Sequence analysis

RIFRAF: a frame-resolving consensus algorithm

Kemal Eren¹ and Ben Murrell^{2,*}

¹Bioinformatics and Systems Biology and ²Department of Medicine, University of California San Diego, La Jolla, CA 92093, USA

*To whom correspondence should be addressed.

Associate Editor: Bonnie Berger

Received on November 30, 2017; revised on April 27, 2018; editorial decision on May 21, 2018; accepted on May 22, 2018

Abstract

Motivation: Protein coding genes can be studied using long-read next generation sequencing. However, high rates of indel sequencing errors are problematic, corrupting the reading frame. Even the consensus of multiple independent sequence reads retains indel errors. To solve this problem, we introduce Reference-Informed Frame-Resolving multiple-Alignment Free template inference algorithm (RIFRAF), a sequence consensus algorithm that takes a set of error-prone reads and a reference sequence and infers an accurate in-frame consensus. RIFRAF uses a novel structure, analogous to a two-layer hidden Markov model: the consensus is optimized to maximize alignment scores with both the set of noisy reads and with a reference. The template-to-reads component of the model encodes the preponderance of indels, and is sensitive to the per-base quality scores, giving greater weight to more accurate bases. The reference-to-template component of the model penalizes frame-destroying indels. A local search algorithm proceeds in stages to find the best consensus sequence for both objectives.

Results: Using Pacific Biosciences SMRT sequences from an HIV-1 *env* clone, NL4-3, we compare our approach to other consensus and frame correction methods. RIFRAF consistently finds a consensus sequence that is more accurate and in-frame, especially with small numbers of reads. It was able to perfectly reconstruct over 80% of consensus sequences from as few as three reads, whereas the best alternative required twice as many. RIFRAF is able to achieve these results and keep the consensus in-frame even with a distantly related reference sequence. Moreover, unlike other frame correction methods, RIFRAF can detect and keep true indels while removing erroneous ones.

Availability and implementation: RIFRAF is implemented in Julia, and source code is publicly available at <https://github.com/MurrellGroup/Rifraf.jl>.

Contact: bmurrell@ucsd.edu

Supplementary information: [Supplementary data](#) are available at *Bioinformatics* online.

1 Introduction

The problem of finding the consensus of a set of sequences is fundamental to bioinformatics, especially in the age of high-throughput sequencing. This paper addresses the task of reconstructing an unknown true template sequence from a set of error-prone reads. Many algorithms that solve this task focus on *de-novo* or reference-guided assembly of short reads (Nagarajan and Pop, 2013; Pal *et al.*, 2006; Paszkiewicz and Studholme, 2010). However, with the advent of third-generation single-molecule sequencing technologies, such as

Pacific Biosciences' SMRT sequencing protocol (Eid *et al.*, 2009), it is now possible to perform full-length sequencing of entire genes or small genomes. Here, we restrict our focus to *amplicon* sequencing—where the sequences have the same start and end points. An example application would be targeted sequencing of an entire gene from a viral population [e.g. (Laird Smith *et al.*, 2016)]. We focus just on the template inference problem, assuming that reads have first been grouped by genetic identity, either using primer ID barcodes (Jabara *et al.*, 2011; Sheward *et al.*, 2012), or some form of clustering.

Inferring template sequences by consensus from a multiple sequence alignment (MSA) may be inaccurate when there are few reads available, or when the reads contain many errors. SMRT sequencing in particular is known to contain mostly indel errors, especially in homopolymer runs. For example, in (Laird Smith *et al.*, 2016), we discovered that 80% of the sequencing errors were indels. If these indels carry over into the inferred template sequence, they cause frameshift errors which corrupt the reading frame, and render the amino acid sequence uninterpretable. If a reference sequence with a trusted reading frame is available, it can be exploited to inform template inference.

Current approaches that attempt to reconstruct in-frame template sequences consider these problems separately. There are approaches to infer the consensus of multiple reads, and there are approaches to correct the reading frame of an already-inferred template sequence. Here, we solve these problems jointly, simultaneously considering evidence from the reads and the reference sequence.

One common approach to inferring a template sequence is from MSAs, from which the consensus is calculated by taking the most common base in each column. Myriad multiple alignment algorithms are available (Pervez *et al.*, 2014), any of which may be used for this task. This paper uses MAFFT (Katoh and Standley, 2013; Katoh *et al.*, 2002) as an example of this strategy when comparing alternatives. A multitude of tools, such as the `cons` command in EMBOSS (Rice *et al.*, 2000), are available for computing the consensus of these alignments. Another approach is to use a partial order alignment (Lee *et al.*, 2002) representation of the set of sequences, and find the consensus sequence using dynamic programming to extract the heaviest bundles (Lee, 2003). This paper uses `poaV2` (<https://sourceforge.net/projects/poamsa/>) for comparison. Other implementations of this approach include `pbdagcon` (<https://github.com/PacificBiosciences/pbdagcon>), which was released by Pacific Biosciences specifically for raw SMRT sequence reads and `nanopolish` (Loman *et al.*, 2015), which wraps `poaV2` for Oxford Nanopore reads. Finally, specialized template inference methods are available for specific sequencing technologies; these methods model the specific behavior of their target protocol, such as read length and error model. In this domain, Pacific Biosciences developed the `Quiver` (Chin *et al.*, 2013) and `Arrow` algorithms (<https://github.com/PacificBiosciences/GenomicConsensus>) for building circular consensus sequences from raw ZMW reads.

