Title: "polishCLR: a Nextflow workflow for polishing PacBio CLR genome assemblies"

Authors: Jennifer Chang[1,2,3]*, Amanda R. Stahlke[4]*, Sivanandan Chudalayandi[3], Benjamin D. Rosen[5], Anna K. Childers[4], Andrew Severin[3]**

1. USDA, Agricultural Research Service, Jamie Whitten Delta States Research Center, Genomics and Bioinformatics Research Unit, 141 Experiment Station Road, Stoneville, MS 38776, USA

2. Oak Ridge Institute for Science and Education, P.O. Box 117, Oak Ridge, TN 37831, USA

3. Genome Informatics Facility, Office of Biotechnology, Iowa State University, Ames, Iowa 50010, USA

JC: jennifer.chang.bioinform@gmail.com, ORCID: 0000-0002-8381-3765

SC: csiva@iastate.edu

4. USDA, Agricultural Research Service, Beltsville Agricultural Research Center, Bee Research Laboratory, 10300 Baltimore Avenue, Beltsville MD 20705, USA

ARS: amandastahlke@usda.gov, ORCID: 0000-0001-5724-598X

AKC: anna.childers@usda.gov ORCID: 0000-0002-0747-8539

5. USDA, Agricultural Research Service, Beltsville Agricultural Research Center, Animal Genomics and Improvement Laboratory, 10300 Baltimore Avenue, Beltsville, MD 20705, USA

BDR: ben.rosen@usda.gov, ORCID: 0000-0001-9395-8346

*co-first authors contributed equally

**Author for Correspondence: Andrew Severin, Genome Informatics Facility, Iowa State University, Ames, Iowa, USA, 50010, (515) 294-1320, severin@iastate.edu

Abstract

Long-read sequencing has revolutionized genome assembly, yielding highly contiguous, chromosome-level contigs. However, assemblies from some third generation long read technologies, such as Pacific Biosciences (PacBio) Continuous Long Reads (CLR), have a high error rate. Such errors can be corrected with short reads through a process called polishing. Although best practices for polishing non-model *de novo* genome assemblies were recently described by the Vertebrate Genome Project (VGP) Assembly community, there is a need for a publicly available, reproducible workflow that can be easily implemented and run on a conventional high performance computing environment. Here, we describe polishCLR (https://github.com/isugifNF/polishCLR), a reproducible Nextflow workflow that implements best practices for polishing assemblies made from CLR data. PolishCLR can be initiated from several input options that extend best practices to suboptimal cases. It also provides re-entry points throughout several key processes including identifying duplicate haplotypes in purge_dups, allowing a break for scaffolding if data are available, and throughout multiple rounds of polishing and evaluation with Arrow and FreeBayes. PolishCLR is containerized and publicly available for the greater assembly community as a tool to complete assemblies from existing, error-prone long-read data.

Main

Long reads, including those generated by third-generation sequencing platforms such as Pacific Biosciences (PacBio) and Oxford Nanopore Technology (ONT), have revolutionized genome assembly (Childers et al., 2021; Hotaling et al., 2021; Rhie et al., 2021). However, until recent advances (Hon et al., 2020; Wang et al., 2021), long-read sequencing technologies have had high error rates (5-15%), especially among indels (Watson & Warr, 2019). Thus, the vast majority of long-read data that is currently publicly available yields assemblies with low overall consensus accuracy, which, if left uncorrected, negatively impacts downstream analyses, such as gene annotation (Watson & Warr, 2019). These assembly errors require correction with an additional higher fidelity read set, such as short-read Illumina data, in a process called polishing (Chin et al., 2016; Helper et al., 2016; Walker et al., 2014).

Polishing can be a complex process, with high computational cost, non-trivial file-handling, and issues around special cases that must be resolved. For example, the long-read contig assembly should ideally be polished with high-fidelity reads from the same individual, but this may not be technically feasible when sufficient DNA cannot be extracted from individual specimens, for example in small-bodied organisms such as many insects. In such cases, it is necessary to modify parameters in the standard workflow. Best practices for *de novo*, chromosome-scale vertebrate genome assembly from error prone PacBio continuous long reads (CLR) reads were recently described (Rhie et al., 2021), however it can be challenging to run this code and reproduce widely. In order to produce the best possible genome assemblies using existing data from species regardless of their position in the tree of life, the genome assembly community needs a publicly available, flexible and reproducible workflow that is containerized so it can be run on any conventional HPC.

