Supplementary Document

Supplementary Document A: Proof of Convergence of the Alternating Minimization with Majority Voting

Recall the objective function of the optimization problem,

$$f(\mathbf{M}, \mathbf{H}) = \min_{\mathbf{M},\mathbf{H}} \frac{1}{2} \|\mathcal{P}_\Omega(\mathbf{F} - \mathbf{M}\mathbf{H}^\top)\|_F^2.$$

Given the current estimate $\mathbf{M}_t$ and $\mathbf{H}_t$, we alternately update

$$\mathbf{M}_{t+1} = \arg \min_{\mathbf{M} \in \{0,1\}^{m \times k}} \frac{1}{2} \|\mathcal{P}_\Omega(\mathbf{F} - \mathbf{M}\mathbf{H}_t^\top)\|_F^2$$

and

$$\mathbf{H}_{t+1} = \arg \min_{\mathbf{H} \in \{0,1\}^{4n \times k}} \frac{1}{2} \|\mathcal{P}_\Omega(\mathbf{F} - \mathbf{M}_{t+1}\mathbf{H}^\top)\|_F^2.$$

Having fixed $\mathbf{H}_t$, $\mathbf{M}_{t+1}$ is updated by examining for each read all the possible viral haplotype associations and selecting the one that minimizes the number of base changes needed to make reads consistent with the observed information in $\mathbf{F}$, which implies that

$$f(\mathbf{M}_{t+1}, \mathbf{H}_t) - f(\mathbf{M}_t, \mathbf{H}_t) \leq 0. \tag{1}$$

Now let us divide $\mathbf{F}$ into k sub-matrices $\mathbf{F}^i$, $1 \leq i \leq k$, each representing the collection of reads assigned to the i^{th} haplotype in $\mathbf{M}_{t+1}$, and rewrite $f(\mathbf{M}_{t+1}, \mathbf{H}_t)$ as

$$f(\mathbf{M}_{t+1}, \mathbf{H}_t) = \frac{1}{2} \sum_{i=1}^{k} \sum_{j=1}^{n} \|\mathcal{P}_{\Omega_{\mathbf{F}^i(\cdot,j)}}(\mathbf{F}^i(\cdot, j) - \mathbf{M}_{t+1}^i \mathbf{H}_t(j, \cdot)^\top)\|_2^2,$$

where $\mathbf{F}^{(i)}(\cdot, j)$ denotes the j^{th} column of $\mathbf{F}^{(i)}$, $\mathbf{M}_{t+1}^i$ has for rows the standard unit vectors $\mathbf{e}_i$, and $\mathbf{H}_t(j, \cdot)$ is the j^{th} row of $\mathbf{H}_t$ at the t^{th} iteration. Since $\mathbf{H}_{t+1}$ is updated by forming consensus sequences using reads in each sub-matrix clustered by $\mathbf{M}_{t+1}$, we obtain

$$\begin{aligned}
&f(\mathbf{M}_{t+1},\mathbf{H}_{t+1}) - f(\mathbf{M}_{t+1},\mathbf{H}_t)\\
&= \frac{1}{2}\sum_{i=1}^{k}\sum_{j=1}^{n} \|\mathcal{P}_{\Omega_{\mathbf{F}^i(\cdot,j)}}(\mathbf{F}^i(\cdot,j) - \mathbf{M}^i_{t+1}\mathbf{H}_{t+1}(j,\cdot)^\top)\|_2^2\\
&\quad - \frac{1}{2}\sum_{i=1}^{k}\sum_{j=1}^{n} \|\mathcal{P}_{\Omega_{\mathbf{F}^i(\cdot,j)}}(\mathbf{F}^i(\cdot,j) - \mathbf{M}^i_{t+1}\mathbf{H}_{t}(j,\cdot)^\top)\|_2^2\\
&= \frac{1}{2}\sum_{i=1}^{k}\sum_{j=1}^{n}((N - N_{\mathbf{H}_{t+1}(j,i)}) - (N - N_{\mathbf{H}_t(j,i)}))\\
&= \frac{1}{2}\sum_{i=1}^{k}\sum_{j=1}^{n}(N_{\mathbf{H}_t(j,i)} - N_{\mathbf{H}_{t+1}(j,i)}) \le 0
\end{aligned} \tag{2}$$

where N denotes the total number of the observed nucleotides in $\mathbf{F}^i(\cdot,j)$ and $N_{\mathbf{H}_t(j,i)}$ is the number of nucleotides corresponding to $\mathbf{H}_t^{(j,i)}$. The combination of expressions (1) and (2) shows that the proposed alternating minimization procedure is guaranteed to converge.

