Hello everyone. Today I'm going to explain what genome assembly is, why it matters, and how modern bioinformatics tools reconstruct complete genomes from sequencing data. I'll also briefly cover annotation and some strengths, weaknesses, and future improvements.

A few important terms that appears in every genome assembly project:

Genome assembly is the computational process of reconstructing the entire DNA sequence of an organism from millions of short fragments called *reads*, produced by sequencing machines.

Because machines cannot read whole chromosomes, we generate many overlapping pieces. Assembly detects these overlaps, usually through graph-based algorithms, and merges them into longer sequences called **contigs**, then **scaffolds**, and eventually a genome draft. We use high-throughput sequencing technologies as Illumina, PacBio and Oxford Nanopore.

Assembly allows us to study genome structure, evolution, genetic variation, and functional genomics—but it requires careful computational work to deal with sequencing errors, repeats, and massive amounts of data.

We can assemble **de novo**, without a reference (from scratch), or use **reference-guided** assembly when a close genome already exists.

You have to make sure your methods are computationally repeatable and reproducible. Also it has to be findable, accessible, interoperable and reusable.
To ensure this, we use:

- **Containers** like Docker or Singularity, which guarantee identical software environments.

- **Workflow managers** like Nextflow or Snakemake for automated, traceable pipelines.

There are two main types of sequencing technologies:

- **Short reads** (Illumina):
  They are cheap and extremely accurate but often cannot resolve repeated sequences, so the assembly remains fragmented.

- **Long reads** (PacBio, Oxford Nanopore):
  They span large repetitive regions, produce long contigs, and allow chromosome-level assemblies, but have higher error rates and usually need polishing.

The best practice today is a **hybrid approach**, combining the accuracy of short reads with the continuity of long reads.

A standard genome assembly pipeline that enables reconstruction of most coding regions and gives a solid draft genome.

1. **Extraction:** this is the wet-lab part that goes to raw-data QC. We obtain DNA from the organism—choosing the right tissue and minimizing contamination is important and convert them into reads.

2. **Sequencing:** normally the appropiate third-generation sequencing technology, usually is Illumina for accuracy and long-read for continuity.

3. **Preprocessing:**

    ○ **Quality control** with FastQC

FastQC is a quality control tool for high throughput sequence data. We can do QC checks on raw sequence data BAM, SAM or FastQ files, provides a quick overview to tell which areas have problems,overall quality and presence of adapters or contaminants, and allows export results to an HTML report. The language is java and they show something like.

To study the quality, we also have some other tools or ways to do it: Metrics: N50, size, coverty.

BUSCO [Benchmarking Universal Single-Copy Orthologs] it's a tool used for conserved genes. It uses the OrthoDB database.

    ○ **Trimming and error correction** with tools like Cutadapt, Trimmomatic, Racon, or Pilon.

To fill missing bases and correct sequencing. Is a validation step where mis-assemblies can also be identified and corrected.

4. **Assembly:**
   Tools depend on the data:

    ○ Short-read assemblers: SPAdes, ABySS, SOAPdenovo

    ○ Long-read assemblers: Canu, Falcon

    ○ Hybrid assemblers: MaSuRCA

There are two major algorithmic strategies:

- **De Bruijn Graphs** for short reads:
  Reads are broken into *k-mers* (artificial division of the read) and represented as nodes of a graph. This method is fast but struggles with repeats and sequencing errors. (k=num nt included).

  We can search in Python with a binary research looking at the last 3 nt in the alphabetically ordered index.

  This assumes perfect matches, but real genomes differ due natural variation, so we have to do approximate matching.

  There are some genome assemblers.

- **Overlap-Layout-Consensus (OLC)** for long reads:
  Finds full overlaps between reads, builds contigs, and produces a consensus. This handles repeats well but requires more computation. There are somec programs (diapo).

Modern workflows often mix both, depending on the dataset.

5. **Post-processing:**

    ○ **Scaffolding**: using read pair information to connect contigs. May contain gaps but reflect chromosome order.

    ○ **Gap filling**: replacing N's with sequence, often using long reads

    ○ **Polishing**: correcting errors in the assembly with extra readings. This step helps to improve the local base accuracy in particular correction base and small indels, as well as some mis-assemblies caused by poor reads alignment.

6. **Evaluation:** this are also quality checks

    - **N50, genome size, coverage**
    - **BUSCO**
    - Mapping reads back to detect misassemblies
    - Fred score: every character have a number associated to a code (ASCII)-

This pipeline enables reconstruction of most coding regions and gives a solid draft genome.

Once we have a good assembly, we move to **annotation**, which assigns biological meaning to the sequence.

The steps are:

1. **Repeat masking**:
   Using tools like RepeatModeler and RepeatMasker to identify and hide repetitive elements that cause false gene predictions. It begins by identifying and masking repetitive sequences to avoid false positives, we usually build a specie-specific repeat library and mask them.

2. **Gene prediction:**

    - **Ab initio** tools like AUGUSTUS predict coding genes from the genome sequence. Can be used for baseline prediction of CDS.

    - **External evidence**—protein alignments, homology from related species, and especially RNA-seq—refines gene structure, splice sites, and UTRs.

3. **Evidence integration:**
   Tools like **MAKER** or **PASA** combine predictions into final gene models.

4. **Quality control & manual curation:**
   Visual inspection with tools like WebApollo ensures there are no fused genes, intron retention, or missing exons. Manual curation and visual inspection are still important.

5. **Publication:**
   The final genome, annotation, raw reads, and metadata are submitted to repositories such as **NCBI BioProject** so others can reuse or improve them.

Strengths:

- Modern assemblers reconstruct most coding regions accurately.

- Hybrid workflows handle huge NGS datasets efficiently.

Weaknesses:

- Short-read assemblies remain fragmented.

- Assemblies highly depend on coverage, sequencing quality, and parameters.

- Some difficult regions—repeats, GC-rich segments—remain unresolved.

Future improvements:

- Increasing long-read input to resolve repeats.
- Tuning k-mer sizes and error correction.
- Using more advanced QC and polishing steps.
- Integrating new assemblers and graph-based technologies.

In summary, genome assembly is a powerful computational process that transforms raw sequencing reads into biologically meaningful genomes. With the combination of short and long reads, proper quality control, modern algorithms, and reproducible workflows, we can now produce high-quality assemblies for model and non-model organisms alike.