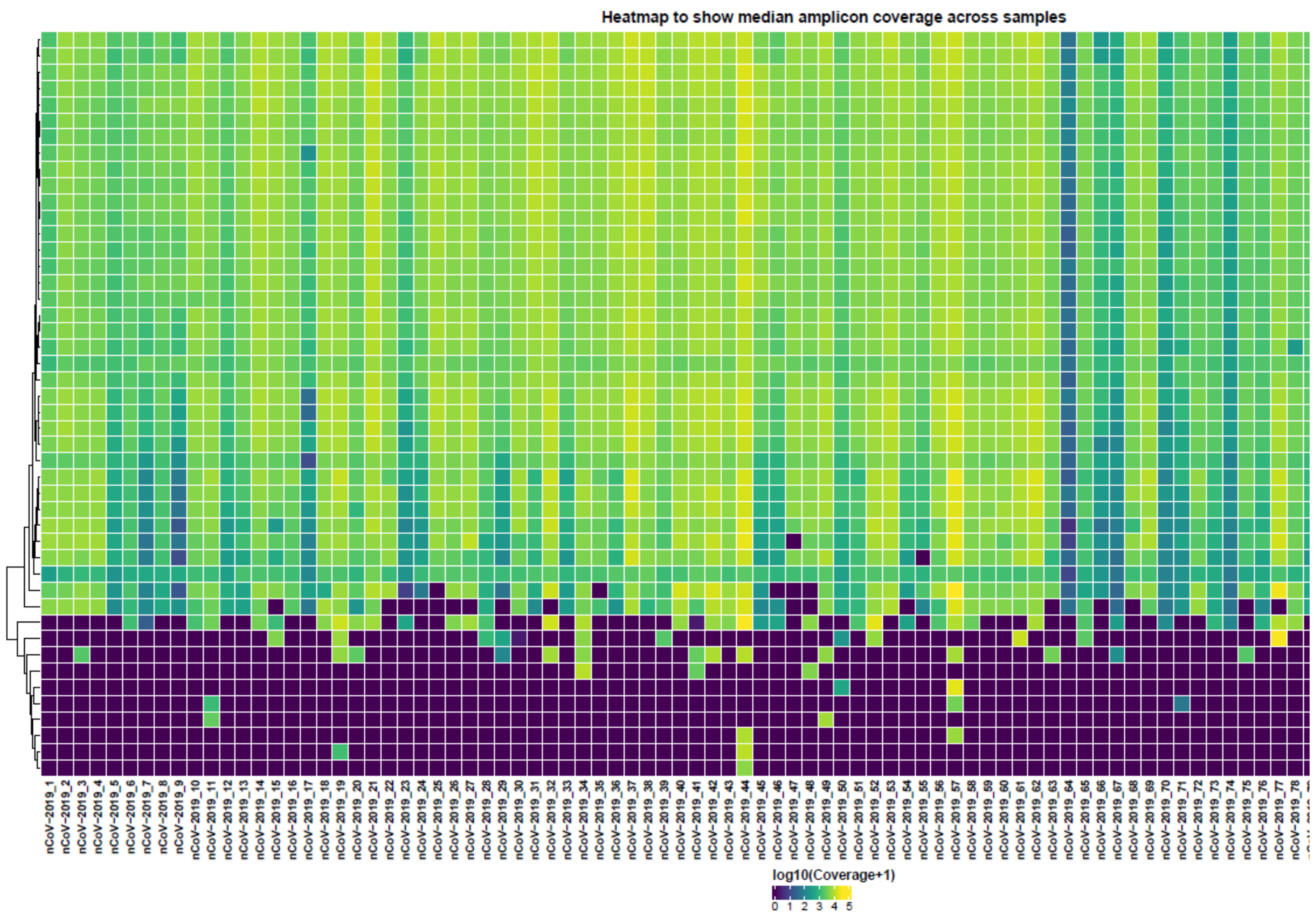
### mosdepth

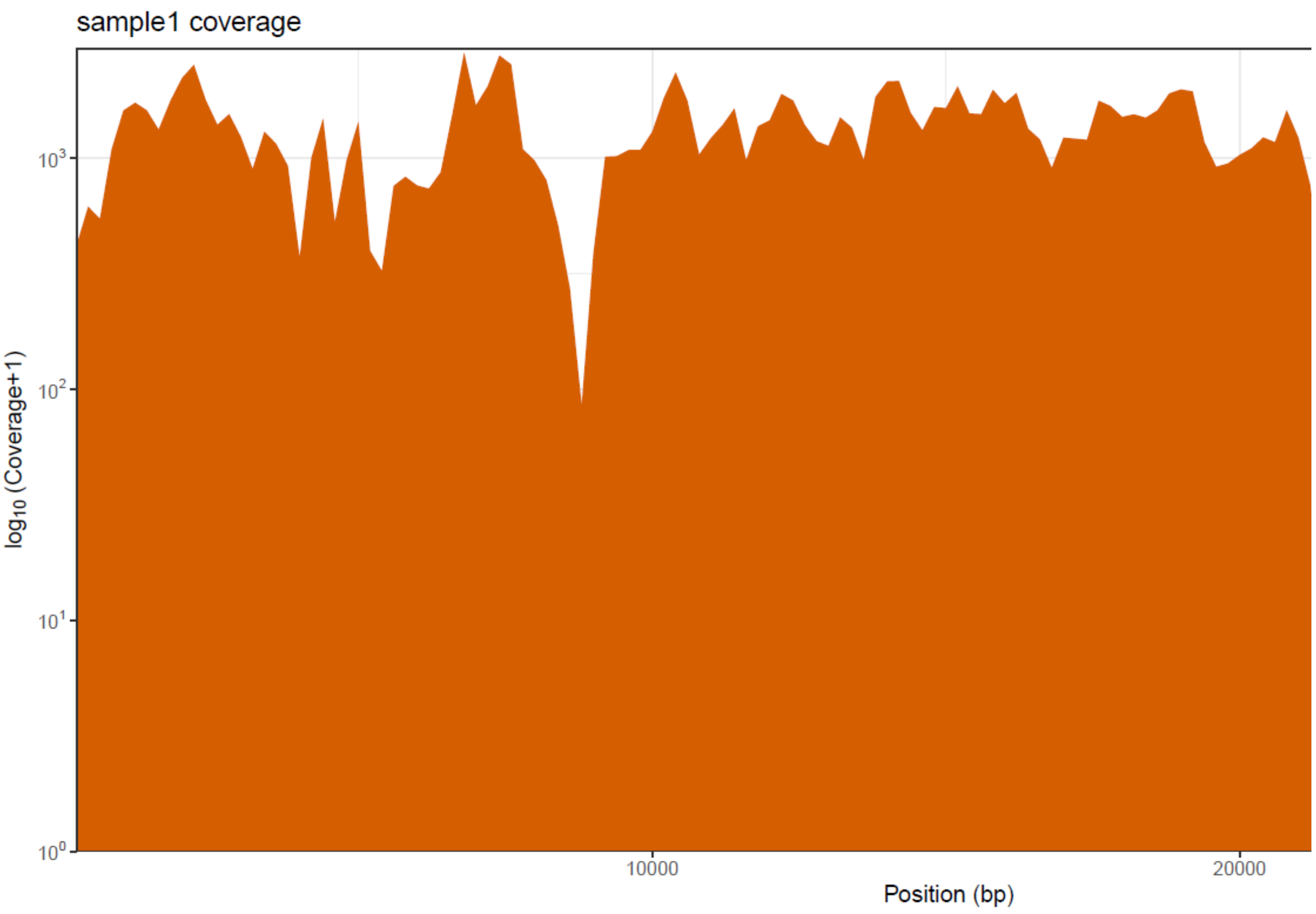
▼ Output files

- variants/bowtie2/mosdepth/genome/
- all_samples.mosdepth.coverage.tsv: File aggregating genome-wide coverage values across all samples used for plotting.

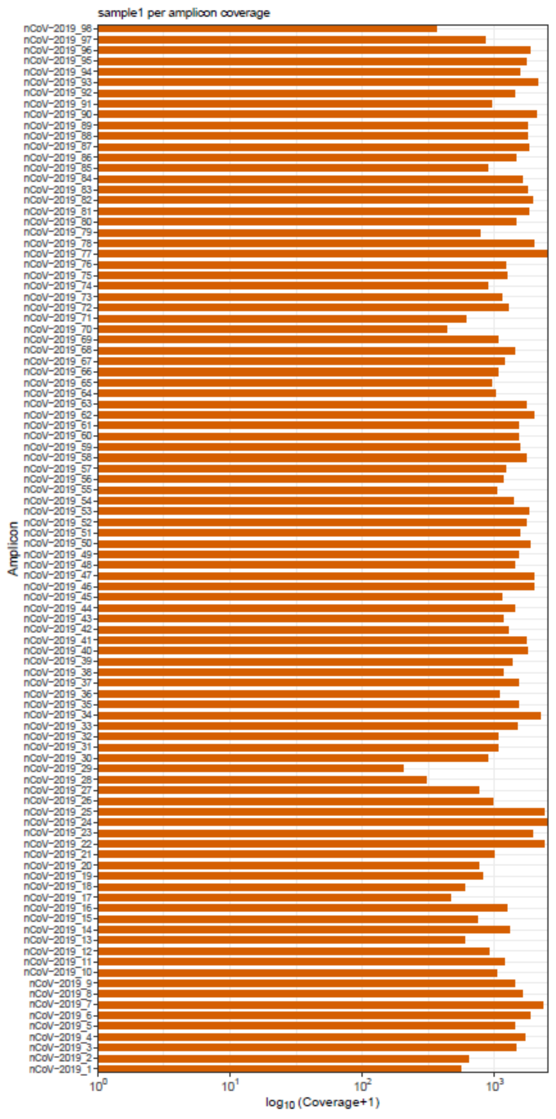
- `*.mosdepth.coverage.pdf`: Whole-genome coverage plot.
- `*.mosdepth.coverage.tsv`: File containing coverage values for the above plot.
- `*.mosdepth.summary.txt`: Summary metrics including mean, min and max coverage values.
- `variants/bowtie2/mosdepth/amplicon/`
- `all_samples.mosdepth.coverage.tsv`: File aggregating per-amplicon coverage values across all samples used for plotting.
- `all_samples.mosdepth.heatmap.pdf`: Heatmap showing per-amplicon coverage across all samples.
- `*.mosdepth.coverage.pdf`: Bar plot showing per-amplicon coverage for an individual sample.
- `*.mosdepth.coverage.tsv`: File containing per-amplicon coverage values for the above plot.
- `*.mosdepth.summary.txt`: Summary metrics including mean, min and max coverage values.

mosdepth is a fast BAM/CRAM depth calculation for WGS, exome, or targeted sequencing. mosdepth is used in this pipeline to obtain genome-wide coverage values in 200bp windows and for `--protocol amplicon` to obtain amplicon/region-specific coverage metrics. The results are then either rendered in MultiQC (genome-wide coverage) or are plotted using custom `R` scripts.





sample1 coverage
log10 (Coverage+1)
10^3
10^2
10^1
10^0
10000
20000
Position (bp)



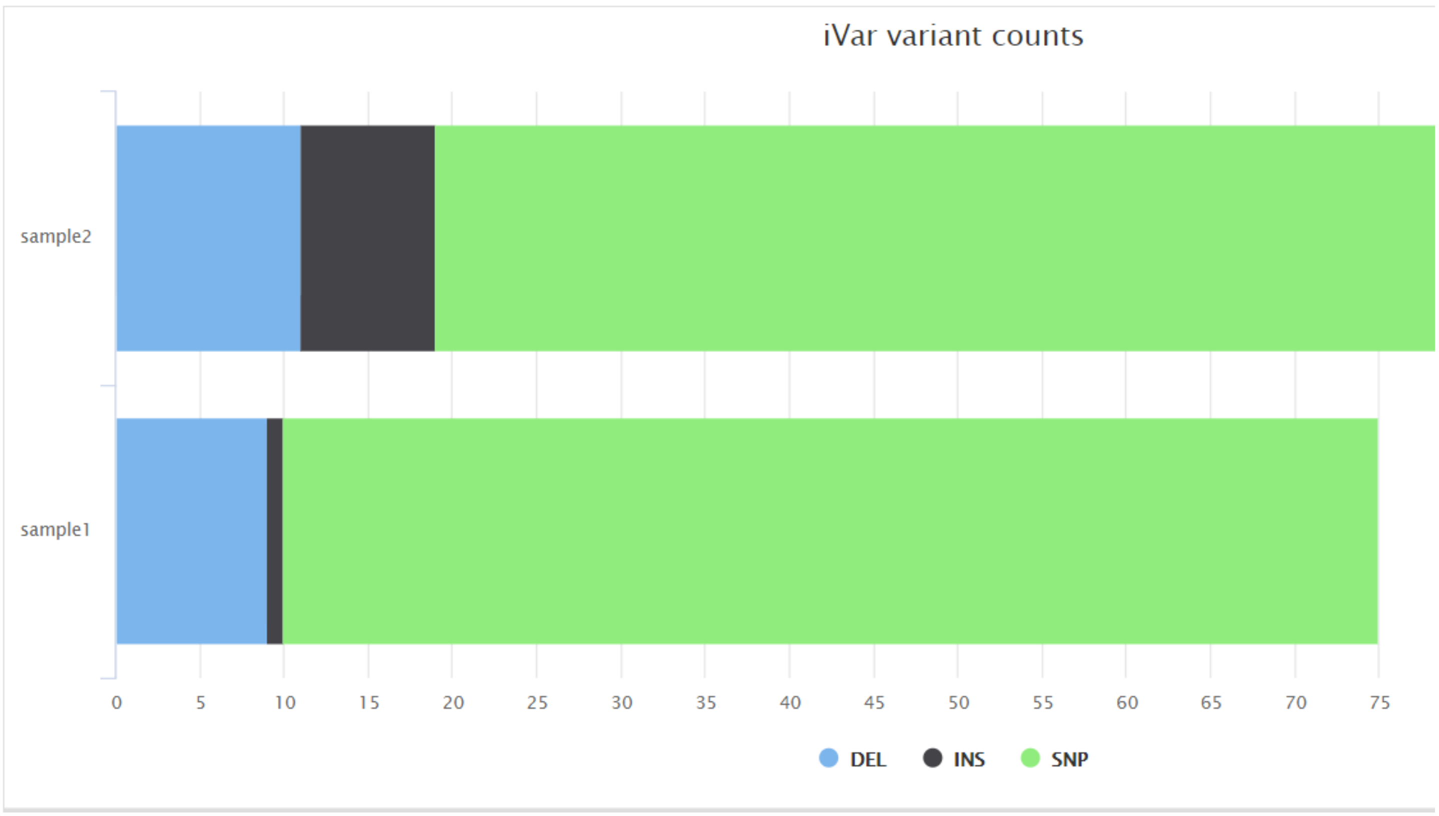
### iVar variants

▼ Output files

- variants/ivar/
- *.tsv: Original iVar variants in TSV format.
- *.vcf.gz: iVar variants in VCF format. Converted using customivar_variants_to_vcf.py python script.
- *.vcf.gz.tbi: iVar variants VCF index file.
- variants/ivar/log/
- *.variant_counts.log: Counts for type of variants called by iVar.
- variants/ivar/bcftools_stats/
- *.bcftools_stats.txt: Statistics and counts obtained from iVar variants VCF file.

iVar is a computational package that contains functions broadly useful for viral amplicon-based sequencing. We use iVar in this pipeline totrim primer sequences for amplicon input data as well as to call variants.

iVar outputs a tsv format which is not compatible with downstream analysis such as annotation using SnpEff. Moreover some issues need to be addressed such as strand-bias filtering and the consecutive reporting of variants belonging to the same codon. This pipeline uses a custom Python script ivar_variants_to_vcf.py to convert the default iVar output to VCF whilst also addressing both of these issues.

