Genome Biology

SOFTWARE Open Access

Trycycler: consensus long-read assemblies for bacterial genomes

Ryan R. Wick[1*], Louise M. Judd[1], Louise T. Cerdeira[1], Jane Hawkey[1], Guillaume Méric[1,2], Ben Vezina[1], Kelly L. Wyres[1] and Kathryn E. Holt[1,3]

* Correspondence: rrwick@gmail.com
[1]Department of Infectious Diseases, Central Clinical School, Monash University, Melbourne, VIC 3004, Australia
Full list of author information is available at the end of the article

Abstract

While long-read sequencing allows for the complete assembly of bacterial genomes, long-read assemblies contain a variety of errors. Here, we present Trycycler, a tool which produces a consensus assembly from multiple input assemblies of the same genome. Benchmarking showed that Trycycler assemblies contained fewer errors than assemblies constructed with a single tool. Post-assembly polishing further reduced errors and Trycycler+polishing assemblies were the most accurate genomes in our study. As Trycycler requires manual intervention, its output is not deterministic. However, we demonstrated that multiple users converge on similar assemblies that are consistently more accurate than those produced by automated assembly tools.

Keywords: Genome assembly, Bacterial genomics, Whole-genome sequencing, Long-read sequencing, Oxford Nanopore sequencing

Background

Long-read assembly is the process of reconstructing a genome from long sequencing reads (>10 kbp), such as those made by Oxford Nanopore Technologies (ONT) or Pacific Biosciences (PacBio) platforms. ONT's long-read sequencing platforms are popular for bacterial sequencing due to their low cost per sample [1, 2]. Since long reads can span larger genomic repeats than short reads (e.g., reads from Illumina sequencing platforms), long-read assembly can produce larger contigs than short-read assembly [3–6]. For bacterial genomes, it is often possible to produce a long-read-only assembly (an assembly made solely from long-read data) which is complete: one fully assembled contig for each replicon in the genome [7, 8]. There are many long-read assemblers appropriate for use on bacterial genomes, including Canu [9], Flye [10], Raven [11], and Redbean [12]. Each has advantages and disadvantages, but in a recent benchmarking study, we found Flye to be the best-performing bacterial genome assembler in many metrics [13].

Since long-read assembly of bacterial genomes can reliably yield chromosome-scale contigs, it is sometimes considered to be a solved problem [14], with much assembler development now focusing on more challenging scenarios such as eukaryotic genomes and metagenomes [15, 16]. However, long-read bacterial assemblies are not perfect. Small-scale errors (such as homopolymer-length errors) are commonly discussed and addressed [7, 17–19], but larger-scale errors (tens to hundreds of base pairs) also occur in most assemblies [13]. Even though most bacterial replicons are circular, long-read assemblers often fail to produce cleanly circularized contigs, where the last base in the contig is immediately followed by the first base. Spurious contigs are often present in assemblies (e.g., from contaminant sequences), and small plasmids can be omitted due to their underrepresentation in ONT read sets [20]. Hybrid assembly, which uses both short and long reads, can mitigate some of these problems, but hybrid assemblers also fail to produce error-free genome assemblies [21] and can introduce confusion if short- and long-read libraries are not constructed from the same DNA extraction [22]. Long-read assembly of bacterial genomes is therefore not a completely solved problem, and there is still much room for improvement.

As assembly is often the first step in bioinformatic pipelines, assembly errors can have negative implications for downstream analysis. Here, we introduce Trycycler, a computational tool which enables high-quality long-read-only assemblies of bacterial genomes. It takes multiple assemblies of the same genome as input and produces a single consensus assembly. Trycycler exploits the fact that while long-read assemblies almost always contain errors, different assemblies of the same genome typically have different errors [13]. Trycycler can therefore combine multiple input assemblies to produce a consensus assembly with fewer errors than any of its inputs.

Results

Approach and implementation

The Trycycler pipeline consists of multiple steps which are run separately (overview in Fig. 1, more detail in Additional file 1: Fig. S1). At the clustering and reconciliation steps, the user may need to make decisions and intervene. This means that Trycycler is

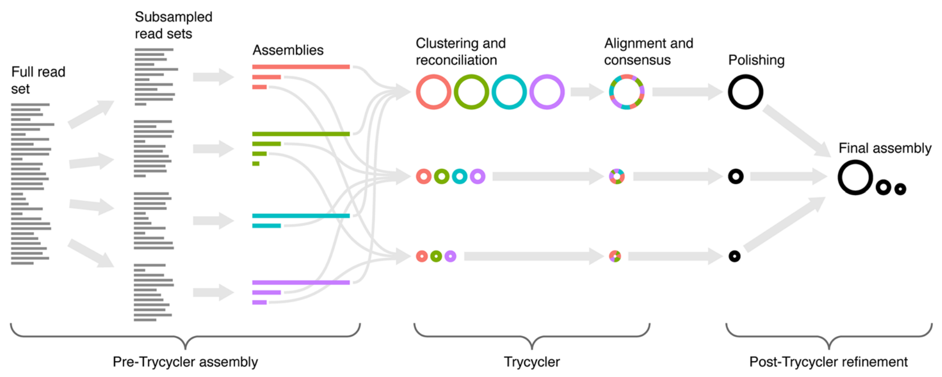

Fig. 1 Overview of the Trycycler long-read assembly pipeline. Before Trycycler is run, the user must generate multiple complete assemblies of the same genome, e.g., by assembling different subsets of the original long-read set. Trycycler then clusters contigs from different assemblies and produces a consensus contig for each cluster. These consensus contigs can then be polished (e.g., with Medaka) and combined into a final high-quality long-read-only assembly

not an automated process appropriate for high-throughput assembly. Trycycler is implemented in Python and uses the NumPy, SciPy, and edlib packages [23–26].

Before Trycycler is run, the user must generate multiple input assemblies of the same genome (Additional file 1: Fig. S1A). The input assemblies should be complete: one contig per replicon. If complete assemblies are not possible (e.g., due to insufficient read length) or read depth is shallow (e.g., <25× depth), then Trycycler is not appropriate. We recommend users generate 12 independent input assemblies, but this value can be adjusted down (to save computational time) or up (to improve robustness). It is desirable to maximize the independence of the input assemblies, as this will reduce the chance that the same error will occur in multiple assemblies. One way to achieve such independence is to use multiple assemblers, as different assembly algorithms can lead to different assembly errors [13]. For example, in the tests reported here, we used Flye [10], Miniasm/Minipolish [13], Raven [11], and Redbean [12]. Random read subsampling can provide further independence, where each assembly is generated from a different subsample of the full read set (Trycycler v0.5.0 has a "subsample" command to facilitate this). Deeper long-read sets are therefore desirable, as they enable more independent subsets.

The first step in the Trycycler pipeline is contig clustering (Additional file 1: Fig. S1B). It aims to group contigs of the same replicon from different input assemblies, so subsequent steps can be carried out on a per-replicon basis. For example, if the genome in question had one chromosome and one plasmid, then Trycycler clustering should produce two clusters: one for the chromosomal contigs and one for the plasmid contigs. To make clusters, Trycycler conducts complete-linkage hierarchical clustering on all pairwise Mash distances between contigs [27]. To aid interpretation, a FastME tree is built using the pairwise distances [28]. After clustering is complete, the user must decide which clusters are valid (i.e., represent completely assembled replicons in the genome) and which are invalid (i.e., represent incomplete, misassembled, or spurious sequences)—a key point of human judgment in the Trycycler process.

