Autocycler: long-read consensus assembly for bacterial genomes

Ryan R Wick[1,2,*] **Benjamin P Howden**[1,2] **and Timothy P Stinear**[1,2]

[1]Department of Microbiology and Immunology, The University of Melbourne at the Peter Doherty Institute for Infection and Immunity, Melbourne, Victoria, Australia and [2]Centre for Pathogen Genomics, The University of Melbourne, Parkville, Victoria, Australia

[*]Corresponding author. rrwick@gmail.com, ryan.wick@unimelb.edu.au

Abstract

Motivation: Long-read sequencing enables complete bacterial genome assemblies, but individual assemblers are imperfect and often produce sequence-level and structural errors. Consensus assembly using Trycycler can improve accuracy, but its lack of automation limits scalability. There is a need for an automated method to generate high-quality consensus bacterial genome assemblies from long-read data.

Results: We present Autocycler, a command-line tool for generating accurate bacterial genome assemblies by combining multiple alternative long-read assemblies of the same genome. Without requiring user input, Autocycler builds a compacted De Bruijn graph from the input assemblies, clusters and filters contigs, trims overlaps and resolves consensus sequences by selecting the most common variant at each locus. It also supports manual curation when desired, allowing users to refine assemblies in challenging or important cases. In our evaluation using Oxford Nanopore Technologies reads from five bacterial isolates, Autocycler outperformed individual assemblers, automated pipelines and other consensus tools, producing assemblies with lower error rates and improved structural accuracy.

Availability and implementation: Autocycler is implemented in Rust, open-source and freely available at `github.com/rrwick/Autocycler`. It runs on Linux and macOS and is extensively documented.

Key words: Genome assembly, Long-read sequencing, Bacterial genomics, Oxford Nanopore Technologies

1. Introduction

Complete genome assemblies are essential for resolving bacterial genome structure and fully characterising accessory elements such as plasmids and prophages[1,2]. Accurate assemblies reduce the risk of errors in downstream analyses such as comparative genomics, annotation and studies of genome dynamics.

Long-read sequencing platforms, such as those from Oxford Nanopore Technologies (ONT), have made complete assemblies of bacterial genomes widely achievable. Long reads can span repetitive elements, allowing assemblers to resolve structural complexity that short reads (e.g. from Illumina platforms) cannot[3]. For most bacterial genomes and high-quality read sets, long-read assemblers can assemble each replicon into a single contig[4].

In practice, however, long-read assemblers are imperfect, and different tools produce different assemblies from the same input read set. Common problems include: incomplete or overlapping circularisation, missing small plasmids, duplicated small plasmids and spurious extra contigs from repeats or contamination[5,6]. No single assembler is reliably the best across all datasets.

Consensus assembly offers a solution. By combining multiple alternative assemblies of the same genome (e.g. those produced by different assemblers or read subsets), consistent sequences can be distinguished from assembler-specific errors[7,8]. The software Trycycler put this idea into practice for bacterial genomes[9]. Compared to assemblies produced by a single tool, Trycycler assemblies usually contain fewer errors, more reliable circularisation and a more complete and less contaminated representation of the genome[9].

Trycycler, however, relies on human interventions and decision-making for several key steps. While this design offers flexibility and control, it limits scalability. As bacterial genomics increasingly involves large datasets of hundreds or thousands of genomes, there is a need for automated methods that can generate high-quality consensus assemblies without manual interventions.

Here we present Autocycler, an automated command-line tool to generate consensus long-read assemblies of bacterial genomes. Like Trycycler, it combines multiple input assemblies to produce a high-quality consensus. Unlike Trycycler, Autocycler is designed to run to completion without user input. It also supports manual intervention for cases where careful output curation is warranted. For most bacterial genomes and read sets with sufficient depth and read length, Autocycler can produce complete assemblies automatically. In more difficult cases, such as genomes with large repeats, genomic heterogeneity or unusual structures like linear replicons, users can step in to refine the output.

2. Implementation

Autocycler constructs a consensus bacterial genome assembly by combining multiple alternative assemblies of the same genome (Figure 1). It is designed to run automatically and produces intermediate files and metrics for every step, allowing the process

