



**Week 1: Advanced Bioinformatics**  
**Dr. Princess Rodriguez**

**MMG 232**  
**Spring 2023**

*You can call me:*

Dr. Rodriguez

OR

Princess

# My Background

- Bachelors in Biology
- Masters in Immunology and Microbial Disease
- Doctor of Philosophy in Cellular Molecular Biology
  - Epigenetics
  - Next Generation Sequencing
  - B cell biology



# Course Structure

- This course was designed for primarily an **undergraduate** audience
- Students who have no formal training in the data or computer sciences
- Graduate students and/or computer science majors be warned that you may find the class pace slow...*still*

# **My background continued...**

- Last semester, was my first-time teaching a lecture-based course called MMG231: Bioinformatics & Data Analysis.
- Some of you took this course with me last semester, others took this course with Dr. Dragon, while others are graduate students who could override the prerequisites.

# What I am keeping *from last semester* (i.e. what you can expect)

"I think the **homework assignments and projects** were a really great way for me to be able to apply what I was learning."

**"During in-class activities"**

"The **final project** was my favorite part of the course. It helped us to engage with the coding, troubleshoot, and learn for ourselves!"

"Dr. Rodriguez also **made ample time** for students who were having difficulty with their code or had questions about it."

"...it seems as if you could get the **same amount of knowledge** from reading the powerpoints in your own time than attending lectures."

# Learning and expertise varies

- Auditory learners, visual learners, while others learn-by-doing
- There are varying levels of expertise in this room. Some of you are already proficient at bash or R therefore, I do try my best to provide students with the same materials.
  - This way you know ahead of time... is this class for me? Should I attend? Do I know the material enough to complete the homework assignment?
    - **If the answer is no, you should probably attend class...**
  - If you do decide to skip class, and I know I covered the *materials at length during class*, I now have the option to say, "Please see the class materials for the day you missed to complete the assignment"

# How I am *trying* to make this course better *than last semester*

*“I just wish there was **more time** to go over the coding parts because if you got messed up then you were left behind.”*

*“I wish we had moved a **little slower** during the coding sections.”*

*“The jump between the first lectures on R where we were looking at the basics to the later lectures where we were coding for advanced genetic analysis was **unrealistic for our skill level**.”*

*“**Include homework or small assignments** so that we are able to apply what we learn in class..” (coding)*

*“The R studio section **felt like way too much** for individuals without prior knowledge”*

*“I would have appreciated more time in class to actually work through **writing our own code** so that we could work through issues and better understand what each line does and how to build a working script.”*