Existing approaches for reading frame correction (such as `FrameBot`, which we use here as a comparator) exploit frame-aware codon alignment to a protein reference, followed by inserting or deleting bases in the target sequence (Wang *et al.*, 2013). Related algorithms include `FALP` and `LAST` (Sheetlin *et al.*, 2014), `Frame-Pro` (Du and Sun, 2016), `HMMFrame` (Zhang and Sun, 2011) and others. Another approach is hybrid sequencing, which supplements long single-molecule reads with short reads (Pervez *et al.*, 2014). Methods such as `HGAP` (Chin *et al.*, 2013) use hybrid sequence data to find and remove indels.

This paper introduces a new method for inferring template sequences of such reads: the Reference-Informed Frame-Resolving multiple-Alignment Free template inference algorithm (RIFRAF). RIFRAF considers evidence from both the reads and the reference simultaneously, allowing reads to inform the frame correction process and is sensitive to the read quality scores to ensure that high-quality bases are more informative. These features allow RIFRAF to make highly accurate predictions, even for a small number of error-prone reads. Unlike other frame-correction methods, RIFRAF can detect true frameshift-causing indels and keep them while removing spurious indels.

2 Materials and methods

RIFRAF addresses the following template inference problem. Let t be an unknown template sequence, which is sequenced N times to generate a set of N pairs of reads and quality scores

$\mathcal{R} = \{(s^n, p^n)\}_{n=1}^N$. Each read s^n is a noisy observation of t , and each p^n is a vector of error probabilities, one for each base in s^n . The i th character in read s is denoted s_i , and the substring from the i th to the j th character is denoted $s_{i:j}$. p_i is the probability that s_i is a substitution or an insertion, or that a deletion has occurred adjacent to position i . Additionally, we assume that t has evolved from a reference sequence r under a process that allows indels that preserve the reading frame (i.e. multiples of 3), but where frameshift indels are very unlikely. The task is to infer, as accurately as possible, the unknown template t , exploiting both the reads and the reference sequence. Exploiting frame information from a reference sequence is especially useful when the sequencing process has a high indel rate.

The structure of the full model is shown in Figure 1. RIFRAF has two layers: a reference-to-template layer and a template-to-reads layer. While each layer is conceptually similar to a hidden Markov model (Durbin *et al.*, 1998), for technical reasons related to exploiting base-call quality scores and due to some computational heuristics we adopt, our implementation does not fall precisely within the HMM framework. We thus describe our approach in the language of pairwise alignments, where the alignment penalty parameters (in the log probability domain) encode our expectations about the evolutionary process (reference-to-template layer), and the sequencing process (template-to-read layer). For convenience of communication, however, we still refer to our alignment scores as ‘likelihoods’.

The reads are conditionally independent of the reference given the template, which allows us to compose an overall likelihood from the two separate model layers for a candidate template sequence. We assume that the sequence $\hat{t}$ that maximizes this overall score will be the best estimate of the unobserved template t . The optimization procedure starts with an initial candidate c sequence and proceeds in an iterative greedy manner, mutating c at every step to improve those objectives. RIFRAF uses a number of techniques to speed up convergence: filtering mutations, accepting multiple mutations, forward and backward alignments, banding, batching, increasing indel penalties and multi-stage optimization.

RIFRAF is implemented in Julia (Bezanson *et al.*, 2017), a high-level scientific computing language.

Fig. 1. Overview of the RIFRAF model. In the top layer, our unknown template t (grey) is assumed to have an evolutionary relationship with a reference r , where r has a trusted reading frame. This evolutionary process has a low rate of frameshift indels. In the bottom layer, the sequencing process generates reads $s^1 \dots s^N$ that have quality score vectors $p^1 \dots p^N$. The goal of RIFRAF is to use the reference and the reads simultaneously to estimate the unknown template

2.1 Objective 1: pairwise alignment to reads

In order to find the optimal template, RIFRAF scores candidate template sequences by a pairwise alignment (Needleman and Wunsch, 1970) of each read s with the current candidate template c . Let A be the $|s| + 1 \times |c| + 1$ dynamic programming matrix for aligning c and s . Each $a_{i,j}$ is the likelihood of aligning prefix $s_{1..i}$ to prefix $c_{1..j}$. $a_{0,0}$ is initialized to 0, and the last cell $a_{|s|+1,|c|+1}$ contains the likelihood for the full alignment: $S(c|s) = a_{|s|+1,|c|+1}$. The overall score $S(c|\mathcal{R})$ of template sequence c is the sum over the alignment scores for all reads: $\sum_{s \in \mathcal{R}} S(c|s)$.

