PeerJ Preprints NOT PEER-REVIEWED

gapFinisher: a reliable gap filling pipeline for SSPACE-LongRead scaffolder output

Juhana I Kammonen Corresp., [1] , **Olli-Pekka Smolander** [1] , **Lars Paulin** [1] , **Pedro AB Pereira** [1] , **Pia Laine** [1] , **Patrik Koskinen** [2] , **Jukka Jernvall** [1] , **Petri Auvinen** [1]

[1] Institute of Biotechnology, University of Helsinki, Helsinki, Finland
[2] Blueprint Genetics Ltd., Helsinki, Finland

Corresponding Author: Juhana I Kammonen
Email address: juhana.kammonen@helsinki.fi

Unknown sequences, or gaps, are largely present in most published genomes across public databases. Gap filling is an important finishing step in *de novo* genome assembly, especially in large genomes. The gap filling problem is nontrivial and while many computational tools exist partially solving the problem, several have shortcomings as to the reliability and correctness of the output, i.e. the gap filled draft genome. SSPACE-LongRead is a scaffolding software that utilizes long reads from multiple third-generation sequencing platforms in finding links between contigs and combining them. The long reads potentially contain sequence information to fill the gaps, but SSPACE-LongRead currently lacks this functionality. We present an automated pipeline called gapFinisher to process SSPACE-LongRead output to fill gaps after the actual scaffolding. gapFinisher is based on controlled use of a gap filling tool called FGAP and works on all standard Linux/UNIX command lines. We conclude that performing the workflows of SSPACE-LongRead and gapFinisher enables users to fill gaps reliably. There is no need for further scrutiny of the existing sequencing data after performing the analysis.

PeerJ Preprints NOT PEER-REVIEWED

Helsinki, Finland 07 Dec 2017 MANUSCRIPT TEXT

gapFinisher: a reliable gap filling pipeline for SSPACE-LongRead scaffolder output

Juhana I Kammonen (1)*, Olli-Pekka Smolander (1), Lars Paulin (1), Pedro AB Pereira (1), Pia Laine (1), Patrik Koskinen (1,2), Jukka Jernvall (3) & Petri Auvinen (1)

(1) DNA Sequencing and Genomics Laboratory, Institute of Biotechnology, University of Helsinki, Finland

(2) Blueprint Genetics Ltd. Finland

(3) Evolutionary Phenomics Group, Institute of Biotechnology, University of Helsinki, Finland

*corresponding author (juhana.kammonen@helsinki.fi)

BIOGRAPHICAL NOTE

The corresponding author is a PhD student representing the Saimaa ringed seal genome project (http://www.saimaaringedseal.org). The PhD thesis topic covers the full genome sequencing and genome annotation of the Saimaa ringed seal (*Pusa hispida saimensis*), considered one of the national animals of Finland and one of the most endangered animals in the world.

KEYWORDS

genome assembly; draft genomes; scaffolding; next-generation sequencing; long read technologies

ABSTRACT

PeerJ Preprints NOT PEER-REVIEWED

Unknown sequences, or gaps, are largely present in most published genomes across public databases. Gap filling is an important finishing step in *de novo* genome assembly, especially in large genomes. The gap filling problem is nontrivial and while many computational tools exist partially solving the problem, several have shortcomings as to the reliability and correctness of the output, i.e. the gap filled draft genome. SSPACE-LongRead is a scaffolding software that utilizes long reads from multiple third-generation sequencing platforms in finding links between contigs and combining them. The long reads potentially contain sequence information to fill the gaps, but SSPACE-LongRead currently lacks this functionality. We present an automated pipeline called gapFinisher to process SSPACE-LongRead output to fill gaps after the actual scaffolding. gapFinisher is based on controlled use of a gap filling tool called FGAP and works on all standard Linux/UNIX command lines. We conclude that performing the workflows of SSPACE-LongRead and gapFinisher enables users to fill gaps reliably. There is no need for further scrutiny of the existing sequencing data after performing the analysis.

INTRODUCTION

Gap filling is one of the final phases of *de novo* genome assembly. First, assembly algorithms produce contiguous sequences of overlapping sequencing reads, commonly known as contigs. A contig is a continuous DNA sequence entity without any ambiguities, unknown bases marked as N. Second, the contigs are connected into longer fragments, scaffolds, using specialized sequencing read data. Until the development of long read technologies, the data used to be primarily mate-pair reads, known also as jumping reads. The mate-pair libraries are usually made of size selected DNA fragments, where fragment size is usually in the order of thousands of base pairs. The ends of these fragments are sequenced and resulting reads are used for scaffolding. Currently long continuous reads e.g. from Pacific Biosciences (Menlo Park, California, United States) RS II or Sequel third-generation sequencing platforms are commonly used. While the scaffolding step links and orders the contigs, it usually leaves variable amounts of unknown sequences, strings of N-characters, in between them. These unknown sequences are called

PeerJ Preprints NOT PEER-REVIEWED

gaps. Finally, the gap filling stage aims to resolve these unknown sequences with additional sequencing data, (e.g. Boetzer *et al.*, 2011; Boetzer & Pirovano, 2012) or without additional data (e.g. Li & Copley, 2013).

In this paper, we present an automated gap filling pipeline called gapFinisher. We pursue a solution to the gap filling problem that utilizes long reads and only unaltered draft genomes to prevent any loss of data. We set strict alignment parameters for the gap filling stage to ensure correctness and uniqueness of the filled gaps. We conclude that applying gapFinisher enables efficient and reliable gap filling by controlling the use of the FGAP algorithm (Piro *et al*., 2014). Furthermore, gapFinisher time complexity proves linear with respect to the size of the input. The system requirements are MATLAB Compilation Runtime (MCR) for FGAP and a Perl interpreter for SSPACE-LR. Besides these, the gapFinisher pipeline does not require any additional software to be installed.

Gap filling

SSPACE-Standard (Boetzer *et al*., 2011) and SSPACE-LongRead (SSPACE-LR) (Boetzer & Pirovano, 2014) are scaffolding tools for paired-end (also mate-pair) reads and long continuous reads, respectively. While this software is available free for academic users, both are commercial products and upgrades and most of the support require a proprietary license. SSPACE-Standard is commonly applied in the first scaffolding step where contigs are oriented and ordered into longer connected sequences and it accepts paired-end data from any next-generation sequencing technology if read-orientation information and mean values and standard deviations of the insert sizes for each read library are provided. SSPACE-LR utilizes Pacific Biosciences filtered subreads (CLR = *Continuous Long Reads*) in finding long links between contigs or existing scaffolds and combining them into "superscaffolds" with new gaps introduced between the sequences. SSPACE-LR first maps the long reads into the contig assembly using the BLASR aligner specialized for long read alignment (Chaisson & Tessler, 2012). Based on these alignments, contigs are then linked into scaffolds and N-characters (gaps) are placed between the connected contigs. While the CLR reads contain information of the actual nucleotide sequence in the gaps, this feature is not exploited

PeerJ Preprints NOT PEER-REVIEWED

in the current version of SSPACE-LR (version 1.1). However, the software can report the exact information about which reads were associated when creating the new scaffold and the new gap(s). In the gapFinisher pipeline, we actually utilize this information to fill the gaps in the newly created scaffolds on the go.

A central part of gap filling is the alignment of long sequences against the contigs. This is challenging due to the relatively high error-rates of contemporary long read data (Laver *et al*., 2015) and the sequencing errors, e.g. (Nakamura *et al*., 2011; Schirmer *et al*., 2015) and local misassemblies at the contig level (Boetzer & Pirovano, 2014). The BLAST local alignment tool (Altschul *et al*., 1990) is the most commonly used approach for the identification of areas of high similarity between multiple sequences. Different scaffolding and gap filling tools apply BLAST either directly (Piro *et al*., 2014), or the method is refined (Chaisson & Tessler, 2012) and applied (English *et al*., 2012; Boetzer & Pirovano, 2014). All tools based on BLAST contain multiple parameters, e.g. for mismatches and gaps, affecting their ability to detect non-perfect matches and it is not always clear how these should be defined.

Several gap filling software exist. GapFiller (Boetzer & Pirovano, 2012) is a commercial program by the authors of SSPACE-tools and often used together with them. GapFiller uses paired-end read information to fill in sequences at contig ends where overlapping reads reach into the gap created on the SSPACE-Standard step by mate-pair reads. Where reads are unable to span the whole length of the unknown sequence, the gap is not completely filled and unknown bases (N-characters) will remain in the output version of the draft genome (Boetzer *et al*., 2011).

