Physalia-course: Population genomics

Thibault Leroy (thibault.leroy@inrae.fr)

Part 1: Bioinformatics
May 26, 2025

General context

The analysis workflow, which includes quality control (QC) of raw sequencing data, trimming, mapping to the reference genome, QC of the mapping results, and PCR deduplication, is universally applicable across various sequencing approaches. These approaches include Pool-Seq, individual sequencing with low coverage, and moderate coverage sequencing strategies. While the downstream analysis beyond SNP calling may differ depending on the experimental design and objectives, the initial steps remain consistent and form the foundation for high-quality data processing.
In the example below, the initial steps are performed on data from a Pool-Seq project; however, these same steps could just as easily be applied to other datasets. Subsequently, distinct SNP calling strategies are employed to accommodate the specific characteristics of three dataset types: Pool-Seq data, low-coverage individual sequencing, and moderate-coverage individual sequencing.

IMPORTANT: Please note that there is no universal pipeline that works for all sequencing data, biological models, and research contexts! The examples provided here are for guidance. In general, default parameter values (e.g. for read mappers such as bwa mem) work for most species, but it might be needed to use some non-default parameter values depending on the context.

Opening an SSH connection on a remote server

To connect to a remote server using SSH, open your terminal and use the following command:
`ssh username@hostname`

To connect on the physalia cluster, we should also provide a certificate (a key for the cybersecurity), use provided by the Physalia-course team. For example, if you are "user2", you will only need to provide the c2.pem certificate.
`chmod 600 c2.pem`
To connect on the server, then use the following command (if you are "user2" of course, if you are another user, adjust the command accordingly):
`ssh -i c2.pem user2@[hostname]`
NOTE: The hostname of the Physalia machine changes daily. The machine's ID address will be provided separately via Slack.
The cluster is expected then to ask you: "*Are you sure you want to continue connecting (yes/no/[fingerprint])?*". Write "yes"
As soon as your are connected on the cluster, please load the conda environment corresponding to the day:
`conda activate Workshop_TL_YB_Calling`

Even if we don't need it at this stage of the course, it is already important to know how to transfer files from your computer (local) to the remote server using SCP. In this case, you should use this command from your local terminal:
`scp -i c2.pem [myfile] user2@hostname:[PATH_TO_DIRECTORY]`
Of course [myfile] should be replaced by the name of the file you would like to send on the cluster, and the [PATH_TO_DIRECTORY] should be replaced by the exact architecture on the cluster (e.g.

Simularly, do transfer files from the remote server to your computer using SCP, use this command from your local terminal:
`scp -i c2.pem user2@[hostname]:[PATH_TO_DIRECTORY]/[myfile] .`
NOTE: When using scp, you need three arguments: the scp itself, followed by two locations, with each separated by a space. In the previous example, the dot at the end of the command is crucial! It signifies "the current directory" on your local machine. This ensures that the file is placed in the directory where you are currently located.
Assuming that you are user2 and that you want to download the full directory of the cluster, use the "-r" option, e.g. `scp -i c2.pem -r user2@[hostname]:~/Share/Day1_bioinfo ~/Physalia-course/` to download all the results in a directory Physalia-course of your own computer (if you want to create this "Physalia-course" directory, write `mkdir Physalia-course` to create a new directory (mkdir stands for make directory).
IMPORTANT: If you wish to download the entire repository, please wait the end of the course today to do so!
The "~" symbol refers to your home directory. You can navigate into a folder using `cd [DIRECTORY]` (e.g. `cd Physalia-course`) and move up one level in

the directory structure with `cd ...` If you are totally unfamiliar with commands like mkdir, cd, ls, or other basic terminal functions, don't worry! Just inform the instructors via Slack as soon as possible in order to allow us to assist you as much as possible.

For today, simply copy the repository from the Share directory to a folder you create in your home directory on the cluster (i.e. no need to use scp to import it to your personal computer)

```
cp -r ~/Share/Day1_bioinfo/ ~
```

Up to step 9 of this proposal, we will focus exclusively on the oak data located in the "data-oak" directory within ~/Day1_bioinfo/. Navigate to this directory with:

```
cd ~/Day1_bioinfo/data-oak
```

Step 1: Evaluating the reference genome

Evaluating the quality of a reference genome is a critical step before analyzing raw sequencing data, yet it is often overlooked. This is particularly true in population genomics, where many analyses rely on assumptions, such as the presence of linkage disequilibrium between highly differentiated markers and causal genes. A fragmented or low-quality reference genome not only limits the ability to generate meaningful results but also contributes to more general issues, such as misaligned reads and inaccurate variant calls. These latter biases can significantly affect the study of genetic diversity, selection, and population structure. Tools like Assemblathon enable the rapid generation of key metrics, such as contiguity (e.g. the number of contigs and N50). Additional analyses, such as examining GC content and transposable element (TE) content across the genome, also provide valuable insights. Time spent thoroughly evaluating the reference genome is always worthwhile!