We allow error types (substitutions, insertions, deletions) to have tunable proportions: $\rho_{\text{mismatch}} + \rho_{\text{insertion}} + \rho_{\text{deletion}} = 1$, calling these proportions an ‘error model’ ρ . These parameters account for the different error profiles of different sequencing technologies. For instance, in SMRT sequencing, indels are more likely than substitutions and, from empirical frequencies in Laird Smith *et al.* (2016), we set $\rho_{\text{mismatch}} = 0.2$ and $\rho_{\text{insertion}}$ and $\rho_{\text{deletion}} = 0.4$. The alignment move scores are derived from these error probabilities.

RIFRAF also incorporates per-base sequence quality scores into the alignment moves. We interpret these quality scores as error probabilities, which is approximately correct for PacBio amplicon data [see Fig. 2 in Laird Smith *et al.* (2016)]. The scores for match, insertion and deletion moves depend on the product of the error model ρ and the quality scores, distributing the error probability the instrument assigned to that base among the possible kinds of error, according to the proportions in the error model ρ :

- i. A diagonal move from $a_{i-1,j-1}$ to $a_{i,j}$ has score $\log(1 - p_i)$ if $s_i = c_j$ (i.e. a match, scored as the log probability of no error), else $\log(\rho_{\text{mismatch}} \times p_i)$, the log probability of a substitution error.
- ii. A vertical move (insertion relative to c) from $a_{i-1,j}$ to $a_{i,j}$ has score $\log(\rho_{\text{insertion}} \times p_i)$, the log probability of an insertion error.
- iii. A horizontal move (deletion relative to c) from $a_{i,j-1}$ to $a_{i,j}$ has score $\log(\rho_{\text{deletion}} \times \max(p_i, p_{i+1}))$. A deletion relative to the candidate template is thus scored according to the log

Fig. 2. Codon moves in the reference alignment dynamic programming matrix. The goal is to favor a template that preserves the reading frame. Thus, in addition to the usual single match, insertion and deletion moves, codon insertions and deletions are also allowed, with a lower penalty than single-base indels

probability of a deletion error in the most error prone base either side of it. If $i = 0$, the score is just $\log(\rho_{\text{deletion}} \times p_1)$; similarly, $i = |s|$, the score is just $\log(\rho_{\text{deletion}} \times p_{|s|})$.

Sums of log probabilities are used for numerical accuracy, and for compatibility with the pairwise alignment algorithm. No affine gap penalties are used in this model.

This scoring system results in more accurate alignments because the penalties for mismatches, insertions and deletions are more severe when the template does not match high-quality regions of the reads. By default, PHRED values are capped at 30 because rarer sources of error that are not sequencing errors (e.g. PCR errors) may have very confident PHRED scores, and we do not wish these to be overly informative. This cap can be adjusted if these sources of error can be ruled out (e.g. if PCR was not used to generate the amplicon library).

2.2 Objective 2: frame-aware alignment to reference

To perform frame correction, RIFRAF requires a reference nucleotide sequence r , which is known to be in-frame. It models the reference sequence r as having diverged from the template t , where the differences between r and t represent evolutionary events, not sequencing error as in Objective 1. The score for the template-reference alignment is modified to reflect this difference. First, two new moves are allowed during alignment: codon insertion and codon deletion, each with their own penalty, as shown in Figure 2. Second, a new parameter w_{indel} is used as a multiplier for the non-codon insertion and deletion penalties. Together, these two modifications bias the alignment to prefer only codon indels, keeping the template in-frame. Because it uses nucleotide alignments, this method may be expected to work better with more closely related reference sequences, where nucleotide similarity is preserved.

We first let RIFRAF converge to a draft template c' without the reference sequence. This draft template is used to approximate the divergence between the true template and the reference, taking the edit distance normalized by the max length $d(r, c')/\max(|r|, |c'|)$ to obtain a per-base probability of template/reference disagreement (which is used in the same manner as the per-base quality scores p in Objective 1). Reference (mis)match, indel and codon error rates are provided as parameters, and the scores for each move are computed from error rate ρ as $\log(\rho)$, as before.

The insertion and deletion scores are multiplied by a penalty w_{indel} , which controls the influence of single insertions and deletions in the reference alignment. If w_{indel} is small, frame-destroying indels may appear in the template, but if it is large, the template will be forced into the reference reading frame, even if the unobserved template really did contain indels. As we show in Section 3, this penalty can be tuned to discard spurious indels while keeping true ones.

RIFRAF combines both objectives into a single score, allowing the reads to inform the frame correction. The score of the template to reference alignment is denoted $S_r(c|r)$, and the full score function is:

$$S(c|\mathcal{R}, r) = S(c|\mathcal{R}) + S_r(c|r).$$

2.3 Optimization procedure

An exhaustive search for the optimal template $c^* = \hat{t}$ would be intractable, so RIFRAF uses a variant of the following greedy search algorithm, with some optimizations to speed up convergence:

- i. Start with an initial candidate sequence c^0 . RIFRAF chooses the read with the lowest expected number of errors.
- ii. For the most recent guess c^k , examine a set of candidate single mutations, such as insertions, deletions and substitutions.

