PREPRINT · NOT PEER REVIEWED

Claude Opus 5.5 for bioinformatics: SwiftSNP, a byte-identical snp-dists that is 336-fold faster than the fastest equivalent tool

Joe Christensen (ORCID 0009-0007-0908-1299)
Translational Neuropsychiatry Unit (TNU), Department of Clinical Medicine, Aarhus University, Denmark
Correspondence: joechristensen@clin.au.dk

Abstract

SwiftSNP is a drop-in replacement for snp-dists, the small C program that many genomic epidemiology pipelines use to compute pairwise SNP distance matrices. On five published bacterial and SARS-CoV-2 alignments of 100 to 8,596 genomes it wrote the same matrix as snp-dists, byte for byte, and was 336-fold faster than the fastest other tool that reproduced that matrix (geometric mean at two threads). The matrix for 8,596 SARS-CoV-2 genomes took snp-dists 27 minutes and SwiftSNP 0.23 s on an AVX-512 CPU; on an ordinary AVX2 laptop SwiftSNP was 53-fold faster than snp-dists at one thread. snp-dists was slow because it compares every pair of genomes twice, at every column and one byte at a time, although in a *Vibrio cholerae* core-genome alignment fewer than one column in 2,000 can differ, and because it prints each matrix cell with its own printf call. I built SwiftSNP with correctness-gated coding agents (Claude Opus 5.5): every change had to keep the output byte-identical to snp-dists. No output byte differed in 188,700 random tests or in 67 real-data runs on three builds, for plain FASTA and gzip input and the snp-dists options. Multithreaded snp-dists writes corrupt matrices; a 21-line patch fixes it.

