Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 10 August 2022 doi:10.20944/preprints202208.0191.v1

Snakemake Workflows for Long-read Bacterial Genome Assembly and Evaluation

Peter Menzel [a] ✉
[a]Labor Berlin - Charité Vivantes GmbH, Sylter Str. 2, Berlin, Germany

With the advancement of long-read sequencing technologies and their more widespread use for bacterial genomics, several methods for generating genome assemblies from error-prone long reads have been developed. These are complemented by various tools for assembly polishing using either long reads, short reads, or reference genomes. End users are therefore left with a plethora of possible combinations of programs for obtaining a final trusted assembly. Hence, there is also the need for measuring completeness and accuracy of such assemblies, for which, again, several evaluation methods implemented in various programs are available. In order to automatically run all these programs, I developed two workflows for the workflow management system Snakemake for bacterial genome assembly and evaluation of assemblies, which provide end users with an easy-to-run method for both tasks. The workflows are available as open source software under the MIT license at github.com/pmenzel/ont-assembly-snake and github.com/pmenzel/score-assemblies.

bacterial genomics | *de novo* assembly | Oxford Nanopore | Snakemake
Correspondence: ***pmenzel@gmail.com***

Introduction

In recent years, long-read sequencing data from Oxford Nanopore (ONT) sequencers has been widely available for the assembly of microbial genomes. However, *de novo* genome assemblies from ONT sequencing reads contain systematic errors introduced by the comparatively high error rate in the sequencing reads. The major error type are insertions and deletions ("indels") in nucleotide homopolymers, which occur if bases are either skipped or read multiple times during sequencing due to varying translocation speed (1, 2). Indels often cause frameshifts in translated protein-coding sequences, leading to issues with *de novo* gene prediction and genome annotation (3). Such assembly errors can be corrected using the original long reads for "polishing" the primary *de novo* assembly in a consensus step. Further, additional short-read sequencing data, *e.g.*, from Illumina, can also be used to correct assembly errors, either by using both long and short reads together in a "hybrid assembly" (4), or by polishing the long-read assembly using the short reads. If other genome assemblies for the particular organism are already available, or assemblies from phylogenetically closely related bacterial strains or species, assembly errors can also be corrected by using such reference nucleotide or amino acid sequences. In recent years, multiple programs were developed for long-read assembly and polishing, resulting in increasingly complex options for end users on how to arrive at a final trusted assembly.

This leads straight to the issue of evaluating the generated assemblies for their correctness. Again, several methods exist for measuring assembly quality by various metrics, often by comparison to already available genome assemblies from the same or closely related species (5, 6).

The workflow manager Snakemake (7) is based on so-called Snakefiles, in which the user defines rules with instructions on how to run programs for creating output files, often via intermediate steps, from a given set of input files. While this approach is similar in spirit to the build system GNU Make, Snakemake offers additional syntax features as well as support for executing the programs on job processing systems. Snakemake also integrates the package management system conda (conda.io), so that individual software environments can be defined on a per-rule basis, allowing the automatic installation of the required programs upon executing a Snakemake workflow. Many bioinformatics programs are already included in the "bioconda" channel (8).

Here, I describe two easy-to-use Snakemake workflows: **ont-assembly-snake** for generating *de novo* bacterial genome assemblies using long and short sequencing reads, and **score-assemblies** for evaluating assemblies by various metrics and generating a summary report for an easy comparison of the measurements.

Results

When Snakemake parses the rules defined in a Snakefile, it creates a directed acyclic graph (DAG). Each rule defines the code for generating a set of output files from a set of input files. Usually, the creation of the DAG starts on the inputs for a special "all" rule, comprising the list of desired output files for the whole workflow. The DAG connects them, often through intermediate outputs, to the input files containing the sequencing data. This list of final output files is typically created by the workflow itself, for example from the names of sequencing data files located in a given folder. However, Snakemake's creation and traversal of the DAG for executing rules allows for an easy and very flexible formulation of the chain of execution of rules by the user himself, without the need for changing the workflow code. This ability is bit in contrast to "feed-forward" workflow managers, in which the same defined workflow is executed on each input dataset.

