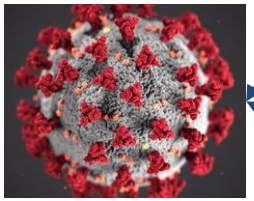
Introduction to viral genome reconstruction using massive sequencing: Sars-cov-2 use case

BU-ISCIII
Bioinformatics Unit, Institute of Health Carlos III
Madrid, Spain

Máster Virología - UCM

November 25 2020

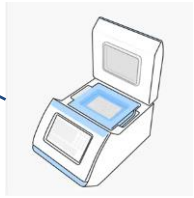
Background



December 2019

SARS-Cov-2

Surveillance:
Genome Sequencing



Detection:
RT-PCR

- RT-PCR Primer & Probe efficiency
- Viral variability
- behaviour
- Correlate viral lineage & severity
- Help vaccine development
- Future established disease control (flu)
- Coordinated & rapid response
- Understand immune response to infection

Sequencing Approaches

Amplicon-based enrichment

Probe capture enrichment

Wet-lab

ONT long reads

Illumina short reads

Dry-lab

ARTIC network

nf-core/viralrecon

Assembly and intrahost/low-frequency variant calling for viral samples

Data (TB)

State of the art

January 22 2020 Artic Network protocols

February 21 2020 Galaxy SARS-Cov2 project

https://github.com/BU-ISCIII/SARS_Cov2_consensus-nf

March 18 2020 Our 1st pipeline started

March 30 2020 nf-core collaboration

April 5 2020 1st COVID19 Virtual BioHackathon

nf-core/viralrecon

Assembly and intrahost/low-frequency variant calling for viral samples

June 1 2020 viralrecon released v1.0.0

https://github.com/jaleezyy/covid-19-signal

June 29 2020 covid-19-signal v1.0.0

https://github.com/nodrogluap/nanostripper

September 3 2020 nanostripper v1.0.0

Viralrecon

nf-core/viralrecon

https://github.com/nf-core/viralrecon

nextflow

Multiple compute infrastructures

Portable

Easy to install

Reproducible

Stable code

The need of standardisation I

Sequencing techniques are starting to be used in clinical diagnosis, and therefore workflows have to assure:

- **Reproducibility**

Results always have to be reproducible

- **Portability**

The analysis workflow must be executable in different platforms

- **Scalability**

The analysis workflow must be able to work with different numbers of samples

Nextflow I

- **Nextflow** is a DSL for parallel and scalable computational pipelines.
- It enables **scalable and reproducible scientific workflows** using software **containers**.
- It **allows the adaptation of pipelines** written in the most common scripting languages.
- Its fluent **DSL** simplifies the implementation and the deployment of complex parallel and reactive workflows on clouds and clusters.

Nextflow II

Fast prototyping

Nextflow allows you to write a computational pipeline by making it simpler to put together many different tasks.

You may reuse your existing scripts and tools and you don't need to learn a new language or API to start using it.

Reproducibility

Nextflow supports Docker and Singularity containers technology.

This, along with the integration of the GitHub code sharing platform, allows you to write self-contained pipelines, manage versions and to rapidly reproduce any former configuration.

Portable

Nextflow provides an abstraction layer between your pipeline's logic and the execution layer, so that it can be executed on multiple platforms without it changing.

It provides out of the box executors for SGE, LSF, SLURM, PBS and HTCondor batch schedulers and for Kubernetes and Amazon AWS cloud platforms.

Unified parallelism

Nextflow is based on the *dataflow* programming model which greatly simplifies writing complex distributed pipelines.

Parallelisation is implicitly defined by the processes input and output declarations. The resulting applications are inherently parallel and can scale-up or scale-out, transparently, without having to adapt to a specific platform architecture.

Continuous checkpoints

All the intermediate results produced during the pipeline execution are automatically tracked.

This allows you to resume its execution, from the last successfully executed step, no matter what the reason was for it stopping.

Stream oriented

Nextflow extends the Unix pipes model with a fluent DSL, allowing you to handle complex stream interactions easily.

It promotes a programming approach, based on functional composition, that results in resilient and easily reproducible pipelines.

Nextflow III: nf-core

The nf-core project is a diverse project spread across many groups. It is A community effort to collect a curated set of analysis pipelines built using Nextflow.

nf-core

nf-core/mag ✓ ☆ 42

annotation assembly binning metagenomics

Assembly and binning of metagenomes

Version **1.2.0** Published 3 weeks ago

nf-core/ampliseq ✓ ☆ 54

16s amplicon-sequencing metagenomics qiime rrna

16S rRNA amplicon sequencing analysis workflow using QIIME2

Version **1.2.0** Published 4 weeks ago

nf-core/sarek ✓ ☆ 93

annotation cancer gatk4 genomics germline pre-processing somatic variant-calling

Analysis pipeline to detect germline or somatic variants (pre-processing, variant calling and annotation) from WGS / targeted sequencing

Version **2.7** Published 1 month ago

nf-core/eager ✓ ☆ 45

adna ancient-dna-analysis ancientdna genome metagenomics pathogen-genomics population-genetics

A fully reproducible and state-of-the-art ancient DNA analysis pipeline

Version **2.3.1** Published 2 months ago

nf-core/cageseq ✓ ☆ 3

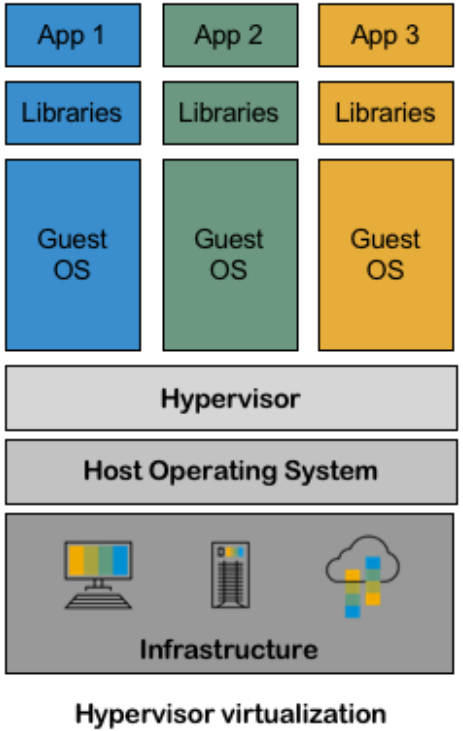
nf-core/rnaseq ✓ ☆ 298

Containers I

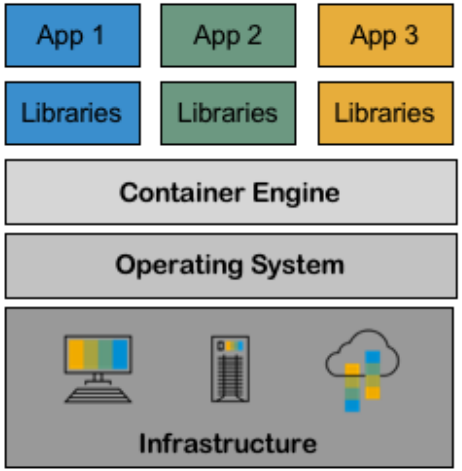
Linux containers is a generic term for an implementation of operating system-level virtualization for the Linux operating system.

Containers allow us to **port** pipelines and **replicate** their exact execution environments across different hardware.

Singularity I



Hypervisor virtualization



Container virtualization

Singularity II

Singularity is a free, cross-platform and open-source computer program that performs operating-system-level virtualization.

One of the main uses of Singularity is to bring containers and **reproducibility to scientific computing** and the HPC world.

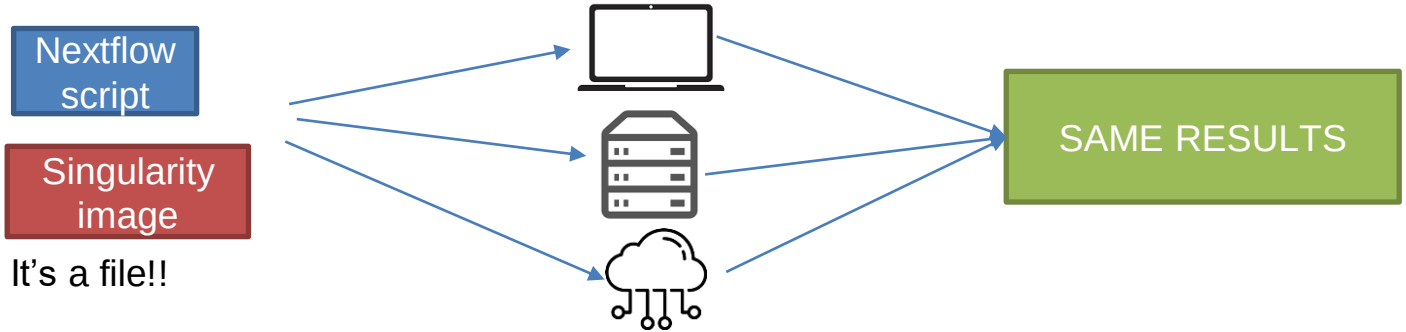
While Docker is broadly used, Singularity is **fully compatible with Docker**, plus Singularity does **not require root permissions** to be executed.