The next step is to "reconcile" each cluster's contig sequences with each other (Additional file 1: Fig. S1C). This involves converting sequences to their reverse complement as necessary to ensure that all sequences in the cluster are in the same orientation. Most bacterial replicons are circular, so Trycycler aligns the start and end of each contig to the other contigs in the cluster to determine if bases need to be added or removed for clean circularization (can be disabled for linear replicons by using the --linear option). It then rotates each sequence to begin at the same position. Some gene sequences (e.g., *dnaA* and *repA*) are often used as starting positions in complete genomes, so Trycycler contains a database of these genes and will preferentially use them as the contig starting position (see the "Methods" section). If no sequence from this database is found (with ≥95% coverage and ≥95% identity), Trycycler will use a randomly chosen unique sequence instead. Cluster reconciliation will fail if a contig cannot be circularized or if any of the pairwise alignments within the cluster have low identity. In such cases, Trycycler will suggest interventions to resolve the issue, but it is up to the user to manually exclude or modify the contig sequences as necessary.

After reconciliation, each cluster's sequences will have a consistent strand and starting position, making them appropriate for global multiple sequence alignment (Additional file 1: Fig. S1D). To improve computational performance, Trycycler subdivides the sequences, using 1-kbp pieces with each piece extended as necessary to ensure that the boundaries between pieces do not start/end in repetitive regions. It uses MUSCLE [29] to produce a multiple sequence alignment for each piece and then stitches the pieces together to produce a single multiple sequence alignment for the full cluster sequences. Trycycler then aligns the entire read set to each contig sequence so it can be assigned to a particular cluster (Additional file 1: Fig. S1E).

The final step in Trycycler's pipeline is the generation of a consensus sequence for each cluster (Additional file 1: Fig. S1F). It does this by dividing the multiple sequence alignment into regions where there is or is not any variation. For all regions where there is variation, Trycycler must choose which variant will go into the consensus. The best variant is defined as the one with the minimum total Hamming distance to the other variants, an approach which favors more common variants. In the event of a tie between two variants, Trycycler aligns the cluster's reads to each possibility and chooses the one which produces the largest total alignment score—i.e., the variant which is in best agreement with the reads. The final Trycycler consensus sequence for the cluster is produced by taking the best variant for each region of variation in the multiple sequence alignment.

After Trycycler finishes, we recommend performing long-read polishing on its consensus sequences (Additional file 1: Fig. S1G). Polishing is not incorporated into Trycycler, as that step can be specific to the long-read sequencing technologies used, e.g., Medaka [30] polishing for ONT assemblies. If short reads are available, short-read polishing (e.g., with Pilon [31]) can also be performed to further improve assembly accuracy.

The code and documentation for Trycycler v0.3.3 (the version used to generate the assemblies in this manuscript) are available at the DOI 10.5281/zenodo.3966493. The current version of Trycycler (v0.5.0) is available on GitHub (github.com/rrwick/Trycycler).

Performance on simulated reads

In silico read simulation allows for a straightforward test of assembly accuracy against a ground truth: reads are generated from a reference genome, the reads are assembled, and the resulting assembly is compared back to the original reference sequence. For this analysis, we simulated short and long reads from 10 reference genomes which belong to the 10 most common bacterial species in RefSeq (Additional file 2: References). We assembled each genome with long-read-only approaches (Miniasm/Minipolish [13], Raven [11], Flye [10], and Trycycler), long-read-first hybrid approaches (Pilon [31] polishing of each long-read-only assembly), and a short-read-first hybrid approach (Unicycler [21]). We quantified the accuracy of each assembly's chromosomal contig using two main metrics: mean identity and worst-100-bp identity (the minimum identity observed among a 100-bp sliding window).

Comparing only the long-read assemblers to each other (Flye, Miniasm/Minipolish and Raven), it was clear that Flye performed best (Additional file 1: Fig. S2). This was true both before Pilon polishing with short reads (mean identity Q41 vs Q38; mean worst-100-bp-identity 95.8% vs 50.8–90.9%) and after Pilon polishing (mean identity Q57 vs Q42–Q55; mean worst-100-bp identity 96.1% vs 50.8–95.7%). Our main results therefore exclude Miniasm/Minipolish and Raven, leaving only the best-performing long-read assembler: Flye.

Figure 2 shows the mean assembly identities and worst-100-bp assembly identities from each approach, using 10 simulated read sets. In both metrics, Trycycler reliably produced higher-quality assemblies than Flye (mean identity Q51 vs Q41; mean worst-100-bp identity 99.5% vs 95.8%). This result also held true for long-read-first hybrid assemblies, where Trycycler+Pilon outperformed Flye+Pilon (mean identity Q74 vs Q57; mean worst-100-bp identity 99.9% vs 96.1%). Unicycler's short-read-first hybrid assemblies performed notably worse than the long-read-first hybrid approaches (mean identity Q25; mean worst-100-bp identity 76.5%).

Performance on real reads

Since simulated reads cannot perfectly emulate real sequencing [32], we also tested assembly methods with real-read sets. We chose seven bacterial isolates for this study (Additional file 3: Genomes), each belonging to a different bacterial species with clinical relevance. The challenge with real reads is the absence of a clear ground truth against which to compare assemblies. To circumvent this issue, we instead produced two independent sets of long+short (ONT+Illumina) reads for each test organism. In brief, a single DNA extraction from each organism was used to prepare two ONT libraries (one ligation and one rapid), and a single Illumina library (the results of which were divided into two non-overlapping read sets); full details are described in the "Methods" section. For each assembly method, we compared the assembly from read set A to the assembly of read set B, differences between them indicating assembly errors. While this approach could suffer from false negatives if both assemblies contained the same error, it cannot suffer from false positives, as wherever two assemblies of the same genome differ, at least one of the two must be in error.

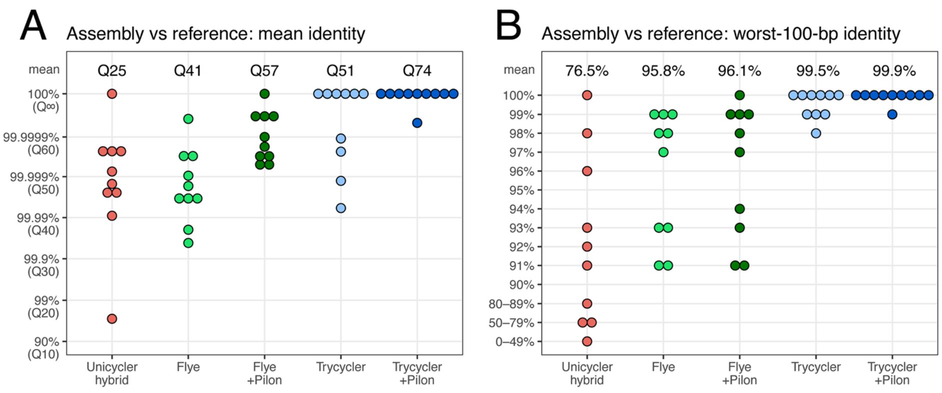

Fig. 2 Results for the tests using simulated reads. For 10 reference genome sequences, we simulated both short and long reads. The read sets were then assembled with Unicycler (short-read-first hybrid assembly), Flye (long-read-only assembly), Flye+Pilon (long-read-first hybrid assembly), Trycycler (long-read-only assembly), and Trycycler+Pilon (long-read-first hybrid assembly). Each assembled chromosome was aligned back to the reference chromosome to determine the mean assembly identity (**A**) and the worst identity in a 100-bp sliding window (**B**). For long-read-only assembly, Trycycler consistently achieved higher accuracy than Flye. Trycycler+Pilon (i.e., using Pilon to polish the Trycycler assembly with short reads) achieved the highest accuracy and did better than alternative hybrid approaches (Unicycler and Flye+Pilon)

We tested the same assemblers as were used in the simulated-read tests but added an additional long-read polishing step with Medaka, an ONT-specific polishing tool. We therefore produced unpolished long-read-only assemblies (with Miniasm/Minipolish, Raven, Flye, and Trycycler), polished long-read-only assemblies (the same assemblers plus Medaka), long-read-first hybrid assemblies (the same assemblers plus Medaka and short-read polishing with Pilon), and short-read-first hybrid assemblies (with Unicycler). Each assembly approach was used on both read set A and read set B for each of the test organisms.