Some important summary statistics:
N50: The length of the shortest scaffold/contig such that 50% of the genome is contained in contigs of this length or longer (after sorting scaffolds from the largest to shortest). It reflects assembly contiguity, with higher values indicating fewer, longer contigs. Higher this N50 value, better the contiguity.
L50: The number of contigs that contribute to the N50 length, showing how fragmented the assembly is. Lower this L50 value, better the contiguity.
Total length: The sum of all contig lengths, should be close to the expected genome size
Largest scaffold/contig: The length of the longest scaffold/contig, showing the maximum span of continuous sequence.
Number of scaffold/contigs: Indicates fragmentation; fewer contigs suggest better assembly.

GC content: The percentage of guanine and cytosine bases, reflecting base composition.
N content: The proportion of ambiguous bases (N) in the assembly, representing unresolved or low-confidence regions. High N content (e.g. >1%) indicates a fragmented or incomplete genome.
NOTE: a contig is a continuous sequence of DNA with no gaps, assembled from overlapping reads, while a scaffold is a series of contigs ordered and oriented relative to each other, often connected by gaps (typically represented by a series of N) where the sequence is uncertain.

To evaluate the quality of a reference assembly with Assemblathon:
```
export PERL5LIB=/home/ubuntu/Share/software/assemblathon2-analysis/
/home/ubuntu/Share/software/assemblathon2-analysis/assemblathon_stats.pl
[reference_genome.fasta]
```

i.e.
```
export PERL5LIB=/home/ubuntu/Share/software/assemblathon2-analysis/
/home/ubuntu/Share/software/assemblathon2-analysis/assemblathon_stats.pl
Qrob_PM1N.fa > Qrob_PM1N.fa.assemblathon.txt
```

Based on the Assemblathon results, how long is the assembled genome? Does its total length exceed the expected genome size for this species (0.75 Gb)? Additionally, is this assembly resolved at the chromosome level?

Step 2: Downloading sequencing data from public repositories like the SRA (Sequence Read Archive)

Sometimes, raw data relevant to your research questions are already available in public repositories, meaning there is no need to do a specific experiment and get some new data. This makes it crucial to know how to efficiently download these datasets. Public databases such as NCBI's Sequence Read Archive (SRA) and the European Bioinformatics Institute (EBI) host a wealth of genomic data that can be easily accessed. It is then possible to open fcp protocols and download the data. Knowing how to download these datasets ensures that you can make the most of existing resources and focus on advancing your research.
I consider that the EBI website is more convenient to use than the one of the NBCI. Anyway, to download oak data from Leroy et al. 2020 (New Phytology), visit the website (https://www.ebi.ac.uk/ena/browser/home) and search for PRJEB32209 in the search window at the top right. Then click on "View all 72 results", click on one of the experiment or run and select the name(s) of the fastq.gz files you would like to download and then "Get download script". The

information you will get from this archive
e.g. to download ERR3284873, the population (253 in Leroy et al. 2020), https://www.ebi.ac.uk/ena/browser/view/ERR3284873
Then, you can simply run the following command in your terminal:

```
wget -nc ftp://ftp.sra.ebi.ac.uk/vol1/fastq/ERR328/003/ERR3284873/ERR3284873_1.fastq.gz
wget -nc ftp://ftp.sra.ebi.ac.uk/vol1/fastq/ERR328/003/ERR3284873/ERR3284873_2.fastq.gz
```

PLEASE DO NOT EXECUTE this code today, as these fastq.gz files are very large!
Note: If you have started the download and wish to stop it, press Ctrl+C in your terminal. Ctrl+C is the command you should now in bash, it does not mean "copy", but "stop", it stops a job running on your console. Try to remind this, Ctrl+C has a totally different meaning in a command-line interface (shell), so do not use Ctrl+C to copy/paste!
Instead of processing extremely large files, today we will work with a limited dataset consisting of a few tens of thousands of sequencing reads (hereafter referred to as "subset"), you will find the data in the "Day1_bioinfo/data-oak/Raw-data" directory

Step 3: Quality control of raw sequencing data

One of the first steps when receiving sequencing data is to assess its quality using tools like FastQC and MultiQC.
FastQC provides a detailed report on various quality metrics of the raw data, such as base quality scores, GC content, adapter contamination, and sequence duplication levels. This helps identify potential issues that could affect downstream analyses.
After running FastQC on each sample, we often use MultiQC to aggregate and summarize the results from multiple FastQC reports into a single, comprehensive overview. MultiQC makes it easier to spot patterns and common issues across datasets, allowing us to make informed decisions on whether additional preprocessing, such as trimming or filtering, is needed before proceeding with mapping or further analysis (although we recommend performing read trimming in all cases). Evaluating the quality control is essential to have a rapid overview of the raw sequencing data, which is a first step to ensure the reliability and accuracy of subsequent genomic analyses.