Note that different types of mutations are allowed in each optimization stage. Keep all that improve the score $S(c^k|\mathcal{R}, r)$. Call the set of candidate mutations $\mathcal{C}$.

- iii. If $\mathcal{C}$ is empty, accept c^k and terminate. Otherwise, choose some subset of $\mathcal{C}$, apply them to c^k to obtain c^{k+1} and iterate.

RIFRAF works in two stages, first optimizing just $S(c|\mathcal{R})$, and then optimizing the full $S(c|\mathcal{R}, r)$.

2.3.1 Filtering mutations

When comparing the template to the reads, we need not consider all possible modifications to the current template. For example, if any candidate mutation to c does not appear in any pairwise alignment of c with a read, that mutation need not be scored. Since it has no support among any observed sequence, it is likely to hurt the alignment score. Similarly, during the frame correction stage, the model only proposes insertions or deletions that appear in the pairwise alignment to reference.

2.3.2 Multiple mutations

Instead of accepting only the best mutation in $\mathcal{C}$, RIFRAF accepts all the mutations that are separated by a certain number of positions: n_{separate} (the default value is 15, i.e. five codons). The candidates are accepted in order from best to worst score. This policy allows RIFRAF to converge in many fewer iterations than if it only accepted one mutation per iteration. In most scenarios, n_{separate} ensures that the changes to the template are relatively independent of one another and therefore the score of one is unlikely to be affected by the acceptance of another. RIFRAF also detects when this heuristic fails, which would cause it to accept mutations that together harm the score: after accepting mutations in $\mathcal{C}$, it also compares the new score to the score that would be obtained from accepting only the single best mutation in $\mathcal{C}$ and optionally accepts that single mutation instead if it results in a better score.

2.3.3 Forward and backward alignments

Recomputing the full alignment matrix for each candidate mutation to c would be prohibitively expensive. For a sequence c from alphabet $\{A, C, G, T\}$, there are $4(|c| + 1)$ insertions, $3|c|$ substitutions and $|c|$ deletions to consider. Computing the alignment matrix $\mathbf{A}$ for each candidate requires $O(cs)$ operations, so each iteration of the proposed algorithm would require $O(Nc^2s)$ operations (we omit $|\cdot|$ in $O(\cdot)$ for clarity). Instead, RIFRAF uses forward and backward alignments to compute the new score for any single change to c by only recomputing a single column of $\mathbf{A}$ (Chin *et al.*, 2013).

To achieve this, we use a technique analogous to the forward-backward algorithm for HMMs. In addition to the prefix alignment matrix $\mathbf{A}$, where a_{ij} is the score for aligning prefix $s_{1..i}$ to prefix $c_{1..j}$, RIFRAF also computes the suffix alignment matrix $\mathbf{B}$, where b_{ij} is the score for aligning suffix $s_{i+1..|s|}$ to suffix $c_{j+1..|c|}$. Note that $a_{|s|,|c|} = b_{0,0}$ is the score for the full alignment. For any j , that alignment score can also be computed from columns $\mathbf{A}_{\cdot j}$ and $\mathbf{B}_{\cdot j}$, by taking the maximum of the element-wise sum over these columns:

$$\forall j \in [0 \dots v] : a_{|s|,|c|} = b_{0,0} = \max_i (a_{ij} + b_{ij}) \quad (1)$$

Modifying c_j leaves unchanged columns $0 \dots j-1$ of $\mathbf{A}$, and also leaves unchanged columns $j \dots v$ of $\mathbf{B}$. Therefore, for all three types of mutations, computing the new score requires that at most only a single new column of $\mathbf{A}$ must be re-calculated.

- i. Substitution at c_j : compute $\mathbf{A}_{\cdot j}$; new score is $\max_i (a_{ij} + b_{ij})$.

- ii. Insertion after c_j : compute $\mathbf{A}_{\cdot j+1}$; new score is $\max_i (a_{ij+1} + b_{ij})$.
- iii. Deletion of c_j : no new column necessary; new score is $\max_i (a_{ij-1} + b_{ij})$.

Using the forward and backward alignments, all possible mutations to the template can be scored in $O(Ncs)$ operations.

During the alignment of the template and reference, additional columns must be recomputed to account for codon insertion and deletion moves.

2.3.4 Banding

Despite the improvements from using forward and backward alignments, each iteration is still approximately quadratic in the length of the template, assuming $|c| \approx |s|$. Alignment banding (Chao *et al.*, 1993, 1992) reduces the number of operations per iteration. For a given bandwidth parameter b , the maximum usable column size in $\mathbf{A}$ and $\mathbf{B}$ is $2b + ||s| - |c|| \ll |s|$, so evaluating a possible mutation requires many fewer operations than recomputing the full column. Alignment moves are only allowed to originate inside the band, so alignment paths must stay within the band boundaries (Fig. 3). With banding, the time complexity per iteration becomes $O(Nc(\sqrt{s} + b))$, since $||s| - |c||$ grows like $\sqrt{|s|}$ under reasonable assumptions.

