Contents

1. Introduction to Linux on HPC 1
   1.1 Getting Started 1
   1.2 The prompt 1
   1.3 Basic Commands 1
   1.4 Manipulating files and directories 3
   1.5 Software packages on HPC 9
2. Nanopore data 11
   2.1 Data structure 11
   2.2 Practice with test data 13
3. EPI2ME: wf-amplicon 16
   3.1 Set HPC environment 17
   3.2 Run Test - Consensus mode 17
   3.3 Run Test - Variant mode 20
   3.4 Run with real data 21

Welcome to the HPC filesystem tutorial. This guide will teach you all the necessary commands and provide you with info on how to work in linux within the context of an HPC environment.

For an extended overview of HPC-documentation I refer to the HPC-DOCS:
https://hpcugent.github.io/vsc_user_docs/

1. Introduction to Linux on HPC

1.1 Getting Started

To get started with the HPC-UGent infrastructure, you need to obtain a VSC account, see HPC manual. Keep in mind that you must keep your private key to yourself! You can look at your public/private key pair as a lock and a key: you give us the lock (your public key), we put it on the door, and then you can use your key to open the door and get access to the HPC infrastructure. Anyone who has your key can use your VSC account! Details on connecting to the HPC infrastructure are available in HPC manual connecting section.

NOTE: If you plan to work in the webbrowser interface only, then you don't need to upload public/private key. For this course, the webbrowse interface only is ok.

1.2 The prompt

The basic interface is the so-called shell prompt, typically ending with `$` (for bash shells).

NOTE: In the `code` examples below the `$` indicates your prompt. This is for information purpose only, you don't need to type this in your commands.

You use the shell by executing commands, and hitting `<enter>`. For example:

```
$ echo hello
hello
```

While typing (long) commands in the terminal, you can go to the start or end of the command line using `Ctrl-A` or `Ctrl-E`.

To go through previous commands, use `<up>` and `<down>`, rather than retyping them.

1.3 Basic Commands

Download example data
For this exercise you'll need to connect to the HPC and open a terminal. I've prepared some folders and files on which you can perform following commands.

Start by copying the `/Linux_Basics_exampledata` to your home folder `~/`.

```
$ cp -r /path/to/ONT-SEQ_PA-Benin2024_Bioinfocourse/Linux_Basics_exampledata ~/
```

Commands & Examples
NOTE: Make sure you are in your home folder before you start with the exercises.
To go to your home folder use following command:

```
$ cd ~/
```

Example 1: Lists files and directories in the current directory.
`$ ls -l`

Example 2: Changes the current directory.
`$ cd /user/gent/433/vscXXXX/ONT-SEQ_PA-Benin2024_Bioinfocourse`

Example 3: Prints the current working directory.
`$ pwd`

Example 4: Creates a new directory.
`$ mkdir my_first-folder`

Example 5: Deletes a file. WARNING: There is no trash, files are permanently deleted!
`$ rm ./old_hello.py`

Example 6: Deletes a directory and its contents recursively.
`$ rm -r ./old_sequences`

Example 7: Copies files or directories.
`$ cp samplesheet96.txt ./my_first-folder`

Example 8: Moves or renames files or directories.
```
$ mv ./my_first-folder/samplesheet96.txt ./my_first-folder/samplesheet96_$(date +%Y-%m-
%d).txt
```

Example 9: Searches for a file by name in a specified path.
`$ find ~/ -name "samplesheet96*"`

Example 10: Displays the contents of a file.
`$ cat hello.py`

Example 11: Opens a file for viewing one page at a time. Press q to quit.
`$ less manual_cp.txt`

Example 12: Shows the first 10 lines of a file.
`$ head manual_cp.txt`

Example 13: Shows the last 10 lines of a file.
`$ tail manual_cp.txt`

Example 14: Displays detailed information about files, including permissions.
`$ ls -lh ~/Linux_Basics_exampledata`

Example 15: Shows currently running processes.
`$ ps`

Example 16: Displays real-time system resource usage.
`$ top`

Example 17: Interactive process viewer (if installed).
`$ htop`

Example 18: Prints text to the terminal.
`$ echo "Hello, world!"`

Example 19: Displays the current date and time.
```
$ date
```

Example 20: Logs out or closes the terminal.
```
$ exit
```

1.4 Manipulating files and directories

Changing permissions: “chmod”
Each file and directory has particular permissions set on it, which can be queried using ls -l.

For example:

```
$ ls -l afile.txt
-rw-rw-r-- 1 vsc40000 agroup 2929176 Apr 12 13:29 afile.txt
```

The -rwxrw-r– specifies both the type of file (- for files, d for directories (see first character)), and the permissions for user/group/others:

1. each triple of characters indicates whether the read (r), write (w), execute (x) permission bits are set or not
2. the 1st part rwx indicates that the owner “vsc40000” of the file has all the rights
3. the 2nd part rw- indicates the members of the group “agroup” only have read/write permissions (not execute)
4. the 3rd part r– indicates that other users only have read permissions

The default permission settings for new files/directories are determined by the so-called umask setting, and are by default:

1. read-write permission on files for user/group (no execute), read-only for others (no write/execute)
2. read-write-execute permission for directories on user/group, read/execute-only for others (no write)

Any time you run `ls -l` you’ll see a familiar line of `-rwx------` or similar combination of the letters `r`, `w`, `x` and `-` (dashes). These are the permissions for the file or directory.

```
$ ls -l
total 1
-rw-r--r--. 1 vsc40000 mygroup 4283648 Apr 12 15:13 articleTable.csv
drwxr-x---. 2 vsc40000 mygroup 40 Apr 12 15:00 Project_GoldenDragon
```

> Here, we see that `articleTable.csv` is a file (beginning the line with -) has read and write permission for the user `vsc40000` (`rw-`), and read permission for the group `mygroup` as well as all other users (`r--` and `r--`).

The next entry is `Project_GoldenDragon`. We see it is a directory because the line begins with a `d`. It also has read, write, and execute permission for the `vsc40000` user (`rwx`). So that user can look into the directory and add or remove files. Users in the `mygroup` can also look into the directory and read the files. But they can’t add or remove files (`r-x`). Finally, other users can read files in the directory, but other users have no permissions to look in the directory at all (`---`).

Maybe we have a colleague who wants to be able to add files to the directory. We use `chmod` to change the modifiers to the directory to let people in the group write to the directory:

```
$ chmod g+w Project_GoldenDragon
$ ls -l
total 1
-rw-r--r--. 1 vsc40000 mygroup 4283648 Apr 12 15:13 articleTable.csv
drwxrwx---. 2 vsc40000 mygroup 40 Apr 12 15:00 Project_GoldenDragon
```

> The syntax used here is `g+x` which means group was given write permission. To revoke it again, we use `g-w`. The other roles are `u` for user and `o` for other.

