

Input data for the exercise

On your laptop, in the Ubuntu virtual environment, navigate to the **Covid_alignment** directory you downloaded in the previous session. This directory contains the input data for this exercise.

```
cd /home/yourname/Documents/Covid_alignment
```

In this command, please replace "yourname" with the username of your Ubuntu virtual environment.

We take a closer look at what is in the directory by listing all the files in the directory:

```
ls -l
```

Next, we take a closer look at each of the files in the directory:

The reference genome .fasta file

One of the files in the **Covid_alignment** directory is the following:

```
MN908947.3_reference_genome.fasta
```

This is the reference genome file.

It is in a standard format for biological (DNA, RNA, protein...) sequences, the .fasta format, which is also indicated by the extension of the file.

.fasta files are plain text files, but with a structured contents.

Usually, as in this case, the reference genome is not generated as part of your sequencing experiment, but has been generated previously and is provided from a public resource, in this case GenBank:

<https://www.ncbi.nlm.nih.gov/nuccore/MN908947>

Reference genomes for higher organism can be very large files. Here, we have the reference genome for a virus. What is the size of this .fasta file? To this purpose, we use `ls` with an extra option to indicate the file size in KB, MB, GB ... :

```
ls -lh MN908947.3_reference_genome.fasta
```

Can we see from the command line what is inside this file?

We can use e.g. the `cat` command to display the entire file on the screen:

```
cat MN908947.3_reference_genome.fasta
```

We can also display only the first 10 lines of the file on the screen, using the `head` command:

```
head MN908947.3_reference_genome.fasta
```

```
>MN908947.3 Severe acute respiratory syndrome coronavirus 2 isolate Wuhan-Hu-1, complete genome
ATTAAAGGTTTATACCTTCCCAGGTAACAAACCAACCAACTTTCGATCTCTTGATAGATCTGTTCTCTA
AA
CGAACTTTAAAATCTGTGTGGCTGTCACTCGGCTGCATGCTTAGTGCACTCACGCAGTATAATTAATA
AC
```

```
TAATTACTGTCGTTGACAGGACACGAGTAACTCGTCTATCTTCTGCAGGCTGCTTACGGTTTCGTCCG
TG
TTGCAGCCGATCATCAGCACATCTAGGTTTCGTCCGGGTGTGACCGAAAGGTAAGATGGAGAGCCTT
GTC
CCTGGTTTCAACGAGAAAACACACGTCCAACCTCAGTTTGCCTGTTTTACAGGTTTCGCGACGTGCTCG
TAC
GTGGCTTTGGAGACTCCGTGGAGGAGGTCTTATCAGAGGCACGTCAACATCTTAAAGATGGCACTTG
TGG
CTTAGTAGAAGTTGAAAAAGGCGTTTTGCCTCAACTTGAACAGCCCTATGTGTTTCATCAAACGTTCC
GAT
GCTCGAACTGCACCTCATGGTCATGTTATGGTTGAGCTGGTAGCAGAACTCGAAGGCATTTCAGTACG
GTC
GTAGTGGTGAGACACTTGGTGTCTTGTCCCTCATGTGGGCGAAATACCAGTGGCTTACCGCAAGGT
TCT
```

How many named sequences does the file contain? To find out, we "pipe" the output of the `cat` command to two other commands, `grep` (for searching patterns in the text file) and `wc` (for counting characters, words and lines in the text file):

```
cat MN908947.3_reference_genome.fasta | grep '^>' | wc -l
```

How many DNA bases does the file contain? We use the same three commands, but with different options:

```
cat MN908947.3_reference_genome.fasta | grep -v '^>' | wc -c
```

The .fasta.fai index file

An index file usually "belongs" to a larger file with which it often shares the base name. The .fasta.fai index file

```
MN908947.3_reference_genome.fasta.fai
```

belongs to the .fasta reference file

```
MN908947.3_reference_genome.fasta
```

Similar to a "Contents" page in a book, an index file is a file that facilitates direct access to specific parts of the large file it belongs to. In the case of reference genome .fasta files, the index file contains information on the sequences contained in this file.

We can look at a .fasta.fai file also with `cat`:

```
cat MN908947.3_reference_genome.fasta.fai
```

Since our .fasta reference file only contains one sequence, the .fasta.fai index file only contains one line:

```
MN908947.3 29903 96 70 71
```

The .fastq sequencing data file

The third relevant file in the **Covid_alignment** directory contains the data from an Oxford Nanopore Technologies sequencing experiment for one Covid sample:

```
ls -l barcode02.fastq
```

Raw data from long or short read Next Generation Sequencing experiments are usually stored in .fastq format, another standard format related to the .fasta format, but with

additional information inside.