Based on the subset, you could have run the following command to investigate the quality of the reads.

```
cd Raw-data/
for i in *subset.fastq.gz; do fastqc $i; done
multiqc *subset*
```

Download the files on your own computer (using scp, see above) and open the fastqc and the multiQC reports in your web browser to explore the results. Note that, given the computation time required to download the raw sequencing data, I have also provided the FastQC and MultiQC results for the full data (see ~/Day1_bioinfo/data-oak/Full_data/). Please note that both FastQC and MultiQC generate HTML reports, which can be only opened on your own computer.
Note: FastQC provides an estimate of the proportion of duplicates based on Kmer strategy, this proportion should not be overintepreted since the approach is expected to largely overestimated the proportion of duplicates, to have a more accurate estimate of duplicates, see step 7.

Step 4: Read trimming

Trimming is a critical preprocessing step in next-generation sequencing (NGS) data analysis that involves removing low-quality bases, adapters, and unwanted sequences (e.g. short sequences after trimming) from raw reads. This step ensures that only high-quality data is used for downstream analyses. Common issues addressed during trimming include sequencing errors at the ends of reads and residual adapter contamination, all of which can negatively impact mapping and bioinformatics pipelines, including variant detection.
Trimmomatic is one of the most widely used tools for read trimming due to its flexibility and effectiveness. It supports various trimming operations tailored to the specific needs of NGS datasets.

Some of the most important parameters in Trimmomatic:
ILLUMINACLIP: Removes adapter sequences using a predefined adapter file. Essential for eliminating contaminants from the sequencing process.
SLIDINGWINDOW: Applies a sliding window approach to trim reads once the average quality score in a specified window size falls below a threshold. Helps remove low-quality regions dynamically.
LEADING: Removes low-quality bases from the start of a read. Useful for fixing sequencing artifacts common at the 5' end.
TRAILING: Removes low-quality bases from the end of a read, where sequencing quality often deteriorates.
MINLEN: Discards reads shorter than a specified length after trimming, ensuring that only informative reads are retained.
CROP: Cuts reads to a specific length, useful for standardizing read lengths across a dataset.

Here an example:

```
cd ~/Day1_bioinfo/data-oak/Raw-data/
for j in ERR3284869 ERR3284873 ERR3284898; do
```

```
acc=$(echo "./$j")
outacc=$(echo "../Trimming/$j")
trimmomatic PE -threads 1 -phred33 "$acc"_1.subset.fastq.gz
"$acc"_2.subset.fastq.gz "$outacc"_1.cleaned.subset.fastq.gz
"$outacc"_1.cleaned_unpaired.subset.fastq.gz "$outacc"_2.cleaned.subset.fastq.gz
"$outacc"_2.cleaned_unpaired.subset.fastq.gz \
ILLUMINACLIP:/home/ubuntu/miniconda3/envs/Workshop_TL_YB_Calling/share/trimmomatic/adapters/
TruSeq3-PE-2.fa:2:30:10 LEADING:3 TRAILING:3 SLIDINGWINDOW:4:15
MINLEN:50
done
```

Note: The script is designed to save the results in a Trimming directory. Therefore, ensure you already have a Trimming directory in the data-oak directory (normally yes, but if not, please create it with the mkdir command: "mkdir Trimming"). Additionally, remember to navigate between directories using "cd [DIRECTORY]" (e.g. cd Trimming) to enter a folder, and "cd .." to move up one level in the directory structure!

Step 5: Mapping reads against a reference genome

Bowtie2 and BWA-MEM are both popular tools for mapping sequencing reads to a reference genome, each with its own strengths. In most cases, the two software relatively similarly perform and be used on the same sequencing data. We could however indicate that Bowtie2 is highly efficient and flexible, particularly for aligning short to medium-length reads, and offers strong performance in handling mismatches and indels. This explains why it was more favored for RNA-seq and metagenomic datasets where these features are critical.
BWA-MEM, on the other hand, is designed for longer reads, such as those generated by Illumina paired-end sequencing or third-generation platforms. It excels in accuracy for gapped alignments and complex genomic regions, making it a (bit more) preferred choice for whole-genome resequencing or structural variant studies.
In this tutorial, we will focus on BWA-MEM2, an enhanced version of BWA-MEM, which offers faster alignment speeds and reduced memory usage. While Bowtie2 is a strong alternative, BWA-MEM2 is particularly suited for the type of data we will be working with.