And now is when you think…

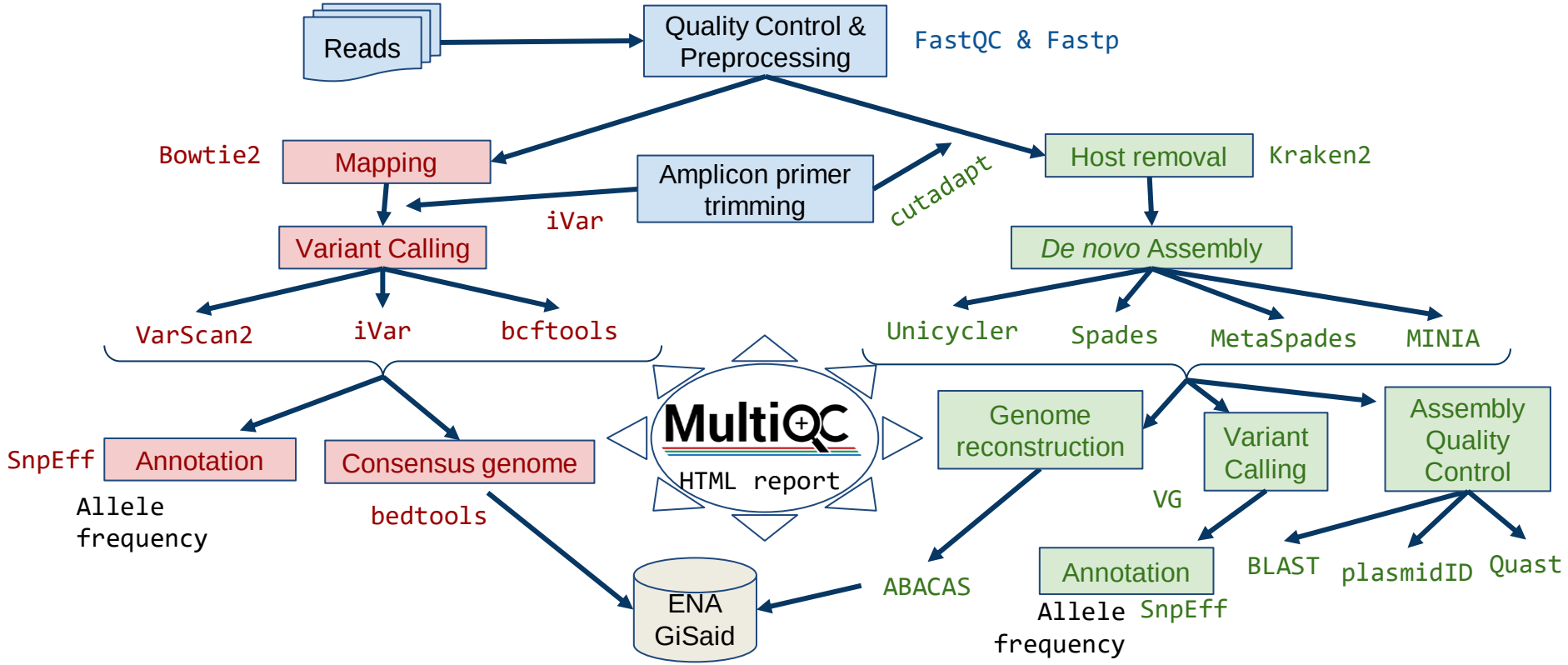
How all this hard to understand stuff can make my life easier???

How all this hard to understand stuff can make my life easier?

- Reproducibility is imperative in clinical bioinformatics.
- Once this is all set you can just:
  - Run your pipeline ANYWHERE getting same results.
  - With ANYWHERE I mean your laptop, your home pc, your computation cluster, just one Amazon server you rent for some time…
  - You don’t need to change ANYTHING in your code, just some config.
  - You don’t need to install ANYTHING anywhere EVERYTHING is installed and in the same versión in your shiny container.



Viralrecon

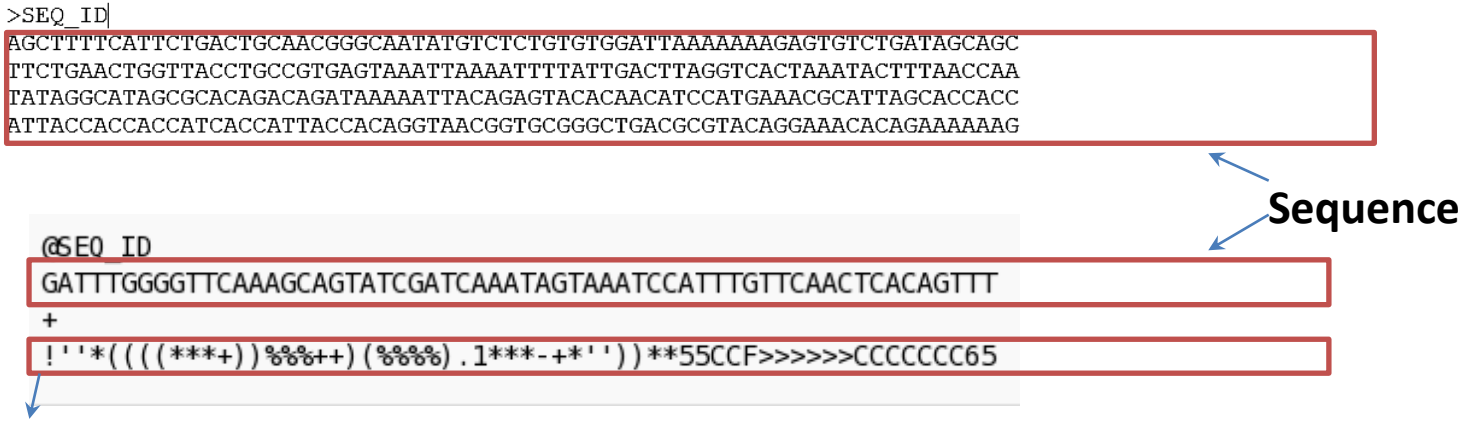

Reads
Quality Control & Preprocessing
FastQC & Fastp
Bowtie2
Mapping
Amplicon primer trimming
cutadapt
Host removal
Kraken2
iVar
Variant Calling
De novo Assembly
VarScan2
iVar
bcftools
Unicycler
Spades
MetaSpades
MINIA
SnpEff
Annotation
Consensus genome
MultiQC
HTML report
Genome reconstruction
Variant Calling
Assembly Quality Control
Allele frequency
bedtools
VG
ENA GiSaid
ABACAS
Annotation
BLAST
plasmidID
Quast
Allele frequency
SnpEff

Input data

Which input format do we have from massive sequencing?

FASTQ format

- Is a FASTA file with quality information
- Within HTS, FASTA contain genomes y FASTQ reads



Quality: must be 1 bit

Sequencing quality assessment

- To assess quality, software uses **Phred per-base quality** score is used
- Is the **first quality control step** after sequencing. There should be one after every step of the analysis
- After quality assessment user can know how **reliable** are their datasets
- QC will determine the next **filtering** step
- Filtering decisions will **impact** directly in **further analysis**
- Many other steps also use this quality as variable in their **algorithms**

FastQC: Basic Statistics

- Self defined overall stats
  - Encoding: Phred33 or Phred64

Basic Statistics

Measure	Value
Filename	bad_sequence.txt
File type	Conventional base calls
Encoding	Illumina 1.5
Total Sequences	395288
Sequences flagged as poor quality	0
Sequence length	40
%GC	47