Supplementary Document B: Additional results

(1) Comparing speed of tensor factorization that relies on majority voting with the one that relies on gradient descent

Table S1: Runtime comparison of majority voting and gradient descent.

		div%									
k	method	1	2	3	4	5	6	7	8	9	10
5	MV	1.00	1.50	1.83	2.64	3.63	3.51	4.17	4.49	4.59	5.95
	GD	31.84	42.40	39.60	78.60	66.04	105.55	93.87	112.49	116.83	202.12
10	MV	3.93	8.75	13.13	19.62	24.96	25.03	28.73	33.51	38.32	36.20
	GD	233.88	321.14	320.79	374.16	426.62	535.04	513.29	752.01	604.85	678.27

Running time comparisons (sec) of majority voting (MV) and gradient descent (GD) for tensor factorization on the simulated data with $\varepsilon = 2 \times 10^{-3}$ and $cov = 500\times$ for a mixture of 5 and 10 strains vs diversity (*div*), measured on a Linux OS servers with Intel Xeon Phi 7250 (1.4GHz) and 96GB RAM.

In Table S2, we compare runtimes of majority voting and gradient descent applied to solving equation (3) in Section 2.2 of the main paper. The dataset is the same as the one used in Section 3.1: mixtures of 5 and 10 strains are sequenced with error rate $\varepsilon = 2 \times 10^{-3}$ and coverage $cov = 500\times$. The speed is measured on a Linux OS server with Intel Xeon Phi 7250 (1.4GHz) and 96GB RAM. As evident from the table, tensor factorization implemented with majority voting is much more time-efficient than the gradient-descent based approach for all strain diversity levels.

(2) Comparing accuracy of viral quasispecies reconstruction based on single-pass tensor factorization (AltHap) with the one that employs multiple tensor factorizations and read removal (TenSQR)

We compare the performance of TenSQR and AltHap (**?**) on the simulated dataset consisting of 5 viral strains whose synthesis is described in Section 3.1. We measure the performances of two schemes in terms of *Recall*, *Precision*, *Predicted Proportion* (*PredProp*), *Reconstruction Rate* (*ReconRate*) and *JSD*. Since AltHap requires the number of haplotypes a priori, we do not report *Predicted Proportion.* As shown in Figure S1, TenSQR significantly outperforms AltHap. This, of course, is expected: AltHap is designed for reconstruction of sequence communities characterized by uniform abundances – an assumption violated in viral quasispecies.

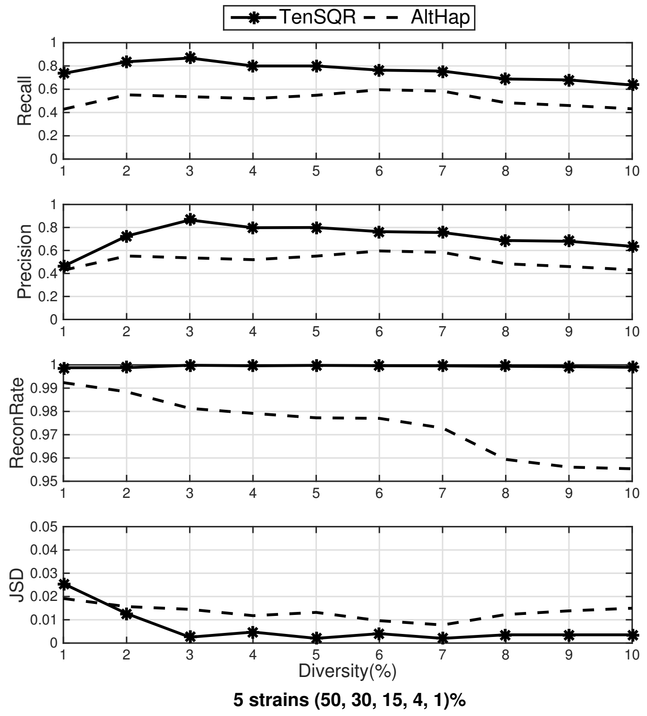

Figure S1: Performance comparison of TenSQR and AltHap in terms of *Recall*, *Precision*, *Reconstruction Rate* (*ReconRate*) and *JSD* on the simulated data with $\varepsilon = 2 \times 10^{-3}$ for a mixture of 5 viral strains.