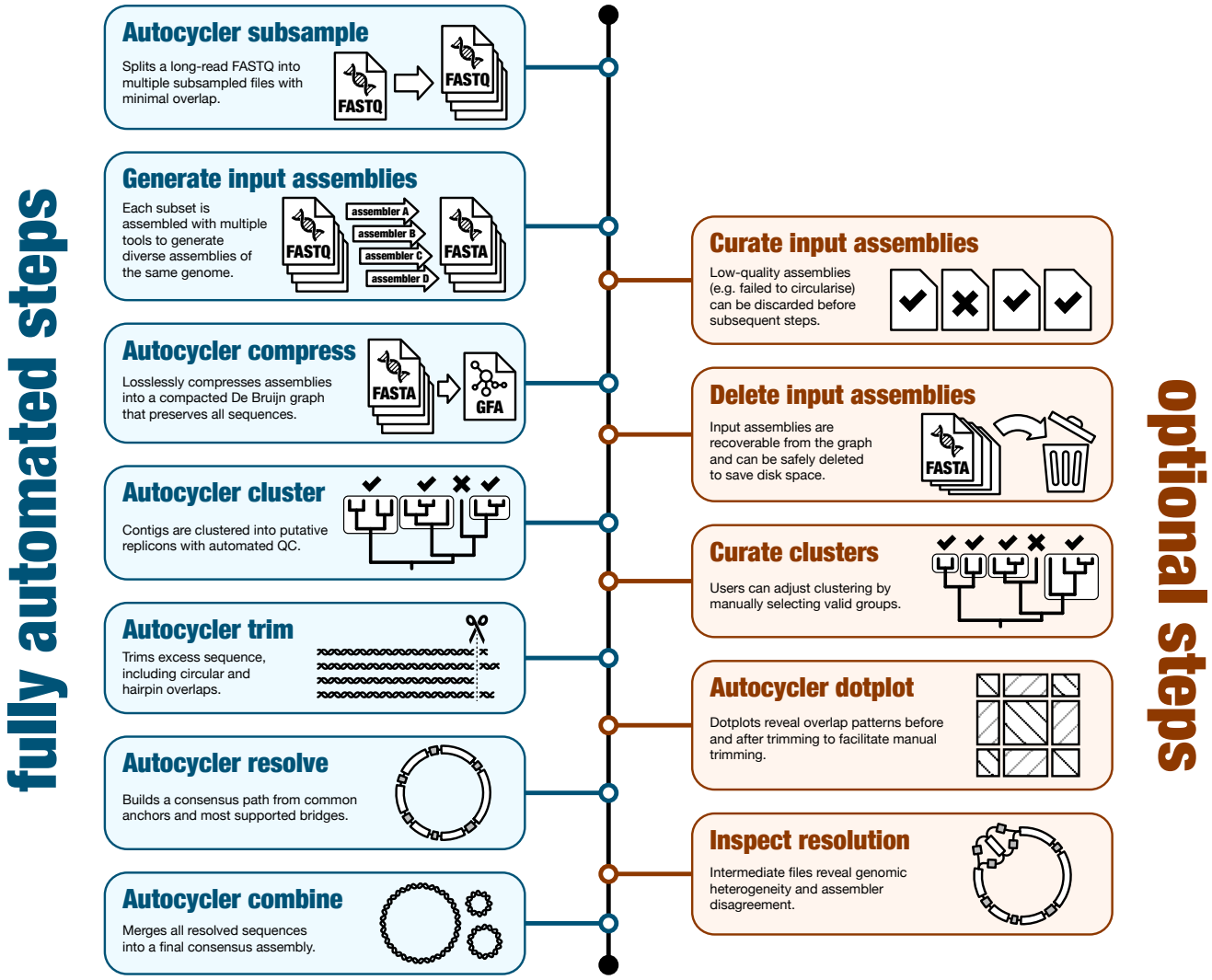

Fig. 1: Overview of the Autocycler workflow. By following only the blue steps on the left, Autocycler can produce consensus genome assemblies with no human intervention. The optional orange steps on the right can be used when accuracy is critical or when Autocycler's metrics (gathered using the `autocycler table` command) indicate potential issues.

to be inspected or curated if needed. In addition to the main consensus assembly workflow, Autocycler includes commands to assist with upstream and downstream tasks.

The `autocycler subsample` command creates read subsets for generating input assemblies. By dividing a single long-read set into minimally overlapping subsets, users can generate assemblies that are more independent of one another. Additionally, the subsampling process reduces very high-depth read sets to medium-depth subsets, which often assemble more cleanly. Users can choose any number of subsets, but we recommend four (the default), which avoids excessive overlap between subsets while still enabling a diverse set of input assemblies.

Input assemblies should be generated using a range of long-read assemblers, as this diversity improves the robustness of the final consensus. For example, assembling four read subsets with eight different assemblers would yield 32 alternative assemblies of the same genome as input for Autocycler. The `autocycler helper` command provides a simple wrapper for several common tools. Ideally, input assemblies will have each replicon in the genome (e.g. chromosome or plasmid) assembled as a single contig. While some fragmentation is tolerated, Autocycler relies on the assumption that most input assemblies are complete. If all input assemblies are fragmented, Autocycler will not be able to produce a complete consensus. For challenging genomes, users may optionally curate the input assemblies (e.g. discard or repair incomplete contigs) before proceeding.

The `autocycler compress` command builds a compacted De Bruijn graph from the input assemblies. This graph substantially reduces disk usage, as shared regions across assemblies are collapsed together. Each input contig is recorded as a path through the graph, preserving its full sequence and allowing reconstruction with `autocycler decompress`. The graph representation provides an efficient structure for comparing and manipulating contigs in subsequent steps, such as clustering and trimming, since their graph paths can be compared directly without the need for sequence alignment.

In the `autocycler cluster` step, pairwise distances between contigs are calculated based on the overlap of their graph paths. These distances are used to build a UPGMA tree, which is divided into initial clusters by applying a fixed distance cutoff. Autocycler then refines these clusters by testing whether splitting them improves clustering balance (each input assembly contributing

a single contig per cluster) and tightness (low intra-cluster distances). Quality control filters remove low-confidence clusters, such as those found in too few assemblies (likely spurious contigs) or those contained within other clusters (likely fragmented contigs or repetitive elements). The final result ideally includes one high-confidence cluster per replicon in the genome. Users may override the default clustering by inspecting the UPGMA tree and manually choosing which clusters to use. This can be helpful in challenging cases where input assemblies are inconsistent, for example to recover a small plasmid present in only a small number of inputs.

