Downloaded from genome.cshlp.org on November 10, 2026 . Published by Cold Spring Harbor Laboratory Press



Ultrafast alignment-free detection of viral genome variation with bronko

Ryan D Doughty, Michael J. Tisza and Todd J. Treangen

Genome Res. published online November 10, 2026
Access the most recent version at doi:10.1101/gr.282104.126

P<P Published online November 10, 2026 in advance of the print journal.

Accepted Manuscript Peer-reviewed and accepted for publication but not copyedited or typeset; accepted manuscript is likely to differ from the final, published version.

Open Access Freely available online through the *Genome Research* Open Access option.

Creative Commons License This manuscript is Open Access.This article, published in *Genome Research*, is available under a Creative Commons License (Attribution-NonCommercial 4.0 International license), as described at http://creativecommons.org/licenses/by-nc/4.0/.

Email Alerting Service Receive free email alerts when new articles cite this article - sign up in the box at the top right corner of the article or **click here.**



To subscribe to *Genome Research* go to:
https://genome.cshlp.org/subscriptions

Published by Cold Spring Harbor Laboratory Press

Downloaded from genome.cshlp.org on November 10, 2026 . Published by Cold Spring Harbor Laboratory Press

Ultrafast alignment-free detection of viral genome variation with bronko

Ryan D. Doughty[1 [0009−0001−8355−0926]], Michael J. Tisza[2 [0000−0003−1168−1617]], and Todd J. Treangen[1,3,4 [0000−0002−3760−564X]]

[1] Department of Computer Science, Rice University, Houston TX 77005, USA

[2] Department of Molecular Virology and Microbiology, Baylor College of Medicine, Houston TX 77030, USA

[3] Department of Bioengineering, Rice University, Houston TX 77005

[4] Ken Kennedy Institute, Rice University, Houston TX 77005, USA

Abstract As viral sequencing datasets continue to grow, traditional alignment-based variant-calling pipelines become computationally prohibitive. To address these challenges, we present bronko, an ultrafast alignment-free framework for detecting viral variation directly from sequencing data. The novel computational approach implemented in bronko enables scaling to massive viral sequencing datasets and comprises four key components: i) a locality-sensitive bucketing function to rapidly identify single-nucleotide polymorphisms (SNPs) relative to reference(s), ii) a direct k-mer count pseudo-mapping approach that approximates a pileup without alignment, iii) a greedy de novo assembly algorithm to reconstruct mutation-dense regions and indels, and iv) a streaming-based sliding-window outlier test to estimate baseline noise across the genome and precisely differentiate real minor variants from noise. Together, these components yield near-linear computational complexity with respect to sequencing depth, enabling bronko to process thousands of viral samples rapidly on modest hardware. Our results are threefold: 1) On simulated amplicon sequencing data, bronko recovers variants with higher precision and comparable recall to existing tools while running up to one to three orders of magnitude faster; 2) bronko generates sequence alignments directly from sequencing data, with SNP content nearly identical to that of whole-genome alignment while requiring only a fraction of the computational time; and 3) application of bronko to longitudinal sequencing data from chronically infected SARS-CoV-2 patients reveals consistent patterns of intrahost diversification and adaptive mutations over time. Altogether, these results demonstrate bronko's potential as a scalable tool for large-scale viral genomic analyses, overcoming longstanding computational barriers to the intrahost and interhost characterization of viral variation.

Keywords: Viral Genomics · K-mers · Alignment-Free

Downloaded from genome.cshlp.org on November 10, 2026 . Published by Cold Spring Harbor Laboratory Press

Introduction

Viral sequencing is now central to modern epidemiology, providing unprecedented insight into viral evolution, transmission, and adaptation (Jansz et al. 2024). By characterizing viral genetic variation across populations, sequencing enables detection of emerging variants of concern, inference of transmission chains, and assessment of selective pressures (Houldcroft et al. 2017). These capabilities proved essential during the COVID-19 pandemic, where global monitoring of SARS-CoV-2 variants directly guided vaccine updates, public health interventions, and our understanding of viral dynamics (Yi et al. 2021). Beyond major circulating variants, patient-level sequencing data also encodes information on intrahost variants (iSNVs), which can provide additional insight into viral evolution (Lauring 2020). For instance, iSNVs have been used to reveal early mutational events that can later fix and propagate through populations as well as identify individual transmission events (Xue et al. 2018; Sapoval et al. 2023; Gold et al. 2025). As sequencing continues to become faster and cheaper, the opportunity to continuously monitor viral evolution worldwide in real-time has never been greater.

Standard variant calling pipelines depend on read alignment, followed by identification of both consensus (major) and intrahost (minor) variants. Even with recent optimized algorithms and software (Edgar 2020; Ekim et al. 2023; Langmead et al. 2012; Li et al. 2009a), both read alignment and variant calling remain computationally expensive affairs in genomics workflows (Berger et al. 2023; Shajii et al. 2016; Xiang et al. 2023). Given the NCBI Sequence Read Archive already hosts over 7 million SARS-CoV-2 sequencing datasets alone ($\backsim$500 terabytes of data), the current alignment-based paradigm is impractical for population-scale studies of viral variation. This motivates the development of methods that can analyze viral sequencing data more efficiently with similar sensitivity and precision.

Over the years, several alignment-free variant calling methods have been developed for human and bacterial genomics. Rather than aligning reads and calling variants, these approaches detect variation directly from reads, either within or across samples, typically trading a modest loss in sensitivity for substantial runtime gains. Many of these tools focus on discovering novel variation that alignment-based methods may miss, using graph-based data structures to capture local sequence differences (Iqbal et al. 2012; Fang et al. 2016; Peterlongo et al. 2017; Chong et al. 2017; Khorsand et al. 2021; Denti et al. 2023). Other methods take explicit k-mer–centric approaches to detect smaller variants. Kestrel (Audano et al. 2018) detects small variants by comparing k-mer abundance profiles between a sample and a reference, enabling rapid inference of sequence differences from coverage imbalances. LAVA (Shajii et al. 2016) clusters k-mers with small edit distances to detect known SNP targets directly. Similarly, kSNP (Gardner et al. 2015; Gardner et al. 2013) and SKA2 (Derelle et al. 2024; Derelle et al. 2025) both rely on split k-mers centered on single variable positions, allowing identification of a mutation when the context around the variant remains unchanged.

Downloaded from genome.cshlp.org on November 10, 2026 . Published by Cold Spring Harbor Laboratory Press

While these alignment-free methods have shown promise in human and bacterial genomics, viral genomes present additional unique challenges. First, viral sequencing datasets can often reach ultra-high depths (>10,000x), amplifying computational inefficiencies in both read alignment and downstream variant calling (Berger et al. 2023). More critically, however, most existing alignment-free methods focus on consensus-level variation and overlook low-frequency iSNVs that are present in most viral sequencing datasets. For viral samples, where some true minor allele frequencies may be below 1%, distinguishing genuine biological signal from sequencing noise and other artifacts requires explicit modeling of error distributions and coverage heterogeneity, particularly in an alignment-free framework where positional context can be limited (McCrone et al. 2016). Although numerous low frequency variant calling approaches have been developed to handle this task, none have been employed in an alignment-free context (Wilm et al. 2012; Xu et al. 2019; Kockan et al. 2017). Fortunately, viral genome analysis provides numerous opportunities for computational optimization as well. First, their compact genome size enables efficient in-memory representations and exhaustive k-mer indexing approaches. Similarly, despite their rapid mutation rate, the mutational spectrum of most RNA and DNA viruses is dominated by single-nucleotide substitutions, with relatively few structural variants or long insertions or deletions (Sanjuán et al. 2010). These simplicities make viral genomes particularly well-suited to k-mer –centric approaches that can capture local nucleotide changes directly from sequencing reads.

Here we introduce bronko, a scalable, alignment-free framework to detect both major and minor single nucleotide variants directly from viral sequencing data. bronko uses a locality-sensitive bucketing (LSB) function to rapidly identify k-mers within one edit distance of each other (Chen et al. 2023). The use of this function enables bronko to perform rapid pseudo-mapping of k-mer counts from viral sequencing data directly to an approximate pileup of each genomic coordinate for one or more reference genomes, bypassing read alignment and SAM manipulation entirely. bronko then employs a streaming outlier-based statistical model to distinguish genuine minor variants from sequencing noise, adapting across the genome to local error, mutational profiles, and coverage fluctuations. bronko also optionally reconstructs genomic regions containing indels or many dense mutations, enabling improved detection and quantification of nearby variants that would otherwise be obscured by k-mer mapping dropouts. Altogether, these design choices enable bronko to run ultra-efficiently and scale near linearly with sequencing depth, making it well-suited for large-scale viral genomic surveillance, longitudinal infection studies, and rapid screening of emerging variants in environmental or clinical contexts.

