




Minimap2-rs: Rust bindings for Minimap2

Joseph Guhlin ^{1,2}

¹ Genomics Aotearoa, University of Otago, Dunedin, New Zealand ² Department of Biochemistry,
University of Otago, Dunedin, New Zealand

DOI: [10.xxxxxx/draft](https://doi.org/10.xxxxxx/draft)

Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

Editor: 

Submitted: 06 October 2025

Published: unpublished

License

Authors of papers retain copyright
and release the work under a
Creative Commons Attribution 4.0
International License ([CC BY 4.0](https://creativecommons.org/licenses/by/4.0/)).

Summary

Long-read sequence alignment underpins modern genomic research, enabling highly contiguous de novo assemblies, structural variant detection, RNA isoform exploration, and many other applications. Minimap2, the long read sequence aligner, stands at the center of these developments, offering fast alignment for long and noisy reads. In parallel, the Rust programming language has seen rapid adoption in the scientific community, providing memory safety, concurrency, and high performance. We present Minimap2-rs, a suite of Rust crates that interface with Minimap2, lowering the barrier to using this versatile aligner in Rust-based applications.

The top-level minimap2 crate offers an opinionated, flexible idiomatic API that manages memory and simplifies foreign function interface (FFI) complexities. A companion Python library, minimappers2, returns high-performance dataframes via Polars, while allowing for intuitive multithreading and data handling. Meanwhile, the minimap2-sys crate provides direct, low-level bindings to Minimap2 for specialized use cases. This modular design facilitates both rapid prototyping and production-ready pipelines, illustrated by two example implementations of multithreading (multi-producer single-consumer channels and Rayon). Continuous integration across x86_64 and arm64 architectures ensures reliability on Linux, macOS, and Android, with tested support for musl-based portability. Already adopted by multiple tools for genome size estimation, tandem repeat genotyping, and transcriptome analysis, Minimap2-rs offers a stable, open-source foundation for building Rust-powered bioinformatics solutions while preserving Minimap2's speed and flexibility.

Statement of Need

Long-read sequence alignment has changed the foundation of recent genomic advances. Thanks to long-read sequence alignment, advances such as the telomere-to-telomere Human genome assembly (Nurk et al., 2022) were completed, the cost to assemble a *de novo* genome has significantly decreased and contiguity has increased (Cheng et al., 2021; Kolmogorov et al., 2019; Koren & Phillippy, 2015; Murigneux et al., 2020; Reis et al., 2023), and is used for RNA-based applications (Jain et al., 2022). The primary algorithm supporting nearly all of these advancements is found in the tool Minimap2, which can align long-read sequences to large references, accounting for the high error rate that may be present in long reads (Li, 2018, 2021). Minimap2 can also accommodate spliced alignments, short-read alignments, and assembly-to-assembly alignments.

Meanwhile, the Rust programming language has seen significant adoption, including amongst scientists (Bugden & Alahmar, 2022; JE & CT, 2020; Matsakis & Klock, 2014). Rust brings memory safety with a strong ownership model and performance to applications and libraries, and consistent build tooling (Bugden & Alahmar, 2022). This ultimately allows for highly portable applications with strong concurrency idioms, encouraging high-performance applications by default. With the growing adoption of Rust, there are now several tools and libraries in Rust for bioinformatics (Buffalo, 2024; Chan, 2024; Huey & Abdennur, 2024; Köster, 2016).

Using Minimap2 directly in Rust requires complicated foreign function interface (FFI) calls. Here, I present Minimap2-rs, which provides FFI bindings to the Minimap2 library, exposing the underlying API directly to Rust and providing a more idiomatic Rust API to work with Minimap2, easing the barrier to entry for others to interface with Minimap2. Minimap2-rs is open source and publicly available through crates.io, the standard library registry for Rust, and GitHub. Minimap2-rs is already used by multiple tools for sequence cleaning (De Coster & Rademakers, 2023), genotyping tandem repeats (De Coster et al., 2024), long-read transcriptome quantification (Jousheghani & Patro, 2024), and long-read-based genome size estimation (Hall & Coin, 2024). Minimap2-rs compiles on the x86_64 and arm64 architectures for Linux, Mac, and Android.

Implementation

Minimap2 is a command-line tool and library, and an official Python library, Mappy, is available separately. Minimap2 uses single-instruction multiple-data (SIMD) CPU features to achieve better speed and parallelization, allowing it to work with large datasets and long-reads. Minimap2-rs consists of three libraries (called 'crates' in Rust): Minimap2, minimappers2, and minimap2-sys. The primary point of entry, the Minimap2 Rust crate (further referred to as minimap2-rs) provides an opinionated, memory-safe library for working with Minimap2 functions via the foreign function interface (FFI) and the given output in Rust data structures (structs), converting base types between C and Rust automatically. Minimappers2 is a Python library wrapping the Minimap2 Rust library, providing seamless multi-threading and returning results via the high-performance, memory-efficient Polars data frames library. Minimap2-sys wraps the Minimap2 library and allows direct interface with the C functions with Rust via unsafe code (Matsakis & Klock, 2014). A functional fourth tool is available, fakeminimap2, which serves as a functioning exemplar of two common multithreading approaches.