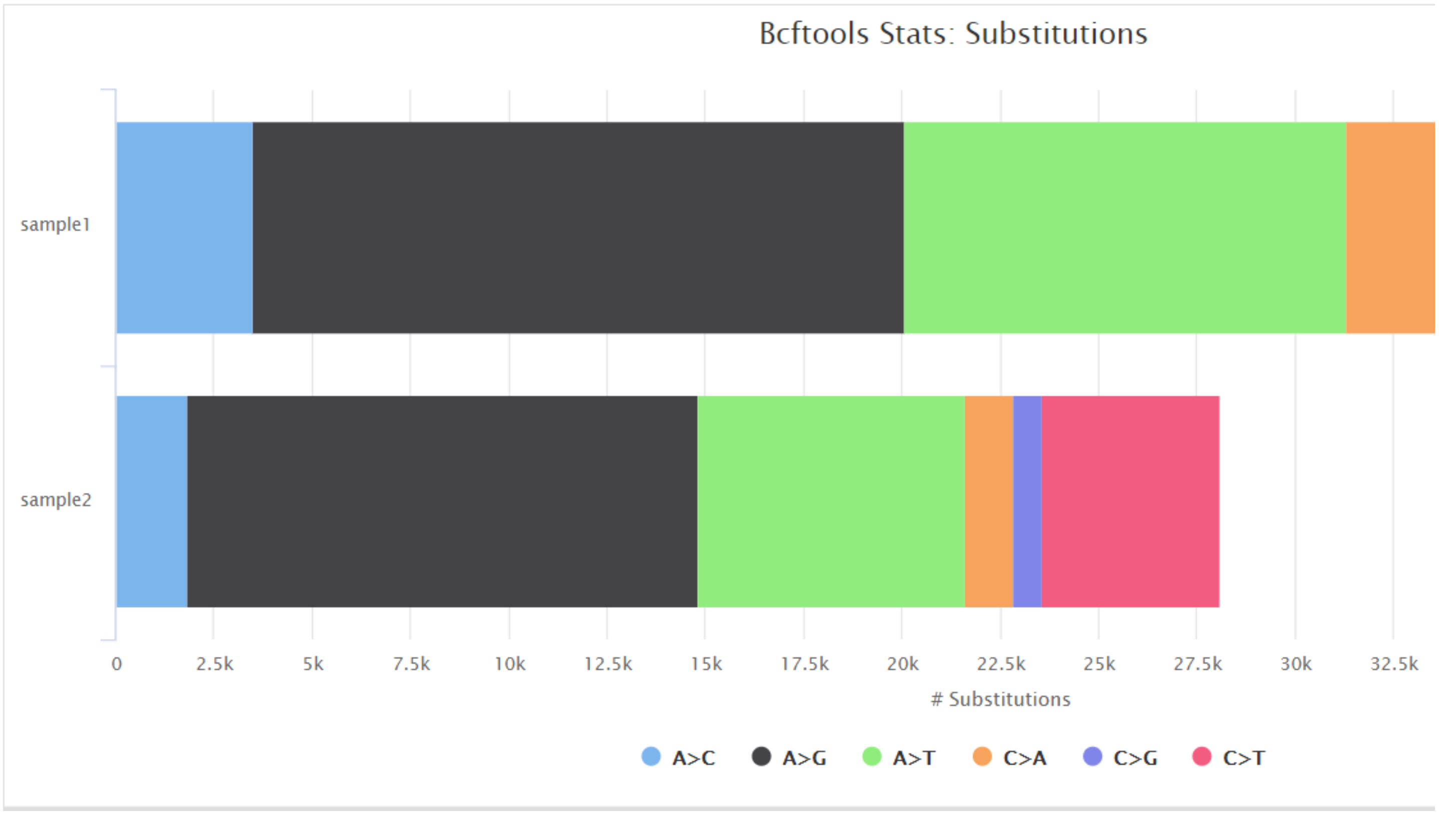


### BCFTools call

▼ Output files

- variants/bcftools/
- *.vcf.gz: Variants VCF file.
- *.vcf.gz.tbi: Variants VCF index file.
- variants/bcftools/bcftools_stats/
- *.bcftools_stats.txt: Statistics and counts obtained from VCF file.

BCFtools can be used to call variants directly from BAM alignment files. It is a set of utilities that manipulate variant calls in VCF and its binary counterpart BCF format. BCFTools is used in the variant calling and *de novo* assembly steps of this pipeline to obtain basic statistics from the VCF output.



### SnpEff and SnpSift

▼ Output files

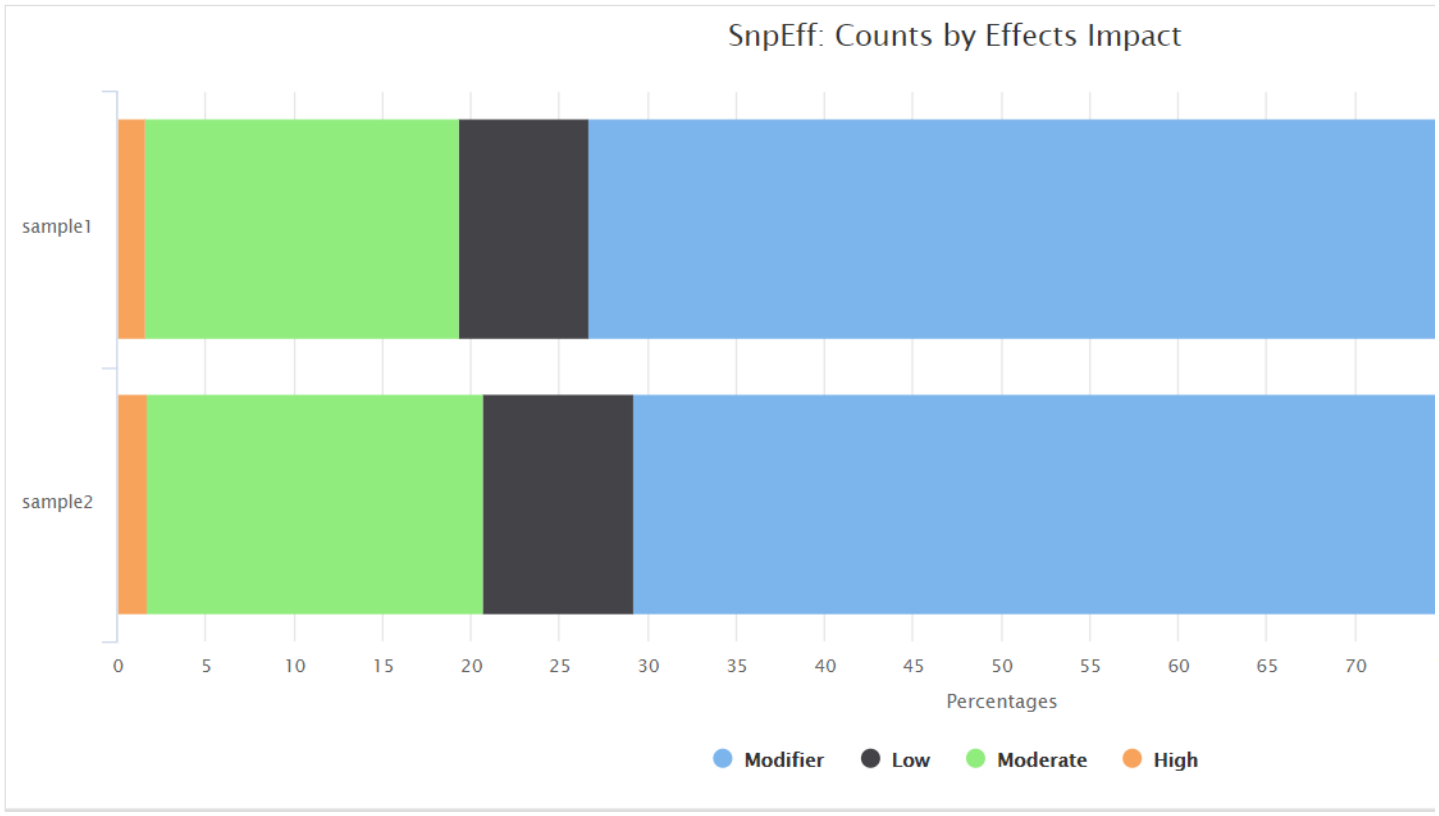
- variants/<CALLER>/snpeff/

- `*.snpeff.csv`: Variant annotation csv file.
- `*.snpeff.genes.txt`: Gene table for annotated variants.
- `*.snpeff.summary.html`: Summary html file for variants.
- `*.snpeff.vcf.gz`: VCF file with variant annotations.
- `*.snpeff.vcf.gz.tbi`: Index for VCF file with variant annotations.
- `*.snpsift.txt`: SnpSift summary table.
- `variants/<CALLER>/snpeff/bcftools_stats/`
- `*.bcftools_stats.txt`: Statistics and counts obtained from VCF file.

**NB:** The value of `<CALLER>` in the output directory name above is determined by the `--variant_caller` parameter (Default: 'ivar' for '--protocol amplicon' and 'bcftools' for '--protocol metagenomic').

SnpEff is a genetic variant annotation and functional effect prediction toolbox. It annotates and predicts the effects of genetic variants on genes and proteins (such as amino acid changes).

SnpSift annotates genomic variants using databases, filters, and manipulates genomic annotated variants. After annotation with SnpEff, you can use SnpSift to help filter large genomic datasets in order to find the most significant variants.



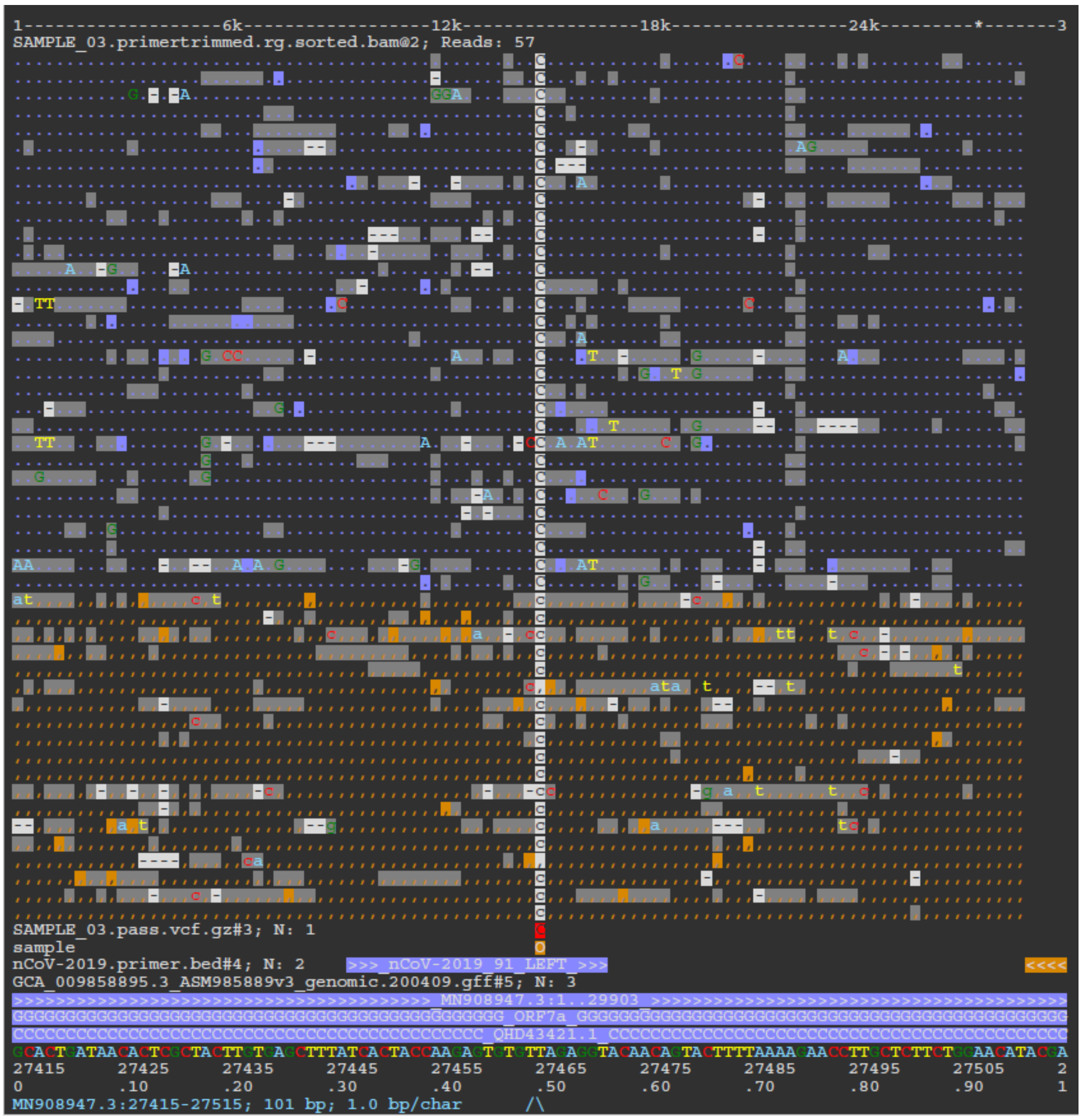
## ASCIIGenome

▼ Output files

- `variants/<CALLER>/asciigenome/<SAMPLE>/`
- `*.pdf`: Individual variant screenshots with annotation tracks in PDF format.

**NB:** The value of `<CALLER>` in the output directory name above is determined by the `--variant_caller` parameter (Default: 'ivar' for '--protocol amplicon' and 'bcftools' for '--protocol metagenomic').

As described in the documentation, ASCIIGenome is a command-line genome browser that can be run from a terminal window and is solely based on ASCII characters. The closest program to ASCIIGenome is probably samtools tview but ASCIIGenome offers much more flexibility, similar to popular GUI viewers like the IGV browser. We are using the batch processing mode of ASCIIGenome in this pipeline to generate individual screenshots for all of the variant sites reported for each sample in the VCF files. This is incredibly useful to be able to quickly QC the variants called by the pipeline without having to tediously load all of the relevant tracks into a conventional genome browser. Where possible, the BAM read alignments, VCF variant file, primer BED file and GFF annotation track will be represented in the screenshot for contextual purposes. The screenshot below shows a SNP called relative to the MN908947.3 SARS-CoV-2 reference genome that overlaps the ORF7a protein and the nCoV-2019_91_LEFT primer from the ARIC v3 protocol.


```
1-------------------6k-----------------12k----------------18k-----------------24k---------*-------3
SAMPLE_03.primertrimmed.rg.sorted.bam@2; Reads: 57
SAMPLE_03.pass.vcf.gz#3; N: 1
sample
nCoV-2019.primer.bed#4; N: 2     >>> nCoV-2019_91_LEFT >>>
GCA_009858895.3_ASM985889v3_genomic.200409.gff#5; N: 3
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> MN908947.3:1..29903 >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
GGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGG ORF7a GGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGG
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC QHD43421.1 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
GCACTGATAACACTCGCTACTTGTGAGCTTTATCACTACCAAGAGTGTGTTAGAGGTACAACAGTACTTTTAAAAGAACCTTGCTCTTCTGGAACATACGA
27415      27425      27435      27445      27455      27465      27475      27485      27495      27505      2
0          .10        .20        .30        .40        .50        .60        .70        .80        .90        1
MN908947.3:27415-27515; 101 bp; 1.0 bp/char       /\
```

## iVar consensus

▼ Output files

- variants/<CALLER>/consensus/ivar/
- *.consensus.fa: Consensus Fasta file generated by iVar.
- *.consensus.qual.txt: File with the average quality of each base in the consensus sequence.
- variants/<CALLER>/consensus/ivar/base_qc/
- *.ACTG_density.pdf: Plot showing density of ACGT bases within the consensus sequence.
- *.base_counts.pdf: Plot showing frequency and percentages of all bases in consensus sequence.
- *.base_counts.tsv: File containing frequency and percentages of all bases in consensus sequence.
- *.N_density.pdf: Plot showing density of N bases within the consensus sequence.
- *.N_run.tsv: File containing start positions and width of N bases in consensus sequence.

**NB:** The value of <CALLER> in the output directory name above is determined by the --variant_caller parameter (Default: 'ivar' for '--protocol amplicon' and 'bcftools' for '--protocol metagenomic').

As described in the iVar variants section, iVar can be used in this pipeline to call variants and for the consensus sequence generation.

## BCFTools and BEDTools

▼ Output files

- variants/<CALLER>/consensus/bcftools/
- *.consensus.fa: Consensus fasta file generated by integrating the high allele-frequency variants called by iVar/BCFTools into the reference genome.
- *.filtered.vcf.gz: VCF file containing high allele-frequency variants (default: >= 0.75) that were integrated into the consensus sequence.
- *.filtered.vcf.gz.tbi: Variants VCF index file for high allele frequency variants.
- variants/<CALLER>/consensus/bcftools/base_qc/
- *.ACTG_density.pdf: Plot showing density of ACGT bases within the consensus sequence.
- *.base_counts.pdf: Plot showing frequency and percentages of all bases in consensus sequence.
- *.base_counts.tsv: File containing frequency and percentages of all bases in consensus sequence.
- *.N_density.pdf: Plot showing density of N bases within the consensus sequence.
- *.N_run.tsv: File containing start positions and width of N bases in consensus sequence.

**NB:** The value of <CALLER> in the output directory name above is determined by the --variant_caller parameter (Default: 'ivar' for '--protocol amplicon' and 'bcftools' for '--protocol metagenomic').

BCFTools is used in the variant calling and *de novo* assembly steps of this pipeline to obtain basic statistics from the VCF output. It can also used be used to generate a consensus sequence by integrating variant calls into the reference genome. In this pipeline, we use `samtools mpileup` to create a mask using low coverage positions, and `bedtools maskfasta` to mask the genome sequences based on these intervals. Finally, `bcftools consensus` is used to generate the consensus by projecting the high allele frequency variants onto the masked genome reference sequence.

### QUAST

▼ Output files

- `variants/<VARIANT_CALLER>/consensus/<CONSENSUS_CALLER>/quast/`
- `report.html`: Results report in HTML format. Also available in various other file formats i.e. `report.pdf`, `report.tex`, `report.tsv` and `report.txt`.

**NB:** The value of `<VARIANT_CALLER>` in the output directory name above is determined by the `--variant_caller` parameter (Default: 'ivar' for '--protocol amplicon' and 'bcftools' for '--protocol metagenomic'). **NB:** The value of `<CONSENSUS_CALLER>` in the output directory name above is determined by the `--consensus_caller` parameter (Default: 'bcftools' for both '--protocol amplicon' and '--protocol metagenomic').

QUAST is used to generate a single report with which to evaluate the quality of the consensus sequence across all of the samples provided to the pipeline. The HTML results can be opened within any browser (we recommend using Google Chrome). Please see the QUAST output docs for more detailed information regarding the output files.