Generally, to speed up computations, software needs to index the reference genome. Note that not all software indexes the genome in the same way, so please make sure to check for potential errors in the logs when running new tools! With bwa-mem2, indexing the sequence is quite straightforward; however, it requires a significant amount of RAM. That's why all the indexes are already included in the same repository as the reference genome. (Note that the bwa-mem2 index command is preceded by a #, which silences it — meaning

that the command will not be executed here!):

```
cd ~/Day1_bioinfo/data-oak/
REFERENCE_GENOME=$(echo "Qrob_PM1N.fa")
#bwa-mem2 index $REFERENCE_GENOME
```

Then, perform mapping for each individual:

```
cd ~/Day1_bioinfo/data-oak/Trimming
for j in ERR3284869 ERR3284873 ERR3284898; do
acc=$(echo "$j")
outacc=$(echo "../Mapping/$j""_subset_trimmedPE")
bwa-mem2 mem ../$REFERENCE_GENOME "$acc"_1.cleaned.subset.fastq.gz
"$acc"_2.cleaned.subset.fastq.gz | samtools view -Sb - > $outacc.bam
done
```

Note: in this example, we will only worked on paired-end data after trimming, but it would be possible to use bwa-mem2 mem with a single fastq.file

OPTIONAL: The example below demonstrates how to map paired-end data after trimming, while excluding single-end reads remaining after trimming (i.e. "cleaned_unpaired" reads). Identify the appropriate command-line to map these single-end reads.

To optimize computational efficiency, most bioinformatics software requires reads to be sorted by their genomic positions. Raw sequencing reads are random fragments of DNA sampled across the entire genome and are mapped individually to the reference genome. Sorting these mapped reads reorganizes them in sequential order, from the first to the last position of the reference genome, facilitating faster and more efficient downstream analyses. Common tools for sorting BAM files include Samtools and Picard.
Here an example with picard:

```
cd ~/Day1_bioinfo/data-oak/Mapping
for i in *.bam; do
# Define paths and variables
INPUT_BAM="$i"
OUTPUT_SORTED_BAM="$i.sorted"
# Sort reads with SortSam
picard SortSam \
I=${INPUT_BAM} \
O=${OUTPUT_SORTED_BAM} \
SORT_ORDER=coordinate \
VALIDATION_STRINGENCY=LENIENT
done
```

Note: For large sequencing datasets, we strongly recommend deleting unnecessary files (e.g. unsorted BAM files) once they are no longer needed. This helps free up valuable storage space on your computing cluster.

Step 6: QC based on the results of mapping

Performing quality control after read mapping is a critical step to assess the success of the alignment process and ensure reliable downstream analyses. Tools like Samtools flagstat and Samtools idxstats provide valuable insights into the quality and distribution of mapped reads.

Samtools flagstat summarizes key alignment statistics, such as the total number of reads, the proportion mapped to the reference genome, and the fraction of properly paired reads in paired-end reads sequencing data. These metrics help evaluate alignment efficiency and identify potential issues, such as low-quality mappings or unexpected levels of unmapped reads. As you will see below, I recommend to regularly use
To use flagstat, execute the following command:

```
for i in *.sorted; do
samtools flagstat $i > $i.flagstat.txt
done
```

Use the function less or more e.g. "`less [file].flagstat.txt`" to explore the results.
OPTIONAL: Try to directly map the raw data (without trimming) and compare the results with flagstat. Do you observe some differences with or without trimming?

Samtools idxstats is another valuable tool to have an overview of the mapping. It provides per-chromosome statistics about mapped and unmapped reads relative to the length of each scaffold in the reference assembly. The output consists of four columns: scaffold name, scaffold length, the number of mapped reads, and the number of unmapped reads. However, it's important to note that the term "unmapped reads" in this context can be somewhat misleading - it refers to reads assigned to a chromosome but with low-confidence alignments, such as low-quality mappings or secondary alignments. By analyzing the idxstats output, researchers can assess genome coverage by dividing the number of reads by the scaffold length and identify potential issues, such as uneven coverage, due to over-represented contigs or contamination. This makes idxstats an essential step in ensuring the quality and reliability of mapped sequencing data.
To use idxstats, execute the following command:

```
for i in *.sorted; do
samtools index $i
samtools idxstats $i > $i.idxstats.txt
done
```

Step 7: Removing PCR duplicates

Marking PCR duplicates is another essential step in sequencing data processing, particularly when dealing with whole-genome sequencing. PCR amplification during library preparation can introduce duplicates from the same DNA fragment, which can bias downstream analyses. By marking duplicates, reads with identical start and end positions are flagged, helping to identify likely PCR artifacts without removing them. This preserves all data for flexibility in later analyses, as some tools can decide to exclude duplicates based on the specific research goal.

Picard MarkDuplicates is a widely used tool for this purpose. It identifies duplicates in BAM files, marks them with a specific flag (0x400), and generates metrics to quantify the duplication rate. This approach ensures that researchers retain all information while providing the option to filter duplicates when necessary.