For each cluster, `autocycler trim` processes the input contigs to remove unwanted sequence. It looks for both circular overlaps, where the start of a contig overlaps with its end, and hairpin overlaps, where the start or end of a contig extends past the hairpin to the opposite strand. It also handles cases where small plasmids are fully duplicated within a single contig. After trimming, sequences with lengths that deviate too far from the cluster median are discarded, leaving a set of consistent contigs for consensus generation. Users may optionally run `autocycler dotplot` before and after trimming to visualise structural features and assess trimming outcomes.

The `autocycler resolve` command generates a consensus sequence for each cluster. It begins by identifying anchors: sequences that appear exactly once in each contig. These anchors serve as a scaffold for constructing bridges, which represent the most common paths between anchors in the input contigs. Autocycler first applies unambiguous bridges and then iteratively resolves ambiguous cases by selecting the most supported paths, ideally producing a single consensus sequence for the cluster. In cases of structural heterogeneity, such as phase-variable loci or assembly inconsistencies, Autocycler includes intermediate output for optional user inspection. When a cluster fails to fully resolve, the user can inspect its graph in Bandage[10] and use the `autocycler clean` command to manually remove or duplicate segments, often useful for the ends of linear plasmids.

Once each cluster has been resolved, the `autocycler combine` command merges them into a final consensus assembly in both FASTA and GFA formats. Autocycler also produces detailed metrics at every step of the pipeline, saved in YAML format (both human- and machine-readable). The `autocycler table` command can be used to gather metrics from many assemblies, making it easy to track success and identify samples that require further attention.

Compared to its predecessor Trycycler, Autocycler is designed to run to completion without user intervention. Two steps in the Trycycler pipeline typically require manual input: defining clusters and reconciling sequences into a consensus. Autocycler improves the former by automatically identifying clusters using UPGMA and quality control heuristics. For the latter, it replaces Trycycler's sequence-alignment-based reconciliation (which often failed in the presence of low-quality contigs) with a more robust De Bruijn graph-based approach. Even for steps that did not require user input in Trycycler, Autocycler is faster due to its efficient data structures and algorithms. Since Autocycler also supports manual curation when needed, it is now the recommended tool for long-read consensus bacterial genome assembly.

Autocycler is implemented in Rust, deterministic and resource-efficient. Most of the computational time in an Autocycler workflow is spent generating input assemblies. Autocycler itself typically completes in minutes and requires only modest resources. It runs on both Linux and macOS, although Linux is preferred due to broader compatibility with long-read assemblers. Extensive documentation, including illustrated examples and guidance for manual curation, is available online at `github.com/rrwick/Autocycler/wiki`.

3. Evaluation

3.1. Methods

Long-read sequencing of 84 diverse bacterial isolates was performed using an Oxford Nanopore Technologies PromethION 2 Solo with the Rapid Barcoding Kit 96 V14 (SQK-RBK114.96). Reads were basecalled with Dorado v0.9.5[11] using the `sup@v5.0.0` model and filtered to retain reads with mean quality $\geq$10. Five isolates were selected, each from a different genus: *Enterobacter hormaechei*, *Klebsiella pneumoniae*, *Listeria innocua*, *Providencia rettgeri* and *Shigella flexneri*. Short-read Illumina sequencing was available for all samples and was used to polish the reference genomes (Table S1). Selection was based on high read depth (>300×) and preliminary assessments showing no evidence of heterogeneity or divergence between the Illumina and ONT datasets.

For each genome, we followed our previously published method to generate a high-accuracy reference assembly[12]. Briefly, the ONT reads were assembled with Trycycler v0.5.5[9] and the resulting genome was polished using Medaka v2.0.1[13], Polypolish v0.6.0[14] and Pypolca v0.3.1[15]. The resulting assemblies were highly accurate and used as ground truth.

ONT reads for each genome were divided into six non-overlapping 50× subsets for a total of 30 read sets. Each read set was assembled using the following long-read assemblers: Canu v2.3[16], Flye v2.9.5[17], hifiasm v0.25.0[18], LJA v0.2[19], metaMDBG (a.k.a. nanoMDBG) v1.1[20], miniasm v0.3[21], Myloasm v0.1.0[22], NECAT v0.0.1[23], NextDenovo v2.5.2[24], Raven v1.8.3[25] and wtdbg2 v2.5[26]. Each of these tools was run via the `autocycler helper` command, which included low-depth contig removal and extra processing for Canu (overlap-trimming and repeat/bubble removal), miniasm (polishing with Minipolish[27]) and NextDenovo (polishing with NextPolish[28]).

In addition, we assembled each read set with the Dragonflye v1.2.1[29] and Hybracter v0.11.2[30] long-read assembly pipelines and the consensus assembly tool MAECI[31] (commit `f1eb3d7`). For Autocycler v0.5.1, we produced an automated assembly (using its `autocycler_full.sh` script) and a manually curated assembly. All assembly commands were run through GNU Time to quantify runtime and memory, using 32 threads on a system with dual AMD EPYC 7742 CPUs and 503 GB RAM.

We also attempted to evaluate MAC2.0[32], but it did not perform correctly on complete bacterial genomes, producing outputs with duplicated sequences. Other consensus assembly tools, such as quickmerge[33] and Metassembler[8], are older and were primarily designed to improve contiguity in fragmented eukaryotic assemblies. These tools are not suitable for refining complete bacterial genomes from long-read data and were therefore excluded from this comparison[34].

Each assembly was compared to its corresponding ground-truth reference using a custom script (`assess_assembly.py`) that aligns the assembly to the reference sequence with minimap2 v2.28[35] and quantifies accuracy metrics including sequence errors (substitutions and indels), missing bases and extra bases. We also assessed assembly accuracy with Inspector v1.3.1[36] and CRAQ v1.0.9[37], which evaluate assemblies based on read alignments rather than a reference, and BUSCO v6.0.0[38], which evaluates assemblies based on the presence of expected single-copy genes.