**Let's go over the class schedule  
and syllabus**

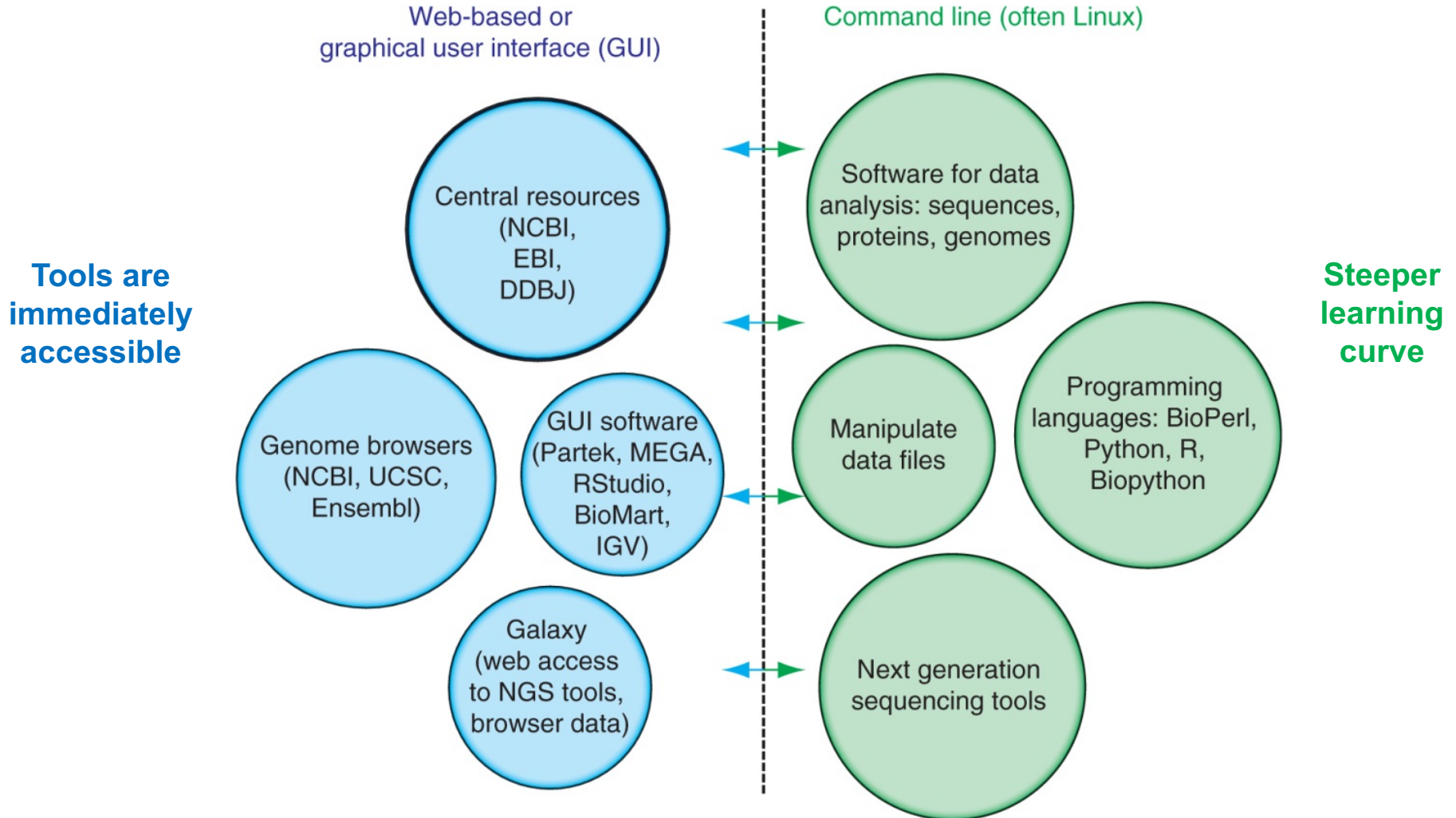
# Goals

- Why work on the command line?
- Terminal (or terminal-like) application for each OS
- Logging in to the VACC
- Interacting with remote servers
- File transfers

# Programming for biology

- Think of learning the command line like any bench experiment
  - Protocols + SOPs
  - Trial + error
  - Reformatting
  - Analysis
  - Optimize to find the most efficient and reproducible way

# Two approaches to bioinformatics



# What is the command line?

- Underneath the Graphical User Interface (GUI) of your computer is the command line that runs your Operating System (OS)
- Working this way gives you access to internal controls, remote servers, and the ability to customize workflows (scripts)
- We access it with a shell (Terminal, PowerShell) which lets you give your computer commands via keyboard rather than a point and click

# Command line + Bioinformatics

## Why bother?

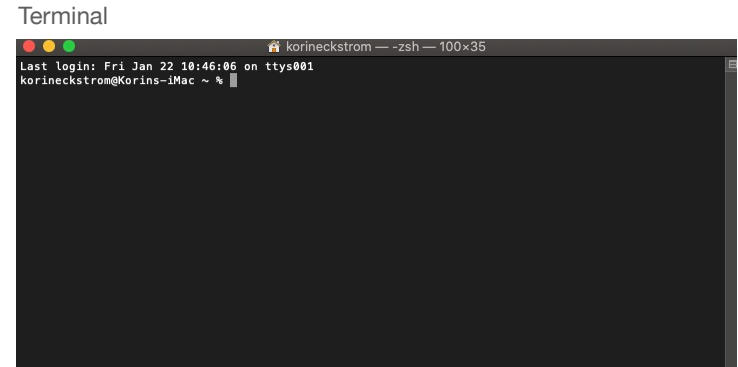
- GUI tools require memory just to run the interface, and most bioinformatics applications are memory intensive to begin with or have additional flexibility on the command line
- Ex. BLAST on the NCBI website is quite limited compared to what you will see in this course
- Most of the time you will be working on some sort of HPC (High Performance Cluster) or remote server, as typical PCs do not have the required storage or compute power
- Reproducibility
- Ability to automate + create pipelines, or work with many files at once