However, it is important to note the importance of marking duplicates can vary depending on the experimental context. For instance, in restriction enzyme-based RAD-Seq data, particularly with single-digest restriction, the start positions of the reads can be identical across different fragments, even in the absence of PCR duplicates. This can be misleading as it may appear as if there are duplicates, when in fact, they represent distinct fragments from the same restriction site. Similarly, with single-end sequencing data, the start positions of reads from different fragments can also overlap, leading to potential false positives in duplicate detection. In these cases, whether marking duplicates should be performed can be evaluated, since filtering too much data based on duplication rates could also impact the downstream analysis. To summarize, for all whole-genome sequencing projects with paired-end sequencing data, we strongly recommend to mark duplicates. For some other data, the decision to mark or remove duplicates should always be made in the context of the specific experimental design and research objectives.
Here, due to practical constraints associated with truncating the data to generate a subset of reads to be manageable during the course, we add an command line with to filter the bam with samtools view before to run the Picard MarkDuplicates command in order to ensure to only have properly paired reads from our BAM files (with -f 0x2).

```
for i in *sorted; do samtools view -h -f 0x2 $i | samtools sort
-o $i.clean; done
```

Now, the picard script below is expected to properly work on your bams:

```
for i in *.bam; do
# Define paths and variables
INPUT_SORTED_BAM="$i.sorted.clean"
OUTPUT_BAM="$i.dedup"
METRICS_FILE="$i.duplication_metrics.txt"
TMP_DIR="bidontmp"
# Run Picard MarkDuplicates
```

```
picard MarkDuplicates </code>   I=${INPUT_SORTED_BAM} \
O=${OUTPUT_BAM} \
M=${METRICS_FILE} \
REMOVE_DUPLICATES=false \
ASSUME_SORTED=true \
VALIDATION_STRINGENCY=LENIENT
done
```

The MarkDuplicates output files (i.e. "duplication_metrics.txt" in the command above) are available in the "./data-oak/Full_data/" respository for both paired-end and single-end reads after trimming and provides key information about PCR duplicates in the sequencing data, including a histogram that shows the distribution of duplicate groups based on the number of reads in each group (e.g. pairs, triplicates). The most important information is the percentage of duplicates relative to the total number of duplicates detected and mapped reads. This percentage is crucial for assessing the extent of PCR duplication in your dataset. A high percentage of duplicates may indicate excessive PCR amplification during library preparation, which could bias downstream analyses, such as variant calling or coverage estimation. Monitoring this value helps determine if duplicate removal or marking is necessary for your analysis.

Compare the proportion of duplicates identified in the MarkDuplicates output with the corresponding values reported in the MultiQC report. Are the results consistent?
Do you expect any differences if you use samtools flagstat on the BAM files after deduplication?

Step 8: Allele counts & mpileup files for pool-seq data

Using mpileup for pool-seq data allows you to generate a compact representation of sequence data at each position across multiple pools, providing depth and allele information. It helps identify variation and detect the presence of specific alleles across different samples.

Here again, due to the constraints associated with the cluster, the commands below are only provided (**PLEASE DO NOT EXECUTE THEM, A FILE IS PROVIDED BELOW: see "Oak_3pools.mpileup.first100kb"**):

```
samtools mpileup -f ../../Qrob_PM1N.fa -q 20 -Q 30 \
ERR3284869ERR3284873_subset_trimmedPE.bam.dedup \
ERR3284873ERR3284873_subset_trimmedPE.bam.dedup \
ERR3284898ERR3284873_subset_trimmedPE.bam.dedup \
> ../Calling/Oak_3pools.mpileup
```

Explanations:
-q 20 corresponds to the minimum base quality score for including bases in the mpileup
-Q 30 corresponds to the minimum mapping quality for including reads in the mpileup

The mpileup output is expected to include the following columns: chromosome, position, reference allele (from the reference assembly), read depth for pool 1, sequenced alleles for pool 1 (where "." indicates that the sequenced allele is the same than the one on the reference genome), and base quality for pool 1. This is followed by the same information for pool 2, and then for pool 3, respectively.

```
Qrob_Chr01    1   T   0   *   *   1   ^8.      C       0   *   *
Qrob_Chr01    2   C   0   *   *   1   .        C       0   *   *
Qrob_Chr01    3   T   0   *   *   1   .        C       0   *   *
Qrob_Chr01    4   G   0   *   *   2   .^9.     F@      0   *   *
Qrob_Chr01    5   A   0   *   *   2   ..       F@      0   *   *
Qrob_Chr01    6   A   0   *   *   3   G.^8.    F@C     1   ^7. @
Qrob_Chr01    7   G   0   *   *   3   A..      FFC     3   ...
FF@
Qrob_Chr01    8   T   0   *   *   4   ...^8.   FD@@    3   ...
EFB
Qrob_Chr01    9   A   0   *   *   3   ...      HEF     3   ...
FHD
Qrob_Chr01    10  T   0   *   *   4   ....     HFF@    3   ...
HHF
```

Mpileup can generate very large files, one relevant strategy to summarize this information is to convert to a syn file from the Popoolation2 suite. When converted to a synchronized format, the format is optimized for handling large-scale pooled data, allowing for accurate comparisons between pools, controlling for coverage biases, and improving the accuracy of allele frequency estimates in diverse populations.