PBJelly (English *et al*., 2012) is a scaffolding and gap filling tool integrated into the Pacific Biosciences (PacBio) SMRT Analysis software suite, the main user interface for data analysis using PacBio long reads. In comparison to other gap filling tools, the PBJelly is run in six separate stages (setup, mapping, support, extraction, assembly and output) and requires additional software libraries, most notably the SMRT Portal software suite and BLASR (Chaisson & Tessler, 2012). Despite it is possible to run PBJelly in a single-core computer, the workflow is clearly designed for high-throughput computing in a grid, e.g. the Sun Grid Engine (Gentsch, 2001). A peculiar default feature of PBJelly is that it by default inflates short gaps

PeerJ Preprints NOT PEER-REVIEWED

(< 25 bp) to a length of exactly 25 bp with the apparent purpose of emphasizing the location of the gaps (English *et al*., 2012).

Gap2Seq (Salmela *et al*., 2016) provides a purely computational solution to the gap filling problem. It works well on prokaryote genomes but does not scale well to larger genomes, where repetitive sequences confuse the algorithm and the sheer size of the genome makes running times infeasibly long.

FGAP (Piro *et al*., 2014) is a gap filling tool that utilizes various types of read data and BLAST alignments to find and fill gaps in draft genomes. The BLAST utility is bundled with the release version of FGAP, but a MATLAB Compilation Runtime is required. The gapFinisher pipeline presented in this paper is based on FGAP and enables more reliable and controlled gap filling.

State of the art

Although FGAP efficiently reduces the number of gaps in various draft genomes (Piro *et al*., 2014), the tool has the rather troubling feature of setting no limit to the number of times an input read is used in gap filling should the BLAST alignment return multiple good hits (Fig. 1). With the default setting of FGAP, undesired multiple alignments of query sequences may occur due to repetitive regions in the draft genome, or overly lenient alignment parameters for the ends of the query sequences (Fig. 1). We could verify this behaviour on an FGAP test run with a preliminary draft genome of an unpublished marine mammal from the *Phocidae* family (Table S4). Ideally, gap filling should be a unique process in the sense that a single input long read would find a single good alignment in the draft genome and fill any gaps in that single location.

(Figure 1 here)

Repeat masking may improve the scaffolding and gap filling of highly repetitive draft genomes. For example, it has been estimated that more than 60% of the 3,3 Gb modern human (*H. sapiens*) genome consists of repetitive sequences, (e.g. de Konig *et al*., 2011). With the repetitive sequences at contig ends eliminated, the scaffolding / gap filling algorithms are less likely to make incorrect alignments. One

PeerJ Preprints NOT PEER-REVIEWED

example of repeat masking software tools is RepeatMasker (Smit *et al*., 2013) which finds short and long interspersed elements as well as simple repeats in the input genomic sequence. RepeatMasker may mask coding regions of the input genome, especially those located at the terminal regions of open reading frames (ORFs). Furthermore, RepeatMasker may mask some shorter potential element-coding sequences such as ribosomal RNAs (Smit *et al*., 2013). Repeat masking may lower the inherent risk of incorrect alignments or multiple alignments in the contig ends. In this paper, however, we pursue a solution that utilizes only unaltered (unmasked) draft genomes to prevent any loss of data.

Solving short gaps of e.g. 1-20 base pairs in length by simple read alignment maps produced by e.g. the Burrows-Wheeler Aligner (Li & Durbin, 2009) or the Bowtie 2 aligner (Langmead & Salzberg, 2012) is not investigated in detail in this study, but may be one of the prospects of solving the gap filling problem for short gaps. For instance, some singular unknown bases and short N-sequences at gap edges are solved by the re-assembly stage of the Pilon assembly polishing tool, where an alignment map file can be supplied as input and a specific option set for gap filling (Walker *et al*., 2014).

MATERIALS & METHODS

The current release of gapFinisher works only on the output of SSPACE-LongRead (Boetzer & Pirovano, 2014). The basic workflow of gapFinisher is illustrated in Figure 1c and in further detail in Figure 2. Thus, before running gapFinisher, the user must successfully run SSPACE-LongRead for a dataset at least once. It is imperative to have the "-k" option enabled when running SSPACE-LongRead. This setting will create the critical "inner-scaffold-sequences" subdirectory that contains for each superscaffold the references to the actual long read sequences (one or more) that created the scaffold. The pipeline will not run if this directory does not exist. When successful, gapFinisher then works as follows (*cf.* Figure 2):

1. Index the draft genome FASTA file and the long read FASTA file

PeerJ Preprints NOT PEER-REVIEWED

2. Generate a list of names of all superscaffolds SSPACE-LongRead (-k 1 option enabled) has created
3. For each superscaffold in the list:
    a. Create a new FGAP working directory for the current superscaffold
    b. Fetch all full CLR reads associated with the current superscaffold
    c. For each of the CLR reads associated with the current superscaffold:
        i. Execute FGAP using the current superscaffold as draft and the CLR read as input
        ii. If FGAP could fill (one or more) gaps in the current superscaffold, save FGAP output as the new draft for the current superscaffold
4. Compile results from each working directory as filled_scaffolds.fasta
5. Compile filled_scaffolds.fasta and the unfilled/untouched scaffolds from the original draft genome as scaffolds_gapfilled_FINAL.fasta
6. [optional] Clean the working directories (to save disk space).

The rapid fetching of reads is based on the operation of the fastaindex (step 1 above) and fastafetch (step 2b above) utilities of the exonerate toolkit (Slater & Birney, 2004) v. 2.4.0. Precompiled executables of these utilities are bundled with the gapFinisher release and fully integrated into the workflow of the gapFinisher pipeline.

(Figure 2 here)

When using Pacific Biosciences filtered subreads with SSPACE-LongRead, it is in theory possible, that separate reads originating from the same well of the PacBio SMRT cell are aligned into separate places by the BLASR aligner (cf. Fig. 1a and Fig. 1b). Filtered subreads from the same well of the SMRT cell always originate from the same molecule and thus should align to locations close to one another. The legacy BLASR (Chaisson & Tessler, 2012) version that SSPACE-LongRead is using has no formal assertion for this. Hence, we set gapFinisher to keep track of the origins of the filtered subreads. The pipeline issues an appropriate warning when gap filling under conflicting read origin is about to happen and aborts the filling of the gap in question. Conflicting read origins further indicate potential errors in the scaffolding step. Consequently, the location and read information of the conflict are included in the warning message and logged.

(Table 1 here)

Here, we predisposed seven separate sequencing read datasets from both bacterial and eukaryote organisms (Table 1) to *de novo* assembly and scaffolding. Finally, we performed gap filling on the created scaffolds with gapFinisher (Table 2). First, we had two *Escherichia coli* (*E. coli*) bacterial genome drafts. Second, we extended the analysis to a set of further four bacterial genomes: *Bibersteinia trehalosi, Mannheimia haemolytica, Francisella tularensis* and *Salmonella enterica*. The bacterial read data are the same that were used as test data for the SSPACE-LongRead scaffolder (Boetzer & Pirovano, 2014) and are available at: http://www.cbcb.umd.edu/software/PBcR/closure/index.html and the Sequencing Read Archive (SRA) links therein. For *B. trehalosi*, we used the reference sequence *Bibersteinia trehalosi* USDA-ARS-USMARC-188 (Harhay *et al*., 2014) Since the publication of SSPACE-LongRead, a reference genome has become available to *M. haemolytica* as well (Eidam *et al*., 2013). Finally, we included an unpublished marine mammal (*Phocidae* family) draft genome in final stage with 236,592 contigs scaffolded into 10,143 superscaffolds with gaps to get a reference on how gapFinisher performs on a much larger genome. The raw sequencing coverage of the draft genome was on average 25X for the Illumina (San Diego, California, United States) reads and 50X for the PacBio CLR reads (Table 1). When assembled with the miniasm (Li, 2016) using all the PacBio reads, we got an additional “PacBio-only” assembled version of the draft genome with 1,314 contigs which we then scaffolded and gap filled (Table 2).