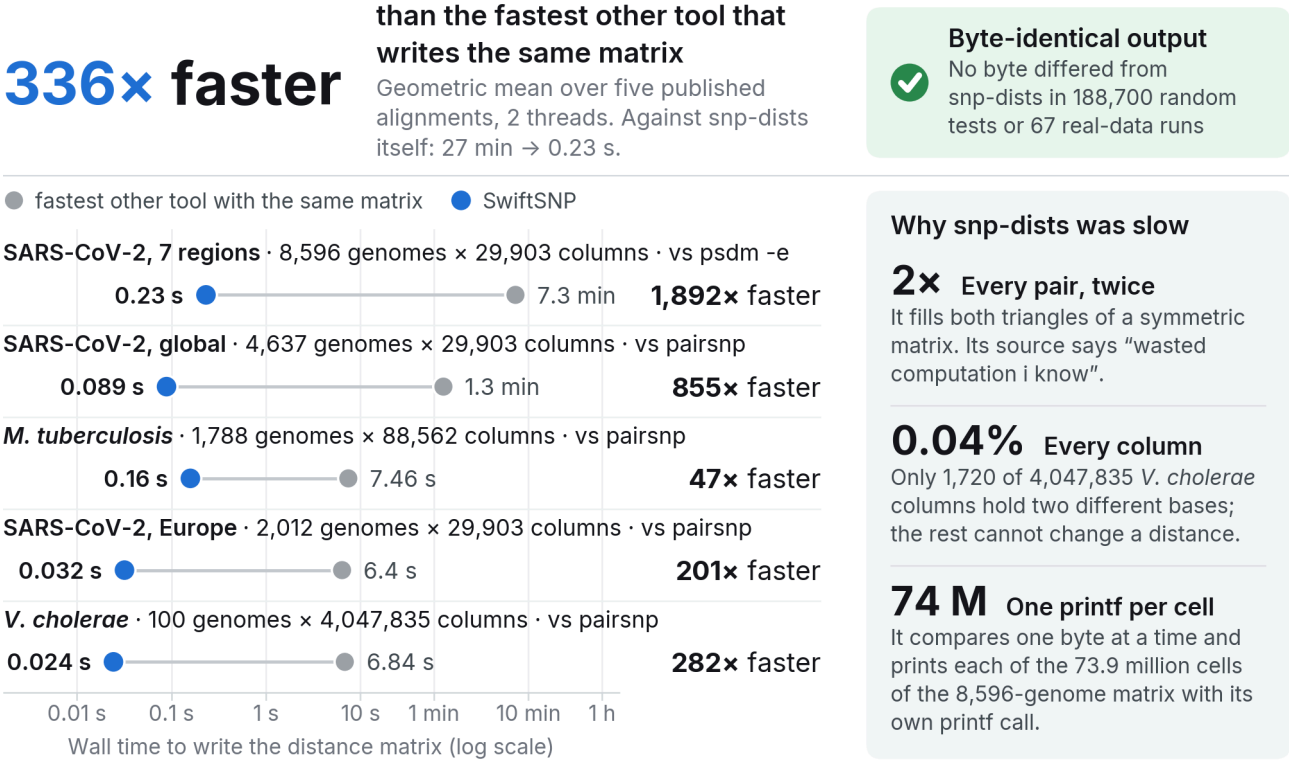

Figure 1. SwiftSNP is 47- to 1,892-fold faster than the fastest other tool that writes the same matrix as snp-dists, 336-fold on average. Wall time of the fastest such tool on each of five published alignments (grey; psdm -e on 8,596 genomes, where pairsnp ran out of memory, and pairsnp elsewhere) and of SwiftSNP (blue), both at two threads on an AVX-512 cloud VM with 2 vCPUs, log scale. The headline is the geometric mean of the five speed-ups. Right: the three habits that made snp-dists slow (Section 2.4).

1 Introduction

Pairwise SNP distance matrices sit between the alignment and the public-health decision in bacterial and viral genomic epidemiology. Each entry counts the positions at which two isolates both carry an unambiguous nucleotide and differ. Pipelines cluster isolates with these counts, apply transmission thresholds such as the 5 and 12 SNPs proposed for *M. tuberculosis* (Walker et al. 2013) and build trees from them. snp-dists (Seemann et al. 2025) is a standard way to compute them, and its cost grows with the square of the number of genomes times the alignment length: on 8,596 SARS-CoV-2 genomes it took 27 minutes. Faster tools exist, pairsnp (Tonkin-Hill 2019, 2022) and psdm (Hall 2024), but some of them count IUPAC ambiguity codes or ignore a different set of characters, so switching to them changes the numbers that thresholds act on.

SwiftSNP writes the same matrix file as snp-dists, byte for byte. On five published alignments it was 336-fold faster than the fastest other tool that gives that matrix and 2,894-fold faster than snp-dists at two threads (Fig. 1), so a pipeline can switch without re-validating a single distance. The waste in snp-dists is simple. Computing each pair once, comparing only the columns that can differ and comparing 64 of them per machine word gave the first 26-fold; making parsing and printing as cheap as the comparisons gave most of the rest. The work also showed that multithreaded snp-dists writes corrupt matrices.

2 Results

2.1 SwiftSNP is three orders of magnitude faster on real alignments

On the five published alignments SwiftSNP was 2,894-fold faster than snp-dists at two threads and 1,567-fold faster at one thread (geometric means; Fig. 1, Table 1). The largest factors came on SARS-CoV-2. On 8,596 genomes snp-dists took 27 minutes and SwiftSNP 0.50 s at one thread (3,303-fold) and 0.23 s at two (7,112-fold). The smallest came on *V. cholerae*, where reading the 412 MB file dominates (497-fold and 1,214-fold), and on its gzip copy, which is bound by decompression (44-fold and 69-fold). Options that keep the default four-letter semantics ran 105- to 2,473-fold faster at one thread, and `-a`, which counts every character and runs on a generic code path, 31- to 59-fold (Supplementary Table S2).

Table 1. SwiftSNP wrote byte-identical output on every real alignment and was 44- to 7,112-fold faster than snp-dists. AVX-512 cloud VM with 2 vCPUs, default options; snp-dists 1.2.0 `-q -b -j 1` ran once, SwiftSNP best of three runs with the output written to a fresh file. Informative columns hold at least two different letters of A, C, G and T after upper-casing, which are the only columns that can add to a distance. Speed-ups use unrounded times; the geometric mean excludes the gzip copy.