### Pangolin

▼ Output files

- `variants/<VARIANT_CALLER>/consensus/<CONSENSUS_CALLER>/pangolin/`
- `*.pangolin.csv`: Lineage analysis results from Pangolin.

**NB:** The value of `<VARIANT_CALLER>` in the output directory name above is determined by the `--variant_caller` parameter (Default: 'ivar' for '--protocol amplicon' and 'bcftools' for '--protocol metagenomic'). **NB:** The value of `<CONSENSUS_CALLER>` in the output directory name above is determined by the `--consensus_caller` parameter (Default: 'bcftools' for both '--protocol amplicon' and '--protocol metagenomic').

Phylogenetic Assignment of Named Global Outbreak LINeages (Pangolin) has been used extensively during the COVID-19 pandemic in order to to assign lineages to SARS-CoV-2 genome sequenced samples. A web application also exists that allows users to upload genome sequences via a web browser to assign lineages to genome sequences of SARS-CoV-2, view descriptive characteristics of the assigned lineage(s), view the placement of the lineage in a phylogeny of global samples, and view the temporal and geographic distribution of the assigned lineage(s).

### Nextclade

▼ Output files

- `variants/<VARIANT_CALLER>/consensus/<CONSENSUS_CALLER>/nextclade/`
- `*.csv`: Analysis results from Nextlade containing genome clade assignment, mutation calling and sequence quality checks.

**NB:** The value of `<VARIANT_CALLER>` in the output directory name above is determined by the `--variant_caller` parameter (Default: 'ivar' for '--protocol amplicon' and 'bcftools' for '--protocol metagenomic'). **NB:** The value of `<CONSENSUS_CALLER>` in the output directory name above is determined by the `--consensus_caller` parameter (Default: 'bcftools' for both '--protocol amplicon' and '--protocol metagenomic').

Nextclade performs viral genome clade assignment, mutation calling and sequence quality checks for the consensus sequences generated in this pipeline. Similar to Pangolin, it has been used extensively during the COVID-19 pandemic. A web application also exists that allows users to upload genome sequences via a web browser.

### Variants long table

▼ Output files

- `variants/<VARIANT_CALLER>/`
- `variants_long_table.csv`: Long format table collating per-sample information for individual variants, functional effect prediction and lineage analysis.

**NB:** The value of `<VARIANT_CALLER>` in the output directory name above is determined by the `--variant_caller` parameter (Default: 'ivar' for '--protocol amplicon' and 'bcftools' for '--protocol metagenomic').

Create variants long format table collating per-sample information for individual variants (BCFTools), functional effect prediction (SnpSift) and lineage analysis (Pangolin).

The more pertinent variant information is summarised in this table to make it easier for researchers to assess the impact of variants found amongst the sequenced sample(s). An example of the fields included in the table are shown below:

```
SAMPLE,CHROM,POS,REF,ALT,FILTER,DP,REF_DP,ALT_DP,AF,GENE,EFFECT,HGVS_C,HGVS_P,HGVS_P_1LETTER,CALLER,LINEAGE
SAMPLE1_PE,MN908947.3,241,C,T,PASS,489,4,483,0.99,orf1ab,upstream_gene_variant,c.-25C>T,.,.,ivar,B.1
SAMPLE1_PE,MN908947.3,1875,C,T,PASS,92,62,29,0.32,orf1ab,missense_variant,c.1610C>T,p.Ala537Val,p.A537V,ivar,B.1
SAMPLE1_PE,MN908947.3,3037,C,T,PASS,213,0,213,1.0,orf1ab,synonymous_variant,c.2772C>T,p.Phe924Phe,p.F924F,ivar,B.1
SAMPLE1_PE,MN908947.3,11719,G,A,PASS,195,9,186,0.95,orf1ab,synonymous_variant,c.11454G>A,p.Gln3818Gln,p.Q3818Q,ivar,B.1
```

# Illumina: De novo assembly

A file called `summary_assembly_metrics_mqc.csv` containing a selection of read alignment and *de novo* assembly related metrics will be saved in the `multiqc/` results directory. The same metrics will also be added to the top of the MultiQC report.

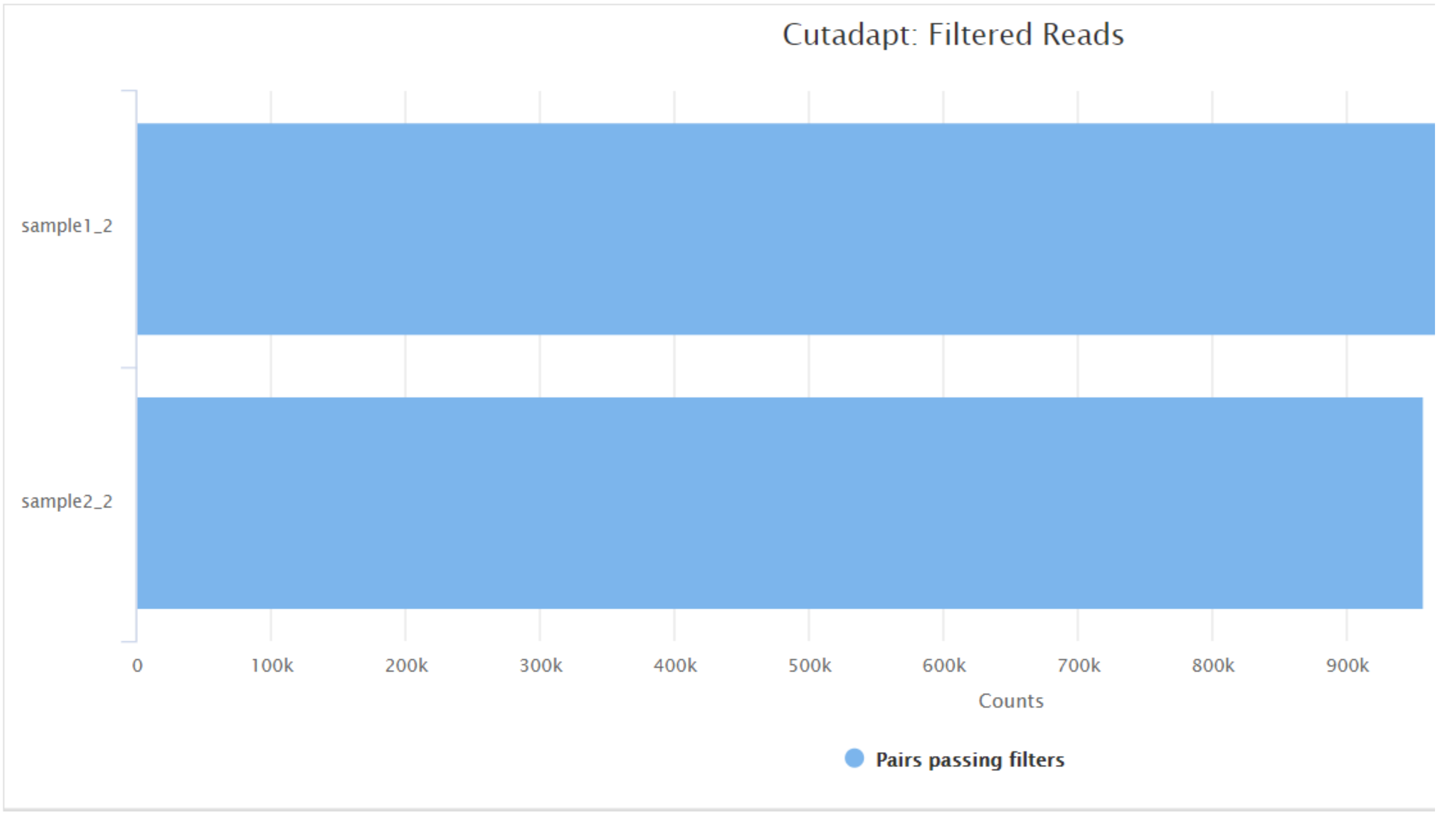
### Cutadapt

▼ Output files

- `assembly/cutadapt/log/`
- `*.cutadapt.log`: Cutadapt log file generated from stdout.
- `assembly/cutadapt/fastqc/`

- `*_fastqc.html`: FastQC report of the trimmed reads.
- `*_fastqc.zip`: Zip archive containing the FastQC report.

In the variant calling branch of the pipeline we are using iVar trim to remove primer sequences from the aligned BAM files for amplicon data. Since in the *de novo* assembly branch we don't align the reads, we use Cutadapt as an alternative option to remove and clean the primer sequences directly from FastQ files.



## SPAdes

▼ Output files

- `assembly/spades/<SPADES_MODE>/`
- `*.scaffolds.fa.gz`: SPAdes scaffold assembly.
- `*.contigs.fa.gz`: SPAdes assembly contigs.
- `*.assembly.gfa.gz`: SPAdes assembly graph in GFA format.
- `assembly/spades/<SPADES_MODE>/bandage/`
- `*.png`: Bandage visualisation for SPAdes assembly graph in PNG format.
- `*.svg`: Bandage visualisation for SPAdes assembly graph in SVG format.

**NB:** The value of `<SPADES_MODE>` in the output directory name above is determined by the `--spades_mode` parameter (Default: 'rnaviral').

SPAdes is an assembly toolkit containing various assembly pipelines. Generically speaking, SPAdes is one of the most popular de Bruijn graph-based assembly algorithms used for bacterial/viral genome reconstruction.

Bandage is a program for visualising *de novo* assembly graphs. By displaying connections which are not present in the contigs file, Bandage opens up new possibilities for analysing *de novo* assemblies.

## Unicycler

▼ Output files

- `assembly/unicycler/`
- `*.scaffolds.fa.gz`: Unicycler scaffold assembly.
- `*.assembly.gfa.gz`: Unicycler assembly graph in GFA format.
- `assembly/unicycler/bandage/`
- `*.png`: Bandage visualisation for Unicycler assembly graph in PNG format.
- `*.svg`: Bandage visualisation for Unicycler assembly graph in SVG format.

Unicycler is an assembly pipeline for bacterial genomes. It can assemble Illumina-only read sets where it functions as a SPAdes-optimiser.

## minia

▼ Output files

- `assembly/minia/`
- `*.contigs.fa`: Minia scaffold assembly.
- `*.unitigs.fa`: Minia unitigs fasta file.
- `*.h5`: Minia h5 output file.

Minia is a short-read assembler based on a de Bruijn graph, capable of assembling a human genome on a desktop computer in a day. The output of Minia is a set of contigs. Minia produces results of similar contiguity and accuracy to other de Bruijn assemblers.

### BLAST

▼ Output files

- `assembly/<ASSEMBLER>/blastn/`
- `*.blastn.txt`: BLAST results against the target virus.
- `*.filter.blastn.txt`: Filtered BLAST results.

**NB:** The value of `<ASSEMBLER>` in the output directory name above is determined by the `--assemblers` parameter (Default: 'spades').

blastn is used to align the assembled contigs against the virus reference genome.

### ABACAS

▼ Output files

- `assembly/<ASSEMBLER>/abacas/`
- `*.abacas.bin`: Bin file that contains contigs that are not used in ordering.
- `*.abacas.crunch`: Comparison file.
- `*.abacas.fasta`: Ordered and orientated sequence file.
- `*.abacas.gaps`: Gap information.
- `*.abacas.gaps.tab`: Gap information in tab-delimited format.
- `*.abacas.MULTIFASTA.fa`: A list of ordered and orientated contigs in a multi-fasta format.
- `*.abacas.tab`: Feature file
- `*.unused_contigs.out`: Information on contigs that have a mapping information but could not be used in the ordering.
- `assembly/<ASSEMBLER>/abacas/nucmer/`: Folder containing the files generated by the NUCmer algorithm used by ABACAS.

**NB:** The value of `<ASSEMBLER>` in the output directory name above is determined by the `--assemblers` parameter (Default: 'spades').

ABACAS was developed to rapidly contiguate (align, order, orientate), visualize and design primers to close gaps on shotgun assembled contigs based on a reference sequence.

### PlasmidID

▼ Output files

- `assembly/<ASSEMBLER>/plasmidid/<SAMPLE>/`
- `*_final_results.html`: Summary file with reference coverage stats and contigs for visualization.
- `*_final_results.tab`: Summary file with reference coverage stats and contigs.
- `images/<SAMPLE>_<REF_NAME>.png`: PNG file with the visualization of the alignment between the viral assembly and the reference viral genome.
- `logs/`: Log files.

**NB:** The value of `<ASSEMBLER>` in the output directory name above is determined by the `--assemblers` parameter (Default: 'spades').

PlasmidID was used to graphically represent the alignment of the reference genome relative to a given assembly. This helps to visualize the coverage of the reference genome in the assembly. To find more information about the output files refer to the documentation.

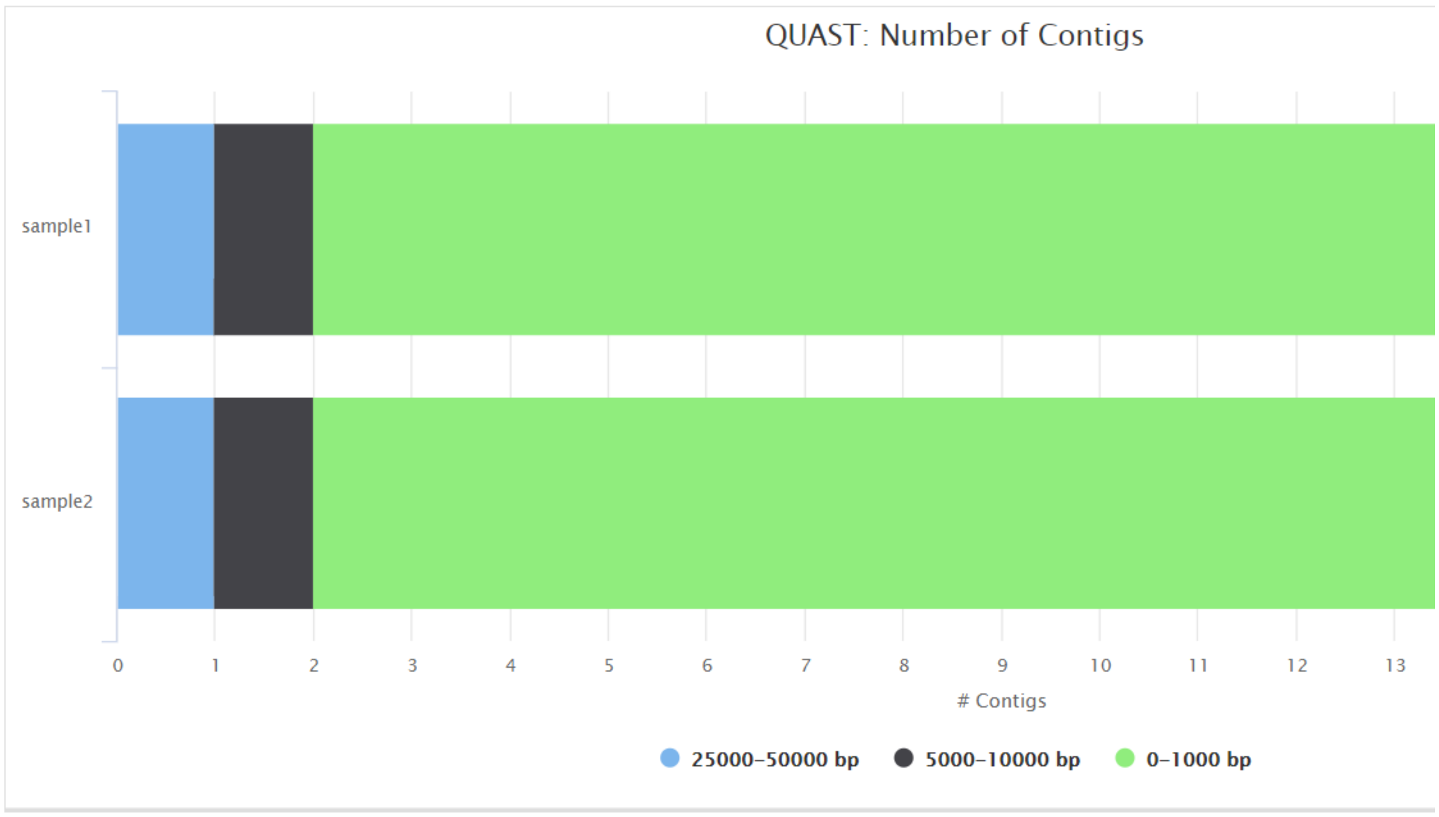
### Assembly QUAST

▼ Output files

- `assembly/<ASSEMBLER>/quast/`
- `report.html`: Results report in HTML format. Also available in various other file formats i.e `report.pdf`, `report.tex`, `report.tsv` and `report.txt`.

**NB:** The value of `<ASSEMBLER>` in the output directory name above is determined by the `--assemblers` parameter (Default: 'spades').

QUAST is used to generate a single report with which to evaluate the quality of the *de novo* assemblies across all of the samples provided to the pipeline. The HTML results can be opened within any browser (we recommend using Google Chrome). Please see the QUAST output docs for more detailed information regarding the output files.



## Illumina: Workflow reporting and genomes

### MultiQC

▼ Output files

- multiqc/
- multiqc_report.html: a standalone HTML file that can be viewed in your web browser.
- multiqc_data/: directory containing parsed statistics from the different tools used in the pipeline.
- summary_variants_metrics_mqc.csv: file containing a selection of read alignment and variant calling metrics. The same metrics will also be added to the top of the MultiQC report.
- summary_assembly_metrics_mqc.csv: file containing a selection of read alignment and*de novo* assembly related metrics. The same metrics will also be added to the top of the MultiQC report.