Methods

Our method bronko performs two primary tasks: (1) rapid pseudo-mapping of k-mer counts to reference genomes of interest, and (2) precise detection of major single-nucleotide variants (SNVs) and intra-host

Downloaded from genome.cshlp.org on November 10, 2026 . Published by Cold Spring Harbor Laboratory Press

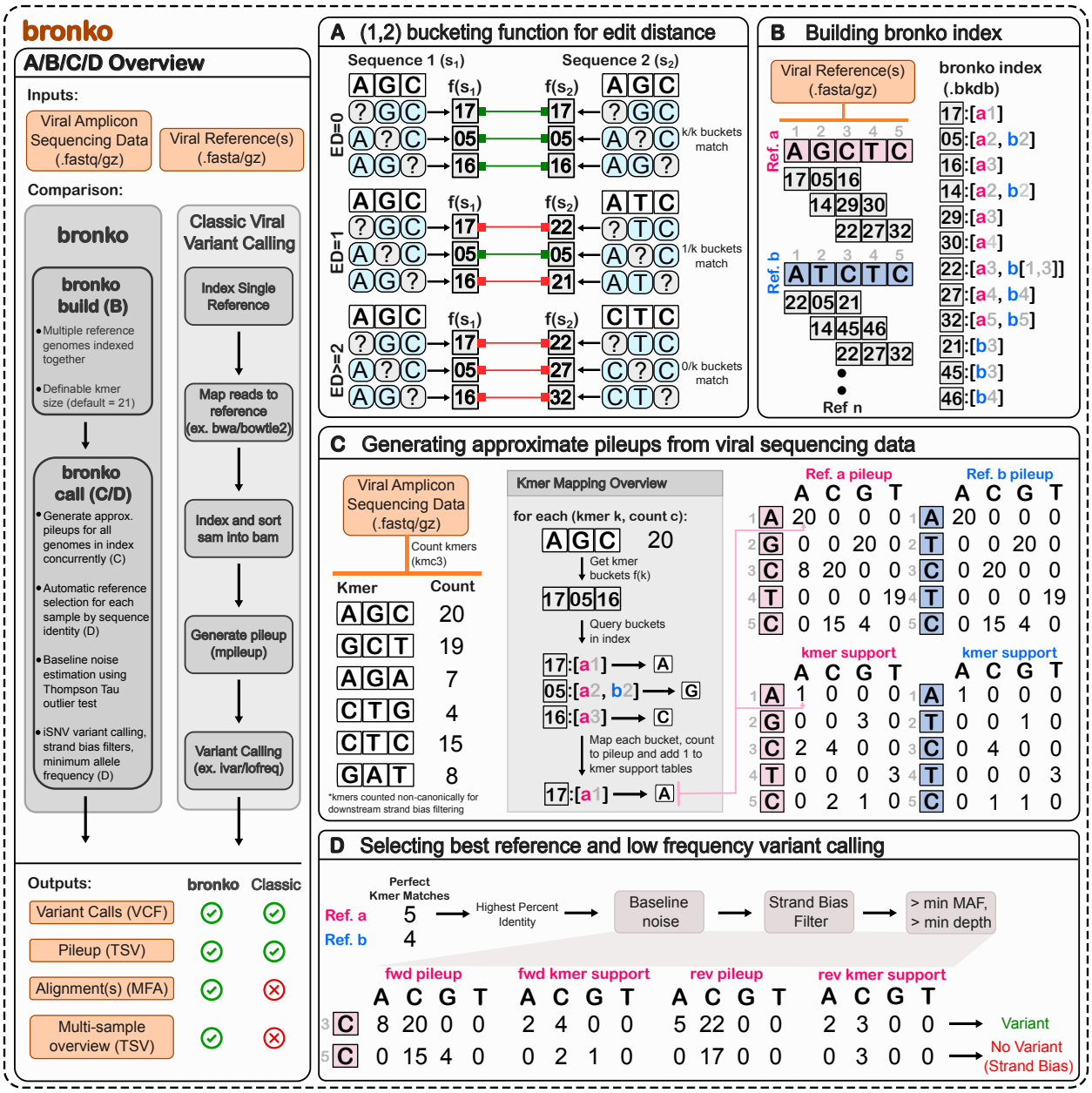


Fig. 1. Overview of bronko. **A**. (1,2) LSB function f for the edit distance. Three different edit distances are shown. White boxes represent k-mers, blue and gray ovals represent variants, and gray boxes represent the bucket values that are returned from f. **B**. Building a bronko database from a set of references. Every k-mer in each reference is transformed into buckets, before being placed in a hash table with the reference(s) and position(s). **C**. Generating approximate pileups and k-mer support tables from k-mer counts. Buckets are retrieved for each k-mer before being queried in the index. These positions are then mapped into the pileup and k-mer support. **D** Reference selection and variant calling. The reference with the highest identity to each sample is selected before baseline noise, strand bias, and minimum AF/depth filters to identify SNPs and iSNVs.

SNVs (iSNVs) from the pseudo-mapping output. Pseudo-mapping is comprised of three components: (i) a $(1, 2)$-sensitive LSB function that groups near-identical k-mers under fixed edit-distance constraints, enabling single-base mismatch detection (**Fig. 1A**); (ii) a bucket-based index that stores positional information of

Downloaded from genome.cshlp.org on November 10, 2026 . Published by Cold Spring Harbor Laboratory Press

all potential variation across references (**Fig. 1B**); and (iii) a single-pass pileup construction algorithm that converts observed k-mer counts into per-base coverages without explicit read alignment (**Fig. 1C**). For users interested in recovering variant dense regions or indels, we also provide an option to reconstruct regions that are not mapped in the initial pileup construction, and follow this with a secondary-pass pileup construction on the new consensus sequence (**Supplementary Fig S1**). Variant detection operates on the constructed pileups and also has three main stages: (i) reference selection by sequence identity; (ii) estimation of a baseline sequencing-error model using an outlier-based statistic computed in a sliding window across the genome; and (iii) variant calling that combines the baseline noise model, strand-bias metrics, and k-mer support at each position to report SNPs/iSNVs. When provided multiple samples, bronko also has the additional ability to generate a multiple sequence alignment from the union of positions with SNPs, which can then be used for large-scale inference of phylogenetic trees.

(1,2) locality-sensitive bucketing functions for edit distance

To sensitively detect single-nucleotide variation (SNVs) within viral sequencing data, we utilize a (1,2) LSB function, denoted $f(s)$, following the formulation described in (Chen et al. 2023). In contrast to traditional k-mer matching approaches that commonly rely on exact matches, the LSB approach allows us to efficiently group near-identical k-mers into common "buckets", while maintaining fixed edit-distance constraints. In bronko, we utilize a $(1, 2)$-sensitive bucketing scheme, which guarantees that any pair of k-mers with edit distance ≤ 1 map to at least one common bucket, while any pair differing by two or more edits (≥ 2) are assigned to disjoint bucket sets. Formally, for each k-mer s of length k, the function $f(s)$ computes an ordered set of k integer-valued bucket identifiers $B_s = (b_0, b_1, \ldots, b_{k-1})$ where each $b_i \in [0, k \times 4^{k-1}]$. Intuitively, each bucket b_i is a unique identifier that represents if position i in the k-mer was a variant position, and the rest of the k-mer remained the same (see **Fig. 1A**). Thus, this formulation allows for identification of not only exact matches between k-mers, but also single nucleotide variation.

For two k-mers k_1 and k_2 with corresponding bucket sets B_1 and B_2, the size of intersection $B_1 \cap B_2$ points directly to the edit distance of the two sequences. Additionally, in the case of a single mismatch (ED $= 1$), the matching bucket provides the precise position of the differing base. Altogether, the following 3 cases can be defined for two k-mers, where $ED(k_1, k_2)$ represents the edit distance between the two k-mers:

$$\begin{cases} ED(k_1, k_2) = 0 \implies |B_1 \cap B_2| = |B_1| = |B_2| = k, \\ ED(k_1, k_2) = 1 \implies |B_1 \cap B_2| = 1, \ \exists\, i \text{ s.t. } B_1[i] = B_2[i] \text{ and } k_1[i] \neq k_2[i], \\ ED(k_1, k_2) \geq 2 \implies |B_1 \cap B_2| = 0. \end{cases}$$

Downloaded from genome.cshlp.org on November 10, 2026 . Published by Cold Spring Harbor Laboratory Press

This framework enables identification of single nucleotide variation directly from k-mers in $O(k)$ time, which altogether allows bronko to rapidly identify most single-nucleotide variants directly from k-mers rather than aligning reads.

By default, we adopt a k-mer length of $k = 21$, although this parameter is tunable. Empirically, we found $k = 21$ to provide the best balance between reducing intra-genomic and inter-genomic collisions while also retaining adequate resolution for variant detection. This choice imposes a strict sensitivity limit in that the central variant(s) will be missed in k-length regions with $\geq$3 variants. To ensure intra-genomic collisions do not affect downstream variant calling, we exclude any buckets that occur more than once within a given reference genome, ensuring that each retained bucket corresponds to a unique sequence location. Because we can use all k-mers overlapping a position to help identify variants, discarding duplicate k-mers has a negligible impact as long as the k-mer size is large enough such that the intra-genomic repeats are rare.

Indexing viral reference sequences

In order to enable rapid k-mer pseudo-mapping to a reference efficiently, we begin by building an index on the buckets of a set of references (see **Fig. 1B**). Formally, given a set of n viral reference sequences $G = \{G_1, G_2, ..., G_n\}$, each k-mer in every reference is first transformed into its canonical form before being mapped to k buckets using the bucketing function f. As each bucket corresponds to a specific nucleotide position within the k-mer, we store the buckets as keys and their associated reference identifiers and positions as values. In practice, we also record whether the k-mer was converted to its reverse complement during canonicalization, allowing us to recover the original strand orientation after matching. While conceptually simple, this indexing scheme enables ultra-rapid identification of most single-nucleotide variants relative to a larger reference set, a capability not included in most existing variant calling pipelines.

Direct *k*-mer pseudo-mapping and approximate pileup construction

Given an index over a collection of viral reference genomes G as described above, we directly map observed k-mer counts from sequencing reads to per-base pileups for all references simultaneously (see **Fig. 1C**). We label this process k-mer pseudo-mapping. Rather than performing conventional read alignment followed by a separate variant-calling stage, pseudo-mapping constructs pileups in a single pass by mapping k-mer counts using indexed bucket positions. This eliminates the need for full read alignment while retaining single-nucleotide resolution in most cases. Because deep viral sequencing datasets are often highly redundant, this approach compresses the relevant variation into a single call, avoiding depth-dependent pileups that scale poorly with coverage. Intuitively, mapping k-mers rather than full reads provides a rapid, more linear-scaling form of compression.

Downloaded from genome.cshlp.org on November 10, 2026 . Published by Cold Spring Harbor Laboratory Press

For each reference genome $g \in G$ of length ℓ_g, we allocate tables to store estimated base coverage and the number of unique k-mers mapping to each position-nucleotide pair (henceforth referred to as k-mer support). We do this for both the forward and reverse strands, resulting in 4 tables. Specifically, we define:

$$P_g^{(f)}, P_g^{(r)} \in \mathbb{N}^{4\times\ell_g}, \quad U_g^{(f)}, U_g^{(r)} \in \mathbb{N}^{4\times\ell_g},$$

where $P_g^{(f)}$ and $P_g^{(r)}$ represent the cumulative pileup counts for the forward (f) and reverse (r) orientations, respectively, and $U_g^{(f)}, U_g^{(r)}$ is the k-mer support. All entries are initialized to zero:

$$P_g^{(f)}[i,j] = P_g^{(r)}[i,j] = U_g^{(f)}[i,j] = U_g^{(r)}[i,j] = 0,$$

for $i \in \{A, C, G, T\}$ and $1 \leq j \leq \ell_g$.

