De novo assembly of viral quasispecies using overlap graphs - Supplemental Methods

Jasmijn A. Baaijens[1], Amal Zine El Aabidine[2], Eric Rivals[2,3*,†], Alexander Schönhuth[1,*,†]

[1] Centrum Wiskunde & Informatica, Amsterdam, Netherlands
[2] LIRMM, CNRS and Université de Montpellier, Montpellier, France
[3] Institut Biologie Computationnelle, CNRS and Université de Montpellier
[*] These authors contributed equally
[†] Corresponding authors

rivals@lirmm.fr, alexander.schoenhuth@cwi.nl

Contents

1 Synthetic benchmarks **2**
1.1 Human immunodeficiency virus 2
1.2 Hepatitis C virus 2
1.3 Zika virus 2

2 Overlap score computation **3**

3 Contig formation and error correction **3**

4 Command lines used for benchmarking **4**

5 Supplemental Code **6**

6 Supplemental References **6**

1 Synthetic benchmarks

For benchmarking experiments we constructed several synthetic data sets using the simulation software SimSeq (https://github.com/jstjohn/SimSeq). We simulated 2x250 bp paired-end reads, with a fragment size of 450 bp and the MiSeq error profile provided with the software. For each benchmark, we selected several viral genomes from the NCBI nucleotide database. We simulated sequencing reads from each of these strains and varied the relative abundance per strain by varying the number of reads simulated. The original strains and relative abundance rates used for each benchmark are listed below. All benchmarks and original strains can be found as Supplemental Data 1, or downloaded from https://bitbucket.org/jbaaijens/savage-benchmarks.

1.1 Human immunodeficiency virus

We simulated three data sets based on five human immunodeficiency virus (HIV) strains. For each of these benchmarks we used the same five HIV strains, based on the gold standard benchmark presented in (Di Giallonardo et al., 2014): $HIV_{89.6}$, HIV_{HXB2}, $HIV_{JR\text{-}CSF}$, $HIV_{NL4\text{-}3}$, and HIV_{YU2}. The corresponding sequences were downloaded from https://github.com/cbg-ethz/5-virus-mix/data/5VM.fasta. Pairwise divergence between these strains ranges from 1% to 6%.

The first 5-strain HIV benchmark has a read coverage of 20 000x, with relative abundance rates of 18%, 26%, 23%, 28%, and 5%, respectively, for each of the strains listed above. In addition, we simulated a 5-strain HIV mix identical to the previous one, only with shorter reads (2x150 bp).

Finally, we simulated an small 5-strain HIV mix for training purposes. This data set has only 600x coverage and the strains are homogeneously distributed, i.e., each strain has a relative frequency of 20%.

As a reference genome for the HIV benchmarking experiments, we used the HXB2 reference sequence (accession K03455).

1.2 Hepatitis C virus

We simulated a 20 000x data set based on ten hepatitis C virus (HCV) strains, subtype 1a. These genomes were downloaded from the NCBI nucleotide database with the following accession numbers and simulated at varying relative frequencies: EU155339 (12%), EU155344 (5%), EU234065 (8%), EU255965 (12%), EU255973 (10%), EU255980 (5%), EU255981 (13%), EU255982 (10%), EU255983 (6%), and EU255989 (19%). Pairwise divergence between these strains ranges from 6% to 9%.

As a reference genome for the HCV benchmarking experiments, we used the HCV genotype 1 reference genome (accession NC_004102).

1.3 Zika virus

We simulated two data sets based on Zika virus (ZIKV) strains, each with a coverage of 20 000x. The first data set contains only 3 strains, all of African lineage: one from Uganda (accession HQ234498), one from Nigeria (accession HQ234500), and one from Senegal (accession HQ234501). Pairwise divergence between these three strains varies between 3% and 10%. For the first data set we used relative abundance rates of 24%, 16%, and 60%, respectively.