Bioinformatic pipelines with complex entrance and decision points, such as polishing, are inherently difficult to track, develop, and debug. Increasing interest in workflow development systems that track data and software provenance, enable scalability and reproducibility, and re-

entrant code (Wratten et al., 2021) have led to the development of several workflow languages, largely inspired by GNU Make (Amstutz et al., 2016; Köster & Rahmann, 2012; Stallman & McGrath, 1991). Nextflow is a Domain Specific Language (Di Tommaso et al., 2017) that currently leads workflow systems in terms of ease of scripting and submitting to cloud computing resources (Fjukstad & Bongo, 2017; Jackson et al., 2021; Leipzig, 2017; Spjuth et al., 2020). A key benefit of Nextflow compared to earlier workflow languages is being able to submit jobs to a local machine, an HPC, or cloud-based compute environments. These features empower a large range of bioinformatic pipelines, for example, initial read processing and annotation lift-over (Federico et al., 2019; Talenti & Prendergast, 2021). In this paper, we describe polishCLR, a reproducible Nextflow workflow which implements the current best practices for polishing CLR assemblies and is flexible to multiple input assembly and sample considerations.

The polishCLR workflow can be easily initiated from three input cases (Fig. 1). In the first case (Case 1), users may start with an unresolved primary assembly with (e.g., the output of FALCON 2-asm (Chin et al., 2016)) or without (e.g., the output of Canu or wtdbg2 (Koren et al., 2017; Ruan & Li, 2020)) associated contigs. Additionally, it can handle a haplotype-resolved but unpolished set (Case 2) (e.g., the output of FALCON-Unzip 3-unzip (Chin et al., 2016)). In the ideal case (Case 3), the pipeline is initiated with a haplotype-resolved, CLR long-read polished set of primary and alternate contigs (e.g., the output of FALCON-Unzip 4-polish). In all cases, inclusion of organellar genomes, e.g., the mitochondrial genome, will improve the polishing of nuclear mitochondrial or plasmid pseudogenes (Howe et al., 2021). Organellar genomes should be generated and polished separately for best results, using pipelines such as the mitochondrial companion to polishCLR, polishCLRmt (Stahlke et al, in prep) or mitoVGP (Formenti et al., 2021). To allow for the inclusion of scaffolding before final polishing (e.g., Durand et al., 2016) and increase the potential for gap-filling across correctly oriented scaffolded contigs, the core workflow is divided into two steps, controlled by a --step parameter flag.

In --step 1, if initiating the workflow under Case 1 or 2, unpolished primary contigs are merged with the organellar genome and associated contigs or alternate haplotypes if available, then polished with a single round of Arrow long-read polishing (Pacific BioScience) before entering the core workflow (controlled by --arrow01 true). During Arrow steps (here and later in Step 2), polishCLR improves re-entry and computational resource use by delineating at least seven Nextflow processes: 1) indexing each contig, 2) creating a pbmm2 index of the assembly, 3) aligning PacBio reads to the assembly, 4) submitting a GCpp Arrow job for each contig in parallel, 5) combining the separate contig variant calling format (VCF) files, 6) reformatting Arrow generated VCF for Merfin filtering (Formenti et al., 2021), and 7) converting the resultant VCF back to FASTA format. Then, in all three cases, the core workflow employs purge_dups v. 1.2.5 (Guan et al., 2020) to remove duplicated sequence at the ends of separated primary contigs, with cutoffs automatically estimated from a generated histogram of k-mers. The histogram is captured as one of the relevant outputs for users to review. Purged primary sequences are then concatenated to the alternate haplotype contigs and the combined alternate set is purged of duplicates. BUSCO completeness metrics (Manni et al., 2021; Seppey et al., 2019; Simao et al., 2015; Waterhouse et al., 2018) are generated for the primary contigs before and after removing duplicated content to ensure that cutoff parameters are effective and do not remove too much genic content. The eukaryotic BUSCO database is used by default, but users may provide a designated lineage (controlled by a --busco-lineage flag). If additional data are available, this de-duplicated primary contig assembly can then be scaffolded by the user before initiating the second phase of the workflow.