# Reproducibility

- Human error
- When we try to do the same thing 100 times we make mistakes
- As long as you told it to do the right thing, a computer can perform the same task thousands of times without error
- Unambiguous record via history and logs
- Easier to communicate steps to others for reproducibility

# Topics we will cover now:

- How to access your shell



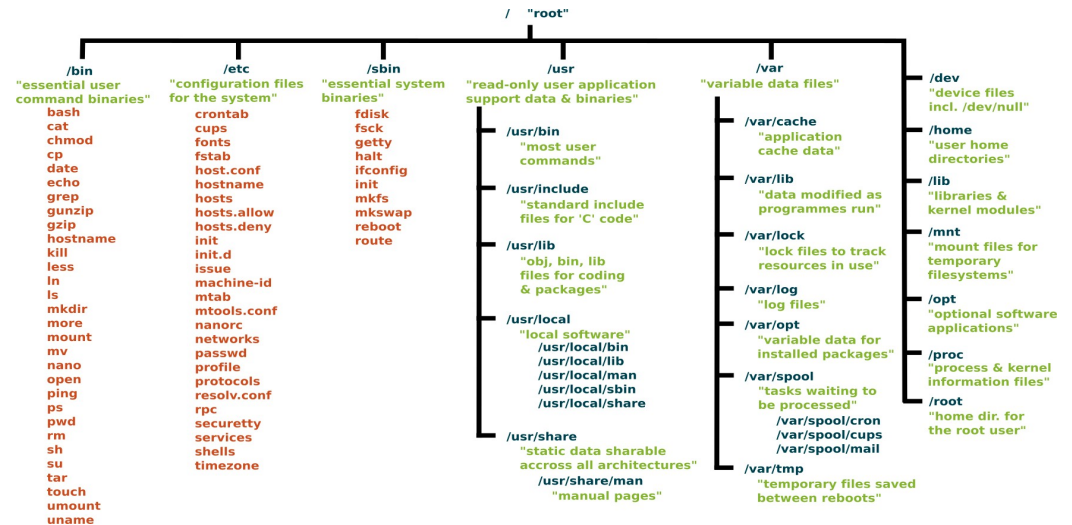
- Remote Servers & Benefits



# Topics we will cover now:

- Syntax = the “grammar” of a programming languages, needs to be **exact** for the computer to understand

