ARL-MR-1153 ● APR 2026

Development of a Genomic Analysis Pipeline for Use on DoD High-Performance Computing Resources

by Margaret M. Hurley

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

ARL-MR-1153 ● APR 2026

Development of a Genomic Analysis Pipeline for Use on DoD High-Performance Computing Resources

Margaret M. Hurley
DEVCOM Army Research Laboratory

REPORT DOCUMENTATION PAGE

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED	
		START DATE	END DATE
April 2026	Memorandum Report	October 1, 2022	September 30, 2025

4. TITLE AND SUBTITLE
Development of a Genomic Analysis Pipeline for Use on DoD High-Performance Computing Resources

5a. CONTRACT NUMBER	5b. GRANT NUMBER	5c. PROGRAM ELEMENT NUMBER
5d. PROJECT NUMBER TRANSFORME ERP	**5e. TASK NUMBER**	**5f. WORK UNIT NUMBER**

6. AUTHOR(S)
Margaret M. Hurley

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
DEVCOM Army Research Laboratory ATTN: FCDD-RLA-HD Aberdeen Proving Ground, MD 21005	ARL-MR-1153

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
DISTRIBUTION STATEMENT A. Approved for public release: distribution unlimited.

13. SUPPLEMENTARY NOTES
ORCID: Margaret M. Hurley, 0000-0003-1239-0482

14. ABSTRACT
Increased use of next-generation sequencing in research has led to a concomitant growth in available software to analyze this data. We rely heavily on high-performance computing resources for this work and discuss suitable software options in each step of the analysis pipeline. Multiple options are provided for each step to take advantage of strengths of the algorithms for the disparate use cases (e.g., prokaryotic vs. eukaryotic organisms) studied during the project. We pay particular attention to the genomic assembly algorithms and the heart of the pipeline and provide a detailed comparison between assembly results from our sample sets.

15. SUBJECT TERMS
Biological and Biotechnology Sciences, genomics, next-generation sequencing, pipeline, high-performance computing, whole-genome assembly

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES
a. REPORT	b. ABSTRACT	C. THIS PAGE		
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UU	22

19a. NAME OF RESPONSIBLE PERSON	19b. PHONE NUMBER *(Include area code)*
Margaret M. Hurley	(520) 691-5026

STANDARD FORM 298 (REV. 5/2020)
Prescribed by ANSI Std. Z39.18

Contents

List of Figures iv

List of Tables iv

Acknowledgments v

1. Introduction 1

2. Methods 1

3. Discussion 8

4. Conclusion 9

5. References 10

List of Symbols, Abbreviations, and Acronyms 14

Distribution List 15

List of Figures

Figure 1. Assembly graph produced by Raven on exemplary dataset. 5
Figure 2. Repeat graph produced by Flye on exemplary dataset. 5
Figure 3. Close-up of assembly graph of largest contig produced by Raven on exemplary dataset..... 6
Figure 4. Close-up of repeat graph of largest contig produced by Flye on exemplary dataset..... 6
Figure 5. Visualization of progressiveMauve alignment of largest contig from Flye assembly vs. Raven assembly. 7
Figure 6. SibeliaZ alignment of largest contig from Flye assembly and Raven assembly, visualized in the dot plot produced by wgatools. 8

List of Tables

Table 1. Metrics of exemplar in-house data throughout the genomic pipeline... 3

Acknowledgments

The author thanks the DoD High Performance Computer Modernization Program for granting computer time. The author also thanks Dr. Beki Renberg for next-generation sequencing data and fruitful discussion and Drs. Meagan Small and Alex Balboa for fruitful discussion.

1. Introduction

Next-generation sequencing (NGS) has revolutionized the study of genomics. NGS has simultaneously improved speed and throughput while lowering cost, which has allowed explosive growth in the study of non-model species that the scientific community has used to create vast catalogs of organisms found throughout the planet. In this report, we outline the genomics pipeline developed in-house at the U.S. Army Combat Capabilities Development Command Army Research Laboratory SynBio Tools Branch for use on all four of the DoD High Performance Computing (HPC) Supercomputing Resource Centers (DSRCs).