Alignment	Genomes	Columns	Informative columns	snp-dists 1 thread (s)	SwiftSNP 1 thread (s)	Speed-up	SwiftSNP 2 threads (s)	Speed-up
SARS-CoV-2, 7 regions	8,596	29,903	13,360	1,648.4	0.499	3,303	0.232	7,112
SARS-CoV-2, global	4,637	29,903	8,889	496.2	0.164	3,026	0.089	5,598
M. tuberculosis	1,788	88,562	88,562	235.0	0.309	760	0.159	1,482
SARS-CoV-2, Europe	2,012	29,903	6,201	90.0	0.036	2,507	0.032	2,834
V. cholerae	100	4,047,835	1,720	29.5	0.059	497	0.024	1,214
V. cholerae, gzip -6	100	4,047,835	1,720	30.4	0.692	44	0.440	69
Geometric mean						**1,567**		**2,894**

Without AVX-512, SwiftSNP is still 42- to 180-fold faster than snp-dists on average, and 16- to 341-fold per alignment (Fig. 2). Built for AVX2 on the same VM it was 42-fold faster at one thread and 75-fold at two, because the fastest code paths, the sparse/dense column split (Section 2.4), the vectorised

column scan and the vector kernel, exist only for AVX-512. On a six-core AVX2 laptop it was 53-fold faster at one thread and 180-fold at twelve.

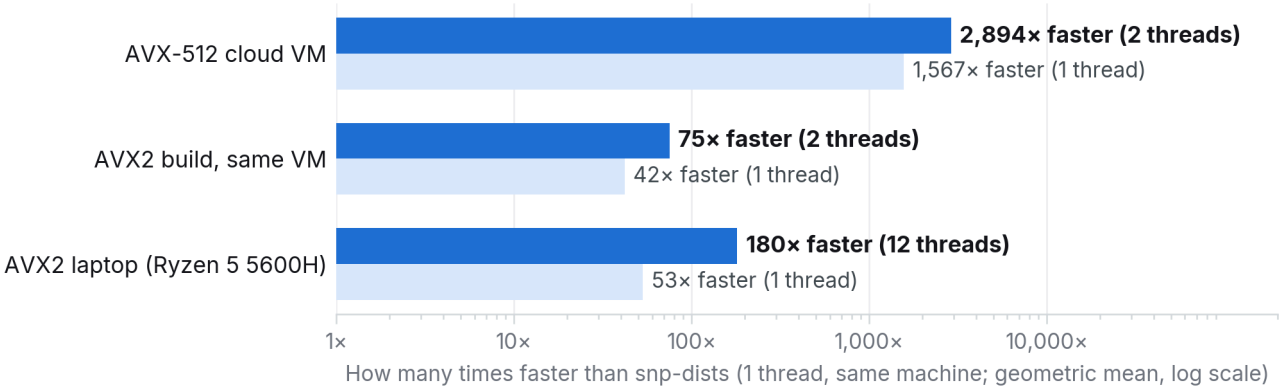

Figure 2. SwiftSNP is faster than snp-dists on every machine tested. Geometric mean over five published alignments of how many times faster SwiftSNP was than snp-dists 1.2.0 at one thread on the same machine, for the AVX-512 build on the cloud VM, the same source built for AVX2 (`-march=haswell`) on that VM, and the AVX2 build on a laptop (AMD Ryzen 5 5600H). Dark bars, all threads; light bars, one thread; log scale. Every output was byte-identical to `snp-dists -j 1`.

2.2 The output is identical, byte for byte

No output byte differed between SwiftSNP and `snp-dists -j 1` in any test (Table 2). In total 188,700 random inputs, with random options and thread counts, went through both programs, and four holdout sets that were never used during development matched every time. The real alignments were downloaded only after development had finished, and SwiftSNP's output on them was identical in all 67 runs, including 15 on a laptop with a different CPU vendor and code path. snp-dists itself gave the same bytes on both machines in all 20 cases checked, so the reference is deterministic.