In --step 2, the primary, alternate, and organellar assemblies are merged and polished with an additional round of Arrow, followed by two rounds of FreeBayes (Garrison & Marth, 2012). Indeed, the iterative nature of polishing benefits from the re-entrant caching and templates of the workflow. By default, this second round of Arrow-identified variants are only filtered via Merfin if the CLR and the Illumina reads came from the same specimen, adding

additional Nextflow processes to the Arrow delineation described above to create a meryl genome database and perform filtering (McCartney et. al, unpublished data). However, if short-read data are from a different specimen than the long-read-based contig assembly, then Merfin filtering can be turned off to avoid over-polishing with the parameter flag --same-specimen false. As with Arrow, polishCLR takes advantage of Nextflow in seven processes to implement FreeBayes: 1) creating contig windows, 2) generating a meryl database from the genome, 3) aligning Illumina short reads, 4) polishing via FreeBayes, 5) combining windowed VCF files, 6) filtering VCFs by Merfin, 7) and converting VCFs to FASTA. Throughout the polishCLR workflow, reports are automatically generated to assess genome assembly quality, including k-mer based completeness and consensus accuracy QV scores via Merqury (Rhie et al., 2020), as well as genome size distribution statistics generated with BBMap (e.g., N50) (Bushnell, 2014). These reports allow users to understand how the assembly changed through each major phase of the workflow. The complete, detailed pipeline can be viewed in Supplemental Figure 1.

The polishCLR workflow is publicly available (https://github.com/isugifNF/polishCLR), reproducible, interoperable, easily portable, and can be run on a conventional HPC or extended to cloud computing resources simply by swapping out the Nextflow config file. Software dependencies are listed in a conda environment file. Its use has been demonstrated on several arthropod species assemblies as part of the Ag100Pest Initiative (Childers et al., 2021). Runtimes and summaries from each of the three starting input cases are included (Supplementary Table S1; Stahlke and Coates, 2022) with a full genome announcement forthcoming (Stahlke et al., unpublished data). The polishCLR workflow will increase the efficiency of polishing many genomes and reduce the potential of human error in this multistep process. Despite the much-reduced error rate of PacBio HiFi and ONT reads, polishing approaches continue to be an important component of accurate genome assembly (Shafin et al., 2021). Although this pipeline was not designed to polish with ONT reads, the workflow is available on GitHub and welcomes any future contributions.

147

148 **Acknowledgements**

149 This work was supported by the U.S. Department of Agriculture, Agricultural Research Service
150 (USDA-ARS) and used resources provided by the SCINet project of the USDA-ARS, ARS
151 project number 0500-00093-001-00-D. JC was supported, in part, by an appointment to the
152 Research Participation Program at the Agricultural Research Service, United States Department
153 of Agriculture, administered by the Oak Ridge Institute for Science and Education through an
154 interagency agreement between the U.S. Department of Energy and USDA-ARS under contract
155 number DE-AC05-06OR23100. This workflow was developed as part of the USDA-ARS
156 Ag100Pest Initiative. The authors thank members of the USDA-ARS Ag100Pest Team and
157 SCINet Virtual Resource Support Core (VRSC) for fruitful discussions and troubleshooting
158 throughout the development of this workflow. All opinions expressed in this paper are the
159 author's and do not necessarily reflect the policies and views of USDA, DOE, or ORAU/ORISE.
160 Mention of trade names or commercial products in this publication is solely for the purpose of
161 providing specific information and does not imply recommendation or endorsement by the U.S.
162 Department of Agriculture. USDA is an equal opportunity provider and employer.

163

References

Amstutz, P., Crusoe, M. R., Tijanić, N., Chapman, B., Chilton, J., Heuer, M., Kartashov, A., Leehr, D., Ménager, H., & Nedeljkovich, M. (2016). Common workflow language, v1. 0.

Bushnell, B. (2014). BBTools software package. *URL http://sourceforge. net/projects/bbmap*, *578*, 579.