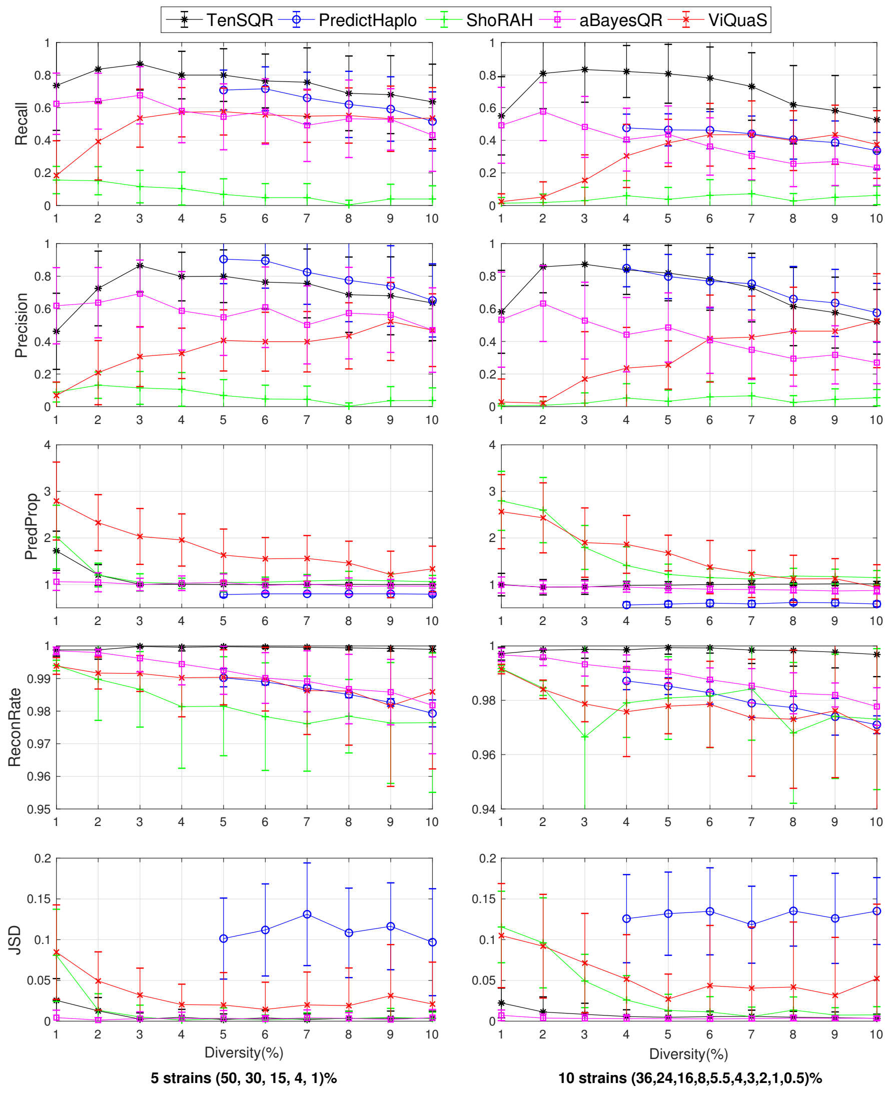

Figure S2: Performance comparison of TenSQR, aBayesQR, ShoRAH, ViQuaS and PredictHaplo in terms of *Recall*, *Precision*, *Predicted Proportion* (*PredProp*), *Reconstruction Rate* (*ReconRate*) and *JSD* on the simulated data with $\varepsilon = 2 \times 10^{-3}$ for a mixture of (a) 5 viral strains and (b) 10 viral strains.