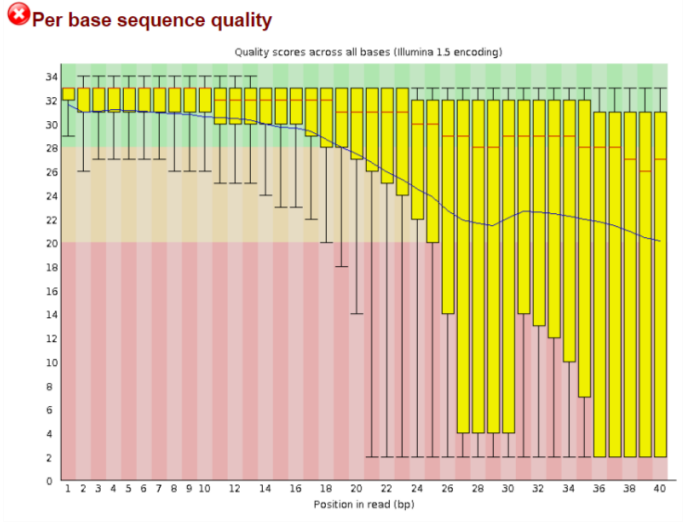
Basic Statistics

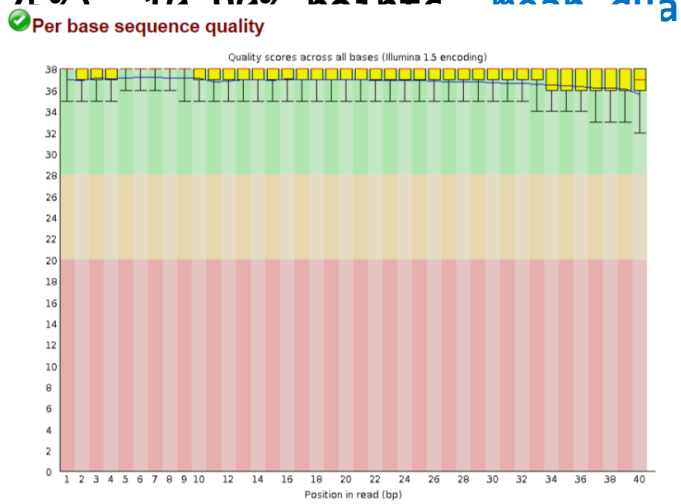
Measure	Value
Filename	good_sequence_short.txt
File type	Conventional base calls
Encoding	Illumina 1.5
Total Sequences	250000
Sequences flagged as poor quality	0
Sequence length	40
%GC	45

FastQC: Per base sequence quality

- Overview of the range of quality values across all bases at each position in the FastQ file

range (25-

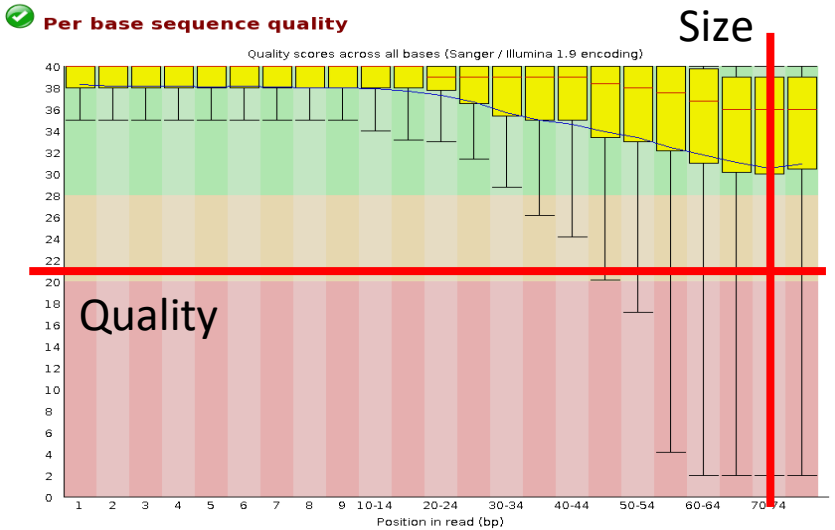




Sequence filtering

- **Remove residual adapters**
  - Depending on used library

- **Filtering parameters**
  - Quality filtering
    - Overall mean quality
    - Local mean quality
      - Sequence end
      - Sliding window
  - Size filtering
    - Overall sequence size
    - Remaining sequence size after filtering



Mapping

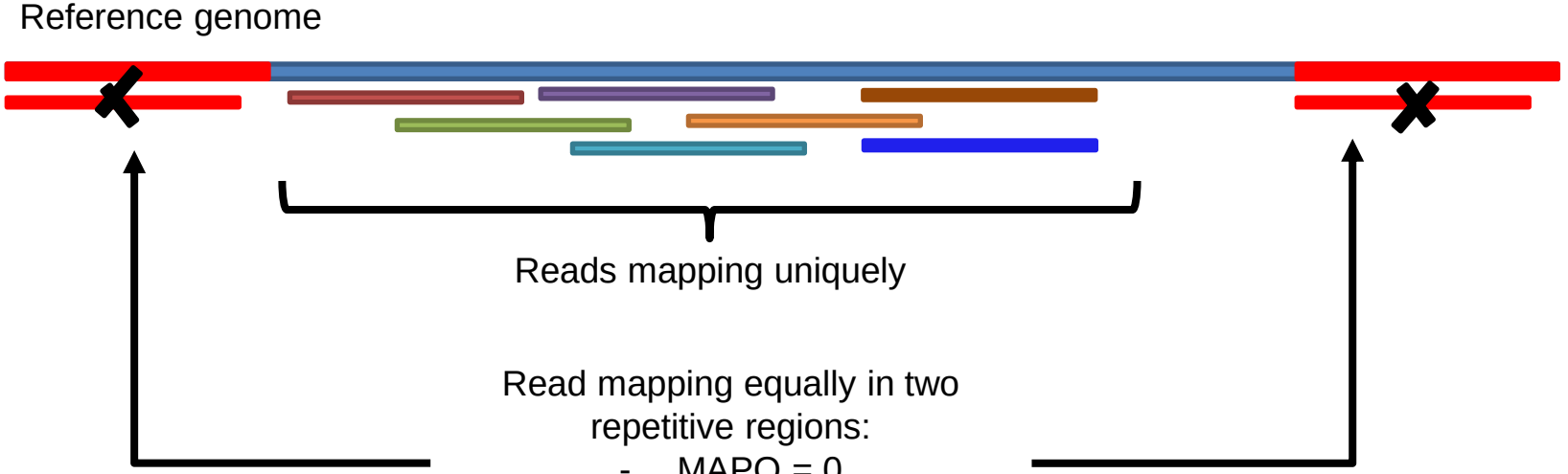
BOWTIE2

Mapping

BOWTIE2

Mapping software looks for the best match for each read in the genome.

Paired-end reads help the mapper to find the perfect spot!



Depends on parameters!!

Input data

Which output format do we have from mapping step?

SAM format

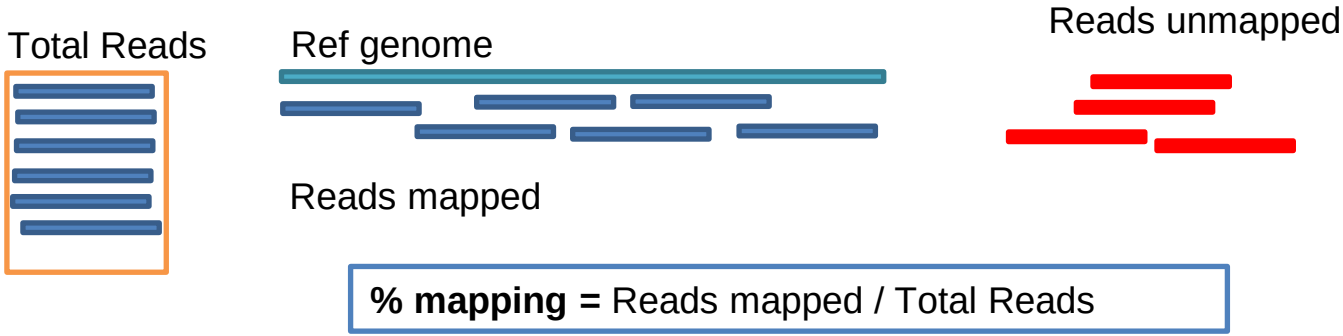
Col	Field	Type	Regexp/Range	Brief description
1	QNAME	String	`[!-?A-~]{1,255}`	Query template NAME
2	FLAG	Int	$[0,2^{16}-1]$	bitwise FLAG
3	RNAME	String	`\*|[!-()+-<>-~][!-~]*`	Reference sequence NAME
4	POS	Int	$[0,2^{31}-1]$	1-based leftmost mapping POSition
5	MAPQ	Int	$[0,2^{8}-1]$	MAPping Quality
6	CIGAR	String	`\*|([0-9]+[MIDNSHPX=])+`	CIGAR string
7	RNEXT	String	`\*|=|[!-()+-<>-~][!-~]*`	Ref. name of the mate/next read
8	PNEXT	Int	$[0,2^{31}-1]$	Position of the mate/next read
9	TLEN	Int	$[-2^{31}+1,2^{31}-1]$	observed Template LENgth
10	SEQ	String	`\*|[A-Za-z=.]+`	segment SEQuence
11	QUAL	String	`[!-~]+`	ASCII of Phred-scaled base QUALity+33

```
@HD VN:1.5 SO:coordinate
@SQ SN:ref LN:45
r001    99 ref  7 30 8M2I4M1D3M = 37  39 TTAGATAAAGGATACTG *
r002     0 ref  9 30 3S6M1P1I4M *  0   0 AAAAGATAAGGATA     *
r003     0 ref  9 30 5S6M       *  0   0 GCCTAAGCTAA        * SA:Z:ref,29,-,6H5M,17,0;
r004     0 ref 16 30 6M14N5M    *  0   0 ATAGCTTCAGC        *
r003  2064 ref 29 17 6H5M       *  0   0 TAGGC              * SA:Z:ref,9,+,5S6M,30,1;
r001   147 ref 37 30 9M         =  7 -39 CAGCGGCAT          * NM:i:1
```

Mapping quality control

- **% mapping:** number of reads mapping againts reference genome.