Assembly accuracy was quantified using the metrics from the simulated-read tests: mean identity and worst-100-bp identity. Instead of being based on an assembly-to-reference alignment (as was done for the simulated-read tests), these metrics used an alignment of the read-set-A-assembled chromosome to the read-set-B-assembled chromosome. For the *Serratia marcescens* genome, read set B failed to produce a complete chromosome with most assembly methods (due to long genomic repeats and a short read N50, see Additional file 3: Reads), so this genome was excluded, leaving six genomes in the analysis. As was the case for the simulated-read tests, Flye assemblies were higher quality than Miniasm/Minipolish and Raven assemblies at all polishing stages (Additional file 1: Fig. S3): unpolished (mean identity Q34 vs Q28–Q32; mean worst-100-bp identity 81.8% vs 20.2–21.8%), Medaka-polished (mean identity Q40 vs Q30–Q35; mean worst-100-bp identity 94.7% vs 28.2–38.0%) and Medaka+Pilon-polished (mean identity Q56 vs Q31–Q37; mean worst-100-bp identity 94.7% vs 28.2–40.0%). Flye was also the only long-read assembler to produce complete chromosomes for both read sets of all six genomes, so Miniasm/Minipolish and Raven were excluded from our main results.

Since the mean identity and worst-100-bp identity metrics could fail to identify all assembly errors in the real-read tests, we also used two other approaches for assessing the quality of de novo assemblies. The first was ALE [33], which uses short-read alignments to the assembled sequence to produce a likelihood score for that assembly (higher scores being better), which we normalized for each genome to produce a z-score. Mapping accuracy, evenness of read depth, and evenness of insert size extracted from the short-read alignments are all used by ALE to generate a likelihood score. The second de novo assessment approach was IDEEL [34, 35], which compares the length of predicted proteins in the assembly to a database of known proteins. Indel errors in the assembly cause frameshifts in coding sequences leading to truncations, so an error-prone assembly will tend to have predicted proteins which are shorter than their best-matching known proteins. We quantified the fraction of predicted proteins in each assembly which were ≥95% the length of their best-matching known protein (higher fractions being better).

Figure 3 shows the real-read results: mean identity, worst-100-bp identity, ALE *z*-scores, and IDEEL full-length proteins. In the mean identity metric, Trycycler performed better than Flye at all levels of polishing (Q37 vs Q34 before polishing; Q42 vs Q40 after Medaka polishing; Q62 vs Q56 after Medaka+Pilon polishing). This advantage was also apparent in the worst-100-bp identity metric (96.7% vs 81.8% before polishing; 97.0% vs 94.7% after Medaka polishing; 98.3% vs 94.7% after Medaka+Pilon polishing). Both long-read-first hybrid approaches (Flye+Medaka+Pilon and Trycycler+Medaka+Pilon) outperformed Unicycler's short-read-first hybrid assemblies (mean

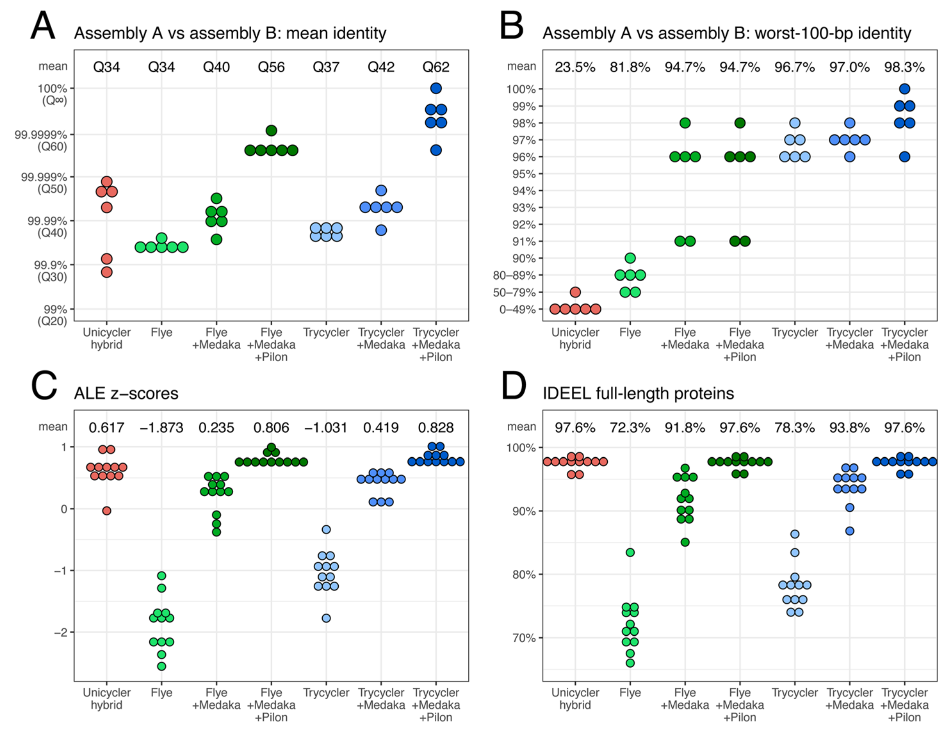

Fig. 3 Results for the real-read tests. For six genomes, we produced two independent hybrid read sets from the same DNA extraction. The read sets were then assembled with Unicycler (short-read-first hybrid assembly), Flye (long-read-only assembly), Flye+Medaka (long-read-only assembly), Flye+Medaka+Pilon (long-read-first hybrid assembly), Trycycler (long-read-only assembly), Trycycler+Medaka (long-read-only assembly), and Trycycler+Medaka+Pilon (long-read-first hybrid assembly). For each genome and each assembly approach, we aligned the two independently assembled chromosomes to each other to determine the mean assembly identity (**A**) and the worst identity in a 100-bp sliding window (**B**). For long-read-only assembly, Trycycler consistently achieved higher accuracy than Flye (both before and after Medaka polishing). Trycycler+Medaka+Pilon achieved the highest accuracy and did better than alternative hybrid approaches (Unicycler and Flye+Medaka+Pilon). We also assessed the accuracy of each of the 12 assembled chromosomes using ALE (**C**) and IDEEL (**D**). ALE assigns a likelihood score (transformed into *z*-scores on a per-genome basis) to each assembly based on its concordance with the Illumina read set. IDEEL identifies the proportion of predicted proteins which are ≥95% the length of their best-matching known protein in a database

identity Q34 and worst-100-bp identity 23.5%). The ALE results are consistent with the identity metrics: Trycycler assemblies had higher mean ALE *z*-scores than Flye assemblies at all polishing levels (–1.031 vs –1.873 before polishing; 0.419 vs 0.235 after Medaka polishing; 0.828 vs 0.806 after Medaka+Pilon polishing) and long-read-first hybrid assemblies were superior to Unicycler assemblies (mean ALE *z*-score of 0.617). IDEEL results showed the same trend, with Trycycler assemblies having more full-length proteins than Flye assemblies (78.3% vs 72.3% before polishing; 93.8% vs 91.8% after Medaka polishing), but all hybrid assemblies performed equivalently in this metric (97.6%).