For the Illumina MiSeq reads, we further applied the Fast Length Adjustment of SHort Reads (FLASH) protocol that finds overlaps at the ends of the MiSeq paired-end reads and joins the reads if found (Magoč & Salzberg, 2011). Thus, about half of the reads in each MiSeq dataset could be combined to longer initial fragments. This feature is likely to improve the *de novo* genome assemblies while longer initial read length may be enough to span short repeats and indels. The uncombined reads were supplied as additional

PeerJ Preprints NOT PEER-REVIEWED

paired-end libraries in all assemblies. The Roche 454 Genome Sequencer data available for the draft genomes was not utilized here, as our benchmark did not include a suitable assembler, e.g. Newbler (Margulies *et al*., 2005) for these data. Furthermore, the performance of Newbler was evaluated rather extensively in the SSPACE-LongRead original publication (Boetzer & Pirovano, 2014) and in most of cases Newbler could not perform as well as the other short read assemblers.

We assembled the draft genomes with the SPAdes (Bankevich *et al*., 2012) and miniasm (Li, 2016) assemblers. SPAdes can employ both Illumina short reads and PacBio CLR reads. In contrast, miniasm only works properly with PacBio CLR reads or other long reads with a sufficient sequencing coverage. This is because the read trimming phase of miniasm requires a read-to-read mapping length of at least 2,000 bp with a minimum of 100 bp non-redundant bases (Li, 2016). This condition is not met by the short read datasets used in this study. An additional and a highly useful feature of miniasm is the minidot plot drawing utility and it was used to create the dotplots for comparisons to the reference genomes (Figs. 3 and S1).

The scaffolding step included the combined use of SSPACE-LongRead (academic license, software version 1.1) (Boetzer & Pirovano, 2014) and the gapFinisher pipeline. We first executed SSPACE-LongRead for all samples to create the superscaffold assemblies for the six bacterial genomes and the unpublished mammal draft genome (*Phocidae* family). The same long read data was applied for the scaffolding of both SPAdes and miniasm contig assemblies. For each scaffold assembly, we then executed gapFinisher to fill the gaps introduced by the scaffolding step. Due to the large size (~2,5 gigabases) of the unpublished mammal genome, the SSPACE-LongRead and gap filling stage for the miniasm assembly had to be executed in two consecutive runs with 25X (50 % of the total coverage) PacBio reads applied to each part. On the other hand, the scaffolding of the mammal SPAdes assembly was executed in five separate stages as part of the actual genome project of the mammal. About 10X coverage of PacBio reads of insert were applied at each stage and gapFinisher executed between the stages. Thus, the results for this

PeerJ Preprints NOT PEER-REVIEWED

assembly (Table 2) show statistics for the final stage and average CLR reads per scaffold is the average of all five stages.

We visualized the different stages of the draft assemblies for all genomes using the subplot utility of the MATLAB toolkit (Figs. 3 and S1). Furthermore, we visualized the final stages of the assembly and scaffolding by aligning the reference genomes and the two drafts from the SPAdes and miniasm assembly pipelines with the progressiveMauve algorithm of the Mauve (Darling *et al*., 2004) alignment and visualization tool (Figs. 4 and S2). Mauve reveals the number and similarity of Locally Collinear Blocks (LCBs) between the input sequences.

In order to assess the performance of the software, all of the SPAdes, miniasm, SSPACE-LongRead and gapFinisher runs were executed in two separate 64-bit Linux computer environments. First, the bacterial genomes were assembled, scaffolded and gap filled in a single-processor computer running Ubuntu Linux 14.04 with 20 GB of RAM, the equivalent to a modern office workstation with a small RAM extension. The processor was Intel (Santa Clara, California, United States) Core (TM) with a frequency of 3,2 GHz. Second, we built the mammal genome in a multi-core supercomputer running Ubuntu Linux 14.04 with 1 TB of RAM and using 16 Advanced Micro Devices (Sunnyvale, California, United States) Opteron(TM) processors with a frequency of 2,5 GHz each. The latter setup is equivalent to a small-scale computer cluster.

RESULTS

The results are presented both from the viewpoint of how finished the draft genomes are before and after gapFinisher and how gapFinisher performs in general and with respect to FGAP. Key statistics of the

PeerJ Preprints NOT PEER-REVIEWED

assembly benchmark results were compiled (Fig. 5) and the genome alignments to the bacterial reference genomes were visualized (Figure .3 and Figure. 4 and supplementary figures S1 and S2).

Genomes

Regarding *de novo* assembly of the genomes, we noticed similar behaviour of the SPAdes assembler as reported by the authors of the SSPACE-LongRead (Boetzer & Pirovano, 2014). Namely, that the SPAdes assembly pipeline introduced repeats at the ends of the contigs that evidently prohibit many CLR reads from aligning into the contig ends and thus the scaffold assembly is left with a higher number of uncombined sequences than expected (Table 2 and Fig. 4a). Nevertheless, scaffolding with SSPACE-LongRead reduced the number of total sequences in all the assemblies. This was especially evident in the *Mannheimia haemolytica* draft genome, where SSPACE-LongRead reduced the number of sequences in the draft assembly from 112 to 17 (84,8 % reduction). A notable increase in basic assembly statistics, such as the N50 contig length and number of sequences, was observed throughout (Table 2). The miniasm assembler (Li, 2016) outperformed the assemblers used in the SSPACE-LongRead test assemblies (Boetzer & Pirovano, 2014) and the SPAdes assembler (Bankevich *et al.*, 2012) in our benchmark in terms of number of output contigs, N50 and gap length (Table 2). On the other hand, the median similarity of the alignments to the bacterial reference genomes is lower across all bacterial draft genomes from the miniasm pipeline (Figs. 4 and S2).

It is evident from the assembly results that both the SPAdes and miniasm assemblers are optimized for the *E. coli* K12 genome: The number of SPAdes assembly contigs was the lowest of the bacterial assemblies in this study, namely 35 (Table 2). The miniasm assembly of the *E. coli* K12 genome was a single sequence (Fig. S1 and Table 2) and thus was the only draft genome not to require scaffolding or gap filling. Furthermore, miniasm was able to construct the full *E. coli* K12 genome from PacBio reads in 3

PeerJ Preprints NOT PEER-REVIEWED

minutes (Fig. 5 and Table S3). The final assembly consists of a single long bacterial genome (Table 2) in 4 Locally Collinear Blocks (LCB's) according to progressiveMauve (Darling *et al*., 2004) alignment (Table 2 and Fig. S2a). The contig assembly results for the other bacterial genomes were more variable with both SPAdes and miniasm (Table 2 and Figs. 1 and 2).

Regarding the overall output of the assemblers, miniasm consistently reports zero N's at the contig assembly stage (Table 2). Furthermore, the miniasm contig assemblies are more contiguous in the sense that they consist of less sequences when compared to the SPAdes assemblies in all cases (Table 2). However, the SPAdes assemblies report some gapped sequences already at the contig assembly of *E. coli O157* (3 bp), *B. trehalosi* (2 bp), *M. haemolytica* (35 bp) and *S. enterica* (655 bp) (Table 2).

Evidently, gapFinisher can fill about 50 % of the gapped sequence (Table 2) in the scaffolded draft genomes and retains the structure of the genomes in all cases (Figs. 3 and 4 and Supplementary figures S1 and S2). The lowest percentage of gaps filled was with the second stage of the mammal genome miniasm scaffolding (4,1 %) and the highest percentage of gaps filled was with the scaffolding of the B. trehalosi SPAdes assembly (85,7 %). At the nucleotide level, several kilobases of gapped sequence is being filled in all cases (Table 2). No large insertions, deletions or inversions are introduced by the gap filling stage with gapFinisher (Table 2 and Figs. 3 and S1). The bacterial initial assemblies were refined to scaffolds using PacBio filtered subreads. There were no cases of gapFinisher warning about separate reads from same SMRT cell well attempting to fill disparate gaps in any of the bacterial genomes.

(Figures 3 and 4 here) (Table 2 here)

Performance

(Figure 5 here)

PeerJ Preprints NOT PEER-REVIEWED

Besides MATLAB Compilation Runtime and a Perl (Christiansen *et al*., 2012) interpreter, gapFinisher does not require any other software to be installed. Furthermore, the gapFinisher pipeline is contained in a single phase, namely the actual execution of the gap filling, where e.g. the PBJelly (English *et al*., 2012) pipeline has six separate phases.