Picard
Samtools



% mapping = Reads mapped / Total Reads

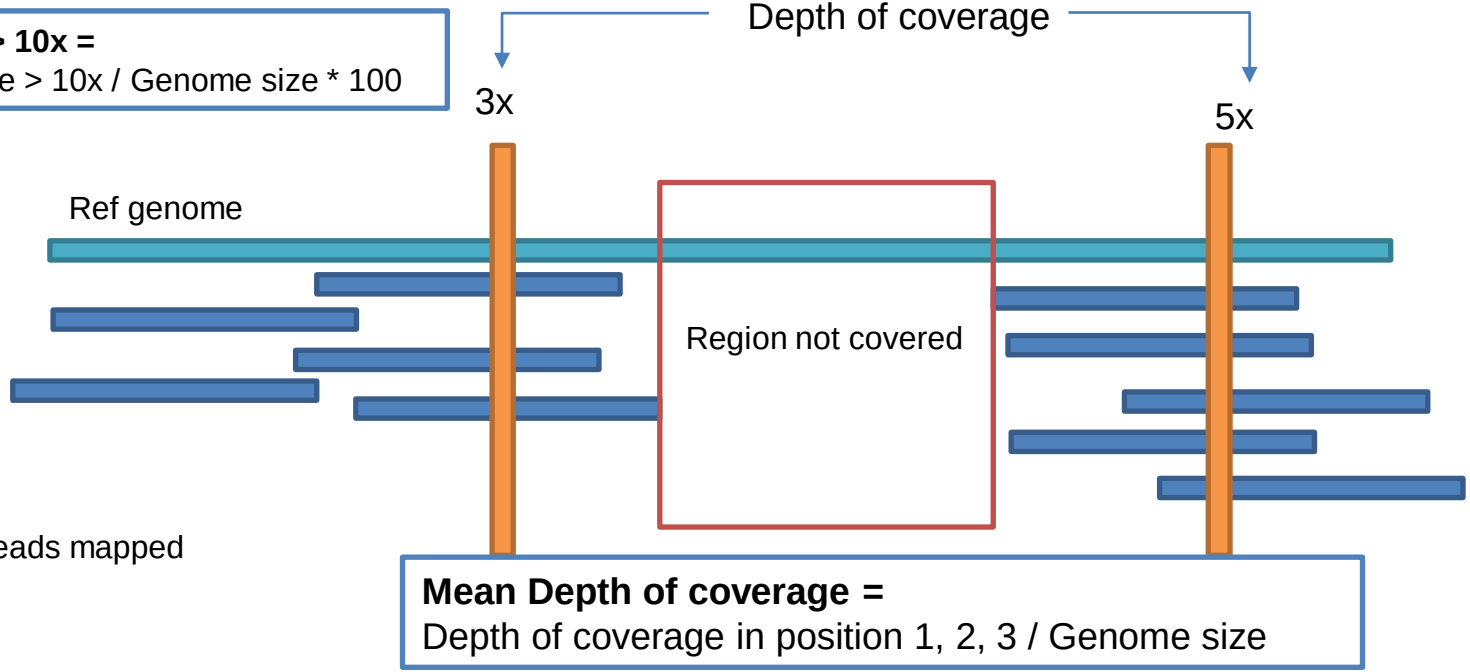
Mandatory parameter for microbial genomics!! It indicates us how many reads we have from our organism of interest. In human genomics this is almost always 99.99% unless something terrible happens. Not here!!!

Mapping quality control

- **% genome > 10x:** percentage of genome covered with more than 10 reads.
- **Mean Depth of coverage:** mean of reads covering a genome position.

Picard
Samtools

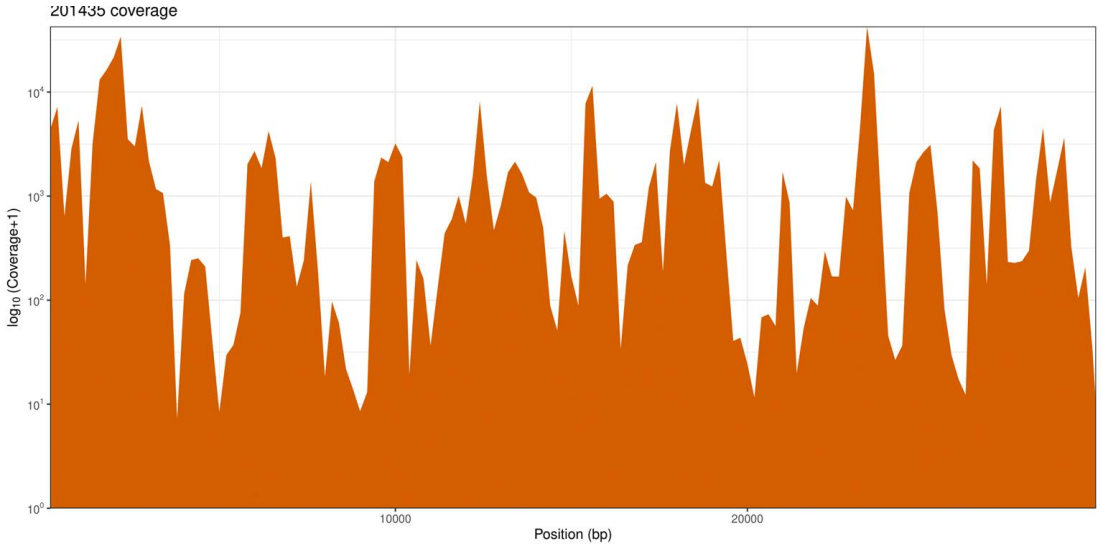
% genome > 10x =
Genome size > 10x / Genome size * 100



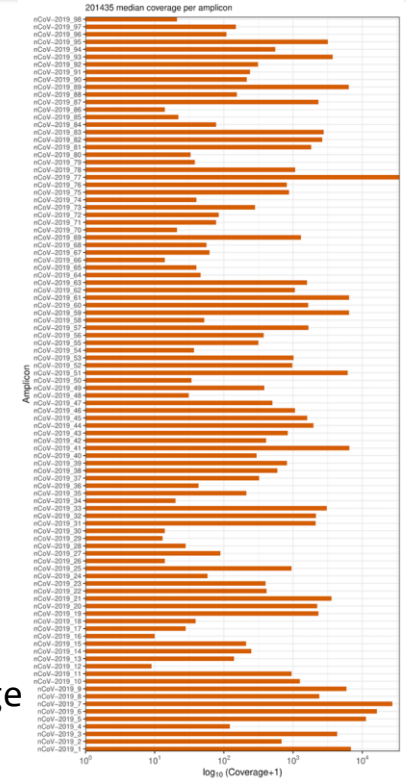
Mean Depth of coverage =
Depth of coverage in position 1, 2, 3 / Genome size