Note that it is very easy to extract depth of coverage from a mpileup, e.g. to know the coverage for the three pools at each position, you can write:

```
awk '{print $1"\t"$2"\t"$4"\t"$7"\t"$10}' Oak_3pools.mpileup.first100kb
> Oak_3pools.mpileup.first100kb.cov
```

Anyway, a mpileup can then be summarized in a synchronized mpileup with popoolation2, please use the command below:

```
mpileup2sync.pl --fastq-type sanger --input Oak_3pools.mpileup.first100kb
--output Oak_3pools.mpileup.first100kb.sync
```

Here is an example of the beginning of a synchronized mpileup file:

```
Qrob_Chr01    1   T   0:0:0:0:0:0 0:1:0:0:0:0 0:0:0:0:0:0
Qrob_Chr01    2   C   0:0:0:0:0:0 0:0:1:0:0:0 0:0:0:0:0:0
Qrob_Chr01    3   T   0:0:0:0:0:0 0:1:0:0:0:0 0:0:0:0:0:0
Qrob_Chr01    4   G   0:0:0:0:0:0 0:0:0:2:0:0 0:0:0:0:0:0
Qrob_Chr01    5   A   0:0:0:0:0:0 2:0:0:0:0:0 0:0:0:0:0:0
Qrob_Chr01    6   A   0:0:0:0:0:0 2:0:0:1:0:0 1:0:0:0:0:0
Qrob_Chr01    7   G   0:0:0:0:0:0 1:0:0:2:0:0 0:0:0:3:0:0
Qrob_Chr01    8   T   0:0:0:0:0:0 0:4:0:0:0:0 0:3:0:0:0:0
Qrob_Chr01    9   A   0:0:0:0:0:0 3:0:0:0:0:0 3:0:0:0:0:0
Qrob_Chr01    10  T   0:0:0:0:0:0 0:4:0:0:0:0 0:3:0:0:0:0
```

Note: For each pool, the format is A-count:T-count:C-count:G-count:N-count:-count, which represents the number of reads supporting the alleles A, T, C, G, ambigious allele (N) and indels (*).*

Can you identify potential mutations in this output? How accurate are the mutations listed above? Explore the file further to uncover additional mutations.

Once the data is formatted, it can easily be imported into the R package poolfstat using the function popsync2pooldata. This makes it easier to call variants and perform downstream analyses adapted for pool-seq data using the functions of this R package (e.g. PCA, F[ST] calculations, etc.). For further details on these steps, please refer to Friday's proposal.

Step 9: SNP calling and vcf files for calling genotypes (moderate coverage data)

Two softwares are particularly used for variant calling in moderate coverage sequencing (e.g. 10–30x): GATK and FreeBayes. In the section below, we will provide information about how to use GATK. In our case, GATK is highly suitable, since GATK is particularly ideal for population-scale studies. In brief, its HaplotypeCaller engine reconstructs haplotypes, improving accuracy for SNPs and indels, even in complex or repetitive regions. GATK's joint genotyping via GVCF files has good sensitivity across samples. It also supports quality recalibration, reducing errors and improving reliability (this will however not be covered today). GATK however requires relatively large computational resources, but the workflows are well-documented and is used by a broad community in the field.

Although GATK and Freebayes have different specificities. Compared to FreeBayes, GATK is a bit more precise in identifying variants in complex regions and multi-allelic sites. FreeBayes, however, is faster and simpler, making it useful for quick analyses. For projects demanding accuracy and robustness, especially

with moderate coverage, GATK remains the preferred choice, while FreeBayes is advantageous/more convenient for some specific data (e.g. non-diploid species) and when the availability of computational resources is more constraint.

Again, the project starts by indexing the reference

```
cd ~/Day1_bioinfo/data-datepalm/
REFERENCE=$(echo "GCF_009389715.1_palm_55x_up_171113_PBpolish2nd_filt_p_genomic.fna")
samtools faidx $REFERENCE
gatk CreateSequenceDictionary -R $REFERENCE
```

With GATK, it is essential to assign distinct Read Group information for each individual. The recommended approach is to use the AddOrReplaceReadGroups function. Specifically, the individual's name should be specified in both the RGID (Read Group ID) and RGSM (Read Group Sample) tags in the command below. This ensures proper identification and differentiation of samples during downstream analysis

```
cd Mapping
for i in *.bam; do
name=$(basename "$i" ".bam" | sed 's/subset_//g')
gatk AddOrReplaceReadGroups \
-I subset_"$name".bam  \
-O subset_"$name".RG.bam  \
--RGID "$name"  \
--RGLB lib1 \
--RGPL ILLUMINA \
--RGPU unit1 \
--RGSM "$name"
done
```