Due to the serial design of the pipeline, gapFinisher running time holds quite neatly at about 3-5 wall-clock seconds per CLR read per scaffold (Table S3). Thus, gapFinisher time complexity can be regarded as linear with relation to the number of input scaffolds and the total coverage of the scaffolding reads. Where average number of CLR reads per created scaffold was high, as was the case with the bacterial genomes of *F. tularensis, M. haemolytica* and *S. enterica*, gapFinisher running time in single-core mode was notably higher (Fig. 5 and Table S3).

We studied random access memory (RAM) use of gapFinisher (Fig. 5 and Table S3). We used a built-in Linux utility (/usr/bin/time) to measure the peak RAM use during each of the assembly stages. Again, the serial design of gapFinisher keeps the RAM use of the gap filling stage at all but nominal level (Fig. 5a), also in the case of a much larger genome (Fig. 5c). In general, the peak RAM use of less than 1 GB we detected in all cases means that gapFinisher could be executed in almost any Linux computer, even most tablets. Nevertheless, the preceding assembly steps tend to use significantly more RAM (Fig. 3a and Fig. 3c). In particular, the larger mammal genome used more than 500 GB of RAM in the contig assembly stage and more than 80 GB of RAM in the SSPACE-LongRead stage (Table 2).

DISCUSSION

Gap filling is a non-trivial problem with many existing solutions today in the form of software tools. The correctness of the outputs of different tools is variable. For a large genome under assembly, the default parameter settings of FGAP clearly are too lenient and may lead to incorrect gap filling in large draft

PeerJ Preprints NOT PEER-REVIEWED

genomes (Table S4). Repeat masking before gap filling with FGAP alone may be recommended, especially because FGAP utilizes BLAST (Altschul *et al*., 1990) directly for the long read alignment.

Typically, contig assemblies do not contain any unknown sequence (N-characters) and the output of miniasm correctly follows this principle (Table 2). However, it is evident from the SPAdes assembler results that a small number of N's may be introduced already at the contig assembly stage (Table 2). This may be due to the N's present in the actual read data that is not uncommon for Illumina sequencing reads but is more unusual for PacBio reads.

gapFinisher is not able to fill all gapped sequences in the draft assembly (Table 2). This is because the CLR reads of the Pacific Biosciences platform do contain base-call errors, (e.g. Laver *et al*., 2015) and gapFinisher employs a strict alignment scheme of the long reads and only fills a gap when a reasonably correct alignment of known sequences at the gap edges is found (Figs. 1c and 2). Consequently, it is possible that some gaps are prevented from filling despite the evidence being there. A solution is to run gapFinisher on less strict parameters and then confirm the correctness of the result using other alignment tools. Nevertheless, gapFinisher with the default settings can reduce the amount of gapped sequence in the example draft genomes by about 50 % in general (Table 2).

Regarding the use of filtered subreads in the bacterial genome assemblies of this study, gapFinisher did not detect any cases where separate reads from the same SMRT cell well would have filled disparate gaps in the genomes. In applications where conflicting read origins could be a problem, it can be circumvented by producing reads of insert from the filtered subreads, albeit with the expense of genome level coverage. On the other hand, the reads of insert pipeline improves the overall quality of the reads which leads to more reliable alignments. Checking the read origin of the filtered subreads is a valuable additional

PeerJ Preprints NOT PEER-REVIEWED

correctness feature of the gapFinisher pipeline not present in the other gap filling tools presented in this study.

We found that the time complexity of gapFinisher is approximately linear with respect to the number of input scaffolds and long read evidence related to each of the scaffolds (Fig. 5 and Supplementary table S3). While the peak RAM use of gapFinisher stays at a nominal level in all the cases of small and large genomes (Fig 5a and Fig. 5c), the runtime varies significantly, even in small genome assemblies (Fig 5b). This feature may be optimized in future development versions. If the user can run gapFinisher in a supercomputer cluster, it is possible to specify the number of threads (option -t) and the utility will divide the input scaffolds into even parts, splitting the total running time by the number of processors assigned. In the case of our datasets, the parallelization would have significantly reduced the runtime of gapFinisher in the gap filling of bacterial genomes *M. haemolytica* and *S. enterica* (Fig. 5b and Supplementary table S3) and the effects could be clearly seen in the case of the mammal genome gap filling with 16 processors in use and parallelizing the workflow (Fig. 5c). gapFinisher is designed to work on all standard Linux/UNIX distributions on command line with as little dependencies as possible. Aside for having to first perform the actual scaffolding using SSPACE-LongRead, all the user needs to do is download gapFinisher and run it.

No matter which next-generation sequencing platform is in use, there exists a distinct base-call error profile affecting the output and the quality of the sequenced reads. Previously, sequence-specific systematic miscalls have been reported in the output of Illumina Genome Analyzer II platform (Dohm *et al.*, 2008; Nakamura *et al.*, 2011). Evidently, the more recent Illumina MiSeq platform is affected by the same miscall profile to some extent (Kammonen *et al.*, 2015; Schirmer *et al.*, 2015). The presence of a relatively high error-rate can also not be disputed in current high-throughput sequencing of long reads (Laver *et al.*, 2015). High error-rate is also a likely explanation to the observed lower overall similarity of locally collinear blocks (LCBs) in the miniasm part of our study (Figs. 4 and S2). Nevertheless, with ever-

PeerJ Preprints NOT PEER-REVIEWED

improving sequencing chemistries and throughput the issue of high error-rates is likely to grow smaller in the future. Error profile aware quality control methods could also help to counter the various miscalls made by NGS platforms.

The actual sequencing coverage of PacBio reads has an apparently significant effect in the finalization of the genomes: In the SSPACE-LongRead bacterial genome study (Boetzer & Pirovano, 2014), it was found that long-read coverage from around 60X upwards did not further improve genome closure on the contig level. Regarding read error-rates, it is already possible to self-correct PacBio CLR reads by using the reads of insert pipeline of the SMRT Analysis toolkit. For each sequenced molecule, an improved consensus sequence is obtained by aligning all the produced subreads together which cancels out the random errors in individual reads. The final quality of the sequence depends on the number of subreads obtained for each single molecule. Thanks to the nearly random error profile of the PacBio RS II instrument, single nucleotide miscalls in the reads will not be propagated to the reads of insert output, that is, the circular consensus (CCS) reads. Furthermore, the new Sequel instrument of Pacific Biosciences has promised 7-fold throughput as compared to the earlier RS II platform, which has major ramifications also for the throughput of corrected reads from the platform.

There may be additional approaches to the gap filling problem. In theory, a simple gap-tolerant alignment of sequencing reads of variable lengths using existing mapping tools would be able to reliably span at least short gaps, say 1-20 bp in length. This is one of the intriguing prospects of solving the gap filling problem, especially as the average read lengths of next-generation sequencing platforms are likely to only increase in the future.

CONCLUSIONS

PeerJ Preprints NOT PEER-REVIEWED

Here, we presented an automated pipeline to solve the gap filling problem using a combination of SSPACE-LongRead (Boetzer & Pirovano, 2014) and FGAP (Piro *et al*., 2014) in a controlled manner and wrapping these methods together in a pipeline called gapFinisher. gapFinisher ensures the uniqueness of the BLAST alignments returned by the FGAP algorithm by iterating through the read data one read and one superscaffold at a time. As evident from the result statistics (Table 2) and the visualizations of the draft genomes (Supplementary figures S1 and S2), the control provided by gapFinisher leads into efficient but reliable gap filling. The effects appear to scale up in a large genome *de novo* assembly (Table 2 and Fig. 5).

The applicability of gapFinisher is currently limited to SSPACE-LongRead academic license version output only and requires the user to be able to run SSPACE-LongRead at least once. Nevertheless, SSPACE-LongRead currently is the only publicly available scaffolding software that is able to produce information about the sequences spanning the gaps in the final scaffolds, i.e. the ”inner-scaffold sequences” subdirectory. Should other utilities with this key feature become available, we will further develop gapFinisher for full compatibility. Our pipeline contributes to filling long gaps and solving the non-trivial task of gap filling after scaffolding draft genomes of multiple organisms. Applying gapFinisher will accelerate the finishing of draft genomes of both prokaryote and eukaryote organisms, even in published genome assemblies.