Childers, A. K., Geib, S. M., Sim, S. B., Poelchau, M. F., Coates, B. S., Simmonds, T. J., Scully, E. D., Smith, T. P. L., Childers, C. P., Corpuz, R. L., Hackett, K., & Scheffler, B. (2021). The USDA-ARS Ag100Pest Initiative: High-Quality Genome Assemblies for Agricultural Pest Arthropod Research. *Insects*, *12*(7). https://doi.org/10.3390/insects12070626

Chin, C. S., Peluso, P., Sedlazeck, F. J., Nattestad, M., Concepcion, G. T., Clum, A., Dunn, C., O'Malley, R., Figueroa-Balderas, R., Morales-Cruz, A., Cramer, G. R., Delledonne, M., Luo, C., Ecker, J. R., Cantu, D., Rank, D. R., & Schatz, M. C. (2016). Phased diploid genome assembly with single-molecule real-time sequencing. *Nat Methods*, *13*(12), 1050-1054. https://doi.org/10.1038/nmeth.4035

Di Tommaso, P., Chatzou, M., Floden, E. W., Barja, P. P., Palumbo, E., & Notredame, C. (2017). Nextflow enables reproducible computational workflows. *Nat Biotechnol*, *35*(4), 316-319. https://doi.org/10.1038/nbt.3820

Durand, N. C., Robinson, J. T., Shamim, M. S., Machol, I., Mesirov, J. P., Lander, E. S., & Aiden, E. L. (2016). Juicebox Provides a Visualization System for Hi-C Contact Maps with Unlimited Zoom. *Cell Syst*, *3*(1), 99-101. https://doi.org/10.1016/j.cels.2015.07.012

Federico, A., Karagiannis, T., Karri, K., Kishore, D., Koga, Y., Campbell, J. D., & Monti, S. (2019). Pipeliner: A Nextflow-based framework for the definition of sequencing data processing pipelines. *Frontiers in genetics*, *10*, 614.

Fjukstad, B., & Bongo, L. A. (2017). A review of scalable bioinformatics pipelines. *Data Science and Engineering*, *2*(3), 245-251.

Formenti, G., Rhie, A., Balacco, J., Haase, B., Mountcastle, J., Fedrigo, O., Brown, S., Capodiferro, M. R., Al-Ajli, F. O., Ambrosini, R., Houde, P., Koren, S., Oliver, K., Smith, M., Skelton, J., Betteridge, E., Dolucan, J., Corton, C., Bista, I., . . . Vertebrate Genomes Project, C. (2021). Complete vertebrate mitogenomes reveal widespread repeats and gene duplications. *Genome Biol*, *22*(1), 120. https://doi.org/10.1186/s13059-021-02336-9

Formenti, G., Rhie, A., Walenz, B. P., Thibaud-Nissen, F., Shafin, K., Koren, S., Myers, E. W., Jarvis, E. D., & Phillippy, A. M. (2021). Merfin: improved variant filtering and polishing via k-mer validation. *bioRxiv*, 2021.2007.2016.452324. https://doi.org/10.1101/2021.07.16.452324

Garrison, E. P., & Marth, G. T. (2012). Haplotype-based variant detection from short-read sequencing. *arXiv: Genomics*.

Guan, D., McCarthy, S. A., Wood, J., Howe, K., Wang, Y., & Durbin, R. (2020). Identifying and removing haplotypic duplication in primary genome assemblies. *Bioinformatics*, *36*(9), 2896-2898. https://doi.org/10.1093/bioinformatics/btaa025

Hepler, N.L., Brown, M., Smith, M.L., Katzenstein, D., Paxinos, E.E. and Alexander, D., 2016. An improved circular consensus algorithm with an application to detect HIV-1 Drug-Resistance associated mutations (DRAMs). In Conference on Advances in Genome Biology and Technology.