The computational workup of the DNA-sequencing data generated from NGS is a vital step in the process, reworking data from millions of sequence fragments to a more pure, cohesive, annotated form useful for further analysis, comparison, and databasing. The choice of software used in this process is dictated by multiple factors including available instrumentation (equipment manufacturer and downstream implications from the technology used such as short or long read), how many organisms are evaluated in the sample (where we focus on the in-house pipeline developed for a single organism), the specifics of the organism of interest (e.g., prokaryotic or eukaryotic), available computer resources (where we make full use of DoD HPC resources), and licensing costs.

Although commercial genomics software has been developed to encapsulate community needs, it is by no means prevalent. Most software options are open source and developed in academic or nonprofit institutions. Although some users lament this "ad hoc" development scenario (Siepel 2019), the result is an active development ecosystem with optimal code diversity and a low entry barrier that allows introduction at the classroom level (Jung et al. 2020), which ensures continued development and usage through the next generation. Usability and accessibility have been further cemented by the implementation of genomics codes into new implementations of scientific workflow management systems such as Nextflow and Snakemake (Wratten et al. 2021; Djaffardjy et al. 2023).

2. Methods

Raw data obtained from a sequencer requires preliminary steps for conversion into a state suitable for genomic analysis. These preliminary steps include base calling, where the electrical signal output from the sequencer is interpreted and assigned as nucleotide bases (A, C, G, and T), resulting in recognizable DNA sequences. In addition, adapter sequences that are artifacts of the sequencing library preparation must be trimmed from the data. For Oxford Nanopore Technology (ONT) data,

these preliminary steps are frequently performed with appropriate software such as Guppy or Dorado. After this is done, the data is ready for workup, and the remaining flow of a genomics analysis pipeline generally follows the path from 1) filtering and quality control (QC), 2) rough taxonomy classification, 3) assembly—where fragments are put together or "assembled" to produce contiguous portions of a sequence—and polishing, 4) assembly evaluation, 5) additional taxonomy classification, and 6) genome annotation.

The in-house genomics pipeline has gone through multiple iterations through the last few years depending upon the phase of the project. The original pipeline circa 2020 was more suited to Illumina data and involved filtering with pre-existing tools in Bactopia (Petit and Read 2020) or metaWRAP (Uritskiy et al. 2018), genome assembly with SPAdes (Prjibelski et al. 2020), evaluation with QUAST (Gurevich et al. 2013; Mikheenko et al. 2018) and CheckM (Parks et al. 2014), taxonomy classification with SourMash (Irber et al. 2024), and functional annotation with Prokka (Seemann 2014). These solid software choices are ideally suited for short-read (e.g., Illumina), prokaryotic-centric data; therefore, they required augmentation with alternative codes later in the project to reflect changes in instrumentation and sample focus. An alternative pipeline suited for ONT long-read data with fragments as large as approximately 4 Mb was developed later, which used MinIONQC (Lanfear et al. 2018) for QC, Filtlong for filtering (Wick 2025), Flye for genome assembly (Kolmogorov et al. 2019), medaka for polishing (Oxford Nanopore Technologies 2025), BUSCO for evaluation (Simão et al. 2015; Seppey et al. 2019), and Prokka for annotation. These are also very good code choices, but a more complete pipeline required the inclusion of eukaryotic-oriented algorithms.

Filtering and QC of sequencing reads is a very important step in the pipeline. FastQC is a widely used, extremely stable and mature Java code that provides a thorough analysis of basic statistics (number of reads, length of reads, %QC) as well as extensive analysis of read quality (Andrews 2023). The QC is typically run both before and after filtering to 1) provide useful guidance on choice of filtering parameters, 2) provide confirmation that filtered data is suitable to proceed with assembly, and 3) flag and assess potential anomalies in the run overall. Previous studies suggested that FastQC may not be the optimal tool for filtering the lower-accuracy, very long-read sequences from ONT results. De Coster et al. (2018) suggest using the NanoPack suite instead. FastQC has presented no difficulties in the current work with either Illumina or ONT data, and it is currently still in use.

