ChatGPT

Hybrid Sequencing Coverage Study Design

Hybrid genome assembly – combining short Illumina reads with long PacBio reads – is a powerful way to resolve repeats and improve assembly continuity. Short reads are very accurate but too short to span many repeats, yielding highly fragmented assemblies [1]. Long reads (PacBio) can span repeats but have higher error; newer PacBio HiFi reads are long (10–20 kb) and ~99% accurate [2]. A typical strategy is to **assemble short reads into accurate contigs** and then use long reads to scaffold them [3]. Our goal is to **simulate various genomes and read sets** to find the minimal Illumina and PacBio depths needed for a "good" hybrid assembly under different conditions.

Genome Simulation

We first create synthetic reference genomes with controllable repeats and features. One approach is to use existing genomes (e.g. bacterial or small eukaryotic genomes from NCBI) and copy or duplicate segments to simulate repeats. A dedicated tool, **Simulome** [4], can generate pseudo-genomes and introduce duplications: it selects regions (genes and flanking sequences) and duplicates them to reach a specified percentage of "repeat" content [4]. For example, setting a 10% duplication would copy regions totaling 10% of the genome. By varying the duplication fraction, one can simulate genomes with low, medium, or high repeat content. Simulome (Python-based) can output both the reference FASTA and annotation; it also supports SNPs, indels, etc. [5] [4].

Alternatively, one could write a simple Python/R script to take a known genome FASTA and splice in extra copies of certain sequences (tandem repeats or dispersed duplicates). If working on Mac or Linux, Python with BioPython (used by Simulome) is convenient. Ensure that the "simulated reference" genome is reasonably sized (e.g. bacterial ~5 Mb, yeast ~12 Mb, or up to tens of Mb) depending on HPC resources, as very large eukaryotic genomes will require substantial compute to assemble. By simulating **different genome sizes and complexities**, we can test how requirements scale with genome length and repeat content.

Read Simulation

Next, we simulate sequencing reads from the synthetic genomes at various depths. For Illumina short reads, widely used simulators include **ART** [6] and **wgsim**. ART (Huang *et al.* 2012) emulates Illumina sequencing error profiles and outputs FASTQ reads [6]. We would specify read length (e.g. 150 bp paired-end), insert size, and desired coverage. For PacBio long reads, one can use **PBSIM** [7] or **SimLoRD**. PBSIM (Ono *et al.* 2013) models PacBio read length and error profiles, generating both Continuous Long Reads (CLR, high error) and Circular Consensus (CCS/HiFi) reads [7]. In fact, PBSIM authors showed that combining ~15× CLR plus ~30× CCS reads yielded excellent assemblies [8]. (In our simulations, we could emulate PacBio HiFi by using CCS mode or a high-accuracy model.)

For each synthetic genome, we would generate read sets at multiple depths. For example: Illumina at 30×, 60×, 100× and PacBio at 10×, 20×, 40× (or finer increments). We could also explore different Illumina read

lengths or insert sizes. For simplicity, start with paired-end Illumina 150 bp and PacBio HiFi reads (~15 kb). Using Python or R scripts, we can automate the simulation runs (tools like ART and PBSIM have command-line interfaces). The simulated reads will be in FASTQ format, which can be quality-trimmed (e.g. with Trimmomatic) if needed, although for clean simulation they start error-free apart from built-in error models.

Assembly and Coverage Analysis

With simulated reads in hand, we perform de novo assembly under various hybrid conditions. Two main strategies exist:

- **Short-read-first (SPAdes/Unicycler)**: assemble Illumina reads into contigs (using an assembler like SPAdes), then scaffold with long reads [3]. Tools like **Unicycler** automate this: they build an initial de Bruijn graph from short reads and then align long reads to resolve the graph [3] [9]. Unicycler has been shown to produce highly accurate bacterial assemblies using few long reads, as it constrains scaffolding by the accurate short-read graph [9].
- **Long-read-first**: assemble (and error-correct) long reads into contigs, then polish with short reads (e.g. using Canu or Flye + Pilon). This often requires higher long-read coverage before polishing.

