Hybracter: enabling scalable, automated, complete and accurate bacterial genome assemblies

George Bouras[1,2,*], Ghais Houtak[1,2], Ryan R. Wick[3], Vijini Mallawaarachchi[4], Michael J. Roach[4,5], Bhavya Papudeshi[4], Lousie M. Judd[3], Anna E. Sheppard[6], Robert A. Edwards[4] and Sarah Vreugde[1,2]

Abstract

Improvements in the accuracy and availability of long-read sequencing mean that complete bacterial genomes are now routinely reconstructed using hybrid (i.e. short- and long-reads) assembly approaches. Complete genomes allow a deeper understanding of bacterial evolution and genomic variation beyond single nucleotide variants. They are also crucial for identifying plasmids, which often carry medically significant antimicrobial resistance genes. However, small plasmids are often missed or misassembled by long-read assembly algorithms. Here, we present Hybracter which allows for the fast, automatic and scalable recovery of near-perfect complete bacterial genomes using a long-read first assembly approach. Hybracter can be run either as a hybrid assembler or as a long-read only assembler. We compared Hybracter to existing automated hybrid and long-read only assembly tools using a diverse panel of samples of varying levels of long-read accuracy with manually curated ground truth reference genomes. We demonstrate that Hybracter as a hybrid assembler is more accurate and faster than the existing gold standard automated hybrid assembler Unicycler. We also show that Hybracter with long-reads only is the most accurate long-read only assembler and is comparable to hybrid methods in accurately recovering small plasmids.

Impact Statement

Complete bacterial genome assembly using hybrid sequencing is a routine and vital part of bacterial genomics, especially for identification of mobile genetic elements and plasmids. As sequencing becomes cheaper, easier to access and more accurate, automated assembly methods are crucial. With Hybracter, we present a new long-read first automated assembly tool that is faster and more accurate than the widely used Unicycler. Hybracter can be used both as a hybrid assembler and with long-reads only. Additionally, it solves the problems of long-read assemblers struggling with small plasmids, with plasmid recovery from long-reads only performing on par with hybrid methods. Hybracter can natively exploit the parallelization of high-performance computing clusters and cloud-based environments, enabling users to assemble hundreds or thousands of genomes with one line of code. Hybracter is available freely as source code on GitHub, via Bioconda or PyPi.

Received 08 January 2024; Accepted 16 April 2024; Published 08 May 2024
Author affiliations: [1]Adelaide Medical School, Faculty of Health and Medical Sciences, The University of Adelaide, Adelaide, Australia; [2]The Department of Surgery – Otolaryngology Head and Neck Surgery, University of Adelaide and the Basil Hetzel Institute for Translational Health Research, Central Adelaide Local Health Network, Adelaide, South Australia, Australia; [3]Department of Microbiology and Immunology, University of Melbourne at the Peter Doherty Institute for Infection and Immunity, Melbourne, Australia; [4]Flinders Accelerator for Microbiome Exploration, College of Science and Engineering, Flinders University, Adelaide, Australia; [5]Adelaide Centre for Epigenetics and South Australian Immunogenomics Cancer Institute, The University of Adelaide, Adelaide, Australia; [6]School of Biological Sciences, The University of Adelaide, Adelaide, Australia.
***Correspondence:** George Bouras, george.bouras@adelaide.edu.au
Keywords: assembly; long-reads; plasmids.
Abbreviations: CDS, coding sequence; DBG, de Bruijn graph; HPC, high-performace computing; InDel, insertion and deletion; SNV, single nucleotide variant.
Data statement: Three supplementary figures and 13 supplementary tables are available with the online version of this article.

DATA SUMMARY

(1) Hybracter is developed using Python and Snakemake as a command-line software tool for Linux and MacOS systems.
(2) Hybracter is freely available under an MIT licence on GitHub (https://github.com/gbouras13/hybracter) and the documentation is available at Read the Docs (https://hybracter.readthedocs.io/en/latest/).
(3) Hybracter is available to install via PyPI (https://pypi.org/project/hybracter/) and Bioconda (https://anaconda.org/bioconda/hybracter). A Docker/Singularity container is also available at https://quay.io/repository/gbouras13/hybracter.
(4) All code used to benchmark Hybracter, including the reference genomes, is publicly available on GitHub (https://github.com/gbouras13/hybracter_benchmarking) with released DOI https://zenodo.org/doi/10.5281/zenodo.10910108 available at Zenodo.
(5) The subsampled FASTQ files used for benchmarking are publicly available at Zenodo with DOI https://doi.org/10.5281/zenodo.10906937.
(6) All super accuracy simplex ATCC FASTQ reads sequenced as a part of this study can be found under BioProject PRJNA1042815.
(7) All Hall *et al*. fast accuracy simplex and super accuracy duplex ATCC FASTQ read files can be found in the SRA under BioProject PRJNA1087001.
(8) All raw Lermaniaux *et al*. FASTQ read files and genomes can be found in the SRA under BioProject PRJNA1020811.
(9) All *Staphylococcus aureus* JKD6159 FASTQ read files and genomes can be found under BioProject PRJNA50759.
(10) All *Mycobacterium tuberculosis* H37R2 FASTQ read files and genomes can be found under BioProject PRJNA836783.
(11) The complete list of BioSample accession numbers for each benchmarked sample can be found in Table S1, available in the online version of this article.
(12) The benchmarking assembly output files are publicly available on Zenodo with DOI https://doi.org/10.5281/zenodo.10906937.
(13) All Pypolca benchmarking outputs and code are publicly available on Zenodo with DOI https://zenodo.org/doi/10.5281/zenodo.10072192.

INTRODUCTION

Reconstructing complete bacterial genomes using *de novo* assembly methods had been considered too costly and time-consuming to be widely recommended in most cases, even as recently as 2015 [1]. This was due to the reliance on short-read sequencing technologies, which does not allow for reconstructing regions with repeats and extremely high GC content [2]. However, since then, advances in long-read sequencing technologies have allowed for the automatic construction of complete genomes using hybrid assembly approaches. Originally, this involved starting with a short-read assembly followed by scaffolding the repetitive and difficult to resolve regions with long-reads [3, 4]. This approach was implemented in the command-line tool Unicycler, which remains the most popular tool for generating complete bacterial genome assemblies [5]. As long-read sequencing has improved in accuracy and availability, with the latest Oxford Nanopore Technologies reads recently reaching Q20 (99%+) median accuracy, a long-read first assembly approach supplemented by short-read polishing has recently been favoured for recovering accurate complete genomes. Long-read first approaches provide greater accuracy and contiguity than short-read first approaches in difficult regions [6–11]. The current gold standard manual assembly tool Trycycler even allows for the potential recovery of perfect genome assemblies [7]. However, Trycycler requires significant microbial bioinformatics expertise and involves manual decisionmaking, creating a significant barrier to useability, scalability and automation [12].

Several tools exist that generate automated long-read first genome assemblies, such as MicroPIPE [13], ASA3P [14], Bactopia [15] and Dragonflye [16]. However, these tools do not consider factors such as genome reorientation [17] and recent polishing best-practices [18], and often contain the assembly workflow as a sub-module within a more expansive end-to-end pipeline. Additionally, none of the existing tools consider the targeted recovery of plasmids. As long-read assemblers struggle particularly with small plasmids, this leads to incorrectly recovered or missing plasmids in bacterial assemblies [19].

We introduce Hybracter, a new command-line tool for automated near-perfect long-read first complete bacterial genome assembly. It implements a comprehensive and flexible workflow allowing for long-read assembly polished with long- and short-reads (with subcommand 'hybracter hybrid' for one or more samples and subcommand 'hybracter hybrid-single' for a single sample) or long-read only assembly polished with long-reads (with subcommand 'hybracter long' for one or more samples and subcommand 'hybracter long-single' for a single sample) (Table 1). For ease of use and familiarity, Hybracter has been designed with a command-line interface containing parameters similar to Unicycler. Additionally, thanks to its Snakemake [20] and Snaketool [21] implementation, Hybracter seamlessly scales from a single isolate to hundreds or thousands of genomes with high computational efficiency and supports deployment on high-performance computing (HPC) clusters and cloud-based environments.

Table 1. Summary of the four primary Hybracter commands

Command	Input	No. of samples	Description	Workflow elements included by default (from Fig. 1)
Hybracter hybrid	Five-column csv sample sheet specified with '--input' containing: • sample name • long-read FASTQ path • estimated chromosome length • R1 short-read FASTQ path • R2 short-read FASTQ path	1+	Long-read first assembly followed by long- then short-read polishing for multiple isolates. Snakemake implementation ensures efficient use of available resources	a, b, c, d, e, f, g, h
Hybracter hybrid-single	• sample name (-s) • long-read FASTQ path (-l) • estimated chromosome length (-c) • R1 short-read FASTQ path (−1) • R2 short-read FASTQ path (−2)	1	Long-read first assembly followed by long- then short-read polishing for a single isolate. Similar command line interface to Unicycler	a, b, c, d, e, f, g, h
Hybracter long	Three-column csv sample sheet specified with '--input' containing: • sample name • long-read FASTQ path • estimated chromosome length	1+	Long-read first assembly followed by long-read polishing for multiple isolates. Snakemake implementation ensures efficient use of available resources	a (no fastp), b, c, d, e, g, h
Hybracter long-single	• sample name (-s) • long-read FASTQ path (-l) • estimated chromosome length (-c)	1	Long-read first assembly followed by long-read polishing on a single isolate.	a (no fastp), b, c, d, e, g, h

METHODS

Assembly workflow

Hybracter implements a long-read first automated assembly workflow based on current best practices [12]. The main subcommands available in Hybracter can be found in Table 1 and the workflow is outlined in Fig. 1. Hybracter begins with long-reads for all subcommands, and uses short-reads for polishing for 'Hybracter hybrid' and 'Hybracter hybrid-single' subcommands.