Also .fastq files are text files. We can look at them e.g with the cat or head or less commands.

One entry in a .fastq file corresponding to one 1 sequenced read is represented in 4 consecutive lines, e.g.:

```
head -4 barcode02.fastq
```

The first line of the entry contains the read ID and some read- and sequencing run-related metadata:

```
@54477693-e552-4d0b-b5a6-b4e3785ee400
runid=062476b43ac9987b98eb5d86472c7fb0786afcbd read=21 ch=199
start_time=2021-02-25T08:53:06Z flow_cell_id=FAP37922
protocol_group_id=Sars_Cov2_WGS_1200_Run2 sample_id=no_sample
barcode=barcode02 barcode_alias=barcode02
```

The second line of the entry contains the actual sequence that was read from the biological sample:

```
GGTGTGCTTCGTTTCGAGATCAATTTGGGTGTTTAACTCGATTCCGTTTGTAGTCGTCTGTGTTTTTCGC
ATTTTATCGTGAAACGCTTTTCGCGTTTTTCATTTATCAGCTTCATTAACAATTTTACCACCCTTAAGTG
CTATCTTTGTTGTTACAACATTAACAACCTTGTCTGAAGTAGTTGCACATGTGTCAACTTAAAGGTAAGT
TATTCTTTTAGCAGCACTACGTATTTGTTTTTCGTAGTTGTTTCAGACAATGACATGAAATCTAACGTTCC
ATAATAAAGAAGCAATGTTTGTGACTTTTTGCTACCTGCGCATTAAATATGACGCGCACTACAGTCAATA
CAAGCACCAAGGTCACGGGGTGTTCATGTTTTCCAACCTTGTATAGGTGAACATATAGTTATTACAACT
ATC
```

The third line, by convention, only contains a "+" in most data sets:

```
+
```

The fourth line contains an ASCII character indicating the quality for each nucleotide in the sequence in the second line, according to the Phred score convention:

```
&$/'-%)-/, '&;) (#&005, ., 02261G?AA>45D?ACJ@@AF=76>40=:><;??>3?
AIKFB=HBBFKD?I2.0400- ./1@GB8@><C@?<8) ).&#)&)%*4;@118./.(%) :822, --1/
%+, *8115?A?;AC:C@-%)/;589&&)./) -...2AIB/-+'1231,:95/1/7.%?
E998:<JM&5978:;>A56:0./102684;=;;BF/,C//0B?
@CDC<=>CGC@ID1*+06BC<DG0)*.+20-'%46?>C@D>:=216, ./&--
$- '(##'5849ABAIGGFEB9399:2>5E?ADA;?97@2=*<3910:58;?
62854,29;9;:>;577DC;65+.>9>6DE>A5>CD?>=+-115IBE1, ,0+)/2-%
%1,&$#&$,)*98@(+8417%
```

When looking at more than one entry from this nanopore sequencing experiment, what do we notice:

```
less barcode02.fastq
```

Oxford Nanopore Technologies offers long read sequencing; one characteristic of this is that each read has a different length, ranging from several 100s to several 1000s of nucleotides.

How many reads are in this .fastq file? For this, we open the file with cat, select a pattern present in the first line of all entries and count the number of occurrences in lines:

```
cat barcode02.fastq | grep 'barcode_alias=barcode02' | wc -l
```

Alignment of the sequencing data to the

reference genome

One of the objectives of sequencing a biological sample is to compare the data of the sample to a reference genome of the same species, e.g. to determine variants, population diversity or mutations.

The first step to perform this comparison of the data of the sample to the reference genome is the so-called alignment (or mapping) of the sample sequences to the reference genome sequence.

Genomic alignment tools constitute an important category of bioinformatic tools.

Install minimap2 aligner

minimap2 aligner is an open source software package. The code for minimap2 aligner is available on github: <https://github.com/lh3/minimap2>.

However, minimap2 can also be installed directly from the Ubuntu code repository:

```
sudo apt install minimap2
```

This command can be executed from any directory in your Ubuntu virtual environment. The sudo command will require that you introduce your user password before the installation command apt install is executed.

Check minimap2 installation

Once the installation command has completed and the command line prompt is ready again, we check that the installation was successful by calling minimap2 without any arguments from any directory:

```
minimap2
```

This displays details on the usage of minimap2 on the screen:

```
Usage: minimap2 [options] <target.fa>|<target.idx> [query.fa] [...]
```

Use minimap2 to align nanopore reads to the reference genome

In order to run the alignment of the reads of our sample to the reference genome, we need to configure the alignment command for minimap2 first.