As stated, early versions of the pipeline performed filtering of raw data with the Filtlong software. The alternative software Nanofilt was transitioned into the pipeline in approximately the Fall of 2022, because it is part of the aforementioned, highly rated NanoPack set of tools. Nanofilt is no longer under active development

and has been supplanted by the RUST-based code Chopper, which we are now adding into the active pipeline (De Coster and Rademakers 2023). Boostrom et al. (2022) have reported some differences in their ability to assemble data depending on whether filtlong or Nanofilt was used. However, this can be attributed to differences in the arguments passed to the two codes, and there was no further discussion or analysis of possible effects from filtering software choice. In our current work, the protocol is to filter to a minimum read length of 1000 base pairs (bp) and typically a minimum quality score of 12. These cutoffs are very easy to maintain despite differences in error rates in Illumina and ONT (Bejaoui et al. 2025). Effects of filtering on read length and total size for an exemplar in-house dataset are shown in Table 1.

Table 1. Metrics of exemplar in-house data throughout the genomic pipeline.

Metric	Original data	Filtered data	Flye assembly	Raven assembly
Mean read length	390.2	1543.0	3504.2	21454.3
Number of reads/contigs	3205020	125350	4584	6
Read/contig length N50	420.0	1494.0	3521.0	35306.0
Standard deviation read/contig length	319.2	688.0	1759.8	17230.8
Total bases	1250690217	193409330	16063251	128726
L50	. . .	. . .	3521	35306
N50	. . .	. . .	1530	2
L90	. . .	. . .	2337	13775
N90	. . .	. . .	3849	4
Length of five largest contigs	. . .	. . .	53110, 21625, 17769, 17532, 14178	53142, 29716, 14385, 10721, 10552

There are many studies in the literature benchmarking and validating genome assembly software with various sequencing equipment (ONT, PacBio, etc.) and organism focus (yeast, bacteria, general prokaryote, general eukaryote, etc.). Overall, there is no consensus of a single superior software choice. The studies that make specific software suggestions are careful to provide caveats by use case. For the current work, the choice of a fast, accurate, supported, general-use long-read assembler software was between Raven (Vaser and Šikić 2021) and Flye.

Raven and Flye represent two of the primary current methodologies for genome assembly: overlap–layout-consensus (OLC) graph-based (Raven) and De Bruijn graph-based (Flye). For technical precision, Flye uses a generalization of the De Bruijn graph known as the repeat graph (Kolmogorov et al. 2019). Both codes are widely used, currently supported, and have been applied to a wide variety of organisms. Boostrom et al. (2022) performed a thorough analysis of code choice for filtering, assembly, or polishing in long-read sequencing analysis of

antimicrobial resistance genes in bacteria and ultimately recommended a combination of both Raven and Flye. Wick and Holt (2021) performed benchmarking of eight long-read assembler codes on a variety of real and simulated prokaryotic whole-genome datasets. They found that no single software performed well on all metrics and suggested a set of four codes (Flye, Miniasm/Minipolish, NextDenovo/NextPolish, and Raven) as high performers overall. Chen et al. (2020) benchmarked six long-read assembly codes focusing on genomic assembly of bacterial pathogens. This work deemed Raven to be the most accurate and robust, in addition to performing best in assembly of low-quality sets. Cosma et al. (2023) tested five long-read assembler codes on real and simulated eukaryotic genomes and reached the conclusion that Flye performed best overall but was not best in all categories nor in all use cases. Chen et al. (2020) and Cosma et al. (2023) agreed with the overall conclusions of Wick and Holt (2021).

Since the in-house genomics work encompasses a wide variety of largely unknown organisms, the pipeline typically uses both Flye and Raven to provide a cross-check of assembly results. Statistics from assembly using both Flye and Raven on an exemplary dataset are given in Table 1. The Flye assembly contains many smaller contigs than the Raven assembly. This result is typical, particularly of low-quality data that are used for a quick identification of unknowns.

Both Flye and Raven run sufficiently quickly on most of our in-house data such that it can be done interactively on HPC resources. For the approximately 840-Mb (193 Mbp) filtered sample data shown in Table 1, Raven ran interactively in approximately 11 s on 40 processors of the Navydsrc cluster Koehr, whereas Flye ran in approximately 97 s on 40 processors of Koehr. Because these are short interactive runs, we provide these results as an estimate rather than a strict benchmark. There are large differences in the size and overall statistics of the assembly results provided by Flye and Raven, as shown in Table 1. However, when we study the largest contigs that constitute the most significant portion of the assembly, the results occasionally coalesce. In this instance, the five largest contigs from each assembly are nearly equivalent in size.