GATK requires that all bam are indexed. To index them, execute this command:

```
for i in *RG.bam; do
samtools index $i
done
```

With GATK, the first step is to use the HaplotypeCaller to generate a GVCF file for each individual. This intermediate file format allows for efficient joint genotyping across multiple samples in subsequent steps.

```
cd ~/Day1_bioinfo/data-datepalm/
for bam in subset_Dayri.RG.bam subset_Halawy.RG.bam subset_Hayany.RG.bam
subset_Khisab.RG.bam subset_Medjool.RG.bam; do
rootname=$(basename "$bam" ".RG.bam")
gatk HaplotypeCaller \
-R "$REFERENCE" \
-I ./Mapping/$bam \
-O ./Calling/$rootname.g.vcf.gz \
```

```
-ERC GVCF \
-L NC_052395.1:1-24100000
done
```
Note: In the code above, we used the -L option to target a specific region of a scaffold, which helps speed up computation. Without this option, the .g.vcf.gz files would be generated for the entire genome. Despite focusing on only a small region, the process still took several minutes to complete for the five individuals (Maybe it's a good time to take a little break?!).

The second step below involves using CombineGVCFs to merge the individual GVCF files into a single file, preparing the data for joint genotyping across all samples.

```
gatk CombineGVCFs -R $REFERENCE \
--variant ./Calling/subset_Dayri.g.vcf.gz \
--variant ./Calling/subset_Halawy.g.vcf.gz \
--variant ./Calling/subset_Hayany.g.vcf.gz \
--variant ./Calling/subset_Khisab.g.vcf.gz \
--variant ./Calling/subset_Medjool.g.vcf.gz \
-O ./Calling/datepalm.combine.NC_052395.1.g.vcf.gz
```

Then, use GenotypeGVCFs to perform joint genotyping on the combined GVCF file, producing a multisample VCF file with the identified variants across all individuals.

```
gatk GenotypeGVCFs \
-R $REFERENCE \
-V ./Calling/datepalm.combine.NC_052395.1.g.vcf.gz \
-O ./Calling/datepalm.NC_052395.1.jointvcf.gz
```

Lastly, apply VariantFiltration to filter the raw VCF file based on quality metrics (e.g. QD, FS) to retain high-confidence variants suitable for downstream analysis.

```
gatk VariantFiltration \
-R $REFERENCE \
-V ./Calling/datepalm.NC_052395.1.jointvcf.gz \
-O ./Calling/datepalm.NC_052395.1.jointvcf.filtered.gz \
--filter-expression "QD < 2.0 || FS > 60.0 || MQ < 40.0" \
--filter-name "LowQual"
```

QD means Quality by Depth and filters variants with low confidence relative to sequencing depth.
FS means Fisher Strand Bias and detect strand bias, consistent systematic error in sequencing reads.
MQ means Mapping Quality and ensures that calling is based on high-confidence alignments.

It is possible to use more filters, e.g. **MQRankSum** (Mapping Quality Rank Sum Test), **ReadPosRankSum** (Read Position Rank Sum Test), **SOR** (Strand Odds Ratio), **DP** (Read Depth), etc.

All outputs on this section (from individual g.vcf to final filtered variants) are based on the VCF format.
he VCF header begins with multiple lines starting with ##. These lines contain valuable metadata, detailing how to interpret the VCF file, including information about the commands and tools used in the pipeline, as well as annotations and parameters applied during variant calling.

```
##fileformat=VCFv4.2
##ALT=<ID=NON_REF,Description="Represents any possible alternative
allele not already represented at this location by REF and ALT">
##FILTER=<ID=LowQual,Description="Low quality">
##FORMAT=<ID=AD,Number=R,Type=Integer,Description="Allelic depths
for the ref and alt alleles in the order listed">
##FORMAT=<ID=DP,Number=1,Type=Integer,Description="Approximate
read depth (reads with MQ=255 or with bad mates are filtered)">
##FORMAT=<ID=GQ,Number=1,Type=Integer,Description="Genotype
Quality">
```

The most critical line of the VCF header is typically the last one, as it introduces the column names for the variant data. For example, in the final VCF file, this line includes standard fields such as #CHROM, POS, ID, REF, ALT, QUAL, FILTER, INFO, and FORMAT, followed by the names of the individual samples. This structure is essential for understanding and interpreting the variant information for each sample.
Using grep, it is easy to recover this information: `zmore datepalm.jointvcf.filtered.gz | grep "#CHROM"`

How many columns do you observe in your vcf? Does this correspond to your expectation?

How many low-quality ("LowQual") and high-quality ("PASS") variants are present in your VCF? To answer this, use tools like grep (=find) and wc -l (=count).