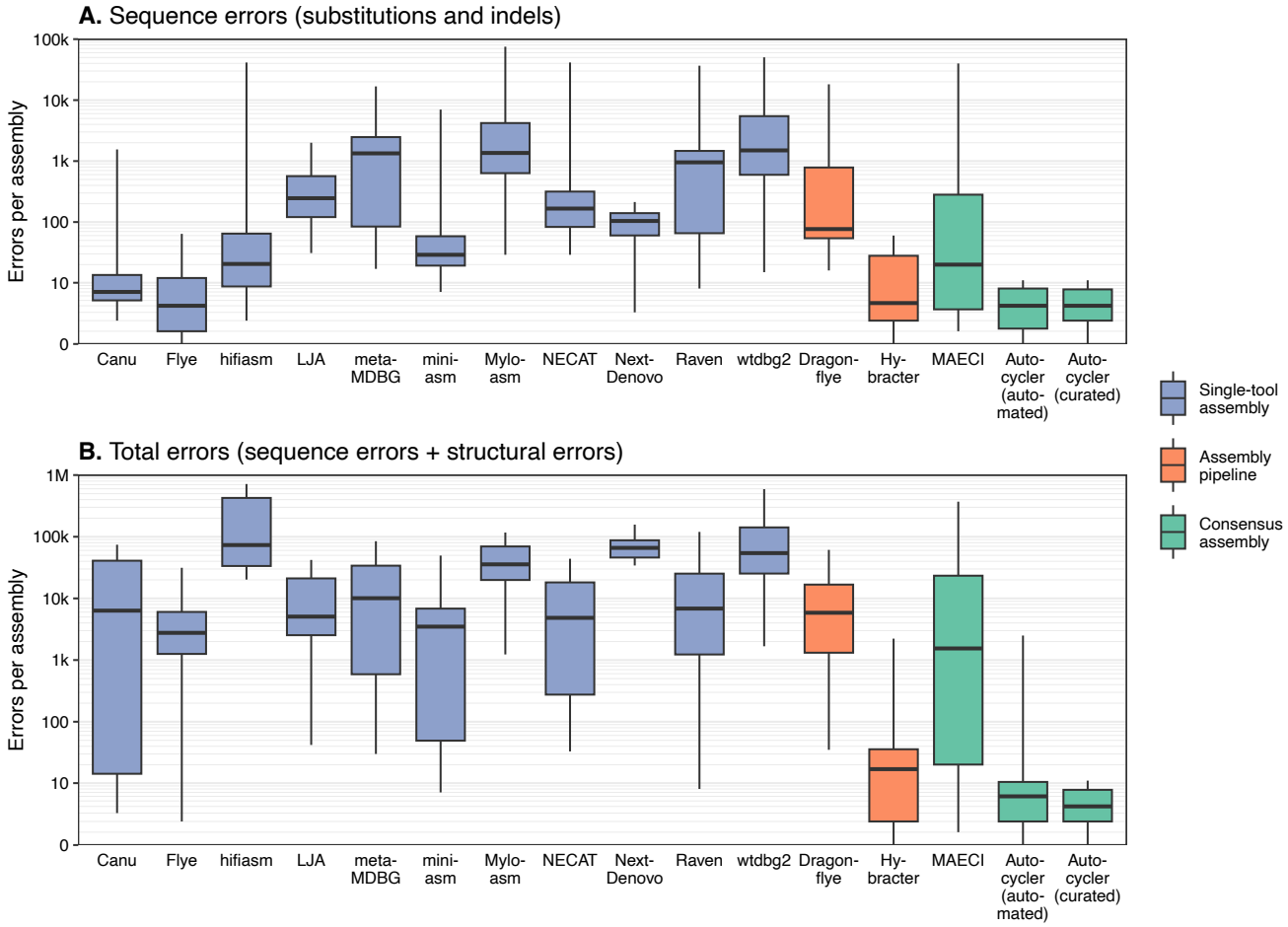

Fig. 2: Assembler benchmarking results from the `assess_assembly.py` script. **(A)** Sequence errors (substitutions and indels); **(B)** Sequence errors and structural assembly errors (missing and extra bases). Lower values indicate better accuracy. Results are coloured by category: individual long-read assembly tools (blue), long-read assembly pipelines (orange) and consensus assembly tools (green). Autocycler results are shown separately for automated and manually curated assemblies. Boxplot whiskers extend to the minimum and maximum values. The y-axes use a pseudo-logarithmic scale that accommodates zeros. See Table S2 and Figure S1 for results broken down by error type and Table S3 for mean values per assembler.

To evaluate Autocycler's performance on challenging datasets, we performed two additional tests. In the low-depth test, the *L. innocua* genome was assembled across a range of read depths (1× to 50×) and assessed using the `assess_assembly.py` script. In the mixed-genome test, we combined reads from the *E. hormaechei* and *K. pneumoniae* genomes at varying ratios (total depth fixed at 100×) and evaluated the resulting assemblies based on contig completeness and genome of origin. Full commands for all analyses are provided in the supplementary data.