As previously stated, Raven and Flye are predicated on different approaches to assembly. In an OLC approach such as Raven, the algorithm checks for overlap among all reads, generates a graph laying out all the reads and overlap information (specifically an overlap graph where nodes represent reads and edges represent the overlap between connected reads), and infers a consensus. In De Bruin graph-based algorithms, reads are chopped into shorter k-mers that represent the nodes, whereas edges connect k-mers that share a common prefix and suffix of length k-1. Detailed overviews and comparison of these algorithms and their results are available in Li et al. (2012) and Rizzi et al. (2019). For the sample dataset discussed here, the

assembly graph produced by Raven is shown in Figure 1, and the repeat graph produced by Flye is shown in Figure 2.

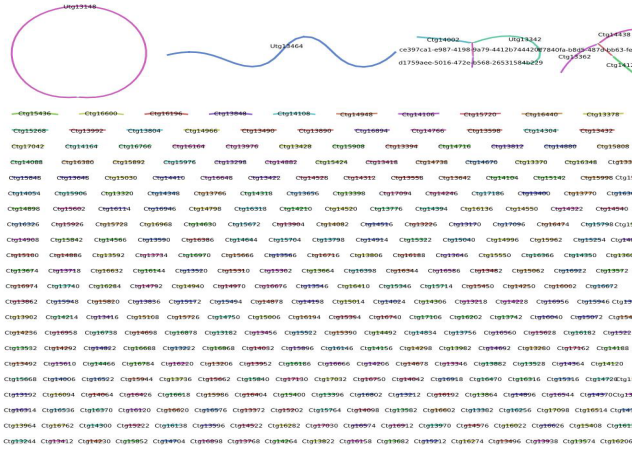

Figure 1. Assembly graph produced by Raven on exemplary dataset.

Figure 2. Repeat graph produced by Flye on exemplary dataset.

We compare the largest contig produced by each method. A close-up of the assembly graph of the largest contig by Raven is shown in Figure 3. A close-up of the repeat graph of the largest contig produced by Flye is shown in Figure 4.

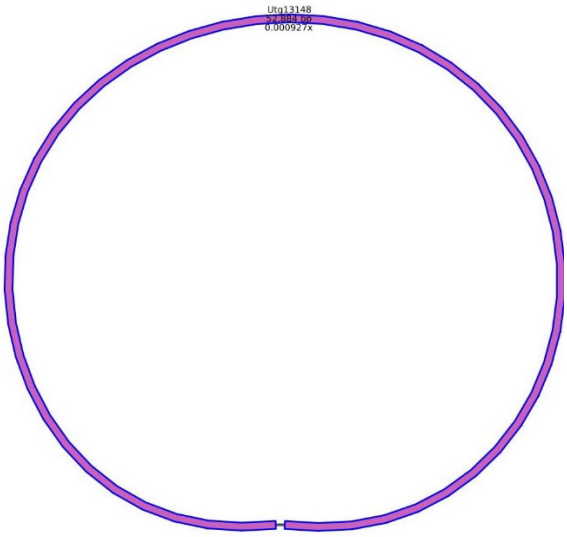

Figure 3. Close-up of assembly graph of largest contig produced by Raven on exemplary dataset.

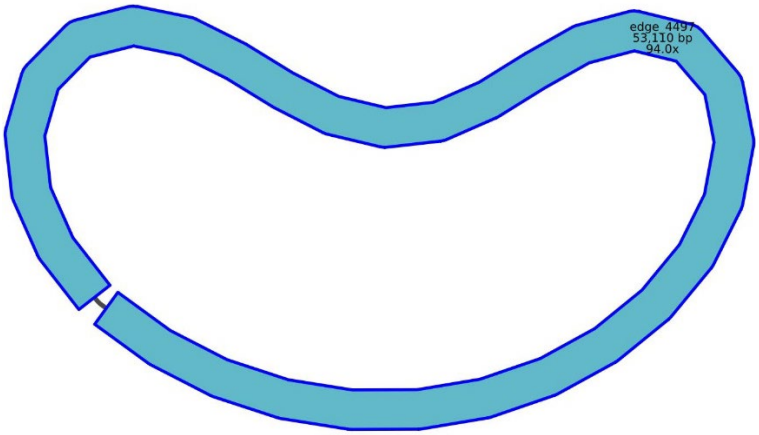

Figure 4. Close-up of repeat graph of largest contig produced by Flye on exemplary dataset.