RIFRAF dynamically increases the bandwidth if the number of differences in the banded alignment is sufficiently larger than the expected number of differences implied by the read's quality scores, under the assumption that the difference between the template candidate c and the true template is much smaller than the number of sequencing errors in s . Let r be the observed number of differences between s and c , and e be the expected number of errors computed from the quality scores p . If the value of r is in the upper tail of a Poisson distribution with mean parameter e , then the bandwidth is doubled and the alignments are re-computed. α controls the size of this upper tail probability, with a default value of 0.1.

2.3.5 Batching

RIFRAF uses a variety of batching strategies to speed up convergence by estimating the full model likelihood from sub-samples of the reads. If the number of reads is greater than a threshold k (default 5), the best k reads by error rate are fixed as the initial batch and RIFRAF runs to convergence. This ensures that RIFRAF first converges without considering the many spurious mutations

Fig. 3. Banded alignment. Alignments must stay within the banded region of the dynamic programming matrix

presumably present in less accurate reads. The resulting initial guess is further refined at the refinement stage, this time with a different random batch of size n_{batch} (default 20) for each iteration. Sequences are chosen for inclusion in the batch by sampling from a multinomial distribution of their error rates, parameterized by parameter ρ between 0 and 1. When $\rho = 1$, all the weight is evenly distributed among the top n_{batch} sequences. Interpolating from $\rho = 1.0$ to $\rho = 0.5$, the probabilities become proportional to the read error rates. Interpolating from $\rho = 0.5$ to $\rho = 0$, the probabilities become uniform. By default, $\rho = 0.9^t$, where t is the number of iterations since random batching activated. Like the fixed batch, this strategy speeds up convergence by initially avoiding inaccurate reads, then gradually letting them contribute to resolve uncertain bases if necessary.

Ideally, n_{batch} is small enough to make each iteration fast, but large enough that RIFRAF converges stably. RIFRAF tries to detect if n_{batch} is too small by monitoring the change in score after each iteration. If the new score is worse than the old score by more than a certain percent (10% by default), n_{batch} is increased to $2n_{\text{batch}}$, then $3n_{\text{batch}}$, etc.

Combining all of the previous optimizations, a single iteration's time complexity is reduced from $O(Nc^2s)$ to $O(n_{\text{batch}}c(\sqrt{s} + b))$.

2.4 Increasing indel penalties

Whenever the algorithm converges to a template c^t , if single indel moves were used in computing $S_r(\cdot)$, the single insertion and deletion scores are multiplied by a parameter w_{indel} , increasingly encouraging the alignment with the reference to use only codon indels, thereby keeping c in-frame. This process repeats up to m times, so the maximum multiplier is $(w_{\text{indel}})^m$. If the penalty is large enough, the template will always be forced into the reference's reading frame, which is the default behavior. However, some template sequences really are out of frame relative to the reference. The indel penalties can be tuned so that RIFRAF correctly identifies true frameshifts, with a small risk of allowing some spurious ones.

2.5 Multi-stage optimization

The full optimization procedure proceeds in stages, allowing RIFRAF to converge quickly by focusing on different objectives in different stages.

- i. Initial stage: Do not use the reference. Propose all mutations to the template that appear in the pairwise alignments. Use the fixed batch, if available. If no reference was provided, stop after this step.
- ii. Frame correction stage: Use the full model, including the reference and reads. Propose indel candidate mutations that appear in the template-reference alignment. Increasingly penalize single indels in alignment of r and c . Use the fixed batch, if available.
- iii. Refinement stage: Propose only substitutions (no indels) to the template that appear in the pairwise alignments, no longer considering the reference. Use random batches, with decreasing ρ .

The initial stage quickly finds a good candidate template from the reads alone. The frame correction stage uses a reference to penalize indels that cause frame shift errors, correcting the reading frame of the template in a way that is maximally compatible with both the reads and the reference. Finally, the refinement stage ensures that the reference influences only the frame of c , and exerts no bias upon the nucleotides themselves. The final stage also fixes biases introduced by the fixed batch in the first two stages. We take c from the

final optimization stage to be $\hat{t}$, our estimate of the unobserved template.

3 Results

We compared RIFRAF to two other methods: MAFFT (Katoh and Standley, 2013) followed by the standard per-column consensus, and POA (Lee et al., 2002) with the heaviest bundle consensus algorithm (Lee, 2003). All three methods were run with and without reference-guided frame correction. RIFRAF natively performs frame correction, but only if it is given a reference sequence. To distinguish these models in this section, we refer to the model with no reference as RIFRAF_{nr}, and the model with a reference as RIFRAF_{ref}. FrameBot (Wang et al., 2013) was used for correcting results from MAFFT and POA; these are referred to as MAFFT_FB and POA_FB.