OPTIONAL: Count the number of homozygous and heterozygous genotypes for each individual, along with the number of missing calls.
Note: Historically, GenotypeGVCFs represented missing genotypes as a . in the VCF output (e.g. "./." calls for diploids). However, if you are using GATK versions 4.2.3.0 through 4.5.0.0, missing genotypes are indicated as 0/0 with an associated read depth (DP) of zero. This change caused confusion in the

community, leading GATK to revert to using dots for missing genotypes in subsequent versions.

Step 10: Genotype likelihood (low coverage data)

This section of the analysis presents additional challenges due to the significant computational load required to process the data. For this reason, the provided commands are included primarily as an indicative reference rather than for direct execution.

The use of ANGSD (Analysis of Next-Generation Sequencing Data) is particularly valuable in scenarios with low-coverage sequencing data. Traditional methods for population genomic analyses often rely on high-confidence genotype calls, which are difficult to obtain when sequencing depth is limited. ANGSD, however, allows analyses to be performed directly on genotype likelihoods instead of called genotypes. This approach leverages probabilistic models to handle uncertainties inherent in low-coverage data effectively.

In this section, we will perform genotype calling for a set of samples collected from green anole lizards inhabiting urban and non-urban environments. For further details and context, please refer to Thursday's practical session.

First, index the reference genome using the following commands:

```
cd ~/Day1_bioinfo/data-anole/
samtools faidx anoCar2.fa
gatk CreateSequenceDictionary -R anoCar2.fa
```

Then, generate a full list of samples:

```
cat subset_urban.list countryside.list > urban_countryside.list
sed 's;subset;./Mapping/subset;g' urban_countryside.list > urban_countryside.path.list
```

Now it is time to perform the calling with ANGSD

```
angsd -bam urban_countryside.path.list \
-ref anoCar2.fa \
-out ./Calling/urban_contryside_anole \
-GL 1 \
-doMajorMinor 1 \
-doMaf 1 \
-doVcf 1 \
-doPost 1 \
-pest ./Calling/urban_contryside_anole.sfs \
-doGlf 2 \
```

```
-doCounts 1 \
-P 2 \
-minMapQ 30 \
-minQ 20 \
-baq 1 \
-uniqueOnly 1 \
-remove_bads 1 \
-setMinDepth 3 \
-setMaxDepth 100
```

Since the command above will take approximately 10 minutes to execute, you have enough time to read through the important information provided below!

In the command above, the parameters correspond to:

-bam: specify the input BAM files for the calling **-out**: define the output file name prefix
-ref: specify the reference genome to use
-GL: define the methods to use to estimate likelihood calculation (1=samtools model,2=GATK model, etc).
-doMajorMinor: this function identifies the major (most frequent) and minor (less frequent) alleles at each site based on read counts and calculates their frequencies.
-doMaf: computes minor allele frequency for each site.
-doVcf: outputs a VCF with probabilistic genotype data based on genotype likelihoods and posterior probabilities.
-doPost: calculates posterior genotype probabilities by combining genotype likelihoods with prior allele frequency information.
-pest: provides the prior allele frequency spectrum used in posterior genotype estimation.
-doGlf: outputs other format for genotype likelihood files (see ANGSD documentation)
-doThetas: Estimates population-level nucleotide diversity (\theta).
-doCounts: Generates read depth and base counts at each site, which can be useful for evaluating the quality of the sequencing and variant calls.
-P: Specifies the number of threads to use for parallel processing.
-minMapQ: Filters out reads with a mapping quality below a given value (e.g. 30), commonly done to improve the reliability of variant calls.
-minQ: Sets the minimum base quality threshold to a given value (e.g. 20). Bases below this quality will be excluded from analysis to reduce errors.
-baq 1: Applies base alignment quality (BAQ) correction, which helps to correct for mismapped reads in regions with complex alignment patterns.
-uniqueOnly 1: Only considers uniquely mapped reads, excluding those that are ambiguously mapped.

-remove_bads 1: Excludes bad reads that have a low mapping quality or are otherwise unreliable.
-setMinDepth: Sets the minimum read depth required to call a variant. Variants with fewer than this number of reads are excluded.
-setMaxDepth 100: Sets the maximum read depth allowed at a site (prevent over-representation of specific regions).

It is convenient to explore the VCF
`zmore urban_contryside_anole.vcf.gz | head -30000 | tail -2`

Interpret the results for the two positions displayed in the VCF above (or any other selected positions from the VCF). What are the most likely genotype calls at these positions? Do the results appear logical and consistent with your expectations?

By using ANGSD, it is possible to estimate allele frequencies, compute diversity statistics, and even detect signals of selection while accommodating the limitations of low-depth sequencing. These capabilities make it an essential tool for population genomic studies where resources or biological constraints restrict sequencing coverage (see Thursday's practical session).