While the above results used Medaka polishing after Trycycler (i.e., Trycycler+Medaka), it is also possible to run Medaka polishing on Trycycler's input assemblies (i.e., Medaka+Trycycler) or on both Trycycler's input assemblies and its final assembly (i.e., Medaka+Trycycler+Medaka). We tried these alternative approaches using the real-read data and found that while all performed similarly (mean identity Q41–Q42), the best results were achieved when Medaka was the final step in the process (Additional file 3: Medaka order).

We therefore recommend the Trycycler+Medaka approach both for its simplicity and accuracy.

Type and location of errors

Additional file 1: Fig. S4 shows the positions of errors in the assemblies of each of the 16 genomes (10 simulated and six real), with repetitive regions of the genomes indicated. Errors in long-read-only assemblies (Flye, Flye+Medaka, Trycycler, and Trycycler+Medaka) were distributed across the genomes, occurring in both repeat and non-repeat sequences. Long-read-first hybrid assemblies (Flye+Medaka+Pilon and Trycycler+Medaka+Pilon) usually had higher error rates in repeat sequences, and in many cases, there were no errors in the non-repeat sequences of the genome. Short-read-first hybrid assemblies (Unicycler) often had clusters of errors which occurred in both repeat and non-repeat sequences. Indel errors were more common than substitution errors for all assemblers: 44% of total errors were insertions, 47% were deletions, and 9% were substitutions. For the real reads, Flye assemblies sometimes had local spikes in error rates (indicating a more serious error or a cluster of errors) before Medaka polishing, but these spikes were not present after Medaka polishing. Trycycler assemblies did not suffer from this same problem. Flye assemblies often had errors at the position corresponding to the original start/end of the contig.

The Flye errors at the start/end of the contig were caused by imperfect circularization: missing or duplicated bases at the start/end of a circular contig, a phenomenon we described in greater detail in a previous benchmarking study of long-read assemblers [13]. These errors were not corrected by Medaka or Pilon because those tools are not aware of contig circularity, i.e., that the contig's last base should immediately precede its first base. Since our analysis involved normalizing all assemblies to a consistent starting position (required for global alignment), missing/duplicated bases at the start/end of a contig registered as a middle-of-the-sequence indel error in our tests. These indel errors reduced the mean identity and, if large enough, the worst-100-bp identity as well.

To assess the effect of circularization errors on Flye accuracy, we manually fixed the circularization of all Flye assemblies using the original reference sequence (in the simulated-read tests) or the Trycycler+Medaka+Pilon assembly (in the real-read tests). Of the 22 Flye assemblies (10 from simulated reads, 12 from real reads), four had perfect circularization, five had duplicated bases, and 13 had missing bases. The worst Flye circularization error was a 13-bp deletion, and the mean magnitude of Flye circularization errors was 3.7 bp (Additional files 2 and 3: Flye circularization). We then reran our analyses using the fixed-circularization version of Flye assemblies, and the results are shown in Additional file 1: Fig. S5 for simulated reads and Additional file 1: Fig. S6 for real reads. Flye performed better in these results, especially in the worst-100-bp identity metric, indicating that in many cases, the circularization error was the largest single error in the Flye assembly. However, Trycycler still produced more accurate assemblies than Flye at each polishing stage (unpolished, Medaka-polished, and Pilon-polished).

Consistency of Trycycler results

Trycycler is not a fully automated pipeline—it requires human judgment and intervention. This raised the question of how well it performs in the hands of different users. To answer this, we recruited five researchers who were experienced in bioinformatics but not involved in Trycycler development. They were given an ONT read set for each of the six genomes used in the real-read tests and tasked with producing a Trycycler assembly without any assistance from the Trycycler developer (using only the Trycycler documentation to guide them). We then compared the resulting assemblies, looking at both presence/absence of contigs as well as chromosomal sequence identity (Fig. 4).

The main source of variation between different users' Trycycler assemblies was the inclusion/exclusion of plasmid contigs (Fig. 4A). Small plasmids often pose problems for long-read assemblers, and this caused them to sometimes be excluded by Trycycler users. Contaminant plasmid contigs (e.g., cross-barcode contamination) were sometimes included in Trycycler assemblies. Replicons with a large-scale error or misassembly occurred in many of the single-assembler assemblies (from Miniasm/Minipolish, Raven, and Flye). These errors included fragmented replicons (e.g., splitting one replicon sequence between two contigs), doubling a replicon in a single contig (e.g., assembling a 6-kbp plasmid into a 12-kbp contig), large-scale circularization problems (e.g., 80 kbp of start/end overlap), and redundant contigs (e.g., producing five contigs for a single replicon). This type of error was very rare in the Trycycler assemblies (present in only one case). Detailed descriptions of all such errors are in Additional file 4: Matrix.

To assess the consistency of assembled sequences, we built a neighbor-joining tree (based on pairwise alignment distances) of the assembled chromosomes for each of the six genomes (Fig. 4B). The developer's Trycycler+Medaka+Pilon assembly was included as a reference sequence, as the real-read test results (Fig. 3) indicate these to be the most accurate representation of the genomes. For each test isolate, the Trycycler assemblies generated by different users were closer to the reference sequence than any of the (automated) single-assembler assemblies (Fig. 4C), and there were comparatively few differences between Trycycler assemblies from different users (Fig. 4D). All differences between Trycycler assemblies generated by different users were small-scale: most were only single-bp differences, and the largest difference was a 4-bp indel in a tandem repeat (Additional file 4: Trycycler vs Trycycler). The most common difference was a 1-bp discrepancy in the length of a homopolymer sequence (accounted for 78.5% of all Trycycler-vs-Trycycler sequence differences).

Discussion

By combining multiple input assemblies into a consensus sequence, Trycycler produced the most accurate long-read-only assemblies in our study (Figs. 2, 3, and 4). Trycycler assemblies only contained small-scale errors (i.e., their accuracy in a 100-bp sliding window remained high), while assemblies produced by single assemblers often contained medium-to-large-scale errors (Figs. 2 and Fig. 3). Trycycler also helped to guard against inexact circularization, inclusion of spurious contigs, and exclusion of genuine contigs. However, Trycycler requires deeper long-read sets (to allow for multiple independent input assemblies via read subsampling), more computational resources, and more human input than single-assembler assemblies.