A full-length sequencing run of Pacific Biosciences SMRT sequencing on *env* from HIV-1 subtype B strain NL4-3 was used for the comparison (Laird Smith et al., 2016). The true sequence of NL4-3 is known, so results could be compared to the ground truth. The filtered data [available on FigShare (<https://doi.org/10.6084/m9.figshare.5643247>)] contains 27 600 reads with expected error rate 1% or better, which were further filtered and processed as follows. To make the problem more challenging and better reveal differences between methods, very high quality sequences were excluded (expected error rate < 0.1%). Short fragments and long reads (often concatemers) were discarded by filtering out sequences 25 bases shorter or longer than the median of 2597. PacBio reads come in random orientations, so reads were converted to their reverse complement, if necessary. Extra bases around the amplicon were removed by aligning to NL4-3 *env* without penalizing terminal gaps, then trimming terminal insertions. After preprocessing, 9473 sequences remained, with a mean error rate of 0.0015 (the distribution of errors appears in Supplementary Fig. S1). All experiments were run for 1000 trials on randomly sampled reads.

Choice of reference. A set of reference sequences—shown in Supplementary Figure S2—were tested to investigate how frame correction accuracy deteriorates for distantly-related references. Note that HIV *env* has extensive genuine indel variation in its variable regions (Korber et al., 2001), and the alignment between the template and the four distantly-related heterologous reference sequences contain regions with gaps and with low sequence identity. The results are shown in Figure 4a and b. Nucleotide results from MAFFT_FB and POA_FB were both equally insensitive to the choice of reference, whereas RIFRAF_{ref}'s results did degrade slightly. However, the reverse is true for the protein sequences, with RIFRAF_{ref}'s performance degrading by half an amino acid on average, and the others degrading by more than one. This difference indicates that RIFRAF_{ref} not only keeps the template in-frame, but also makes better choices of inferring which nucleotides are truly indel errors. Finally, RIFRAF_{ref} was the most accurate, regardless of choice of reference. As expected, RIFRAF_{ref}'s frame correction strategy works best with a closely related reference, but these results show that it is capable of working even with a distant reference. Except where noted, the most distantly related reference, B.BR, was used for the rest of the results, so our accuracy estimates should be interpreted as conservative.

Number of sequences. Clusters of 2, 3, 4, 5, 6, 8, 10, 15 and 20 reads were randomly sampled for this experiment. The fraction of perfectly reconstructed template sequences per 1000 trials appears

in Figure 4c. For fewer than 10 sequences, both versions of RIFRAF dominate the other corresponding methods. For instance, RIFRAF_{ref} gets over 90% correct with access to only four reads. POA_FB does not achieve similar results until $N = 8$, and MAFFT_FB does not until between $N = 10$ and 15. Interestingly, POA's results actually degrade significantly for $n > 6$, but POA_FB continues to improve, because POA tends to include extra bases on the ends of the consensus sequence which are then removed by FrameBot. These extra bases also affect the average number of nucleotide errors

(Fig. 4d): for $N = 20$, POA averages one error per sequence, whereas all the other methods average none.

The average number of protein errors (Fig. 4e) highlights the importance of frame correction. Frame shifts cause the translated inferred template sequences to differ greatly from the true protein sequence, especially for $n < 15$. For $N = 2$, fully half of each protein sequence is wrong on average, regardless of method. Even for $N = 20$, sequences from RIFRAF_{nr} and POA contain about 100 errors. On the other hand, the corrected sequences (shown in Fig. 4f

Fig. 4. (a, b) Edit distance for increasingly distant references, run with $N = 3$ full-length reads. All methods do better with closely-related reference, but their rate of performance degradation is important because a related reference may not always be available. (c) Fraction of correct sequences and (d) mean edit distance between the inferred and true template for increasing read counts. (e, f) are from the same experiment as (c, d), but examine translated protein sequences. (e) show results for all methods, and (f) shows only frame-corrected results. Note Y axis scale. Finally, mean execution time, (g) varying by number of reads and (h) varying by read length. Note that intervals on the x-axis are not linear

Fig. 5. ROC curves for true indel experiments, with max indel penalty multiplier m varying from 0 to 12. The orange point denotes results from RIFRAF_{nx}, while the remainder of the curve was generated by RIFRAF_{ref}. The green point corresponds to a max indel penalty multiplier $m=6$. Both a related reference (HXB2, blue) and distant reference (B.BR, red) were used. (a) $N=3$, (b) $N=5$ and (c) $N=10$

for clarity) contain nearly no errors for $n > 10$. RIFRAF_{ref} again performs best here, approaching zero errors even for $N=3$.

Interestingly, frame correction of MAFFT and POA often made the nucleotide sequences *less* accurate, whereas it improves RIFRAF_{ref}. This result supports the idea that RIFRAF's method of integrating frame correction into the template inference algorithm makes it more accurate by allowing all reads to inform the correction process. FrameBot, which only has access to a single consensus sequence, cannot use the extra information in the reads and therefore cannot achieve the same accuracy.

Execution times appear in Figure 4g. Without frame correction, all three methods are comparable for small numbers of sequences, but RIFRAF_{nx} scales better, due to its batching scheme. Frame correction adds a constant factor to all three methods' execution times. RIFRAF_{ref}'s constant factor is larger, but, because it scales better, it overtakes the others between $N=10$ and $N=15$.