Amplicon QC results

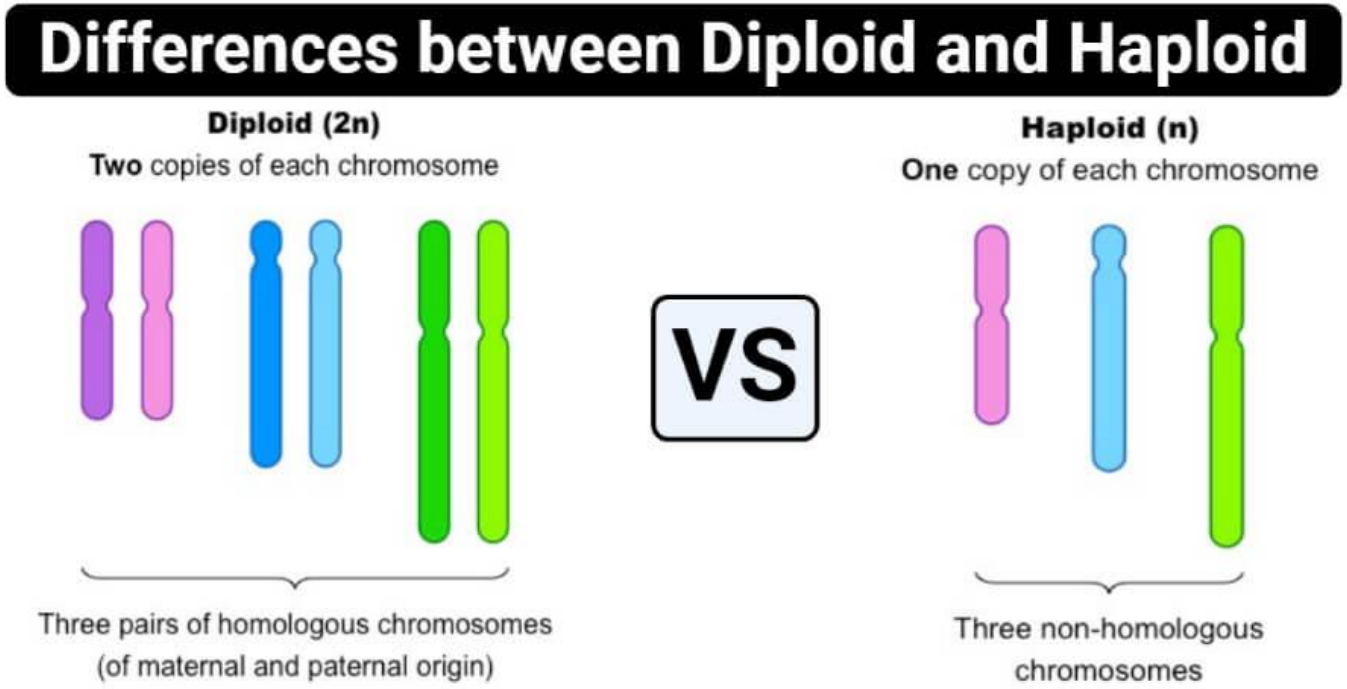
Genome coverage



Amplicon coverage



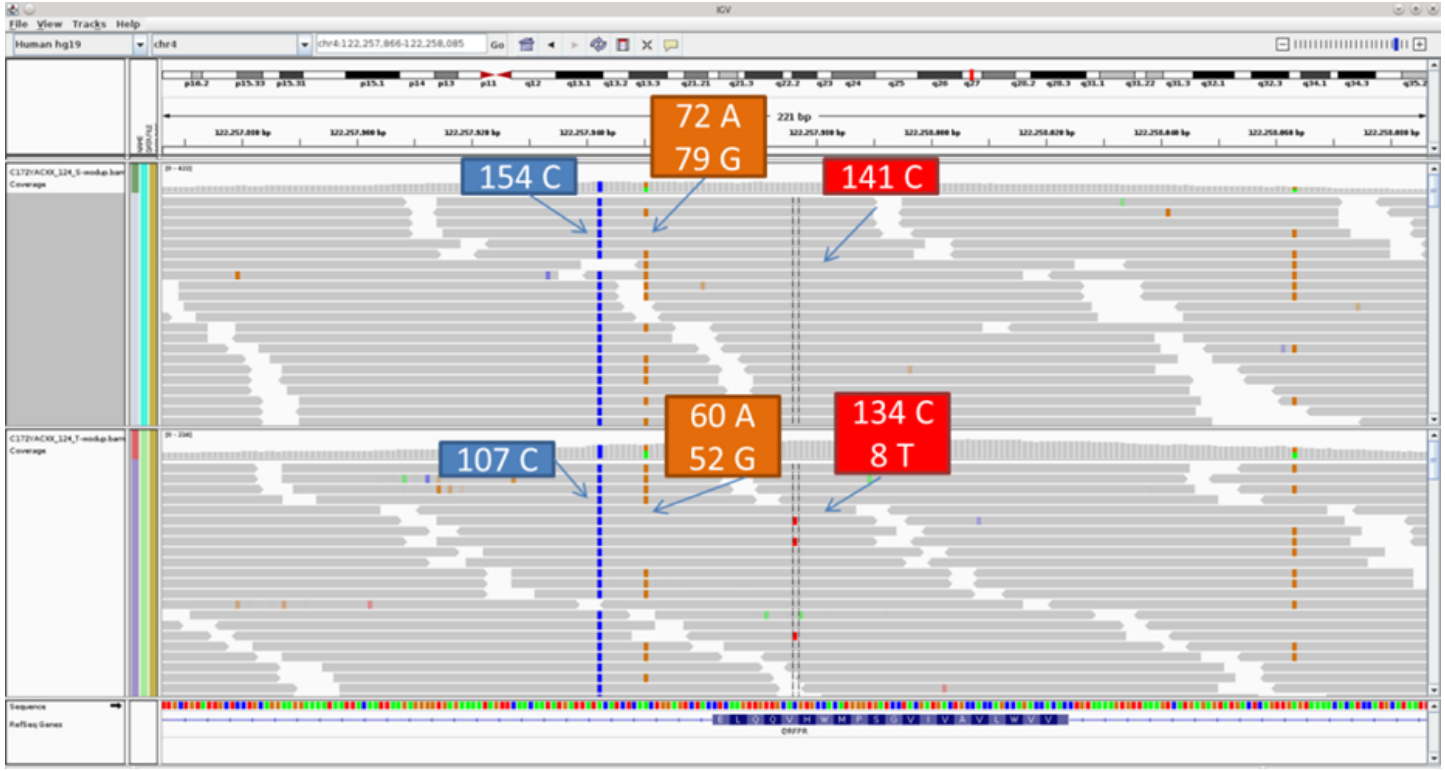
Diploid vs Haploid



Diploid vs Haploid

How do we see this differences on massive sequencing?

Diploid vs Haploid

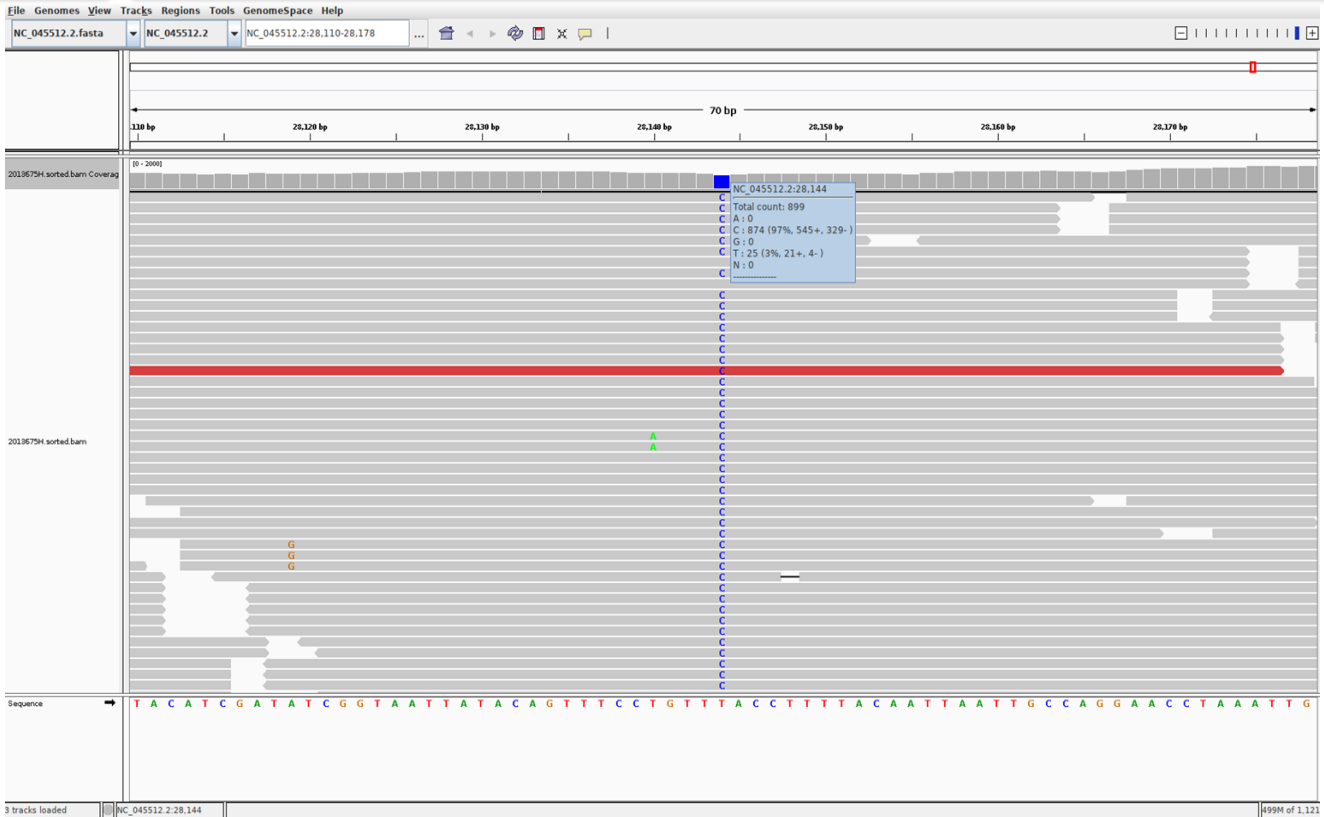


We must say variant callers which type of genotypes we are looking for.