In order to map sequencing data to each table, we first enumerate and count all observed k-mers non-canonically from the viral sequencing data using kmc3 (Kokot et al. 2017), yielding a set $K = \{(k_i, c_i)\}_{i=1}^{m}$ where k_i denotes a unique k-mer and c_i its observed count. We start by discarding k-mers with counts below a small threshold $n_{\min}$ (default $n_{\min} = 3$). Because in most sequencing datasets, a large percentage of k-mers are rare, this filtering step effectively helps to remove potentially spurious k-mers that could otherwise contribute false minor alleles while also speeding up the process by reducing the number of k-mers that need to be mapped.

For each retained k-mer k with count c_k, we compute its canonical form k_c and subsequently apply the bucketing function $f(k_c)$ to obtain the ordered set $(b_0, b_1, \ldots, b_{k-1})$, where each b_i corresponds to the bucket representing a variant position at position i of the k-mer. Each bucket b_i is then queried against the index to identify a potential match to the reference at location j, along with the strandedness of the match ρ. For each such match, we identify the corresponding nucleotide $k[i]$ in the k-mer and update the appropriate pileup and k-mer support tables:

$$P_g^{(\rho)}[k[i], j] = \max\left(c_k, P_g^{(\rho)}[k[i], j]\right) \qquad U_g^{(\rho)}[k[i], j] \mathrel{+}= 1.$$

Selecting the best reference for a sample

In the case where the index G contains more than a single reference, we automatically select the best reference after pseudo-mapping to each one individually. Given that viral genomes are relatively small and most of the genomes encode proteins, we use a simple heuristic that estimates the percent identity across the full genome of a sample to each reference $g \in G$. To do this, we calculate the following:

Downloaded from genome.cshlp.org on November 10, 2026 . Published by Cold Spring Harbor Laboratory Press

$$I_g = \frac{p_g}{2 * l_g}$$

where p_g is the number of k-mers that were mapped perfectly to reference g and l_g is the length of the genome. The number of perfect k-mers can be efficiently calculated during the mapping process by counting the k-mers in which all buckets have consecutive positions in the reference. The total is divided by 2 to account for k-mers mapping to both the forward and reverse strands, given k-mers are counted non-canonically. This process ensures that we select a reference that is closest to the actual sample, reducing the number of major variants that could disrupt minor variant calling in the sample.

Indel reconstruction, consensus generation, and two-pass pseudo-mapping

Following reference selection, bronko optionally performs local assembly to enable detection of indels or other variant dense regions, which then improves the identification and quantification of neighboring SNPs and iSNVs (see **Supplementary Fig. S1**). Due to bronko's use of a $(1, 2)$ LSB function, k-mers spanning indels or multiple clustered SNPs will not map to the reference, resulting in local reductions in the observed sequencing depth in these regions (generally from near the local sequencing depth to near zero, see **Supplementary Fig. S1A**).

bronko uses this feature of the pileups to identify and reconstruct gaps in the consensus sequences de novo. bronko first identifies positions at which the depth exhibits a substantial decrease or increase between neighboring nucleotides. A depth-drop position i and a subsequent depth-rise position j are considered candidate reconstruction boundaries when their separation satisfies a predefined distance threshold d, i.e $|i - j| \leq d$. Nearby drop-rise pairs are then grouped together to define candidate reconstruction intervals (**Supplementary Fig. S1B**).

For each candidate interval, bronko then identifies seed k-mers immediately outside the reconstruction boundaries. The initial seed positions are placed two nucleotides upstream and downstream of the respective boundaries. A candidate seed k-mer is accepted when the depth across the k-mer is consistent, indicating that the seed lies outside the region affected by the mapping dropout. If the depth is inconsistent, the seed position is iteratively shifted away from the reconstruction boundary until a k-mer that satisfies the depth-consistency criterion is identified or a maximum search distance is reached. When multiple candidate indels occur within a sufficiently small genomic distance such that no nearby seed k-mers can be set, adjacent intervals may be merged to produce a single reconstruction interval spanning multiple variants.

Given a pair of seed k-mers, bronko attempts to reconstruct the sequence between them de novo. The reconstruction follows a greedy traversal over candidate k-mers, analogous to a walk through a de Bruijn graph. To enable this, bronko stores the first and last buckets of each k-mer in the initial pseudo-mapping

Downloaded from genome.cshlp.org on November 10, 2026 . Published by Cold Spring Harbor Laboratory Press

pass, along with the nucleotide it represents and the depth of that k-mer (see **Supplementary Fig. S1C**). When reconstructing alleles, bronko retrieves the corresponding bucket information from the stored k-mer hash table, and the observed depth is used to evaluate the candidate extension. The nucleotide producing the highest-depth successor is selected, as long as the depth is within 50% (above or below) the depth of the current k-mer (see **Supplementary Fig. S1D**).

The selected nucleotide is appended to the current sequence, producing the next k-mer, and the process is repeated iteratively. Reconstruction terminates when the right seed k-mer is reached, or when a predefined maximum reconstruction length is exceeded, in which case the reconstruction is considered unsuccessful. For successful reconstructions, the resulting sequence between the two seed k-mers is treated as the consensus allele for the reconstructed region. bronko then incorporates the reconstructed sequence into the reference and constructs a new k-mer index. The k-mer mapping procedure is subsequently repeated against this reconstructed reference. By using a consensus representing all complex variation in the sample, this second mapping pass enables k-mers to map that previously were not able to, resulting in improved depth estimates for neighboring SNPs and iSNVs.

Low frequency variant calling

Once a reference genome is selected or final consensus sequence is reconstructed, variant calling can be performed on the pileups for that reference. There have been many methods over the years that use different statistical methods in order to differentiate real mutational signal from sequencing error and other artifacts. We utilize a streaming version of the Thompson–Tau outlier test, adapted from the approach implemented in Outlyzer (Muller et al. 2016). We selected this method due to its strong performance in external benchmarks (Xiang et al. 2023), showing (i) the highest recall at higher minor allele frequencies ($>1\%$), (ii) high precision at lower minor allele frequencies ($<1\%$), and (iii) good computational efficiency and robustness to changes in sequencing depth. Our implementation extends this approach to a streaming context, allowing continuous evaluation of local noise distributions without recomputing statistics for each sliding window across the genome (see **Fig. 2**).

Within a sliding window of fixed length w, the algorithm maintains running estimates of the number of non-major alleles (n_w) with minor allele frequency (MAF) greater than 0, the sum of MAFs (s_w), and the sum of squared MAFs (s_w^2). Because there can be at most three minor alleles at each genomic position, all MAFs within the window are stored in an array of length $3w$. The three possible minor alleles for a given reference position p are indexed as $i = (p \bmod w) \times 3$.

Downloaded from genome.cshlp.org on November 10, 2026 . Published by Cold Spring Harbor Laboratory Press

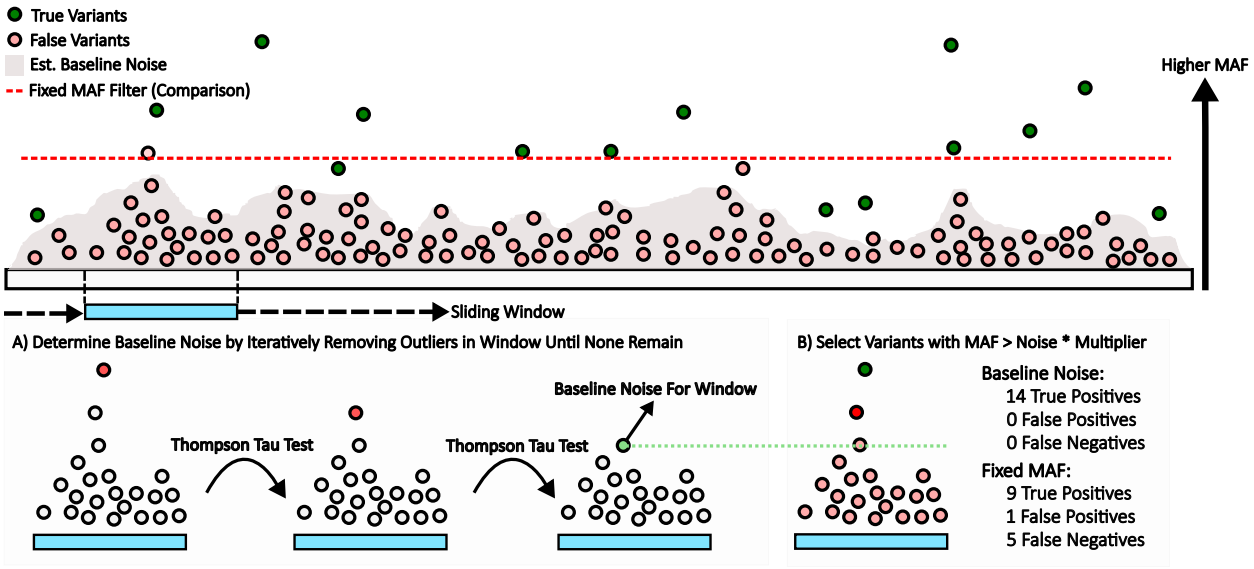


Fig. 2. Baseline noise filter for low-frequency variants. Real variants are shown in green, errors in pink. The gray shadow represents the estimated baseline noise, and the red dashed line shows the fixed MAF threshold for comparison. **A.** Sliding window outliers are iteratively removed using a modified Thompson Tau test to estimate local noise **B.** Variants with MAF above the window's baseline (scaled by a multiplier) are kept, others discarded

After updating the minor allele frequencies at position p, as well as n_w, s_w, and s_w^2, the running mean (μ_w) and variance (σ_w^2) are computed using Welford's approach for updating variance (Welford 1962; Efanov et al. 2021):

$$\mu_w = \frac{s_w}{n_w}, \qquad \sigma_w^2 = \frac{s_w^2}{n_w} - \mu_w^2.$$

To efficiently remove outliers, two auxiliary data structures are maintained. First, a short, sorted list of length $\ell \ll w$ (default 50) containing the top-ℓ MAFs in the current window, and second, a binary indicator array of length $3w$ marking whether each MAF belongs to the top-ℓ list. When a new minor allele enters the window, it is compared to the smallest element in the top-ℓ list. If larger, it is inserted in sorted order and flagged in the indicator array. Otherwise, it is ignored. When an element exits the window, the indicator is checked, and if present in the top-ℓ list, it is removed and the list is adjusted. This structure allows rapid identification of potential outliers without recomputing the max from the entire window.