First, long-read input FASTQs are input and long-read sets are filtered and subsampled to a depth of 100× with Filtlong [22], which prioritizes the longest and highest quality reads, outperforming random subsampling (see Table S11). The reads also have adapters trimmed using Porechop_ABI [23], with optional contaminant removal against a host genome using modules from Trimnami (e.g. if the bacterium has been isolated from a host) [24]. Quality control of short-read input FASTQs is performed with fastp [25] (Fig. 1a). The estimated depth of the short-reads is determined using Seqkit [26].

Long-reads are then assembled with Flye [27]. If at least one contig is recovered above the cut-off '-c' chromosome length specified by the user for the sample, that sample will be denoted as 'complete'. All such contigs will then be marked as chromosomes and kept for downstream polishing and reorientation if marked as circular by Flye. If zero contigs are above the cut-off chromosome length, the assembly will be denoted as 'incomplete', and all contigs will be kept for downstream polishing (Fig. 1b).

For all complete samples, targeted plasmid assembly is then conducted using Plassembler [28] (Fig. 1c). All samples (i.e. complete and incomplete) are then polished once with Medaka [29], which can be turned off using '--no_medaka' (Fig. 1d). It is recommended to turn off Medaka using '--no_medaka' for highly accurate Q20+ read sets where Medaka has been shown to introduce false positive changes [11]. For all complete samples only, chromosome(s) marked as circular by Flye will then be reoriented to begin with the *dnaA* chromosomal replication initiator gene using Dnaapler [30]. These reoriented chromosomes are then polished for a second time with Medaka to ensure the sequence around the original chromosome breakpoint is polished.

If the user has provided short-reads with Hybracter hybrid, all sample assemblies (complete and incomplete) are then polished with Polypolish [18] followed by Pypolca [31, 32] (Fig. 1f). The exact parameters depend on the depth of short-read sequencing [31]. If the estimated short-read coverage is below 5×, only Polypolish with '--careful' is run, as Pypolca can rarely introduce false positive errors at low depths. If the estimated short-read coverage is between 5× and 25×, Polypolish with --careful parameter is

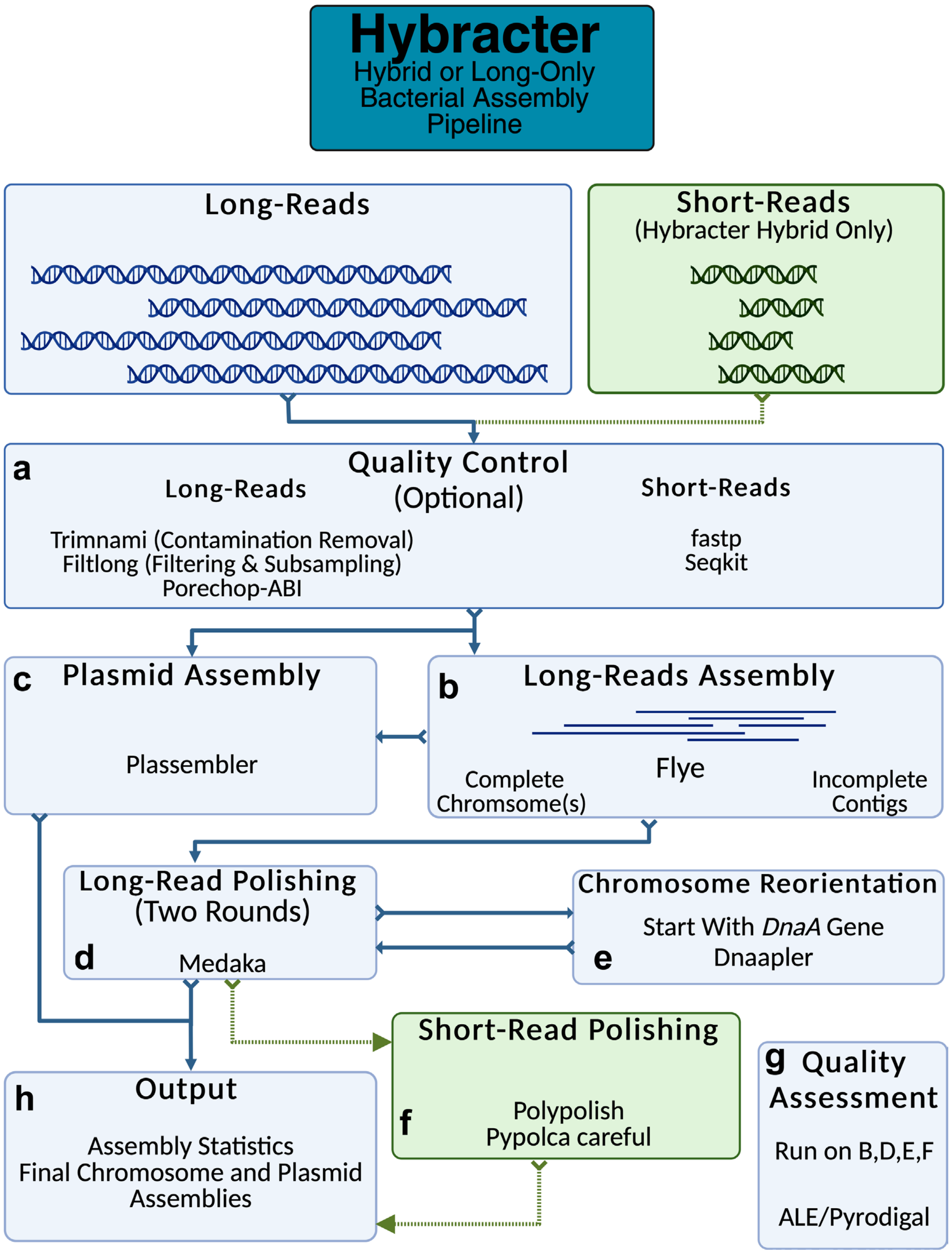

Fig. 1. Outline of the Hybracter workflow.

run followed by Pypolca with --careful parameter. Above 25× coverage, Polypolish with default parameters followed by Pypolca with --careful is run. This is because Pypolca --careful has been shown to be the best polisher at depths above 5×, and because Polypolish is able to fix potential errors in repeats Pypolca may miss. By default, the last short-read polishing round is chosen as the final assembly. Alternatively, users can choose the highest scoring polishing round according to the reference-free ALE [33] score.

Table 2. Description of the primary Hybracter output files

Output file	Description
{sample}_final.fasta	Final assembly FASTA file for the sample. Contains all chromosome(s) and plasmids for complete isolates and all contigs for incomplete isolates
{sample}_chromosome.fasta	Final assembly FASTA file for the chromosomes(s) in a complete sample
{sample}_plasmid.fasta	Final assembly FASTA file for the plasmids in a complete sample
hybracter_summary.tsv	A TSV file combining the {sample}_summary.tsv files for all samples
{sample}_summary.tsv	A TSV file containing columns denoting for the sample: • Assembly completeness • Total assembly length • Number of contigs assembled • The polishing round deemed to be most accurate and selected as the final assembly • The length of the longest contig • The estimated coverage of the longest contig • The number of circular plasmids recovered by Plassembler
{sample}_per_contig_stats.tsv	A TSV file containing columns denoting for the sample: • Contig name • Contig type (chromosome or plasmid) (complete samples only) • Contig length • Contig GC% • Contig circularity (complete samples only)

If only long-reads are available (Hybracter long), the mean coding sequence (CDS) length is calculated for each assembly using Pyrodigal [34, 35], with larger mean CDS lengths indicating a better quality assembly. The polishing round with the highest mean CDS length is chosen as the final assembly (Fig. 1g).

For each sample, a final output assembly FASTA file is created, along with per contig and overall summary statistic TSV files, as well as separate chromosome and plasmid FASTA files for samples denoted as complete (Fig. 1h). An overall 'hybracter_summary.tsv' file is also generated, which summarizes outputs for all samples. All main output files are explained in more detail in Table 2. All the main outputs can be found in the 'FINAL_OUTPUT' subdirectory, while all other intermediate output files are available in other subdirectories for users who would like extra information about their assemblies, including all assembly assessments, comparisons of all changes introduced by polishing, and Flye and Plassembler output summaries. A full list of these supplementary outputs can be found in Hybracter's Documentation (https://hybracter.readthedocs.io/en/latest/output/).

Tool selection

Tools were selected for inclusion in Hybracter either based on benchmarking from the literature, or they were specifically developed for inclusion in Hybracter. Flye [27] was chosen as the long-read assembler because it is more accurate for bacterial genome assembly than other long-read assemblers with comparable runtimes, such as Raven [36], Redbean [37] and Miniasm [38], while being dramatically faster than the comparably accurate Canu [6, 39]. Medaka [29] was chosen as the long-read polisher because of its ability to improve assembly continuity in addition to accuracy [12, 40]. The benchmarking results of this study also emphasize that it is particularly good at fixing insertion and deletion (InDel) errors, which cause problematic frameshifts and frequently lead to fractured or truncated gene predictions. However, it should be re-iterated that for modern Q20+ datasets, Medaka may introduce errors [11] and should not be used (using --no_medaka with Hybracter). Polypolish and Pypolca in various combinations depending on short-read depth were selected as short-read polishers, as these have been shown to achieve the highest performance with the lowest chance of introducing errors when used in combination [31].