The second data set 12 new strains, in addition to the 3 strains listed above. For each of the 3 original strains, we generated 4 new strains by randomly introducing point mutations at a rate of 1%, 1%, 2%, and 2%, respectively. This resulted in a total of 15 African lineage ZIKV strains, with a pairwise divergence between 1% and 12%. We simulated reads for each of the 3 original strains at a relative abundance rate of 2%, and for the 4 derivatives of each original strain we used relative abundance rates of 4%, 6%, 8%, and 13.3%, respectively.

As a reference genome for the ZIKV benchmarking experiments, we used the African lineage reference genome (accession NC_012532).

2 Overlap score computation

The overlap score represents the probability that the DNA fragments underlying two sequencing reads are identical on their overlap. By making use of the base calling quality scores, also known as Phred scores, mismatches that are unlikely to be the result of a sequencing error are penalized.

Let R_A, R_B be two single-end reads sequencing reads and let i be the position i on R_A at which the overlap with R_B starts. Furthermore, for any position k of the overlap, let $q_A[k]$ and $q_B[k]$ be the respective probabilities that $R_A[k]$ and $R_B[k]$ were sequenced incorrectly. We assume that if a base was called wrongly, then each of the remaining bases is equally likely. Let $Q_A[k][X]$ be the probability that the underlying DNA sequence of R_A equals base X at position k, for $X \in \{A, C, T, G\}$, then

$$Q_A[k][X] = \begin{cases} 1 - q_A[k] & \text{if } X = R_A[k] \\ q_A[k]/3 & \text{otherwise.} \end{cases} \tag{1}$$

Using $Q_A[k]$, the probability that the true sequences underlying R_A and R_B agree for each position k of their overlap equals

$$P(A, B) = \prod_{k \in [L]} \sum_{X \in \{A,C,T,G\}} Q_A[k][X] \cdot Q_B[k][X], \tag{2}$$

where $L := |R_B| - i$ is the length of the overlap. Since $P(A, B)$ depends on the length of the overlap L, we define the overlap score of R_A and R_B as $\sqrt[L]{P(A, B)}$.

The overlap graph that we construct contains exactly one vertex for every read (single- or paired end). There is an edge between a pair of vertices if the overlap of the corresponding reads scores higher than a given threshold δ. In case of an overlap involving a paired end read, we compute the overlap score separately for each of the read ends, and we draw an edge if and only if both overlaps score sufficiently high.

3 Contig formation and error correction

When merging two or more reads, we determine the resulting consensus sequence and quality scores as described in (Töpfer et al., 2014). Let C be the consensus sequence to be determined from a clique $R_1, \ldots, R_t$ and assume that R_1 is the leftmost read. Then there exist edges $R_1 \to R_2, \ldots, R_1 \to R_t$ and corresponding to these edges we have positions $\mathtt{pos}_2, \ldots, \mathtt{pos}_t$, where $\mathtt{pos}_i$ indicates the position on R_1 where the overlap with read R_i starts. To simplify expressions, we define $\mathtt{pos}_1 = 0$. Then the consensus nucleotide at position k is determined by a weighted position-wise majority vote:

$$C[k] = \operatorname*{argmax}_{x \in \{A,C,T,G\}} \sum_{i=1}^{t} Q_{R_i}[k - \mathtt{pos}_i][X]. \tag{3}$$

The consensus quality score calculation at each position of the read employs similar techniques as the overlap score computations described above. First, the probability p_{agree} that all reads in the clique agree on this position is computed:

$$p_{\text{agree}} = \sum_{X \in \{A,C,T,G\}} \prod_{i=1}^{t} Q_{R_i}[k - \mathtt{pos}_i][X]. \tag{4}$$

Next, we compute the probability p_{correct} that the reads in the clique actually agree that the consensus base $C[k]$ is correct:

$$p_{\text{correct}} = \prod_{i=1}^{t} Q_{R_i}[k - \texttt{pos}_i][C[k]]. \tag{5}$$