For each window, baseline sequencing noise is determined by iteratively removing outliers using a modified Thompson–Tau test until no outliers remain, updating the window statistics after each removal. The maximum MAF in the window x_i is removed if $\delta_i > \tau * \sigma_w$, where $\delta_i = |x_i - \mu_w|$ and the dynamic threshold τ is derived from the Student's t-distribution with $n_w - 2$ degrees of freedom:

Downloaded from genome.cshlp.org on November 10, 2026 . Published by Cold Spring Harbor Laboratory Press

$$\tau = \frac{t_{\alpha,n_w-2}(n_w-1)}{\sqrt{n_w}\sqrt{n_w-2+t^2_{\alpha,n_w-2}}}.$$

Here, α is the desired false-positive rate (we utilize 10^{-3}). Because we test for upper-tail outliers only, the one-tailed form of $t_{\alpha,N-2}$ is used. After each removal, the statistics n_w, s_w, s_w^2, μ_w, σ_w^2, and τ are updated, and the next largest MAF is tested. This process continues until the maximum MAF value within the window does not reach the criteria to be considered an outlier.

Once the baseline noise profile is estimated across the genome, candidate variants are retained only if their MAF exceeded the baseline at their position by a specified multiplier. While outlyzer applies a uniform 2x multiplier threshold (Muller et al. 2016), we found that using a lower default multiplier of 1.5× and gradually increasing it up to 2× for low-frequency variants (MAF $<$ 1–2%) improved both precision and recall. Formally, the adaptive multiplier for a variant with MAF v is defined as $m(v) = b_0 + p_0 a^v$, where $a = 0.03$, $b_0 = 1.5$, and $p_0 = 0.5$. This function yields a maximum multiplier of 2× for MAFs approaching 0 and converges toward 1.5× for variants near MAF $>$ 1.5%.

Additional filters for minor variants In addition to our baseline noise filter, we also employ a strand filter to remove variants with significant strand bias. We utilize the strand odds ratio (SOR) filter from GATK, with a default threshold of ≤ 3. The GATK strand odds filter was chosen over other strand frequency filters due to its performance on higher-depth sequencing data (Guo et al. 2012). By default, we also calculate and utilize an adjusted SOR, which discounts the strand bias for variants with stronger evidence of being above the baseline noise. Specifically, the adjustment increases for variants with a larger difference between their MAF and the baseline noise at that position, as well as with the number of alternative-supporting reads. We formulate the adjustment D as:

$$D = \min\left(d \log_{10}\left(\max(N_{\text{alt}}, 1)\right) \log_{10}\left(\frac{\text{MAF}}{\text{MAF}_{\text{noise}}}\right), D_{\max}\right),$$

where d is an adjustment multiplier (default 3), N_{alt} is the number of alternative-supporting reads, MAF is the observed minor allele frequency, $\text{MAF}_{\text{noise}}$ is the baseline noise for that position, and $D_{\max}$ is set to the initial SOR threshold (default 3). This limits the maximum adjustment such that variants with an initial SOR greater than twice the threshold cannot be adjusted below the threshold. D is subtracted from the GATK strand odds ratio to obtain the final adjusted SOR.

We also report and utilize metrics of strand balance, which measures the percentage a particular variant appears on one strand in comparison to the other. We report strand balance overall (forward count for

Downloaded from genome.cshlp.org on November 10, 2026 . Published by Cold Spring Harbor Laboratory Press

reference + alternative allele compared to the reverse count for the reference and alternative allele) as well as strand balance for the reference and alternative separately.

Lastly, we utilize the number of k-mers supporting any given variant from the tables $U_g^{(\rho)}$. The motivation behind this filter was to remove variants that can accumulate from artifacts near the ends of amplicon sequences. Because we cannot easily utilize quality information directly from sequencing data, which often would remove variants of this type, we utilize the robustness of multiple k-mers supporting a single position to provide a secondary proxy. By default we require both strands to be present in 2 or more k-mers, although these parameters are configurable.

Multiple Sequence Alignment, Software Outputs and Configuration

When multiple sequencing samples are provided, bronko can generate an alignment for each set of sequences aligned to the same reference. To do so, we perform variant calling as described above on each sample independently. If querying against a larger reference set, any genome with > 3 samples mapped to it is deemed suitable for alignment. bronko then takes the union of SNV positions (positions with variants ≥ 0.5 MAF) across all samples for each reference. The consensus variant is taken for each sample at all of these positions, which are then used to build a multiple sequence alignment for that reference.

Mapping statistics For each sample, we also generate some overview statistics that can assist with quality control in downstream pipelines. The first are the breadth and depth of coverage of each sample. Breadth of coverage is defined as the percentage of bases in the selected reference with a total depth > 0. This is normalized by the genome length to get a final value from 0 to 1. The depth of coverage reported is simply the average depth across all of those non-zero positions. We also report the number of k-mers that were mapped as "perfect" (edit distance = 0) or "variant" (edit distance = 1) as well as unmapped (edit distance >= 2). These can help provide insight into the quality of the sequencing dataset, although we have found that the number of variant and unmapped k-mers scales well with sequencing depth in practice.

Software development and parameters bronko is implemented in Rust and available publicly on bioconda. All experiments in this paper were run using bronko v0.1.7 on an AMD EPYC 7742 64-Core Processor using 10 threads unless otherwise noted.

Results

Low frequency variant calling on simulated amplicon sequencing data

We first assessed bronko's performance in detecting iSNVs using simulated viral amplicon sequencing data with a defined ground truth across two complementary simulation protocols. First, we simulated 100 tiled-amplicon sequencing datasets of RSV-A spanning varying levels of intrahost diversity (see Supplementary

Downloaded from genome.cshlp.org on November 10, 2026 . Published by Cold Spring Harbor Laboratory Press

Methods for full details). Briefly, we downloaded 20 real RSV-A genomes from a single outbreak, aligned them using Parsnp2 (Kille et al. 2024), and constructed a maximum-likelihood phylogenetic tree using RAxML-NG (Kozlov et al. 2019). We then used PhastSim (De Maio et al. 2022) to simulate 20 genomes along this tree, using the inferred transition probabilities and nucleotide frequencies and a mutation rate selected to produce approximately 5–15 mutations per genome relative to the ancestral sequence. These genomes, along with the ARTIC RSV-A primers and specified proportions of each genome, were used to simulate tiled-amplicon sequencing data with Bygul (`https://github.com/andersen-lab/Bygul`). This approach allowed us to independently control the number of genomes and the frequency of intrahost mutations while retaining the structure of a tiled-amplicon sequencing experiment. We also repeated this experiment with Ebolavirus (see Supplementary Materials). However, this simulation framework did not model several common sources of error in amplicon sequencing, including PCR artifacts and amplification biases.

To evaluate variant calling under more complex sequencing conditions and challenging sources of technical variation, we additionally generated 81 simulated Human Papillomavirus 16 (HPV16) datasets using GENOMICON-Seq (Stosic et al. 2025), comprising 810 total amplicon sequencing samples (10 samples per dataset). These datasets spanned combinations of cycle threshold values, input genome copy numbers, viral mutation rates, and sequencing depths, enabling a comprehensive evaluation of variant calling performance across a range of sample qualities and sequencing depths. Full experimental parameters are provided in the Supplementary Materials and Supplementary Data.

We benchmarked bronko against two widely adopted viral variant calling tools: LoFreq (Wilm et al. 2012), and iVar (Grubaugh et al. 2019). LoFreq uses base-quality–aware statistical modeling for low-frequency variants, while iVar applies minimal filtering and primarily reports variants above a user-defined MAF threshold. Performance of each tool was evaluated for precision, recall, and F1 score, considering single-nucleotide variants only.

In **Fig. 3**, we show the results of the RSV-A (**Fig. 3A-D**) and HPV16 (**Fig. 3E-H**) experiments. As shown in **Fig. 3A** and **Fig. 3E**, bronko consistently demonstrated high precision across increasing MAF thresholds in both simulations. In the RSV-A experiments, bronko achieved near-perfect precision even at MAF thresholds as low as 0.1%, with precision increasing to essentially 100% above 1%. Both LoFreq and iVar showed comparable performance at high frequencies, but exhibited reduced precision at lower MAF thresholds. Recall in the RSV-A experiments was high for all three variant callers at higher MAF thresholds, although performance decreased at lower thresholds, with bronko generally maintaining comparable or higher recall to both LoFreq and iVar at the lowest MAF thresholds. bronko achieved consistently high F1 scores across the tested MAF thresholds, with performance comparable to or exceeding the other variant callers.