You can put multiple changes on the same line: `chmod o-rwx,g-rxw,u+rx,u-w somefile`.
This will take everyone's permission away except the user's ability to read or execute the file.

You can also use the `-R` flag to affect all the files within a directory, but this is dangerous. It's best to refine your search using find and then pass the resulting list to chmod since it's not usual for all files in a directory structure to have the same permissions.

Example 1: Apply chmod to all files (regular files only) within a directory
This will set 644 permissions for all files (not directories) `find /path/to/directory -type f -exec chmod 644 {} \;`

Example 2: Apply chmod to all directories within a directory
This will set 755 permissions for all directories (not files) `find /path/to/directory -type d -exec chmod 755 {} \;`

Example 3: Apply chmod to all files modified in the last 7 days
This will search for all files modified in the last 7 days and set 644 permissions `find /path/to/directory -type f -mtime -7 -exec chmod 644 {} \;`

Example 4: Apply chmod to files with a specific extension (e.g., .txt)
This will search for all .txt files and set 644 permissions `find /path/to/directory -type f -name "*.txt" -exec chmod 644 {} \;`

Example 5: Apply chmod recursively with different permissions for files and directories
First, set 644 permissions for all files (regular files) `find /path/to/directory -type f -exec chmod 644 {} \;`

Then, set 755 permissions for all directories `find /path/to/directory -type d -exec chmod 755 {} \;`

1.4.1.1 Exercises: Changing permissions with "chmod"

Here are some exercises on `chmod` based on your directory structure. Try each command and observe the changes using `ls -l` before and after.

Exercise 1: Make `hello.sh` Executable
Goal: Grant execute (x) permission to the script so it can run as a program.

```
chmod +x hello.sh
ls -l hello.sh
```

Check if the x permission appears for the user (`rwxr--r--`). Now try running it:

```
./hello.sh
```

Exercise 2: Remove Read Permissions from `manual_cp.txt`
Goal: Prevent yourself and others from reading the file.

```
chmod a-r manual_cp.txt
ls -l manual_cp.txt
```

Now try opening it:

```
cat manual_cp.txt  # Should give a "Permission denied" error
```

Restore permissions after testing:

```
chmod u+r manual_cp.txt
```

Exercise 3: Grant Full Access to `samplesheet96.txt` for Everyone
Goal: Allow all users to read, write, and execute `samplesheet96.txt`.

```
chmod 777 samplesheet96.txt
ls -l samplesheet96.txt
```

*Now everyone can modify and execute the file. This is **not recommended** for sensitive files!*

Exercise 4: Restrict `old_hello.py` to Read and Write for Owner Only
Goal: Make the file private so only the owner can read and modify it.

```
chmod 600 old_hello.py
ls -l old_hello.py
```

Now other users cannot read or modify it.

Exercise 5: Add a Sticky Bit to `sequences`
Goal: Ensure that only the file owner can delete their own files inside `sequences`, even if others have write access.

```
chmod +t sequences
ls -ld sequences  # Check for the "t" at the end of the permissions (drwxr-xr-t)
```

Now, even if other users have write permissions, they cannot delete files they don't own.

Exercise 6: Recursively Set Read & Execute for All Users in `old_sequences`
Goal: Ensure all files in `old_sequences` are readable and executable but not writable by others.

```
chmod -R a+rx old_sequences
ls -l old_sequences/
```

Check if all files inside now have `r-x` for everyone.

Exercise 7: Set Group Write Permissions for `Tabular-file.tab`
Goal: Allow your group members to edit the file.

```
chmod g+w Tabular-file.tab
ls -l Tabular-file.tab
```

Now, users in the same group (`vsc43352`) can modify the file.

Exercise 8: Remove Execute Permission from `hello.py`
Goal: Prevent accidental execution of a Python script.

```
chmod -x hello.py
ls -l hello.py
```

Now, you must explicitly use `python3 hello.py` instead of `./hello.py`.

Exercise 9: Convert Between Symbolic and Octal Modes

Goal: Change `samplesheet96_2025-04-02.txt` to the same permissions as `samplesheet96.txt` using octal mode.

1. **Check the original permissions:**

   ```
   ls -l samplesheet96.txt
   ```

2. **Use `stat` to see the octal mode:**

   ```
   stat -c "%a" samplesheet96.txt
   ```

   Suppose it returns 644.

3. **Apply the same mode to the new file:**

   ```
   chmod 644 samplesheet96_2025-04-02.txt
   ```

Bonus Challenge: Set sequences to 750
Goal: Make sequences accessible only to the owner and group, while others have no access.

```
chmod 750 sequences
ls -ld sequences
```

Now only you and your group can access it, while others are blocked.

Final Check After completing the exercises, list all files again to see the changes:

```
ls -l
```

Compressing files (zip, unzip, tar)

Files should usually be stored in a compressed file if they're not being used frequently. This means they will use less space and thus you get more out of your quota. Some types of files (e.g., CSV files with a lot of numbers) compress as much as 9:1. The most commonly used compression format on Linux is gzip. To compress a file using gzip, we use:

```
$ ls -lh myfile
-rw-r--r--. 1 vsc40000 vsc40000 4.1M Dec 2 11:14 myfile
$ gzip myfile
$ ls -lh myfile.gz
-rw-r--r--. 1 vsc40000 vsc40000 1.1M Dec 2 11:14 myfile.gz
```

Note: if you zip a file, the original file will be removed. If you unzip a file, the compressed file will be removed. To keep both, we send the data to stdout and redirect it to the target file:

```
$ gzip -c myfile > myfile.gz
$ gunzip -c myfile.gz > myfile
```

"zip" and "unzip"

Windows and macOS seem to favour the zip file format, so it's also important to know how to unpack those. We do this using unzip:

```
$ unzip myfile.zip
```

If we would like to make our own zip archive, we use zip:

```
$ zip myfiles.zip myfile1 myfile2 myfile3
```

Working with tarballs: “tar”

Tar stands for “tape archive” and is a way to bundle files together in a bigger file.

You will normally want to unpack these files more often than you make them. To unpack a `.tar` file you use:

```
$ tar -xf tarfile.tar
```

Often, you will find `gzip` compressed `.tar` files on the web. These are called tarballs. You can recognize them by the filename ending in `.tar.gz`. You can uncompress these using gunzip and then unpacking them using tar. But tar knows how to open them using the `-z` option:

```
$ tar -zxf tarfile.tar.gz
$ tar -zxf tarfile.tgz
```

Order of arguments

Note: Archive programs like `zip`, `tar` use arguments in the “opposite direction” of copy commands.

```
$ cp source1 source2 source3 target
$ zip zipfile.zip source1 source2 source3
$ tar -cf tarfile.tar source1 source2 source3
```

If you use `tar` with the source files first then the first file will be overwritten. You can control the order of arguments of tar if it helps you remember:

$ tar -c source1 source2 source3 -f tarfile.tar Exercises: Compressing files (zip, unzip, tar)

Here are some **zip, unzip, and tar** exercises based on your directory structure. Each exercise includes commands and explanations.