3.2. Results

Among the single-tool assemblers, Canu and Flye consistently produced the fewest sequence-level errors, typically with fewer than 10 substitutions and indel errors per assembly (Figure 2A, Figure S1, Table S2). All other long-read assemblers had higher error rates. Across all single-tool assemblers, structural inaccuracies were common (Figure 2B, Figure S1), with assemblies often missing genomic elements (e.g. small plasmids) or containing spurious extra sequence (e.g. duplicated ends of circular contigs).

Of the long-read assembly pipelines, Hybracter outperformed Dragonflye. By integrating Plassembler[39], Hybracter improves plasmid recovery and avoids structural errors such as duplication of plasmids. However, for the *E. hormaechei* and *K. pneumoniae* genomes, Hybracter showed elevated error rates in large plasmids, likely due to errors introduced by Unicycler[40] within Plassembler (Table S4). Dragonflye performed worse than Flye alone, likely due to its default use of Racon polishing (instead of Flye's internal polisher) and the `--nano-raw` option which is suboptimal for modern ONT reads. With adjusted parameters (`--racon 0 --opts '-i 1' --nanohq`), Dragonflye's performance matched that of Flye (Table S5).

MAECI was the only consensus assembly tool tested apart from Autocycler. Despite incorporating three input assemblers (Canu, Flye and wtdbg2), MAECI did not consistently outperform Canu or Flye.

When run in an automated manner, Autocycler produced assemblies with the lowest sequence error counts of any method (median: 4 errors per assembly, range: 0–11). It was also structurally accurate in most cases, successfully recovering all replicons for four of the five genomes. This is in part because `autocycler_full.sh` uses Plassembler when generating input assemblies. However, the smallest plasmid of the *E. hormaechei* genome (2.5 kbp) was occasionally missed. In manually curated

runs, the missing plasmid could be identified by inspecting the clustering tree (output by `autocycler cluster`) and included in the final assembly by overriding the default clustering. These curated Autocycler assemblies had no structural errors reported by `assess_assembly.py`, Inspector, CRAQ or BUSCO (Table S2, Figures S1–S4).

Autocycler was the slowest assembly method tested, with a median runtime of 1 hour 44 minutes (Table S2, Figure S5), due to the time required to generate multiple input assemblies. Raven was the fastest, with a median runtime of 30 seconds per assembly. Autocycler had the second-highest peak RAM usage after NECAT, with a median of 11.3 GB, determined by the most memory-intensive assembler in its pipeline (NECAT). metaMDBG was the most memory-efficient, with a median of 1.2 GB.

In the low-depth test, Autocycler produced high-quality assemblies (no structural errors and six or fewer sequence errors) down to a depth of 23× (Table S6, Figure S6). Between 13× and 22×, results were more variable, with some assemblies failing and others exhibiting major structural errors. At depths of 12× and below, all assemblies failed. In the mixed-genome test, Autocycler assemblies remained uncontaminated when the secondary genome was present at less than ∼0.5× depth (Table S7, Figure S7). At contamination levels between ∼0.5× and ∼20×, Autocycler assemblies increasingly included plasmids from the secondary genome – first high-copy-number small plasmids, then low-copy-number large plasmids. At contamination levels >20×, the results became erratic, with assemblies sometimes including both chromosomes or neither. Across all tests, Autocycler produced only complete circular contigs, with a single exception at a 69:31 mixture where the *K. pneumoniae* chromosome was fragmented.

4. Discussion and conclusions

Consensus assembly offers a clear accuracy advantage over single-tool assembly. In our benchmarking, even the best-performing individual assemblers (Canu and Flye) consistently made avoidable errors. Consensus approaches mitigate such issues by averaging over multiple inputs, reducing both small-scale errors and structural inaccuracies. Trycycler[9] provided a robust framework for generating consensus bacterial genome assemblies, but it requires substantial user intervention, limiting its scalability. Autocycler brings the benefits of consensus assembly into an automated workflow, enabling accurate bacterial genome assembly at scale.

Autocycler does not guarantee perfect results, as its consensus reflects the input assemblies. If the inputs are fragmented (e.g. due to a repeat longer than the read length), Autocycler cannot produce a fully resolved consensus. When most inputs share the same error, that error can persist into the final assembly. In our evaluation, there were typically fewer than 10 sequence errors (substitutions and indels) per Autocycler assembly (Table S8), most commonly homopolymer-length errors resulting from systematic basecalling issues in ONT reads[41]. These errors are often inconsistent between runs, as they tend to occur at ambiguous sites where input assemblies disagree (e.g. with roughly equal support for two alternatives), which is why the curated Autocycler assemblies in this study did not always have the same error count as the automated Autocycler assemblies. The frequency of these errors depends on factors such as pore type (R10.4.1 is more accurate than R9.4.1), basecalling model (sup is more accurate than hac or fast) and bacterial strain. To address these errors, short-read polishing can be applied after Autocycler, yielding hybrid assemblies with maximal accuracy[15].

The only structural error observed in automated Autocycler assemblies in this study was the omission of a small plasmid from the *E. hormaechei* genome. Small plasmids are a common point of failure for long-read assemblers[6,42]. We included Plassembler among the input assemblers to improve small-plasmid recovery, but it often failed to circularise this plasmid due to a homopolymer sequence, leading to its exclusion during Autocycler's clustering step. However, Autocycler supports manual intervention at key steps in its pipeline, and in this case, manual review of the clustering enabled recovery of this plasmid. This flexible design allows Autocycler to function as both a scalable automated tool and a framework for high-accuracy curated reference genome assembly.