The two presented workflows each consist of a Snakefile containing the workflow logic, as well as accessory files for defining conda environments and scripts for report generation using R Markdown (9).

© 2022 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 10 August 2022 doi:10.20944/preprints202208.0191.v1

ont-assembly-snake

Following programs are included in workflow version 1.0:

Read filtering		
Filtlong	github.com/rrwick/Filtlong	
Long read assembly		
Raven	github.com/lbcb-sci/raven	(10)
Flye	github.com/fenderglass/Flye	(11)
Miniasm	github.com/lh3/miniasm	(12)
Hybrid assembly		
Unicycler	github.com/rrwick/Unicycler	(13)
Long read polishing		
Racon	github.com/lbcb-sci/racon	(14)
Medaka	github.com/nanoporetech/medaka	
Short read polishing		
Pilon	github.com/broadinstitute/pilon	(15)
Polypolish	github.com/rrwick/Polypolish	(16)
Reference-based polishing		
Homopolish	github.com/ythuang0522/homopolish	(17)
Proovframe	giuthub.com/thackl/proovframe	(18)

This workflows make use of the above mentioned possibility to let the user define the chain of rule execution, by simply creating empty folders with names containing keywords that describe the desired read filtering, assembly as well as polishing steps. These folder names are used as input for the "all" rule of the Snakefile. When processing the DAG, Snakemake follows the paths from outputs to inputs and execute the rules for each step, creating intermediate assemblies up to the final assemblies defined by the user. The keywords for some programs can be extended by appending numbers, *e.g.*, for setting the number of iterations for assembly polishing.
These three examples show the chaining of keywords:

Example folder names

```
sample1+filtlongMB500_flye+racon2+medaka
sample2_flye4+medaka+polypolish+proovframe
sample3+filtlongPC90_raven2+medaka+pilon
```

The first assembly is done by reducing the sequencing reads to comprise only 500 megabases from the longest reads with highest quality scores. Those reads are then assembled with Flye, followed by two rounds of polishing with Racon, followed by long-read polishing with Medaka. The second assembly uses all sequencing reads, which are assembled by Flye including four rounds of internal polishing, followed by Medaka, short read polishing with Polypolish and lastly polishing with Proovframe, which requires at least one provided reference proteome. The third sample uses only the sequencing reads comprising the top 90% bases, which are assembled using Raven with two rounds of internal polishing, followed by long and short read polishing with Medaka and Pilon, respectively.
In principle it is very easy to include additional programs in the workflow by simply adding additional rules to the Snakefile.

score-assemblies

This workflow is a classical workflow that runs the same set of programs on the user-provided assembly files in FASTA format, for example the assemblies generated by ont-assembly-snake.
Following programs are included in version 1.0:

Quast	quast.sourceforge.net/quast	(19)
BUSCO	busco.ezlab.org	(20)
DNAdiff (MUMmer)	github.com/mummer4/mummer	(21)
NucDiff	github.com/uio-cels/NucDiff	(22)
Pomoxis	github.com/nanoporetech/pomoxis	
IDEEL, using:	github.com/mw55309/ideel	
Prodigal	github.com/hyattpd/Prodigal	(23)
Diamond	github.com/bbuchfink/diamond	(24)