Exercise 1: Create a `.zip`file from a local file.
Goal: Create a `.zip` archive containing `manual_cp.txt`.

```
zip manual_cp.zip manual_cp.txt
ls -l manual_cp.zip
```

Now the file is compressed into `manual_cp.zip`.

Exercise 2: Extract `manual_cp.zip`
Goal: Extract the zipped file back to its original form.

```
unzip manual_cp.zip
ls -l
```

The extracted file should appear in the directory.

Exercise 3: Compress Multiple Files into One ZIP Archive
Goal: Create a `.zip` file containing `hello.py`, `hello.sh`, and `Tabular-file.tab`.

```
zip my_archive.zip hello.py hello.sh Tabular-file.tab
ls -l my_archive.zip
```

All three files are stored in `my_archive.zip`.

Exercise 4: Compress the `sequences` Directory Using ZIP
Goal: Create a `.zip` archive of the `sequences` directory.

```
zip -r sequences.zip sequences
ls -l sequences.zip
```

The `-r` flag ensures that all files inside the directory are included.

Exercise 5: Extract `sequences.zip`
Goal: Extract the entire directory from the `.zip` archive.

```
unzip sequences.zip
ls -l
```

The `sequences` directory is restored.

Exercise 6: Create a Tarball (`.tar`) Without Compression
Goal: Archive multiple files into a `.tar` file without compression.

```
tar -cf archive.tar hello.py hello.sh Tabular-file.tab
ls -l archive.tar
```

*The `.tar` file now contains all three files but is **not compressed**.*

Exercise 7: Create a Gzipped Tarball (`.tar.gz`)
Goal: Archive and compress the `old_sequences` directory.

```
tar -czf old_sequences.tar.gz old_sequences
ls -l old_sequences.tar.gz
```

The `-z` flag enables gzip compression, creating a smaller file.

Exercise 8: Extract a `.tar.gz` File
Goal: Extract the `old_sequences.tar.gz` archive.

```
tar -xzf old_sequences.tar.gz
ls -l
```

The `old_sequences` directory is restored.

Exercise 9: List Contents of a `.tar.gz` File Without Extracting
Goal: View files inside `old_sequences.tar.gz` without extracting.

```
tar -tzf old_sequences.tar.gz
```

This shows all files inside the tarball.

Exercise 10: Extract Only `samplesheet96.txt` From `archive.tar`
Goal: Extract a single file from a tar archive.

```
tar -xf archive.tar samplesheet96.txt
ls -l samplesheet96.txt
```

Only `samplesheet96.txt` is extracted.

Bonus Challenge: Tar a Directory and Extract It to Another Location
Goal: Archive `sequences` and extract it elsewhere.

```
tar -czf sequences.tar.gz sequences
mkdir test_restore
tar -xzf sequences.tar.gz -C test_restore
ls test_restore/
```

The `sequences` directory is extracted into `test_restore` instead of the current directory.

Final Check
After completing the exercises, list your directory contents:

```
ls -l
```

1.5 Software packages on HPC

All software on the HPC is available through `modules` which need to be activated first before you can start using them.
Follow the steps below to learn how to work with these commands.

Exercise 1: Use the command `module` to list all activated modules

1. Open a terminal and an interactive session! (see "02-Day1-HPC-intro_2025.ppx" if you don't remember how to do this)
2. Check which modules are currently loaded/activated on your system. The command ´ml´ you'll see below is short for `module`.

```
ml list
```

```
# OUTPUT
Currently Loaded Modules:
  1) env/vsc/doduo (S)   2) env/slurm/doduo (S)   3) env/software/doduo (S)   4) cluster/doduo (

  Where:
   S:  Module is Sticky, requires --force to unload or purge
```

Note: This tells you that you are working on the doduo cluster and all basic modules are loaded but no software packages are yet activated

Exercise 2: Use the command `ml spider` to see if your favourite software package is available

Run following command to check if a nanpack module is available.

```
ml spider nanopack
```

```
# OUTPUT
--------------------------------------------------------------------------------------------------------
  NanoPack: NanoPack/20230602-foss-2023a
--------------------------------------------------------------------------------------------------------
    Description:
      Collection of long read processing and analysis tools.

    You will need to load all module(s) on any one of the lines below before the "NanoPack/2023
```

```
      env/software/accelgor
      env/software/doduo
      env/software/donphan
      env/software/gallade
      env/software/joltik
      env/software/litleo
      env/software/shinx

    Help:
      Description
      ===========
      Collection of long read processing and analysis tools.


      More information
      ================
       - Homepage: https://daler.github.io/pybedtools
```

Note: Here you see the full name of the package "NanoPack/20230602-foss-2023a" and all the cluster on which you can activate this module/software package