The script to run the gapFinisher pipeline is published under GNU's general public license and can be downloaded at: http://www.helsinki.fi/~jkammone/gapFinisher.zip

Competing interests: None declared.

ACKNOWLEDGEMENTS

The authors would like to acknowledge all supporting organizations and collaborators. The authors would like to thank Dr. Ari Löytynoja for constructive comments on the manuscript text.

REFERENCES

Altschul SF, Gish W, Miller W *et al.* Basic local alignment search tool. *Journal of Molecular Biology*, 1990;215(3):403-10.

Bankevich A, Nurk S, Antipov D *et al.* SPAdes: a new genome assembly algorithm and its applications to single-cell sequencing. *Journal of Computational Biology*, 2012;19(5): 455-477.

Boetzer M, Henkel CV, Jansen HJ *et al.* Scaffolding pre-assembled contigs using SSPACE. *Bioinformatics* 2011;4(27): 578-579.

Boetzer M & Pirovano W. Toward almost finished genomes with GapFiller. *Genome Biology* 2012;13(6): R56.

Boetzer M & Pirovano W. SSPACE-LongRead: scaffolding bacterial draft genomes using long read sequence information. *BMC Bioinformatics* 2014;15(1): 211.

Camacho C, Madden T, Ma N *et al.* BLAST command line applications user manual. National Center for Biotechnology Information (US). http://www.ncbi.nlm.nih.gov/books/NBK1763/. (09 October 2017, date last accessed)

Chaisson MJ & Tessler G. Mapping single molecule sequencing reads using basic local alignment with successive refinement (BLASR): application and theory. *BMC Bioinformatics* 2012;13:238.

Christiansen T, Orwant J, Wall L, Foy B. Programming Perl. O'Reilly Media 2012.

Darling ACE, Mau B, Blattner FR & Perna NT. Mauve: Multiple Alignment of Conserved Genomic Sequence With Rearrangements. *Genome Research*, 2004;14(7): 1394–1403.

Dohm JC, Lottaz C, Borodina T & Himmelbauer H. Substantial biases in ultra-short read data sets from high-throughput DNA sequencing. *Nucleic Acids Research* 2008;16(36): e105.

Eidam C, Poehlein A, Brenner Michael G *et al.* Complete Genome Sequence of *Mannheimia haemolytica* Strain 42548 from a Case of Bovine Respiratory Disease. *Genome announcements* 2013;1(3): e00318-13.

English AC, Richards S, Han Y *et al*. Mind the gap: upgrading genomes with Pacific Biosciences RS long-read sequencing technology. *PloS ONE*, 2012;7(11), e47768.

PeerJ Preprints NOT PEER-REVIEWED

Gentzsch W. Sun Grid Engine: Towards Creating a Compute Power Grid. In: *CCGRID '01: Proceedings of the 1st International Symposium on Cluster Computing and the Grid.* 2001;35.

Harhay GP, McVey DS, Koren S *et al.* Complete Closed Genome Sequences of Three *Bibersteinia trehalosi* Nasopharyngeal Isolates from Cattle with Shipping Fever. *Genome announcements* 2014;2(1): e00084-14.

Kammonen JI, Smolander OP, Sipilä T *et al.* Increased transcriptome sequencing efficiency with modified Mint-2 digestion-ligation protocol. *Analytical Biochemistry*, 2015;477:38-40.

de Koning AJ, Gu W, Castoe TA *et al.* Repetitive elements may comprise over two-thirds of the human genome. *PLoS Genetics*, 2011;7(12), e1002384.

Langmead B & Salzberg SL. Fast gapped-read alignment with Bowtie 2. *Nature Methods,* 2012;9(4):357-359.

Laver T, Harrison J, O'Neill PA *et al.* Assessing the performance of the Oxford Nanopore Technologies MinION. *Biomolecular Detection and Quantification* 2015;3(3):1-8.

Li H. Minimap and miniasm: fast mapping and de novo assembly for noisy long sequences. *Bioinformatics* 2016;32(14):2103-10.

Li H & Durbin R. Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics*, 2009;25(14):1754-1760.

Li YI & Copley RR. Scaffolding low quality genomes using orthologous protein sequences. *Bioinformatics* 2013;29(2): 160-165.

MATLAB and Statistics Toolbox Release 2015a, The MathWorks, Inc., Natick, Massachusetts, United States.

Magoč T & Salzberg SL. FLASH: fast length adjustment of short reads. *Bioinformatics* 2011;27(21): 2957-2963.

Margulies M, Egholm M, Altman WE *et al*. Genome sequencing in microfabricated high-density picolitre reactors. *Nature,* 2005;437:376–380.

Nakamura K, Oshima T, Morimoto T *et al.* Sequence-specific error profile of Illumina sequencers. *Nucleic Acids Research*, 2011;13(39): e90.

Piro VC, Faoro H, Weiss VA *et al*. FGAP: an automated gap closing tool. *BMC Research Notes* 2014;7:371.

Salmela L, Sahlin K, Mäkinen V & Tomescu A. Gap Filling as Exact Path Length Problem. *Journal of Computational Biology* 2016;23(5):347-61.

Schirmer M, Ijaz UZ, D'Amore R *et al.* Insight into biases and sequencing errors for amplicon sequencing with the Illumina MiSeq platform. *Nucleic Acids Research*, 2015;6(43), e37.

Slater GS & Birney E. Automated generation of heuristics for biological sequence comparison. *BMC Bioinformatics* 2005;6:31.

Smit AFA, Hubley R & Green P. 2013-2015. *RepeatMasker Open-4.0*. Retrieved from: Smit, AFA, Hubley, R & Green, P. RepeatMasker Open-4.0.
2013-2015 <http://www.repeatmasker.org>http://www.repeatmasker.org (15 November 2017, date last accessed)

Walker BJ, Abeel T, Shea T *et al.* Pilon: An Integrated Tool for Comprehensive Microbial Variant Detection and Genome Assembly Improvement. *PLoS ONE* 2014;9(11): e112963.

PeerJ Preprints NOT PEER-REVIEWED

FIGURE CAPTIONS

Figure 1. *Visualization of the FGAP* (Piro *et al.*, 2014) *and gapFinisher workflows.* **a)** FGAP is expected to find identity between a long read (blue bar) and two contigs (gray blocks) separated by a gap (N's) and then fill the gaps with the sequence. **b)** In practice, FGAP is allowed to find multiple places for one long read and nonhomologous gaps with the same sequence. **c)** gapFinisher uses the association of the long read and the scaffold reported by BLASR (Chaisson & Tessler, 2012) of SSPACE-LongRead (Boetzer & Pirovano, 2014) and ensures that each long read is only used once in gap filling.

Figure 2. *A more detailed visualization of the gapFinisher pipeline workflow.* **a)** SSPACE-LR (Boetzer & Pirovano, 2014) reports new scaffolds and these are iterated through one scaffold at a time. **b)** SSPACE-LR output shows the PacBio reads associated with the gaps in the scaffolds. **c)** These reads are then circulated through the FGAP (Piro *et al.*, 2014) pipeline with only the single scaffold as input data. This logical step prevents same PacBio reads from being used in parts of the draft genome other than the current scaffold. Measures are then taken to either **d1)** replace the unknown sequence with that of the long read **(**=fill gap) or **d2)** reject the alignment and leave the gap to the genome as is.

Figure 3. *minidot* (Li, 2016) *plots of the Mannheimia haemolytica draft genome at different stages of the assembly.* **Top top left:** Image key and reading direction. **Top left:** SPAdes contig assembly, **top center:** scaffold stage of SPAdes contigs, **top right:** gap filling stage, of the *M. haemolytica* draft genome. **Bottom left:** miniasm (Li, 2016) contig assembly, **bottom center:** scaffold stage of miniasm contigs, **bottom right:** gap filling stage, of the *M. haemolytica* draft genome. The red diagonal lines indicate continuous regions of alignment between the draft assembly and the *M. haemolytica* reference sequence. The blue diagonal lines indicate regions with inverted alignment. The red and blue dots indicate repeats and inverted repeats, respectively. Draft assembly contig/scaffold boundaries are shown as grey vertical lines. The alignment plots are provided for each of the bacterial genomes as supplementary figure S1.