Table 2. No output byte differed in any comparison. Each run compared the complete standard output of SwiftSNP with that of `snp-dists -q -j 1` on the same input and options; the random tests also required the same exit status on failure.

Test	Runs	Result
Random inputs against the live snp-dists (1–40 sequences, random options and threads)	188,700	all identical
Synthetic holdout sets, never used during development (4 sets)	24	all identical
Real alignments, AVX-512 build: default options on 6 inputs, 7 more option sets on 2; 1 and 2 threads	40	all identical
Real alignments, AVX2 build on the same VM; 1 and 2 threads	12	all identical
Real alignments, AVX2 laptop (AMD Zen 3); 1 and 12 threads	15	all identical
Renamed *V. cholerae* alignment with three bases changed	2	identical; 588 cells changed in both

2.3 SwiftSNP is 336-fold faster than the fastest tool that gives the same matrix

Only pairsnp (Python) and psdm with `-e` reproduced the snp-dists matrix on every alignment they completed, and SwiftSNP was 336-fold faster than the faster of the two (geometric mean over the five alignments at two threads; Figs. 1 and 3). pairsnp was the fastest exact tool on four alignments, from 7.5 s against 0.16 s on *M. tuberculosis* (47-fold) to 76 s against 0.089 s on SARS-CoV-2 global (855-fold). On 8,596 genomes it ran out of memory at 5.8 GiB, and psdm `-e` took 439 s against 0.23 s (1,892-fold).

The other tools differ from snp-dists by design. pairsnp-cpp counts IUPAC codes as partial matches and differed in 0.23–0.49% of SARS-CoV-2 cells, by up to 7 SNPs. psdm with default settings ignores

only N and gaps and differed in 29–41% of cells, by up to 151 SNPs. snp-dists itself was 1.6- to 1.9-fold faster at two threads than at one, but all nine of its two-thread matrices were corrupt (Section 2.5).

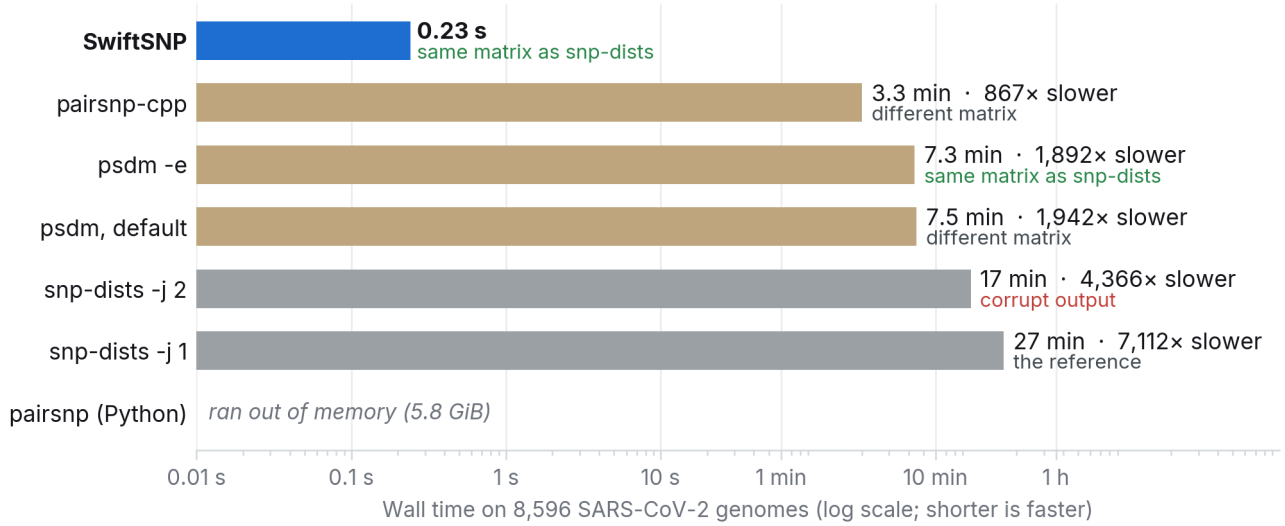

Figure 3. On 8,596 SARS-CoV-2 genomes every other tool is 867- to 7,112-fold slower than SwiftSNP, and only psdm -e also writes the snp-dists matrix. Wall time at two threads on the AVX-512 VM except `snp-dists -j 1`, the reference; log scale. Matrices were compared cell by cell with `snp-dists -j 1` after matching rows by name. pairsnp (Python) runs as a single process and ran out of memory. Supplementary Figure S1 shows all five alignments; versions are in Section 4.5.