Downloaded from genome.cshlp.org on November 10, 2026 . Published by Cold Spring Harbor Laboratory Press

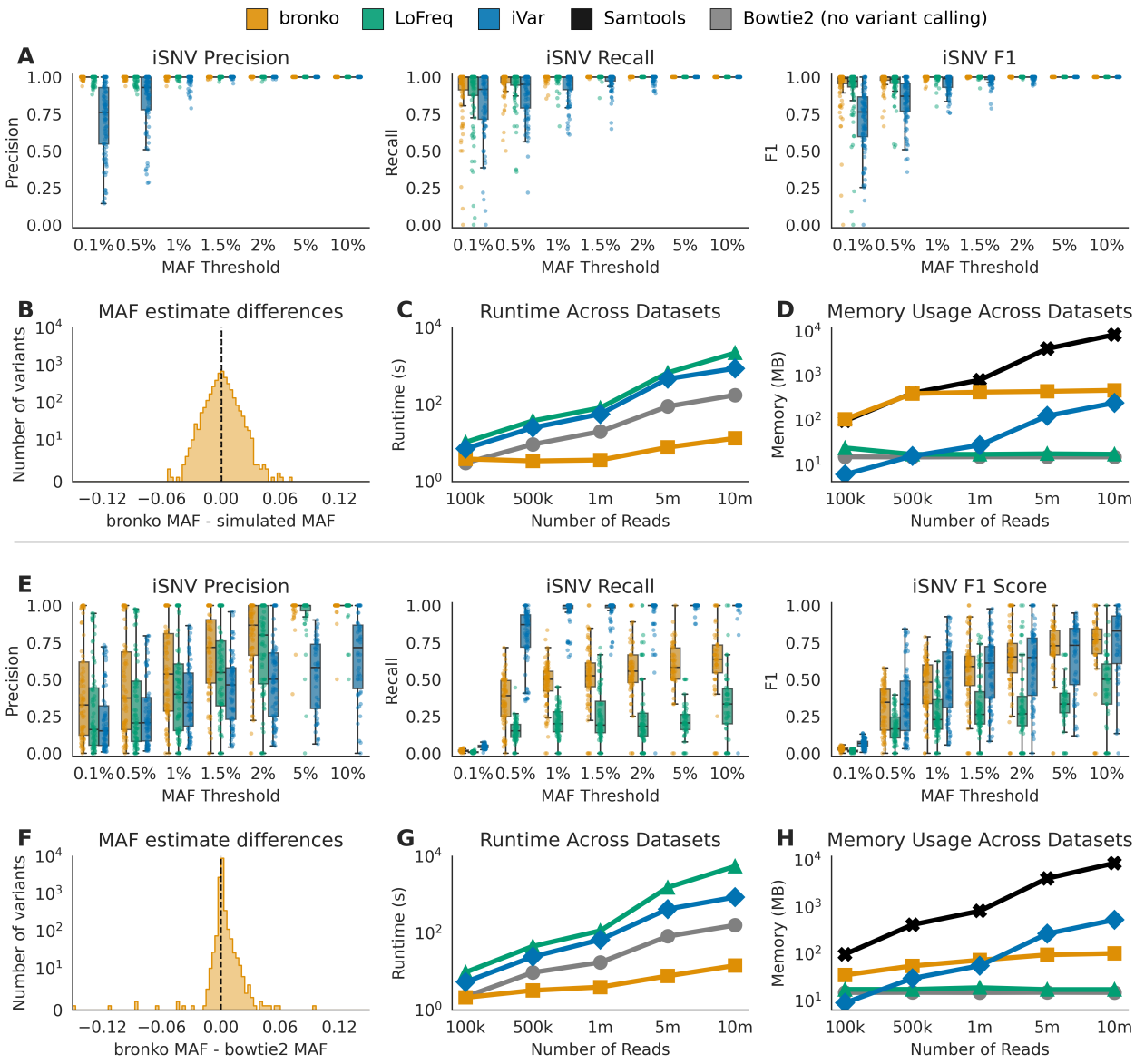


Fig. 3. Evaluation of iSNV variant calling across two simulation frameworks: Bygul (A–D) and GENOMICON-Seq (E–H), using RSV-A and HPV16, respectively. **A**. iSNV precision, recall, and F1 score for bronko, LoFreq, and iVar over increasing MAF thresholds for 100 simulated RSV-A samples generated using Bygul. Each dot represents a single sample. **B**. Histogram of the difference between bronko-estimated MAF and the truth simulation MAF for variants called by bronko. **C**. Runtime (seconds) across datasets of increasing size for each tool. Bowtie 2 and SAMtools time is incorporated into the full time for LoFreq and iVar. **D**. Memory usage across tools. iVar and LoFreq utilize SAMtools (black). Memory utilization without this step is shown in displayed colors. **E**. iSNV precision, recall, and F1 score for bronko, LoFreq, and iVar over increasing MAF thresholds for 810 simulated HPV16 samples generated using GENOMICON-Seq. Each dot represents the average over 10 samples simulated with the same parameters. **F**. Histogram of the difference between bronko-estimated MAF and the Bowtie 2-derived MAF for variants called by bronko. **G**. Runtime (seconds) across datasets of increasing size for each tool. Bowtie 2 and SAMtools time is incorporated into the full time for LoFreq and iVar. **H**. Memory usage across tools. iVar and LoFreq utilize SAMtools (black). Memory utilization without this step is shown in displayed colors.

The more challenging HPV16 simulations produced greater variation in variant calling performance across tools (**Fig. 3E**). bronko generally maintained higher precision than both LoFreq and iVar across MAF

Downloaded from genome.cshlp.org on November 10, 2026 . Published by Cold Spring Harbor Laboratory Press

thresholds, with iVar exhibiting greater variability and lower precision, even at higher MAF thresholds. Recall for bronko increased with higher MAF thresholds, reaching approximately 60–70% at the highest thresholds, while iVar maintained near perfect recall across most thresholds. LoFreq exhibited lower recall than both bronko and iVar across the tested thresholds, likely due to the presence of substantial strand biases in the simulation (see Supplementary Materials for further details). When relaxing bronko's own strand bias filter, bronko's recall became comparable to iVar without any significant loss in precision, raising the F1 score to nearly 1 for MAF $\geq 5\%$ (**Supplementary Fig. S3**). Despite iVar's higher recall, its lower and more variable precision resulted in bronko achieving comparable F1 scores across most MAF thresholds (**Fig. 3E**). Altogether, these results show that bronko achieves comparable performance to existing approaches, even in challenging variant calling conditions.

Given bronko constructs its pileup through direct pseudo-mapping of k-mer counts rather than traditional alignment, we also evaluated its ability to accurately estimate minor allele frequencies relative to alignment-based methods (see **Fig. 3B,F**). Specifically, we compared the difference between MAFs reported by bronko and those derived from truth allele frequencies (B) and Bowtie 2-based pileups then used by iVar and LoFreq (F). Overall, bronko exhibited good concordance with the truth sets, with 98.5% and 99.6% of minor variants having an absolute discordance of less than 0.03 across the two experiments.

On the computational performance side, bronko demonstrated a significant reduction in computational requirements compared to both LoFreq and iVar (**Fig. 3C-D,G-H**). At 1 million reads, bronko completed variant calling in less than 3 seconds in both experiments, compared with 56 and 73 seconds for iVar and 82 and 119 seconds for LoFreq, respectively. When the number of reads increased to 10 million, bronko maintained runtimes of approximately 10 seconds, whereas iVar required 871 and 890 seconds (14.5 and 14.8 minutes) and LoFreq required 2,199 and 5,409 seconds (36.7 and 90.1 minutes) across the two experiments. Thus, bronko was approximately two to three orders of magnitude faster than the alignment-based approaches. Furthermore, bronko consistently required less than 500 MB of memory across sequencing depths. In contrast, the complete alignment-based pipelines exhibited substantially greater memory requirements, primarily due to intermediate SAM/BAM file manipulation using SAMtools (Li et al. 2009b). At 10 million reads, peak memory usage for both pipelines reached approximately 8.5 GB, largely driven by intermediate SAM file handling. Even when excluding this overhead, iVar exhibited somewhat greater memory usage on larger datasets, in part because it directly pipes SAMtools mpileup output into the variant caller. LoFreq itself, in contrast, consistently used less than 100 MB of memory regardless of dataset size. Nevertheless, both alignment-based approaches require intermediate alignment and SAM/BAM processing, making bronko substantially more memory efficient when considering the complete variant calling workflow (**Fig. 3D,H**).

Downloaded from genome.cshlp.org on November 10, 2026 . Published by Cold Spring Harbor Laboratory Press

Table 1. Alignment SNP overlap and performance comparison across simulated datasets. Standard denotes bronko's default procedure, while --indels shows performance with the optional indel reconstruction procedure. Runtime and memory usage are also shown separately.

Dataset (# samples)	Total Data Size (TB)	Runtime, Memory		SNPs			Shared SNPs (%)
		Alignment	bronko	Alignment	bronko	Shared	
CoV (n=543)	1.03	48.54 hrs 2.12 GB	**1.59 hrs 1.987 GB** standard	0	0	3647	100
HIV (S1) (n=41)	0.012	15.88 min **0.35 GB**	**1.99 min** 0.74 GB standard	1175	11	9041	88.5
			2.50 min 1.10 GB --indels	20	13	10196	**99.8**
HIV (S2) (n=39)	0.011	13.05 min **0.35 GB**	**1.65 min** 0.72 GB standard	134	0	3754	96.6
			2.20 min 1.07 GB --indels	0	0	3888	**100**
HIV (S3) (n=37)	0.011	12.92 min **0.34 GB**	**1.57 min** 0.72 GB standard	0	0	664	100

Given these experiments relied on a single reference genome, we additionally evaluated the performance of bronko's `build` and `call` commands across varying genome lengths and numbers of reference genomes (**Supplementary Figs. S4–S5**). We found that bronko's performance remained stable for genomes up to approximately 1 Mbp, with performance beginning to degrade at genome lengths larger than 1 Mbp. When reference genomes had uniform lengths, runtime scaled approximately linearly with the number of input genomes.