PeerJ Preprints NOT PEER-REVIEWED

Figure 4. *Mauve* (Darling *et al*., 2004) *alignments of the Mannheimia haemolytica genome*. The visualizations are from **a)** before and **b)** after the scaffolding/gap filling stage. The corresponding Locally Collinear Blocks (LCB) in the three genome versions are indicated by different colors of horizontal bars. The darker lines within the blocks indicate local median similarity while the light lines show the range of local similarity values. White areas indicate low or no similarity. Blocks below the center line indicate regions that align in the reverse complement (inverse) orientation. **a):** *M. haemolytica* reference sequence (**red bar**), SPAdes (Bankevich *et al*., 2012) assembly contig sequences (**green bar**), and miniasm (Li, 2016) assembly contig sequences (**blue bar**). **b):** *M. haemolytica* reference sequence (**red bar**), and gap filled scaffolds using the SPAdes assembly contig sequences (**green bar**), and the miniasm assembly contig sequences (**blue bar**).

Figure 5. *Results from the performance benchmark of the assembly, scaffolding and gap filling tools used.* The exact values are reported in supplementary table S3. **a)** Peak random access memory (RAM) use in gigabytes (GB) in the six bacterial assemblies. **b)** Runtimes (in minutes) of the bacterial assemblies. The *F. tularensis* and *S. enterica* assembly runtimes are omitted due to the large number of CLR reads per scaffold reported in the gap filling stage (953 and 317, respectively) and the consequent long runtimes. **c)** Peak RAM use (GB) and runtimes (in hours) of the assembly, scaffolding and gap filling algorithms in the marine mammal (*Phocidae* family) genome assembly.

Figure S1. *minidot* (Li, 2016) *plots of the six bacterial genomes at different stages of the assembly*. **a)** *E. coli* K12, **b)** *E. coli* O157:H7, **c)** *B. trehalosi*, **d)** *M. haemolytica*, **e)** *F. tularensis*, **f)** *S. enterica*. **Top top left:** Image key and reading direction. **Top row** (in all subfigures): SPAdes (Bankevich *et al*., 2012) contig assembly, scaffolding and gap filling (gapFinisher) stages of the assembly. **Bottom row** (in all subfigures):

PeerJ Preprints NOT PEER-REVIEWED

miniasm (Li, 2016) contig assembly, scaffolding and gap filling (gapFinisher) stages of the assembly. The scaffolding and gap filling stages are missing for the *E. coli* K12 assembly (**a**)) since the genome was in a single sequence (i.e. closed) after miniasm.

Figure S2. *Mauve* (Darling *et al*., 2004) *alignments of the six bacterial genomes at different stages of the assembly.* **a)** *E. coli* K12, **b)** *E. coli* O157:H7, **c)** *B. trehalosi*, **d)** *M. haemolytica*, **e)** *F. tularensis*, **f)** *S. enterica*. **Top part** (in all subfigures): progressiveMauve alignment of the respective bacterial reference genome (**red bar**), the SPAdes (Bankevich *et al*., 2012) contig draft genome (**green bar**) and the miniasm (Li, 2016) contig draft genome (**blue bar**). **Bottom part** (in all subfigures): progressiveMauve alignment of the respective bacterial reference genome (**red bar**), the SPAdes assembly pipeline gap filled (gapFinisher) scaffolds (**green bar**) and the miniasm assembly pipeline gap filled (gapFinisher) scaffolds (**blue bar**). Only the contig assembly stage (top part) is shown for the *E. coli* K12 assembly (subfigure **a)**) since the genome was in a single sequence (i.e. closed) after miniasm.

PeerJ Preprints NOT PEER-REVIEWED

TABLE CAPTIONS

Table 1. *Next-generation sequencing read statistics and sequencing coverage for the sample datasets*. The bacterial data are from 2013 and originate from the Sequencing Read Archive (SRA). The sequencing chemistries were not accurately described in the original datasets but the bacterial MiSeq read data represent either Illumina (San Diego, California, United States) sequencing-by-synthesis chemistry v1 or v2. The mammal MiSeq read data are a mixture of Illumina sequencing-by-synthesis chemistry v2 and v3. The bacterial Pacific Biosciences (Menlo Park, California, United States) RS reads represent PacBio SMRT sequencing chemistries that are earlier than P4-C2 and the mammal PacBio RS reads are a mixture of PacBio SMRT sequencing chemistries P5-C3 and P6-C4.

Table 2. *De novo assembly, scaffolding and gap filling statistics for the model genomes.* For clarity, only the most significant statistics are shown here and the full statistics provided as supplementary table S3.

Table S3. *All de novo assembly, scaffolding and gap filling statistics for the model genomes*. In addition, the performance benchmark statistics are included in the last three columns.

Table S4. *Gap filling data used and FGAP* (Piro *et al*., 2014) *default test results reported for an unpublished draft genome of a marine mammal from the Phocidae* family. An admittedly small number of Pacific Biosciences RS II platform circular consensus reads (*reads of insert*) with summed length of about 280 kbp filled 45.5 million unknown bases in the draft genome, a result reported by FGAP with the default alignment settings. By further changing the FGAP command line options, one can adjust the number of BLAST (Altschul *et al*., 1990) hits returned to perform the alignment. By default, this is 200 hits. We further ran another test, where we reduced this amount to two so that only the best two BLAST hits would be considered in the gap filling. Still, more than 4.5 million N's were reportedly filled with our test set, a far greater number of bases than contained by the original read data used. The default BLAST alignment parameters of FGAP for opening and extending a gap are both set to the value 1. The default values in command line applications of BLAST for opening and extending a gap are set as 5 and 2, respectively, as written in the BLAST user guide by Camacho *et al*. Thus, depending on the total score of the alignment, gap opening in the alignment is up to 80% and gap extension up to 50% more likely than the BLAST defaults. The minimum raw score of a BLAST hit in FGAP is set to value 25, a typical raw score value of highly dissimilar sequences irrespective of the gap penalty parameters. Moreover, a maximum of 200 BLAST results may be returned for a 70 percent identity in alignment length of 300 bp by default. In general, the default parameters of FGAP appear too lenient and may fill gaps based on alignments that are incorrect and may appear multiple times where unique alignments are desired.