The set of input assemblers used with Autocycler is flexible and can be tailored to the user's needs. For example, Canu is a good choice when accuracy is a priority but may be excluded when faster runtimes are needed. Using multiple assemblers increases robustness, as no single tool performs best across all datasets. In this study, we used the `autocycler_full.sh` script provided with Autocycler, which runs eight assemblers: Canu, Flye, metaMDBG, miniasm, NECAT, NextDenovo, Plassembler and Raven. It excludes Myloasm and wtdbg2, which tend to produce high sequence error rates; hifiasm, which frequently generates extra contigs; LJA, which is recommended only for PacBio HiFi reads; and Hybracter, which internally runs both Flye and Plassembler, already included separately. This script is in Autocycler's `pipelines` directory, which invites users to contribute alternative pipelines, e.g. using different assemblers, parameters or workflow managers such as Nextflow[43]. We have not conducted a systematic evaluation of which assembler combinations perform best for different genome types or sequencing conditions, and this remains an important direction for future work.

Although not evaluated in this study, linear replicons pose additional challenges for genome assembly. Assemblers may erroneously extend hairpin ends or terminate open ends inconsistently. While Autocycler includes logic to detect and trim hairpin overlaps, full resolution of linear sequences still frequently requires manual intervention via the `autocycler clean` command. Improved support for such cases remains an area for improvement for both long-read assemblers and Autocycler.

Structural heterogeneity in the input assemblies is collapsed by Autocycler, which resolves each cluster by selecting the most supported path. As a result, assemblies from heterogeneous genomes will reflect the most common structural configuration present in the input assemblies. Significant heterogeneity may be visible in a cluster's intermediate output file (`4_merged.gfa`), but a more comprehensive characterisation is better performed after assembly using a structural variant caller such as Sniffles[44].

While Autocycler is designed for haploid prokaryotic isolate genomes, it may also be applicable in other contexts, with caveats. For eukaryotic genomes, Autocycler's assumption that input assemblies are complete makes it unsuitable for chromosomes that cannot be assembled end to end. Phased diploid assemblies also pose challenges, as both haplotypes are likely to be grouped together during clustering, so users would need to separate haplotypes and run Autocycler on each independently. Repetitive ends of linear chromosomes are difficult for Autocycler to resolve and would likely require manual finishing. For metagenomes, assemblies are generally too fragmented for direct use with Autocycler, but if a metagenome contains high-depth components that assemble completely, these

can be isolated and processed with Autocycler as individual genomes.

Autocycler is open-source, well documented and easy to install. It requires only modest system resources (excluding input assembly generation) and provides intermediate outputs to support transparency and manual curation. It fills a key gap in the current assembly tool landscape: existing consensus tools either underperform or do not scale, while long-read assembly pipelines rely on a single assembler and inherit its limitations. Because it relies on multiple inputs, an Autocycler-based pipeline is more computationally intensive than using a single assembler, but it usually yields better assemblies. We therefore recommend Autocycler for long-read bacterial genome projects where maximum assembly accuracy is required.

5. Competing interests

The authors declare that there are no conflicts of interest.

6. Author contributions statement

RRW conceived and programmed Autocycler with input from TPS. RRW performed the benchmarks and analysed the data. RRW, BPH and TPS wrote and reviewed the manuscript.

7. Funding

RRW is supported by an ARC Discovery Early Career Researcher Award (DE250100677). TPS is supported by an NHMRC Research Fellowship (APP1105525) and ARC Discovery Project (DP240102465). BPH is supported by an NHMRC Research Fellowship (APP1196103).

8. Data availability

Supplementary figures, tables and methods are available at `github.com/rrwick/Autocycler-paper`. Assemblies, reference genomes and read sets used in the analysis are available at `figshare.unimelb.edu.au/projects/Autocycler/247142`.

9. Acknowledgements

This research was performed in part at the Centre for Pathogen Genomics Innovation Hub, Department of Microbiology and Immunology, University of Melbourne at the Peter Doherty Institute for Infection and Immunity.

This paper acknowledges the PulseNet Asia-Pacific team at the Centre for Pathogen Genomics and the Microbiological Diagnostic Unit Public Health Laboratory (MDU PHL) for contributing data and isolates to the study. Support for PulseNet Asia-Pacific is funded by the US Centers for Disease Control and Prevention (CDC) Global Antimicrobial Resistance Laboratory and Response Network through the Association of Public Health Laboratories. MDU PHL is funded by the Victorian Government, Australia.

We thank the Autocycler alpha testers for their valuable feedback prior to the tool's public release: Alex Krause, Bogdan Iorga, Dan Whiley, Danielle Ingle, Erin Young, George Bouras, Josh Zhang, Mariel Beiers, Marko Verce, Matthew Croxen, Mona Taouk, Munazzah Maqbool, Oliver Schwengers, Sarah Baines, Steve Baeyen, Sudaraka Mallawaarachchi, Tatum Mortimer, Tue Sparholt Jørgensen, Tung Trinh and Ying Xu.