Sequence length. Figure 4h also shows execution time for varying sequence lengths. For more details on this experimental setup, see Supplementary Section 2. RIFRAF_{ref} scales less well than the other methods, taking about twice as long as MAFFT_FB and POA_FB for the full-length sequences. However, it is comparable with the others at $\ell=900$, and faster than the others for $\ell < 600$. This difference in speed is due to RIFRAF's iterative approach, which requires recomputing parts of each pairwise alignment after every iteration.

Detecting true frameshifts. In the other experiments, strict frameshift penalties were used to ensure the template stays in-frame. However, sometimes frameshifts are biologically plausible, such as in integrated (but non-functional) proviral Env sequences, or in the cytoplasmic tail of Env leading to a truncation, but preserving infectivity. If true frameshifts may occur in the template sequence, it may be preferable to relax this frameshift penalty. RIFRAF_{ref} can be tuned to accept frameshift indels with enough support in the reads, with only a small increase in the frequency of spurious frameshift indels. To demonstrate this, single base insertions and deletions were added to NL4-3 in both homopolymer and non-homopolymer regions (details in SI Section 3). w_{indel} was set to 1.05, and the max frameshift indel penalty multiplier m varied from 0 to 12. We call an in-frame sequence a 'positive', so increasing m increases the false positive rate by forcing sequences with real frameshifts incorrectly into frame. To get the true positive rate, RIFRAF_{ref} was also run on the unmodified sequences. Note that while we introduce only a single true indel into our 'negative' cases, the analysis is always at the whole-sequence level. We are not just detecting the presence or

absence of the specific indel we introduce. Thus to achieve a high true positive and low false positive rate, RIFRAF_{ref} must successfully ignore spurious indels at *any* position in the 'positive' cases, while successfully identifying the real indel we introduce in each 'negative' case. The resulting ROC curves, which appear in Figure 5 for $N \in 3, 5, 10$, show that RIFRAF_{ref} can find true indels while controlling the false positive rate, using either a closely related reference or a distant one. A useful trade-off occurs for $m=6$, which scores close to the maximum true positive rate while keeping the false positive rate close to zero.

In agreement with the accuracy results, a more closely related reference (HXB2) improved inference for $N=3$ for this task. As expected, real homopolymer indels in homopolymer regions are harder to discriminate than non-homopolymer indels (See Supplementary Section 3 for more detailed results).

4 Conclusion

RIFRAF uses quality scores and a reference sequence to infer accurate frame-corrected template sequences. It can often find the correct template, even from small numbers of reads or with a distant reference, as shown in our experimental results. RIFRAF with frame correction can be slower than inferring a template by taking a consensus from a MSA, but in experiments with real SMRT sequences it infers template sequences that are significantly more accurate. The benefits of using a reference to reduce frameshift errors are especially apparent when comparing translated amino acid sequences, where a single frameshift causes the entire downstream sequence to be incorrect. Finally, RIFRAF can detect and retain true frameshifts in the template relative to the reference. To our knowledge, it is the only method capable of keeping such frameshifts while correcting others.

While RIFRAF performs well with distantly related reference sequences, performance is improved when using closely related references. However, when sequencing diverse populations, we note that it is always possible to first infer a set of autologous sequences from clusters or primer ID bundles that have a large number of reads and so should be accurate. These can then be used as references to correct the reading frame of the less-represented members of the population, providing an improved accuracy over just using a more distantly related reference. We recommend using this strategy whenever possible.

RIFRAF can improve the ability to resolve minority variants in sequenced populations. Its ability to find results comparable to MAFFT with three times fewer reads will be essential for identifying

minority variants in the population with greater precision. More generally, RIFRAF will be useful whenever an accurate template sequence must be inferred from a small number of full-length sequences, especially when quality scores and a reference sequence are available.

When sequencing any population, it is often advisable to sequence a clonal representative of that population first (NL4-3 *env* here), to investigate the sequencing performance for that case. We recommend using such sequence datasets to investigate the behavior of RIFRAF on new genes, especially if the user seeks to detect real frameshifts. To this end, we provide a Jupyter notebook that allows one to replicate the accuracy and ROC analyses from this manuscript on any clonal amplicon dataset.

RIFRAF will continue to be developed along multiple lines. First, the current approach for performing frame correction needs to be faster, to keep pace with the increasing volume of available sequence data. Further work needs to be done to speed it up via optimization or algorithmic advances. Possible approaches include: re-using partial alignments, speeding up alignments with *k*-mer seeding and only correcting the frame of obviously problematic regions.

One promising area of improvement would be to construct a version of RIFRAF that is a fully-specified probabilistic model. This could include an explicit generative model of the quality scores themselves, which would be difficult to handle in the current HMM framework. The benefits of this would be that key parameters, such as the frameshift indel penalty and the codon move probabilities, could be learned from the datasets themselves, rather than being specified *a priori*. This would likely facilitate another useful feature, which would be the calculation of calibrated quality scores for the template sequence, in order to communicate remaining regions of uncertainty to the user.