- Directories



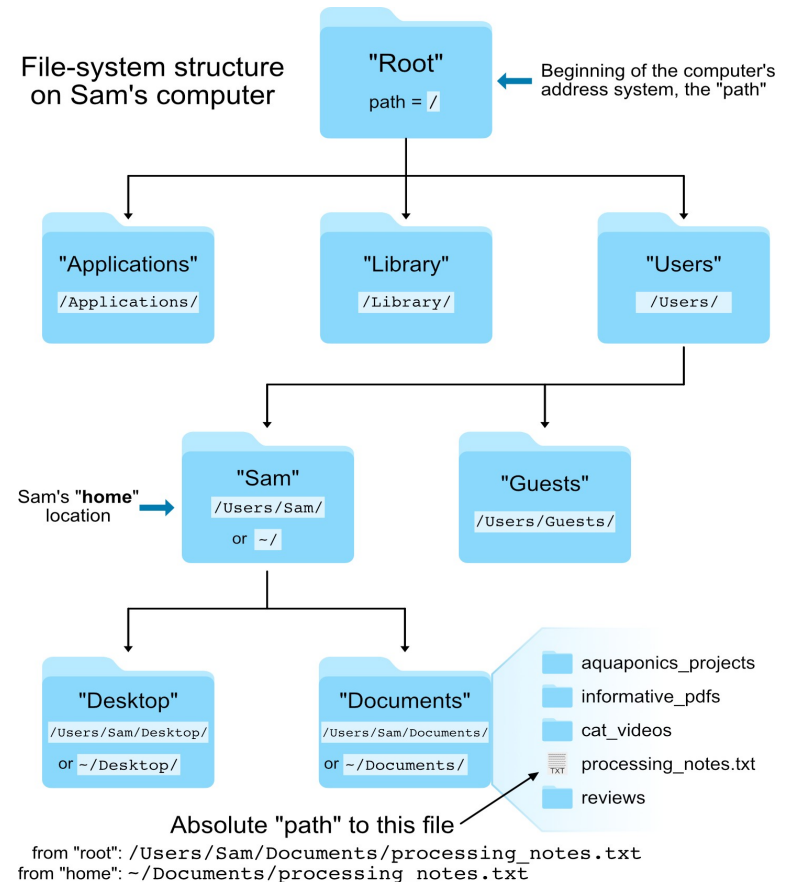
# Topics we will cover:

## 3-week overview

## PATH

### Where am I? Where is that file?

- Can't click or drag and drop, and the computer can only find things if you tell it where they are
- PATH vs \$PATH
- Absolute vs relative
- This would get annoying, so there are ways to add things you use a lot to your user profile so that you don't need the PATH every time
- There are two locations all Unix-based systems share:
  - "Root" = where the address system of the computer starts
  - "Home" = where the user's location starts



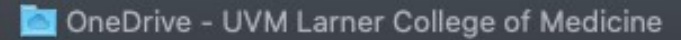
# Topics we will cover:

## 3-week overview

## Naming Files

- Case sensitive
  - File.txt  $\neq$  file.txt
- Spaces without “” won’t be read correctly
- Adding dates can help with versions
- File naming tips:
  - Underscores, e.g.
  - Dashes, e.g. file-name.xxx
  - No separation, e.g.
  - Camel case, where the first letter of each section of text is capitalized, e.g. FileName.xxx
- Can store a lot of useful information in the names of your files
  - Not just “DifferentialExpression.R”, but “Eckstrom\_DESeq2\_COVID\_20210204.R”
  - Dates, analysis type, investigator, location, etc.

GUI:

A screenshot of a OneDrive folder icon with the text "OneDrive - UVM Larner College of Medicine" next to it.

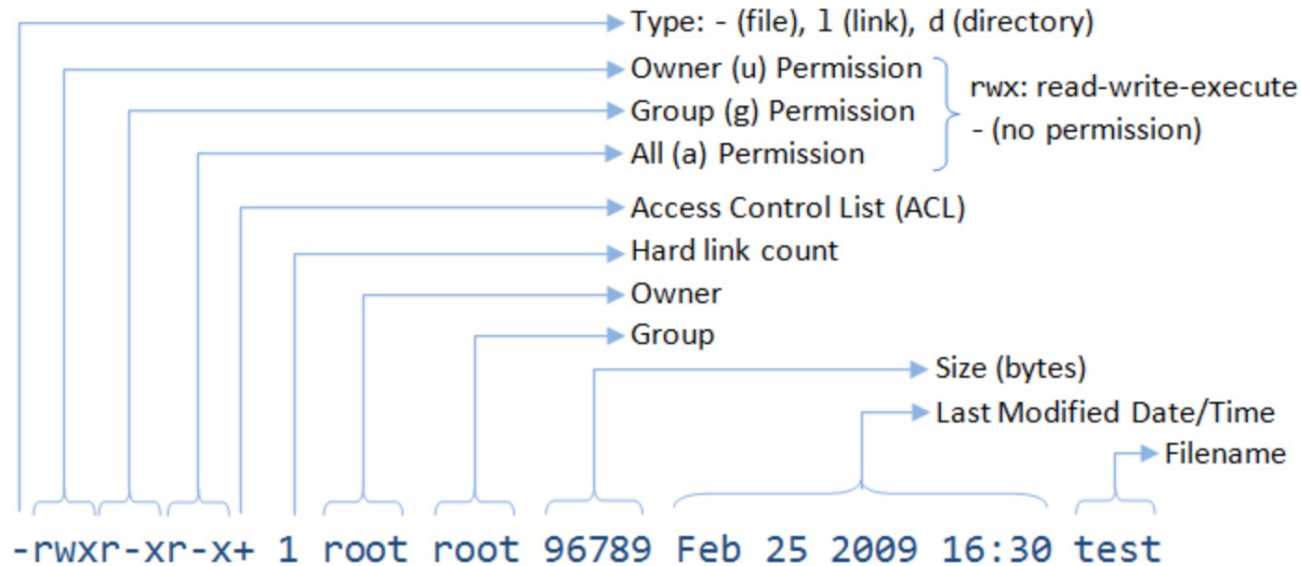
becomes....