1. Activate the nanopack module

```
ml NanoPack/20230602-foss-2023a
```

2. Check if the module is loaded

```
vsc43352@gligar09:~$ml list
```

```
# OUTPUT
Currently Loaded Modules:
  1) env/vsc/doduo                   (S)   26) foss/2023a                                 51) st
  2) env/slurm/doduo                 (S)   27) bzip2/1.0.8-GCCcore-12.3.0                 52) gz
  3) env/software/doduo              (S)   28) ncurses/6.4-GCCcore-12.3.0                 53) lz
  4) cluster/doduo                   (S)   29) libreadline/8.2-GCCcore-12.3.0             54) zs
  5) GCCcore/12.3.0                        30) Tcl/8.6.13-GCCcore-12.3.0                  55) IC
  6) zlib/1.2.13-GCCcore-12.3.0            31) SQLite/3.42.0-GCCcore-12.3.0               56) Bo
  7) binutils/2.40-GCCcore-12.3.0          32) libffi/3.4.4-GCCcore-12.3.0                57) sn
  8) GCC/12.3.0                            33) Python/3.11.3-GCCcore-12.3.0               58) Ra
  9) numactl/2.0.16-GCCcore-12.3.0         34) gfbf/2023a                                 59) Ab
 10) XZ/5.4.2-GCCcore-12.3.0               35) cffi/1.15.1-GCCcore-12.3.0                 60) RE
 11) libxml2/2.11.4-GCCcore-12.3.0         36) cryptography/41.0.1-GCCcore-12.3.0         61) ut
 12) libpciaccess/0.17-GCCcore-12.3.0      37) virtualenv/20.23.1-GCCcore-12.3.0          62) Ar
 13) hwloc/2.9.1-GCCcore-12.3.0            38) Python-bundle-PyPI/2023.06-GCCcore-12.3.0  63) Ka
 14) OpenSSL/3                             39) pybind11/2.11.1-GCCcore-12.3.0             64) Na
 15) libevent/2.1.12-GCCcore-12.3.0        40) SciPy-bundle/2023.07-gfbf-2023a            65) nc
 16) UCX/1.14.1-GCCcore-12.3.0             41) nanomath/1.4.0-foss-2023a                  66) pl
 17) PMIx/4.2.4-GCCcore-12.3.0             42) Biopython/1.83-foss-2023a                  67) li
 18) UCC/1.2.0-GCCcore-12.3.0              43) cURL/8.0.1-GCCcore-12.3.0                  68) Br
 19) OpenMPI/4.1.5-GCC-12.3.0              44) Pysam/0.22.0-GCC-12.3.0                    69) fr
 20) OpenBLAS/0.3.23-GCC-12.3.0            45) nanoget/1.19.3-foss-2023a                  70) ex
 21) FlexiBLAS/3.3.1-GCC-12.3.0            46) NanoStat/1.6.0-foss-2023a                  71) ut
 22) FFTW/3.3.10-GCC-12.3.0                47) NanoFilt/2.8.0-foss-2023a                  72) fo
 23) gompi/2023a                           48) minimap2/2.26-GCCcore-12.3.0               73) xo
 24) FFTW.MPI/3.3.10-gompi-2023a          49) NanoLyse/1.2.1-foss-2023a                 74) X1
 25) ScaLAPACK/2.2.0-gompi-2023a-fb       50) plotly.py/5.16.0-GCCcore-12.3.0           75) Tk

  Where:
   S:  Module is Sticky, requires --force to unload or purge
```

Note: Besides the Nanopack/20230602-foss-2023a module you'll see a lot more is added to the list. These are all the dependencies Nanopack needs to be able to function properly.

2. Nanopore data

Remember that each of you extracted DNA and did PCR on his/her own collectd fieldsamples in Benin last year.
You can find a more detailed list with extra information in the teams *General* channel:
`Document > General > Specieslist-overview`

Before we start any Analysis on this data we first need to learn how to interprete the Nanopore sequence output and learn how to use the different tools.
For this, I prepared a test data set (see below) with sequence output from 2 barcodes, i.e. 2 samples.
Once you mastered all the skills you need, we can continue with a more extensive dataset.

To keep the running time of the analysis minimal (more data is longer runtime), I suggest to select **no more than 10 Barcodes** each.
But first let's have a look at the data structure and start with the test data.

2.1 Data structure

But first things first.
Have a look at the content of the input data folder **/ONT-SEQ_PA-Benin2024_Input-Data/**. You should be able to navigate to this folder by now.
Hint: you can use the `ls` command, or `cd` to access the directory and use the command `tree` to create an overview of the content.

Main folder
The structure looks like this:

```
path/to/ONT-SEQ_PA-Benin2024_Input-Data/
.
+-- ONT-SEQ-24_PA-01
|   +-- fastq_pass
|   |   +-- barcode01
|   |   +-- ...
|   |   \-- barcode24
|   +-- reference.fa
|   +-- report_AMF003_20241018_1203_c8d4ac6d.html
|   \-- samplesheet_ONT-SEQ-24_01-withreference.txt
+-- ONT-SEQ-24_PA-02
|   +-- fastq_pass
|   |   +-- barcode01
|   |   +-- ...
|   |   +-- barcode24
|   +-- reference.fa
|   +-- report_axf308_20241022_1153_b65829bf.html
|   \-- samplesheet_ONT-SEQ-24_02-withreference.txt
+-- ONT-SEQ-25_PA-04
```

```
|   +-- fastq_pass
|   |   +-- barcode01
|   |   +-- ...
|   |   +-- barcode96
|   +-- report_FBA60703_20250120_1336_292963a6.html
|   \-- samplesheet_ONT-SEQ-25_04.txt
+-- ONT-SEQ-25_PA-05
|   +-- fastq_pass
|   |   +-- barcode01
|   |   +-- ...
|   |   +-- barcode48
|   +-- report_FBA60703_20250128_1308_e2391328.html
|   \-- samplesheet_ONT-SEQ-25_05.txt
+-- information
\-- scripts
```

fastq-pass folder
Each `barcode` folder contains the reads for this particular barcode in `fastq.gz` format:

```
../fastq_pass/barcode01$tree
.
+-- AMF003_pass_barcode01_c8d4ac6d_b8b1e3e8_0.fastq.gz
+-- AMF003_pass_barcode01_c8d4ac6d_b8b1e3e8_10.fastq.gz
+-- AMF003_pass_barcode01_c8d4ac6d_b8b1e3e8_11.fastq.gz
+-- AMF003_pass_barcode01_c8d4ac6d_b8b1e3e8_12.fastq.gz
+-- AMF003_pass_barcode01_c8d4ac6d_b8b1e3e8_13.fastq.gz
+-- AMF003_pass_barcode01_c8d4ac6d_b8b1e3e8_14.fastq.gz
+-- AMF003_pass_barcode01_c8d4ac6d_b8b1e3e8_15.fastq.gz
```

4 Sequence experiments were performed on your samples: *ONT-SEQ-24_PA-0X*.
You can find more detailed list with extra information in the teams *General* channel:
`Document > General > Specieslist-overview`

1. ONT-SEQ-24_PA-01: 24 amplicons - ITS+LSU sequences `(*)`
2. ONT-SEQ-24_PA-02: 24 amplicons - ITS+LSU sequences `(*)`
3. ONT-SEQ-24_PA-04: 96 amplicons - ITS sequences

4. ONT-SEQ-24_PA-05: 48 amplicons - ITS

`(*)` That means both ITS and LSU amplicon products for the **same** sample where mixed befor sequencing. The bioinformatic tool we will able to construct both ITS and LSU sequences at the same time.

The 2 other experiments (5 and 6) contain only ITS sequences per specimen.