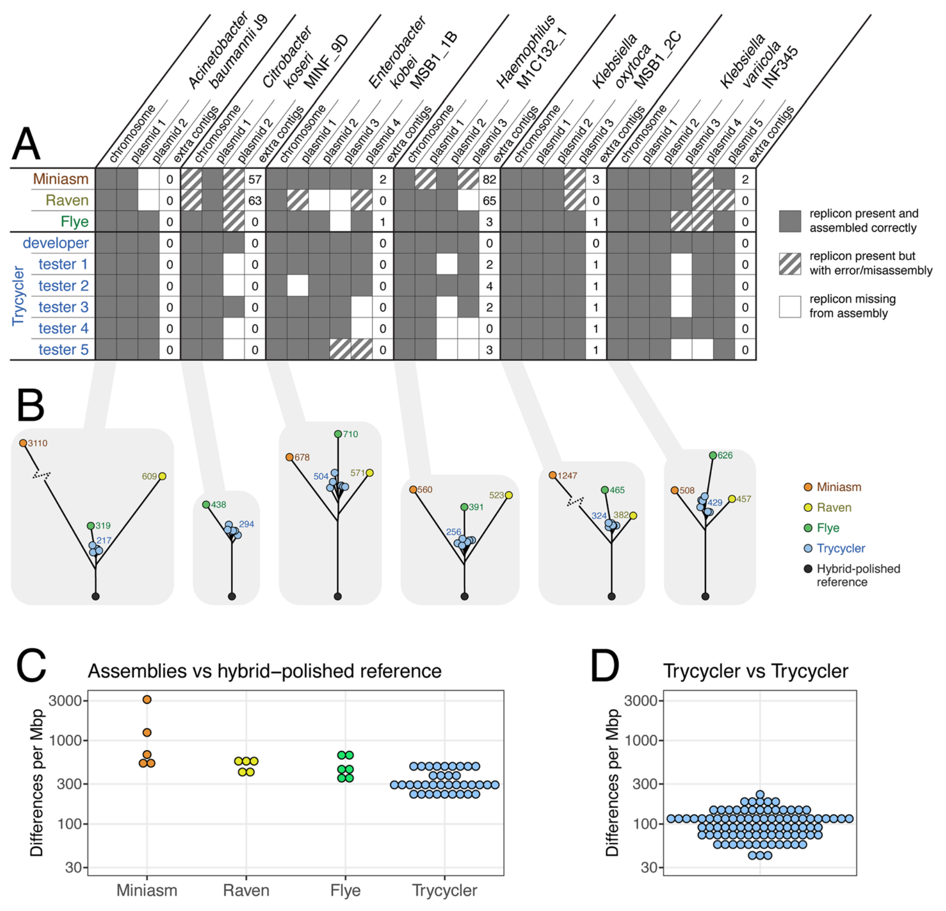

Fig. 4 Results for the multi-user test which assessed the consistency of Trycycler assemblies when run by different users. Results include assemblies from three different long-read assemblers (Miniasm/Minipolish, Raven, and Flye, all automated and deterministic for a given set of reads and parameters, i.e., independent of user) and Trycycler assemblies from six different users (the developer of Trycycler and five testers). **A** Presence/absence matrix for the replicons in the test genomes. Each replicon was classified as either present in the assembly, absent from the assembly, or present but with an error/misassembly (see Additional file 4: Matrix for more detail). The number of additional contigs (e.g., spurious or contaminant sequences) is also indicated for each assembly. All Trycycler assemblies contained an accurate chromosome, and only one Trycycler assembly contained misassemblies. However, in many cases, the Trycycler testers excluded a true plasmid (most commonly a small plasmid) or included an additional plasmid (most commonly constructed from cross-barcode contaminating reads). **B** Neighbor-joining trees of all available assemblies for each of the chromosomes, based on pairwise alignment distances. Hybrid-polished (Medaka+Pilon) versions of the developer's Trycycler assemblies were included as reference sequences. The values indicate the number of single-bp differences per Mbp between each assembly and the polished reference (values for Trycycler are the mean of all six Trycycler assemblies). For each genome, the Trycycler assemblies cluster tightly and are closer to the polished reference than those from other long-read assemblers. **C** Differences between each assembled chromosome and the hybrid-polished reference. Values are single-bp differences per Mbp of sequence. Trycycler assemblies contain fewer differences, on average, compared to the single-assembler assemblies. **D** Pairwise differences between Trycycler assemblies of each chromosome. Values are single-bp differences per Mbp of sequence, and there are 90 values (6 genomes × 15 unique pairwise combinations per genome)

Creating a Trycycler assembly often requires judgment calls and manual intervention, particularly after Trycycler's clustering step where users must decide which contig clusters are valid (represent true replicons in the genome), which clusters are invalid (spurious, misassembled, or contaminant sequences), and whether any contig sequences need to be trimmed or excluded to allow for cluster reconciliation. While fully automated pipelines are useful for high-throughput analyses, this would be difficult and undesirable for Trycycler. There are many ways that long-read assembly can fail, and Trycycler does not restrict which input assemblers can be used, so planning for all failure modes is not possible. Trycycler also allows users to subjectively judge the "assemblability" of their read set based on the coherence of the contig clustering. If input contigs cluster well, users can proceed with confidence. If not, they should investigate why the reads failed to produce consistent assemblies, e.g., insufficient read length or genome heterogeneity. See Additional file 1: Fig. S7 for examples of this process in the real-read tests.

Our multi-user consistency test showed that the cluster-selection step was a significant source of variability in Trycycler results, manifesting as missing/extra replicons in the assembly, a problem exacerbated by cross-barcode contamination and the fact that long-read assemblers often struggle with small plasmid sequences. This demonstrates that user skill and experience is an important factor in producing an ideal Trycycler assembly. To mitigate this concern, we have provided extensive documentation for Trycycler, with sample data, example analyses, and FAQs to guide users. Notably though, Trycycler chromosome sequences generated by different users were more similar to one another than to any of the sequences generated by the deterministic single assemblers (Fig. 4b).

Even ideal Trycycler assemblies still contain small-scale errors. Our real-read tests achieved an accuracy of Q37, equivalent to about one error per 5 kbp of sequence, most of which (>80%, see Additional file 4: Trycycler vs polished) were in homopolymer sequences. While we have previously shown that many errors in a long-read *Klebsiella pneumoniae* assembly occurred in Dcm methylation motifs [19, 36], this was not the case for genomes in this study (Additional file 4: Trycycler vs polished). Residual small-scale errors in assemblies result from systematic basecalling errors (i.e., when many of the reads covering a genomic position contain the same error) and therefore cannot be entirely avoided in the assembly process. Instead, these errors can be addressed before assembly (during basecalling) or after assembly (with polishing). Superior ONT basecalling can be achieved with improvements in sequencing chemistry, neural networks, and training sets, so future developments in these areas will result in better Trycycler assembly accuracy.

Polishing is a post-assembly processing step to improve sequence accuracy, and it can be carried out using either long or short reads. Our study showed that Medaka, a long-read polishing tool for ONT reads, was able to fix approximately half of the errors in long-read assemblies. Medaka was also effective at repairing many of the worst errors in a Flye assembly, making Flye+Medaka assemblies nearly as accurate as Trycycler+Medaka assemblies. Subsequent short-read polishing with Pilon was able to bring sequence identity close to 100%, with most of the remaining unfixed errors residing in genomic repeats (where short-read alignment is unreliable). Our study also found short-read-first hybrid assembly (short-read assembly followed by long-read scaffolding, as performed by Unicycler) to be less reliable than long-read-first hybrid assembly (long-read assembly followed by short-read polishing). However, in cases where short reads are deep but long reads are shallow (not tested in this study), Unicycler is likely to perform better, as this was the case it was designed for [21].

Conclusions

The goal of any assembly approach is to produce a representation of the underlying genome with the fewest errors. Assuming there is a single, unambiguous underlying genome (i.e., no heterogeneity), the ideal result is a base-for-base exact match of the genome: a perfect assembly. Our study shows that for bacterial genomes, a Trycycler+ Medaka+Pilon approach can deliver assemblies which are very close to this goal: approximately one error per 2 Mbp, equivalent to two errors in an *E. coli* genome. Future improvements in sequencing technologies, basecalling, and assembly/polishing algorithms may make perfect bacterial assemblies a reality, and only when this is reliably achievable can we truly call bacterial genome assembly a "solved problem."

Methods

Starting gene database

To generate Trycycler's database of preferred contig-starting gene sequences, we produced consensus sequences of common genes at the start of complete contigs on RefSeq. All complete bacterial genomes on RefSeq were downloaded, and the name of the first gene in each contig was extracted. These names were tallied and sorted to produce a list of common starting gene names, e.g., "Chromosomal replication initiator protein DnaA" and "Replication initiation protein." The gene sequences with these names were extracted and clustered using complete-linkage hierarchical clustering (coverage threshold of 100% and sequence identity threshold of 95%). We then produced an ancestral state reconstruction consensus sequence for each cluster using MUSCLE [29], FastTree [37], and TreeTime [38] to generate the final set of 7171 contig starting sequences.