Despite the differences in the representation, the differences in the algorithm, and cosmetic differences in layout (because the end focus of the graph is connectivity, there is no meaningful separation between the circle of Figure 3 and the bean shape of Figure 4), we see that for this particular contig, everything reduces to a simple, similar result with very little room for difference in interpretation. The two contigs are approximately the same length, with Flye producing a contig of 53110 bp and Raven producing a contig of 53142 bp. We have aligned the two contigs (largest contig from Flye assembly and largest contig from Raven assembly) using three different alignment software tools to demonstrate the multiple tools used in standard practice. Figure 5 is a visualization of the XMFA-formatted alignment file produced by the progressiveMauve algorithm of Darling et al. (2010) on the two contigs. Figure 5 shows direct alignment of two large subsections of the contigs:

one subsection is green and the other is pink. The two contigs are clearly largely identical, but the two large subsections are positioned differently within the contigs overall. The excellent Mauve tool is no longer supported independently because it has been subsumed into Geneious (2025) as a plug-in.

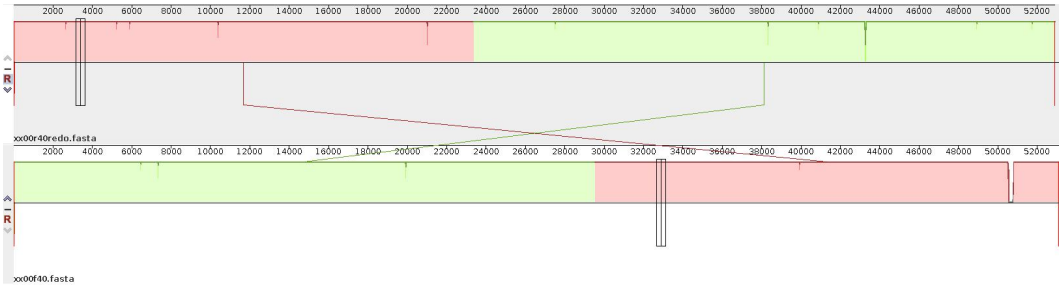

Figure 5. Visualization of progressiveMauve alignment of largest contig from Flye assembly vs. Raven assembly.

Similar information is displayed in a different fashion in Figure 6, which visualizes the dot plot representation of the MAF-formatted alignment file from the SibeliaZ software of Minkin and Medvedev (2020). Production of the dot plot image requires further work-up with the wgatools software of Wei et al. (2025). Although visually dissimilar, Figures 5 and 6 represent the same information from the alignment; that is, two large subregions of high identity within the two contigs, rearranged relative to each other. Finally, for enthusiasts of the classic National Center for Bioinformatic Information (NCBI) BLAST package of Camacho et al. (2009), the two contigs were aligned with blastn using defaults. This showed very specifically that bp 1–23468 of the Raven assembly aligned with bp 29634–53110 of the Flye assembly with identity of 23465/23480 (99%) and 15/23480 gaps, and bp 1–19955 of the Flye assembly aligned with bp 23469–43418 of the Raven with identity of 19944/19961 (99%) and 17/19961 gaps. This is basic quantification of everything visualized in Figures 5 and 6.