MultiQC is a visualization tool that generates a single HTML report summarizing all samples in your project. Most of the pipeline QC results are visualised in the report and further statistics are available in the report data directory.

Results generated by MultiQC collate pipeline QC from FastQC, fastp, Cutadapt, Bowtie 2, Kraken 2, samtools, picard CollectMultipleMetrics, BCFTools, SnpEff and QUAST.

The default multiqc config file has been written in a way in which to structure these QC metrics to make them more interpretable in the final report.

The pipeline has special steps which also allow the software versions to be reported in the MultiQC output for future traceability. For more information about how to use MultiQC reports, see http://multiqc.info.

An example MultiQC report generated from a full-sized dataset can be viewed on thenf-core website.

# Nanopore: Pipeline overview

## Nanopore: Preprocessing

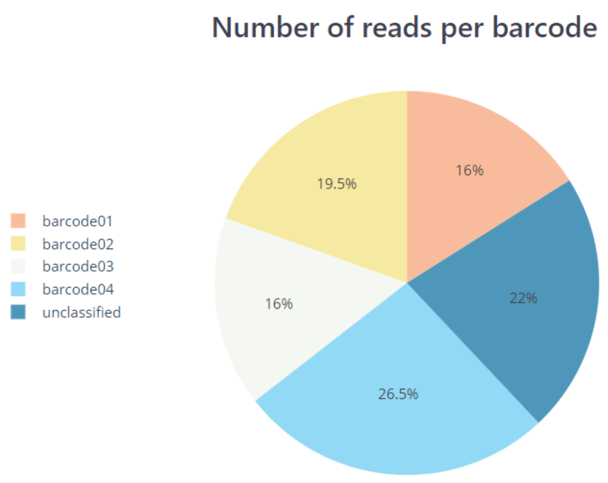
A file called `summary_variants_metrics_mqc.csv` containing a selection of read alignment and variant calling metrics will be saved in the`multiqc/<CALLER>/` output directory which is determined by the `--artic_minion_caller` parameter (Default: `nanopolish/`). The same metrics will also be added to the top of the MultiQC report.

### Nanopore: pycoQC

▼ Output files

- `pycoqc/`
- `*.html` and `.json` file that includes a run summary and graphical representation of various QC metrics including distribution of read length, distribution of read quality scores, mean read quality per sequence length, output per channel over experiment time and percentage of reads per barcode.

PycoQC compute metrics and generate QC plots using the sequencing summary information generated by basecalling/demultiplexing tools such as Guppy e.g. distribution of read length, read length over time, number of reads per barcode and other general stats.



### Nanopore: artic guppyplex

▼ Output files

- guppyplex/
- *.fastq.gz files generated by aggregate pre-demultiplexed reads from MinKNOW/Guppy. These files are not saved by default but can be via a custom config file such as the one below.

```
params {
    modules {
        'nanopore_artic_guppyplex' {
            publish_files = ['fastq.gz':'']
        }
    }
}
```

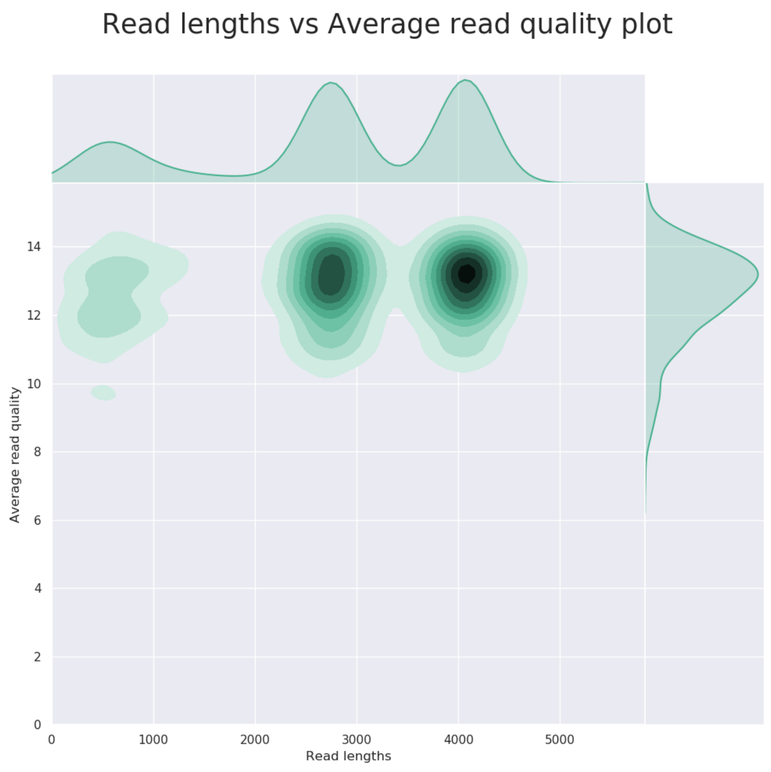
The artic guppyplex tool from the ARTIC field bioinformatics pipeline is used to perform length filtering of the demultiplexed Nanopore reads obtained per barcode. This essentially filters out chimeric reads that may be generated by the ARTIC protocol. The pipeline uses a default minimum and maximum read length of 400 and 700, respectively as tailored for the nCoV-2019 primer set. However, you may need to adjust these for different primer schemes e.g. by using the minimum length of the amplicons (--min-length) as well as the maximum length plus 200 (--max-length).

### Nanopore: NanoPlot

▼ Output files

- nanoplot/<SAMPLE>/
- Per-sample *.html files for QC metrics and individual *.png image files for plots.

NanoPlot it a tool that can be used to produce general quality metrics from various Nanopore-based input files including fastq files e.g. quality score distribution, read lengths and other general stats.



## Nanopore: Variant calling

### Nanopore: artic minion

▼ Output files

- <CALLER>/
- *.consensus.fasta: Consensus fasta file generated by artic minion.
- *.pass.unique.vcf.gz: VCF file containing unique variants passing quality filters.
- *.pass.unique.vcf.gz.tbi: VCF index file containing unique variants passing quality filters.
- *.pass.vcf.gz: VCF file containing variants passing quality filters.
- *.pass.vcf.gz.tbi: VCF index file containing variants passing quality filters.
- *.primers.vcf: VCF file containing variants found in primer-binding regions.
- *.merged.vcf: VCF file containing all detected variants.
- *.fail.vcf: VCF file containing variants failing quality filters.
- *.sorted.bam: BAM file generated by initial alignment.
- *.sorted.bam.bai: BAM index file generated by initial alignment.
- *.trimmed.rg.sorted.bam: BAM file without primer-binding site trimming.
- *.trimmed.rg.sorted.bam.bai: BAM index file without primer-binding site trimming.
- *.primertrimmed.rg.sorted.bam: BAM file generated after primer-binding site trimming.
- *.primertrimmed.rg.sorted.bam.bai: BAM index file generated after primer-binding site trimming.

**NB:** The value of <CALLER> in the output directory name above is determined by the --artic_minion_caller parameter (Default: 'nanopolish').

The artic minion tool from the ARTIC field bioinformatics pipeline is used to align reads, call variants and to generate the consensus sequence. By default, artic minion uses Minimap2 to align the reads to the viral genome, however you can use BWA instead using the --artic_minion_aligner bwa parameter. Similarly, the default variant caller used by artic minion is Nanopolish, however, you can use Medaka instead via the --artic_minion_caller medaka parameter. Medaka is faster than Nanopolish, performs mostly the same and can be run directly from fastq input files as opposed to requiring the fastq, fast5 and sequencing_summary.txt files required to run Nanopolish. You must provide the appropriate Medaka model via the --artic_minion_medaka_model parameter if using --artic_minion_caller medaka.

## Nanopore: Downstream analysis

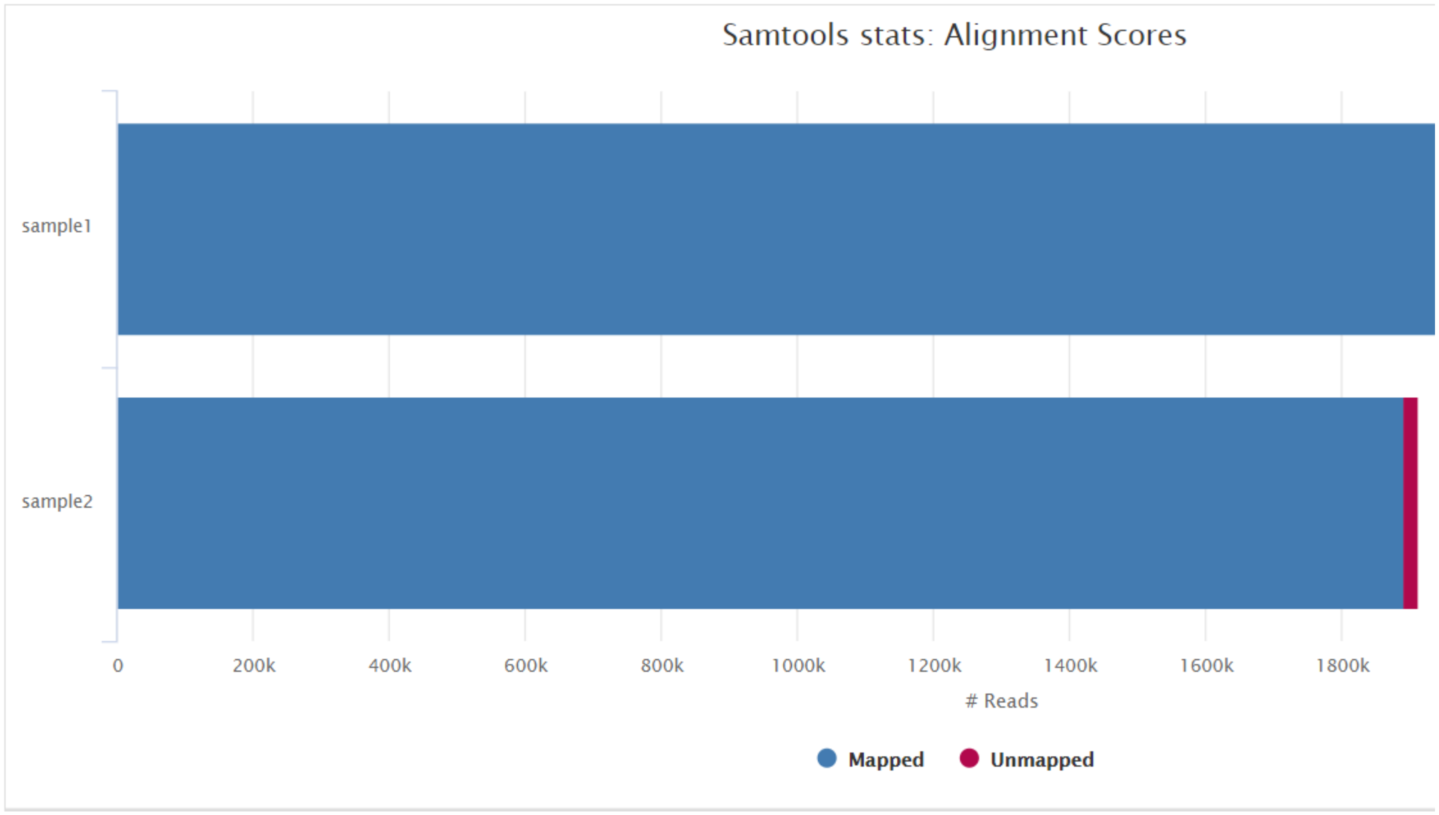
### Nanopore: SAMtools

▼ Output files

- <CALLER>/
- *.mapped.sorted.bam: Coordinate sorted BAM file containing read alignment information.
- *.mapped.sorted.bam.bai: Index file for coordinate sorted BAM file.
- <CALLER>/samtools_stats/
- SAMtools *.mapped.sorted.bam.flagstat, *.mapped.sorted.bam.idxstats and *.mapped.sorted.bam.stats files generated from the alignment files.

**NB:** The value of <CALLER> in the output directory name above is determined by the --artic_minion_caller parameter (Default: 'nanopolish').

BAM files containing the original alignments from either Minimap2 or BWA are further processed with SAMtools to remove unmapped reads as well as to generate read mapping statistics.