Simulated-read tests

One reference genome was used from each of the 10 most common bacterial species in RefSeq: *Escherichia coli*, *Salmonella enterica*, *Staphylococcus aureus*, *Streptococcus pneumoniae*, *Klebsiella pneumoniae*, *Mycobacterium tuberculosis*, *Pseudomonas aeruginosa*, *Listeria monocytogenes*, *Neisseria meningitidis*, and *Campylobacter jejuni* (Additional file 2: Genomes). Badread (v0.1.5) was used to simulate a long-read set for each genome [32]. The parameters (read length, read accuracy, chimera rate, etc.) were varied between sets to test a variety of inputs. To ensure assemblability, all read sets were 100× depth or greater and the mean read length was longer than the longest repeat in the genome (as determined by a self-vs-self MUMmer alignment [39]). To simulate short reads for each genome, we used ART (v2016-06-05) and the parameters (simulation profile, depth, read length, and fragment length) were varied between genomes [40]. Simulation parameters and summary statistics for each simulated read set are in Additional file 2: Read simulation. Before assembly, we conducted quality-control filtering using fastp v0.20.1 [41] for short reads (using default parameters) and Filtlong [42] v0.2.0 for long reads (using a minimum read length of 1 kbp and a kept-base percentage of 95%). Simulated reads are available in Supplementary data.

Unicycler [21] (v0.4.8) assemblies were conducted on each hybrid (short and long) read set using the --no_correct option to disable read error correction because the documentation for SPAdes [43] (Unicycler's underlying assembler) recommends disabling read error correction for high-depth whole genome bacterial reads. Miniasm/Minipolish [4] (v0.3/v0.1.3), Raven [11] (v1.2.2), and Flye [10] (v2.7.1) assemblies were conducted on each long-read set using default parameters for each. Trycycler assemblies were performed using default parameters and following the procedure outlined in the Trycycler documentation (12 input assemblies made from subsampled read sets of 50× depth). Versions of Flye assemblies with repaired start/end indels were produced by manually comparing the Flye sequence to the reference genome sequence. All long-read-only assemblies were then polished with Bowtie2 [44] (v2.3.4.1) and Pilon [31] (v1.23). For Bowtie2 read alignment, we set min/max fragment lengths using values from the Unicycler assembly log (1st and 99th fragment size percentiles). We conducted multiple rounds of Bowtie2+Pilon polishing, stopping when it ceased to make any changes or at five rounds, whichever came first. See Supplementary data for the exact assembly and polishing commands used. Complete chromosomal assembly was assessed by a manual inspection of the assembly graphs and looking for an appropriately sized circular contig. All simulated read assemblies produced a single chromosomal contig with one exception: the Unicycler assembly for the *N. meningitidis* genome. However, the Unicycler assembly graph for *N. meningitidis* contained a single unbranching loop, so we merged the resulting contigs to produce a single chromosomal sequence.

To quantify the accuracy of the assemblies, we manually extracted the chromosomal contig from each assembly's graph. We then made the contig consistent with the reference sequence by normalizing the strand (changing the sequence to its reverse complement if necessary) and starting position (moving bases from the beginning of the contig to the end) to match the reference genome. The pairwise_align.py script (available in Supplementary data) was then used to perform a global sequence alignment between each contig and its reference sequence using the edlib library [26]. From this alignment, we produced two metrics: the mean sequence identity (the number of matching bases divided by the full alignment length) and the worst-100-bp identity (the minimum number of matching bases in a 100-bp sliding window over the alignment). We then used the error_positions.py script (available in Supplementary data) to identify the position, type, and size of each assembly error and quantify the accuracy in repeat and non-repeat sequences.

Real-read tests

The seven bacterial isolates used in this study each belong to a different species: *Acinetobacter baumannii*, *Citrobacter koseri*, *Enterobacter kobei*, an unnamed *Haemophilus* species (given the placeholder name *Haemophilus sp002998595* in GTDB R202 [45, 46]), *Klebsiella oxytoca*, *Klebsiella variicola*, and *Serratia marcescens*. Isolates were cultured overnight at 37°C in Luria-Bertani broth and DNA was extracted using GenFind v3 according to the manufacturer's instructions (Beckman Coulter). The same DNA extract was used to sequence each isolate using three different approaches: ONT ligation, ONT rapid, and Illumina. For ONT ligation, we followed the protocol for the SQK-LSK109 ligation sequencing kit and EXP-NBD104 native barcoding expansion (Oxford Nanopore Technologies). For ONT rapid, we followed the protocol for the SQK-RBK004 rapid barcoding kit (Oxford Nanopore Technologies). All ONT libraries were sequenced on MinION R9.4.1 flow cells. For Illumina, we followed a modified Illumina DNA Prep protocol (catalogue number 20018705), whereby the reaction volumes were quartered to conserve reagents. Illumina libraries were sequenced on the NovaSeq 6000 using SP reagent kits v1.0 (300 cycles, Illumina Inc.), producing 150-bp paired-end reads with a mean insert size of 331 bp. All ONT read sets were basecalled and demultiplexed using Guppy v3.6.1. The resulting Illumina read pairs were shuffled and evenly split into two separate read sets. We then produced two non-overlapping hybrid read sets (A and B) for each genome. Read set A consisted of the ONT ligation reads plus half of the Illumina reads. Read set B consisted of the ONT rapid reads plus the other half of the Illumina reads. All reads are available in Supplementary data.

Read sets A and B for each isolate (14 total read sets) were subjected to the same read QC and assembly methods as were used for the simulated read sets, to generate long-read-only and hybrid assemblies for comparison. Versions of Flye assemblies with repaired start/end indels were produced by manually comparing the Flye sequence to the Trycycler+Medaka+Pilon assembly. We separately polished each contig from each long-read-only assembly using Medaka v1.3.2, using Trycycler-partitioned reads, the r941_min_high_g360 model (to match the basecalling model used) and default parameters. We then polished each long-read Medaka-polished assembly using Pilon as described above. See Supplementary data for the exact assembly and polishing commands used.

To quantify the accuracy of the resulting assemblies, we manually extracted the chromosomal contig from each, where possible. The *Serratia marcescens* 17-147-1671 read set B assemblies usually failed to produce a complete chromosomal contig (only Unicycler succeeded), so that genome was excluded from further analyses. For the six remaining genomes, we normalized all chromosomes to the same strand and starting position, then used the pairwise_align.py script (available in Supplementary data) to perform a global sequence alignment between read set A and read set B chromosomes using the edlib library [26]. From this alignment, we produced the same metrics as were used in the simulated-read tests: mean sequence identity and worst-100-bp identity. We then used the error_positions.py script (available in Supplementary data) to identify the position, type, and size of each assembly error and quantify the accuracy in repeat and non-repeat sequences.

To produce ALE scores, we aligned the full short-read set (i.e., before it was split into read sets A and B) to each assembled chromosome using Bowtie2 [44] (v2.3.4.1). The alignments were then given to ALE to produce a single likelihood score [33]. ALE analyses were done on each assembly, so generated 12 values (read sets A and B for each of the six genomes) for each assembly method. ALE scores are not an absolute metric of assembly quality, only a relative metric for comparing different assemblies of the same genome. We therefore normalized the ALE scores using the mean and standard deviation for each genome to produce ALE *z*-scores.

IDEEL analysis of genomes requires a protein database, so we download all UniProt/TrEMBL [47] release 2020_05 sequences. From this, we built a Diamond [48] (v2.0.4) index which was used by IDEEL [34]. Predicted proteins in the assembly were classified as full-length if IDEEL found them to be ≥95% the length of the best-matching known protein in the database. IDEEL analyses were done on each assembly and generated 12 values (read sets A and B for each of the six genomes) for each assembly method.