References

1. Arun Gonzales Decano, Catherine Ludden, Theresa Feltwell, Kim Judge, Julian Parkhill, and Tim Downing. Complete assembly of Escherichia coli sequence type 131 genomes using long reads demonstrates antibiotic resistance gene variation within diverse plasmid and chromosomal contexts. *mSphere*, June 2019.
2. Emily L. Gulliver, Vicki Adams, Vanessa Rossetto Marcelino, Jodee Gould, Emily L. Rutten, David R. Powell, Remy B. Young, Gemma L. D'Adamo, Jamia Hemphill, Sean M. Solari, Sarah A. Revitt-Mills, Samantha Munn, Thanavit Jirapanjawat, Chris Greening, Jennifer C. Boer, Katie L. Flanagan, Magne Kaldhusdal, Magdalena Plebanski, Katherine B. Gibney, Robert J. Moore, Julian I. Rood, and Samuel C. Forster. Extensive genome analysis identifies novel plasmid families in Clostridium perfringens. *Microbial Genomics*, April 2023.
3. Sergey Koren, Gregory P. Harhay, Timothy P. L. Smith, James L. Bono, Dayna M. Harhay, Scott D. Mcvey, Diana Radune, Nicholas H. Bergman, and Adam M. Phillippy. Reducing assembly complexity of microbial genomes with single-molecule sequencing. *Genome Biology*, September 2013.
4. Ryan R. Wick and Kathryn E. Holt. Benchmarking of long-read assemblers for prokaryote whole genome sequencing. *F1000Research*, December 2019. Number: 2138.
5. Ian Boostrom, Edward A. R. Portal, Owen B. Spiller, Timothy R. Walsh, and Kirsty Sands. Comparing long-read assemblers to explore the potential of a sustainable low-cost, low-infrastructure approach to sequence antimicrobial resistant bacteria with Oxford Nanopore sequencing. *Frontiers in Microbiology*, March 2022.
6. Jared Johnson, Marty Soehnlen, and Heather M. Blankenship. Long read genome assemblers struggle with small plasmids. *Microbial Genomics*, May 2023.
7. Shin-Hung Lin and Yu-Chieh Liao. CISA: Contig integrator for sequence assembly of bacterial genomes. *PLoS ONE*, March 2013.
8. Alejandro Hernandez Wences and Michael C. Schatz. Metassembler: merging and optimizing de novo genome assemblies. *Genome Biology*, December 2015.
9. Ryan R. Wick, Louise M. Judd, Louise T. Cerdeira, Jane Hawkey, Guillaume Méric, Ben Vezina, Kelly L. Wyres, and Kathryn E. Holt. Trycycler: consensus long-read assemblies for bacterial genomes. *Genome Biology*, December 2021. Number: 1.
10. Ryan R. Wick, Mark B. Schultz, Justin Zobel, and Kathryn E. Holt. Bandage: interactive visualization of *de novo* genome assemblies. *Bioinformatics*, October 2015. Number: 20.
11. Oxford Nanopore Technologies PLC. Dorado: Oxford Nanopore's basecaller. `https://github.com/nanoporetech/dorado`, 2025. Version 0.9.5.
12. Ryan R. Wick, Louise M. Judd, and Kathryn E. Holt. Assembling the perfect bacterial genome using Oxford Nanopore and Illumina sequencing. *PLOS Computational Biology*, March 2023. Number: 3.
13. Oxford Nanopore Technologies PLC. Medaka: Sequence correction provided by ONT Research. `https://github.com/nanoporetech/medaka`, 2024. Version 2.0.1.
14. Ryan R. Wick and Kathryn E. Holt. Polypolish: short-read polishing of long-read bacterial genome assemblies. *PLOS Computational Biology*, January 2022. Number: 1.