### Nanopore: mosdepth

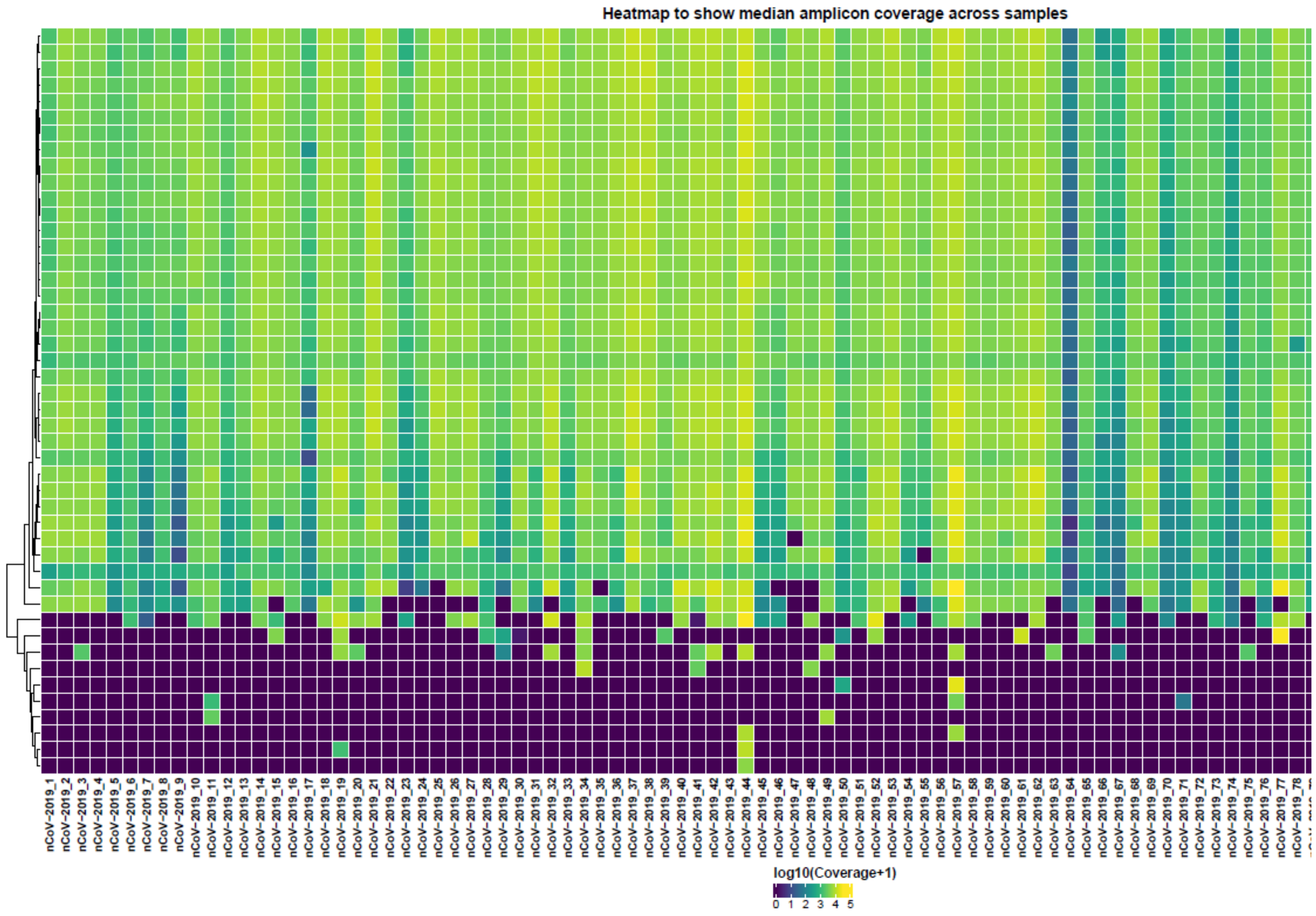
▼ Output files

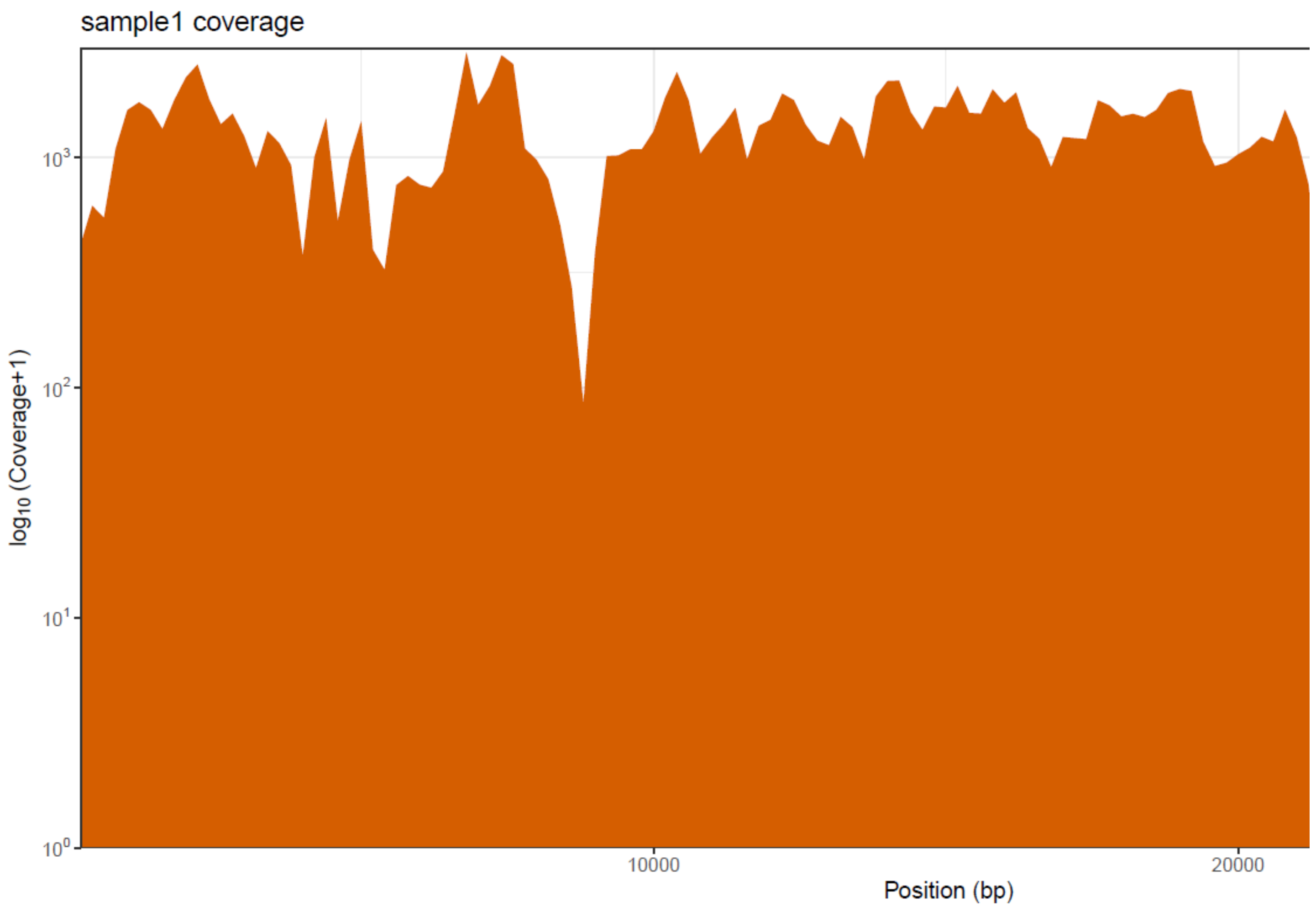
- <CALLER>/mosdepth/genome/
- all_samples.mosdepth.coverage.tsv: File aggregating genome-wide coverage values across all samples used for plotting.

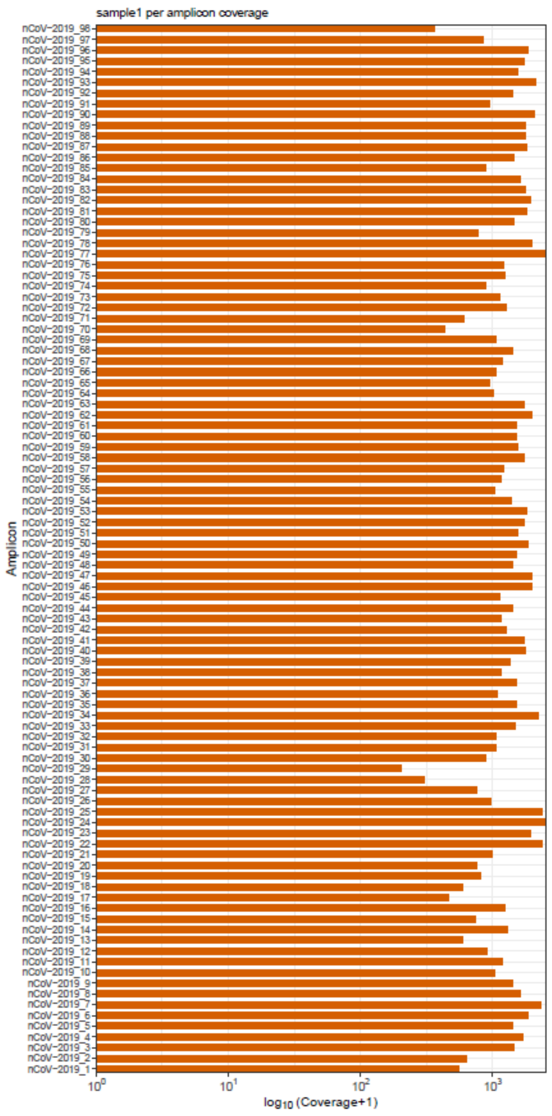
- *.mosdepth.coverage.pdf: Whole-genome coverage plot.
- *.mosdepth.coverage.tsv: File containing coverage values for the above plot.
- *.mosdepth.summary.txt: Summary metrics including mean, min and max coverage values.
- <CALLER>/mosdepth/amplicon/
- all_samples.mosdepth.coverage.tsv: File aggregating per-amplicon coverage values across all samples used for plotting.
- all_samples.mosdepth.heatmap.pdf: Heatmap showing per-amplicon coverage across all samples.
- *.mosdepth.coverage.pdf: Bar plot showing per-amplicon coverage for an individual sample.
- *.mosdepth.coverage.tsv: File containing per-amplicon coverage values for the above plot.
- *.mosdepth.summary.txt: Summary metrics including mean, min and max coverage values.

**NB:** The value of <CALLER> in the output directory name above is determined by the --artic_minion_caller parameter (Default: 'nanopolish').

mosdepth is a fast BAM/CRAM depth calculation for WGS, exome, or targeted sequencing. mosdepth is used in this pipeline to obtain genome-wide coverage values in 200bp windows and to obtain amplicon/region-specific coverage metrics. The results are then either rendered in MultiQC (genome-wide coverage) or are plotted using custom R scripts.




sample1 coverage
log10 (Coverage+1)
10^3
10^2
10^1
10^0
10000
20000
Position (bp)



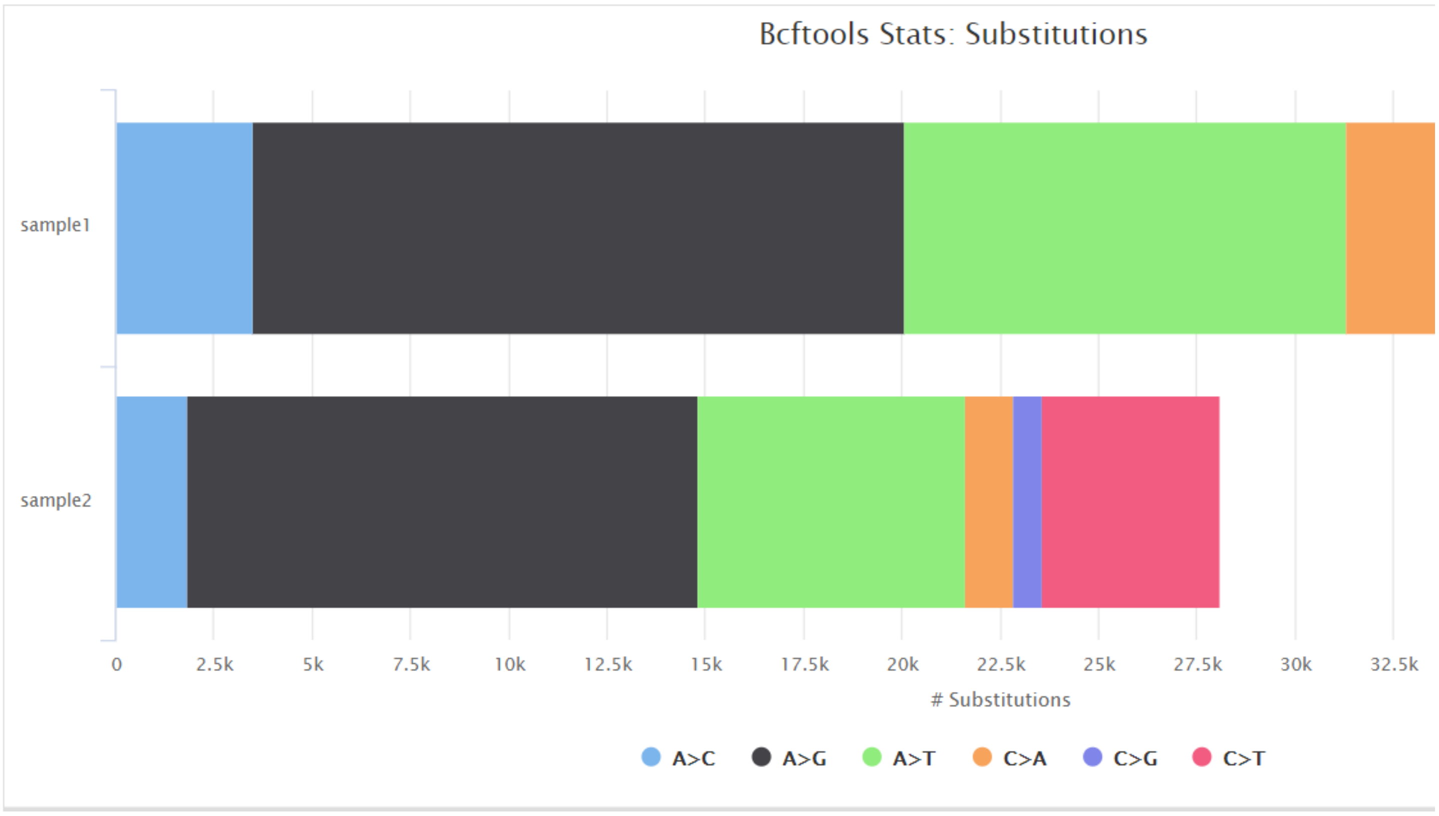
## Nanopore: BCFTools

▼ Output files

- `<CALLER>/bcftools_stats/`
- `*.bcftools_stats.txt`: Statistics and counts obtained from VCF file.

**NB:** The value of `<CALLER>` in the output directory name above is determined by the `--artic_minion_caller` parameter (Default: 'nanopolish').

BCFtools is a set of utilities that manipulate variant calls in VCF and its binary counterpart BCF format. It can also used be used to generate statistics and counts obtained from VCF files as used here.



## Nanopore: SnpEff and SnpSift

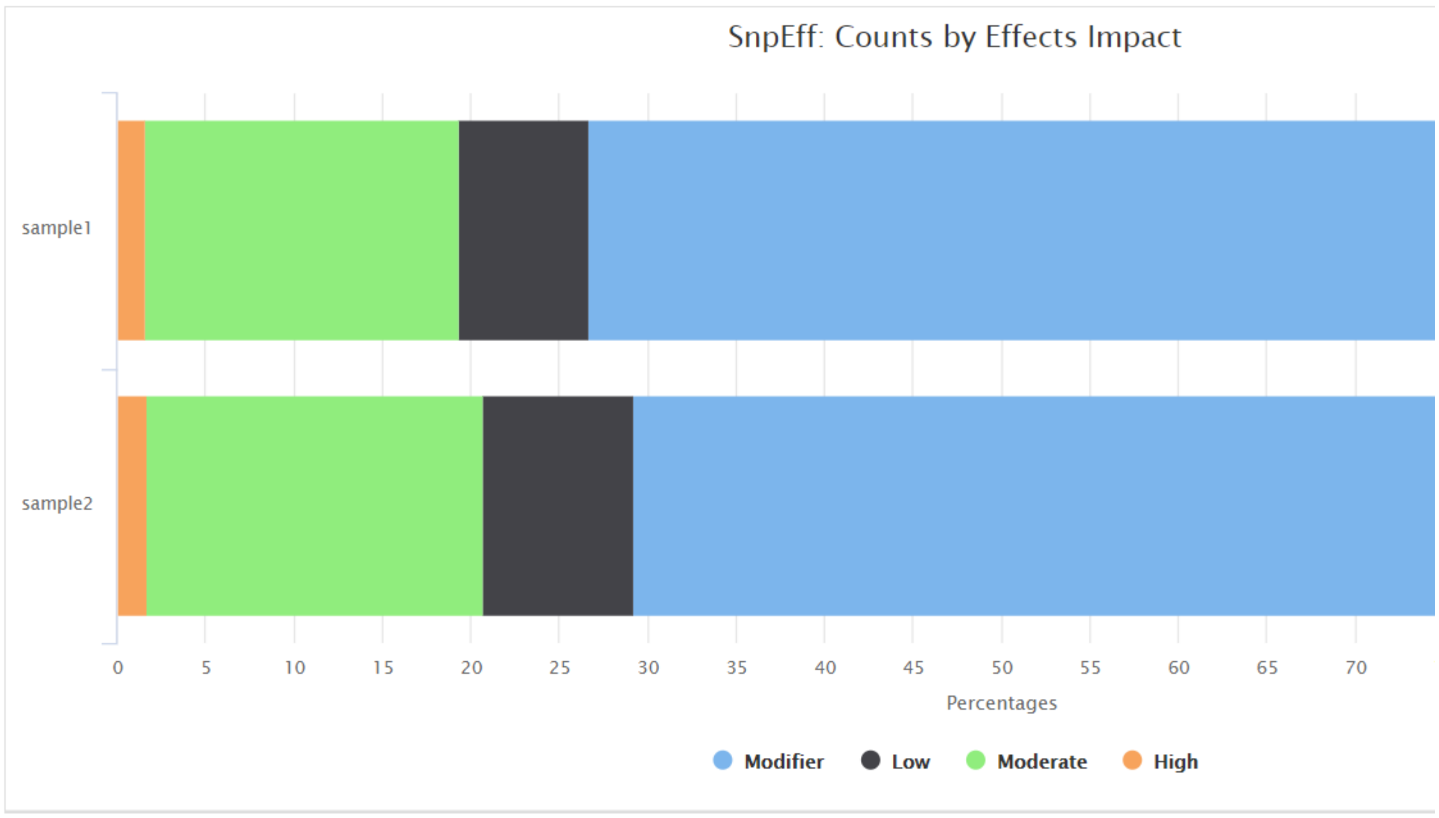
▼ Output files

- `<CALLER>/snpeff/`
- `*.snpeff.csv`: Variant annotation csv file.
- `*.snpeff.genes.txt`: Gene table for annotated variants.
- `*.snpeff.summary.html`: Summary html file for variants.
- `*.snpeff.vcf.gz`: VCF file with variant annotations.
- `*.snpeff.vcf.gz.tbi`: Index for VCF file with variant annotations.
- `*.snpsift.txt`: SnpSift summary table.
- `<CALLER>/snpeff/bcftools_stats/`
- `*.snpeff.bcftools_stats.txt`: Statistics and counts obtained from SnpEff VCF file.

**NB:** The value of `<CALLER>` in the output directory name above is determined by the `--artic_minion_caller` parameter (Default: 'nanopolish').

SnpEff is a genetic variant annotation and functional effect prediction toolbox. It annotates and predicts the effects of genetic variants on genes and proteins (such as amino acid changes).

SnpSift annotates genomic variants using databases, filters, and manipulates genomic annotated variants. After annotation with SnpEff, you can use SnpSift to help filter large genomic datasets in order to find the most significant variants.



## Nanopore: QUAST

▼ Output files

- `<CALLER>/quast/`
- `report.html`: Results report in HTML format. Also available in various other file formats i.e. `report.pdf`, `report.tex`, `report.tsv` and `report.txt`.

**NB:** The value of `<CALLER>` in the output directory name above is determined by the `--artic_minion_caller` parameter (Default: 'nanopolish').

QUAST is used to generate a single report with which to evaluate the quality of the consensus sequence across all of the samples provided to the pipeline. The HTML results can be opened within any browser (we recommend using Google Chrome). Please see the QUAST output docs for more detailed information regarding the output files.

## Nanopore: Pangolin

▼ Output files

- `<CALLER>/pangolin/`
- `*.pangolin.csv`: Lineage analysis results from Pangolin.

**NB:** The value of `<CALLER>` in the output directory name above is determined by the `--artic_minion_caller` parameter (Default: 'nanopolish').

Phylogenetic Assignment of Named Global Outbreak LINeages (Pangolin) has been used extensively during the COVID-19 pandemic to assign lineages to SARS-CoV-2 genome sequenced samples. A web application also exists that allows users to upload genome sequences via a web browser to assign lineages to genome sequences of SARS-CoV-2, view descriptive characteristics of the assigned lineage(s), view the placement of the lineage in a phylogeny of global samples, and view the temporal and geographic distribution of the assigned lineage(s).

## Nanopore: Nextclade

▼ Output files

- `<CALLER>/nextclade/`
- `*.csv`: Analysis results from Nextlade containing genome clade assignment, mutation calling and sequence quality checks.

**NB:** The value of `<CALLER>` in the output directory name above is determined by the `--artic_minion_caller` parameter (Default: 'nanopolish').

Nextclade performs viral genome clade assignment, mutation calling and sequence quality checks for the consensus sequences generated in this pipeline. Similar to Pangolin, it has been used extensively during the COVID-19 pandemic. A web application also exists that allows users to upload genome sequences via a web browser.