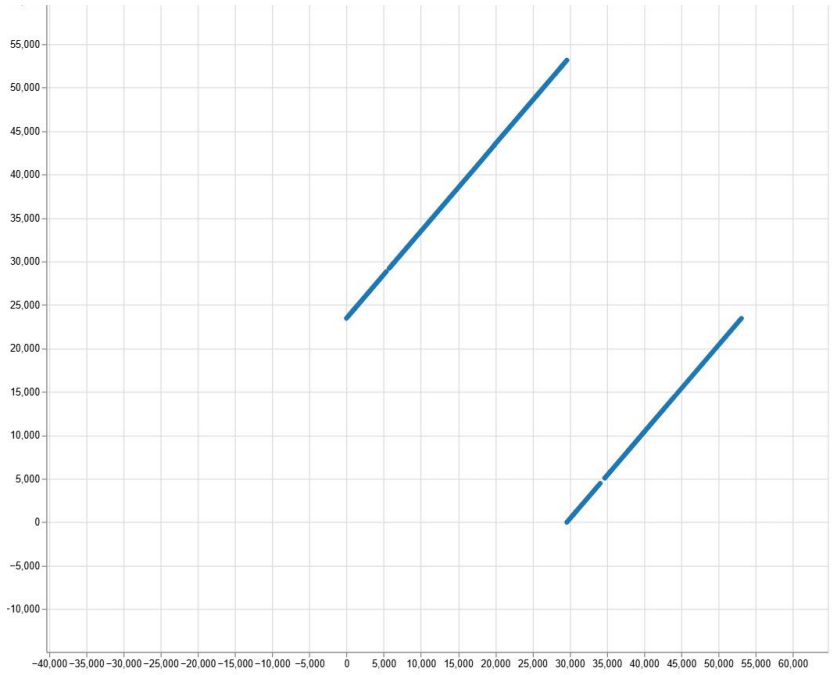

Figure 6. SibeliaZ alignment of largest contig from Flye assembly and Raven assembly, visualized in the dot plot produced by wgatools.

3. Discussion

We have demonstrated that Raven and Flye can produce very similar contigs. However, we used BLAST all-against-all to investigate the remaining largest five contigs in the exemplary dataset and found no meaningful overlap. The organism in question was tentatively identified as eukaryotic, meaning that the whole genome will be comparatively large and contain large repeat groups. Li et al. (2012) have suggested that the longer fragments used by the OLC-based algorithms can facilitate handling of repeat groups, but that this is a multi-faceted problem requiring across-the-board improvements in data collection as well as algorithms. For the time being, we will continue to use both Flye and Raven in our genomics pipeline because results to date demonstrate strengths and weaknesses in both.

After assemblies are generated, general contig statistics, including length and quality distribution, are generated by available tools (NanoComp, nanoQC, cramino, etc.) within the Nanopack2 suite (De Coster and Rademakers 2023). Additional genome quality metrics may be assessed by QUAST (Gurevich et al. 2013) as well as estimation of genome completeness and redundancy with BUSCO (Simão et al. 2015; Seppey et al. 2019; Manni et al. 2021). The Genome relative Abundance and Average Size (GAAS) software package (Angly et al. 2009) is occasionally used, and future work will assess the relatively new compleasm software as an alternative to BUSCO (Huang and Li 2023).

Taxonomy classification from sequencing is a black art and far from a mature science, particularly for novel organisms. The ubiquitous BLAST+ of the National Institutes of Health's NCBI is undoubtedly one of the most highly used servers in the community (Camacho et al. 2009). Fortunately, this is available for installation on local resources to avoid use of a public webserver, and the databases, although cumbersome to download and install, are updated relatively frequently. The Kraken2 classification system (Wood et al. 2019; Kraken2 2025) is also widely used due to its speed and relative accuracy on known organisms. However, we stress that existing classification algorithms rely heavily upon comparison to reference genomes and are generally not optimum for identification of novel organisms. Due to the extreme variability in results depending on details of the use case, a review of the current state of the (black) art of taxonomy classification with a list of additional codes and databases is beyond the state of this report and is left for a planned future technical report.

The final step in the genomic pipeline is annotation, that is, the process of mapping (identifying and labeling) functional portions of the genome. For prokaryotic systems, we use the software Prokka (Seemann 2014) because it is suitable and efficient for use on local resources. Like so many other portions of the pipeline, annotation of eukaryotic species is more difficult than with prokaryotes, and codes are less mature. We have implemented the funannotate software (Palmer and Stajich 2020) with reasonable success. We are continually monitoring the literature for alternative and additional codes for this portion of the pipeline.

4. Conclusion

Software pipelines are an ever-changing topography in any scientific field and genomics is no exception. The data in this report provides a snapshot of current practices, a repository of useful references, and a starting place for discussion.