2.4 snp-dists spends most of its time on work that cannot change the answer

Three habits of snp-dists account for most of its run time (Fig. 4).

It computes every pair twice. The matrix is symmetric, but snp-dists fills both triangles, and its source says so: “does full matrix, wasted computation i know”. Computing each pair once, on both threads, made the synthetic benchmark 3.7-fold faster.

It compares every column. A column can add to a distance only if it holds two different nucleotides, and most columns of a whole-genome alignment do not: 1,720 of 4,047,835 in *V. cholerae* and 21–45% of the 29,903 columns of the SARS-CoV-2 sets (Fig. 4a). Dropping the rest gave a further 3.1-fold, and comparing 64 columns per machine word, as 2-bit codes plus a validity bit counted with popcount, another 2.3-fold.

It prints one cell at a time. snp-dists writes each cell with its own printf call: 16 million calls for a 4,000-genome matrix and 74 million for the 8,596-genome SARS-CoV-2 matrix. Once comparisons were cheap, printing dominated, and formatting rows from a digit table on both threads cut the 4,000-genome benchmark set from 1.28 s to 0.30 s.

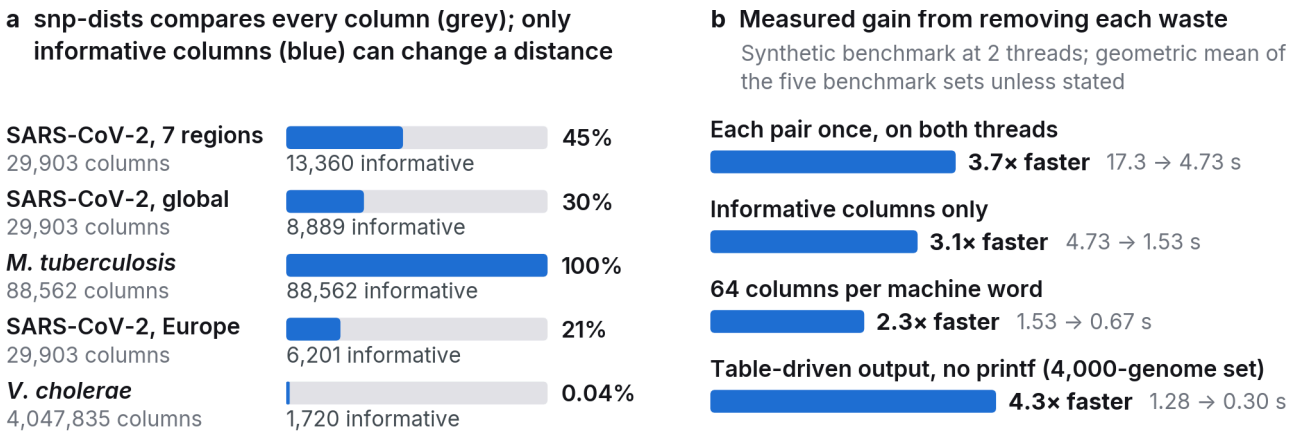

Figure 4. Most of what snp-dists computes cannot change the answer. (a) Columns of each real alignment (grey) and the informative columns among them (blue), which hold at least two different nucleotides and are the only ones that can add to a distance; snp-dists compares every column for every pair. (b) How much faster the code became as each source of waste was removed, measured in this order on the synthetic benchmark at two threads: geometric mean over the five benchmark sets, except the printf row, which is the 4,000-genome set alone.