Hon, T., Mars, K., Young, G., Tsai, Y. C., Karalius, J. W., Landolin, J. M., Maurer, N., Kudrna, D., Hardigan, M. A., Steiner, C. C., Knapp, S. J., Ware, D., Shapiro, B., Peluso, P., & Rank, D. R. (2020). Highly accurate long-read HiFi sequencing data for five complex genomes. *Sci Data*, *7*(1), 399. https://doi.org/10.1038/s41597-020-00743-4

Hotaling, S., Sproul, J. S., Heckenhauer, J., Powell, A., Larracuente, A. M., Pauls, S. U., Kelley, J. L., & Frandsen, P. B. (2021). Long Reads Are Revolutionizing 20 Years of Insect Genome Sequencing. *Genome Biol Evol*, *13*(8). https://doi.org/10.1093/gbe/evab138

Howe, K., Chow, W., Collins, J., Pelan, S., Pointon, D.-L., Sims, Y., Torrance, J., Tracey, A., & Wood, J. (2021). Significantly improving the quality of genome assemblies through curation. *GigaScience*, *10*(1). https://doi.org/10.1093/gigascience/giaa153

Jackson, M., Kavoussanakis, K., & Wallace, E. W. J. (2021). Using prototyping to choose a bioinformatics workflow management system. *PLoS Comput Biol*, *17*(2), e1008622. https://doi.org/10.1371/journal.pcbi.1008622

Koren, S., Walenz, B. P., Berlin, K., Miller, J. R., Bergman, N. H., & Phillippy, A. M. (2017). Canu: scalable and accurate long-read assembly via adaptive k-mer weighting and repeat separation. *Genome Res*, *27*(5), 722-736. https://doi.org/10.1101/gr.215087.116

Köster, J., & Rahmann, S. (2012). Snakemake—a scalable bioinformatics workflow engine. *Bioinformatics*, *28*(19), 2520-2522.

Leipzig, J. (2017). A review of bioinformatic pipeline frameworks. *Brief Bioinform*, *18*(3), 530-536. https://doi.org/10.1093/bib/bbw020

Manni, M., Berkeley, M. R., Seppey, M., Simao, F. A., & Zdobnov, E. M. (2021). BUSCO Update: Novel and Streamlined Workflows along with Broader and Deeper Phylogenetic Coverage for Scoring of Eukaryotic, Prokaryotic, and Viral Genomes. *Mol Biol Evol*, *38*(10), 4647-4654. https://doi.org/10.1093/molbev/msab199

McCartney, A. M., Shafin, K., Alonge, M., Bzikadze, A. V., Formenti, G., Fungtammasan, A., ... & Rhie, A. Chasing perfection: validation and polishing strategies for telomere-to-telomere genome assemblies. *unpublished data* *https://doi.org/10.1101/2021.07.02.450803, last accessed February 2, 2022*

Rhie, A., McCarthy, S. A., Fedrigo, O., Damas, J., Formenti, G., Koren, S., Uliano-Silva, M., Chow, W., Fungtammasan, A., Kim, J., Lee, C., Ko, B. J., Chaisson, M., Gedman, G. L., Cantin, L. J., Thibaud-Nissen, F., Haggerty, L., Bista, I., Smith, M., . . . Jarvis, E. D. (2021). Towards complete and error-free genome assemblies of all vertebrate species. *Nature*, *592*(7856), 737-746. https://doi.org/10.1038/s41586-021-03451-0

Rhie, A., Walenz, B. P., Koren, S., & Phillippy, A. M. (2020). Merqury: reference-free quality, completeness, and phasing assessment for genome assemblies. *Genome Biol*, *21*(1), 245. https://doi.org/10.1186/s13059-020-02134-9.

Ruan, J., & Li, H. (2020). Fast and accurate long-read assembly with wtdbg2. *Nat Methods*, *17*(2), 155-158. https://doi.org/10.1038/s41592-019-0669-3

Seppey, M., Manni, M., & Zdobnov, E. M. (2019). BUSCO: Assessing Genome Assembly and Annotation Completeness. *Methods Mol Biol*, *1962*, 227-245. https://doi.org/10.1007/978-1-4939-9173-0_14