All programs, except BUSCO and IDEEL, require one or more reference assemblies to be provided, with which the input assemblies will be compared. Quast provides a summary of the usual assembly metrics, such as number of contigs or N50, but also aligns the assembly to a reference genome, giving statistics about genome rearrangements, mismatches and indels. Similarly, DNAdiff and NucDiff also report alignment metrics. The program assess_assembly from the Pomoxis package calculates an overall Q-score and error rate, whereas assess_homopolymers calculates the fraction of correct homopolymers for each homopolymer length. BUSCO measures the completeness of single-copy clade-specific core genes in the assemblies. By default, it uses a set of core genes that are near universal in all bacteria, but one can also provide a taxon name for selecting a clade-specific dataset. The IDEEL module will run Prodigal for prediction of open reading frames (ORFs), which will be searched against the Uniprot (25) "Sprot" database and, optionally, a user-provided set of reference protein sequences. It then collects statistics about the number and length of found alignments to database sequences, which gives an idea about the completeness and accuracy of predicted ORFs.
While some of the included programs perform similar tasks, such as approximating a whole genome alignment, and could therefore be considered redundant, they differ in the used heuristics and reported metrics. Fortunately, the execution time of all programs is quite fast on bacterial genomes, which are only a few megabases long, so that it is computationally not a problem to include them all.
The output of the workflow consists of several summary tables and plots, as well as an overall report in HTML format. The report contains sortable tables with the reported metrics from each program as well as a summary table, which contains all metrics per assembly in one line, allowing an easy comparison of the assemblies.
Optionally, the workflow can also run Bakta (26) for comprehensive of annotation of protein-coding and non-coding genes in each genome, which in turn uses various programs and databases for gene prediction and homology search. This annotation can also be used for submission to genome databases.