We would likely test one or both approaches. For example, run SPAdes or Unicycler in hybrid mode (both accept Illumina+PacBio). Assembling multiple large datasets may require HPC resources (multi-threading and ample RAM).

Assembly evaluation: For each assembly, measure metrics like contig N50, number of contigs, and completeness (e.g. using QUAST or BUSCO). These metrics should be collected for each coverage combination. We would then plot metrics vs. coverage: for instance, contig N50 as a heatmap or 3D surface over Illumina and PacBio depth. We expect curves of diminishing returns: e.g. N50 rises steeply with PacBio coverage up to ~20–30× (HiFi), then plateaus (consistent with Zhang *et al.* who found >20× HiFi suffices for yeast) [10]. Similarly, too little Illumina coverage will leave small local errors or gaps. By analyzing when metrics "saturate," we identify the **minimal depths** needed. For example, previous benchmarks suggest ~20–30× PacBio HiFi is often enough for microbial/eukaryotic genomes [10], whereas older continuous reads needed ~50×–100× [11].

We can also analyze the **assembly graph** itself (using tools like Bandage) to see how increasing coverage resolves complex nodes (repeats). For instance, low coverage may leave a tangled graph around repeats; at higher coverage, long reads span those nodes, simplifying the graph. Quantifying graph connectivity (e.g. number of tips or unresolved repeats) vs. coverage can help determine thresholds.

Variable Scenarios

To be thorough, we should vary several factors:

- **Genome size/complexity:** Test small bacterial genomes (~5 Mb), mid-size (yeast ~12 Mb), and larger eukaryotes (e.g. a plant genome segment ~100 Mb). Larger genomes will require more data and CPU, but illustrate scaling.
- **Repeat content:** Use genomes with few repeats vs. many (e.g. add extra duplications via Simulome). Highly repetitive genomes will require more long-read coverage to span repeats [1].

- **Read error profiles:** Simulate both high-quality HiFi reads and noisier CLR reads. The required coverage differs: e.g. HiFi (~99% accurate) may work with ~20× [10], whereas noisy CLR might need ~50× for good assembly [11].
- **Organism differences:** One could simulate prokaryotes (haploid, small) versus diploid eukaryotes. Diploidy/heterozygosity complicates assembly (leading to separate contigs for divergent alleles). For a first study, sticking to haploid or highly inbred models avoids this complexity.
- **Illumina/PacBio mix:** Evaluate pure Illumina, pure PacBio, and various hybrids. Pure Illumina will generally fail to assemble through large repeats [1]; hybrids should perform better. The question is: *How many reads of each type do we need?* For example, one might find that above ~30× Illumina + 20× PacBio HiFi gives near-maximum contiguity, with more data offering little gain.

Each scenario is an experimental condition. The data can be summarized in tables (e.g. N50 vs. Illumina/PacBio coverage) and figures (contiguity vs. coverage curves). This will form the core results of the paper.

Software and Tools

Key software choices (all Linux/Mac-compatible):

- **Genome simulation:** *Simulome* (Python) [4] for pseudo-genomes with specified repeats. Alternatively, custom Python/Biopython scripts.
- **Read simulation:** *ART* for Illumina reads [6], *PBSIM* for PacBio reads [7]. (Other options: `wgsim` or `dwgsim` for Illumina; `SimLoRD` or *PBSIM3* for long reads.)
- **Assembly:** *SPAdes* (with `--pacbio` option) or *Unicycler* [3] for hybrid assembly; or *Flye/Canu* for long-read-first assembly. Polishing short-read-first assemblies may not need extra polishing, but long-read assemblies should be polished with *Pilon* or *Racon*.
- **Evaluation:** *QUAST* to compare assemblies to the known reference; *BUSCO* for gene completeness metrics.
- **Graph analysis:** *Bandage* to visualize assembly graphs, if desired.
- **Workflow:** Orchestrate simulations and assemblies via shell scripts, Snakemake, or Python. Use multi-threading and possibly cluster jobs, since assemblies (especially at high coverage) can be CPU-intensive.