5. References

Andrews S. FastQC: a quality control tool for high throughput sequence data. 2023. [accessed 2025 Aug 14]. http://www.bioinformatics.bbsrc.ac.uk/projects/fastqc

Angly FE et al. The GAAS metagenomic tool and its estimations of viral and microbial average genome size in four major biomes. PLoS Comput Biol. 2009;5(12):e1000593. https://doi.org/10.1371/journal.pcbi.1000593

Bejaoui S et al. Comparison of Illumina and Oxford Nanopore sequencing data quality for *Clostridioides difficile* genome analysis and their application for epidemiological surveillance. BMC Genomics. 2025;26(1):92. https://doi.org/10.1186/s12864-025-11267-9

Boostrom I, Portal EAR, Spiller OB, Walsh TR, Sands K. Comparing long-read assemblers to explore the potential of a sustainable low-cost, low-infrastructure approach to sequence antimicrobial resistant bacteria with Oxford Nanopore sequencing. Front Microbiol. 2022;13:796465. https://doi.org/10.3389/fmicb.2022.796465

Camacho C et al. BLAST+: architecture and applications. BMC Bioinform. 2009;10:421.

Chen Z, Erickson DL, Meng J. Benchmarking long-read assemblers for genomic analyses of bacterial pathogens using Oxford Nanopore sequencing. Int J Mol Sci. 2020;21(23):9161. https://doi.org/10.3390/ijms21239161

Cosma BM et al. Evaluating long-read de novo assembly tools for eukaryotic genomes: insights and considerations. GigaScience. 2023;12:giad100. https://doi.org/10.1093/gigascience/giad100

Darling AE, Mau B, Perna NT. progressiveMauve: multiple genome alignment with gene gain, loss, and rearrangement. PLoS One. 2010;5(6):e11147. https://doi.org/10.1371/journal.pone.0011147

De Coster W, D'Hert S, Schultz DT, Cruts M, Van Broeckhoven C. NanoPack: visualizing and processing long-read sequencing data. Bioinformatics. 2018;34(15):2666–2669. https://doi.org/10.1093/bioinformatics/bty149

De Coster W, Rademakers R. NanoPack2: Population-scale evaluation of long-read sequencing data. Bioinformatics. 2023;39(5):btad311. https://doi.org/10.1093/bioinformatics/btad311

Djaffardjy M et al. Developing and reusing bioinformatics data analysis pipelines using scientific workflow systems. Comput Struct Biotechnol J. 2023;21:2075–2085. https://doi.org/10.1016/j.csbj. 2023.03.003

Geneious. Prime. [accessed 2025 Aug 13]. https://www.geneious.com

Gurevich A, Saveliev V, Vyahhi N, Tesler G. QUAST: quality assessment tool for genome assemblies. Bioinformatics. 2013;29(8):1072–1075. https://doi.org/10.1093/bioinformatics/btt086

Huang N, Li H. Compleasm: a faster and more accurate reimplementation of BUSCO. Bioinformatics. 2023;39(10):btad595. https://doi.org/10.1093/bioinformatics/btad595

Irber L et al. CT Sourmash v4: a multitool to quickly search, compare, and analyze genomic and metagenomic data sets. J Open Source Softw. 2024;9(98):6830. https://doi.org/10.21105/joss.06830

Jung H et al. Twelve quick steps for genome assembly and annotation in the classroom. PLoS Comput Biol. 2020;16(11):e1008325. https://doi.org/10.1371/journal.pcbi.1008325

Kolmogorov M et al. Assembly of long, error-prone reads using repeat graphs. Nat Biotechnol. 2019;37:540–546. https://doi.org/10.1038/s41587-019-0072-8

Kraken2 indexes. [accessed 2025 Sep 9]. https://benlangmead.github.io/aws-indexes/k2

Lanfear R, Schalamun M, Kainer D, Wang W, Schwessinger B. MinIONQC: fast and simple quality control for MinION sequencing data. Bioinformatics. 2018;bty654. https://doi.org/10.1093/bioinformatics/bty654

Li Z et al. Comparison of the two major classes of assembly algorithms: overlap-layout-consensus and de-bruijn-graph. Brief Funct Genomics. 2012;11(1):25–37. https://doi.org/10.1093/bfgp/elr035

Manni M, Berkeley MR, Seppey M, Simao FA, Zdobnov EM. BUSCO update: novel and streamlined workflows along with broader and deeper phylogenetic coverage for scoring of eukaryotic, prokaryotic, and viral genomes. arXiv; 2021 June 22. arXiv:2106.11799. https://doi.org/10.48550/arXiv.2106.11799

Mikheenko A, Prjibelski A, Saveliev V, Antipov D, Gurevich A. Versatile genome assembly evaluation with QUAST-LG. Bioinformatics. 2018;34(13):i142–i150. https://doi.org/10.1093/bioinformatics/bty266