References

1. Ryan R Wick, Louise M Judd, and Kathryn E Holt. Performance of neural network basecalling tools for oxford nanopore sequencing. *Genome biology*, 20(1):1–10, 2019.
2. Clara Delahaye and Jacques Nicolas. Sequencing dna with nanopores: Troubles and biases. *PLOS ONE*, 16(10):1–29, 10 2021. doi: 10.1371/journal.pone.0257521.
3. Mick Watson and Amanda Warr. Errors in long-read assemblies can critically affect protein prediction. *Nature Biotechnology*, 37(2):124–126, February 2019. ISSN 1546-1696.
4. Zhao Chen, David L. Erickson, and Jianghong Meng. Benchmarking hybrid assembly approaches for genomic analyses of bacterial pathogens using illumina and oxford nanopore sequencing. *BMC genomics*, 21:631, September 2020. ISSN 1471-2164. doi: 10.1186/s12864-020-07041-8.
5. Ryan R. Wick and Kathryn E. Holt. Benchmarking of long-read assemblers for prokaryote whole genome sequencing. *F1000Research*, 8:2138, 2019.
6. Mantas Sereika, Rasmus Hansen Kirkegaard, Søren Michael Karst, Thomas Yssing Michaelsen, Emil Aarre Sørensen, Rasmus Dam Wollenberg, and Mads Albertsen. Oxford nanopore r10.4 long-read sequencing enables the generation of near-finished bacterial genomes from pure cultures and metagenomes without short-read or reference polishing. *Nature methods*, 19:823–826, July 2022. ISSN 1548-7105. doi: 10.1038/s41592-022-01539-7.
7. Johannes Köster and Sven Rahmann. Snakemake—a scalable bioinformatics workflow engine. *Bioinformatics*, 28(19):2520–2522, 2012.
8. Björn Grüning, Ryan Dale, Andreas Sjödin, Brad A Chapman, Jillian Rowe, Christopher H Tomkins-Tinch, Renan Valieris, and Johannes Köster. Bioconda: sustainable and comprehensive software distribution for the life sciences. *Nature methods*, 15(7):475–476, 2018.
9. JJ Allaire, Yihui Xie, Jonathan McPherson, Javier Luraschi, Kevin Ushey, Aron Atkins, Hadley Wickham, Joe Cheng, Winston Chang, and Richard Iannone. *rmarkdown: Dynamic Documents for R*, 2022. R package version 2.14.
10. Robert Vaser and Mile Šikić. Time-and memory-efficient genome assembly with raven. *Nature Computational Science*, 1(5):332–336, 2021.
11. Mikhail Kolmogorov, Jeffrey Yuan, Yu Lin, and Pavel A Pevzner. Assembly of long, error-prone reads using repeat graphs. *Nature biotechnology*, 37(5):540–546, 2019.
12. Heng Li. Minimap and miniasm: fast mapping and de novo assembly for noisy long sequences. *Bioinformatics*, 32(14):2103–2110, 2016.
13. Ryan R Wick, Louise M Judd, Claire L Gorrie, and Kathryn E Holt. Unicycler: resolving bacterial genome assemblies from short and long sequencing reads. *PLoS computational biology*, 13(6):e1005595, 2017.
14. R Vaser, I Sović, N Nagarajan, and M Šikić. Fast and accurate de novo genome assembly from long uncorrected reads. *Genome Res*, 27:737–746, 05 2017. doi: 10.1101/gr.214270.116.
15. BJ Walker, T Abeel, T Shea, M Priest, A Abouelliel, S Sakthikumar, CA Cuomo, Q Zeng, J Wortman, SK Young, and AM Earl. Pilon: an integrated tool for comprehensive microbial variant detection and genome assembly improvement. *PLoS One*, 9:e112963, 2014. doi: 10.1371/journal.pone.0112963.
16. Ryan R Wick and Kathryn E Holt. Polypolish: Short-read polishing of long-read bacterial genome assemblies. *PLoS computational biology*, 18(1):e1009802, 2022.
17. Yao-Ting Huang, Po-Yu Liu, and Pei-Wen Shih. Homopolish: a method for the removal of systematic errors in nanopore sequencing by homologous polishing. *Genome biology*, 22 (1):1–17, 2021.
18. Thomas Hackl, Florian Trigodet, A Murat Eren, Steven J Biller, John M Eppley, Elaine Luo, Andrew Burger, Edward F DeLong, and Matthias G Fischer. proovframe: frameshift-correction for long-read (meta) genomics. *bioRxiv*, 2021.
19. Alexey Gurevich, Vladislav Saveliev, Nikolay Vyahhi, and Glenn Tesler. Quast: quality assessment tool for genome assemblies. *Bioinformatics*, 29(8):1072–1075, 2013.
20. Mosè Manni, Matthew R Berkeley, Mathieu Seppey, Felipe A Simão, and Evgeny M Zdobnov. Busco update: novel and streamlined workflows along with broader and deeper phylogenetic coverage for scoring of eukaryotic, prokaryotic, and viral genomes. *Molecular Biology and Evolution*, 38(10):4647–4654, 2021.
21. Guillaume Marçais, Arthur L Delcher, Adam M Phillippy, Rachel Coston, Steven L Salzberg, and Aleksey Zimin. Mummer4: A fast and versatile genome alignment system. *PLoS computational biology*, 14(1):e1005944, 2018.
22. K Khelik, K Lagesen, GK Sandve, T Rognes, and AJ Nederbragt. NucDiff: in-depth characterization and annotation of differences between two sets of DNA sequences. *BMC Bioinformatics*, 18:338, Jul 2017. doi: 10.1186/s12859-017-1748-z.
23. D Hyatt, GL Chen, PF Locascio, ML Land, FW Larimer, and LJ Hauser. Prodigal: prokaryotic gene recognition and translation initiation site identification. *BMC Bioinformatics*, 11:119, 2010. doi: 10.1186/1471-2105-11-119.
24. B Buchfink, C Xie, and DH Huson. Fast and sensitive protein alignment using DIAMOND. *Nat Methods*, 12:59–60, Jan 2015. doi: 10.1038/nmeth.3176.
25. UniProt Consortium. Uniprot: the universal protein knowledgebase in 2021. *Nucleic acids research*, 49:D480–D489, January 2021. ISSN 1362-4962. doi: 10.1093/nar/gkaa1100.
26. Oliver Schwengers, Lukas Jelonek, Marius Alfred Dieckmann, Sebastian Beyvers, Jochen Blom, and Alexander Goesmann. Bakta: rapid and standardized annotation of bacterial genomes via alignment-free sequence identification. *Microbial genomics*, 7, November 2021. ISSN 2057-5858. doi: 10.1099/mgen.0.000685.