A screenshot of a file path in a command prompt or terminal window, showing the path "/OneDrive\ -\ UVM\ Larner\ College\ of\ Medicine/".

# Topics we will cover:

## 3-week overview

### File Permissions



# Topics we will cover:

## 3-week overview

## Organization

- Many pipelines don't just generate the final results file, but all of the intermediate steps it took to get there
- Having everything in one directory can not only be overwhelming, but can make automation that much harder
  - You want to assemble a bunch of genomes from your trimmed fastq files, but your raw data is in the same directory and they all have the same name
- Good organization can save you many headaches later on!
- You might do something like:
  - RawReads/
  - TrimmedReads/
  - Scripts/

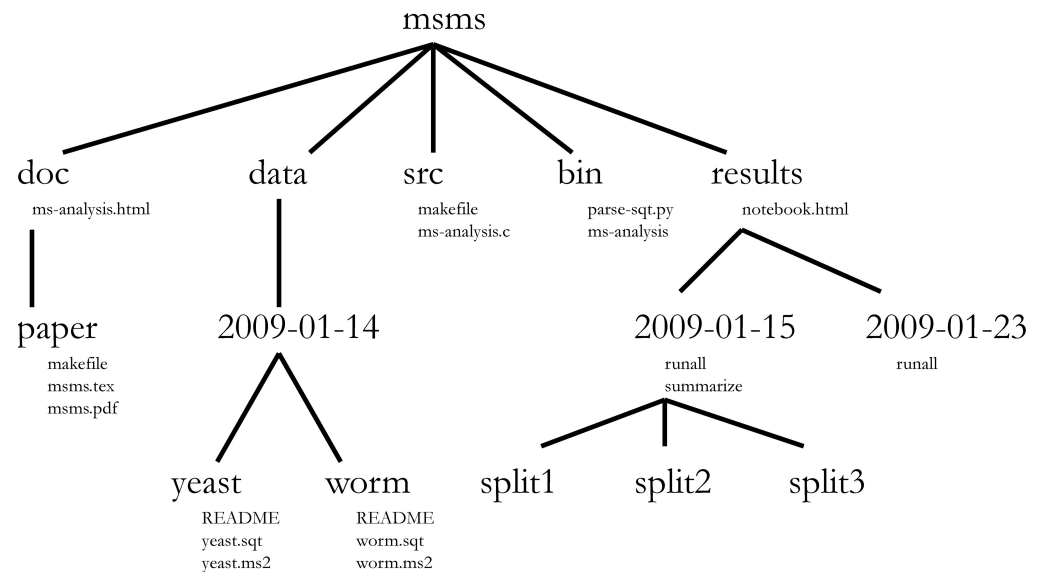
```
samplename/  
├── 0.basecall  
│   ├── samplename.fq  
│   └── nanoplots  
├── 1.assemble  
│   ├── samplename_merged.fasta  
│   ├── samplename_raw_assembly.fa  
│   ├── samplename_raw_assembly.fa.amb  
│   ├── samplename_raw_assembly.fa.ann  
│   ├── samplename_raw_assembly.fa.bwt  
│   ├── samplename_raw_assembly.fa.fai  
│   ├── samplename_raw_assembly.fa.pac  
│   ├── samplename_raw_assembly.fa.paf  
│   ├── samplename_raw_assembly.fa.sa  
│   ├── assemble_100m (if specified)  
│   └── assemble_250m (if specified)  
├── 2.polish  
│   ├── samplename_polished.corrected.fasta  
│   ├── samplename_polished.fasta  
│   ├── samplename_polished.fasta.bam  
│   ├── samplename_polished.fasta.bam.bai  
│   ├── samplename_polished.fasta.fai  
│   ├── samplename_polished.fasta.misassemblies.tsv  
│   ├── medaka (if specified)  
│   ├── pilon (if specified)  
│   └── racon (if specified)  
├── 3.circularization  
│   ├── 1.candidate_genomes  
│   ├── 2.circularization  
│   ├── 3.circular_sequences #circularized genomes  
│   ├── 4.samplename_circularized.corrected.fasta  
│   ├── 4.samplename_circularized.fasta  
│   ├── 4.samplename_circularized.fasta.bam  
│   ├── 4.samplename_circularized.fasta.bam.bai  
│   ├── 4.samplename_circularized.fasta.fai  
│   └── 4.samplename_circularized.fasta.misassemblies.tsv  
└── 5.final  
    ├── samplename_final.fa  
    └── samplename_final.fa.fai
```