Sequence folder

This folder contains the following:

- `fastq_pass` the passed fastq files i.e. your sequence reads

- `reference.fa` a reference file with an ITS and a LSU sequence, only for the reads from 1st and 2nd experiment

- `report...html` a sequence report file, general info on the sequence run (including all samples)

- `samplesheet_....txt`a samplesheet (with the MBB number and sequence barcode).

```
/ONT-SEQ_PA-Benin2024_Input-Data/ONT-SEQ-24_PA-01$tree -L 1
.
+-- fastq_pass
+-- reference.fa
+-- report_AMF003_20241018_1203_c8d4ac6d.html
\-- samplesheet_ONT-SEQ-24_01-withreference.txt
```

Take a look inside the folder and the samplesheet file. *Hint: use the command `cd`, `ls` and `cat`
You will notice that opening a `html` file is not possible in the terminal.
To look at this file, you'll need to download it.

2.2 Practice with test data

Before we start with the actual field data, we first are going to run a test dataset, to see if everything runs smoothly.

Download the test data set

1. Copy the folder `/ONT-SEQ_PA-Benin2024_Input-Data/test_data` to your home folder

2. Verify the content and check if you have the sequence folders, the samplesheet and the reference file (for variant mode).

```
vsc43352@gligar09:~$ls
test_data
vsc43352@gligar09:~$ cd test_data
vsc43352@gligar09:~$/test_data$tree -L 2
.
+-- consensus_mode
|   +-- barcode01
|   +-- barcode02
|   \-- samplesheet_ONT-SEQ-25_05.txt
\-- variant_mode
    +-- barcode12
    +-- barcode14
    +-- reference.fa
    \-- samplesheet_ONT-SEQ-24_02-withreference.txt

6 directories, 3 files
```

2.2.1 NanoPack: Quality Control Nanopack is a tool set of different long read processing and analysis tools. With these tools you can assess the quality of your sequencerun and of 'individual' sequence reads.

Tool: https://github.com/wdecoster/nanopack
Publication: NanoPack: visualizing and processing long-read sequencing data >Wouter De Coster, Svenn D'Hert, Darrin T Schultz, Marc Cruts, Christine Van Broeckhoven, NanoPack: visualizing and processing long-read sequencing dta, Bioinformatics, Volume 34, Issue 15, August 2018, Pages 2666–2669, https://doi.org/10.1093/bioinformatics/bty149

2.2.2 Run Nanopack: Beginner **Exercise 1: Use the command `Nanostat` to check statistics on your sequence data.**
Generating Basic Statistics from a FASTQ File.
You are provided with a compressed FASTQ file (`reads.fastq.gz`). Generate a statistics report and save the output in a folder called `statreports` and filname `nanostats.out`.

1. First navigate to the folder with your sequence reads in the folder ´/test_data/consensus_mode/barcode01´. *Hint: these commands may come in handy ´cd´, ´pwd´ and ´ls´.*
2. When working on the HPC you first need to "activate" the Software

```
$ module load NanoPack/20230602-foss-2023a
```

3. Run the command.

```
# Command
NanoStat --fastq *fastq.gz --outdir statreports -n nanostats.out
```

Note: the ´´ in the command is making sure the NanoStat command is beeing executed on all files present in this folder. 4. Check the output in the newly created subfolder ´statsreports´

```
# OUTPUT
/test_data/consensus_mode/barcode01/statreports$ls
nanostats.out
/test_data/consensus_mode/barcode01/statreports$cat nanostats.out
General summary:
Mean read length:                  438.6
Mean read quality:                  11.6
Median read length:                425.0
Median read quality:                12.6
Number of reads:                24,302.0
Read length N50:                   511.0
STDEV read length:                 154.7
Total bases:                10,659,020.0
Number, percentage and megabases of reads above quality cutoffs
>Q10:   21811 (89.7%) 10.0Mb
>Q15:   2256 (9.3%) 1.4Mb
>Q20:   3 (0.0%) 0.0Mb
>Q25:   0 (0.0%) 0.0Mb
>Q30:   0 (0.0%) 0.0Mb
Top 5 highest mean basecall quality scores and their read lengths
1:      21.4 (530)
2:      21.0 (705)
3:      20.3 (585)
4:      19.7 (274)
5:      19.3 (467)
Top 5 longest reads and their mean basecall quality score
1:      999 (14.6)
2:      970 (13.0)
3:      943 (13.1)
4:      902 (12.9)
5:      888 (14.9)
```

Hint: These commands are usefull here ´cd´, ´ls´, ´cat´

Exercise 2: Use the command `NanoPlot` to create visual plots of your data.

1. First navigate to the folder with your sequence reads in the folder ´/test_data/consensus_mode/barcode01´. *Hint: these commands may come in handy ´cd´, ´pwd´ and ´ls´.*

2. Run the Command

```
# Command
NanoPlot  --fastq *.fastq.gz -o NanoPlot-output
```

3. Check the output

```
# OUTPUT
/test_data/consensus_mode/barcode01/NanoPlot-output$ll
total 3265
-rw-r--r-- 1 vsc43352 vsc43352  463602 Apr 24 15:07 LengthvsQualityScatterPlot_dot.html
-rw-r--r-- 1 vsc43352 vsc43352   51397 Apr 24 15:07 LengthvsQualityScatterPlot_dot.png
-rw-r--r-- 1 vsc43352 vsc43352  689449 Apr 24 15:07 LengthvsQualityScatterPlot_kde.html
-rw-r--r-- 1 vsc43352 vsc43352  121391 Apr 24 15:07 LengthvsQualityScatterPlot_kde.png
-rw-r--r-- 1 vsc43352 vsc43352    6049 Apr 24 15:07 NanoPlot_20250424_1507.log
-rw-r--r-- 1 vsc43352 vsc43352 1362962 Apr 24 15:07 NanoPlot-report.html
-rw-r--r-- 1 vsc43352 vsc43352     788 Apr 24 15:07 NanoStats.txt
-rw-r--r-- 1 vsc43352 vsc43352    8079 Apr 24 15:07 Non_weightedHistogramReadlength.html
-rw-r--r-- 1 vsc43352 vsc43352   30027 Apr 24 15:07 Non_weightedHistogramReadlength.png
-rw-r--r-- 1 vsc43352 vsc43352    8383 Apr 24 15:07 Non_weightedLogTransformed_HistogramReadleng
-rw-r--r-- 1 vsc43352 vsc43352   31773 Apr 24 15:07 Non_weightedLogTransformed_HistogramReadleng
-rw-r--r-- 1 vsc43352 vsc43352    8102 Apr 24 15:07 WeightedHistogramReadlength.html
-rw-r--r-- 1 vsc43352 vsc43352   28858 Apr 24 15:07 WeightedHistogramReadlength.png
-rw-r--r-- 1 vsc43352 vsc43352    8524 Apr 24 15:07 WeightedLogTransformed_HistogramReadlength.h
-rw-r--r-- 1 vsc43352 vsc43352   30033 Apr 24 15:07 WeightedLogTransformed_HistogramReadlength.p
-rw-r--r-- 1 vsc43352 vsc43352  167169 Apr 24 15:07 Yield_By_Length.html
-rw-r--r-- 1 vsc43352 vsc43352   37288 Apr 24 15:07 Yield_By_Length.png
```

Note: You can download the NanoPlot-report.html file to your computer and open it locally.