Shafin, K., Pesout, T., Chang, P. C., Nattestad, M., Kolesnikov, A., Goel, S., Baid, G., Kolmogorov, M., Eizenga, J. M., Miga, K. H., Carnevali, P., Jain, M., Carroll, A., & Paten, B. (2021). Haplotype-aware variant calling with PEPPER-Margin-DeepVariant enables high accuracy in nanopore long-reads. *Nat Methods*, *18*(11), 1322-1332. https://doi.org/10.1038/s41592-021-01299-w

Simao, F. A., Waterhouse, R. M., Ioannidis, P., Kriventseva, E. V., & Zdobnov, E. M. (2015). BUSCO: assessing genome assembly and annotation completeness with single-copy orthologs. *Bioinformatics*, *31*(19), 3210-3212. https://doi.org/10.1093/bioinformatics/btv351

Spjuth, O., Capuccini, M., Carone, M., Larsson, A., Schaal, W., Novella, J. A., Stein, O., Ekmefjord, M., Di Tommaso, P., & Floden, E. (2020). Approaches for containerized scientific workflows in cloud environments with applications in life science. *Preprints*.

Stahlke, A.R.; Coates, B.S.. (2022). Data from polishCLR: Example input genome assemblies. Ag Data Commons. https://doi.org/10.15482/USDA.ADC/1524676. Accessed 2022-02-09.

Stallman, R. M., & McGrath, R. (1991). GNU Make-A Program for Directing Recompilation.

Talenti, A., & Prendergast, J. (2021). nf-LO: A Scalable, Containerized Workflow for Genome-to-Genome Lift Over. *Genome Biology and Evolution*, *13*(9). https://doi.org/10.1093/gbe/evab183

Walker, B. J., Abeel, T., Shea, T., Priest, M., Abouelliel, A., Sakthikumar, S., Cuomo, C. A., Zeng, Q., Wortman, J., Young, S. K., & Earl, A. M. (2014). Pilon: an integrated tool for comprehensive microbial variant detection and genome assembly improvement. *PLoS One*, *9*(11), e112963. https://doi.org/10.1371/journal.pone.0112963

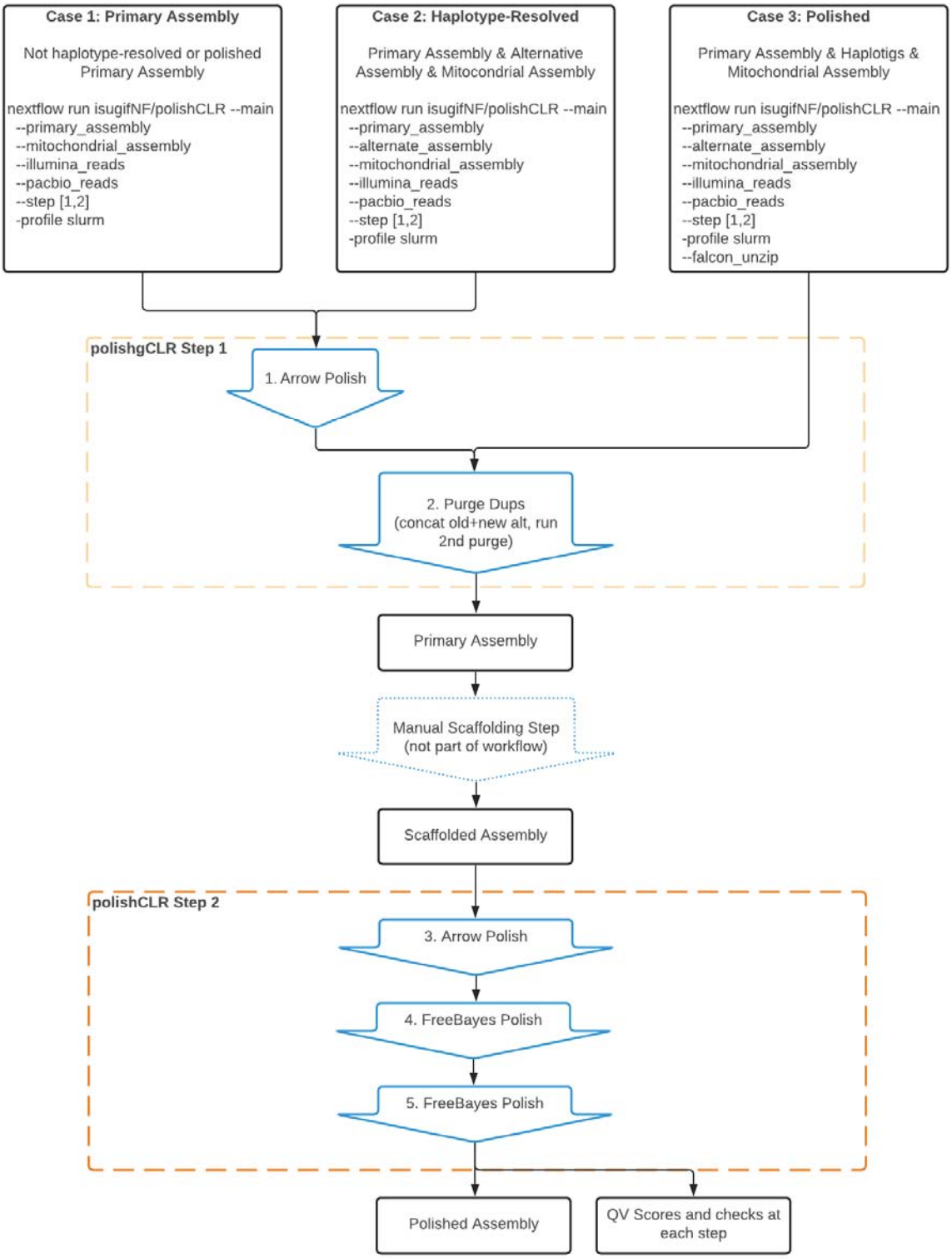
Wang, Y., Zhao, Y., Bollas, A., Wang, Y., & Au, K. F. (2021). Nanopore sequencing technology, bioinformatics and applications. *Nat Biotechnol*, *39*(11), 1348-1365. https://doi.org/10.1038/s41587-021-01108-x