We developed three standalone programs included in Hybracter. These are Dnaapler [30], Plassembler [28] and Pypolca [31]. Dnaapler was developed to ensure the chromosome(s) identified by Hybracter are reoriented to consistently begin with the *dnaA* chromosomal replication initiator gene. Full implementation details can be found in the manuscript, with expanded functionality beyond this use case [30]. Plassembler was developed to improve the runtime and accuracy when assembling plasmids in bacterial isolates. Full implementation details can be found in the manuscript for hybrid mode [28]. Hybracter long utilizes Plassembler containing a post-publication improvement for long-reads only ('Plassembler long') released in v1.3. Plassembler long assembles plasmids from only long-reads by treating long-reads as both short-reads and long-reads. Plassembler long does this by utilizing Unicycler in its pipeline to create a de Bruijn graph-based assembly, treating the long-reads as unpaired single-end reads, which are then scaffolded with the same long-read set.

The third tool is Pypolca [31, 32]. Pypolca is a Python re-implementation of the POLCA short-read genome polisher, originally created specifically for inclusion in Hybracter and with an almost identical output format and performance. Compared to POLCA, Pypolca features improved useability with a simplified command line interface, allows the user to specify an output directory and introduces a '--careful' parameter. The performance of Pypolca, and particularly Pypolca with the --careful parameter, is described in the manuscript [31].

Benchmarking

To compare Hybracter's functionality and performance, we benchmarked its performance against other software tools. We focused on the most popular state-of-the-art assembly tools for automated hybrid and long only bacterial genome assemblies. All code to replicate these analyses can be found at the repository (https://github.com/gbouras13/hybracter_benchmarking). All programs and dependency versions used for benchmarking can be found in Table S4. For the hybrid tools, we chose Unicycler and Dragonflye with both long-read and short-read polishing (denoted 'Dragonflye hybrid'). Dragonflye was chosen as it is a popular long-read first assembly pipeline [16]. Both tools were run using default parameters. By default, Dragonflye conducts a long-read assembly with Flye that is polished by Racon [41] followed by Polypolish. For the long-read only tool, we chose Dragonflye with long-read Racon-based polishing only (denoted 'Dragonflye long').

We used 30 samples for benchmarking, representing genomes from a variety of Gram-negative and Gram-positive bacteria. We chose these samples as they have real hybrid read sets in combination with manually curated genome assemblies produced using either Trycycler or Bact-builder [42], a consensus-building pipeline based on Trycycler. These samples came from five different studies below. We used the published genomes from these studies as representatives of the 'ground truth' for these samples. Where read coverage exceeded 100× samples were subsampled to approximately 100× coverage of the approximate genome size with Rasusa v0.7.0 [43], as this better reflects more realistic read depth of real life isolate sequencing. Nanoq v0.10.0 [44] was used to generate quality control statistics for the subsampled long-read sets. Four isolates did not have 100× long-read coverage – the entire long-read set was used instead. A full summary table of the read lengths, quality, Nanopore kit and base-calling models used in these studies can be found in Table S2. Hybracter v0.7.0 was used to conduct benchmarking. Medaka long-read polishing was used for all samples except the five ATCC super-accuracy model basecalled duplex read samples, where '--no_medaka' was used.

These samples contained varying levels of long-read quality (reflecting improvements in Oxford Nanopore Technologies long-read technology), with the median Q score of long-read sets ranging from 10.6 to 26.8. The five studies are:

(1) Five ATCC strain isolates (*Salmonella enterica* ATCC 10708, *Vibrio paragaemolyticus* ATCC 17802, *Escherichia coli* ATCC 25922, *Campylobacter jejuni* ATCC 33560 and *Listeria monocytogenes* ATCC-BAA-679) with R10 chemistry super-accuracy model basecalled simplex long-reads made available as a part of this study.
(2) The same five ATCC isolates with R10 chemistry fast model basecalled long-reads, and R10 chemistry super-accuracy model basecalled duplex long-reads from Hall *et al.* [45].
(3) Twelve diverse carbapenemase-producing Gram-negative bacteria from Lerminiaux *et al.* [9].
(4) *Staphylococcus aureus* JKD6159 sequenced with both R9 and R10 chemistry long-read sets from Wick *et al.* [46].
(5) *Mycobacterium tuberculosis* HR37v from Chitale *et al.* [42].

The full details for each individual isolate used can be found in Tables S1 and S2.

Chromosome accuracy

The assembly accuracy of the chromosomes recovered by each benchmarked tool was compared using Dnadiff v1.3 packaged with MUMmer v3.23 [47]. Comparisons were performed on the largest assembled contig (denoted as the chromosome) by each method, other than for *Vibrio parahaemolyticus* ATCC 17802, where the two largest contigs were chosen as it has two chromosomes.

Plasmid recovery performance and accuracy

Plasmid recovery performance for each tool was compared using the following methodology. Summary statistics are presented in Table 3. See Table S7 for a full sample-by-sample analysis. All samples were analysed using the four-step approach outlined below using summary length and GC% statistics for all contigs and the output of Dnadiff v1.3 comparisons generated for each sample and tool combination against the reference genome plasmids:

(1) The number of circularized plasmid contigs recovered for each isolate was compared to the reference genome. If the tool recovered a circularized contig homologous to that in the reference, it was denoted as completely recovered. Specifically, a contig was denoted as completely recovered if it had a genome length within 250 bp of the reference plasmid, a GC% within 0.1% of the reference plasmid and whether the Total Query Bases covered was within 250 bp of the Total Reference Bases from Dnadiff. For Dragonflye assemblies, some plasmids were duplicated or multiplicated due to known issues with the

Table 3. The total number of plasmids recovered by each tool; there were 59 total reference plasmids in the 30 samples

Tool	Complete plasmids recovered	Total plasmids partially recovered or misassembled	Total plasmids missed	Additional plasmids recovered not in reference	Samples with additional non-plasmid contigs recovered
Hybracter hybrid	65	4	0	2	10
Unicycler	60	6	3	1	2
Dragonflye hybrid	44	16	9	1	10
Hybracter long	60	5	4	2	3
Dragonflye long	44	16	9	1	10

long-read first assembly approach for small plasmids [6, 19, 48]. Any circularized contigs that were multiplicated compared to the reference plasmid were therefore denoted as misassembled.

(2) For additional circularized contigs not found in the reference recovered, these were tested for homology against the NCBI nt database using the web version of blastn [48]. If there was a hit to a plasmid, the Plassembler output within Hybracter was checked for whether the contig had a Mash hit (i.e. a Mash distance of 0.2 or lower) to plasmids in the PLSDB [49]. If there was a hit, the contig was denoted as an additional recovered plasmid. There were two in total (see Table S7 and supplementary data).

(3) Plasmids with contigs that were either not circularized but homologous to a reference plasmid, or circularized but incomplete (failing the genome length and Dnadiff criteria in 1) were denoted as partially recovered or misassembled.

(4) Reference plasmids without any homologous contigs in the assembly were denoted as missed.

Additional non-circular contigs that had no homology with reference plasmids and were not identified as plasmids in step 2 were analysed on a contig-by-contig basis and denoted as additional non-plasmid contigs (see Table S7 for contig-by-contig analysis details).

Runtime performance comparison

To compare the performance of Hybracter, we compared wall-clock runtime consumption on a machine with an Intel Core i9-13900 CPU at 5.60 GHz on a machine running Ubuntu 20.04.6 LTS with a total of 32 available threads (24 total cores). We ran all tools with eight and 16 threads and with 32 GB of memory to provide runtime metrics comparable to commonly available consumer hardware. Hybracter hybrid and long were run with 'hybracter hybrid-single' and 'hybracter long-single' for each isolate to generate a comparable per-sample runtime for comparison with the other tools. The summary results are available in Table 4 and the detailed results for each specific tool and thread combination are found in Table S8.

Table 4. Wall-clock runtime summary statistics for each tool

Tool	Type	8 Threads (h:min:s)	16 Threads (h:min:s)

Hybracter hybrid	Hybrid	Median=00:15:03
		Minimum=00:04:29
		Maximum=00:54:41
			Median=00:13:44
			Minimum=00:03:27
			Maximum=00:44:36

Dragonflye hybrid	Hybrid	Median=00:04:34
		Minimum=00:01:32
		Maximum=00:07:27
			Median=00:03:46
			Minimum=00:01:22
			Maximum=00:06:01

Unicycler	Hybrid	Median=00:50:25
		Minimum=00:12:04
		Maximum=01:13:32
			Median=00:34:10
			Minimum=00:08:36
			Maximum=00:48:23

Hybracter long	Long	Median=00:11:46
		Minimum=00:03:26
		Maximum=00:36:09
			Median=00:10:20
			Minimum=00:03:17
			Maximum=00:29:50

Dragonflye long	Long	Median=00:04:10
		Minimum=00:01:22
		Maximum=00:06:01
			Median=00:04:34
			Minimum=00:01:32
			Maximum=00:07:27

Depth analysis

To assess the effect of long-read depth on assembly accuracy, we chose *Lerminiaux* Isolate B (*Enterobacter cloacae*) and subsampled the long-read depth at each interval of 5× from 10× to 100× estimated genome size. All five tools were run on these read sets. Where a complete chromosome was assembled, Dnadiff (as described above) was used to compare the chromosome assembly to the reference.