Minkin I, Medvedev P. Scalable multiple whole-genome alignment and locally collinear block construction with SibeliaZ. Nat Commun. 2020;11:6327. https://doi.org/10.1038/s41467-020-19777-8

Oxford Nanopore Technologies. [accessed 2025 Sep 9]. https://github.com/nanoporetech/medaka

Palmer JM, Stajich J. Funannotate v1.8.1: eukaryotic genome annotation (v1.8). Zenodo. 2020. [accessed 2025 Aug 14]. https://doi.org/10.5281/zenodo.1134477

Parks DH, Imelfort M, Skennerton CT, Hugenholtz P, Tyson GW. Assessing the quality of microbial genomes recovered from isolates, single cells, and metagenomes. Genome Res. 2014;25:1043–1055.

Petit III RA, Read TD. Bactopia: a flexible pipeline for complete analysis of bacterial genomes. mSystems. 2020;5. https://doi.org/10.1128/mSystems.00190-20

Prjibelski A, Antipov D, Meleshko D, Lapidus A, Korobeynikov A. Using SPAdes de novo assembler. Curr Protoc Bioinform. 2020;70:e102. https://doi.org/10.1002/cpbi.102

Rizzi R et al. Overlap graphs and de Bruijn graphs: data structures for de novo genome assembly in the big data era. Quant Biol. 2019;7:278–292. https://doi.org/10.1007/s40484-019-0181-x

Seemann T. Prokka: rapid prokaryotic genome annotation. Bioinformatics. 2014;30(14):2068–2069. https://doi.org/10.1093/bioinformatics/btu153

Seppey M, Manni M, Zdobnov EM. BUSCO: assessing genome assembly and annotation completeness. Methods Mol Biol. 2019;1962:227–245. https://doi.org/10.1007/978-1-4939-9173-0_14

Siepel A. Challenges in funding and developing genomic software: roots and remedies. Genome Biol. 2019;20:147. https://doi.org/10.1186/s13059-019-1763-7

Simão FA, Waterhouse RM, Ioannidis P, Kriventseva EV, Zdobnov EM. BUSCO: assessing genome assembly and annotation completeness with single-copy orthologs. Bioinformatics. 2015;31(19):3210–3212. https://doi.org/10.1093/bioinformatics/btv351

Uritskiy GV, DiRuggiero J, Taylor J. MetaWRAP-a flexible pipeline for genome-resolved metagenomic data analysis. Microbiome. 2018;6(1):158. https://doi.org/10.1186/s40168-018-0541-1

Vaser R, Šikić M. Time- and memory-efficient genome assembly with Raven. Nat Comput Sci. 2021;1:332–336. https://doi.org/10.1038/s43588-021-00073-4

Wei W et al. wgatools: an ultrafast toolkit for manipulating whole-genome alignments. Bioinformatics. 2025;41(4):btaf132. https://doi.org/10.1093/bioinformatics/btaf132

Wick RR. Filtlong [accessed 2025 Sep 9]. https://github.com/rrwick/Filtlong

Wick RR, Holt KE. Benchmarking of long-read assemblers for prokaryote whole genome sequencing F1000Research. 2021;8:2138. https://doi.org/10.12688/f1000research.21782.4

Wood DE, Lu J, Langmead B. Improved metagenomic analysis with Kraken 2. Genome Biol. 2019;20:257. https://doi.org/10.1186/s13059-019-1891-0

Wratten L, Wilm A, Göke J. Reproducible, scalable, and shareable analysis pipelines with bioinformatics workflow managers. Nat Methods. 2021;18:1161–1168. https://doi.org/10.1038/s41592-021-01254-9

List of Symbols, Abbreviations, and Acronyms

bp	Base Pairs
DNA	Deoxyribonucleic Acid
DoD	Department of Defense
DSRC	DoD Supercomputing Resource Center
GAAS	Genome Relative Abundance and Average Size
HPC	High Performance Computing
NCBI	National Center for Bioinformatic Information
NGS	Next-Generation Sequencing
OLC	Overlap–Layout-Consensus
ONT	Oxford Nanopore Technology
QC	Quality Control

1 (PDF)	DEFENSE TECHNICAL INFORMATION CTR DTIC OCA
1 (PDF)	DEVCOM ARL FCDD RLB CB TECH LIB