Diploid: 1/1, 0/1 or 1/1

Haploid:1 or 0.

Variant calling



Variant callers walk through each alignment column independently and evaluate if there is a variant.

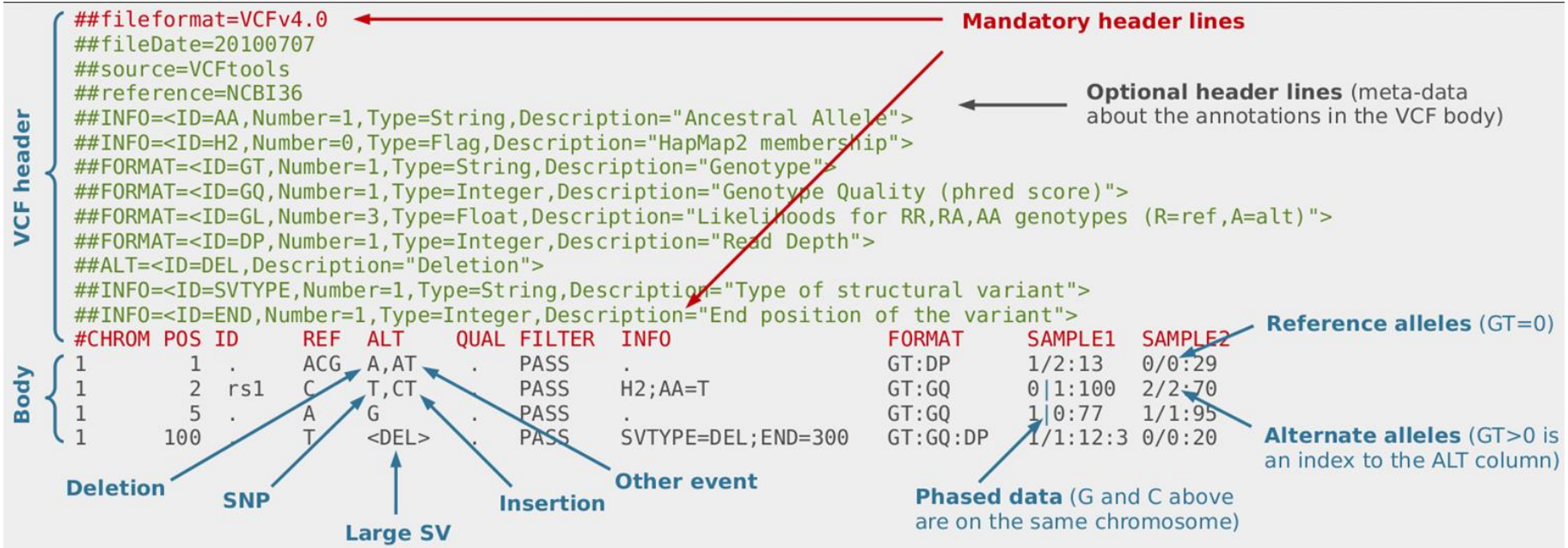
A variant is called in haploid genomes when it's supported by at least 90% of the reads.

VARSCAN2
IVAR

Input data

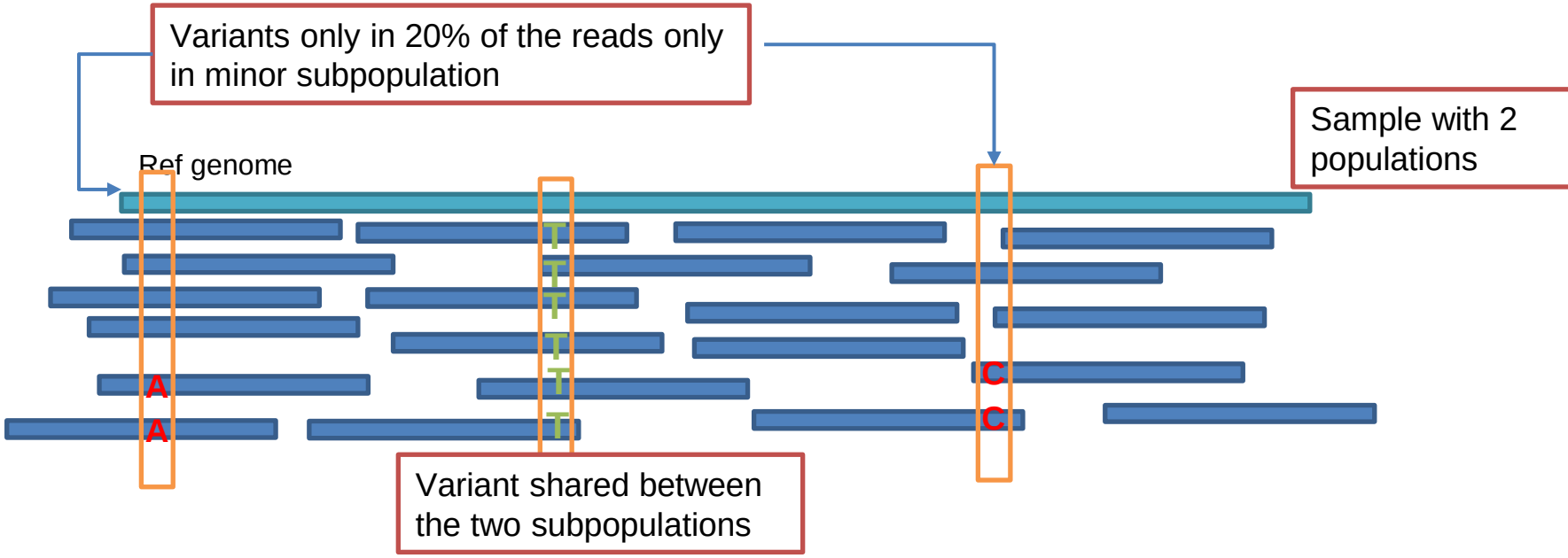
Which output format do we have from variant calling step?

VCF format


VCF header
Body
##fileformat=VCFv4.0
##fileDate=20100707
##source=VCFtools
##reference=NCBI36
##INFO=<ID=AA,Number=1,Type=String,Description="Ancestral Allele">
##INFO=<ID=H2,Number=0,Type=Flag,Description="HapMap2 membership">
##FORMAT=<ID=GT,Number=1,Type=String,Description="Genotype">
##FORMAT=<ID=GQ,Number=1,Type=Integer,Description="Genotype Quality (phred score)">
##FORMAT=<ID=GL,Number=3,Type=Float,Description="Likelihoods for RR,RA,AA genotypes (R=ref,A=alt)">
##FORMAT=<ID=DP,Number=1,Type=Integer,Description="Read Depth">
##ALT=<ID=DEL,Description="Deletion">
##INFO=<ID=SVTYPE,Number=1,Type=String,Description="Type of structural variant">
##INFO=<ID=END,Number=1,Type=Integer,Description="End position of the variant">
#CHROM POS ID REF ALT QUAL FILTER INFO FORMAT SAMPLE1 SAMPLE2
1 1 . ACG A,AT . PASS . GT:DP 1/2:13 0/0:29
1 2 rs1 C T,CT . PASS H2;AA=T GT:GQ 0|1:100 2/2:70
1 5 . A G . PASS . GT:GQ 1|0:77 1/1:95
1 100 . T <DEL> . PASS SVTYPE=DEL;END=300 GT:GQ:DP 1/1:12:3 0/0:20
Mandatory header lines
Optional header lines (meta-data about the annotations in the VCF body)
Reference alleles (GT=0)
Alternate alleles (GT>0 is an index to the ALT column)
Deletion
SNP
Large SV
Insertion
Other event
Phased data (G and C above are on the same chromosome)

Viral subpopulation - Quasispecies

- Just as in clonal subpopulations in tumor samples, we can have viral subpopulations called quasispecies in viral samples.
- We detect them using the alternative allele frequency.