# Topics we will cover:

## 3-week overview

### Organization User Level

- If that's for an individual project, your VACC account might end up looking something like this
- Each new project gets a directory
- Versions or samples within these
- Tools or programs in one place that can be accessed from anywhere



# Topics we will cover:

## Submitting Jobs

### Slurm scheduler

- If everyone tried to run their scripts at the same time, nodes would get mixed up, large jobs would take all the resources, and inefficiencies would be created
- To prevent this, HPCs run using “batch systems”, which allows users to submit jobs requesting specific resources + with specific instructions in the form of a script
  - Instead of a single command at a time, you can submit whole workflows this way
- Knowledge Base: [Run a job](#)

```
1 #!/bin/bash
2 #SBATCH --partition=bigmem
3 #SBATCH --nodes=1
4 #SBATCH --ntasks=4
5 #SBATCH --mem=50G
6 #SBATCH --time=20:00:00
7 #SBATCH --job-name=racon_r1_metaflye
8 # %x=job-name %j=jobid
9 #SBATCH --output=%x_%j.out
10 # Notify me via email -- please change the username!
11 #SBATCH --mail-user=korin.eckstrom@med.uvm.edu
12 #SBATCH --mail-type=ALL
13 #
14 # change to the directory where you submitted this script
15 cd ${SLURM_SUBMIT_DIR}
16 #
17 # your job execution follows:
18 echo "Starting sbatch script myscript.sh at:`date`"
19 # echo some slurm variables for fun
20 echo " running host:    ${SLURMD_NODENAME}"
21 echo " assigned nodes:  ${SLURM_JOB_NODELIST}"
22 echo " jobid:           ${SLURM_JOBID}"
23
24
25 cd /users/k/e/keckstro/scratch/working/directory/for/a/project
26 source activate ONT_tools
27
28 for i in /users/k/e/keckstro/scratch/working/directory/for/a/project/*_flye.fasta
29 do
30     SAMPLE=$(echo ${i} | sed "s/_flye\.fasta//")
31     echo ${SAMPLE}_flye.fasta
32     minimap2 -ax map-ont ${SAMPLE}_flye.fasta ${SAMPLE}_trimmed.fastq > ${SAMPLE}_to_draft.sam
33     racon ${SAMPLE}_trimmed.fastq ${SAMPLE}_to_draft.sam ${SAMPLE}_flye.fasta > ${SAMPLE}_racon_r1.fasta
34 done
```

**Let's get started**