Waterhouse, R. M., Seppey, M., Simao, F. A., Manni, M., Ioannidis, P., Klioutchnikov, G., Kriventseva, E. V., & Zdobnov, E. M. (2018). BUSCO Applications from Quality Assessments to Gene Prediction and Phylogenomics. *Mol Biol Evol*, *35*(3), 543-548. https://doi.org/10.1093/molbev/msx319

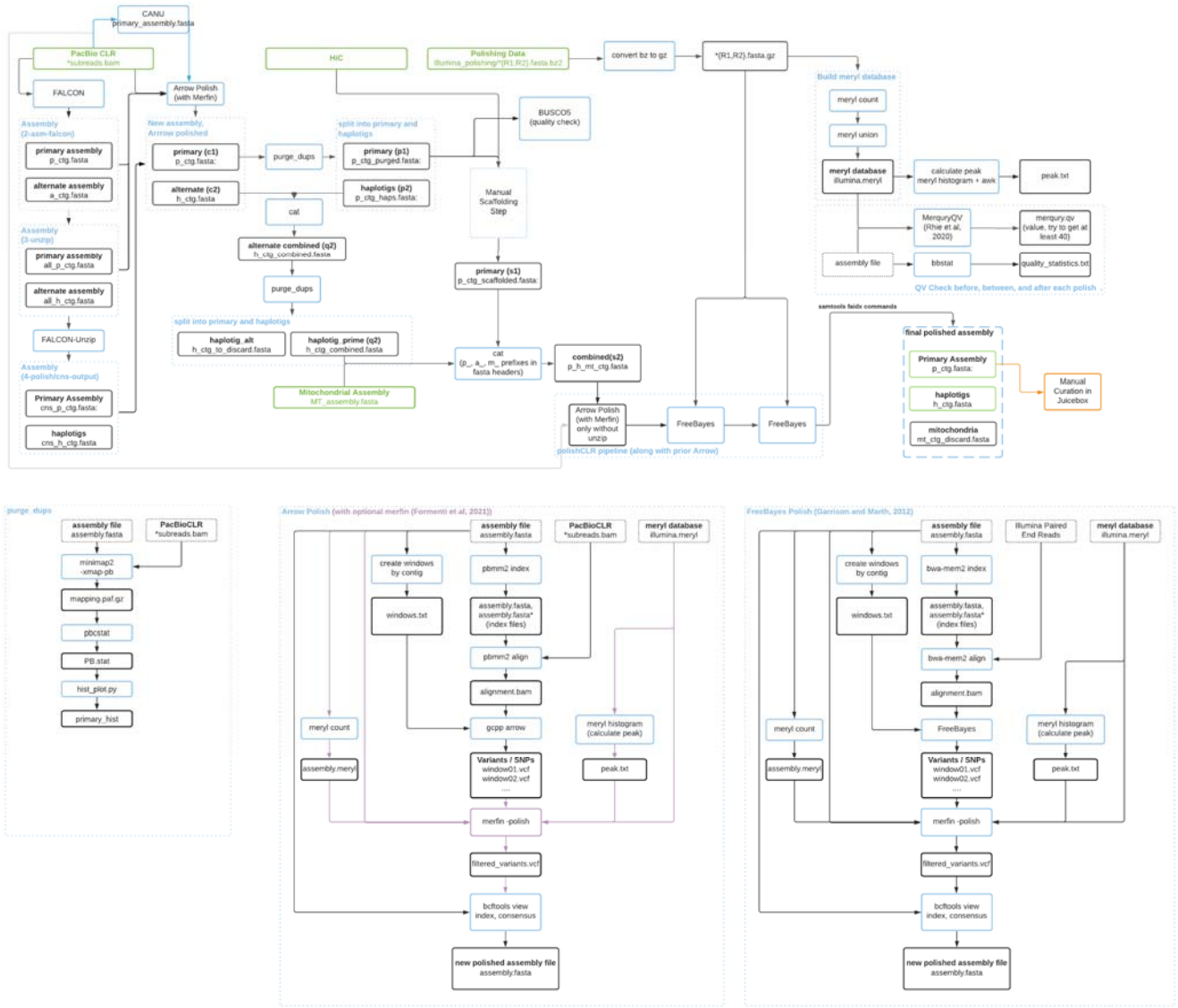
Watson, M., & Warr, A. (2019). Errors in long-read assemblies can critically affect protein prediction. *Nat Biotechnol*, *37*(2), 124-126. https://doi.org/10.1038/s41587-018-0004-z

Wratten, L., Wilm, A., & Goke, J. (2021). Reproducible, scalable, and shareable analysis pipelines with bioinformatics workflow managers. *Nat Methods*, *18*(10), 1161-1168. https://doi.org/10.1038/s41592-021-01254-9

Figures & Tables



288 **Figure 1.** Diagram of polishCLR workflow for three input cases. Polishing steps 1 and 2 are run
289 separately to accommodate an optional scaffolding step. Blue arrows indicate processes while
290 black boxes indicate products. The dotted arrow indicates that the manual scaffolding step is
291 optional and not within the scope of this pipeline.
292



