Bioinformatics, YYYY, 0–0
doi: 10.1093/bioinformatics/xxxxx
Advance Access Publication Date: DD Month YYYY
Manuscript Category

Subject Section

A program for assembling incomplete viral genome scaffolds

Heather Wells

[1]Department of XXXXXXX, Address XXXX etc., [2]Department of XXXXXXX, Address XXXX etc.

*To whom correspondence should be addressed.

Associate Editor: XXXXXXX

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Abstract

Motivation: Existing *de novo* assembly programs are often unable to completely assemble viral genomes from wildlife samples with extensive host and environmental contamination as well as variable sequence and coverage. This program was designed to join scaffolds into as few contigs as possible, ideally one, for use as a reference in further downstream analysis.

Results: The program is able to assemble disjoint contigs and reads into a single continuous sequence. Conversion into 2-bit format allows for efficient storage of millions of reads and *k*-mers in memory.

Contact: hlw2124@cumc.columbia.edu

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

Emerging infectious diseases are a significant threat to global public health and agriculture and an estimated 60% are of zoonotic origin (Jones, Patel et al. 2008). In particular, novel viruses from wildlife have been responsible for several major epidemics in recent history, including Ebola virus (Goldstein, Anthony et al. 2018), Severe acute respiratory syndrome (SARS) (Lau, Woo et al. 2005, Li, Shi et al. 2005), Middle East respiratory syndrome (MERS) (Ithete, Stoffberg et al. 2013, Anthony, Gilardi et al. 2017), and Zika virus (Buechler, Bailey et al. 2017). Global surveillance efforts such as the USAID-PREDICT project sample wildlife populations from across the globe in an attempt to describe viral diversity and discover viruses of pandemic potential before human emergence occurs (Geoghegan and Holmes 2017, Carroll, Daszak et al. 2018).

Any potential for analysis of novel viral sequences for markers of pathogenicity or virulence requires that full-length, complete genomes be generated from deep sequencing data; however, because of extensive sequence variability, environmental contamination, and variable read coverage, viral genomes can be tricky for existing mapping and assembly programs (Yang, Charlebois et al. 2012, Shepard, Meno et al. 2016). Novel viruses from wildlife are often so divergent from their nearest known relative that reference-based mapping programs such as Bowtie2 are not suitable and *de novo* assembly is required. Existing *de novo* assembly programs designed specifically for much longer and less variable diploid genomes still often are not able to handle the diversity and mutation rate of viral genomes. A single host typically has several minor variants of the same virus called quasi-species (Domingo, Sheldon et al. 2012), alternatively spliced mRNA from the same species (Dubois, Terrier et al. 2014), and even recombinant or reassortant viruses between species (Perez-Losada, Arenas et al. 2015). These variants are important for analysis of infection dynamics and viral evolution, but a single, full-length consensus representation of the virus genome (a 'backbone') is a critical first step for any analysis of intra-host variation. The program presented here was designed to aid in the generation of full-length viral genomes from incomplete *de novo* assemblies of short-read sequencing data.

2 Methods

All programming was written using Python version 2.7. The program was written to function as an addition to a bioinformatics pipeline that includes host subtraction, *de novo* assembly, and BLAST filtering to retrieve and forward-orient viral contigs. Forward orientation of the contigs during this stage is required for the program. The contigs and remaining unused reads after assembly are used as input and the set of as few continuous contigs as possible is returned. Contigs and reads are converted to 2-bit format so that large read files and *k*-mer dictionaries can be stored in memory.

The algorithm consists of three basic steps: (1) contig merging, (2) contig extension using reads, and (3) repeated contig merging. The program breaks sequences into *k*-mers and finds shared *k*-mers between the sequences being compared. The *k*-mers are stored in a dictionary such that each *k*-mer key has a list of values that indicate the reads that contain that *k*-mer, its position within each read, and its orientation. For the first step, the left end of each contig is broken down and tested for shared *k*-mers with all other contigs on the right. Because the contigs are forward-oriented during the BLAST stage before input, they do not need to be tested in the reverse direction. After making all possible merges, the program then looks for shared *k*-mers between each contig and any read in any direction. Once a shared *k*-mer is found, the program will then find the set of all nucleotides that are in the *k*+1 position for all reads that contain that *k*-mer and accept the most common nucleotide. The program will then shift the *k*-mer forward by the accepted nucleotide and remove the last nucleotide and look for that shared *k*-mer with other reads. The accepted nucleotide is saved for contig extension. This process is continued until no more shared *k*-mers are found and each extension is added to the contig. Finally, after extension, the contigs are again checked for overlap and merged such that the fewest contigs possible remain.