Minimap2-rs is the primary interface for using Minimap2 in downstream Rust applications. An aligner struct is created, and presets, mapping, and indexing options are configured through this struct. All presets from the Minimap2 command-line software (e.g., map-ont, map-pb, map-hifi, asm20) and low-level access to further index and mapping options in the Minimap2 C library are available. The crate is configurable with optional features (1). Minimap2 is built using Rust's tooling, thus adding this library only requires adding it as a dependency. The code repository provides two examples of multithreading implementation with minimap2-rs using multi-producer single-consumer channels (MPSC) and parallelization using the popular Rayon crate.

Table 1: Rust crate-level features available for minimap2-rs. {} Rust features enable conditional compilation and reduce dependencies when not needed. 1 Enabled by default.

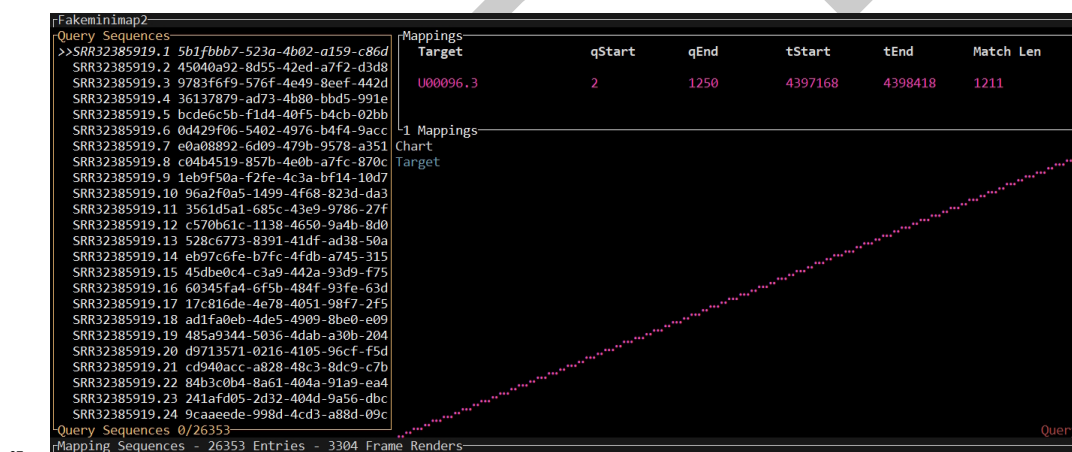
Feature	Description
map-file 1	Adds a convenience function to parse and map a FASTA/Q file to a provided index or reference.
htslib	Supports returning results as HTS records.
curl	Enables curl support for htslib.
simd	Enables the SIMD Everywhere library in minimap2 compilation.
zlib-ng	Uses zlib-ng in the minimap2 index for faster reading of compressed files.
static	Builds minimap2-rs as a static library in Rust.
sse2only	Builds minimap2 with only SSE2 support.

Minimappers2 is an alternative to Mappy. It provides an interface to minimap2-rs via Python, returns results as high-performance data frames, and supports multithreading. This provides an alternative Pythonic interface to Minimap2, while adding native multi-threading. The Python interface is created using PyO3. The results are returned as a Polars data frame, and the

80 function `.to_pandas()` supports conversion to Pandas data frames. The number of threads to
81 use is set with `.threads(n)`, where n specifies the number of threads.

82 Minimap2-sys is the low-level direct FFI interface to the Minimap2 library, consisting primarily of
83 unsafe function calls, automatically generated bindings from the bindgen crate, and some minor
84 things to improve support and prevent memory leaks in downstream applications. The majority
85 of the code here is in the build.rs file, which is responsible for building the Minimap2 C library,
86 enabling architecture-specific features, and allowing Minimap2 to be built to support Rust
87 interfacing while supporting additional platforms. The currently supported operating systems
88 are Linux and Mac OS, as well as the architectures x86_64, aarch64, and arm64. Maximum
89 portability of downstream binaries can be achieved by compiling for musl, an alternative libc
90 implementation. These are tested with continuous integration to prevent any reversions or loss
91 of system support. All features are also tested at this time. Experimental Android support is
92 also tested on aarch64 and x86_64 via continuous integration.

93 Fakeminimap2 provides further in-depth examples of multi-threading using both MPSC channels
94 and Rayon. Further, Fakeminimap2 provides a terminal user-interface (TUI), allowing for
95 interactive tables and visualizations of alignments (). This TUI also supports mouse interaction
96 in supported terminals.



87 Results

88 This project provides an idiomatic and memory-safe interface from the popular systems-level
89 Rust language to the widely used long-read mapping tool Minimap2. Minimap2-rs is amenable
90 to multi-threading, which is essential for long reads and large genomes, and provides well-
91 commented working examples of two different multi-threading approaches. With continuous
92 integration, this will serve as a stable base for building Rust-based long-read sequencing
93 applications.