15. George Bouras, Louise M. Judd, Robert A. Edwards, Sarah Vreugde, Timothy P. Stinear, and Ryan R. Wick. How low can you go? Short-read polishing of Oxford Nanopore bacterial genome assemblies. *Microbial Genomics*, June 2024. Number: 6.
16. Sergey Koren, Brian P. Walenz, Konstantin Berlin, Jason R. Miller, Nicholas H. Bergman, and Adam M. Phillippy. Canu: scalable and accurate long-read assembly via adaptive k-mer weighting and repeat separation. *Genome Research*, May 2017. Number: 5.
17. Mikhail Kolmogorov, Jeffrey Yuan, Yu Lin, and Pavel A. Pevzner. Assembly of long, error-prone reads using repeat graphs. *Nature Biotechnology*, May 2019. Number: 5.
18. Haoyu Cheng, Gregory T. Concepcion, Xiaowen Feng, Haowen Zhang, and Heng Li. Haplotype-resolved de novo assembly using phased assembly graphs with hifiasm. *Nature Methods*, February 2021.
19. Anton Bankevich, Andrey V. Bzikadze, Mikhail Kolmogorov, Dmitry Antipov, and Pavel A. Pevzner. Multiplex de Bruijn graphs enable genome assembly from long, high-fidelity reads. *Nature Biotechnology*, July 2022.
20. Gaëtan Benoit, Sébastien Raguideau, Robert James, Adam M. Phillippy, Rayan Chikhi, and Christopher Quince. High-quality metagenome assembly from long accurate reads with metaMDBG. *Nature Biotechnology*, September 2024.
21. Heng Li. Minimap and miniasm: fast mapping and de novo assembly for noisy long sequences. *Bioinformatics*, July 2016. Number: 14.
22. Jim Shaw and Heng Li. Myloasm: metagenomic assembly with (noisy) long reads. `https://github.com/bluenote-1577/myloasm`, 2025. Version 0.1.0.
23. Ying Chen, Fan Nie, Shang-Qian Xie, Ying-Feng Zheng, Qi Dai, Thomas Bray, Yao-Xin Wang, Jian-Feng Xing, Zhi-Jian Huang, De-Peng Wang, Li-Juan He, Feng Luo, Jian-Xin Wang, Yi-Zhi Liu, and Chuan-Le Xiao. Efficient assembly of nanopore reads via highly accurate and intact error correction. *Nature Communications*, December 2021. Number: 1.
24. Jiang Hu, Zhuo Wang, Zongyi Sun, Benxia Hu, Adeola Oluwakemi Ayoola, Fan Liang, Jingjing Li, José R. Sandoval, David N. Cooper, Kai Ye, Jue Ruan, Chuan-Le Xiao, Depeng Wang, Dong-Dong Wu, and Sheng Wang. NextDenovo: an efficient error correction and accurate assembly tool for noisy long reads. *Genome Biology*, April 2024.
25. Robert Vaser and Mile Šikić. Time- and memory-efficient genome assembly with Raven. *Nature Computational Science*, May 2021. Number: 5.
26. Jue Ruan and Heng Li. Fast and accurate long-read assembly with wtdbg2. *Nature Methods*, February 2020.
27. Ryan R. Wick. Minipolish: A tool for Racon polishing of miniasm assemblies. `https://github.com/rrwick/Minipolish`, 2020. Version 0.1.3.
28. Jiang Hu, Junpeng Fan, Zongyi Sun, and Shanlin Liu. NextPolish: a fast and efficient genome polishing tool for long-read assembly. *Bioinformatics*, April 2020. Number: 7.
29. Robert A. Petit III. Dragonflye: Assemble bacterial isolate genomes from nanopore reads. `https://github.com/rpetit3/dragonflye`, 2024. Version 1.2.1.
30. George Bouras, Ghais Houtak, Ryan R. Wick, Vijini Mallawaarachchi, Michael J. Roach, Bhavya Papudeshi, Louise M. Judd, Anna E. Sheppard, Robert A. Edwards, and Sarah Vreugde. Hybracter: enabling scalable, automated, complete and accurate bacterial genome assemblies. *Microbial Genomics*, May 2024. Number: 5.
31. Jidong Lang. MAECI: A pipeline for generating consensus sequence with nanopore sequencing long-read assembly and error correction. *PLOS ONE*, May 2022.
32. Li Tang, Min Li, Fang-Xiang Wu, Yi Pan, and Jianxin Wang. MAC: Merging assemblies by using adjacency algebraic model and classification. *Frontiers in Genetics*, January 2020.
33. Mahul Chakraborty, James G. Baldwin-Brown, Anthony D. Long, and J. J. Emerson. Contiguous and accurate *de novo* assembly of metazoan genomes with modest long read coverage. *Nucleic Acids Research*, July 2016.
34. Hind Alhakami, Hamid Mirebrahim, and Stefano Lonardi. A comparative evaluation of genome assembly reconciliation tools. *Genome Biology*, December 2017.
35. Heng Li. Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics*, September 2018. Number: 18.
36. Yu Chen, Yixin Zhang, Amy Y. Wang, Min Gao, and Zechen Chong. Accurate long-read de novo assembly evaluation with Inspector. *Genome Biology*, November 2021.
37. Kunpeng Li, Peng Xu, Jinpeng Wang, Xin Yi, and Yuannian Jiao. Identification of errors in draft genome assemblies at single-nucleotide resolution for quality assessment and improvement. *Nature Communications*, October 2023.
38. Fredrik Tegenfeldt, Dmitry Kuznetsov, Mosè Manni, Matthew Berkeley, Evgeny M Zdobnov, and Evgenia V Kriventseva. OrthoDB and BUSCO update: annotation of orthologs with wider sampling of genomes. *Nucleic Acids Research*, January 2025.
39. George Bouras, Anna E. Sheppard, Vijini Mallawaarachchi, and Sarah Vreugde. Plassembler: an automated bacterial plasmid assembly tool. *Bioinformatics*, July 2023.
40. Ryan R. Wick, Louise M. Judd, Claire L. Gorrie, and Kathryn E. Holt. Unicycler: resolving bacterial genome assemblies from short and long sequencing reads. *PLOS Computational Biology*, June 2017. Number: 6.
41. Mantas Sereika, Rasmus Hansen Kirkegaard, Søren Michael Karst, Thomas Yssing Michaelsen, Emil Aarre Sørensen, Rasmus Dam Wollenberg, and Mads Albertsen. Oxford Nanopore R10.4 long-read sequencing enables the generation of near-finished bacterial genomes from pure cultures and metagenomes without short-read or reference polishing. *Nature Methods*, July 2022. Number: 7.
42. Nicole Lerminiaux, Ken Fakharuddin, Michael R. Mulvey, and Laura Mataseje. Do we still need Illumina sequencing data? Evaluating Oxford Nanopore Technologies R10.4.1 flow cells and the Rapid v14 library prep kit for Gram negative bacteria whole genome assemblies. *Canadian Journal of Microbiology*, May 2024.
43. Paolo Di Tommaso, Maria Chatzou, Evan W Floden, Pablo Prieto Barja, Emilio Palumbo, and Cedric Notredame. Nextflow enables reproducible computational workflows. *Nature Biotechnology*, April 2017. Number: 4.
44. Moritz Smolka, Luis F. Paulin, Christopher M. Grochowski, Dominic W. Horner, Medhat Mahmoud, Sairam Behera, Ester Kalef-Ezra, Mira Gandhi, Karl Hong, Davut Pehlivan, Sonja W. Scholz, Claudia M. B. Carvalho, Christos Proukakis, and Fritz J. Sedlazeck. Detection of mosaic and population-level structural variants with Sniffles2. *Nature Biotechnology*, January 2024. Number: 10.