Sequencing

DNA extraction was performed with the DNeasy Blood and Tissue kit (Qiagen). Illumina library preparation was performed using Illumina DNA prep (Illumina) according to the manufacturer's instructions. Short-read whole genome sequencing was performed n an Illumina MiSeq with a 250 bp paired-end kit. An Oxford Nanopore Technologies library preparation ligation sequencing library was prepared using the ONT SQK-NBD114-96 kit and the resultant library was sequenced using an R10.4.1 MinION flow cell (FLO-MIN114) on a MinION Mk1b device. Data were base-called with Super-Accuracy Basecalling (SUP) using the basecaller model dna_r10.4.1_e8.2_sup@v3.5.1.

Pypolca benchmarking

Pypolca v0.2.0 was benchmarked against POLCA (in MaSuRCA v4.1.0) [32] using the 18 isolates described above. These were all 12 Lerminiaux *et al*. isolates, the R10 JKD6159 isolate [46] and the five ATCC samples we sequenced as a part of this study. Benchmarking was conducted on an Intel Core i7-10700K CPU at 3.80 GHz on a machine running Ubuntu 20.04.6 LTS. All short-read FASTQs used for benchmarking are identical to those used to benchmark Hybracter. The assemblies used for polishing were intermediate chromosome assemblies from Flye v2.9.2 [50] generated within Hybracter. The outputs from Pypolca and POLCA were compared using Dnadiff v1.3 packaged with MUMmer v3.23 [47]. Overall, Pypolca and POLCA yielded extremely similar results. In total, 16/18 assemblies were identical. ATCC 33560 had two SNPs between Pypolca and POLCA and *Lerminiaux* Isolate I also had two SNPs.

RESULTS

Chromosome accuracy performance

All tools recovered complete circular contigs for each chromosome. Single nucleotide variants (SNVs), small InDels (<60 bp) and large InDels (>60 bp) were compared as a measure of assembly accuracy. To account for differences in genomic size between isolates, SNVs and small InDel counts were normalized by genome length.

The summary results are presented in Table 5 and visualized in Figs 2 and S1-3. The detailed results for each tool and sample are presented in Table S5. Of the hybrid tools, Dragonflye hybrid and Hybracter hybrid produced the fewest SNVs (both with median

Table 5. Small (<60 bp) InDels, SNVs and large (>60 bp) InDels of chromosome assemblies for all benchmarked Isolates

Tool	Type	Small InDels	SNVs	Small InDels+SNVs	Large InDels
Hybracter hybrid	Hybrid	Median=0 Minimum=0 Maximum=41	Median=0 Minimum=0 Maximum=26	Median=1 Minimum=0 Maximum=67	Total=9 Median=0 Minimum=0 Maximum=2
Dragonflye hybrid	Hybrid	Median=2.5 Minimum=0 Maximum=112	Median=0 Minimum=0 Maximum=64	Median=4.5 Minimum=0 Maximum=154	Total=70 Median=2 Minimum=0 Maximum=12
Unicycler	Hybrid	Median=11 Minimum=0 Maximum=125	Median=34 Minimum=0 Maximum=165	Median=57.5 Minimum=3 Maximum=290	Total=87 Median=1 Minimum=0 Maximum=16
Hybracter long	Long	Median=16 Minimum=1 Maximum=743	Median=21.5 Minimum=0 Maximum=156	Median=54 Minimum=1 Maximum=852	Total=11 Median=1 Minimum=0 Maximum=3
Dragonflye long	Long	Median=125 Minimum=2 Maximum=4814	Median=34.5 Minimum=0 Maximum=2172	Median=170.5 Minimum=2 Maximum=6332	Total=68 Median=2 Minimum=0 Maximum=12

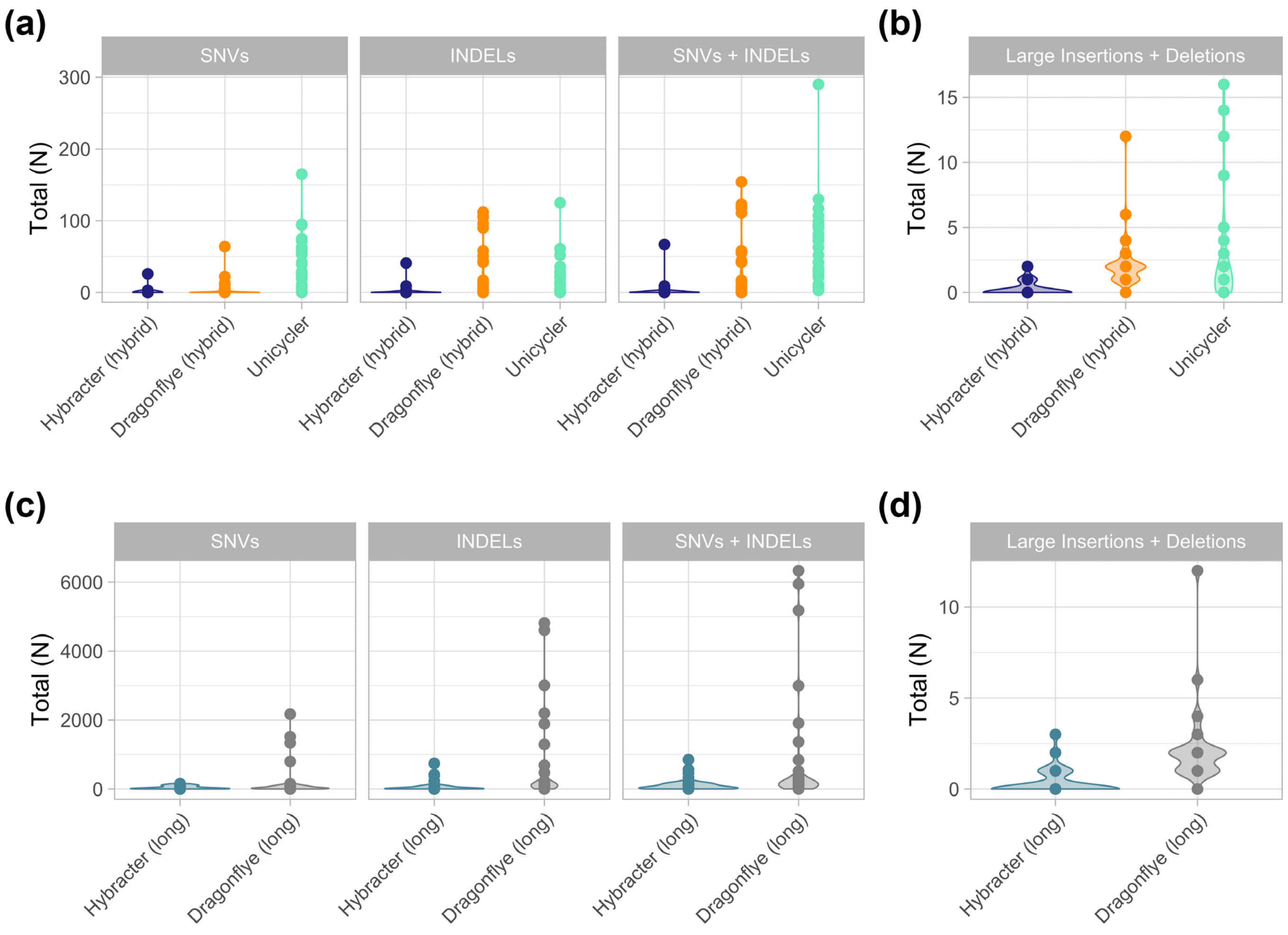

Fig. 2. Comparison of the counts of single nucleotide variants (SNVs) and small (<60 bp) insertions and deletions (InDels) (a) and the total number of large (>60 bp) InDels (b) for the hybrid tools benchmarked (Hybracter hybrid in dark blue, Dragonflye hybrid in orange and Unicycler in green). The counts of SNVs and small InDels (c) and the total number of large InDels (d) for the long tools benchmarked (Hybracter long in light blue, Dragonflye long in grey) are also shown. All data presented are from the benchmarking output run with eight threads.

0) followed by Unicycler (median 34). Hybracter hybrid produced the fewest InDels (median 0), followed by Dragonflye hybrid (median 2.5) and Unicycler (median 11). Hybracter hybrid also produced the fewest InDels plus SNVs (median 1), followed by Dragonflye hybrid (median 4.5) and Unicycler (median 57.5).

Additionally, Hybracter hybrid showed superior performance in terms of large InDels, with a median of 0 and a total of 9 large InDels across the 30 samples, compared to 2 and 70 for Dragonflye hybrid, and 1 and 87 for Unicycler.

Overall, Hybracter hybrid produced the most accurate chromosome assemblies. For 12 isolates, Hybracter assembled a perfect chromosome (Lerminiaux *et al.* [9] isolates A, B, C, D, G, H, I, J, L, *Staphylococcus aureus* JKD6159 with R10 chemistry and *L. monocytogenes* ATCC BAA-679 with simplex and duplex super-accuracy model basecalled reads).