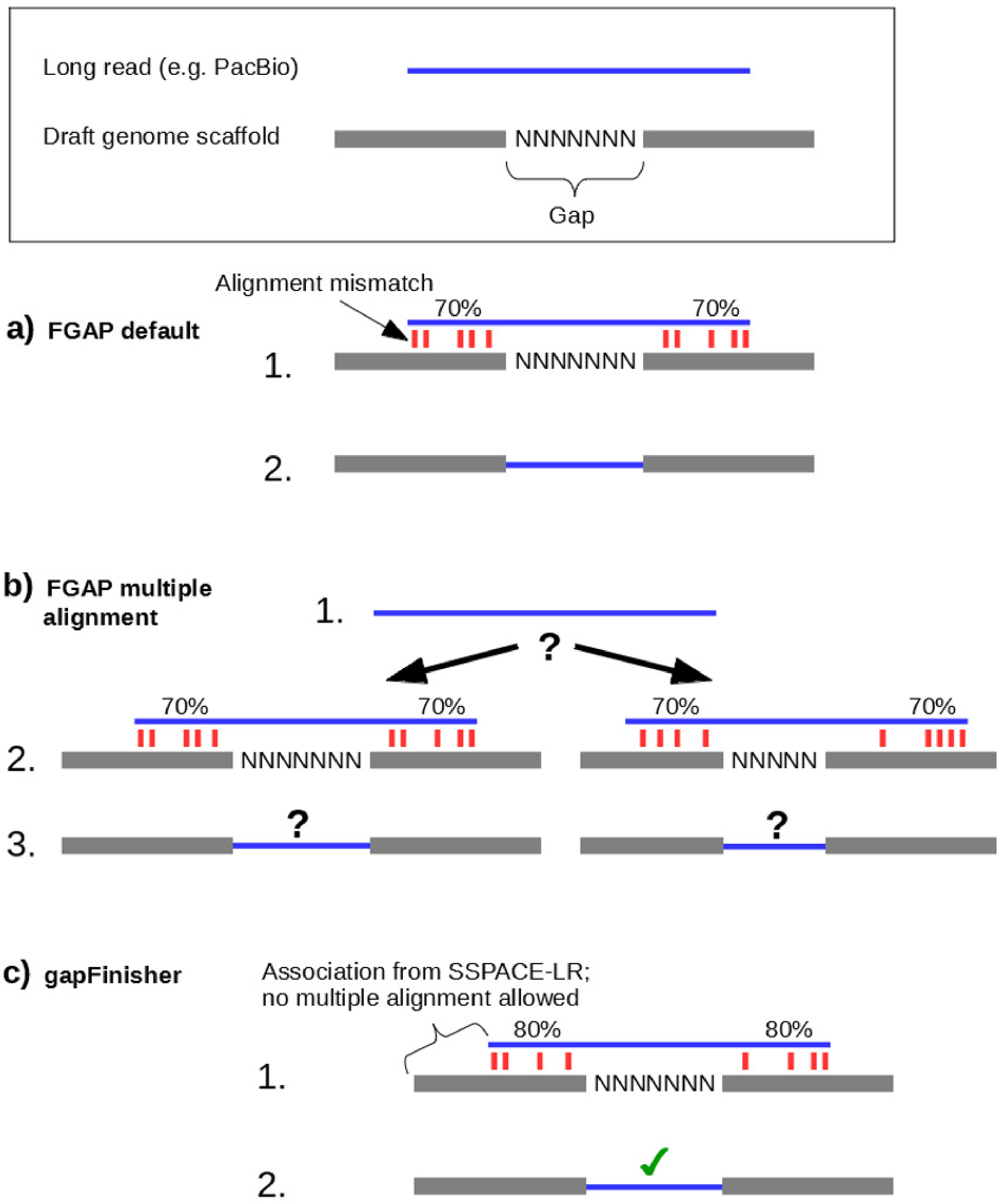
PeerJ Preprints NOT PEER-REVIEWED

Figure 1

Visualization of the FGAP (Piro *et al.*, 2014) and gapFinisher workflows.

a) FGAP is expected to find identity between a long read (blue bar) and two contigs (gray blocks) separated by a gap (N's) and then fill the gaps with the sequence. **b)** In practice, FGAP is allowed to find multiple places for one long 5 read and nonhomologous gaps with the same sequence. **c)** gapFinisher uses the association of the long read and the scaffold reported by BLASR (Chaisson & Tessler, 2012) of SSPACE-LongRead (Boetzer & Pirovano, 2014) and ensures that each long read is only used once in gap filling.

PeerJ Preprints NOT PEER-REVIEWED



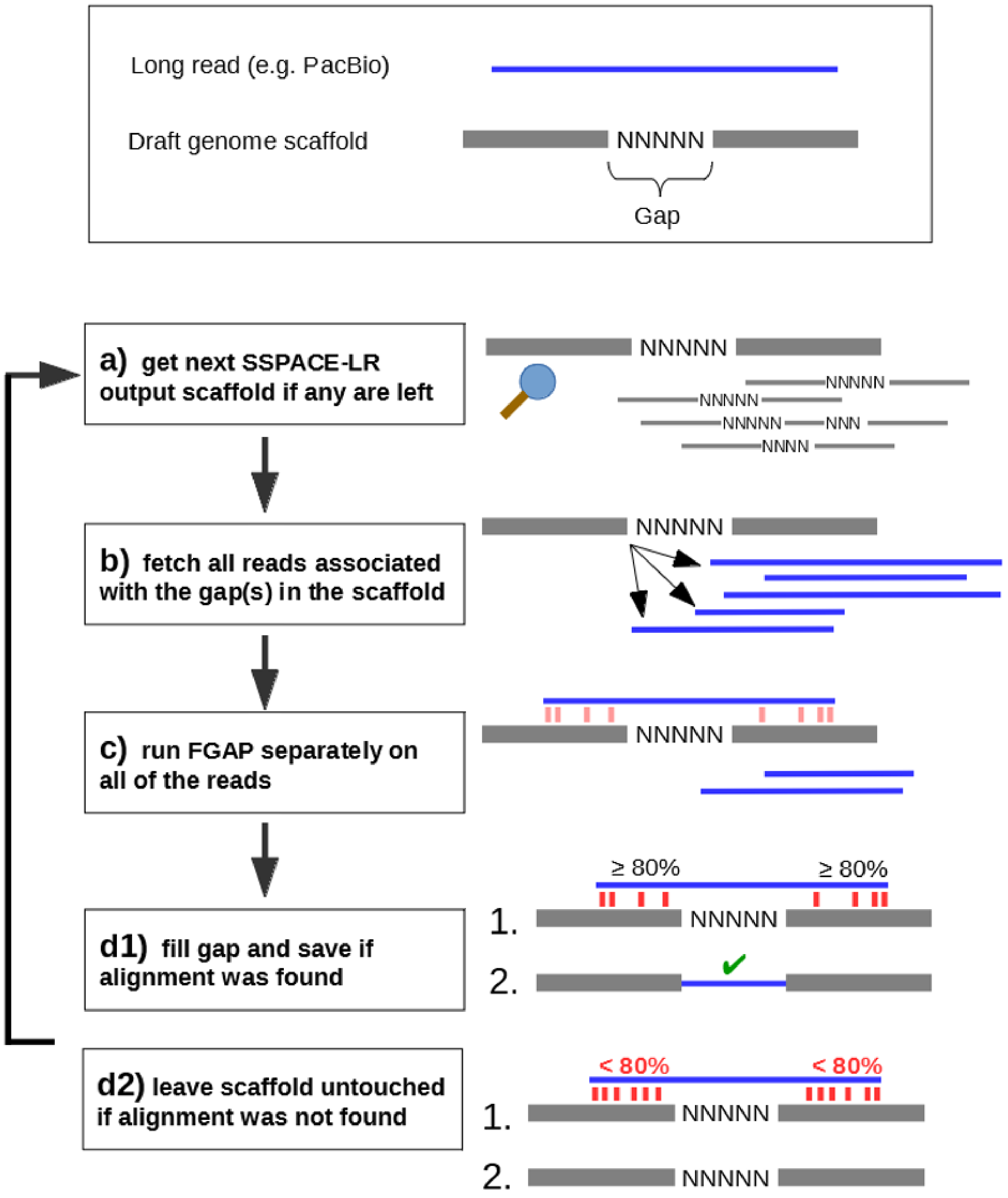
PeerJ Preprints NOT PEER-REVIEWED

Figure 2

A more detailed visualization of the gapFinisher pipeline workflow.

a) SSPACE-LR (Boetzer & Pirovano, 2014) reports new scaffolds and these are iterated through one scaffold at a time. **b)** SSPACE-LR output shows the long reads associated with the gaps in the scaffolds. **c)** These reads are then circulated through the FGAP (Piro *et al.*, 2014) pipeline with only the single scaffold as input data. This logical step prevents same reads from being used in parts of the draft genome other than the current scaffold. Measures are then taken to either **d1)** replace the unknown sequence with that of the long read (=*fill gap*) or **d2)** reject the alignment and leave the gap to the genome as is.