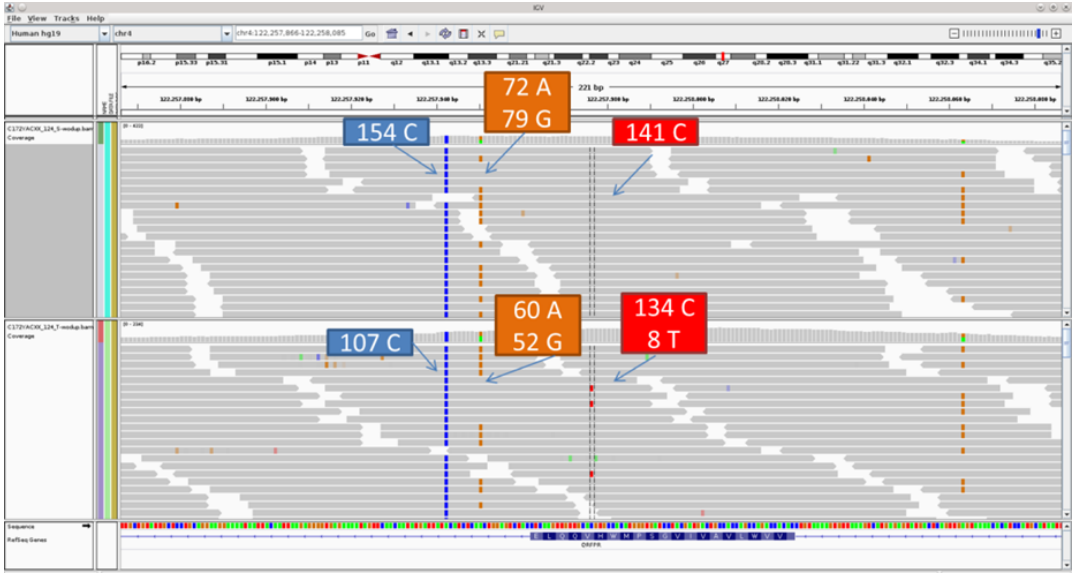


Population Allele frequency vs Sample Allele frequency

- **Population allele frequency:** probability of finding an allele in the population. Number of individuals carrying an allele vs total of individuals in the population.

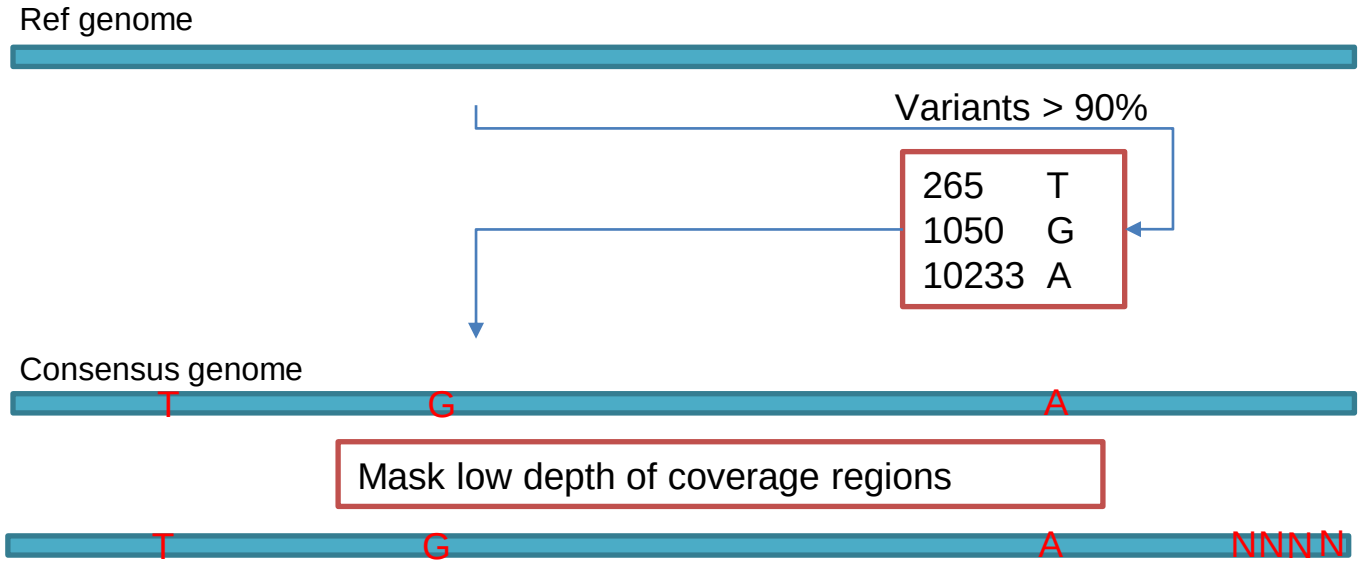
Population Allele frequency vs Sample Allele frequency

- **Alternate/Base allele frequency**: number of reads supporting the alternate allele vs total of reads.



Consensus genome

- Select variants: > 90% allele frequency
- Include variants in reference genome.
- Mask low frequency positions: <10x.



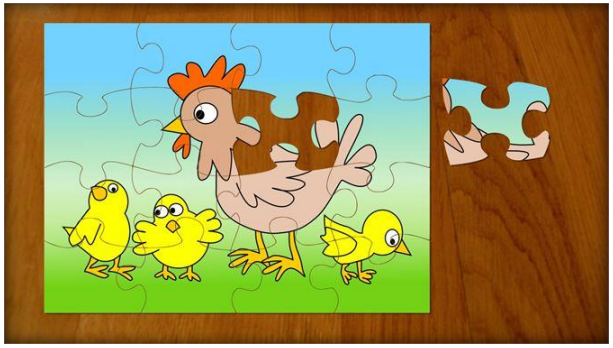
Assembly

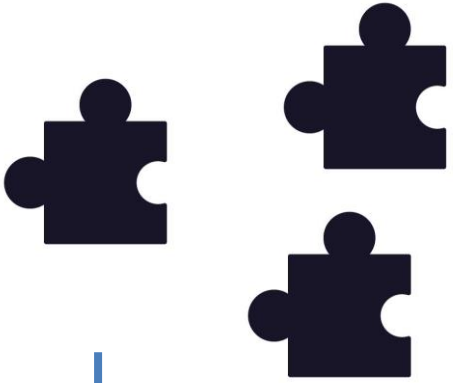
Reconstruct a representation of the original DNA from shorter DNA sequences or small fragments known as reads

- ***De novo:*** with no previous knowledge of the genome to be assembled. It overlap the end of the end of each read in order to créate a longer sequence.

- ***Assembly with reference:*** A similar but not identical genome guides the assembly process. Map reads over supplied genome.

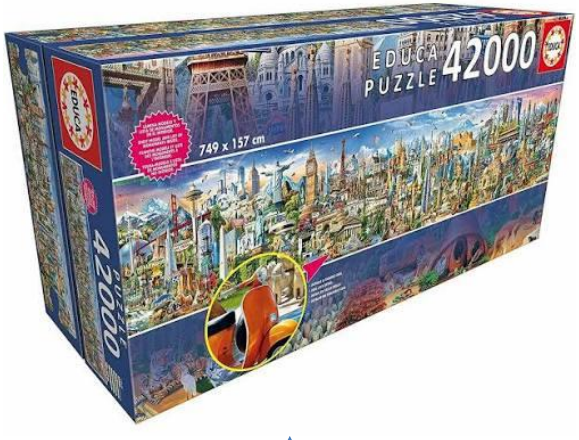
Assembly





The bigger the pieces the easier to reconstruct the puzzle!

Assembly



Obviously this a little bit more complicated….

We have LOADS of really little pieces and a big puzzle in proportion

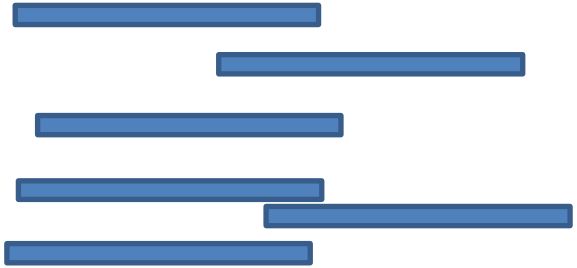
Assembly

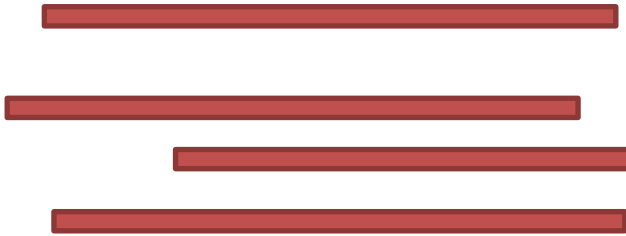
Obviously this a little bit more complicated….

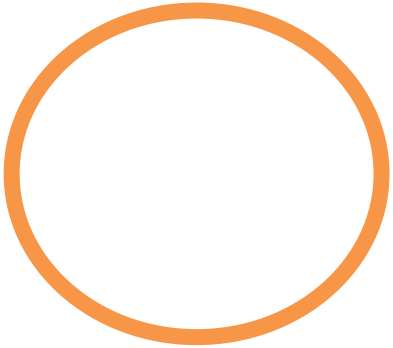
Actually we don’t even have the box image most of the time

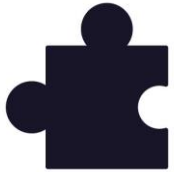
Assembly

Reads → Contigs → Genome









Assembly

Reconstruct a representation of the original DNA from shorter DNA sequences or small fragments known as reads

- ***De novo:*** with no previous knowledge of the genome to be assembled. It overlap the end of the end of each read in order to créate a longer sequence.