Hybracter hybrid also produced several near-perfect assemblies (defined as <10 total SNVs plus InDels with no large insertions or deletions), including on some lower quality fast model basecalled reads (Table S5).

Similar results were found in the long-read only tool comparison. Hybracter long produced the fewest SNVs (median 21.5) compared to Dragonflye long (median 34.5). Hybracter long consistently had far fewer small InDels (median 16) and large InDels (total 11 across 30 samples) compared to Dragonflye long (median 125 and total 68 respectively). No perfect chromosomes were assembled by either long-only tool, though Hybracter long did assemble three near-perfect chromosomes (*L. monocytogenes* ATCC BAA-679 with simplex and duplex super-accuracy model basecalled reads and *Salmonella enterica* ATCC 10708 with duplex super-accuracy model basecalled reads) and several chromosomes with fewer than 50 total small InDels plus SNVs and

0 large InDels (Lerminiaux isolates A, G, H, L, J, and *Staphylococcus aureus* JKD6159 with R10 chemistry, *Salmonella enterica* ATCC 10708 with simplex super-accuracy model basecalled reads).

Overall, Hybracter long showed consistently worse performance than the hybrid tools Hybracter hybrid and Dragonflye hybrid tools (though not Unicycler) as measured by SNVs and small InDels. Combined with the lack of perfect assemblies even for duplex super-accuracy model basecalled read assemblies, this suggests the continuing utility of short-read polishing for the isolates surveyed.

Plasmid recovery performance and accuracy

Hybracter in both hybrid and long modes was superior at recovering plasmids compared to the other tools in the same class (Table 3). Hybracter hybrid was able to completely recover 65/69 possible plasmids (the other four were partially recovered), compared to 60/69 for Unicycler and only 44/69 for Dragonflye hybrid. Hybracter hybrid did not miss a single plasmid, while Unicycler missed 3/69 (all in *Klebsiella pneumoniae* Isolate E from Lerminiaux *et al.*) and Dragonflye hybrid completely missed 9/69. In terms of plasmid accuracy, Hybracter hybrid and Unicycler were similar in terms of SNVs plus small InDels, with medians of 1.62 and 2.02 per 100 kb respectively (Table S9), while Hybracter hybrid produced fewer large InDels than Unicycler (44 vs. 63 in total).

Interestingly, Hybracter long showed strong performance at recovering plasmids despite using only long-reads, completely recovering 60/69 plasmids and completely missing only 4/69. This performance was far superior to Dragonflye long (44/69 completely recovered, 9/69 missed). In terms of accuracy, both long tools were similar and unsurprisingly less accurate than the hybrid tools in terms of SNVs plus small InDels (medians of 8.74 per 100 kb for Hybracter long and 7.66 per 100 kb for Dragonflye long).

All five tools detected an additional 5411 bp plasmid in *Lerminiaux* Isolate G not found in the reference sequence and Hybracter in both hybrid and long modes detected a further 2519 bp small plasmid from this genome.

Hybracter hybrid recovers more plasmids than either Unicycler or Dragonflye because it uses a dedicated plasmid assembler, Plassembler. In addition, Hybracter long using only long-reads had an identical complete plasmid recovery rate to Unicycler, which uses both long- and short-reads (60/69 for both). These results suggest that Hybracter long, by applying algorithms designed for short-reads on long-reads, largely solves the existing difficulties of recovering small plasmids from long-reads, at least on the benchmarking dataset of predominantly R10 Nanopore reads [19, 51]. Even on the lower quality fast basecalled ATCC reads, Hybracter long performed well, with only one sample failing to produce a plasmid assembly similar to higher quality datasets (*Salmonella enterica* ATCC 10708 – see Tables S6 and S7).

Another notable result from Hybracter hybrid is that in 10/30 samples, it assembled additional non-plasmid contigs, which occurred in only 2/30 isolates for Unicycler. This is a limitation of Hybracter hybrid, as the extra sensitivity to recover plasmids comes with the cost of more false positive non-plasmid contigs that may be low-depth artefacts of sequencing. Hybracter has a 'depth_filter' parameter (defaulting to 0.25× of the chromosome depth) that filters out all non-circular putative plasmid contigs below this value.

It should be noted, however, that these contigs are not always an assembly artefact and can provide additional information regarding the quality control and similarity of short- and long-read sets. In Plassembler implemented within Hybracter hybrid, the existence of such contigs is often indicative of mismatches between long- and short-read sets [28], suggesting that there may be some heterogeneity between long- and short-reads in those samples.

Runtime performance comparison

As shown in Table 4 and Fig. 3, median wall-clock times with eight threads for Dragonflye hybrid (4 min 34 s) were smaller than Hybracter hybrid (15 min 03 s), which were in turn smaller than Unicycler (50 min 25 s). For the long-only tools, Dragonflye long (4 min 10 s) was faster than Hybracter long (11 min 46 s). Hybracter long was consistently slightly faster than Hybracter hybrid (Table 4).

The difference in runtime performance between Hybracter and Dragonflye is predominantly the result of the included targeted plasmid assembly and the reorientation and assessment steps in Hybracter that are not included in Dragonflye. Additionally, the results suggest limited benefits to running Hybracter with more than eight threads. As explained in the following section, if a user has multiple isolates to assemble, a superior approach is to modify the configuration file specifying more efficient resource requirements for each job in Hybracter.

Parallelization allows for improved efficiency

Hybracter allows users to specify and customize a configuration file to maximize resource usage and runtime efficiency. Users can modify the desired threads, memory and time requirements for each type of job that is run within Hybracter to suit their

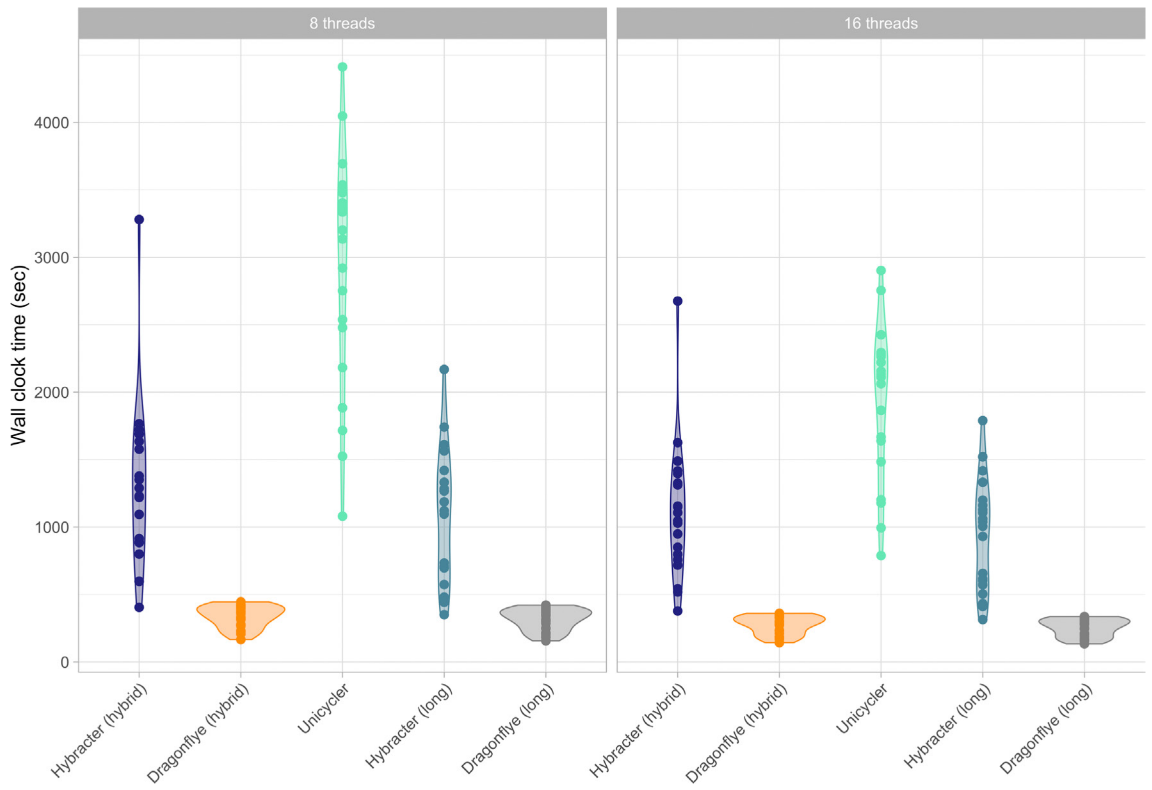

Fig. 3. Comparison of wall-clock runtime (in seconds) of Hybracter hybrid, Dragonflye hybrid, Unicycler, Hybracter long and Dragonflye long when run with eight and 16 threads.

computational resources. So that resources are not idle for most users on single sample assemblies, large jobs such as the Flye and Plassembler assembly steps default to 16 threads and 32 GB of memory.