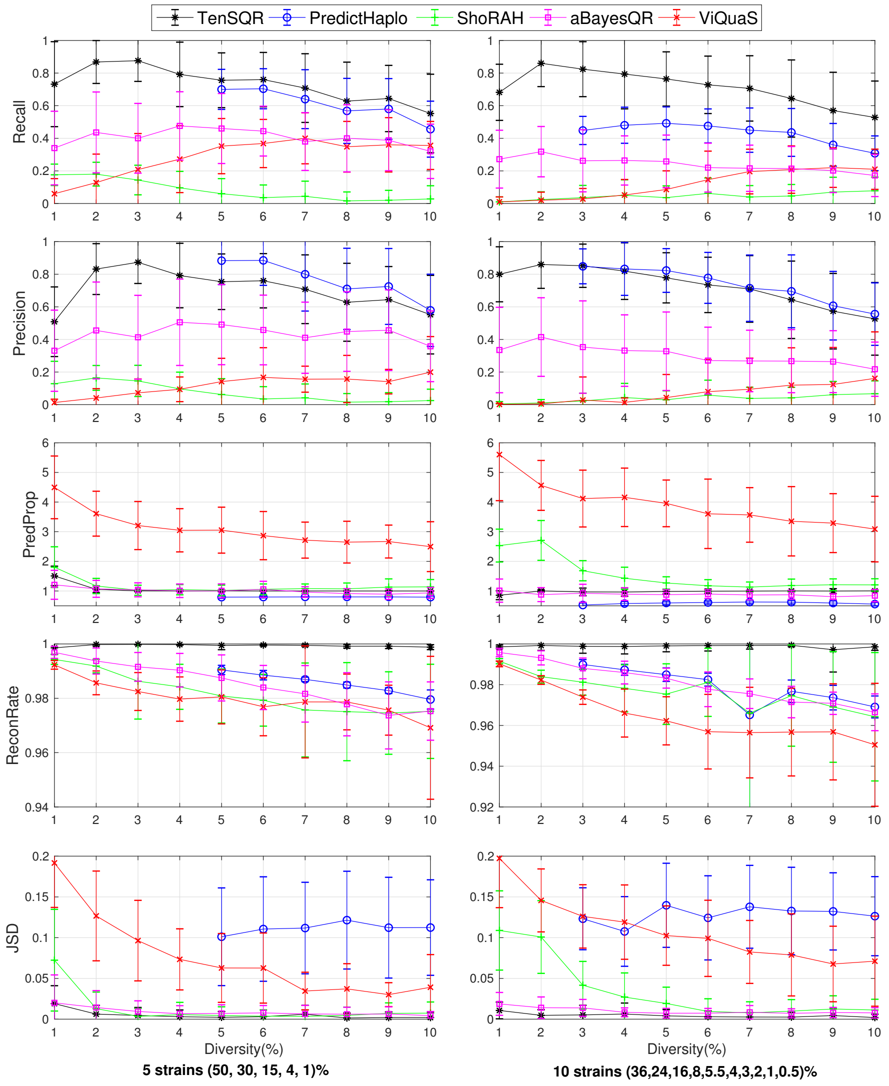

Figure S3: Performance comparison of TenSQR, aBayesQR, ShoRAH, ViQuaS and PredictHaplo in terms of *Recall*, *Precision*, *Predicted Proportion* (*PredProp*), *Reconstruction Rate* (*ReconRate*) and *JSD* on the simulated data with $\varepsilon = 7 \times 10^{-3}$ for a mixture of (a) 5 viral strains and (b) 10 viral strains.

Table S2: Performance comparisons of TenSQR, aBayesQR and PredictHap on a real HIV-1 5-virus-mix data.