- ***Assembly with reference:*** A similar but not identical genome guides the assembly process. Map reads over supplied genome.

Assembly: contig y scaffold

- **Contig:** continuous sequence made up of overlaping shorter sequences
- **Scaffold:** two or more contigs located and rearranged according to spatial information(pair-end, mate pair, reference)

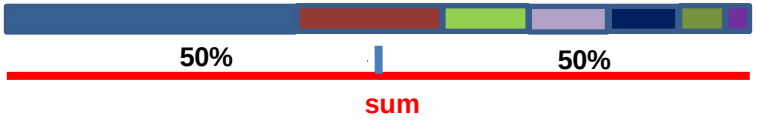
https://www.biostars.org/p/253222/

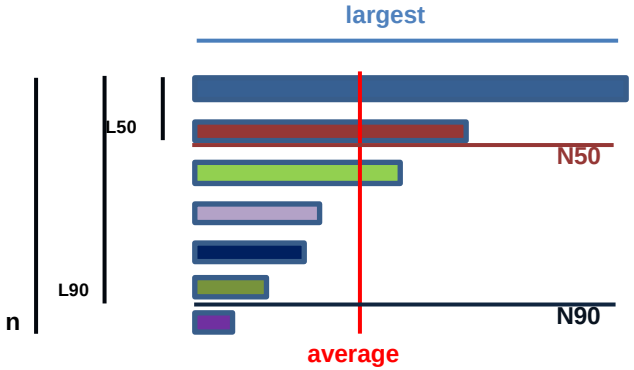
Assembly polishing and quality control

- Abacas sort contigs according to reference genome.
- Scaffolding
- Visualization

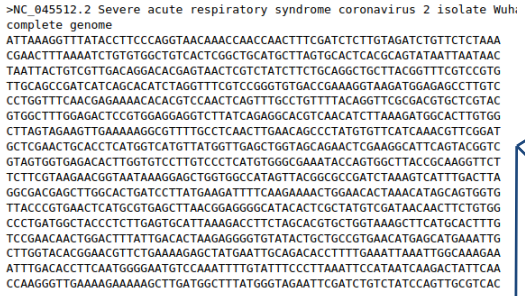
Assembly polishing and quality control

- sum = total bases number
- n = contigs number
- average = average contig length
- largest = largest contig
- N50 = length of the shortest contig where 50% of sum is held
- L50 = number of contigs which have 50% of the genome
- N90 = length of the shortest contig where 90% of sum is held.
- L90 = number of contigs which have 90% of the genome





And...after all this?



.fasta consensus genome

DB

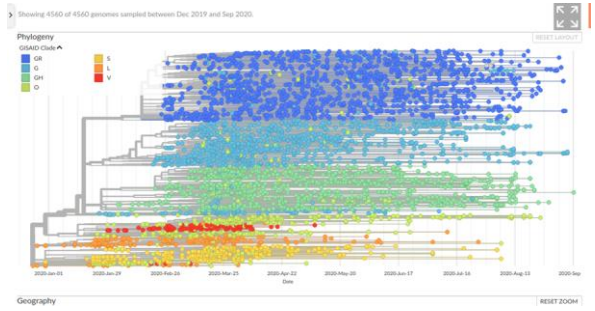
ENA

European Nucleotide Archive

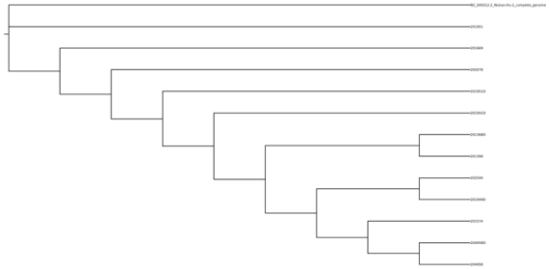
GISAID

2008 - 2020

Clade classification

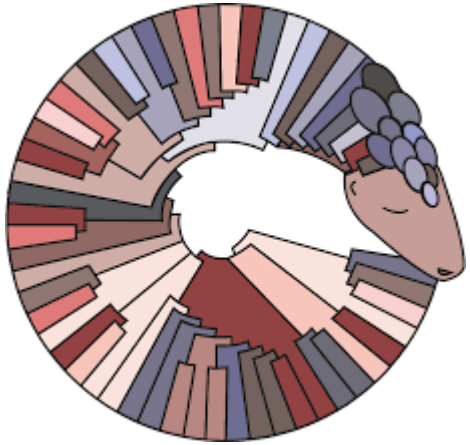


Outbreak detection



Other analyses

Linage assignment: Pangolin



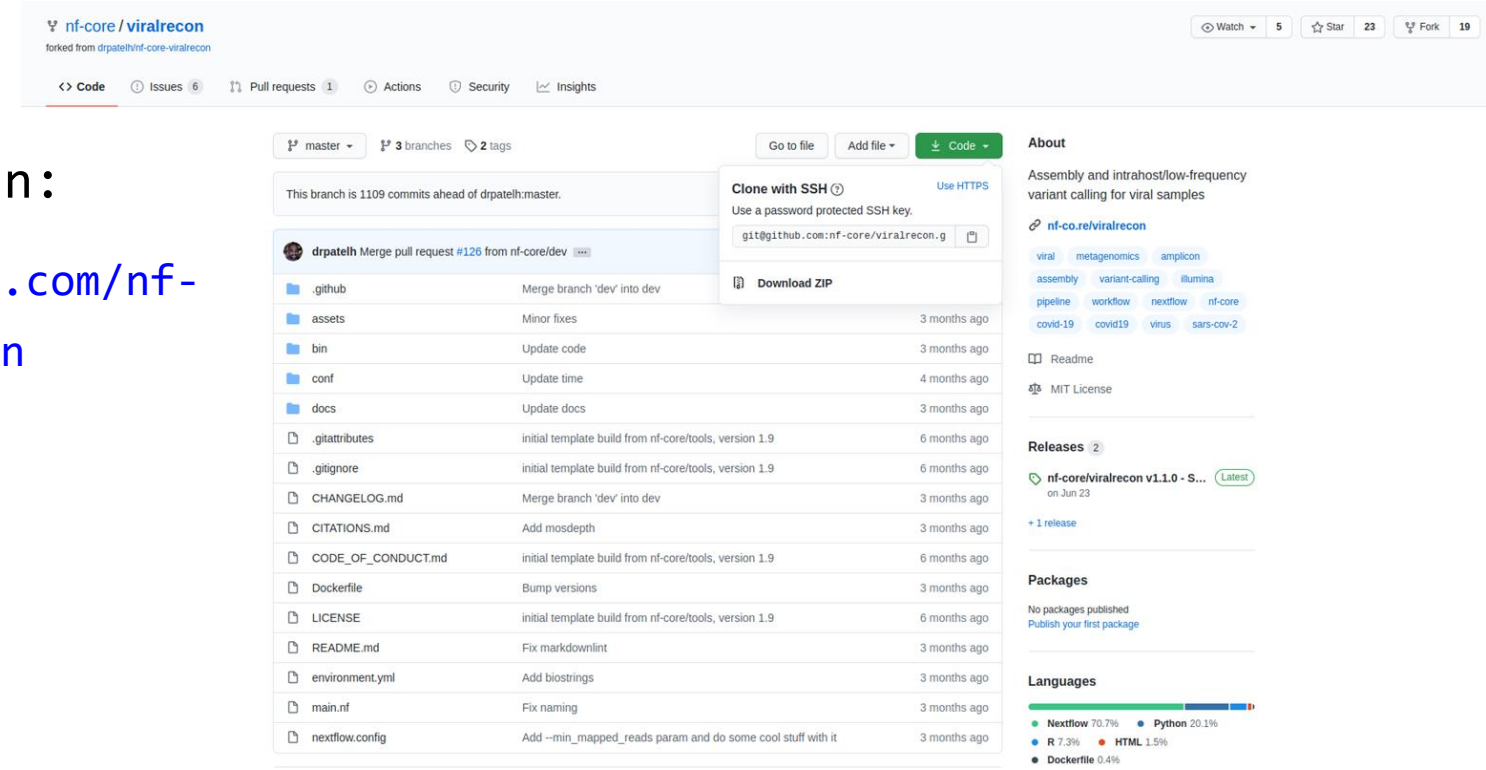
pangolin assigns lineages to query sequences as described in Rambaut et al 2020.

https://cov-lineages.org/

Conclusions

You can find viralrecon in:

https://github.com/nf-core/viralrecon



Thanks for your attention!

(Find us in https://github.com/BU-ISCIII)

Special mention to the nf-core community (https://nf-co.re/)