Multi-user consistency tests

Each of the five Trycycler testers was given the ONT rapid read set for the six genomes used in the real-read tests (all real genomes excluding *Serratia marcescens* 17-147-1671) and produced one Trycycler assembly (without Medaka or Pilon polishing) for each. The number of input assemblies and which assemblers were used are available in Additional file 4: Tester assemblers. We then compared the assemblies produced by single tools (Flye, Raven, and Miniasm/Minipolish), by Trycycler (from the developer and the five testers), and a hybrid-assembled reference (the developer's Trycycler+Medaka+Pilon assembly).

For each genome, we clustered the contigs from all assemblies (using Trycycler cluster), and using the developer's Trycycler assembly as the reference, we classified the genome replicons for each assembly as either present, present with misassemblies, or absent (Additional file 4: Matrix). Each chromosome was rotated to a consistent starting position and a multiple sequence alignment was performed (using Trycycler MSA). We then extracted pairwise distances from the alignment (using the msa_to_distance_matrix.py script, available in Supplementary data) and built a FastME [28] tree from the distances. The distances were then normalized to the genome size (using the normalise_distance_matrix_to_mbp.py script, available in Supplementary data) to quantify the differences between each assembled chromosome for each of the genomes.

Supplementary Information

The online version contains supplementary material available at https://doi.org/10.1186/s13059-021-02483-z.

Additional file 1. Supplementary Figs. S1–S7.
Additional file 2. Supplementary tables for the simulated-read tests.
Additional file 3. Supplementary tables for the real-read tests.
Additional file 4. Supplementary tables for the multi-user consistency tests.
Additional file 5. Review history.

Review history

The review history is available as Additional file 5.

Peer review information

Barbara Cheifet was the primary editor of this article and managed its editorial process and peer review in collaboration with the rest of the editorial team.

Authors' contributions

RRW developed the software, carried out the analyses, and drafted the manuscript. LMJ performed all culturing, DNA preparation, and sequencing. LTC, JH, GM, BV, and KW performed assemblies for the multi-user consistency tests. KEH supervised the project and oversaw funding acquisition. All authors contributed to editing and approved the final manuscript.

Funding

This work was supported, in whole or in part, by the Bill & Melinda Gates Foundation [OPP1175797]. Under the grant conditions of the Foundation, a Creative Commons Attribution 4.0 Generic License has already been assigned to the Author Accepted Manuscript version that might arise from this submission. This work was also supported by an Australian Government Research Training Program Scholarship, and KEH is supported by a Senior Medical Research Fellowship from the Viertel Foundation of Victoria. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Availability of data and materials

Trycycler source code and documentation can be found at github.com/rrwick/Trycycler (GPLv3 license) [49]. Supplementary figures, tables, and code can be found at github.com/rrwick/Trycycler-paper (GPLv3 license) [50]. Reads, assemblies, and reference sequences can be found at bridges.monash.edu/articles/dataset/Trycycler_paper_dataset/14890734 (CC BY 4.0 license) [51].

Declarations

Ethics approval and consent to participate

Not applicable

Competing interests

The authors declare that they have no competing interests.

Author details

[1]Department of Infectious Diseases, Central Clinical School, Monash University, Melbourne, VIC 3004, Australia.
[2]Cambridge Baker Systems Genomics Initiative, Baker Heart & Diabetes Institute, Melbourne, VIC 3004, Australia.
[3]Department of Infection Biology, London School of Hygiene & Tropical Medicine, WC1E 7HT, London, UK.

Received: 8 July 2021 Accepted: 31 August 2021
Published online: 14 September 2021

References

1. Taylor TL, Volkening JD, DeJesus E, Simmons M, Dimitrov KM, Tillman GE, et al. Rapid, multiplexed, whole genome and plasmid sequencing of foodborne pathogens using long-read nanopore technology. Sci Rep. 2019;9(1):1–11. https://doi.org/10.1038/s41598-019-52424-x.
2. Elliott I, Batty EM, Ming D, Robinson MT, Nawtaisong P, De Cesare M, et al. Oxford nanopore MinION sequencing enables rapid whole genome assembly of *Rickettsia typhi* in a resource-limited setting. Am J Trop Med Hyg. 2020;102(2): 408–14. https://doi.org/10.4269/ajtmh.19-0383.
3. Myers EW. The fragment assembly string graph. Bioinformatics. 2005;21(Suppl. 2):79–85. https://doi.org/10.1093/bioinformatics/bti1114.
4. Li H. Minimap and miniasm: fast mapping and *de novo* assembly for noisy long sequences. Bioinformatics. 2016;32(14): 2103–10. https://doi.org/10.1093/bioinformatics/btw152.
5. Jung H, Winefield C, Bombarely A, Prentis P, Waterhouse P. Tools and strategies for long-read sequencing and *de novo* assembly of plant genomes. Trends Plant Sci. 2019;24(8):700–24. https://doi.org/10.1016/j.tplants.2019.05.003.
6. Eisenstein M. Closing in on a complete human genome. Nature. 2021;590(7847):679–81. https://doi.org/10.1038/d41586-021-00462-9.
7. Loman NJ, Quick J, Simpson JT. A complete bacterial genome assembled *de novo* using only nanopore sequencing data. Nat Methods. 2015;12(8):733–5. https://doi.org/10.1038/nmeth.3444.
8. Koren S, Phillippy AM. One chromosome, one contig: complete microbial genomes from long-read sequencing and assembly. Curr Opin Microbiol. 2015;23:110–20. https://doi.org/10.1016/j.mib.2014.11.014.
9. Koren S, Walenz BP, Berlin K, Miller JR, Phillippy AM. Canu: scalable and accurate long-read assembly via adaptive k-mer weighting and repeat separation. Genome Res. 2017;27(5):722–36. https://doi.org/10.1101/gr.215087.116.
10. Kolmogorov M, Yuan J, Lin Y, Pevzner PA. Assembly of long, error-prone reads using repeat graphs. Nat Biotechnol. 2019;37(5):540–6. https://doi.org/10.1038/s41587-019-0072-8.
11. Vaser R, Šikić M. Time- and memory-efficient genome assembly with Raven. Nat Comput Sci. 2021;1(5):332–6. https://doi.org/10.1038/s43588-021-00073-4.
12. Ruan J, Li H. Fast and accurate long-read assembly with wtdbg2. Nat Methods. 2020;17(2):155–8. https://doi.org/10.1038/s41592-019-0669-3.
13. Wick RR, Holt KE. Benchmarking of long-read assemblers for prokaryote whole genome sequencing. F1000Research. 2019;8(2138). https://doi.org/10.12688/f1000research.21782.1.
14. Eisenstein M. An ace in the hole for DNA sequencing. Nature. 2017;550(7675):285–8. https://doi.org/10.1038/550285a.
15. Koren S, Rhie A, Walenz BP, Dilthey AT, Bickhart DM, Kingan SB, et al. *De novo* assembly of haplotype-resolved genomes with trio binning. Nat Biotechnol. 2018;36(12):1174–82. https://doi.org/10.1038/nbt.4277.
16. Kolmogorov M, Bickhart DM, Behsaz B, Gurevich A, Rayko M, Shin SB, et al. metaFlye: scalable long-read metagenome assembly using repeat graphs. Nat Methods. 2020;17(11):1103–10. https://doi.org/10.1038/s41592-020-00971-x.
17. Goodwin S, McPherson JD, McCombie WR. Coming of age: ten years of next-generation sequencing technologies. Nat Rev Genet. 2016;17(6):333–51. https://doi.org/10.1038/nrg.2016.49.
18. Jain M, Koren S, Quick J, Rand AC, Sasani TA, Tyson JR, et al. Nanopore sequencing and assembly of a human genome with ultra-long reads. Nat Biotechnol. 2018;36(4):338–45. https://doi.org/10.1038/nbt.4060.
19. Wick RR, Judd LM, Holt KE. Performance of neural network basecalling tools for Oxford Nanopore sequencing. Genome Biol. 2019;20(1):129. https://doi.org/10.1186/s13059-019-1727-y.
20. Wick RR, Judd LM, Wyres KL, Holt KE. Recovery of small plasmid sequences via Oxford Nanopore sequencing. Microbial Genomics. 2021;7(8):1–8. https://doi.org/10.1099/mgen.0.000631.
21. Wick RR, Judd LM, Gorrie CL, Holt KE. Unicycler: resolving bacterial genome assemblies from short and long sequencing reads. PLoS Comput Biol. 2017;13(6):e1005595. https://doi.org/10.1371/journal.pcbi.1005595.
22. Wick RR, Judd LM, Gorrie CL, Holt KE. Completing bacterial genome assemblies with multiplex MinION sequencing. Microbial Genomics. 2017;3(10):1–7. https://doi.org/10.1099/mgen.0.000132.
23. Van Rossum G, Drake FL. Python 3 reference manual. Scotts Valley, CA: CreateSpace; 2009.
24. Harris CR, Millman KJ, van der Walt SJ, Gommers R, Virtanen P, Cournapeau D, et al. Array programming with NumPy. Nature. 2020;585(7825):357–62. https://doi.org/10.1038/s41586-020-2649-2.
25. Virtanen P, Gommers R, Oliphant TE, Haberland M, Reddy T, Cournapeau D, et al. SciPy 1.0: fundamental algorithms for scientific computing in Python. Nat Methods. 2020;17(3):261–72. https://doi.org/10.1038/s41592-019-0686-2.
26. Šošić M, Šikić M. Edlib: a C/C++ library for fast, exact sequence alignment using edit distance. Bioinformatics. 2017;33(9): 1394–5. https://doi.org/10.1093/bioinformatics/btw753.
27. Ondov BD, Treangen TJ, Melsted P, Mallonee AB, Bergman NH, Koren S, et al. Mash: fast genome and metagenome distance estimation using MinHash. Genome Biol. 2016;17(1):1–14. https://doi.org/10.1186/s13059-016-0997-x.