These three changes took the synthetic benchmark from 17.3 s to 0.67 s, a 26-fold gain, and making parsing and printing as cheap as the comparisons gave most of the rest. SwiftSNP reads the whole file and parses it with a scanner that predicts the layout of each record and checks the prediction inside a vectorised column scan, falling back to the reference parser on anything unusual, and it decompresses gzip with libdeflate (Biggers 2026) on two threads. The one algorithmic idea beyond bit-parallel comparison is a sparse/dense column split. A column whose minority set M is small separates only pairs that contain a member of M, so its cost is proportional to $|M|{\cdot}N$ rather than N^2; the remaining dense columns go through the vector kernel. In the SARS-CoV-2 alignments 97–99% of informative columns are sparse, separating 6 to 11 genomes on average from the rest, which is why they gave the largest factors. In *M. tuberculosis* all 88,562 columns are informative, but 98% are sparse, with 3.4 minority isolates on average.

2.5 Multithreaded snp-dists writes corrupt matrices

snp-dists 1.2.0 prints each row field by field inside an OpenMP parallel loop (main.c lines 181–186 and 203–212), so with more than one thread the fields of different rows interleave. In five `-j 2` runs on a seeded test alignment, 91 to 114 of 120 rows were malformed, with fused labels such as `s0s60`. A 21-line patch that computes each row in parallel and prints it in order restored the `-j 1` output in 110 of 110 test runs. Three smaller problems turned up. A FASTQ record whose quality string is shorter than its sequence makes snp-dists stop reading and exit with status 0, printing a matrix of the earlier records only; a 5-line patch makes it fail. The help text gives the `-x` default as 99999, while the code uses 9,999,999; a 2-line patch corrects it. pairsnp 0.0.7 fails under numpy 2.5 because it calls `np.fromstring` in binary mode. SwiftSNP fixes the first bug and, as byte identity requires, reproduces the second and third. Supplementary Section S3 gives minimal reproductions and the patches.

3 Discussion

SwiftSNP's margin over snp-dists comes from removing work the answer does not need. Its margin over the fastest exact alternative, 47- to 1,892-fold, is smaller because pairsnp already exploits the sparsity of SNP columns, and it is smallest on *M. tuberculosis*, whose informative columns have smaller minorities than those of SARS-CoV-2.

Byte identity is a stronger promise than agreement on a test set, and it is the promise that lets a user switch without re-validation. Testing every change against snp-dists made it checkable while the code changed. It forced the fast parser to reproduce the quirks of kseq (Li n.d.), the parser snp-dists uses,

such as names cut at NUL bytes and a carriage return kept at the start of a sequence, and it forced an exact fallback behind every fast path. The protection held on data the tests had never seen: real alignments with N runs, gaps, IUPAC codes and gzip input, on a different CPU vendor, exposed no difference. The same approach carries over to any tool whose output can be compared byte for byte.

Limitations. The largest factors need AVX-512 BW; without it SwiftSNP was 42- to 53-fold faster than snp-dists at one thread, and ARM processors were not tested. All speed-ups are relative to snp-dists at one thread, its only setting with correct output, and snp-dists at two threads was 1.6- to 1.9-fold faster than that. The headline covers plain FASTA with default options; gzip input ran 44- to 69-fold faster than snp-dists and `-a` 31- to 59-fold. The two threads were all the CPUs of a shared cloud VM whose timings varied by ±15%, and an earlier SwiftSNP run on 8,596 genomes took 0.341 s instead of 0.232 s at two threads, which would make the two-thread mean 2,679-fold.

A pipeline that switches to SwiftSNP gets the same matrix file, so every threshold, cluster and tree built from it stays as it was. On an AVX-512 machine the 27-minute snp-dists run on 8,596 SARS-CoV-2 genomes takes a quarter of a second, and on an ordinary AVX2 laptop SwiftSNP was 21- to 107-fold faster than snp-dists at one thread on each of the five alignments.