All these tools can run on Mac or Linux (HPC). Python and R are useful for data handling and plotting results. For example, one can use Python (pandas, matplotlib) or R to parse QUAST output and generate figures of metrics vs. coverage.

Recommendations and Considerations

- **Coverage benchmarks:** Based on literature, start with ~30–50× Illumina and ~20–50× PacBio as test points. Zhang *et al.* (2022) found yeast needed >20× HiFi and >80× ONT for high quality [10]. Hamada *et al.* suggested ~15× CLR + 30× CCS (HiFi) for PacBio [8]. These are rough guides.
- **Repeat analysis:** High repeat content genomes will shift the requirements upward. We should document how increasing duplication (e.g. 10%, 30%, 60% repeats) affects the minimal coverage needed.
- **Metrics of success:** Define “good coverage” as achieving near-complete assemblies (e.g. >95% genome in large contigs, high BUSCO scores) with no major gaps. If simulated reads include known variants, measure variant recall too.

- **Replicates:** Simulations can have random components (read sampling, Simulome randomness), so replicate key scenarios to assess variability.
- **Reporting:** The final report should detail the pipeline (possibly with a flowchart), list software and versions, and present results as graphs/tables. All claims (e.g. "PacBio bridges repeats" or "20× HiFi suffices") should cite sources – for example, Unicycler's publication [3], PacBio coverage blogs or papers [12] [10], and our simulation tools references [4] [6] [8].

In summary, this study involves iterating through simulation (genomes, reads), assembly, and evaluation under varied conditions (genome size, repeat level, coverage). By analyzing how assembly quality metrics improve with more Illumina or PacBio data, we can identify minimal coverage requirements. Using established tools for simulation (Simulome, ART, PBSIM) and assembly (SPAdes/Unicycler), and examining results with QUAST/BUSCO, will yield an informative set of guidelines. This approach has been used implicitly in prior studies [3] [8] [10], but our controlled simulation will allow a systematic quantification across scenarios.

Sources: Established tools (Simulome [4], ART [6], PBSIM [8]) and studies on hybrid assembly were used as references. For example, Unicycler's paper demonstrates hybrid assembly methods [3] and benchmarks on yeast provide coverage targets [10]. PacBio's own resources note that long reads can span repeats that short reads cannot [1] and that ~20× PacBio HiFi can rival much higher ONT coverage [12]. These guide our simulation design and expectations.

---

[1] [2] Evaluating Illumina-, Nanopore-, and PacBio-based genome assembly strategies with the bald notothen, Trematomus borchgrevinki - PMC
https://pmc.ncbi.nlm.nih.gov/articles/PMC9635638/

[3] [9] Unicycler: Resolving bacterial genome assemblies from short and long sequencing reads - PMC
https://pmc.ncbi.nlm.nih.gov/articles/PMC5481147/

[4] [5] Simulome: a genome sequence and variant simulator - PMC
https://pmc.ncbi.nlm.nih.gov/articles/PMC5870732/

[6] ART: a next-generation sequencing read simulator - PMC
https://pmc.ncbi.nlm.nih.gov/articles/PMC3278762/

[7] [8] PBSIM: PacBio reads simulator—toward accurate genome assembly | Bioinformatics | Oxford Academic
https://academic.oup.com/bioinformatics/article/29/1/119/273243

[10] Benchmarking of long-read sequencing, assemblers and polishers for yeast genome | Briefings in Bioinformatics | Oxford Academic
https://academic.oup.com/bib/article/23/3/bbac146/6576452

[11] Ideal coverage required for PacBio error correction using HGAP
https://www.biostars.org/p/135851/

[12] Sequencing 101: Sequencing coverage - PacBio
https://www.pacb.com/blog/sequencing-101-sequencing-coverage/