Supplemental Figure 1. Detailed diagram of the polishCLR pipeline. PacBio CLR long reads, Hi-C data, Illumina short reads, and organellar input data are shown in the green boxes. A detailed view of the Arrow and FreeBayes polish steps are expanded below separately, with the optional Merfin filtering during the Arrow polishing step shown in purple. Merfin filtering is part of all FreeBayes polishing steps.

Case	Input stage of Falcon assembly	Input Genome Size (Mb) / Number of contigs	Starting QV	Final QV	CPU hours	Output Genome Size (Mb) / Number of contigs
1	2-asm-falcon/	501.287 / 789	31.8218	40.3033	211.3	500.578 / 799
2	3-unzip/	515.499 / 1125	31.8492	38.9997	224.2	511.878 / 1022
3	4-polish/	509.063 / 882	38.8556	41.9163	195.0	509.052 / 882

Supplemental Table 1. The polishCLR workflow was benchmarked on the primary contigs of *Helicoverpa zea* generated by FALCON (Chin et al. 2016). Metrics for each assembly include starting pseudo-haploid primary and alternate combined genome size (Mb) and number of contigs, initial quality scores, CPU hours through the pipeline final quality scores, and final genome size and number of contigs. This table provides an indication of scalability of the pipeline on a SLURM managed HPC.