Exercise 3: Use the command `Chopper` to filter data based on read quality.

1. First navigate to the folder with your sequence reads in the folder ´/test_data/consensus_mode/barcode01´. *Hint: these commands may come in handy ´cd´, ´pwd´ and ´ls´.*

2. Locate and load the `chopper` Module. Check if is has been activated

```
# Commands to use
ml spider chopper
ml chopper/0.9.0-GCCcore-12.3.0
module list
```

3. Run the command

```
# COMMAND
cat *.fastq.gz > allreads.fastq.gz && chopper -q 15 -i allreads.fastq.gz > Q15_filtered.reads.fa
```

Where: - `cat *.fastq.gz > allreads.fastq.gz` => Concatenate all fastq.gz files in the folder to one file `allreads.fastq.gz`. - && => this can be used to peform 2 consecutive commands one after the other, but performs the 2nd only when the 1st is finished successfully.
- `chopper` takes the file allreads.fastq.gz, performs a quality filtering of Q15 and writes the reads to a new file Q15_filtered.reads.fastq 4. Check the output

```
# OUTPUT
/test_data/consensus_mode/barcode01$cat *.fastq.gz > allreads.fastq.gz && chopper -q 15 -i allre
Kept 2256 reads out of 24302 reads
```

```
/test_data/consensus_mode/barcode01$ls
allreads.fastq.gz                                      FBA60703_pass_barcode01_e2391328_167a61ad_
FBA60703_pass_barcode01_e2391328_167a61ad_0.fastq.gz  FBA60703_pass_barcode01_e2391328_167a61ad_
FBA60703_pass_barcode01_e2391328_167a61ad_1.fastq.gz  FBA60703_pass_barcode01_e2391328_167a61ad_
FBA60703_pass_barcode01_e2391328_167a61ad_2.fastq.gz  FBA60703_pass_barcode01_e2391328_167a61ad_
```

*Note: You should by now have these files and folders in your working directory /test_data/consensus_mode/barcode01

Exercise 4: Use the command `NanoComp` to compare.
`NanoComp` is an excellent tool to compare 2, or more, sequence files. In this example you will compare the original `allreads.fastq.gz` and the filtered reads `Q15_filtered.reads.fastq`.

1. First navigate to the folder with your sequence reads in the folder ´/test_data/consensus_mode/barcode01´. *Hint: these commands may come in handy ´cd´, ´pwd´ and ´ls´.*

2. Run the command

```
# COMMAND
NanoComp --fastq allreads.fastq.gz Q15_filtered.reads.fastq --outdir compare_filtered --plot vio
```

3. Check the ouput

```
# OUTPUT
/test_data/consensus_mode/barcode01$cd compare_filtered/
/test_data/consensus_mode/barcode01/compare_filtered$ls
NanoComp_20250424_1550.log      NanoComp_N50.png                          NanoComp_OverlayHist
NanoComp_lengths_violin.html    NanoComp_number_of_reads.html             NanoComp_OverlayHist
NanoComp_lengths_violin.png     NanoComp_number_of_reads.png              NanoComp_OverlayHist
NanoComp_log_length_violin.html NanoComp_OverlayHistogram.html            NanoComp_OverlayLogH
NanoComp_log_length_violin.png  NanoComp_OverlayHistogram_Normalized.html NanoComp_OverlayLogH
NanoComp_N50.html               NanoComp_OverlayHistogram_Normalized.png  NanoComp_OverlayLogH
```

*Note: You can dowlod the file `NanoComp-report.html` and open the file on your computer. See the difference?

2.2.3 Run Nanopack: Advanced If you want to explore more features on the `NanoPack` program, you can visit the website:
https://passelma42.github.io/Field-Sequencing/Tutorial-NanoPack/.

!Important!:

- Don't run the `conda` commands, no need when using the HPC
- Focus on the Exercises and don't forget to activate the necessary modules before running the commands
- All test data for the Advanced Nanopack exercises can be downloaded here: `/ONT-SEQ_PA-Benin2024_Bioinfocourse/nanopack_Advanced_exampledata`

3. EPI2ME: wf-amplicon

EPI2ME provides best practice bioinformatics analyses for nanopore sequencing. It offers bioinformatics for all levels of expertise and is designed by Oxford Nanopore especially to work on long read nanopore sequencing data.
There is a desktop version available, with all necessary software packaged in a single workflow. Depending

on your research there are different workflows available tailored to your needs.
These workflows are also available on the command line. No need to install the bioinformatic tools yourself, except the necessary tools to load the workflows.

We will be using the wf-amplicon workflow. As mentioned before, when using this on your own computer, no need to install any bioinformatic tools except Nextflow, docker or singularity/apptainer.

FYI
General info: https://epi2me.nanoporetech.com/
Workflows Overview: https://epi2me.nanoporetech.com/wfindex/
Installation instructions: https://epi2me.nanoporetech.com/epi2me-docs/installation/
Amplicon Workflow: https://github.com/epi2me-labs/wf-amplicon

3.1 Set HPC environment

Before we can start using the `wf-amplicon` we need to prepare our HPC environment. For this you need to copy the script `singularity_environment_set.sh` and execute.
These are the steps to follow:

1. Change directory to your VSC-DATA folder

```
$ cd $VSC_DATA
```

2. copy the folder scripts to your current location i.e. your VSC_DATA folder

```
$ cp -r <path to>/ONT-SEQ_PA-Benin2024_Input-Data/scripts .
```

3. Change to the created folder

```
$ cd scripts
```

4. Give executable rights to the script

```
$ chmod  u+x  singularity_environment_set.sh
```

5. Execute the script using source which will allow the export commands affect your current shell environment

```
$ source singularity_environment_set.sh
```

6. Check if the singularity folder has been created and also if it contains the 3 subfolders `cache`, `tmp`, and `image`

```
$ ll $VSC_SCRATCH/singularity
```

7. Check if the environment is ready and set to go. These commands should give you the actual paths to the folders

```
echo $SINGULARITY_CACHEDIR
echo $SINGULARITY_TMPDIR
echo $NXF_SINGULARITY_CACHEDIR
```

3.2 Run Test - Consensus mode

In the previous chapter you already worked on this data to packtice the `Nanopack` tool. Now it is time to run the `wf-amplicon` pipeline.