4 Methods

4.1 Target

I took snp-dists 1.2.0 (commit 3cd3988) as the target. SwiftSNP accepts the options and input formats of snp-dists, writes standard output identical byte for byte to `snp-dists -q -j 1` at every thread count, and exits non-zero wherever snp-dists fails. The reference is `-j 1` because multithreaded snp-dists output is corrupt (Section 2.5). SwiftSNP vendors a patched libdeflate 1.26 for gzip input. I developed SwiftSNP with correctness-gated coding agents (Claude Opus 5.5, Anthropic), following SwiftNJ (Christensen 2026): every code change had to pass the correctness test in Section 4.2.

4.2 Correctness testing

Each test ran SwiftSNP and the live snp-dists binary on the same random input: 1–40 sequences of length 0–1,000 with lowercase letters, N, gaps, IUPAC codes, stray symbols, empty and duplicate names, line wrapping, CRLF line ends, FASTQ records, gzip or unequal lengths, random options (`-a -k -c -b -L -m -t -x`) and a random thread count from {1, 2, 3, 4, 7}. Standard output had to match byte for byte and the exit status had to agree on failure. The five synthetic benchmark sets and four holdout sets were checked against checksums of the snp-dists output.

4.3 Data

Five synthetic benchmark sets and four holdout sets, generated from fixed seeds, cover long alignments with few informative columns, SNP-only cores, large numbers of genomes, IUPAC codes, N runs, lowercase letters and gzip input (Supplementary Table S1). The five published alignments were used whole and unmodified: a *V. cholerae* O1 core-genome alignment made with Snippy (Seemann 2015; Wilkinson 2025), also compressed with gzip -6 as a sixth input; an *M. tuberculosis* alignment of 1,788 isolates at 88,562 variable sites (Mixão et al. 2023; isolates from Walker et al. 2015); and three SARS-CoV-2 sets from Nextstrain's open builds (Hadfield et al. 2018; aligned with Nextclade, Aksamentov et al. 2021), with N runs, gaps and IUPAC codes. I downloaded them after development had finished, and they were never on the machine on which the code was developed.

4.4 Timing

Headline timings come from a cloud VM with an Intel Xeon at 2.80 GHz (Cascade Lake class, 2 vCPUs, AVX-512 BW, gcc 13.3.0). snp-dists `-j 1` ran once per input; SwiftSNP ran three times at one and at two threads, with the output file deleted before each run, and the best time counts. Times are wall times

of the whole command, including parsing and writing the matrix to a file, and every speed-up is relative to `snp-dists -j 1` on the same machine. The same source was also built with `-march=haswell` (AVX2 code paths) on the VM, and built and run on a laptop (AMD Ryzen 5 5600H, Zen 3, 6 cores, AVX2, no AVX-512) at one and twelve threads while other jobs were running.

4.5 Baselines

The comparison tools were psdm 0.3.0 (Hall 2024), with default settings and with `-e` set to the file's non-ACGT characters so that it ignores what snp-dists ignores; pairsnp-cpp 0.3.1 (Tonkin-Hill 2022); pairsnp 0.0.7 (Python, single process, with numpy 1.26 because it fails under numpy 2.5; Tonkin-Hill 2019); and snp-dists `-j 2`, all at two threads on the same VM. Each matrix was compared cell by cell with `snp-dists -j 1` after matching rows by name. SwiftSNP's times in the comparison come from the real-data protocol, because the comparison tool's timer adds up to 50 ms to each run; with that timer the geometric-mean margin is 263-fold instead of 336-fold (Supplementary Section S2).

Data and code availability

The code is available at https://github.com/joe-b-christensen/swiftsnp and archived on Zenodo (DOI: 10.5281/zenodo.23503646). SwiftSNP is free software under the GNU General Public License v3.0, the licence of snp-dists from which it derives; it vendors a patched libdeflate 1.26 (MIT licence) and, like snp-dists, kseq.h (MIT licence). The repository contains the source code, its build script and the scripts that check the output against snp-dists and time both programs (prepare.py, evaluate.py, gen_data.py).