If this probability p_{correct} is less than 0.999, i.e., the Phred quality score of the base would be less than 30, we substitute an 'N' nucleotide instead of the consensus base. Finally, we define the new quality score $Q_C[k]$ as the probability that base $C[k]$ is incorrect, given that $R_1, \ldots, R_t$ agree on position k:

$$Q_C[k] = 1 - \frac{p_{\text{correct}}}{p_{\text{agree}}}. \tag{6}$$

Note that the higher the number of reads supporting a given contig position, the higher one expects the quality of the resulting consensus base call to be. Consequently, the extremities of the contigs, which in general have less support than the center of the read, will be of lower quality. Therefore, the algorithm allows running the first iteration in *error correction* mode. In this case, the extremities of the contig are removed, such that all remaining nucleotides are supported by at least a specified number of sub-reads. When applying error correction, all reads that are not part of any contig are removed, since those were not corrected.

4 Command lines used for benchmarking

Read trimming and merging: CutAdapt (Martin, 2011), PEAR (Zhang et al., 2014)
Generic assembly tools: SGA (Simpson and Durbin, 2012), SOAPdenovo2 (Luo et al., 2012), SPAdes (Bankevich et al., 2012), metaSPAdes (Nurk et al., 2016)
Specialized assembly tools: ShoRAH (Zagordi et al., 2011) and PredictHaplo (Prabhakaran et al., 2014), MLEHaplo (Malhotra et al., 2016b)
Ad-hoc reference construction: VICUNA (Yang et al., 2012)
Others: BWA-MEM (Li, 2013), MultiRes (Malhotra et al., 2016a), metaQUAST (Mikheenko et al., 2016)

BWA-MEM *version 0.7.15-r1140*

```
bwa mem reference.fasta reads.fastq
```

MLEHaplo *downloaded from https://github.com/raunaq-m/MLEHaplo at June 21, 2016*

```
perl -Mlocal::lib MLEHaplo.pl
-fas input.fasta
-out contigs.mlehaplo.fasta
-k 55
-MR input.multires.k55.txt
-IS 450
-M 15
-GL 9500
```

MultiRes *downloaded from https://github.com/raunaq-m/MultiRes at June 21, 2016*

```
perl MultiRes.pl -fq input.fastq -k 55 -out input.multires.k55.txt
```

PEAR *version 0.9.6*

```
pear -f forward.fastq -r reverse.fastq -o pear
```

PredictHaplo *version 0.4*

```
PredictHaplo-Paired config.txt
```

PredictHaplo was run using default settings; the corresponding config file can be downloaded from https://bitbucket.org/jbaaijens/savage-benchmarks and as Supplemental Code.

SAVAGE: *version 0.2.1*

```
snakemake -s savage/Snakefile
```

The SAVAGE Snakefile is included in the SAVAGE package, publicly available for download from https://bitbucket.org/jbaaijens/savage. The config files used for each benchmark are available at https://bitbucket.org/jbaaijens/savage-benchmarks and as Supplemental Code.

SGA: *version 0.10.13*

```
sga_default.sh forward.fastq reverse.fastq
```

The SGA pipeline script is available as Supplemental Code and also at https://bitbucket.org/jbaaijens/savage-benchmarks. We experimented with the parameter settings for SGA a lot, but obtained best results using overlap based error correction and default settings for all subprocesses.

ShoRAH: *version 0.8.2*

```
python shorah.py -b paired.sorted.bam -f reference.fasta
```

SOAPdenovo2: *version 2.04*

```
SOAPdenovo-127mer all -s soap_config.txt -o graph.k127 -p 16 -K 127
-B 0.01
```

We tried several k-mer sizes (55, 77, 99, 111, and 127) but obtained best results for $k = 127$ on all data sets. All other parameters were set to default values. The corresponding config file can be downloaded from https://bitbucket.org/jbaaijens/savage-benchmarks and as Supplemental Code.