3.2.2 Check samplesheet

1. Navigate to the directory `/test_data/consensus_mode`

2. Take a look at the samplesheet

```
# SAMPLESHEET LAYOUT
/test_data/consensus_mode$cat samplesheet_ONT-SEQ-25_05.txt
barcode,alias,type
barcode01,MBB-24-154_barcode01_ONT-SEQ-25_PA-05,test_sample
barcode02,MBB-24-156_barcode02_ONT-SEQ-25_PA-05,test_sample
```

This shows you 3 columns `barcode`, `alias` and `type`.

3.2.3 Run the program To run the workflow the only software we need to activate is `Nextflow`. Activating all other software will be taken care of automatically while running the workflow. Make sure you are in the folder that contains all your `barcodexx` folders. In this case it should be `/test_data/consensus_mode`.

1. Activate the nextflow software

```
$ module load Nextflow/24.10.2
```

2. Navigate to the folder `/test_data/consensus_mode` which contains the `barcodexx` folders.
3. Run the nextflow command to activate the `wf-amplicon`

```
# consensus_mode command
nextflow run epi2me-labs/wf-amplicon \
    --fastq ./consensus_mode \
    --sample_sheet ./consensus_mode/samplesheet_ONT-SEQ-25_05.txt \
    --out_dir output-consensus_mode \
    -profile singularity
```

This analysis will pull the necessary containers with all needed software packages, then it will run the tools on your data.
You can monitor the progress on your terminal.
Because we work with a small test dataset it should not take more than 5 minutes to run.

```
# OUTPUT CONSENSUS MODE
/test_data/output-consensus_mode$ll
total 4776
-rw-r--r-- 1 vsc43352 vsc43352    1448 Apr 30 12:05 all-consensus-seqs.fasta
-rw-r--r-- 1 vsc43352 vsc43352     107 Apr 30 12:05 all-consensus-seqs.fasta.fai
drwxr-xr-x 2 vsc43352 vsc43352    4096 Apr 30 12:05 execution
drwxr-xr-x 4 vsc43352 vsc43352    4096 Apr 30 12:05 MBB-24-154_barcode01_ONT-SEQ-25_PA-05
drwxr-xr-x 4 vsc43352 vsc43352    4096 Apr 30 12:05 MBB-24-156_barcode02_ONT-SEQ-25_PA-05
-rw-r--r-- 1 vsc43352 vsc43352    1597 Apr 30 12:01 params.json
-rw-r--r-- 1 vsc43352 vsc43352     138 Apr 30 12:01 sample_sheet.csv
-rw-r--r-- 1 vsc43352 vsc43352     251 Apr 30 12:05 versions.txt
-rw-r--r-- 1 vsc43352 vsc43352 2434136 Apr 30 12:05 wf-amplicon-report.html
```

3.2.4 Analyse output Taking a look at the output:

Check here for more info: https://github.com/epi2me-labs/wf-amplicon?tab=readme-ov-file#output-options

Title	File path	Description	Per sample or aggregated
Workflow report	`./wf-amplicon-report.html`	Report for all samples.	aggregated
Sanitized reference file	`./reference_sanitized_seqIDs.fasta`	Some programs used by the workflow don’t like special characters (like colons) in the sequence IDs in the reference FASTA file. The reference is thus “sanitized” by replacing these characters with underscores. Only generated in variant calling mode.	aggregated
Sanitized reference index file	`./reference_sanitized_seqIDs.fasta.fai`	FAI index for the sanitised reference FASTA file.	aggregated
Alignments BAM file	`./{{ alias }}/alignments/{{ alias }}.aligned.sorted.bam`	BAM file with alignments of input reads against the references (in variant calling mode) or the created consensus (in de-novo consensus mode).	per-sample
Alignments index file	`./{{ alias }}/alignments/{{ alias }}.aligned.sorted.bam.bai`	Index for alignments BAM file.	per-sample
De-novo consensus FASTQ file	`./{{ alias }}/consensus/consensus.fastq`	Consensus sequence file generated by de-novo consensus pipeline.	per-sample
Consensus FASTA file	`./{{ alias }}/consensus/medaka.consensus.fasta`	Consensus sequence file generated by variant calling pipeline.	per-sample
Variants VCF file	`./{{ alias }}/variants/medaka.annotated.vcf.gz`	VCF file of variants detected against the provided reference.	per-sample
Variants index file	`./{{ alias }}/variants/medaka.annotated.vcf.gz.csi`	Index for variants VCF file.	per-sample
Combined de-novo consensus sequences	`./all-consensus-seqs.fasta`	FASTA file containing all de-novo consensus sequences.	aggregated
Combined de-novo consensus sequences index	`./all-consensus-seqs.fasta.fai`	FAI index for the FASTA file with the combined de-novo consensus sequences.	aggregated
IGV config JSON file	`./igv.json`	JSON file with IGV config options to be used by the EPI2ME Desktop Application.	aggregated

Important for screening your results is the `wf-amplicon-report.html` and for downstream applications offcourse your `fasta` files.

3.3 Run Test - Variant mode

Running in variant mode allows us to run the same analysis as before but since our test data contains reads from both ITS and LSU, we can bioinformatically construct consensus sequences per sample for both genes at the same time.

3.3.2 Check samplesheet

1. Navigate to the directory `/test_data/variant_mode`

2. Take a look at the samplesheet

```
# SAMPLESHEET LAYOUT
/test_data/variant_mode$cat samplesheet_ONT-SEQ-24_02-withreference.txt
barcode,alias,type,ref
barcode12,MBB-24-016_barcode12_ONT-SEQ-24_02,test_sample,ITS LSU
barcode14,MBB-24-019_barcode14_ONT-SEQ-24_02,test_sample,ITS LSU
```

This shows you 4 columns `barcode`, `alias`, `type` and `ref`.
3. Take a look at the reference file

```
# REFERENCE FILE
/test_data/variant_mode$cat reference.fa
>ITS
GAAGTAAA...
>LSU
TATCAAT...
```

This file is a fasta file showing 2 sequences with IDs `ITS` and `LSU`, the same as in your samplesheet `ref` column. This reference file is used to “search” for ITS or LSU sequences in your sequence reads i.e. in the `barcodexx` folders.

3.3.3 Run the program To run the workflow the only software we need to activate is `Nextflow`. Activating all other software will be taken care of automatically while running the workflow. Make sure you are in the folder that contains all your `barcodexx` folders. In this case it should be `/test_data/consensus_mode`.

1. Activate the nextflow software

```
$ module load Nextflow/24.10.2
```

2. Run the nextflow command to activate the `wf-amplicon`

```
# consensus_mode command
nextflow run epi2me-labs/wf-amplicon \
    --fastq ./variant_mode \
    --sample_sheet ./variant_mode/samplesheet_ONT-SEQ-24_02-withreference.txt \
    --reference ./variant_mode/reference.fa \
    --out_dir output-variant_mode \
    -profile singularity
```

This analysis will pull the necessary containers with all needed software packages, then it will run the tools on your data. You can monitor the progress on your terminal.
Since we work with a small test dataset it should not take more than 5 minutes to run.