The synthetic datasets are regenerated deterministically by prepare.py. The real alignments are public:

- *V. cholerae* O1: Wilkinson E, Global Vibrio cholerae O1 dataset, Zenodo (2025), doi:10.5281/zenodo.16909942, file cholera_global_n100_2025.09.01.fasta; licence CC BY 4.0.
- *M. tuberculosis*: Mixão V et al., Multiple Sequence Alignment of a diverse dataset with 1788 Mycobacterium tuberculosis isolates, Zenodo (2023), doi:10.5281/zenodo.7772652, file MTb_original_align_profile.fasta; licence CC BY 4.0 (isolates from Walker et al. 2015; ENA project PRJNA282721).
- SARS-CoV-2: Nextstrain open builds, https://data.nextstrain.org/files/ncov/open/{global,europe,africa,asia,north-america,oceania,south-america}/aligned.fasta.xz, downloaded on 11 October 2026; built by Nextstrain from public NCBI GenBank and Robert Koch Institute data and released as open data. Because these builds are regenerated regularly, the exact sequence sets used here are archived with the Zenodo record (DOI: 10.5281/zenodo.23503646).

The repository's `data/fetch_real_data.sh` downloads all five sources.

Competing interests

None declared.

Funding

None.

References

Aksamentov I, Roemer C, Hodcroft EB, Neher RA (2021). Nextclade: clade assignment, mutation calling and quality control for viral genomes. Journal of Open Source Software 6(67):3773. doi:10.21105/joss.03773

Biggers E (2026). libdeflate: heavily optimized library for DEFLATE/zlib/gzip compression and decompression, version 1.26. https://github.com/ebiggers/libdeflate

Christensen J (2026). SwiftNJ: Fast Exact Neighbour Joining via Correctness-Gated Coding Agents. bioRxiv preprint. doi:10.64898/2026.05.28.728410

Hadfield J, Megill C, Bell SM, Huddleston J, Potter B, Callender C, Sagulenko P, Bedford T, Neher RA (2018). Nextstrain: real-time tracking of pathogen evolution. Bioinformatics 34(23):4121–4123. doi:10.1093/bioinformatics/bty407

Hall MB (2024). psdm: compute a pairwise SNP distance matrix from one or two alignment(s), version 0.3.0. https://github.com/mbhall88/psdm; Zenodo doi:10.5281/zenodo.5706784

Li H (n.d.). kseq.h, in klib: a generic library in C. https://github.com/attractivechaos/klib

Mixão V, Pinto M, Sobral D, Di Pasquale A, Gomes JP, Borges V (2023). Multiple Sequence Alignment of a diverse dataset with 1788 Mycobacterium tuberculosis isolates. Zenodo. doi:10.5281/zenodo.7772652

Seemann T (2015). Snippy: rapid haploid variant calling and core genome alignment. https://github.com/tseemann/snippy

Seemann T, Klötzl F, Page AJ (2025). snp-dists: pairwise SNP distance matrix from a FASTA sequence alignment, version 1.2.0. https://github.com/tseemann/snp-dists

Tonkin-Hill G (2019). pairsnp: a simple package for calculating pairwise SNP distances, Python package version 0.0.7. https://github.com/gtonkinhill/pairsnp

Tonkin-Hill G (2022). pairsnp-cpp: fast pairwise SNP distance matrices from multiple sequence alignments, version 0.3.1. https://github.com/gtonkinhill/pairsnp-cpp

Walker TM, Ip CLC, Harrell RH, et al. (2013). Whole-genome sequencing to delineate Mycobacterium tuberculosis outbreaks: a retrospective observational study. Lancet Infectious Diseases 13(2):137–146. doi:10.1016/S1473-3099(12)70277-3

Walker TM, Kohl TA, Omar SV, et al. (2015). Whole-genome sequencing for prediction of Mycobacterium tuberculosis drug susceptibility and resistance: a retrospective cohort study. Lancet Infectious Diseases 15(10):1193–1202. doi:10.1016/S1473-3099(15)00062-6

Wilkinson E (2025). Global Vibrio cholerae O1 dataset. Zenodo. doi:10.5281/zenodo.16909942