28. Lefort V, Desper R, Gascuel O. FastME 2.0: a comprehensive, accurate, and fast distance-based phylogeny inference program. Mol Biol Evol. 2015;32(10):2798–800. https://doi.org/10.1093/molbev/msv150.
29. Edgar RC. MUSCLE: multiple sequence alignment with high accuracy and high throughput. Nucleic Acids Res. 2004; 32(5):1792–7. https://doi.org/10.1093/nar/gkh340.
30. Wright C, Wykes M. Medaka [Internet]. GitHub. 2020. https://github.com/nanoporetech/medaka.
31. Walker BJ, Abeel T, Shea T, Priest M, Abouelliel A, Sakthikumar S, et al. Pilon: an integrated tool for comprehensive microbial variant detection and genome assembly improvement. PLoS One. 2014;9(11). https://doi.org/10.1371/journal.pone.0112963.
32. Wick RR. Badread: simulation of error-prone long reads. J Open Source Software. 2019;4(36):1316. https://doi.org/10.21105/joss.01316.
33. Clark SC, Egan R, Frazier PI, Wang Z. ALE: a generic assembly likelihood evaluation framework for assessing the accuracy of genome and metagenome assemblies. Bioinformatics. 2013;29(4):435–43. https://doi.org/10.1093/bioinformatics/bts723.
34. Stewart RD, Auffret MD, Warr A, Walker AW, Roehe R, Watson M. Compendium of 4,941 rumen metagenome-assembled genomes for rumen microbiome biology and enzyme discovery. Nat Biotechnol. 2019;37(8):953–61. https://doi.org/10.1038/s41587-019-0202-3.
35. Watson M, Warr A. Errors in long-read assemblies can critically affect protein prediction. Nat Biotechnol. 2019;37(2):124–6. https://doi.org/10.1038/s41587-018-0004-z.
36. Evans-roberts K, Maxwell A, Marinus MG, Løbner-Olesen A. DNA methylation. EcoSal Plus. 2014;6(1). https://doi.org/10.1128/ecosalplus.ESP-0003-2013.
37. Price MN, Dehal PS, Arkin AP. FastTree 2 - approximately maximum-likelihood trees for large alignments. PLoS One. 2010;5(3). https://doi.org/10.1371/journal.pone.0009490.
38. Sagulenko P, Puller V, Neher RA. TreeTime: maximum-likelihood phylodynamic analysis. Virus Evol. 2018;4(1):1–9. https://doi.org/10.1093/ve/vex042.
39. Kurtz S, Phillippy A, Delcher AL, Smoot M, Shumway M, Antonescu C, et al. Versatile and open software for comparing large genomes. Genome Biol. 2004;5(2):R12. https://doi.org/10.1186/gb-2004-5-2-r12.
40. Huang W, Li L, Myers JR, Marth GT. ART: a next-generation sequencing read simulator. Bioinformatics. 2012;28(4):593–4. https://doi.org/10.1093/bioinformatics/btr708.
41. Chen S, Zhou Y, Chen Y, Gu J. Fastp: an ultra-fast all-in-one FASTQ preprocessor. Bioinformatics. 2018;34(17):i884–90. https://doi.org/10.1093/bioinformatics/bty560.
42. Wick RR. Filtlong [Internet]. GitHub. 2018. https://github.com/rrwick/Filtlong.
43. Bankevich A, Nurk S, Antipov D, Gurevich AA, Dvorkin M, Kulikov AS, et al. SPAdes: a new genome assembly algorithm and its applications to single-cell sequencing. J Comput Biol. 2012;19(5):455–77. https://doi.org/10.1089/cmb.2012.0021.
44. Langmead B, Salzberg SL. Fast gapped-read alignment with Bowtie 2. Nat Methods. 2012;9(4):357–9. https://doi.org/10.1038/nmeth.1923.
45. Chaumeil PA, Mussig AJ, Hugenholtz P, Parks DH. GTDB-Tk: a toolkit to classify genomes with the genome taxonomy database. Bioinformatics. 2020;36(6):1925–7. https://doi.org/10.1093/bioinformatics/btz848.
46. Parks DH, Chuvochina M, Chaumeil PA, Rinke C, Mussig AJ, Hugenholtz P. A complete domain-to-species taxonomy for Bacteria and Archaea. Nat Biotechnol. 2020;38(9):1079–86. https://doi.org/10.1038/s41587-020-0501-8.
47. The UniProt Consortium. UniProt: a worldwide hub of protein knowledge. Nucleic Acids Res. 2019;47(D1):D506–15. https://doi.org/10.1093/nar/gky1049.
48. Buchfink B, Xie C, Huson DH. Fast and sensitive protein alignment using DIAMOND. Nat Methods. 2015;12(1). https://doi.org/10.1038/nmeth.3176.
49. Wick RR, Judd LM, Cerdeira LT, Hawkey J, Méric G, Vezina B, et al. Trycycler. GitHub. 2021. https://doi.org/10.5281/zenodo.4620349.
50. Wick RR, Judd LM, Cerdeira LT, Hawkey J, Méric G, Vezina B, et al. Trycycler paper. GitHub. 2021. https://doi.org/10.5281/zenodo.5279701.
51. Wick RR, Judd LM, Cerdeira LT, Hawkey J, Méric G, Vezina B, et al. Trycycler paper dataset. Bridges. 2021. https://doi.org/10.26180/14890734.v2.

Publisher's Note