To emphasize the efficiency benefits of parallelization, the 12 Lerminiaux *et al.* isolates were also assembled using 'hybracter hybrid' with a customized configuration file designed to improve efficiency on the machine used for benchmarking. Specifically, the configuration was changed to specify eight threads and 16 GB of memory allocated to large jobs (assembly, polishing and assessment) and four threads and 8 GB of memory allocated to medium jobs (reorientation). More details on changing Hybracter's configuration file to suit specific systems can be found in the documentation (https://hybracter.readthedocs.io/en/latest/configuration/). We limited the overall 'hybracter hybrid' run with 32 GB of memory and 16 threads to provide a fair comparison. The overall 'hybracter hybrid' run was then compared to the sum of the 12 'hybracter hybrid-single' runs. Overall, the 12 isolates took 01 h 48 min 57 s in the combined run, as opposed to 04 h 38 min 45 s from the sum of the 12 'hybracter hybrid-single' and 07 h 04 min 04 s from the sum of the 12 Unicycler runs. This inbuilt parallelization of Hybracter provides significant efficiency benefits if multiple samples are assembled simultaneously. The performance benefit of Hybracter afforded by Snakemake integration in parallel computing systems may be variable over different architectures, but this provides an example case of potential efficiency and convenience benefits.

Long-read depth does not affect hybrid assembly accuracy if a complete chromosome is assembled

Finally, we tested the effect of long-read depth on the accuracy of assemblies with all five tools at an estimated long-read depth from 10× to 100× at every interval of 5× for an example isolate (Lerminiaux Isolate B, *Enterobacter cloacae*) with super-accuracy model basecalled simplex reads (Fig. 4 and Table S12). At 10× and 15× sequencing depth, only Unicycler was able to assemble a complete chromosome. From 20× and above, all five tools were able to assemble complete chromosomes. For the hybrid tools, once a complete chromosome was assembled, increasing long-read depth had a negligible impact on accuracy results (Fig. 4). Notably, Hybracter hybrid was able to produce perfect assemblies from as low as 20× long-read depth. For long-read only tools, increasing long-read depth did affect accuracy. Increasing depth improved SNV accuracy for both Hybracter long and Dragonflye long (Fig. 4c). For small InDels, Hybracter long improved with extra depth, while Dragonflye long actually performed worse (Fig. 4a). Depth had minimal impact on large InDels (Fig. 4b).

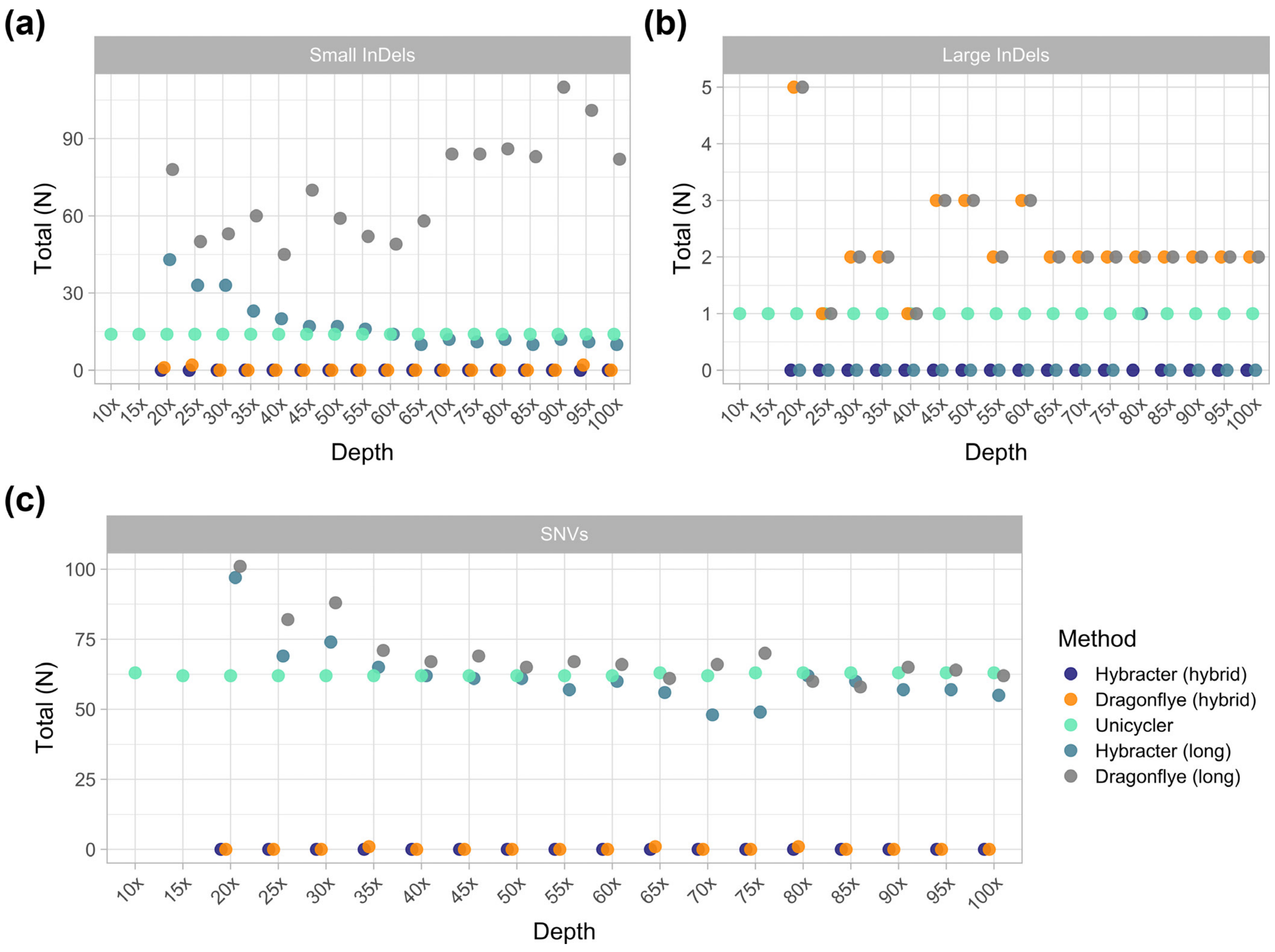

Fig. 4. Comparison of the counts of small (<60 bp) (a) and large (>60 bp) (b) insertions and deletions (InDels) and SNVs (c) for Hybracter hybrid, Dragonflye hybrid, Unicycler, Hybracter long and Dragonflye long chromosome assemblies of Lerminiaux Isolate B (*Enterobacter cloacae*) at 5× intervals of sequencing depth from 10× to 100×.

DISCUSSION

As long-read sequencing has improved in accuracy with reduced costs, it is now routine to use a combination of long- and short-reads to generate complete bacterial genomes [3, 5]. Recent advances in assembly algorithms and accuracy improvements mean that a long-read first hybrid assembly should be favoured with short-reads being used after assembly for polishing [12], as opposed to the short-read first assembly approach (where long-reads are only used for scaffolding a short-read assembly) utilized by the current gold standard automated assembler Unicycler. The Unicycler approach is more prone to larger scale InDel errors as well as smaller scale errors such as those caused by homopolymers or methylation motifs [6, 11, 52, 53]. Additionally, it should be noted that it is already possible (while perhaps not routine) to generate perfect hybrid bacterial genome assemblies using manual consensus approaches requiring human intervention, such as Trycycler [7, 54] . While manual approaches such as Trycycler generally yield superior results to automated approaches, manually assembling many complete genomes is challenging as considerable time, resources and bioinformatics expertise are required.

The results of this study emphasize that the long-read first hybrid approach consistently yields superior assemblies than the short-read first hybrid approach and should therefore be preferred going forward. The only exception where a short-read first approach is to be preferred is where a limited depth of long-read sequencing data is available (<20× depth). In this instance, long-read first hybrid approaches may struggle to assemble a complete chromosome, while short-read first approaches like Unicycler may be able to (Fig. 4).

Interestingly, in the course of conducting benchmarking for this study, we found a large number of discrepancies between older short-read first assembled 'reference genomes' for *Staphylococcus aureus* JKD6159 [55] and the five ATCC genomes benchmarked compared to updated Trycycler long-read first references (see Table S13). The number of discrepancies ranged from 44 to 8255 across the six genomes. Therefore, we recommend that older short-read first reference genomes be updated if possible using a long-read assembly approach (such as with Trycycler).

This study also shows that automated perfect hybrid genome assemblies are already possible with Hybracter. This study and others [9, 54] also confirm that a long-read first hybrid approach remains preferable to long-read only assembly with Nanopore reads, as short-reads continue to provide accuracy improvements in polishing steps. However, it is foreseeable that short-reads will soon provide little or no accuracy improvements and will not be needed to polish long-read only assemblies to perfection. Already, perfect long-read only assemblies are possible, at least with manual intervention using Trycycler [7]. Accordingly, automated perfect bacterial genome assemblies may soon become possible from long-reads only. Hybracter also allows users to turn long-read polishing off altogether. It is already established that long-read polishing can introduce errors and make long-read only assemblies worse with highly accurate Nanopore and PacBio reads [11, 31]. Therefore, this feature may become increasingly useful as long-read sequencing continues to improve in accuracy and we recommend its use for highly accurate Q20+ long-reads.

Hybracter was created to bridge the gap from the present to the future of automated perfect hybrid and long-read only bacterial genome assemblies. The results of this study show that Hybracter in hybrid mode is both faster and more accurate than the current gold standard tool for hybrid assembly, Unicycler, and is more accurate than Dragonflye in both modes. It should be noted that if users want fast chromosome only assemblies where accuracy is not essential (for applications such as species identification or sequence typing), Dragonflye remains a good option due to its speed.