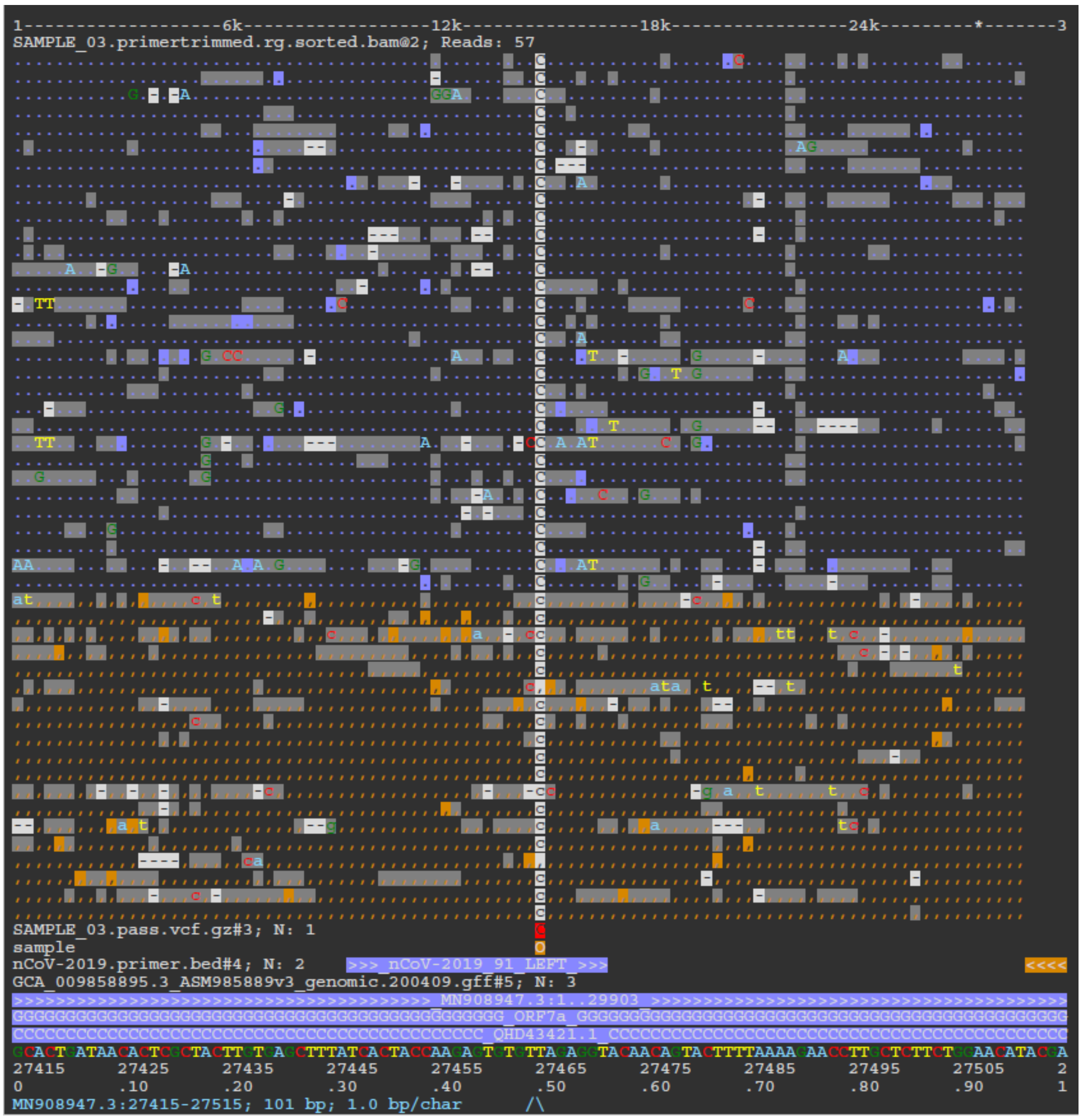
## Nanopore: ASCIIGenome

▼ Output files

- `<CALLER>/asciigenome/<SAMPLE>/`
- `*.pdf`: Individual variant screenshots with annotation tracks in PDF format.

**NB:** The value of `<CALLER>` in the output directory name above is determined by the `--artic_minion_caller` parameter (Default: 'nanopolish').

As described in the documentation, ASCIIGenome is a command-line genome browser that can be run from a terminal window and is solely based on ASCII characters. The closest program to ASCIIGenome is probably samtools tview but ASCIIGenome offers much more flexibility, similar to popular GUI viewers like the IGV browser. We are using the batch processing mode of ASCIIGenome in this pipeline to generate individual screenshots for all of the variant sites reported for each sample in the VCF files. This is incredibly useful to be able to quickly QC the variants called by the pipeline without having to tediously load all of the relevant tracks into a conventional genome browser. Where possible, the BAM read alignments, VCF variant file, primer BED file and GFF annotation track will be represented in the screenshot for contextual purposes. The screenshot below shows a SNP called relative to the MN908947.3 SARS-CoV-2 reference genome that overlaps the ORF7a protein and the nCoV-2019_91_LEFT primer from the ARIC v3 protocol.



```
1-------------------6k-----------------12k----------------18k----------------24k---------*-------3
SAMPLE_03.primertrimmed.rg.sorted.bam@2; Reads: 57
SAMPLE_03.pass.vcf.gz#3; N: 1
sample
nCoV-2019.primer.bed#4; N: 2      >>>_nCoV-2019_91_LEFT_>>>                                   <<<<
GCA_009858895.3_ASM985889v3_genomic.200409.gff#5; N: 3
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>_MN908947.3:1..29903_>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
GGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGG_ORF7a_GGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGG
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC_QHD43421.1_CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
GCACTGATAACACTCGCTACTTGTGAGCTTTATCACTACCAAGAGTGTGTTAGAGGTACAACAGTACTTTTAAAAGAACCTTGCTCTTCTGGAACATACGA
27415     27425     27435     27445     27455     27465     27475     27485     27495     27505     2
0         .10       .20       .30       .40       .50       .60       .70       .80       .90       1
MN908947.3:27415-27515; 101 bp; 1.0 bp/char     /\
```

## Nanopore: Variants long table

▼ Output files

- `<CALLER>/`
- `variants_long_table.csv`: Long format table collating per-sample information for individual variants, functional effect prediction and lineage analysis.

**NB:** The value of `<CALLER>` in the output directory name above is determined by the `--artic_minion_caller` parameter (Default: 'nanopolish').

Create variants long format table collating per-sample information for individual variants (BCFTools), functional effect prediction (SnpSift) and lineage analysis (Pangolin).

The more pertinent variant information is summarised in this table to make it easier for researchers to assess the impact of variants found amongst the sequenced sample(s). An example of the fields included in the table are shown below:

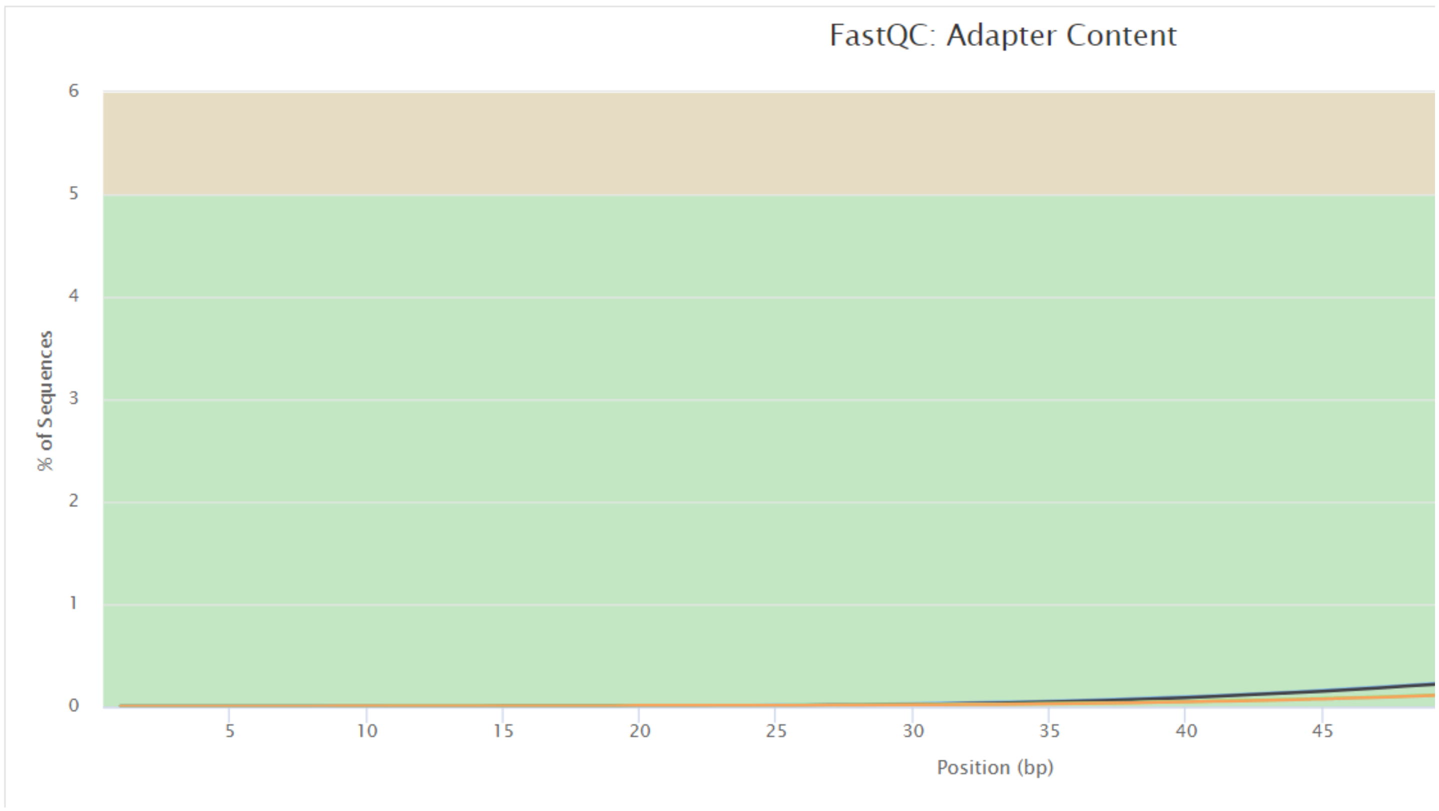
```
SAMPLE,CHROM,POS,REF,ALT,FILTER,DP,REF_DP,ALT_DP,AF,GENE,EFFECT,HGVS_C,HGVS_P,HGVS_P_1LETTER,CALLER,LINEAGE
SAMPLE1_PE,MN908947.3,241,C,T,PASS,489,4,483,0.99,orf1ab,upstream_gene_variant,c.-25C>T,.,.,ivar,B.1
SAMPLE1_PE,MN908947.3,1875,C,T,PASS,92,62,29,0.32,orf1ab,missense_variant,c.1610C>T,p.Ala537Val,p.A537V,ivar,B.1
SAMPLE1_PE,MN908947.3,3037,C,T,PASS,213,0,213,1.0,orf1ab,synonymous_variant,c.2772C>T,p.Phe924Phe,p.F924F,ivar,B.1
SAMPLE1_PE,MN908947.3,11719,G,A,PASS,195,9,186,0.95,orf1ab,synonymous_variant,c.11454G>A,p.Gln3818Gln,p.Q3818Q,ivar,B.1
```

## Nanopore: Workflow reporting

### Nanopore: MultiQC

▼ Output files

- `multiqc/<CALLER>/`
- `multiqc_report.html`: a standalone HTML file that can be viewed in your web browser.
- `multiqc_data/`: directory containing parsed statistics from the different tools used in the pipeline.
- `summary_variants_metrics_mqc.csv`: file containing a selection of read alignmnet and variant calling metrics. The same metrics will also be added to the top of the MultiQC report.



Results generated by MultiQC collate pipeline QC from pycoQC, samtools, mosdepth, BCFTools, SnpEff and QUAST.

The default multiqc config file has been written in a way in which to structure these QC metrics to make them more interpretable in the final report.

The pipeline has special steps which also allow the software versions to be reported in the MultiQC output for future traceability. For more information about how to use MultiQC reports, see http://multiqc.info.

An example MultiQC report generated from a full-sized dataset can be viewed on thenf-core website.

### Reference genome files

▼ Output files

- genome/
- bowtie2/: Bowtie 2 index for viral genome.
- blast_db/: BLAST database for viral genome.
- kraken2_db/: Kraken 2 database for host genome.
- snpeff_db/: SnpEff database for viral genome.
- snpeff.config: SnpEff config file for viral genome.
- Unzipped genome fasta file for viral genome
- Unzipped genome annotation GFF file for viral genome

A number of genome-specific files are generated by the pipeline because they are required for the downstream processing of the results. If thesave_reference parameter is provided then the Bowtie 2 alignment indices, BLAST and Kraken 2 databases downloaded/generated by the pipeline will be saved in the genome/ directory. It is recommended to use the --save_reference parameter if you are using the pipeline to build a Kraken 2 database for the host genome. This can be quite a time-consuming process and it permits their reuse for future runs of the pipeline or for other purposes.

## Pipeline information

▼ Output files

- pipeline_info/
- Reports generated by Nextflow: execution_report.html, execution_timeline.html, execution_trace.txt and pipeline_dag.dot/pipeline_dag.svg.
- Reports generated by the pipeline: pipeline_report.html, pipeline_report.txt and software_versions.yml. The pipeline_report* files will only be present if the--email / --email_on_fail parameter's are used when running the pipeline.
- Reformatted samplesheet files used as input to the pipeline:samplesheet.valid.csv.

Nextflow provides excellent functionality for generating various reports relevant to the running and execution of the pipeline. This will allow you to troubleshoot errors with the running of the pipeline, and also provide you with other information such as launch commands, run times and resource usage.

# nf-core/mag: Output

### Introduction

This document describes the output produced by the pipeline. Most of the plots are taken from the MultiQC report, which summarises results at the end of the pipeline.

The directories listed below will be created in the results directory after the pipeline has finished. All paths are relative to the top-level results directory.

## Pipeline overview

The pipeline is built using Nextflow and processes data using the following steps:

- Quality control of input reads - trimming and contaminant removal
- Taxonomic classification of trimmed reads
- Digital sequencing normalisation
- Assembly of trimmed reads
- Protein-coding gene prediction of assemblies
- Virus identification of assemblies
- Binning and binning refinement of assembled contigs
- Taxonomic classification of binned genomes
- Genome annotation of binned genomes
- Additional summary for binned genomes
- Ancient DNA
- MultiQC - aggregate report, describing results of the whole pipeline
- Pipeline information - Report metrics generated during the workflow execution

Note that when specifying the parameter `--coassemble_group`, for the corresponding output filenames/directories of the assembly or downsteam processes the group ID, or more precisely the term `group-[group_id]`, will be used instead of the sample ID.

## Quality control

These steps trim away the adapter sequences present in input reads, trims away bad quality bases and sicard reads that are too short. It also removes host contaminants and sequencing controls, such as PhiX or the Lambda phage. FastQC is run for visualising the general quality metrics of the sequencing runs before and after trimming.

### FastQC

▼ Output files

- `QC_shortreads/fastqc/`
- `[sample]_[1/2]_fastqc.html`: FastQC report, containing quality metrics for your untrimmed raw fastq files
- `[sample].trimmed_[1/2]_fastqc.html`: FastQC report, containing quality metrics for trimmed and, if specified, filtered read files

FastQC gives general quality metrics about your sequenced reads. It provides information about the quality score distribution across your reads, per base sequence content (%A/T/G/C), adapter contamination and overrepresented sequences. For further reading and documentation see the FastQC help pages.

### fastp

fastp is a all-in-one fastq preprocessor for read/adapter trimming and quality control. It is used in this pipeline for trimming adapter sequences and discard low-quality reads. Its output is in the results folder and part of the MultiQC report.

▼ Output files

- `QC_shortreads/fastp/[sample]/`
- `fastp.html`: Interactive report
- `fastp.json`: Report in json format

### AdapterRemoval2

AdapterRemoval searches for and removes remnant adapter sequences from High-Throughput Sequencing (HTS) data and (optionally) trims low quality bases from the 3' end of reads following adapter removal. It is popular in the field of palaeogenomics. The output logs are stored in the results folder, and as a part of the MultiQC report.

▼ Output files

- `QC_shortreads/adapterremoval/[sample]/`
- `[sample]_ar2.settings`: AdapterRemoval log file.

### Remove PhiX sequences from short reads

The pipeline uses bowtie2 to map the reads against PhiX and removes mapped reads.

▼ Output files

- `QC_shortreads/remove_phix/`
- `[sample].phix_removed.bowtie2.log`: Contains a brief log file indicating how many reads have been retained.

### Host read removal

The pipeline uses bowtie2 to map short reads against the host reference genome specified with `--host_genome` or `--host_fasta` and removes mapped reads. The information about discarded and retained reads is also included in the MultiQC report.

▼ Output files

- `QC_shortreads/remove_host/`
- `[sample].host_removed.bowtie2.log`: Contains the bowtie2 log file indicating how many reads have been mapped.
- `[sample].host_removed.mapped*.read_ids.txt`: Contains a file listing the read ids of discarded reads.

### Remove Phage Lambda sequences from long reads

The pipeline uses Nanolyse to map the reads against the Lambda phage and removes mapped reads.

▼ Output files

- `QC_longreads/NanoLyse/`
- `[sample]_nanolyse.log`: Contains a brief log file indicating how many reads have been retained.

### Filtlong and porechop

The pipeline uses filtlong and porechop to perform quality control of the long reads that are eventually provided with the TSV input file.