SPAdes: *version 3.8.1*

```
python spades.py -o results/ -1 forward.fastq -2 reverse.fastq
```

We also experimented with SPAdes in "careful" mode (`--careful`), but overall we obtained best results in default mode.

metaSPAdes: *version 3.8.1*

```
python metaspades.py -o results/ -1 forward.fastq -2 reverse.fastq
```

VICUNA: *version 1.3*

```
vicuna-omp-v1.0 config.txt
```

VICUNA was run using default settings; the corresponding config file can be downloaded from https://bitbucket.org/jbaaijens/savage-benchmarks and as Supplemental Code.

QUAST: *version 4.3*

```
python customized_metaquast.py -m 500 -R ground_truth.fasta contigs.fasta
```

The customized metaquast script is available at https://bitbucket.org/jbaaijens/savage-benchmarks and as Supplemental Code.

5 Supplemental Code

The source code of our SAVAGE implementation, as well as all scripts and config files used for experiments, can be found as Supplemental Code.

6 Supplemental References

Bankevich A, Nurk S, Antipov D, Gurevich A, Dvorkin M, Kulikov A, Lesin V, Nikolenko S, Pham S, Prijbelski A, et al.. 2012. SPAdes: A new genome assembly algorithm and its applications to single-cell sequencing. *J Comp Biol* **19**: 455–477.

Di Giallonardo F, Töpfer A, Rey M, Prabhakaran S, Duport Y, Leemann C, Schmutz S, Campbell N, Joos B, Lecca M, et al.. 2014. Full-length haplotype reconstruction to infer the structure of heterogeneous virus populations. *Nucleic Acids Res* **42**: e115.

Li H. 2013. Aligning sequence reads, clone sequences and assembly contigs with bwa-mem. ArXiv:1303.3997.

Luo R, Liu B, Xie Y, Li Z, Huang W, Yuan J, He G, Chen Y, Pan Q, Liu Y, et al.. 2012. Soapdenovo2: an empirically improved memory-efficient short-read de novo assembler. *GigaScience* **1**: 18.

Malhotra R, Mukhopadhyay M, Poss M, and Acharya R. 2016a. A frame-based representation of genomic sequences for removing errors and rare variant detection in ngs data. ArXiv:1604.04803.

Malhotra R, Wu S, Mukhopadhyay M, Rodrigo A, Poss M, and Acharya R. 2016b. Maximum likelihood de novo reconstruction of viral populations using paired end sequencing data. ArXiv:1502.04239.

Martin M. 2011. Cutadapt removes adapter sequences from high-throughput sequencing reads. *EMBnet.journal* **17**: 10–12.

Mikheenko A, Saveliev V, and Gurevich A. 2016. Metaquast: evaluation of metagenome assemblies. *Bioinformatics* **32**: 1088–1090.

Nurk S, Meleshko D, Korobeynikov A, and Pevzner P. 2016. metaSPAdes: a new versatile de novo metagenomics assembler. Technical report, arXiv:1604.03071.

Prabhakaran S, Rey M, Zagordi O, Beerenwinkel N, and Roth V. 2014. HIV haplotype inference using a propagating dirichlet process mixture model. *IEEE Trans Comp Biol Bioinf* **11**: 182–191.

Simpson J and Durbin R. 2012. Efficient de novo assembly of large genomes using compressed data structures. *Genome Res* **22**: 549–556.

Töpfer A, Marschall T, Bull R, Luciani F, Schönhuth A, and Beerenwinkel N. 2014. Viral quasispecies assembly via maximal clique enumeration. *PLoS Comput Biol* **10**: e1003515.

Yang X, Charlebois P, Gnerre S, Coole M, Lennon N, Levin J, Qu J, Ryan E, Zody M, and Henn M. 2012. De novo assembly of highly diverse viral populations. *BMC Genomics* **13**: 475.

Zagordi O, Bhattacharya A, Eriksson N, and Beerenwinkel N. 2011. ShoRAH: estimating the genetic diversity of a mixed sample from next-generation sequencing data. *BMC Bioinf* **12**: 119.

Zhang J, Kobert K, Flouri T, and Stamatakis A. 2014. Pear: A fast and accurate illumina paired-end read merge. *Bioinformatics* **30**: 614–620.