94 Acknowledgements

95 The author wishes to thank Wouter De Coster for kicking off this project with a tweet and
96 Heng Li for creating Minimap2 and supporting this library. The author also thank the numerous
97 contributors to GitHub, both through feature requests, finding bugs, and especially contributions
98 via pull requests. Finally, a thank you to Peter Dearden for helping with manuscript prep and
99 proof-reading.

Conflict of Interest

None declared.

Funding

This work was funded by a grant from Genomics Aotearoa (High Quality Genomes II), which is itself funded by the New Zealand Ministry of Business, Innovation and Employment.

Data Availability

The software repository is available on GitHub at <https://github.com/jguhlin/minimap2-rs> and all published versions of this crate are available at crates.io at <https://crates.io/crates/minimap2>. Minimappers2 is available through Pypi at <https://pypi.org/project/minimappers2/>. The software is available under the MIT or Apache 2.0 License, at the user's discretion. Fakeminimap2 data displayed used Nanopore reads from NCBI SRA SRR32385919, and whole genome sequence NCBI Genbank U00096.3.

Buffalo, V. (2024). GRanges: A rust library for genomic range data. *bioRxiv*, 2024–2005.

Bugden, W., & Alahmar, A. (2022). Rust: The programming language for safety and performance. *arXiv Preprint arXiv:2206.05503*.

Chan, K. O. (2024). Next-generation bioinformatics: An ultrafast and user-friendly tool for phylogenomic data exploration. *Molecular Ecology Resources*, 24(7), e13993.

Cheng, H., Concepcion, G. T., Feng, X., Zhang, H., & Li, H. (2021). Haplotype-resolved de novo assembly using phased assembly graphs with hifiasm. *Nature Methods*, 18(2), 170–175.

De Coster, W., Höijer, I., Bruggeman, I., D'Hert, S., Melin, M., Ameur, A., & Rademakers, R. (2024). Visualization and analysis of medically relevant tandem repeats in nanopore sequencing of control cohorts with pathSTR. *Genome Research*, 34(11), 2074–2080.

De Coster, W., & Rademakers, R. (2023). NanoPack2: Population-scale evaluation of long-read sequencing data. *Bioinformatics*, 39(5), btad311.

Hall, M. B., & Coin, L. J. M. (2024). Genome size estimation from long read overlaps. *bioRxiv*, 2024.11.27.625777. <https://doi.org/10.1101/2024.11.27.625777>

Huey, J. D., & Abdennur, N. (2024). Bigtools: A high-performance BigWig and BigBed library in rust. *Bioinformatics*, 40(6).

Jain, M., Abu-Shumays, R., Olsen, H. E., & Akeson, M. (2022). Advances in nanopore direct RNA sequencing. *Nature Methods*, 19(10), 1160–1164.

JE, O., & CT, T. (2020). Why scientists are turning to rust. *Nature*, 588, 185.

Jousheghani, Z. Z., & Patro, R. (2024). Oarfish: Enhanced probabilistic modeling leads to improved accuracy in long read transcriptome quantification. *bioRxiv*. <https://doi.org/10.1101/2024.02.28.582591>

Kolmogorov, M., Yuan, J., Lin, Y., & Pevzner, P. A. (2019). Assembly of long, error-prone reads using repeat graphs. *Nature Biotechnology*, 37(5), 540–546.

Koren, S., & Phillippy, A. M. (2015). One chromosome, one contig: Complete microbial genomes from long-read sequencing and assembly. *Current Opinion in Microbiology*, 23, 110–120.

Köster, J. (2016). Rust-bio: A fast and safe bioinformatics library. *Bioinformatics*, 32(3),

- 152 444–446.
- 153 Li, H. (2018). Minimap2: Pairwise alignment for nucleotide sequences. *Bioinformatics*, 34(18),
154 3094–3100.
- 155 Li, H. (2021). New strategies to improve minimap2 alignment accuracy. *Bioinformatics*,
156 37(23), 4572–4574.
- 157 Matsakis, N. D., & Klock, F. S. (2014). The rust language. *ACM SIGAda Ada Letters*, 34(3),
158 103–104.
- 159 Murigneux, V., Rai, S. K., Furtado, A., Bruxner, T. J., Tian, W., Harliwong, I., Wei, H.,
160 Yang, B., Ye, Q., Anderson, E., & others. (2020). Comparison of long-read methods for
161 sequencing and assembly of a plant genome. *GigaScience*, 9(12), giaa146.
- 162 Nurk, S., Koren, S., Rhie, A., Rautiainen, M., Bzikadze, A. V., Mikheenko, A., Vollger, M. R.,
163 Altemose, N., Uralsky, L., Gershman, A., & others. (2022). The complete sequence of a
164 human genome. *Science*, 376(6588), 44–53.
- 165 Reis, A. L. van der, Beckley, L. E., Olivar, M. P., & Jeffs, A. G. (2023). Nanopore short-
166 read sequencing: A quick, cost-effective and accurate method for DNA metabarcoding.
167 *Environmental DNA*, 5(2), 282–296.