Hybracter especially excels in recovering complete plasmid genomes compared to other tools. By incorporating Plassembler, Hybracter recovers more complete plasmid genomes than Unicycler in hybrid mode. Further, Hybracter long is comparable to Unicycler and Hybracter hybrid when using long-reads only for plasmid recovery.

The high error rates of long-read sequencing technologies have prevented the application of assembly approaches designed for highly accurate short-reads, such as constructing de Bruijn graphs (DBGs) based on strings of a particular length k (k-mers) [56–58]. This resulted in bioinformaticians initially utilizing less efficient algorithms designed with long-reads in mind, such as utilizing overlap graphs in place of DBGs [27, 37, 39, 59, 60]. While DBGs have been used for long-read assembly in some applications [61–63], adoption, especially in microbial genomics, has been limited.

Although long-read first assembly methods enable complete chromosome and large plasmid reconstruction, it is well established that long-read only assemblers struggle to assemble small (<20 kbp) plasmids accurately, often leading to missing or multiplicated assemblies [6, 51]. These errors may be exacerbated if ligation chemistry-based sequencing kits are used [51]. Therefore, hybrid DBG based short-read first assemblies are traditionally recommended for plasmid recovery [12].

Implemented in our post-publication changes to Plassembler described in this study, Hybracter solves the problem of small plasmid recovery using long-reads. It achieves this by implementing a DBG-based assembly approach with Unicycler. The same read set is used twice, first as unpaired pseudo 'short'-reads and then as long-reads; the long-read set scaffolds a DBG-based assembly based on the same read set. This study demonstrates that current long-read technologies, such as R10 Nanopore reads, are now accurate enough that some short-read algorithms are applicable. Our results also suggest that similar DBG-based algorithmic approaches could be used to enhance the recovery of small replicons in long-read datasets beyond the use case presented here of plasmids in bacterial isolate assemblies. This could potentially enhance the recovery of replicons such as bacteriophages [64] or other small contigs from metagenomes using only long-reads [10, 50].

Finally, consistent and resource-efficient assemblies that are as accurate as possible in recovering both plasmids and chromosomes are crucial, particularly for larger studies investigating plasmid epidemiology and evolution. Antimicrobial resistance genes carried on plasmids can have complicated patterns of transmission involving horizontal transfer between different bacterial species and lineages, transfer between different plasmid backbones, and integration into and excision from the bacterial chromosome [65–67]. Accurate plasmid assemblies are crucial in genomic epidemiology studies investigating transmission of antimicrobial-resistant bacteria within outbreak settings, as well as in a broader One Health context, where hundreds or even thousands of assemblies may be analysed [68–71]. Hybracter will facilitate the expansion of such studies, allowing for faster and more accurate automated complete genome assemblies than existing tools. Additionally, by utilizing Snakemake [20] with a Snaketool [21] command line interface, Hybracter is easily and efficiently parallelized to optimize available resources over various large-scale computing architectures. Individual jobs (such as each assembly, reorientation, polishing or assessment step) within Hybracter are automatically sent to different resources on an HPC cluster using the HPC's job scheduling system like Slurm [72]. Hybracter can natively use any Snakemake-supported cloud-based deployments such as Kubernetes, Google Cloud Life Sciences, Tibanna and Azure Batch.

CONCLUSION

Hybracter is substantially faster than the current gold standard automated tool Unicycler, assembles chromosomes more accurately than existing methods and is superior at recovering complete plasmid genomes. By applying DBG-based algorithms designed for short-reads on current generation long-reads, Hybracter long also solves the problem of long-read only assemblers entirely missing or duplicating small circular elements such as plasmids. Hybracter is resource efficient and natively supports deployment on HPC clusters and cloud environments for massively parallel analyses. We believe Hybracter will prove to be an extremely useful tool for the automated recovery of complete bacterial genomes from hybrid and long-read only sequencing data suitable for massive datasets.

Funding information
G.H. was supported by The University of Adelaide International Scholarships and a THRF Postgraduate Top-up Scholarship. A.E.S. was supported by a University of Adelaide Barbara Kidman Women's Fellowship. R.A.E. was supported by an award from the NIH NIDDK RC2DK116713 and an award from the Australian Research Council DP220102915. S.V. was supported by a Passe and Williams Foundation senior fellowship.

Acknowledgements
This work was supported with supercomputing resources provided by the Phoenix HPC service at the University of Adelaide. We would particularly like to thank Fabien Voisin for his integral role in maintaining and running Phoenix. We would also like to thank Brad Hart for useful comments in testing Hybracter and Simone Pignotti, Yu Wan and Oliver Schwengers for providing helpful comments and GitHub pull requests.

Conflicts of interest
The authors declare that there are no conflicts of interest.

References

1. Land M, Hauser L, Jun S-R, Nookaew I, Leuze MR, *et al*. Insights from 20 years of bacterial genome sequencing. *Funct Integr Genomics* 2015;15:141–161.
2. Goldstein S, Beka L, Graf J, Klassen JL. Evaluation of strategies for the assembly of diverse bacterial genomes using MinION long-read sequencing. *BMC Genomics* 2019;20:23.
3. De Maio N, Shaw LP, Hubbard A, George S, Sanderson ND, *et al*. Comparison of long-read sequencing technologies in the hybrid assembly of complex bacterial genomes. *Microb Genom* 2019;5:e000294.
4. Wick RR, Judd LM, Gorrie CL, Holt KEY. Completing bacterial genome assemblies with multiplex MinION sequencing. *Microb Genom* 2017;3:e000132.
5. Wick RR, Judd LM, Gorrie CL, Holt KE. Unicycler: resolving bacterial genome assemblies from short and long sequencing reads. *PLoS Comput Biol* 2017;13:e1005595.
6. Wick RR, Holt KE. Benchmarking of long-read assemblers for prokaryote whole genome sequencing. *F1000Res* 2019;8:2138.
7. Wick RR, Judd LM, Cerdeira LT, Hawkey J, Méric G, *et al*. Trycycler: consensus long-read assemblies for bacterial genomes. *Genome Biol* 2021;22:266.
8. Wick R. ONT-only accuracy with R10.4.1. Ryan Wick's bioinformatics blog; 2023. https://rrwick.github.io/2023/05/05/ont-only-accuracy-with-r10.4.1.html https://doi.org/10.5281/zenodo.7898220
9. Lerminiaux N, Fakharuddin K, Mulvey MR, Mataseje L. 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. *Can J Microbiol* 2024.
10. Sereika M, Kirkegaard RH, Karst SM, Michaelsen TY, Sørensen EA, *et al*. 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. *Nat Methods* 2022;19:823–826.
11. Wick R. ONT-only accuracy: 5 kHz and Dorado. Ryan Wick's bioinformatics blog; 2023. https://rrwick.github.io/2023/10/24/ont-only-accuracy-update.html https://doi.org/10.5281/zenodo.10038672
12. Wick RR, Judd LM, Holt KE. Assembling the perfect bacterial genome using Oxford Nanopore and Illumina sequencing. *PLoS Comput Biol* 2023;19:e1010905.
13. Murigneux V, Roberts LW, Forde BM, Phan M-D, Nhu NTK, *et al*. MicroPIPE: validating an end-to-end workflow for high-quality complete bacterial genome construction. *BMC Genomics* 2021;22:474.
14. Schwengers O, Hoek A, Fritzenwanker M, Falgenhauer L, Hain T, *et al*. ASA3P: an automatic and scalable pipeline for the assembly, annotation and higher-level analysis of closely related bacterial isolates. *PLoS Comput Biol* 2020;16:e1007134.
15. Petit RA, Read TD. Bactopia: a flexible pipeline for complete analysis of bacterial genomes. *mSystems* 2020;5:e00190-20.
16. Petit III RA. *Dragonflye: Assemble Bacterial Isolate Genomes from Nanopore Reads*.
17. Hunt M, Silva ND, Otto TD, Parkhill J, Keane JA, *et al*. Circlator: automated circularization of genome assemblies using long sequencing reads. *Genome Biol* 2015;16:294.
18. Wick RR, Holt KE. Polypolish: short-read polishing of long-read bacterial genome assemblies. *PLoS Comput Biol* 2022;18:e1009802.
19. Johnson J, Soehnlen M, Blankenship HM. Long read genome assemblers struggle with small plasmids. *Microb Genom* 2023;9:001024.
20. Mölder F, Jablonski KP, Letcher B, Hall MB, Tomkins-Tinch CH, *et al*. Sustainable data analysis with Snakemake. *F1000Res* 2021;10:33.
21. Roach MJ, Pierce-Ward NT, Suchecki R, Mallawaarachchi V, Papudeshi B, *et al*. Ten simple rules and a template for creating workflows-as-applications. *PLoS Comput Biol* 2022;18:e1010705.
22. Wick RR. Filtlong; 2018
23. Bonenfant Q, Noé L, Touzet H. Porechop_ABI: discovering unknown adapters in Oxford Nanopore Technology sequencing reads for downstream trimming. *Bioinform Adv* 2023;3:vbac085.
24. Roach MJ. *Trimnami: Trim Lots of Metagenomics Samples All at Once*. 2023.
25. Chen S, Zhou Y, Chen Y, Gu J. fastp: an ultra-fast all-in-one FASTQ preprocessor. *Bioinformatics* 2018;34:i884–i890.
26. Shen W, Le S, Li Y, Hu F. SeqKit: a cross-platform and ultrafast toolkit for FASTA/Q file manipulation. *PLoS One* 2016;11:e0163962.
27. Kolmogorov M, Yuan J, Lin Y, Pevzner PA. Assembly of long, error-prone reads using repeat graphs. *Nat Biotechnol* 2019;37:540–546.
28. Bouras G, Sheppard AE, Mallawaarachchi V, Vreugde S. Plassembler: an automated bacterial plasmid assembly tool. *Bioinformatics* 2023;39:btad409.
29. medaka: Sequence correction provided by ONT Research; (n.d.)
30. Bouras G, Grigson SR, Papudeshi B, Mallawaarachchi V, Roach MJ. Dnaapler: a tool to reorient circular microbial genomes. *JOSS* 2024;9:5968.
31. Bouras G, Judd LM, Edwards RA, Vreugde S, Stinear TP, *et al*. How low can you go? Short-read polishing of Oxford Nanopore bacterial genome assemblies. *Bioinformatics* 2024.