3 Results

A toy data set was designed using a single reference sequence with contigs representative of common *de novo* assembly errors, including gaps between contigs, contigs with mis-assembled sequence on the ends, and contigs that slightly overlap but were not joined by the assembler. A small set of unused reads was designed by taking reads that mapped to the region from which the contigs were designed and subtracting all reads that mapped to the contigs. The position of the contigs and reads is shown in Figure 1.

3.1 Performance

The program was able to successfully assemble the contigs and reads into a single continuous contig in around 40 seconds on a machine with a 2.8Ghz processor. The majority of the time was consumed breaking the reads into *k*-mers.

Even though the processing time would be quite large, the compression of the reads and *k*-mers into 2-bit format means that a large number of reads and *k*-mers can be stored in memory. Each 100nt read only requires 25 bytes to store in 2-bit format, and storing read names as unsigned integers means that read names up to 20,000,000 can be stored in an additional 25 bits for a total of just over 28 bytes per read. Additionally, supposing a *k*-mer size of 31, each *k*-mer requires just under 8 bytes, each read name an additional 25 bits, a location unsigned integer in 7 bits, and an orientation signal in a single bit, for a total of ~12 bytes per *k*-mer. Each 100nt read will consist of 70 *k*-mers, resulting in 840 bytes of *k*-mers per read. Thus, if no single *k*-mer is shared between two reads, approximately 18,000,000 reads and their *k*-mers can be stored on a machine with 16GB of memory. The reality is likely much higher since many *k*-mers will be found multiple times in different reads; however, Python tends to over-allocate space with lists as dictionary values and is not perfectly space efficient. The results of the memory calculations are summarized in Table 1.

3.1.1 Limitations

(1) If two contigs from *de novo* assembly overlap but have at least one mismatch, the program can only merge them if there is a shared *k*-mer on either side of the mismatch.

(2) Further, contigs will not assemble if their overlapping regions are greater than the twice the trimsize. Generally, this should be avoided by subtracting out reads that mapped to within the contig, but the program allows setting the trimsize to a custom value if this becomes a problem.

(3) For both contig merging and read extension, the program will not retain any ambiguities in cases of mismatches and will simply choose the most common nucleotide. The goal is only to obtain a single continuous backbone for which further variant analysis can be performed using the backbone as a reference with established mapping and variant calling programs.

(4) If only two reads are found and the extension nucleotides are not identical, the program will not output an error but instead will choose one of the nucleotides for extension. This can cause the entire extension to terminate if that nucleotide is not represented in any other *k*-mers in further reads.

(5) Finally, the program is able to hold large files in memory but suffers an abysmal performance rate.

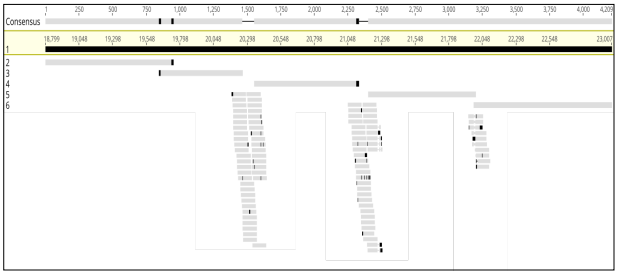

Fig. 1. Toy data set. The example data includes five contigs and a set of reads bridging the gaps between contigs.

3.2 Future Directions

Improving the efficiency of the algorithm is critical to improve its utility for large files. Future work will also focus on improving the behavior of the algorithm when encountering mismatches and ambiguities. This may include a variant calling module with a cutoff coverage to call a site a true ambiguity, a phred-score lookup to make a call between 50/50 nucleotides, and/or local alignment capabilities.

Table 1. Algorithm memory usage. Space for contig storage is negligible.

Dict	Key	Value	Total	Per read
Reads	Read name 3-4 bytes	Read sequence 25 bytes	~28 bytes	28 bytes
k-mers (k=31)	k-mer sequence 8 bytes	[read, position, orientation] 4-5 bytes	~12 bytes	840 bytes
Total				868 bytes per read

Source Code

The current version of the program can be found at:

Funding

This work has been supported by the .....

Conflict of Interest: none declared.