To this purpose, we consult the user manual on the github page of minimap2 at <https://github.com/lh3/minimap2>.

For the alignment of Oxford Nanopore genomic reads, the command recommended in the user manual is the following:

```
minimap2 -ax map-ont ref.fa ont.fq.gz > aln.sam
```

This is a slightly compressed way of saying:

```
minimap2 -a -x map-ont ref.fa ont.fq.gz > aln.sam
```

How is this command composed and what do the different parts of it mean?

minimap2 calls the minimap2 package

-a is a flag that specifies that the output should be in SAM format.

-x map-ont is an argument that tells minimap2 to use a "preset" for alignment of Oxford Nanopore reads.

ref . fa indicates the file name for the .fasta file of the reference genome.

ont . fq . gz indicates the file name for the .fastq file containing the sequencing data of our sample.

> aln . sam indicates that the output produced by the command should be redirected to a file called aln . sam .

When executing this command from any directory you are in, what happens?

Have any new files been generated in this directory upon the execution? We can check for that listing all files in the directory and sorting by time of modification:

```
ls -lt
```

We try to execute the same command again from the **Covid_alignment** directory.

We navigate into the directory:

```
cd /home/yourname/Documents/Covid_alignment
```

and execute the same command:

```
minimap2 -ax map-ont ref . fa ont . fq . gz > aln . sam
```

What happens? How do we have to modify the command so that it finds all required input files? We use the correct file names for the reference genome .fasta file and for the .fastq sequencing data file. We also give a meaningful name to the output file, so we can trace it back to the sample we are analysing:

```
minimap2 -ax map-ont MN908947.3_reference_genome.fasta  
barcode02.fastq > barcode02.sam
```

Results of the minimap2 alignment command

Which new files have been generated in this directory upon the execution? How big are they? We can check for that listing all files in the directory, sorting by time of modification and displaying the file size in human-readable format:

```
ls -lth
```

The output file from the alignment command, barcode02 . sam is in yet another standard genomics file format, the .sam format.

.sam files are plain text files and can be viewed e.g. with less:

```
less barcode02.sam
```

Files in .sam format have a header with metadata:

```
@SQ SN:MN908947.3 LN:29903@PG ID:minimap2 PN:minimap2 VN:2.17-r941  
CL:minimap2 -ax map-ont MN908947.3_reference_genome.fasta  
barcode02.fastq
```

followed by the alignment entries:

```
54477693-e552-4d0b-b5a6-b4e3785ee400 16 MN908947.3 8267 60  
39M1I81M1I9M2I4M1I11M2D52M1D16M1D9M2I15M2I60M112S * 0 0  
GATAGTTGTAATAACTATATGTTACCTATAACAAAGTTGGAAAACATGACACCCCGTGACCTTGGTGC
```

```

TTGTATTGACTGTAGTGCGCGTCATATTAATGCGCAGGTAGCAAAAAGTCACAAACATTGCTTCTTTAT
TATGGAACGTTAGATTTTCATGTCATTGTCTGAACAACACTACGAAAACAAATACGTAGTGCTGCTAAAAGA
ATAACTTACCTTTAAGTTGACACATGTGCAACTACTTCAGACAAGTTGTTAATGTTGTAACAACAAAGA
TAGCACTTAAGGGTGGTAAATTGTTAATGAAGCTGATAAATGAAAAACGCGAAAGCGTTTCACGATAA
AATGCGAAAACACAGACGACTACAAACGGAATCGAGGTTAAACACCCAAATTGATCTCGAACGAAGCAC
ACC %7148+(@89*), $&#&$, 1%%-2/)+0, , 1EBI511-+=>?DC>5A>ED6>9>.
+56;CD775;>;9;92,45826?;85:0193<*=2@79?;ADA?
E5>2:9939BAEFGGIABA9485'##('-$- &/., 612=:=>D@C>?
64%' -02+.* )0GD<CB60+*1DI@CGC>==<CDC@?B0//C,/FB;;=;486201/.0:65A>;:
8795&MJ<:899E?%%.7/1/59:,1321'+-/BIA2...-)/.)&&985;/)%-@C:CA;?A?
5118*,++%/1-- ,228:)%(%./ .811@;4*%)&#&.) )8<?@C<>@8BG@1//.-0040.2I?
DKFBBH=BFKIA?3>???;<>:=04>67=FA@JCA?D54>AA?G16220,..,500&#(;&' /- )
%- '/$& NM:i:15 ms:i:518 AS:i:518 nn:i:0 tp:A:P cm:i:31 s1:i:205 s2:i:
0 de:f:0.0361 rl:i:0

```

Like other genomic data files, .sam alignment files can get quite big. Therefore, plain text .sam files are usually stored as binary .bam files. .sam files can be compressed into .bam files using the software samtools.