Evaluation of SNPs from multiple sequence alignments across diverse viral datasets

To evaluate bronko's ability to produce a multiple sequence alignment from multiple viral sequencing datasets, we simulated viral outbreaks and compared the SNPs included in the multiple sequence alignments produced by bronko to those included from the core-genome alignment tool Parsnp2 (Kille et al. 2024) (see **Table 1**). First, we simulated 543 virus genome sequences emulating the early spread of SARS-CoV-2 (CoV). To do so, we used favites-lite (Moshiri 2025), which returns mutated sequences based on a known transmission network. Simulated reads were then generated for each sequence using ART (Huang et al. 2012) at 30,000x coverage, totaling over 1 TB of sequencing data across all samples. We ran bronko to call variants and generate an alignment across all samples with the reference sequence from the origin of the outbreak. Separately, each set of reads was mapped to the reference using Bowtie 2 (Langmead et al. 2012) and consensus sequences

Downloaded from genome.cshlp.org on November 10, 2026 . Published by Cold Spring Harbor Laboratory Press

were generated using SAMtools mpileup (Li et al. 2009b). Parsnp2 was then used to align the consensus sequences to get a final core-genome alignment for comparison to bronko.

bronko successfully processed the full dataset in just over 90 minutes, averaging just over 10 seconds per sample. In contrast, the read alignment-based approach required more than 48 hours on 32 cores, with Parsnp2 adding fewer than 3 minutes to produce the final core-genome alignment. Both pipelines used approximately 2 GB of RAM. Notably, the comparison workflow did not include the detection of minor iSNVs, which bronko identifies natively and thus is included in the runtime. The SNPs identified by bronko and Parsnp2 were identical (3,647 variants total), highlighting strong performance of bronko for another use case. However, this dataset, where genomes differed between 2-14 (median 6) SNPs from the reference, represents a relatively simple case for bronko's algorithm as long as there are no local hot spots of variation. To evaluate our approach on a more challenging dataset, we repeated this experiment with three smaller sets of simulated HIV genomes (full details and discussion available in supplementary materials). In short, we simulated three datasets (S1, S2, and S3), varying only the height parameter in the Non-Homogeneous Yule model used to generate the HIV phylogeny (36, 12, and 2, respectively). This design allowed us to assess how increasing divergence between reference and samples affects bronko's performance, with the three sets representing 3%, 1%, and 0.2% divergence, respectively. Overall, we found bronko to perform better as the divergence decreased, with only a mild drop in sensitivity for increased divergence. A large percentage of the false negatives could be directly attributed to decreased sensitivity within k-length regions containing ≥ 3 variants (see Supplementary Table S1 and Supplementary Materials). Enabling indel reconstruction substantially improved recovery in these regions, increasing the proportion of Parsnp2 SNPs recovered from 88.5% to 99.8% on dataset S1 and from 96.6% to 100% on dataset S2 (Table 1). This improvement comes at a modest computational cost, with runtime increasing from 1.99 to 2.50 minutes for S1 and from 1.65 to 2.20 minutes for S2. These results suggest that indel reconstruction is particularly beneficial for more divergent datasets or those containing locally dense variation. Therefore, we recommend this option for datasets where indels or divergent sequences are suspected.

We also compared bronko directly to SKA2, an alternative alignment-free SNP calling approach developed for bacterial epidemiology (**see Supplementary Tables S2–S3**). On the most divergent HIV dataset (S1), bronko recovered 88.5% (99.8% with indel reconstruction) of the variants identified by Parsnp2, whereas SKA2 recovered only 35.5%. Similarly, on dataset S2, bronko recovered 96.6% (100% with indel reconstruction) of variants, compared to 64.9% for SKA2. These results demonstrate a substantial sensitivity advantage for bronko over SKA2, particularly as sample divergence approaches 1% and beyond. We note, however, that SKA2 was developed primarily for bacterial rather than viral genomics, and therefore this comparison should be interpreted as an informative comparison of methods designed for different applications.

Downloaded from genome.cshlp.org on November 10, 2026 . Published by Cold Spring Harbor Laboratory Press

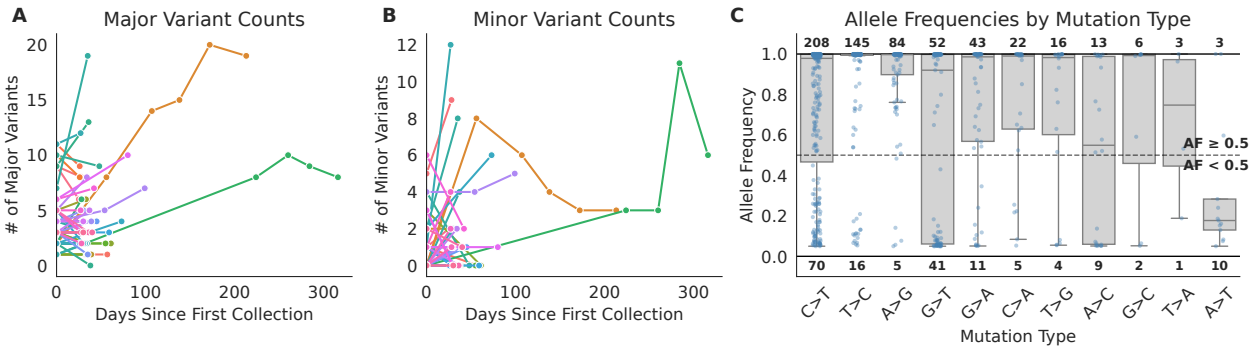


Fig. 4. Analysis of chronically infected patients with SARS-CoV-2 **A**. Number of major variants for each individual across subsequent sampling periods, with each individual shown with a different color line **B**. Number of minor variants for each individual across subsequent sampling periods, with each individual shown using a different color line **C** Frequency of major and minor mutations for each possible substitution. Mutations are sorted from left to right in order of the total number of major variants. The number of major variants (>=50% MAF) is shown above the boxplot, while the sum of minor variants (<50% MAF) is shown below

Tracking viral evolution within chronically-infected SARS-CoV-2 patients

To demonstrate a real-world application of bronko, we re-analyzed a subset of a dataset from a recent study of chronically infected SARS-CoV-2 patients from the UK (Ghafari et al. 2025). In the original work, the authors primarily investigated variants with a MAF $\geq$ 20%, citing low viral titers in many samples. We focused on high-quality samples (mean sequencing depth $\geq$ 3000$\times$ and breadth of coverage $\geq$ 97%) to investigate additional potentially relevant low-frequency variants above 5%, a conservative MAF threshold that aligned with high precision in our simulated results. We started by downloading samples with associated SRA accessions, resulting in 939 individual samples. Each of these was processed using bronko using a database with 2 representative strains from each Nextstrain (Hadfield et al. 2018) clade of SARS-CoV-2 (the first and last by sampling date, filtered for complete genomes with no ambiguous bases). This resulted in 100 total references from many different clades and timepoints throughout the pandemic. After quality filtering for breadth and depth of coverage, we were left with 372 samples. Of these, only 59 individuals contained more than 1 sample, reducing the final number of samples included in our analyses to 128.

For the remaining 128 samples, 773 total mutations were identified with respect to the selected reference genomes after additional strand balance and problematic site filtering (see Supplementary materials), comprising of 595 major variants (MAF $\geq$ 0.5) and 178 minor variants (0.05 $\leq$ MAF $<$ 0.5). Among them, 374 (221 major, 153 minor) were unique within an individual patient, while the remaining 399 were recurrent within individual patients (374 major variants across 177 individuals/positions combinations; 25 minor variants across 17 individuals/positions). This pattern supports prior observations (Xue et al. 2018), that major variants tend to persist in an individual whereas minor variants are typically more transient. Additionally, as seen in **Fig. 4A/B**, both the number of major variants and number of minor variants remained mostly

Downloaded from genome.cshlp.org on November 10, 2026 . Published by Cold Spring Harbor Laboratory Press

stable across samples within individuals, except for individuals with longer infection periods (>100 days), suggesting that substantial diversification of SARS-CoV-2 does not occur in a typical infection period (<14 days).

We next examined within-host temporal shifts between variant classes (a minor variant transitioning into a major variant, or vice versa) across timepoints within an individual. Across all mutations and patients, we found 11 (8 minor to major, 3 major to minor) such transitions among five individuals. Of the 8 minor to major transitions, one individual, shown in orange in **Fig. 4A/B** was responsible for three of these. Two of the three developed into a major variant from below 10% MAF. Four other patients contributed two, one, one, and one additional minor to major variant transitions. Of these, only one other developed from below 20% MAF, with the other four initially being identified between 20 and 50% MAF. While it is difficult to make significant conclusions from such a small sample, these cases reiterate the value of tracking minor variants that could develop into major variants, even despite their generally transient nature.

Finally, we characterized the spectrum of allele frequencies across mutation types for both major and minor mutations (see **Fig. 4C**). Again, in alignment with many previous studies (Markov et al. 2023; Abbasian et al. 2023), C>T, T>C, A>G, and G>T mutations dominated. Although this example is only a small sample set in chronically infected individuals, we also observed that proportionally, some mutation types (C>T, T>C, A>G, and G<A, C>A) had more major variants than minor, suggesting that these mutations may be more likely to establish themselves in an individual. On the other hand, others such as G>T and A>C less commonly established as major variants, suggesting they are rarely advantageous for the virus. The other potential explanation is that these variants are more likely to be artifacts of the sequencing and PCR process.

Discussion

In this work we introduced bronko, an alignment-free framework for identifying viral genome variation directly from sequencing reads. By constructing pileups from k-mers with small edit distance to the reference instead of relying on read alignment, bronko bypasses the computational bottlenecks inherent to traditional read-alignment based pipelines. Across simulated viral sequencing datasets, bronko achieved comparable precision and recall in identifying minor variants to established variant calling tools such as LoFreq and iVar, while operating 2 to 3 orders of magnitude faster. bronko also was able to reconstruct alignments across larger sample sets that compared favorably to classical consensus-generation and core-genome alignment, with challenges only arising when the divergence between genomes was very high. In practice, the impacts of high divergence between reference(s) and the data are limited by two factors. First, bronko's flexible reference database allows users to leverage existing sets of viral reference sequences that are related to the given sample

Downloaded from genome.cshlp.org on November 10, 2026 . Published by Cold Spring Harbor Laboratory Press

population, which reduces the cases of high divergence between the query and reference. Secondly, bronko's de novo variant reconstruction enables sensitive detection of variants within genomic regions containing indels or multiple closely spaced mutations, provided that sequencing fully covers the given region. Together, this framework enables rapid reference selection and variant calling across large-scale studies of diverse lineages where a single reference may cause additional biases.

bronko's use of a (1,2) LSB function allows it to identify mutations as long as there are not 3 variants within a k-length region. However, even in this case, both flanking variants would be identified, and only the middle of the three variants would be missed. This is a sensitivity improvement over split-k-mer based methods such as kSNP and SKA2, which at their core fail to detect any variants occurring in a $\lfloor \frac{k}{2} \rfloor$ base region. While these methods are generally well-suited to bacterial genomes, where larger genome sizes, higher divergence, and more dispersed variation make missed sites less consequential, viral genomes demand greater sensitivity in many cases, particularly if divergence increases towards and beyond 1%. In practice, however, several parameters can influence this theoretical sensitivity. Most obviously, the k-mer size (default 21) directly influences the sensitivity and specificity, and is configurable by the user. To mitigate sequencing artifacts near the ends of reads, which can often present at low frequency, bronko by default also fixes bases at the end of k-mers such that they cannot be used to detect variants. This approach reduces the number of spurious variants, but also decreases sensitivity in variant-dense regions. This tradeoff reflects our broader design principle of prioritizing precision over sensitivity in bronko, particularly for minor variants, which can closely resemble sequencing artifacts and are therefore inherently challenging to identify accurately (McCrone et al. 2016). Nonetheless, even with the fixed bases at the end of reads, we are still able to identify variants as long as there is no variant within the number of fixed bases plus the number of expected supporting k-mers (by default 2+2), still outperforming split-k-mer based methods. For users interested in maximum sensitivity when identifying major variants, these parameters can simply be set to zero to achieve the maximum theoretical sensitivity described above.