No direct host read removal is performed for long reads. However, since within this pipeline filtlong uses a read quality based on k-mer matches to the already filtered short reads, reads not overlapping those short reads might be discarded. The lower the parameter `--longreads_length_weight`, the higher the impact of the read qualities for filtering. For further documentation see the filtlong online documentation.

### Quality visualisation for long reads

NanoPlot is used to calculate various metrics and plots about the quality and length distribution of long reads. For more information about NanoPlot see the online documentation.

▼ Output files

- `QC_longreads/NanoPlot/[sample]/`
- `raw_*.[png/html/txt]`: Plots and reports for raw data
- `filtered_*.[png/html/txt]`: Plots and reports for filtered data

## Digital normalization with BBnorm

If the pipeline is called with the `--bbnorm` option, it will normalize sequencing depth of libraries prior assembly by removing reads to 1) reduce coverage of very abundant kmers and 2) delete very rare kmers (see `--bbnorm_target` and `--bbnorm_min` parameters). When called in conjunction with `--coassemble_group`, BBnorm will operate on interleaved (merged) FastQ files, producing only a single output file. If the `--save_bbnorm_reads` parameter is set, the resulting FastQ files are saved together with log output.

▼ Output files

- `bbmap/bbnorm/[sample]\*.fastq.gz`
- `bbmap/bbnorm/log/[sample].bbnorm.log`

## Taxonomic classification of trimmed reads

### Kraken

Kraken2 classifies reads using a k-mer based approach as well as assigns taxonomy using a Lowest Common Ancestor (LCA) algorithm.

▼ Output files

- `Taxonomy/kraken2/[sample]/`
- `kraken2.report`: Classification in the Kraken report format. See the kraken2 manual for more details
- `taxonomy.krona.html`: Interactive pie chart produced by KronaTools

### Centrifuge

Centrifuge is commonly used for the classification of DNA sequences from microbial samples. It uses an indexing scheme based on the Burrows-Wheeler transform (BWT) and the Ferragina-Manzini (FM) index.

More information on the Centrifuge website

▼ Output files

- `Taxonomy/centrifuge/[sample]/`
- `report.txt`: Tab-delimited result file. See the centrifuge manual for information about the fields
- `kreport.txt`: Classification in the Kraken report format. See the kraken2 manual for more details
- `taxonomy.krona.html`: Interactive pie chart produced by KronaTools

## Assembly

Trimmed (short) reads are assembled with both megahit and SPAdes. Hybrid assembly is only supported by SPAdes.

### MEGAHIT

MEGAHIT is a single node assembler for large and complex metagenomics short reads.

▼ Output files

- `Assembly/MEGAHIT/`
- `[sample/group].contigs.fa.gz`: Compressed metagenome assembly in fasta format
- `[sample/group].log`: Log file
- `QC/[sample/group]/`: Directory containing QUAST files and Bowtie2 mapping logs
  - `MEGAHIT-[sample].bowtie2.log`: Bowtie2 log file indicating how many reads have been mapped from the sample that the metagenome was assembled from, only present if `--coassemble_group` is not set.
  - `MEGAHIT-[sample/group]-[sampleToMap].bowtie2.log`: Bowtie2 log file indicating how many reads have been mapped from the respective sample ("sampleToMap").
  - `MEGAHIT-[sample].[bam/bai]`: Optionally saved BAM file of the Bowtie2 mapping of reads against the assembly.

### SPAdes

SPAdes was originally a single genome assembler that later added support for assembling metagenomes.

▼ Output files

- Assembly/SPAdes/
- [sample/group]_scaffolds.fasta.gz: Compressed assembled scaffolds in fasta format
- [sample/group]_graph.gfa.gz: Compressed assembly graph in gfa format
- [sample/group]_contigs.fasta.gz: Compressed assembled contigs in fasta format
- [sample/group].log: Log file
- QC/[sample/group]/: Directory containing QUAST files and Bowtie2 mapping logs
  - SPAdes-[sample].bowtie2.log: Bowtie2 log file indicating how many reads have been mapped from the sample that the metagenome was assembled from, only present if --coassemble_group is not set.
  - SPAdes-[sample/group]-[sampleToMap].bowtie2.log: Bowtie2 log file indicating how many reads have been mapped from the respective sample ("sampleToMap").
  - SPAdes-[sample].[bam/bai]: Optionally saved BAM file of the Bowtie2 mapping of reads against the assembly.

### SPAdesHybrid

SPAdesHybrid is a part of the SPAdes software and is used when the user provides both long and short reads.

▼ Output files

- Assembly/SPAdesHybrid/
- [sample/group]_scaffolds.fasta.gz: Compressed assembled scaffolds in fasta format
- [sample/group]_graph.gfa.gz: Compressed assembly graph in gfa format
- [sample/group]_contigs.fasta.gz: Compressed assembled contigs in fasta format
- [sample/group].log: Log file
- QC/[sample/group]/: Directory containing QUAST files and Bowtie2 mapping logs
  - SPAdesHybrid-[sample].bowtie2.log: Bowtie2 log file indicating how many reads have been mapped from the sample that the metagenome was assembled from, only present if --coassemble_group is not set.
  - SPAdesHybrid-[sample/group]-[sampleToMap].bowtie2.log: Bowtie2 log file indicating how many reads have been mapped from the respective sample ("sampleToMap").
  - SPAdesHybrid-[sample].[bam/bai]: Optionally saved BAM file of the Bowtie2 mapping of reads against the assembly.

### Metagenome QC with QUAST

QUAST is a tool that evaluates metagenome assemblies by computing various metrics. The QUAST output is also included in the MultiQC report, as well as in the assembly directories themselves.

▼ Output files

- Assembly/[assembler]/QC/[sample/group]/QUAST/
- report.*: QUAST report in various formats, such as html, pdf, tex, tsv, or txt
- transposed_report.*: QUAST report that has been transposed into wide format (tex, tsv, or txt)
- quast.log: QUAST log file
- metaquast.log: MetaQUAST log file
- icarus.html: Icarus main menu with links to interactive viewers
- icarus_viewers/contig_size_viewer.html: Diagram of contigs that are ordered from longest to shortest
- basic_stats/cumulative_plot.pdf: Shows the growth of contig lengths (contigs are ordered from largest to shortest)
- basic_stats/GC_content_plot.pdf: Shows the distribution of GC content in the contigs
- basic_stats/[assembler]-[sample/group]_GC_content_plot.pdf: Histogram of the GC percentage for the contigs
- basic_stats/Nx_plot.pdf: Plot of Nx values as x varies from 0 to 100%.
- predicted_genes/[assembler]-[sample/group].rna.gff: Contig positions for rRNA genes in gff version 3 format
- predicted_genes/barrnap.log: Barrnap log file (ribosomal RNA predictor)

## Gene prediction

Protein-coding genes are predicted for each assembly.

▼ Output files

- Annotation/Prodigal/
- [assembler]-[sample/group].gff.gz: Gene Coordinates in GFF format
- [assembler]-[sample/group].faa.gz: The protein translation file consists of all the proteins from all the sequences in multiple FASTA format.
- [assembler]-[sample/group].fna.gz: Nucleotide sequences of the predicted proteins using the DNA alphabet, not mRNA (so you will see 'T' in the output and not 'U').
- [assembler]-[sample/group]_all.txt.gz: Information about start positions of genes.

## Virus identification in assemblies

### geNomad

geNomad identifies viruses and plasmids in sequencing data (isolates, metagenomes, and metatranscriptomes)

▼ Output files

- VirusIdentification/geNomad/[assembler]-[sample/group]*/
- [assembler]-[sample/group]*_annotate
  - [assembler]-[sample/group]*_taxonomy.tsv: Taxonomic assignment data
- [assembler]-[sample/group]*_aggregated_classification
  - [assembler]-[sample/group]*_aggregated_classification.tsv: Sequence classification in tabular format

- [assembler]-[sample/group]*_find_proviruses
    - [assembler]-[sample/group]*_provirus.tsv: Characteristics of proviruses identified by geNomad
- [assembler]-[sample/group]*_summary
    - [assembler]-[sample/group]*_virus_summary.tsv: Virus classification summary file in tabular format
    - [assembler]-[sample/group]*_plasmid_summary.tsv: Plasmid classification summary file in tabular format
    - [assembler]-[sample/group]*_viruses_genes.tsv: Virus gene annotation data in tabular format
    - [assembler]-[sample/group]*_plasmids_genes.tsv: Plasmid gene annotation data in tabular format
    - [assembler]-[sample/group]*_viruses.fna: Virus nucleotide sequences in FASTA format
    - [assembler]-[sample/group]*_plasmids.fna: Plasmid nucleotide sequences in FASTA format
    - [assembler]-[sample/group]*_viruses_proteins.faa: Virus protein sequences in FASTA format
    - [assembler]-[sample/group]*_plasmids_proteins.faa: Plasmid protein sequences in FASTA format
- [assembler]-[sample/group]*.log: Plain text log file detailing the steps executed by geNomad (annotate, find-proviruses, marker-classification, nn-classification, aggregated-classification and summary)

## Binning and binning refinement

### Contig sequencing depth

Sequencing depth per contig and sample is generated by MetaBAT2's `jgi_summarize_bam_contig_depths --outputDepth`. The values correspond to `(sum of exactly aligned bases) / ((contig length)-2*75)`. For example, for two reads aligned exactly with `10` and `9` bases on a 1000 bp long contig the depth is calculated by `(10+9)/(1000-2*75)` (1000bp length of contig minus 75bp from each end, which is excluded).

These depth files are used for downstream binning steps.

▼ Output files

- GenomeBinning/depths/contigs/
- [assembler]-[sample/group]-depth.txt.gz: Sequencing depth for each contig and sample or group, only for short reads.

### MetaBAT2

MetaBAT2 recovers genome bins (that is, contigs/scaffolds that all belongs to a same organism) from metagenome assemblies.

▼ Output files

- GenomeBinning/MetaBAT2/
- bins/[assembler]-[binner]-[sample/group].*.fa.gz: Genome bins retrieved from input assembly
- unbinned/[assembler]-[binner]-[sample/group].unbinned.[1-9]*.fa.gz: Contigs that were not binned with other contigs but considered interesting. By default, these are at least 1 Mbp (`--min_length_unbinned_contigs`) in length and at most the 100 longest contigs (`-max_unbinned_contigs`) are reported

All the files and contigs in these folders will be assessed by QUAST and BUSCO.

All other files that were discarded by the tool, or from the low-quality unbinned contigs, can be found here.

▼ Output files

- GenomeBinning/MetaBAT2/discarded/
- *.lowDepth.fa.gz: Low depth contigs that are filtered by MetaBAT2
- *.tooShort.fa.gz: Too short contigs that are filtered by MetaBAT2
- GenomeBinning/MetaBAT2/unbinned/discarded/
- *.unbinned.pooled.fa.gz: Pooled unbinned contigs equal or above `--min_contig_size`, by default 1500 bp.
- *.unbinned.remaining.fa.gz: Remaining unbinned contigs below `--min_contig_size`, by default 1500 bp, but not in any other file.

All the files in this folder contain small and/or unbinned contigs that are not further processed.

Files in these two folders contain all contigs of an assembly.

### MaxBin2

MaxBin2 recovers genome bins (that is, contigs/scaffolds that all belongs to a same organism) from metagenome assemblies.

▼ Output files

- GenomeBinning/MaxBin2/
- bins/[assembler]-[binner]-[sample/group].*.fa.gz: Genome bins retrieved from input assembly
- unbinned/[assembler]-[binner]-[sample/group].noclass.[1-9]*.fa.gz: Contigs that were not binned with other contigs but considered interesting. By default, these are at least 1 Mbp (`--min_length_unbinned_contigs`) in length and at most the 100 longest contigs (`-max_unbinned_contigs`) are reported.

All the files and contigs in these folders will be assessed by QUAST and BUSCO.

▼ Output files

- GenomeBinning/MaxBin2/discarded/
- *.tooshort.gz: Too short contigs that are filtered by MaxBin2
- GenomeBinning/MaxBin2/unbinned/discarded/
- *.noclass.pooled.fa.gz: Pooled unbinned contigs equal or above `--min_contig_size`, by default 1500 bp.
- *.noclass.remaining.fa.gz: Remaining unbinned contigs below `--min_contig_size`, by default 1500 bp, but not in any other file.

All the files in this folder contain small and/or unbinned contigs that are not further processed.

Files in these two folders contain all contigs of an assembly.

### CONCOCT

CONCOCT performs unsupervised binning of metagenomic contigs by using nucleotide composition, coverage data in multiple samples and linkage data from paired end reads.

▼ Output files

- `GenomeBinning/CONCOCT/`
- `bins/[assembler]-[binner]-[sample/group].*.fa.gz`: Genome bins retrieved from input assembly
- `stats/[assembler]-[binner]-[sample/group].csv`: Table indicating which contig goes with which cluster bin.
- `stats/[assembler]-[binner]-[sample/group]*_gt1000.csv`: Various intermediate PCA statistics used for clustering.
- `stats/[assembler]-[binner]-[sample/group]_*.tsv`: Coverage statistics of each sub-contig cut up by CONOCOCT prior in an intermediate step prior to binning. Likely not useful in most cases.
- `stats/[assembler]-[binner]-[sample/group].log.txt`: CONCOCT execution log file.
- `stats/[assembler]-[binner]-[sample/group]_*.args`: List of arguments used in CONCOCT execution.
-

All the files and contigs in these folders will be assessed by QUAST and BUSCO, if the parameter `-postbinning_input` is not set to `refined_bins_only`.

Note that CONCOCT does not output what it considers 'unbinned' contigs, therefore no 'discarded' contigs are produced here. You may still need to do your own manual curation of the resulting bins.

### DAS Tool

DAS Tool is an automated binning refinement method that integrates the results of a flexible number of binning algorithms to calculate an optimized, non-redundant set of bins from a single assembly. nf-core/mag uses this tool to attempt to further improve bins based on combining the MetaBAT2 and MaxBin2 binning output, assuming sufficient quality is met for those bins.

DAS Tool will remove contigs from bins that do not pass additional filtering criteria, and will discard redundant lower-quality output from binners that represent the same estimated 'organism', until the single highest quality bin is represented.

⚠ If DAS Tool does not find any bins passing your selected threshold it will exit with an error. Such an error is 'ignored' by nf-core/mag, therefore you will not find files in the `GenomeBinning/DASTool/` results directory for that particular sample.

▼ Output files

- `GenomeBinning/DASTool/`
- `[assembler]-[sample/group]_allBins.eval`: Tab-delimited description with quality and completeness metrics for the input bin sets. Quality and completeness are estimated by DAS TOOL using a scoring function based on the frequency of bacterial or archaeal reference single-copy genes (SCG). Please see note at the bottom of this section on file names.
- `[assembler]-[sample/group]_DASTool_summary.tsv`: Tab-delimited description with quality and completeness metrics for the refined output bin sets.
- `[assembler]-[sample/group]_DASTool_contig2bin.tsv`: File describing which contig is associated to which bin from the input binners.
- `[assembler]-[sample/group]_DASTool.log`: Log file from the DAS Tool run describing the command executed and additional runtime information.
- `[assembler]-[sample/group].seqlength`: Tab-delimited file describing the length of each contig.
- `bins/[assembler]-[binner]Refined-[sample/group].*.fa`: Refined bins in fasta format.
- `unbinned/[assembler]-DASToolUnbinned-[sample/group].*.fa`: Unbinned contigs from bin refinement in fasta format.

By default, only the raw bins (and unbinned contigs) from the actual binning methods, but not from the binning refinement with DAS Tool, will be used for downstream bin quality control, annotation and taxonomic classification. The parameter `--postbinning_input` can be used to change this behaviour.

⚠ Due to ability to perform downstream QC of both raw and refined bins in parallel (via `-postbinning_input`), bin names in DAS Tools's `*_allBins.eval` file will include `Refined`. However for this particular file, they *actually* refer to the 'raw' input bins. The pipeline renames the input files prior to running DASTool to ensure they can be disambiguated from the original bin files in the downstream QC steps.

### Tiara

Tiara is a contig classifier that identifies the domain (prokarya, eukarya) of contigs within an assembly. This is used in this pipeline to rapidly and with few resources identify the most likely domain classification of each bin or unbin based on its contig identities.

▼ Output files

- `Taxonomy/Tiara/`
- `[assembler]-[sample/group].tiara.txt` - Tiara output classifications (with probabilities) for all contigs within the specified sample/group assembly
- `log_[assembler]-[sample/group].txt` - log file detailing the parameters used by the Tiara model for contig classification.
- `GenomeBinning/tiara_summary.tsv` - Summary of Tiara domain classification for all bins.

Typically, you would use `tiara_summary.tsv` as the primary file to see which bins or unbins have been classified to which domains at a glance, whereas the files in `Taxonomy/Tiara` provide classifications for each contig.

### Bin sequencing depth

For each bin or refined bin the median sequencing depth is computed based on the corresponding contig depths.