3.3.4 Analyse output In variant-mode the output is slightly different.

```
# OUTPUT VARIANT MODE
/test_data/output-variant_mode$ll
total 5001
drwxr-xr-x 2 vsc43352 vsc43352    4096 Apr 30 12:23 execution
drwxr-xr-x 5 vsc43352 vsc43352    4096 Apr 30 12:22 MBB-24-016_barcode12_ONT-SEQ-24_02
drwxr-xr-x 5 vsc43352 vsc43352    4096 Apr 30 12:22 MBB-24-019_barcode14_ONT-SEQ-24_02
-rw-r--r-- 1 vsc43352 vsc43352    1645 Apr 30 12:21 params.json
-rw-r--r-- 1 vsc43352 vsc43352    1666 Apr 30 12:22 reference_sanitized_seqIDs.fasta
-rw-r--r-- 1 vsc43352 vsc43352      38 Apr 30 12:22 reference_sanitized_seqIDs.fasta.fai
-rw-r--r-- 1 vsc43352 vsc43352     152 Apr 30 12:21 sample_sheet.csv
-rwxr--r-- 1 vsc43352 vsc43352    2286 Apr 30 12:35 variantmode-consensus-rename2.0.sh
-rw-r--r-- 1 vsc43352 vsc43352    7072 Apr 30 12:41 variant-mode-consensus-sequences.fasta
-rw-r--r-- 1 vsc43352 vsc43352     251 Apr 30 12:22 versions.txt
-rw-r--r-- 1 vsc43352 vsc43352 2529804 Apr 30 12:23 wf-amplicon-report.html
```

The main difference is that you're not only getting consensusfiles but also a vcf file. This file contains variants detected against the provided reference.
Can be usefull if you are working with closely related species and you are interested in Single Nucleotid Polymorphism (SNP) information.
In our case the sequences in the reference file were only a means to be able to sort out all ITS and LSU reads and assemble consensus sequences from both.
>More info on VCF files can be found here: https://en.wikipedia.org/wiki/Variant_Call_Format

What we do miss here is a combined fasta file with all our ITS and LSU sequences and matching headers. This is something the workflow doesn't provide for us.
For this feature we will use the script `variantmode-consensus-rename2.0.sh`.

Follow these instructions to run the script and build a combined fasta file for all your consensus sequences:

1. Copy the script to your outputfolder

```
$ cp <path-to>/scripts/variantmode-consensus-rename2.0.sh <path to>/test_data/output-variant_moc
```

2. Change permissions to be able to execute the script

```
$ cd <path-to>/test_data/output-variant_mode
$ chmod u+x variantmode-consensus-rename2.0.sh
```

3. Make sure you are in the output folder and execute following command

```
for dir in ./MBB*/; do ./variantmode-consensus-rename2.0.sh "$dir" -o variant-mode-consensus-sec
```

This commmand will search for all **MBB-xx-xx_barcodexx_ONT-SEQ-xx_xx** in the output folder, find all consensus sequences per barcode and generate one combined fasta file called `variant-mode-consensus-sequences.fasta`.

3.4 Run with real data

Still with us? Time to put theory into practice. You can all start with your first analysis.

1. 10 samples are selected for you. Check the **Mycoblitz2024_specimenlist** on Teams to find out which samples are selected for you.
2. Find and download your samples from the `<path to>/ONT-SEQ_PA-Benin2024_Input-Data/ONT-SEQ-24_PA-01` download folder.
3. Run either the consensus-mode and/or variant-mode (only possible for experiments 1 and 2)
4. Check the results

Optional: If you finish earlier, you can try a 2nd analysis just for practice! Make sure to start a 2nd working folder Analysis2.

The samplesheet: have a look
>Remember the structure of the dataset explained in Chapter **2.Nanopore data**.
Each *ONT-SEQ-2X_PA-0X* folder contains a different samplesheet.

```
# EXAMPLE SAMPLE SHEET
<path to>/ONT-SEQ_PA-Benin2024_Input-Data/ONT-SEQ-24_PA-01$cat samplesheet_ONT-SEQ-24_01-withref

barcode,alias,type,ref
barcode01,MBB-24-001_barcode01_ONT-SEQ-24_01,test_sample,ITS LSU
barcode02,MBB-24-002_barcode02_ONT-SEQ-24_01,test_sample,ITS LSU
barcode03,MBB-24-003_barcode03_ONT-SEQ-24_01,test_sample,ITS LSU
barcode04,MBB-24-004_barcode04_ONT-SEQ-24_01,test_sample,ITS LSU
```

In this samplesheet `ONT-SEQ-24_PA-01-withreference.txt` you see that `barcode01 = MBB-24-001` (for this sequence experiment only!).
Now, have a look in the master species list in teams:
You can find the detailed list with sample information in the teams *General* channel:
`General > Files > Specieslist-overview > Mycoblitz2024-specimenlist.xlsx`

Example workflow: Copy your selection to your Analysis folder.

- I want to download samples from sequence experiment `ONT-SEQ-24_PA-01`

- The selected barcodes for me are: barcode 01 up to 10
  1.Change directory to your VSC-DATA folder

```
cd $VSC_DATA
```

2.Make a directory to copy your sequence reads to

```
mkdir Analysis01
```

3.Change to the created folder

```
cd Analysis01
```

4.Recursively copy the barcode01 folder to your Analysis01 folder

```
cp -r <path to>/ONT-SEQ_PA-Benin2024_Input-Data/ONT-SEQ-24_PA-01/barcode01 $VSC_DATA/Analysis01
```

5.Check the content of the folder, the barcode01 folder should be copied to this Location

```
ls $VSC_DATA/Analysis01
```

REPEAT FOR THE 9 OTHER BARCODEFOLDERS YOU SELECTED
HINT: use a forloop to iterate over different barcode folders in one command

```
for i in {01..10}; do cp -r <path to>/ONT-SEQ_PA-Benin2024_Input-Data/ONT-SEQ-24_PA-01/barcode$i
```

Remark: Not only copy the `barcode` folders but also the `samplesheet` from the sequencing experiment you take the `barcodes` folders from.

When this exercise is finishe, you can run a 2nd Analysis. Just pick 10 barcode folders from the same sequencing experiment and run the exercise again.
Be aware tha only experiments `ONT-SEQ-24_01` and `ONT-SEQ-24_02` allow you to run the wf-amplicon workflow in variant-mode!