Additionally, as proposed in (Chen et al. 2023; Yuan et al. 2024), several alternative LSB functions exist beyond the (1,2) scheme implemented in bronko. These variants of LSB functions could, in principle, improve sensitivity in capturing clustered regions of local diversity. However, the strength of the (1,2) function lies in its simplicity, with each bucket directly corresponding to a potential SNP. In contrast, more complex LSB functions (ex. (2,3)) introduce additional complexity, such as more than k buckets per k-mer, which complicates the mapping protocols. Additionally, allowing for more distant matches increases the potential for unintentional hits, which may necessitate a larger k-mer size or other algorithmic adjustments. Exploring these alternative functions, therefore, presents an interesting direction for future work, although it remains to be seen whether the increased function complexity would yield meaningful gains in performance.

Downloaded from genome.cshlp.org on November 10, 2026 . Published by Cold Spring Harbor Laboratory Press

We also aim to scale bronko's framework to larger genomes. At present, the runtime and memory costs of enumerating each k-mer k times makes bronko more well-suited for viral genomes and other small targets, however ongoing optimization could further extend efficient support to larger bacterial genomes and other larger targets. Finally, while we showed our low frequency variant calling approach performs well, there are many other potential avenues for improvement, including integrating explicit models on sequencing and PCR errors, or utilizing existing datasets to classify artifacts from real signal (Mostefai et al. 2024). Overall, low frequency variant calling remains challenging, and users should be careful interpreting calls from any tool.

Code Availability

bronko is implemented in Rust and publicly available at `https://github.com/treangenlab/bronko` or via conda at `https://anaconda.org/channels/bioconda/packages/bronko/overview`. All results, evaluations, and other code used in this study are available at `https://github.com/treangenlab/bronko-test`. The bronko source code used in this work, along with the code and materials used for experimental evaluation, is also available in the Supplemental Material

Acknowledgments

R.D.D is supported by a training fellowship from the Gulf Coast Consortia, on the NLM Training Program in Biomedical Informatics & Data Science (T15LM007093). M.J.T is supported for this effort by NIH grant 1U54AG089335-01. T.J.T is supported by NIH grant P01-AI152999 and NSF awards IIS-2239114 and EF-2126387. This work was also supported in part by the Texas Medical Center Genomic Center for Infectious Diseases (TMC-GCID) through NIH NIAID grant 1U19AI144297. We would also like to thank Dr. Michael Nute for his valuable feedback on the manuscript. Lastly, the authors would like to thank the anonymous reviewers from both RECOMB and Genome Research for their helpful comments and feedback on this work.

Author Contributions

R.D.D Conceptualization, Methodology, Software, Formal analysis, Investigation, Data curation, Visualization, Writing – original draft, Writing – review & editing. **M.J.T** Conceptualization, Supervision, Writing – review & editing. **T.J.T** Conceptualization, Supervision, Writing – review & editing.

Competing Interests

The authors declare no competing interests.

Downloaded from genome.cshlp.org on November 10, 2026 . Published by Cold Spring Harbor Laboratory Press

References

Abbasian, M H, M Mahmanzar, K Rahimian, B Mahdavi, S Tokhanbigli, B Moradi, M M Sisakht, and Y Deng Feb. 2023. Global landscape of SARS-CoV-2 mutations and conserved regions. In: *Journal of Translational Medicine* 21, 152. DOI: 10.1186/s12967-023-03996-w.

Audano, P A, S Ravishankar, and F O Vannberg May 2018. Mapping-free variant calling using haplotype reconstruction from k-mer frequencies. In: *Bioinformatics* 34.10, 1659–1665. DOI: 10.1093/bioinformatics/btx753.

Berger, B and Y W Yu Apr. 2023. Navigating bottlenecks and trade-offs in genomic data analysis. en. In: *Nature Reviews Genetics* 24.4. Publisher: Nature Publishing Group, 235–250. DOI: 10.1038/s41576-022-00551-z.

Chen, K and M Shao July 2023. Locality-sensitive bucketing functions for the edit distance. In: *Algorithms for Molecular Biology* 18.1, 7. DOI: 10.1186/s13015-023-00234-2.

Chong, Z, J Ruan, M Gao, W Zhou, T Chen, X Fan, L Ding, A Y Lee, P Boutros, J Chen, et al. Jan. 2017. novoBreak: local assembly for breakpoint detection in cancer genomes. en. In: *Nature Methods* 14.1. Publisher: Nature Publishing Group, 65–67. DOI: 10.1038/nmeth.4084.

De Maio, N, W Boulton, L Weilguny, C R Walker, Y Turakhia, R Corbett-Detig, and N Goldman 2022. phastSim: efficient simulation of sequence evolution for pandemic-scale datasets. In: *PLoS computational biology* 18.4, e1010056.

Denti, L, P Khorsand, P Bonizzoni, F Hormozdiari, and R Chikhi Apr. 2023. SVDSS: structural variation discovery in hard-to-call genomic regions using sample-specific strings from accurate long reads. en. In: *Nature Methods* 20.4. Publisher: Nature Publishing Group, 550–558. DOI: 10.1038/s41592-022-01674-1.

Derelle, R, K Madon, J Hellewell, V Rodríguez-Bouza, N Arinaminpathy, A Lalvani, N J Croucher, S R Harris, J A Lees, and L Chindelevitch Apr. 2025. Reference-Free Variant Calling with Local Graph Construction with ska lo (SKA). In: *Molecular Biology and Evolution* 42.4, msaf077. DOI: 10.1093/molbev/msaf077.

Derelle, R, J v Wachsmann, T Mäklin, J Hellewell, T Russell, A Lalvani, L Chindelevitch, N J Croucher, S R Harris, and J A Lees Oct. 2024. Seamless, rapid, and accurate analyses of outbreak genomic data using split k-mer analysis. en. In: *Genome Research* 34.10. Company: Cold Spring Harbor Laboratory Press Distributor: Cold Spring Harbor Laboratory Press Institution: Cold Spring Harbor Laboratory Press Label: Cold Spring Harbor Laboratory Press Publisher: Cold Spring Harbor Lab, 1661–1673. DOI: 10.1101/gr.279449.124.

Downloaded from genome.cshlp.org on November 10, 2026 . Published by Cold Spring Harbor Laboratory Press

Edgar, R June 2020. URMAP, an ultra-fast read mapper. en. In: *PeerJ* 8. Publisher: PeerJ Inc., e9338. DOI: 10.7717/peerj.9338.

Efanov, A A, S A Ivliev, and A G Shagraev Mar. 2021. "Welford's algorithm for weighted statistics". In: *2021 3rd International Youth Conference on Radio Electronics, Electrical and Power Engineering (REEPE)*, 1–5. DOI: 10.1109/REEPE51337.2021.9387973.

Ekim, B, K Sahlin, P Medvedev, B Berger, and R Chikhi July 2023. Efficient mapping of accurate long reads in minimizer space with mapquik. en. In: *Genome Research* 33.7. Company: Cold Spring Harbor Laboratory Press Distributor: Cold Spring Harbor Laboratory Press Institution: Cold Spring Harbor Laboratory Press Label: Cold Spring Harbor Laboratory Press Publisher: Cold Spring Harbor Lab, 1188–1197. DOI: 10.1101/gr.277679.123.

Fang, H, E A Bergmann, K Arora, V Vacic, M C Zody, I Iossifov, J A O'Rawe, Y Wu, L T Jimenez Barron, J Rosenbaum, et al. Dec. 2016. Indel variant analysis of short-read sequencing data with Scalpel. en. In: *Nature Protocols* 11.12. Publisher: Nature Publishing Group, 2529–2548. DOI: 10.1038/nprot.2016.150.

Gardner, S N, T Slezak, and B G Hall Sept. 2015. kSNP3.0: SNP detection and phylogenetic analysis of genomes without genome alignment or reference genome. In: *Bioinformatics* 31.17, 2877–2878. DOI: 10.1093/bioinformatics/btv271.

Gardner, S N and B G Hall Dec. 2013. When Whole-Genome Alignments Just Won't Work: kSNP v2 Software for Alignment-Free SNP Discovery and Phylogenetics of Hundreds of Microbial Genomes. en. In: *PLOS ONE* 8.12. Publisher: Public Library of Science, e81760. DOI: 10.1371/journal.pone.0081760.

Ghafari, M, S A Kemp, M Hall, J Clarke, L Ferretti, L Thomson, R Studley, A S Walker, T Golubchik, and K Lythgoe Sept. 2025. SARS-CoV-2 genomic diversity and within-host evolution in individuals with persistent infection in the UK: an observational, longitudinal, population-based surveillance study. English. In: *The Lancet Microbe* 6.9. Publisher: Elsevier. DOI: 10.1016/j.lanmic.2025.101154.

Gold, T L, K M McPhaul, H Lin, R Doughty, I M Berry, F Hong, J Lai, T J Treangen, J Srebric, and D K Milton 2025. Impact of Ventilation on Respiratory Virus Transmission in College Residence Hall Cohorts: Potential for Causal Inference About Mode of Transmission. en. In: *Influenza and Other Respiratory Viruses* 19.7. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/irv.70133, e70133. DOI: 10.1111/irv.70133.

Grubaugh, N D, K Gangavarapu, J Quick, N L Matteson, J G De Jesus, B J Main, A L Tan, L M Paul, D E Brackney, S Grewal, et al. Jan. 2019. An amplicon-based sequencing framework for accurately measuring intrahost virus diversity using PrimalSeq and iVar. In: *Genome Biology* 20.1, 8. DOI: 10.1186/s13059-018-1618-7.