▼ Output files

- `GenomeBinning/depths/bins/`
- `bin_depths_summary.tsv`: Summary of bin sequencing depths for all samples. Depths are available for samples mapped against the corresponding assembly, i.e. according to the mapping strategy specified with `--binning_map_mode`. Only for short reads.
- `bin_refined_depths_summary.tsv`: Summary of sequencing depths for refined bins for all samples, if refinement was performed. Depths are available for samples

mapped against the corresponding assembly, i.e. according to the mapping strategy specified with `--binning_map_mode`. Only for short reads.
- `[assembler]-[binner]-[sample/group]-binDepths.heatmap.png`: Clustered heatmap showing bin abundances of the assembly across samples. Bin depths are transformed to centered log-ratios and bins as well as samples are clustered by Euclidean distance. Again, sample depths are available according to the mapping strategy specified with `--binning_map_mode`.

### QC for metagenome assembled genomes with QUAST

[QUAST]() is a tool that evaluates genome assemblies by computing various metrics. The QUAST output is in the bin directories shown below. This QUAST output is not shown in the MultiQC report.

▼ Output files

- `GenomeBinning/QC/QUAST/[assembler]-[bin]/`
- `report.*`: QUAST report in various formats, such as html, pdf, tex, tsv, or txt
- `transposed_report.*`: QUAST report that has been transposed into wide format (tex, tsv, or txt)
- `quast.log`: QUAST log file
- `metaquast.log`: MetaQUAST log file
- `icarus.html`: Icarus main menu with links to interactive viewers
- `icarus_viewers/contig_size_viewer.html`: Diagram of contigs that are ordered from longest to shortest
- `basic_stats/cumulative_plot.pdf`: Shows the growth of contig lengths (contigs are ordered from largest to shortest)
- `basic_stats/GC_content_plot.pdf`: Shows the distribution of GC content in the contigs
- `basic_stats/[assembler]-[bin]_GC_content_plot.pdf`: Histogram of the GC percentage for the contigs
- `basic_stats/Nx_plot.pdf`: Plot of Nx values as x varies from 0 to 100%.
- `predicted_genes/[assembler]-[bin].rna.gff`: Contig positions for rRNA genes in gff version 3 format
- `predicted_genes/barrnap.log`: Barrnap log file (ribosomal RNA predictor)
- `GenomeBinning/QC/`
- `[assembler]-[binner]-[domain]-[refinement]-[sample/group]-quast_summary.tsv`: QUAST output summarized per sample/condition.
- `quast_summary.tsv`: QUAST output for all bins summarized

### QC for metagenome assembled genomes

#### BUSCO

[BUSCO]() is a tool used to assess the completeness of a genome assembly. It is run on all the genome bins and high quality contigs obtained by the applied binning and/or binning refinement methods (depending on the `--postbinning_input` parameter). By default, BUSCO is run in automated lineage selection mode in which it first tries to select the domain and then a more specific lineage based on phylogenetic placement. If available, result files for both the selected domain lineage and the selected more specific lineage are placed in the output directory. If a lineage dataset is specified already with `--busco_db`, only results for this specific lineage will be generated.

▼ Output files

- `GenomeBinning/QC/BUSCO/`
- `[assembler]-[bin]_busco.log`: Log file containing the standard output of BUSCO.
- `[assembler]-[bin]_busco.err`: File containing potential error messages returned from BUSCO.
- `short_summary.domain.[lineage].[assembler]-[bin].txt`: BUSCO summary of the results for the selected domain when run in automated lineage selection mode. Not available for bins for which a viral lineage was selected.
- `short_summary.specific_lineage.[lineage].[assembler]-[bin].txt`: BUSCO summary of the results in case a more specific lineage than the domain could be selected or for the lineage provided via `--busco_db`.
- `[assembler]-[bin]_buscos.[lineage].fna.gz`: Nucleotide sequence of all identified BUSCOs for used lineages (domain or specific).
- `[assembler]-[bin]_buscos.[lineage].faa.gz`: Aminoacid sequence of all identified BUSCOs for used lineages (domain or specific).
- `[assembler]-[bin]_prodigal.gff`: Genes predicted with Prodigal.

If the parameter `--save_busco_db` is set, additionally the used BUSCO lineage datasets are stored in the output directory.

▼ Output files

- `GenomeBinning/QC/BUSCO/`
- `busco_downloads/`: All files and lineage datasets downloaded by BUSCO when run in automated lineage selection mode. (Can currently not be used to reproduce analysis, see the [nf-core/mag website documentation]() how to achieve reproducible BUSCO results).
- `reference/*.tar.gz`: BUSCO reference lineage dataset that was provided via`--busco_db`.

Besides the reference files or output files created by BUSCO, the following summary files will be generated:

▼ Output files

- `GenomeBinning/QC/`
- `busco_summary.tsv`: A summary table of the BUSCO results, with % of marker genes found. If run in automated lineage selection mode, both the results for the selected domain and for the selected more specific lineage will be given, if available.

#### CheckM

[CheckM]() CheckM provides a set of tools for assessing the quality of genomes recovered from isolates, single cells, or metagenomes. It provides robust estimates of genome completeness and contamination by using collocated sets of genes that are ubiquitous and single-copy within a phylogenetic lineage

By default, nf-core/mag runs CheckM with the `check_lineage` workflow that places genome bins on a reference tree to define lineage-marker sets, to check for completeness and contamination based on lineage-specific marker genes. and then subsequently runs `qa` to generate the summary files.

▼ Output files

- `GenomeBinning/QC/CheckM/`

- [assembler]-[binner]-[domain]-[refinement]-[sample/group]_qa.txt: Detailed statistics about bins informing completeness and contamamination scores (output of checkm qa). This should normally be your main file to use to evaluate your results.
- [assembler]-[binner]-[domain]-[refinement]-[sample/group]_wf.tsv: Overall summary file for completeness and contamination (output of checkm lineage_wf).
- [assembler]-[binner]-[domain]-[refinement]-[sample/group]/: intermediate files for CheckM results, including CheckM generated annotations, log, lineage markers etc.
- checkm_summary.tsv: A summary table of the CheckM results for all bins (output of checkm qa).

If the parameter --save_checkm_reference is set, additionally the used the CheckM reference datasets are stored in the output directory.

▼ Output files

- GenomeBinning/QC/CheckM/
- checkm_downloads/: All CheckM reference files downloaded from the CheckM FTP server, when not supplied by the user.
  - checkm_data_2015_01_16/*: a range of directories and files required for CheckM to run.

### GUNC

Genome UNClutterer (GUNC) is a tool for detection of chimerism and contamination in prokaryotic genomes resulting from mis-binning of genomic contigs from unrelated lineages. It does so by applying an entropy based score on taxonomic assignment and contig location of all genes in a genome. It is generally considered as a additional complement to CheckM results.

▼ Output files

- GenomeBinning/QC/gunc_summary.tsv
- GenomeBinning/QC/gunc_checkm_summary.tsv
- [gunc-database].dmnd
- GUNC/
- raw/
  - [assembler]-[binner]-[domain]-[refinement]-[sample/group]/GUNC_checkM.merged.tsv: Per sample GUNC output containing with taxonomic and completeness QC statistics.
- checkmmerged/
  - [assembler]-[binner]-[domain]-[refinement]-[sample/group]/GUNC.progenomes_2.1.maxCSS_level.tsv: Per sample GUNC output merged with output from CheckM

GUNC will be run if specified with --run_gunc as a standalone, unless CheckM is also activated via --qc_tool 'checkm', in which case GUNC output will be merged with the CheckM output using gunc merge_checkm.

If --gunc_save_db is specified, the output directory will also contain the requested database (progenomes, or GTDB) in DIAMOND format.

## Taxonomic classification of binned genomes

### CAT

CAT is a toolkit for annotating contigs and bins from metagenome-assembled-genomes. The nf-core/mag pipeline uses CAT to assign taxonomy to genome bins based on the taxnomy of the contigs.

▼ Output files

- Taxonomy/CAT/[assembler]/[binner]/
- [assembler]-[binner]-[domain]-[refinement]-[sample/group].ORF2LCA.names.txt.gz: Tab-delimited files containing the lineage of each contig, with full lineage names
- [assembler]-[binner]-[domain]-[refinement]-[sample/group].bin2classification.names.txt.gz: Taxonomy classification of the genome bins, with full lineage names
- Taxonomy/CAT/[assembler]/[binner]/raw/
- [assembler]-[binner]-[domain]-[refinement]-[sample/group].concatenated.predicted_proteins.faa.gz: Predicted protein sequences for each genome bin, in fasta format
- [assembler]-[binner]-[domain]-[refinement]-[sample/group].concatenated.predicted_proteins.gff.gz: Predicted protein features for each genome bin, in gff format
- [assembler]-[binner]-[domain]-[refinement]-[sample/group].ORF2LCA.txt.gz: Tab-delimited files containing the lineage of each contig
- [assembler]-[binner]-[domain]-[refinement]-[sample/group].bin2classification.txt.gz: Taxonomy classification of the genome bins
- [assembler]-[binner]-[domain]-[refinement]-[sample/group].log: Log files

If the parameters --cat_db_generate and --save_cat_db are set, additionally the generated CAT database is stored:

▼ Output files

- Taxonomy/CAT/CAT_prepare_*.tar.gz: Generated and used CAT database.

### GTDB-Tk

GTDB-Tk is a toolkit for assigning taxonomic classifications to bacterial and archaeal genomes based on the Genome Database Taxonomy GTDB. nf-core/mag uses GTDB-Tk to classify binned genomes which satisfy certain quality criteria (i.e. completeness and contamination assessed with the BUSCO analysis).

▼ Output files

- Taxonomy/GTDB-Tk/[assembler]/[binner]/[sample/group]/
- gtdbtk.[assembler]-[binner]-[sample/group].{bac120/ar122}.summary.tsv: Classifications for bacterial and archaeal genomes (see the GTDB-Tk documentation for details).
- gtdbtk.[assembler]-[binner]-[domain]-[refinement]-[sample/group].{bac120/ar122}.classify.tree.gz: Reference tree in Newick format containing query genomes placed with pplacer.
- gtdbtk.[assembler]-[binner]-[domain]-[refinement]-[sample/group].{bac120/ar122}.markers_summary.tsv: A summary of unique, duplicated, and missing markers within the 120 bacterial marker set, or the 122 archaeal marker set for each submitted genome.
- gtdbtk.[assembler]-[binner]-[domain]-[refinement]-[sample/group].{bac120/ar122}.msa.fasta.gz: FASTA file containing MSA of submitted and reference genomes.
- gtdbtk.[assembler]-[binner]-[domain]-[refinement]-[sample/group].{bac120/ar122}.filtered.tsv: A list of genomes with an insufficient number of amino acids in MSA.
- gtdbtk.[assembler]-[binner]-[domain]-[refinement]-[sample/group].*.log: Log files.
- gtdbtk.[assembler]-[binner]-[domain]-[refinement]-[sample/group].failed_genomes.tsv: A list of genomes for which the GTDB-Tk analysis failed, e.g. because Prodigal could not detect any genes.
- Taxonomy/GTDB-Tk/gtdbtk_summary.tsv: A summary table of the GTDB-Tk classification results for all bins, also containing bins which were discarded based on the

BUSCO QC, which were filtered out by GTDB-Tk (listed in `*.filtered.tsv`) or for which the analysis failed (listed in `*.failed_genomes.tsv`).

## Genome annotation of binned genomes

### Prokka

Whole genome annotation is the process of identifying features of interest in a set of genomic DNA sequences, and labelling them with useful information. Prokka is a software tool to annotate bacterial, archaeal and viral genomes quickly and produce standards-compliant output files.

▼ Output files

- `Annotation/Prokka/[assembler]/[bin]/`
- `[assembler]-[binner]-[bin].gff`: annotation in GFF3 format, containing both sequences and annotations
- `[assembler]-[binner]-[bin].gbk`: annotation in GenBank format, containing both sequences and annotations
- `[assembler]-[binner]-[bin].fna`: nucleotide FASTA file of the input contig sequences
- `[assembler]-[binner]-[bin].faa`: protein FASTA file of the translated CDS sequences
- `[assembler]-[binner]-[bin].ffn`: nucleotide FASTA file of all the prediction transcripts (CDS, rRNA, tRNA, tmRNA, misc_RNA)
- `[assembler]-[binner]-[bin].sqn`: an ASN1 format "Sequin" file for submission to Genbank
- `[assembler]-[binner]-[bin].fsa`: nucleotide FASTA file of the input contig sequences, used by "tbl2asn" to create the .sqn file
- `[assembler]-[binner]-[bin].tbl`: feature Table file, used by "tbl2asn" to create the .sqn file
- `[assembler]-[binner]-[bin].err`: unacceptable annotations - the NCBI discrepancy report.
- `[assembler]-[binner]-[bin].log`: contains all the output that Prokka produced during its run
- `[assembler]-[binner]-[bin].txt`: statistics relating to the annotated features found
- `[assembler]-[binner]-[bin].tsv`: tab-separated file of all features (locus_tag, ftype, len_bp, gene, EC_number, COG, product)

### MetaEuk

In cases where eukaryotic genomes are recovered in binning,MetaEuk is also available to annotate eukaryotic genomes quickly with standards-compliant output files.

▼ Output files

- `Annotation/MetaEuk/[assembler]/[bin]`
- `[assembler]-[binner]-[bin].fas`: fasta file of protein sequences identified by MetaEuk
- `[assembler]-[binner]-[bin].codon.fas`: fasta file of nucleotide sequences corresponding to the protein sequences fasta
- `[assembler]-[binner]-[bin].headersMap.tsv`: tab-separated table containing the information from each header in the fasta files
- `[assembler]-[binner]-[bin].gff`: annotation in GFF3 format

## Additional summary for binned genomes

▼ Output files

- `GenomeBinning/bin_summary.tsv`: Summary of bin sequencing depths together with BUSCO, CheckM, QUAST and GTDB-Tk results, if at least one of the later was generated. This will also include refined bins if `--refine_bins_dastool` binning refinement is performed. Note that in contrast to the other tools, for CheckM the bin name given in the column "Bin Id" does not contain the ".fa" extension.

## Ancient DNA

Optional, only running when parameter `-profile ancient_dna` is specified.

#### `PyDamage`

Pydamage, is a tool to automate the process of ancient DNA damage identification and estimation from contigs. After modelling the ancient DNA damage using the C to T transitions, Pydamage uses a likelihood ratio test to discriminate between truly ancient, and modern contigs originating from sample contamination.

▼ Output files

- `Ancient_DNA/pydamage/analyze`
- `[assembler]_[sample/group]/pydamage_results/pydamage_results.csv`: PyDamage raw result tabular file in `.csv` format. Format described here: pydamage.readthedocs.io/en/0.62/output.html
- `Ancient_DNA/pydamage/filter`
- `[assembler]_[sample/group]/pydamage_results/pydamage_results.csv`: PyDamage filtered result tabular file in `.csv` format. Format described here: pydamage.readthedocs.io/en/0.62/output.html

#### `variant_calling`

Because of aDNA damage, *de novo* assemblers sometimes struggle to call a correct consensus on the contig sequence. To avoid this situation, the consensus is optionally re-called with a variant calling software using the reads aligned back to the contigs when `--run_ancient_damagecorrection` is supplied.

▼ Output files

- `variant_calling/consensus`
- `[assembler]_[sample/group].fa`: contigs sequence with re-called consensus from read-to-contig alignment
- `variant_calling/unfiltered`
- `[assembler]_[sample/group].vcf.gz`: raw variant calls of the reads aligned back to the contigs.
- `variant_calling/filtered`
- `[assembler]_[sample/group].filtered.vcf.gz`: quality filtered variant calls of the reads aligned back to the contigs.

### MultiQC

▼ Output files

- `multiqc/`
- `multiqc_report.html`: a standalone HTML file that can be viewed in your web browser.
- `multiqc_data/`: directory containing parsed statistics from the different tools used in the pipeline.
- `multiqc_plots/`: directory containing static images from the report in various formats.

MultiQC is a visualization tool that generates a single HTML report summarising all samples in your project. Most of the pipeline QC results are visualised in the report and further statistics are available in the report data directory.

Results generated by MultiQC collate pipeline QC from supported tools e.g. FastQC. The pipeline has special steps which also allow the software versions to be reported in the MultiQC output for future traceability. For more information about how to use MultiQC reports, see http://multiqc.info.

The general stats table at the top of the table will by default only display the most relevant pre- and post-processing statistics prior to assembly, i.e., FastQC, fastp/Adapter removal, and Bowtie2 PhiX and host removal mapping results.

Note that the FastQC raw and processed columns are right next to each other for improved visual comparability, however the processed columns represent the input reads *after* fastp/Adapter Removal processing (the dedicated columns of which come directly after the two FastQC set of columns). Hover your cursor over each column name to see the which tool the column is derived from.

Summary tool-specific plots and tables of following tools are currently displayed (if activated):

- FastQC (pre- and post-trimming)
- fastp
- Adapter Removal
- bowtie2
- BUSCO
- QUAST
- Kraken2 / Centrifuge
- PROKKA

## Pipeline information

▼ Output files

- `pipeline_info/`
- Reports generated by Nextflow: `execution_report.html`, `execution_timeline.html`, `execution_trace.txt` and `pipeline_dag.dot`/`pipeline_dag.svg`.
- Reports generated by the pipeline: `pipeline_report.html`, `pipeline_report.txt` and `software_versions.yml`. The `pipeline_report*` files will only be present if the `--email` / `--email_on_fail` parameter's are used when running the pipeline.
- Parameters used by the pipeline run: `params.json`.

Nextflow provides excellent functionality for generating various reports relevant to the running and execution of the pipeline. This will allow you to troubleshoot errors with the running of the pipeline, and also provide you with other information such as launch commands, run times and resource usage.