References

Anthony, S. J., K. Gilardi, V. D. Menachery, T. Goldstein, B. Ssebide, R. Mbabazi, I. Navarrete-Macias, E. Liang, H. Wells, A. Hicks, A. Petrosov, D. K. Byarugaba, K. Debbink, K. H. Dinnon, T. Scobey, S. H. Randell, B. L. Yount, M. Cranfield, C. K. Johnson, R. S. Baric, W. I. Lipkin and J. A. Mazet (2017). "Further Evidence for Bats as the Evolutionary Source of Middle East Respiratory Syndrome Coronavirus." MBio **8**(2).

Buechler, C. R., A. L. Bailey, A. M. Weiler, G. L. Barry, M. E. Breitbach, L. M. Stewart, A. J. Jasinska, N. B. Freimer, C. Apetrei, J. E. Phillips-Conroy, C. J. Jolly, J. Rogers, T. C. Friedrich and D. H. O'Connor (2017). "Seroprevalence of Zika Virus in Wild African Green Monkeys and Baboons." mSphere **2**(2).

Carroll, D., P. Daszak, N. D. Wolfe, G. F. Gao, C. M. Morel, S. Morzaria, A. Pablos-Mendez, O. Tomori and J. A. K. Mazet (2018). "The Global Virome Project." Science **359**(6378): 872-874.

Domingo, E., J. Sheldon and C. Perales (2012). "Viral quasispecies evolution." Microbiol Mol Biol Rev **76**(2): 159-216.

Dubois, J., O. Terrier and M. Rosa-Calatrava (2014). "Influenza viruses and mRNA splicing: doing more with less." MBio **5**(3): e00070-00014.

Geoghegan, J. L. and E. C. Holmes (2017). "Predicting virus emergence amid evolutionary noise." Open Biol **7**(10).

Goldstein, T., S. J. Anthony, A. Gbakima, B. H. Bird, J. Bangura, A. Tremeau-Bravard, M. N. Belaganahalli, H. L. Wells, J. K. Dhanota, E. Liang, M. Grodus, R. K. Jangra, V. A. DeJesus, G. Lasso, B. R. Smith, A. Jambai, B. O. Kamara, S. Kamara, W. Bangura, C. Monagin, S. Shapira, C. K. Johnson, K. Saylors, E. M. Rubin, K. Chandran, W. I. Lipkin and J. A. K. Mazet (2018). "The discovery of Bombali virus adds further support for bats as hosts of ebolaviruses." Nat Microbiol **3**(10): 1084-1089.

Ithete, N. L., S. Stoffberg, V. M. Corman, V. M. Cottontail, L. R. Richards, M. C. Schoeman, C. Drosten, J. F. Drexler and W. Preiser (2013). "Close relative of human Middle East respiratory syndrome coronavirus in bat, South Africa." Emerg Infect Dis **19**(10): 1697-1699.

Jones, K. E., N. G. Patel, M. A. Levy, A. Storeygard, D. Balk, J. L. Gittleman and P. Daszak (2008). "Global trends in emerging infectious diseases." Nature **451**(7181): 990-993.

Lau, S. K., P. C. Woo, K. S. Li, Y. Huang, H. W. Tsoi, B. H. Wong, S. S. Wong, S. Y. Leung, K. H. Chan and K. Y. Yuen (2005). "Severe acute respiratory syndrome coronavirus-like virus in Chinese horseshoe bats." Proc Natl Acad Sci U S A **102**(39): 14040-14045.

Li, W., Z. Shi, M. Yu, W. Ren, C. Smith, J. H. Epstein, H. Wang, G. Crameri, Z. Hu, H. Zhang, J. Zhang, J. McEachern, H. Field, P. Daszak, B. T. Eaton, S. Zhang and L. F. Wang (2005). "Bats are natural reservoirs of SARS-like coronaviruses." Science **310**(5748): 676-679.

Perez-Losada, M., M. Arenas, J. C. Galan, F. Palero and F. Gonzalez-Candelas (2015). "Recombination in viruses: mechanisms, methods of study, and evolutionary consequences." Infect Genet Evol **30**: 296-307.

Shepard, S. S., S. Meno, J. Bahl, M. M. Wilson, J. Barnes and E. Neuhaus (2016). "Viral deep sequencing needs an adaptive approach: IRMA, the iterative refinement meta-assembler." BMC Genomics **17**: 708.

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