Downloaded from genome.cshlp.org on November 10, 2026 . Published by Cold Spring Harbor Laboratory Press

574 Guo, Y, J Li, C.-I Li, J Long, D C Samuels, and Y Shyr Nov. 2012. The effect of strand bias in Illumina
575 short-read sequencing data. In: *BMC Genomics* 13.1, 666. DOI: 10.1186/1471-2164-13-666.
576 Hadfield, J, C Megill, S M Bell, J Huddleston, B Potter, C Callender, P Sagulenko, T Bedford, and R A
577 Neher Dec. 2018. Nextstrain: real-time tracking of pathogen evolution. In: *Bioinformatics* 34.23, 4121–
578 4123. DOI: 10.1093/bioinformatics/bty407.
579 Houldcroft, C J, M A Beale, and J Breuer Mar. 2017. Clinical and biological insights from viral genome
580 sequencing. en. In: *Nature Reviews Microbiology* 15.3. Publisher: Nature Publishing Group, 183–192.
581 DOI: 10.1038/nrmicro.2016.182.
582 Huang, W, L Li, J R Myers, and G T Marth Feb. 2012. ART: a next-generation sequencing read simulator.
583 In: *Bioinformatics* 28.4, 593–594. DOI: 10.1093/bioinformatics/btr708.
584 Iqbal, Z, M Caccamo, I Turner, P Flicek, and G McVean Feb. 2012. De novo assembly and genotyping
585 of variants using colored de Bruijn graphs. en. In: *Nature Genetics* 44.2. Publisher: Nature Publishing
586 Group, 226–232. DOI: 10.1038/ng.1028.
587 Jansz, N and G J Faulkner May 2024. Viral genome sequencing methods: benefits and pitfalls of current
588 approaches. In: *Biochemical Society Transactions* 52.3, 1431–1447. DOI: 10.1042/BST20231322.
589 Khorsand, P and F Hormozdiari Jan. 2021. Nebula: ultra-efficient mapping-free structural variant genotyper.
590 In: *Nucleic Acids Research* 49.8, e47. DOI: 10.1093/nar/gkab025.
591 Kille, B, M G Nute, V Huang, E Kim, A M Phillippy, and T J Treangen May 2024. Parsnp 2.0: scalable
592 core-genome alignment for massive microbial datasets. In: *Bioinformatics* 40.5, btae311. DOI: 10.1093/
593 bioinformatics/btae311.
594 Kockan, C, F Hach, I Sarrafi, R H Bell, B McConeghy, K Beja, A Haegert, A W Wyatt, S V Volik, K N Chi,
595 et al. Jan. 2017. SiNVICT: ultra-sensitive detection of single nucleotide variants and indels in circulating
596 tumour DNA. In: *Bioinformatics* 33.1, 26–34. DOI: 10.1093/bioinformatics/btw536.
597 Kokot, M, M Długosz, and S Deorowicz Sept. 2017. KMC 3: counting and manipulating k-mer statistics. In:
598 *Bioinformatics* 33.17, 2759–2761. DOI: 10.1093/bioinformatics/btx304.
599 Kozlov, A M, D Darriba, T Flouri, B Morel, and A Stamatakis 2019. RAxML-NG: a fast, scalable and
600 user-friendly tool for maximum likelihood phylogenetic inference. In: *Bioinformatics* 35.21, 4453–4455.
601 DOI: 10.1093/bioinformatics/btz305.
602 Lagström, S, S U Umu, M Lepistö, P Ellonen, R Meisal, I K Christiansen, O H Ambur, and T B Rounge
603 2019. TaME-seq: An efficient sequencing approach for characterisation of HPV genomic variability and
604 chromosomal integration. In: *Scientific Reports* 9.1, 524.
605 Langmead, B and S L Salzberg Apr. 2012. Fast gapped-read alignment with Bowtie 2. en. In: *Nature Methods*
606 9.4. Publisher: Nature Publishing Group, 357–359. DOI: 10.1038/nmeth.1923.

Downloaded from genome.cshlp.org on November 10, 2026 . Published by Cold Spring Harbor Laboratory Press

607 Lauring, A S Sept. 2020. Within-Host Viral Diversity: A Window into Viral Evolution. eng. In: *Annual*
608 *Review of Virology* 7.1, 63–81. DOI: 10.1146/annurev-virology-010320-061642.
609 Li, H and R Durbin July 2009. Fast and accurate short read alignment with Burrows–Wheeler transform.
610 In: *Bioinformatics* 25.14, 1754–1760. DOI: 10.1093/bioinformatics/btp324.
611 Li, H, B Handsaker, A Wysoker, T Fennell, J Ruan, N Homer, G Marth, G Abecasis, R Durbin, and
612 1000 Genome Project Data Processing Subgroup Aug. 2009. The Sequence Alignment/Map format and
613 SAMtools. In: *Bioinformatics* 25.16, 2078–2079. DOI: 10.1093/bioinformatics/btp352.
614 Markov, P V, M Ghafari, M Beer, K Lythgoe, P Simmonds, N I Stilianakis, and A Katzourakis June 2023.
615 The evolution of SARS-CoV-2. en. In: *Nature Reviews Microbiology* 21.6. Publisher: Nature Publishing
616 Group, 361–379. DOI: 10.1038/s41579-023-00878-2.
617 McCrone, J T and A S Lauring July 2016. Measurements of Intrahost Viral Diversity Are Extremely Sensitive
618 to Systematic Errors in Variant Calling. In: *Journal of Virology* 90.15, 6884–6895. DOI: 10.1128/JVI.
619 00667-16.
620 Moshiri, N 2025. "Scalable Epidemic Simulation Using FAVITES-Lite". en. In: *Computational Virology*. Ed.
621 by Muley, V Y. New York, NY: Springer US, 173–193. DOI: 10.1007/978-1-0716-4546-8_10.
622 Mostefai, F, J.-C Grenier, R Poujol, and J Hussin Dec. 2024. Refining SARS-CoV-2 intra-host variation
623 by leveraging large-scale sequencing data. In: *NAR Genomics and Bioinformatics* 6.4, lqae145. DOI:
624 10.1093/nargab/lqae145.
625 Muller, E, N Goardon, B Brault, A Rousselin, G Paimparay, A Legros, R Fouillet, O Bruet, A Tranchant,
626 F Domin, et al. Nov. 2016. OutLyzer: software for extracting low-allele-frequency tumor mutations from
627 sequencing background noise in clinical practice. en. In: *Oncotarget* 7.48. Publisher: Impact Journals,
628 79485–79493. DOI: 10.18632/oncotarget.13103.
629 Peterlongo, P, C Riou, E Drezen, and C Lemaitre Oct. 2017. *DiscoSnp++: de novo detection of small variants*
630 *from raw unassembled read set(s)*. en. Pages: 209965 Section: New Results. DOI: 10.1101/209965.
631 Sanjuán, R, M R Nebot, N Chirico, L M Mansky, and R Belshaw Oct. 2010. Viral Mutation Rates. In: *Journal*
632 *of Virology* 84.19. Publisher: American Society for Microbiology, 9733–9748. DOI: 10.1128/jvi.00694-
633 10.
634 Sapoval, N, Y Liu, E G Lou, L Hopkins, K B Ensor, R Schneider, L B Stadler, and T J Treangen May 2023.
635 Enabling accurate and early detection of recently emerged SARS-CoV-2 variants of concern in wastewater.
636 en. In: *Nature Communications* 14.1. Publisher: Nature Publishing Group, 2834. DOI: 10.1038/s41467-
637 023-38184-3.

Downloaded from genome.cshlp.org on November 10, 2026 . Published by Cold Spring Harbor Laboratory Press

638 Shajii, A, D Yorukoglu, Y William Yu, and B Berger Sept. 2016. Fast genotyping of known SNPs through
639 approximate k-mer matching. In: *Bioinformatics* 32.17, i538–i544. DOI: 10.1093/bioinformatics/
640 btw460.
641 Stosic, M S, J.-M Costanzi, O H Ambur, and T B Rounge July 2025. GENOMICON-Seq enables realistic
642 simulation of amplicon and exome sequencing for low-frequency mutation detection. en. In: *Scientific*
643 *Reports* 15.1. Publisher: Nature Publishing Group, 23003. DOI: 10.1038/s41598-025-05267-8.
644 Welford, B P Aug. 1962. Note on a Method for Calculating Corrected Sums of Squares and Products. In:
645 *Technometrics* 4.3, 419–420. DOI: 10.1080/00401706.1962.10490022.
646 Wilm, A, P P K Aw, D Bertrand, G H T Yeo, S H Ong, C H Wong, C C Khor, R Petric, M L Hibberd, and N
647 Nagarajan Dec. 2012. LoFreq: a sequence-quality aware, ultra-sensitive variant caller for uncovering cell-
648 population heterogeneity from high-throughput sequencing datasets. In: *Nucleic Acids Research* 40.22,
649 11189–11201. DOI: 10.1093/nar/gks918.
650 Xiang, X, B Lu, D Song, J Li, K Shu, and D Pu Nov. 2023. Evaluating the performance of low-frequency
651 variant calling tools for the detection of variants from short-read deep sequencing data. en. In: *Scientific*
652 *Reports* 13.1. Publisher: Nature Publishing Group, 20444. DOI: 10.1038/s41598-023-47135-3.
653 Xu, C, X Gu, R Padmanabhan, Z Wu, Q Peng, J DiCarlo, and Y Wang Apr. 2019. smCounter2: an ac-
654 curate low-frequency variant caller for targeted sequencing data with unique molecular identifiers. In:
655 *Bioinformatics* 35.8, 1299–1309. DOI: 10.1093/bioinformatics/bty790.
656 Xue, K S, L H Moncla, T Bedford, and J D Bloom Sept. 2018. Within-host evolution of human influenza
657 virus. In: *Trends in microbiology* 26.9, 781–793. DOI: 10.1016/j.tim.2018.02.007.
658 Yi, K, S Y Kim, T Bleazard, T Kim, J Youk, and Y S Ju Aug. 2021. Mutational spectrum of SARS-
659 CoV-2 during the global pandemic. en. In: *Experimental & Molecular Medicine* 53.8. Publisher: Nature
660 Publishing Group, 1229–1237. DOI: 10.1038/s12276-021-00658-z.
661 Yuan, X, K Chen, X Li, Q Shi, and M Shao July 2024. Learning locality-sensitive bucketing functions. In:
662 *Bioinformatics* 40.Supplement_1, i318–i327. DOI: 10.1093/bioinformatics/btae228.