		p17	p24	p2-p6	PR	RT	RNase	int	vif	vpr	vpu	gp120	gp41	nef
TenSQR	PredProp	1	1.6	1	1	1.4	1	1	1	1	1.6	2.2	1.2	0.8
	RR_{HXB2}	100(18.7)	98.9(13.1)	100(17.4)	100(9.9)	99.2(12.1)	100(9.2)	100(8.1)	100(9.6)	100(7.2)	92.8(5.9)	96.0(18)	99.0(11.5)	0(0)
	$RR_{89.6}$	100(18.4)	100(19.6)	100(20.1)	100 (17.2)	98.0(13.5)	100(17.2)	100(16.7)	100(25)	100(19.3)	94.0(15)	97.2(10.3)	100(27.8)	95.7(26)
	$RR_{JR\text{-}CSF}$	100(33.8)	100(33)	100(33.6)	100(21.7)	100(20.7)	100(24.6)	100(23.3)	100(20.5)	100(20.3)	100(31.4)	98.3(33.5)	97.7(18.8)	99.8(19)
	$RR_{NL4\text{-}3}$	100(17)	99.3(19.7)	100(17.2)	100(21.4)	99.5(26.7)	100(37.7)	100(41.2)	100(38.4)	100(46.2)	100(38.8)	99.8(9.2)	99.5(23.2)	99.7(42.7)
	RR_{YU2}	100(12.1)	99.3(14.6)	100(7.7)	99.7(29.8)	99.7(14.5)	100(11.4)	100(10.7)	100(6.5)	100(7.1)	100(4.1)	94.9(10.5)	100(10.2)	98.6(12.3)
aBayesQR	PredProp	1	1	1	1	1	1	1	1	1.2	1	0.8	0.8	1.2
	$RR(F)_{HXB2}$	100(16.3)	99.4(21.1)	100(22.2)	100(12.5)	98.5(24.3)	100(16.1)	99.9(9.7)	100(9.2)	100(16.4)	99.6(17)	98(30.3)	0(0)	95.8(11.4)
	$RR(F)_{89.6}$	100(27.1)	98.7(17)	100(17.3)	100(17.3)	98.6(18.1)	100(19.7)	100(22.2)	100(20.6)	100(16.3)	92(10.4)	96.5(20.2)	98.9(23.7)	95.5(16.4)
	$RR(F)_{JR\text{-}CSF}$	100(31.3)	99.6(24.6)	100(25.8)	100(29.9)	99(21.5)	100(22.1)	100(20.8)	100(32.7)	100(27)	98.8(26.7)	97.7(21.4)	99.1(29.7)	98.2(21.1)
	$RR(F)_{NL4\text{-}3}$	100(12.9)	100(21.6)	100(25.6)	100(20.1)	98.9(17.7)	100(30)	100(39.5)	99.8(28.5)	100(23.2)	100(41.3)	96.3(28)	98.8(36.6)	100(31.8)
	$RR(F)_{YU2}$	100(12.4)	99.7(15.8)	100(9.2)	100 (20.3)	99.2 (18.5)	100(12.2)	99.5(7.9)	99.7(9)	100(17.1)	100(4.6)	0(0)	98.6(10.1)	99.2(14)
PredictHaplo	PredProp	1	0.6	1	1	1	0.8	0.8	0.8	1	0.8	0.8	0.8	0.8
	$RR(F)_{HXB2}$	100(17.8)	0(0)	100(18.7)	100(15.2)	100(12.2)	98.9(25.4)	100(12.1)	100(17.7)	100(10.2)	93.17(10.8)	0(0)	0(0)	0(0)
	$RR(F)_{89.6}$	100(19.9)	100(46.4)	100(21.7)	100(22.2)	100(19.4)	100(18.2)	99.8(27.6)	100(20.9)	100(22.1)	0(0)	97.8(20.7)	100(26.7)	98.87(20.7)
	$RR(F)_{JR\text{-}CSF}$	100(31.9)	100(21.8)	100(30.3)	100(26.9)	100(23.4)	100(23.2)	100(22.3)	100(24.9)	100(23.7)	100(34.1)	99.7(42.7)	100(28.9)	100(23.2)
	$RR(F)_{NL4\text{-}3}$	100(17)	99.1(31.8)	100(16.4)	100(20.9)	100(30.2)	100(33.2)	100(38.1)	100(36.6)	100(35.5)	100(47.1)	100(28.6)	100(32.7)	100(39.3)
	$RR(F)_{YU2}$	100(13.4)	0(0)	100(12.9)	100(14.8)	100(14.7)	0(0)	0(0)	0(0)	100(8.5)	100(7.9)	98.6(7.9)	100(11.7)	100(16.9)

Predicted Proportion (PredProp) and Reconstruction Rate (RR (%)) for TenSQR, aBayesQR and PredictHaplo applied to reconstruction of HIV-1HXB2, HIV-189.6, HIV-1JR-CSF, HIV-1NL4-3 and HIV-1YU2 for all 13 genes of the HIV-1 dataset. Frequencies are reported in parenthesis.

Supplementary Document C: Estimating insertions

In addition to deletions, strains of a quasispecies may also contain insertions as compared to a reference genome. Here we propose a pipeline to detect and reconstruct insertions by processing paired-end reads that were previously filtered out due to inability to match them confidently with the reference genome; therefore, those reads have not been used in tensor factorization. The idea is that some of the discarded reads that have high base-calling quality scores might have not been matched successfully to the reference because they originate from a region that was inserted into one of the strains of the quasispecies.

The three major steps of the proposed procedure are the following:

1. Infer the origin of the filtered-out paired-end reads having high base-calling quality scores.

Let us denote two sequences in a paired-end read as (r_h, r_l) where the first one is a read having high mapping score and the other one a read which is not mapped or is only partially aligned to the reference (i.e., has a low mapping score) due to an insertion. In order to infer the origin of the paired-end read (r_h, r_l), we align the mapped read r_h to the position identified by the alignment software and count the number of mismatches between the read r_h and each of k reconstructed strains. The strain q_I having the smallest Hamming distance to the paired-end read is inferred as the origin of the paired-end read (r_h, r_l) and thus contains an insertion.