32. Zimin AV, Salzberg SL. The genome polishing tool POLCA makes fast and accurate corrections in genome assemblies. *PLoS Comput Biol* 2020;16:e1007981.
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;29:435–443.
34. Larralde M. Pyrodigal: python bindings and interface to prodigal, an efficient method for gene prediction in prokaryotes. *JOSS* 2022;7:4296.
35. Hyatt D, Chen G-L, Locascio PF, Land ML, Larimer FW, *et al*. Prodigal: prokaryotic gene recognition and translation initiation site identification. *BMC Bioinformatics* 2010;11:119.
36. Vaser R, Šikić M. Time- and memory-efficient genome assembly with Raven. *Nat Comput Sci* 2021;1:332–336.
37. Ruan J, Li H. Fast and accurate long-read assembly with wtdbg2. *Nat Methods* 2020;17:155–158.
38. Li H. Minimap and miniasm: fast mapping and de novo assembly for noisy long sequences. *Bioinformatics* 2016;32:2103–2110.
39. Koren S, Walenz BP, Berlin K, Miller JR, Bergman NH, *et al*. Canu: scalable and accurate long-read assembly via adaptive *k*-mer weighting and repeat separation. *Genome Res* 2017;27:722–736.
40. Zhang X, Liu C-G, Yang S-H, Wang X, Bai F-W, *et al*. Benchmarking of long-read sequencing, assemblers and polishers for yeast genome. *Brief Bioinformatics* 2022;23:bbac146.
41. Vaser R, Sović I, Nagarajan N, Šikić M. Fast and accurate de novo genome assembly from long uncorrected reads. *Genome Res* 2017;27:737–746.
42. Chitale P, Lemenze AD, Fogarty EC, Shah A, Grady C, *et al*. A comprehensive update to the *Mycobacterium tuberculosis* H37Rv reference genome. *Nat Commun* 2022;13:7068.
43. Hall MB. Rasusa: randomly subsample sequencing reads to a specified coverage. *JOSS* 2022;7:3941.
44. Steinig E, Coin L. Nanoq: ultra-fast quality control for nanopore reads. *JOSS* 2022;7:2991.
45. Hall MB, Wick RR, Judd LM, Nguyen ANT, Steinig EJ, *et al*. Benchmarking reveals superiority of deep learning variant callers on bacterial nanopore sequence data. *Bioinformatics* 2024.
46. Wick RR, Judd LM, Monk IR, Seemann T, Stinear TP. Improved genome sequence of Australian methicillin-resistant *Staphylococcus aureus* strain JKD6159. *Microbiol Resour Announc* 2023;12:e0112922.
47. Kurtz S, Phillippy A, Delcher AL, Smoot M, Shumway M, *et al*. Versatile and open software for comparing large genomes. *Genome Biol* 2004;5:R12.
48. Sayers EW, Bolton EE, Brister JR, Canese K, Chan J, *et al*. Database resources of the national center for biotechnology information. *Nucleic Acids Res* 2022;50:D20–D26.
49. Galata V, Fehlmann T, Backes C, Keller A. PLSDB: a resource of complete bacterial plasmids. *Nucleic Acids Res* 2019;47:D195–D202.
50. Kolmogorov M, Bickhart DM, Behsaz B, Gurevich A, Rayko M, *et al*. metaFlye: scalable long-read metagenome assembly using repeat graphs. *Nat Methods* 2020;17:1103–1110.
51. Wick RR, Judd LM, Wyres KL, Holt KEY. Recovery of small plasmid sequences via Oxford Nanopore sequencing. *Microb Genom* 2021;7:000631.
52. Marinus MG, Løbner-Olesen A. DNA Methylation. *EcoSal Plus* 2014;6:10.
53. Wick RR, Judd LM, Holt KE. Performance of neural network basecalling tools for Oxford Nanopore sequencing. *Genome Biol* 2019;20:129.
54. Sanderson ND, Kapel N, Rodger G, Webster H, Lipworth S, *et al*. Comparison of R9.4.1/Kit10 and R10/Kit12 Oxford Nanopore flowcells and chemistries in bacterial genome reconstruction. *Microb Genom* 2023;9:000910.
55. Chua K, Seemann T, Harrison PF, Davies JK, Coutts SJ, *et al*. Complete genome sequence of *Staphylococcus aureus* strain JKD6159, a unique Australian clone of ST93-IV community methicillin-resistant *Staphylococcus aureus*. *J Bacteriol* 2010;192:5556–5557.
56. Bankevich A, Nurk S, Antipov D, Gurevich AA, Dvorkin M, *et al*. SPAdes: a new genome assembly algorithm and its applications to single-cell sequencing. *J Comput Biol* 2012;19:455–477.
57. Li D, Liu C-M, Luo R, Sadakane K, Lam T-W. MEGAHIT: an ultra-fast single-node solution for large and complex metagenomics assembly via succinct de Bruijn graph. *Bioinformatics* 2015;31:1674–1676.
58. Compeau PEC, Pevzner PA, Tesler G. How to apply de Bruijn graphs to genome assembly. *Nat Biotechnol* 2011;29:987–991.
59. Wong J, Coombe L, Nikolić V, Zhang E, Nip KM, *et al*. Linear time complexity de novo long read genome assembly with GoldRush. *Nat Commun* 2023;14:2906.
60. Amarasinghe SL, Su S, Dong X, Zappia L, Ritchie ME, *et al*. Opportunities and challenges in long-read sequencing data analysis. *Genome Biol* 2020;21:30.
61. Ekim B, Berger B, Chikhi R. Minimizer-space de Bruijn graphs: whole-genome assembly of long reads in minutes on a personal computer. *Cell Syst* 2021;12:958–968.
62. Bankevich A, Bzikadze AV, Kolmogorov M, Antipov D, Pevzner PA. Multiplex de Bruijn graphs enable genome assembly from long, high-fidelity reads. *Nat Biotechnol* 2022;40:1075–1081.
63. Lin Y, Yuan J, Kolmogorov M, Shen MW, Chaisson M, *et al*. Assembly of long error-prone reads using de Bruijn graphs. *Proc Natl Acad Sci U S A* 2016;113:E8396–E8405.
64. Mallawaarachchi V, Roach MJ, Decewicz P, Papudeshi B, Giles SK, *et al*. Phables: from fragmented assemblies to high-quality bacteriophage genomes. *Bioinformatics* 2023;39:btad586.
65. Mathers AJ, Stoesser N, Chai W, Carroll J, Barry K, *et al*. Chromosomal integration of the *Klebsiella pneumoniae* carbapenemase gene, bla_{KPC}, in *Klebsiella* species is elusive but not rare. *Antimicrob Agents Chemother* 2017;61:e01823-16.
66. Houtak G, Bouras G, Nepal R, Shaghayegh G, Cooksley C, *et al*. The intra-host evolutionary landscape and pathoadaptation of persistent *Staphylococcus aureus* in chronic rhinosinusitis. *Microb Genom* 2023;9:001128.
67. Sheppard AE, Stoesser N, Wilson DJ, Sebra R, Kasarskis A, *et al*. Nested Russian doll-like genetic mobility drives rapid dissemination of the carbapenem resistance gene blaKPC. *Antimicrob Agents Chemother* 2016;60:3767–3778.
68. Hawkey J, Wyres KL, Judd LM, Harshegyi T, Blakeway L, *et al*. ESBL plasmids in *Klebsiella pneumoniae*: diversity, transmission and contribution to infection burden in the hospital setting. *Genome Med* 2022;14:97.
69. Matlock W, Lipworth S, Chau KK, AbuOun M, Barker L, *et al*. Enterobacterales plasmid sharing amongst human bloodstream infections, livestock, wastewater, and waterway niches in Oxfordshire, UK. *Elife* 2023;12:e85302.
70. Roberts LW, Enoch DA, Khokhar F, Blackwell GA, Wilson H, *et al*. Long-read sequencing reveals genomic diversity and associated plasmid movement of carbapenemase-producing bacteria in a UK hospital over 6 years. *Microb Genom* 2023;9:001048.
71. Lerminiaux N. Plasmid genomic epidemiology of blaKPC carbapenemase-producing *Enterobacterales* in Canada, 2010–2021. *Antimicrob Agents Chemother*2023:e00860-23.
72. Yoo AB, Jette MA, Grondona M. (eds). *SLURM: Simple Linux Utility for Resource Management. in Job Scheduling Strategies for Parallel Processing*. Berlin, Heidelberg: Springer, 2003. pp. 44–60.