Install samtools

samtools is an open source software package. The code for samtools is available on github as part of the htstlib package: <http://www.htstlib.org/>.

However, like minimap2, samtools can also be installed directly from the Ubuntu code repository:

```
sudo apt install samtools
```

This command can be executed from any directory in your Ubuntu virtual environment. The sudo command will require that you introduce your user password before the installation command apt install is executed.

Check samtools installation

Once the installation command has completed and the command line prompt is ready again, we check that the installation was successful by calling samtools without any arguments from any directory:

```
samtools
```

This displays details on the usage of samtools on the screen:

```
Usage: samtools <command> [options]
```

Use samtools to compress .sam to .bam alignment files

In order to compress the .sam file to a .bam file, we use the samtools function samtools view. We check the detailed options:

```
samtools view
```

```
Usage: samtools view [options] <in.bam>|<in.sam>|<in.cram> [region ...]
```

We configure the command as follows:

```
samtools view -h -B barcode02.sam > barcode02.bam
```

How is this command composed and what do the different parts of it mean?

`samtools view` is the main `samtools` function we are calling.

`-h` indicates that the header lines will be included in the output.

`-B` indicates that the output will be in `.bam` format.

`barcode02.sam` is the input file.

`> barcode02.bam` indicates the name of a (new) output file in `.bam` format that we want to generate.

Which new files have been generated in this directory upon the execution? How big are they? We can check for that listing all files in the directory, sorting by time of modification and displaying the file size in human-readable format:

```
ls -lth
```

How big is the difference in file size between the `.sam` and the `.bam` files?

The `.bam` file is a binary file, hence it cannot be viewed with commands like `head`, `cat`, or `less`.

When we try nevertheless, what happens?

```
less barcode02.bam
```

The correct way to look at the contents of the `.bam` file is as follows:

```
samtools view -h barcode02.bam | less
```

Use `samtools` to sort and index `.bam` alignment files

Similar to the reference genome `.fasta` files, also `.bam` alignment files can be indexed which is a requirement for many downstream tools working with `.bam` files as an input.

We try to index the `barcode02.sam` file using another function of `samtools`:

```
samtools index barcode02.bam
```

What happens?

In order to index the file, we need to sort first the alignment entries in the file by genomic coordinate. For this, we use:

```
samtools sort barcode02.bam > barcode02_sorted.bam
```

This command reads the unsorted `barcode02.bam` file and outputs another, coordinate-sorted `.bam` file with a new name that we choose, `barcode02_sorted.bam`.

We look again at the files in the directory:

```
ls -lth
```

How big is the difference in file size between the `barcode02.bam` and the `barcode02_sorted.bam` files? How can this difference be explained?

We inspect the order of the entries between the `barcode02.bam` file and the `barcode02_sorted.bam` files:

```
samtools view barcode02.bam | less
```

```
samtools view barcode02_sorted.bam | less
```

We look into the barcode02_sorted.bam file, displaying also the header lines:

```
samtools view -h barcode02_sorted.bam | less
```

Some information has been added to the header of the file through the last steps of the analysis:

```
@HD VN:1.6 S0:coordinate@SQ SN:MN908947.3 LN:29903@PG ID:minimap2
PN:minimap2 VN:2.17-r941 CL:minimap2 -ax map-ont
MN908947.3_reference_genome.fasta barcode02.fastq@PG ID:samtools
PN:samtools PP:minimap2 VN:1.10 CL:samtools view -h -b barcode02.sam
@PG ID:samtools.1 PN:samtools PP:samtools VN:1.10 CL:samtools sort
barcode02.bam@PG ID:samtools.2 PN:samtools PP:samtools.1 VN:1.10
CL:samtools view -h barcode02_sorted.bam
```

Now, we can proceed to indexing the sorted .bam file:

```
samtools index barcode02_sorted.bam
```

Which new files have been created after this command?

Next steps

The sorted and indexed .bam file can be used as input for the next steps of genome analysis, such as genomic variant analysis using suitable so-called variant calling tools and generating variant call files in .vcf format.