2. Alignment of r_l

Having identified strain q_I as the origin of the paired-end read (r_h, r_l), we next attempt to find the most likely position of r_l relative to q_I. To this end, we examine subsequences that consist of the first and last l nucleotides of r_l and compare them to every l-nucleotides long subsequence of q_I. Note that we assume $l < L_{r_l}$, where L_{r_l} denotes the length of read r_l. To identify the best placement of r_l along q_I, we measure alignment scores for $2(L_g - l)$ windows, where L_g is the length of q_I, and choose the one with the largest score. More specifically, the alignment score is updated for each window at each position of $1 \leq i \leq l$; $score_{i+1} \leftarrow score_i + N_{match}$ if the i^{th} nucleotide of the l-nucleotides long subsequences is matched with one in q_I, otherwise the score is updated as $score_{i+1} \leftarrow score_i - (N_{mismatch} + 1)^2$ where N_{match} and $N_{mismatch}$ are the number of consecutive matches and mismatches, respectively; the scoring function penalizes consecutive mismatches while favoring consecutive matches to form a reliable estimate of the placement for r_l.

Given a relative position of r_l along q_I, we infer the inserted position (i.e., the position at which r_l starts to differ from q_I due to insertions) by relying on the same scoring function used to determine the best alignment position. In particular, the score is computed for each position from 1 to L_{r_I} for reads whose l-nucleotides-long prefix maps to q_I and from L_{r_I} to i for those whose l-nucleotides-long suffix maps to q_I; then the position at which insertion starts is estimated as $I + 1$, where $I = \arg\max_i score_i$. For computational efficiency, the updates of $score_i$ are terminated once $score_i < 0$.

Table S3: Performance of recovering insertions.

		$div\%$									
l_{ins}	$f_{min}\%$	1	2	3	4	5	6	7	8	9	10
100	5	0.28	0.12	0.08	0.06	0.12	0.10	0.02	0.10	0.04	0.18
		(0.95)	(0.98)	(0.99)	(0.99)	(0.99)	(0.99)	(0.99)	(0.99)	(0.99)	(0.99)
	10	0.14	0.08	0.10	0.04	0.04	0.04	0.04	0.02	0.02	0
		(0.98)	(0.99)	(0.99)	(0.97)	(0.98)	(0.98)	(0.99)	(0.99)	(0.98)	(0.99)
200	5	0.18	0.20	0.14	0.06	0.12	0.06	0.12	0.04	0.14	0.08
		(0.99)	(0.99)	(0.99)	(0.99)	(0.99)	(0.99)	(0.99)	(0.99)	(0.99)	(0.99)
	10	0.12	0.10	0.02	0.06	0	0.06	0.04	0.02	0.02	0
		(1)	(1)	(0.99)	(1)	(1)	(0.99)	(0.99)	(0.99)	(0.99)	(0.99)

Performance of insertions recovery in terms of the ratio of *False Negative* and *Reconstruction Rate* of insertion (in parentheses) averaged over 50 instances on the simulated dataset with $\varepsilon = 2 \times 10^{-3}$ and $cov = 500\times$ for a mixture of 2 strains, as a function of the diversity(div) and the frequency of the low abundant strain (f_{min}) which contains an insertion of the length (l_{ins}) 100bp and 200bp.

3. Recover inserted sequences

Given the alignment and the position at which the insertion starts, recovery of the insertion is readily performed by constructing two consensus sequences – one built using the collection of reads whose l-nucleotides-long prefix maps to q_I and the other whose l-nucleotides-long suffix maps to q_I. Finally, the two consensus sequences are aligned and merged to recover the entire insertion.

To study insertions recovery, we generated data that emulate the same sequencing platform considered in Sections 3.1 and 3.2 of the paper. Sets of quasispecies consisting of two strains where the length of the abundant strain is 1300bp while the other strain (present at frequencies f_{min} of 5 and 10%) includes insertions of length $l_{ins} = 100$ and 200bp. 50 instances are generated for each of 40 benchmark sets of sequencing reads. Strains in the quasispecies have diversity that varies from 1% to 10%; sequencing is performed at the coverage of 500× per strain and is affected by sequencing errors incurring with probability $\varepsilon = 2 \times 10^{-3}$. Performance of insertion detection is evaluated by means of the false negative rate of the detected insertions and the reconstruction rate of the inserted sequences. Table S3 reports the results of recovering insertions in viral strains. As indicated by the reconstructed rate shown in parentheses, once insertions in the strains are detected, our approach is able to accurately reconstruct the inserted sequences.