Another improvement would include amino acid matching penalties in the reference-to-template alignment, which would allow even more distantly related reference sequences to be used, where the nucleotide homology has been completely obliterated. The ability to use a reference model that combines multiple in-frame reference sequences (as is standard for profile HMMs) may also be useful for some applications, using the site-wise distribution of amino acids to inform the reading frame of the template. Finally, RIFRAF is extensible to other systems and sequencing technologies. In particular, we plan to investigate its behavior and tune its error model for Oxford Nanopore Technologies (ONT) data, and to work on supporting hybrid sequencing runs such as SMRT sequencing plus ONT, or hybrid SMRT sequencing with short Illumina reads (that have much lower indel error rates, and could be helpful for frame correction). The method may also be extended to support amplicons containing both non-coding and coding regions, which may contain different (potentially overlapping) reading frames.

The RIFRAF source code is available at <https://github.com/MurrellGroup/Rifraf.jl>.

Acknowledgement

The authors are grateful to Sasha Murrell for copy editing.

Funding

This work was supported by the National Institute Of Allergy And Infectious Diseases of the National Institutes of Health under Award Numbers R01AI120851, R01AI120009, and UM1AI068618, the National Institute on

Drug Abuse under Award Number R21DA041007, and in part by the University of California, San Diego Center for AIDS Research (P30AI036214). K.E. was supported in part by R21AI115701. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health.

Conflict of Interest: none declared.

References

- Bezanson,J. *et al.* (2017) Julia: a fresh approach to numerical computing. *SIAM Rev.*, **59**, 65–98.
- Chao,K.M. *et al.* (1992) Aligning two sequences within a specified diagonal band. *Bioinformatics*, **8**, 481–487.
- Chao,K.M. *et al.* (1993) Constrained sequence alignment. *Bull. Math. Biol.*, **55**, 503–524.
- Chin,C.-S. *et al.* (2013) Nonhybrid, finished microbial genome assemblies from long-read SMRT sequencing data. *Nat Methods*, **10**, 563–569.
- Du,N. and Sun,Y. (2016) Improve homology search sensitivity of PacBio data by correcting frameshifts. *Bioinformatics*, **32**, i529–i537.
- Durbin,R. *et al.* (1998) *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge university press.
- Eid,J. *et al.* (2009) Real-time DNA sequencing from single polymerase molecules. *Science*, **323**, 133–138.
- Jabara,C.B. *et al.* (2011) Accurate sampling and deep sequencing of the hiv-1 protease gene using a primer id. *Proc. Natl. Acad. Sci.*, **108**, 20166–20171.
- Katoh,K. and Standley,D.M. (2013) MAFFT multiple sequence alignment software version 7: improvements in performance and usability. *Mol. Biol. Evol.*, **30**, 772–780.
- Katoh,K. *et al.* (2002) MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform. *Nucleic Acids Res.*, **30**, 3059–3066.
- Korber,B. *et al.* (2001) Evolutionary and immunological implications of contemporary hiv-1 variation. *Br. Med. Bull.*, **58**, 19–42.
- Laird Smith,M. *et al.* (2016) Rapid sequencing of complete env genes from primary HIV-1 samples. *Virus Evol.*, **2**, vew018.
- Lee,C. (2003) Generating consensus sequences from partial order multiple sequence alignment graphs. *Bioinformatics*, **19**, 999–1008.
- Lee,C. *et al.* (2002) Multiple sequence alignment using partial order graphs. *Bioinformatics*, **18**, 452–464.
- Loman,N.J. *et al.* (2015) A complete bacterial genome assembled de novo using only nanopore sequencing data. *Nat. Methods*, **12**, 733–735.
- Nagarajan,N. and Pop,M. (2013) Sequence assembly demystified. *Nat. Rev. Genet.*, **14**, 157–167.
- Needleman,S.B. and Wunsch,C.D. (1970) A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol.*, **48**, 443–453.
- Pal,S.K. *et al.* (2006) Evolutionary computation in bioinformatics: a review. *IEEE Trans. Syst. Man Cybernetics Part C: Appl. Rev.*, **36**, 601–615.
- Paszkiwicz,K. and Studholme,D.J. (2010) De novo assembly of short sequence reads. *Brief. Bioinformatics*, **11**, 457–472.
- Pervez,B.M. *et al.* (2014) Evaluating the accuracy and efficiency of multiple sequence alignment methods. *Evol. Bioinformatics*, **10**, 205–217.
- Rice,P. *et al.* (2000) Emboss: the European molecular biology open software suite. *Trends Genet.*, **6**, 276–277.
- Sheetlin,S.L. *et al.* (2014) Frameshift alignment: statistics and post-genomic applications. *Bioinformatics*, **30**, 3575–3582.
- Sheward,D.J. *et al.* (2012) Degenerate primer ids and the birthday problem. *Proc. Natl. Acad. Sci.*, **109**, E1330–E1330.
- Wang,Q. *et al.* (2013) Ecological patterns of nifH genes in four terrestrial climatic zones explored with targeted metagenomics using framebot, a new informatics tool. *mBio*, **4**, e00592-13–e00592-13.
- Zhang,Y. and Sun,Y. (2011) HMM-FRAME: accurate protein domain classification for metagenomic sequences containing frameshift errors. *BMC Bioinformatics*, **12**, 198.