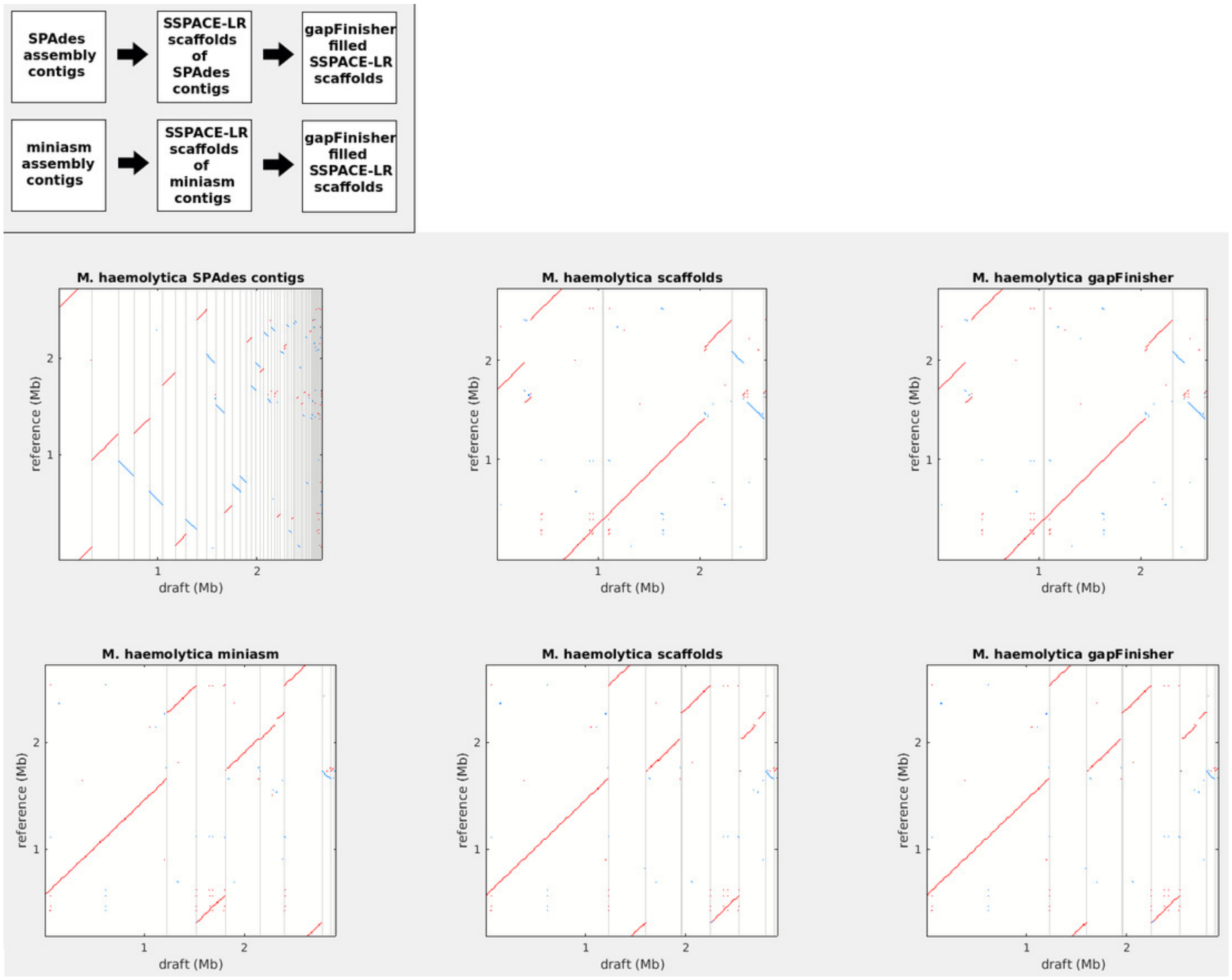
PeerJ Preprints NOT PEER-REVIEWED


Long read (e.g. PacBio)
Draft genome scaffold
NNNNN
Gap
a) get next SSPACE-LR output scaffold if any are left
NNNNN
NNNN
NNN
b) fetch all reads associated with the gap(s) in the scaffold
c) run FGAP separately on all of the reads
d1) fill gap and save if alignment was found
≥ 80%
≥ 80%
1.
2.
d2) leave scaffold untouched if alignment was not found
< 80%
< 80%
1.
2.

PeerJ Preprints NOT PEER-REVIEWED

Figure 3

minidot (Li, 2016) plots of the *Mannheimia haemolytica* draft genome at different stages of the assembly.



PeerJ Preprints NOT PEER-REVIEWED

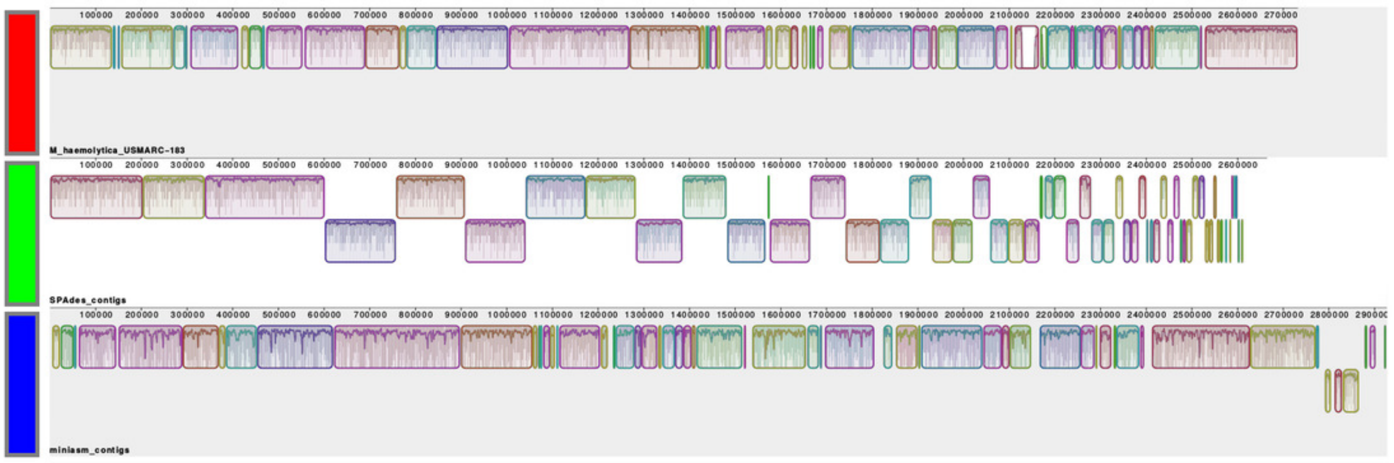
Figure 4

Mauve (Darling *et al.*, 2004) alignments of the *Mannheimia haemolytica* genome.

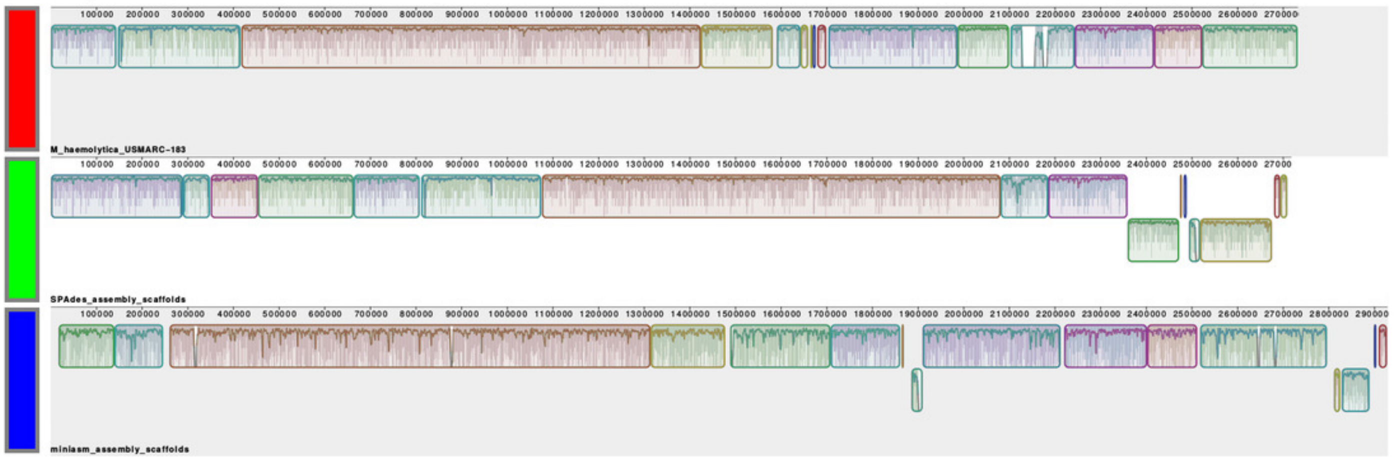
The visualizations are from **a)** before and **b)** after the scaffolding/gap filling stage. The corresponding Locally Collinear Blocks (LCB) in the three genome versions are indicated by different colors of horizontal bars. The darker lines within the blocks indicate local median similarity while the light lines show the range of local similarity values. White areas indicate low or no similarity. Blocks below the center line indicate regions that align in the reverse complement (inverse) orientation. **a)** *M. haemolytica* reference sequence (red bar), SPAdes (Bankevich *et al.*, 2012) assembly contig sequences (green bar), and miniasm (Li, 2016) assembly contig sequences (blue bar). **b)** *M. haemolytica* reference sequence (red bar), and gap filled scaffolds using the SPAdes assembly contig sequences (green bar), and the miniasm assembly contig sequences (blue bar).

PeerJ Preprints NOT PEER-REVIEWED

a)



b)

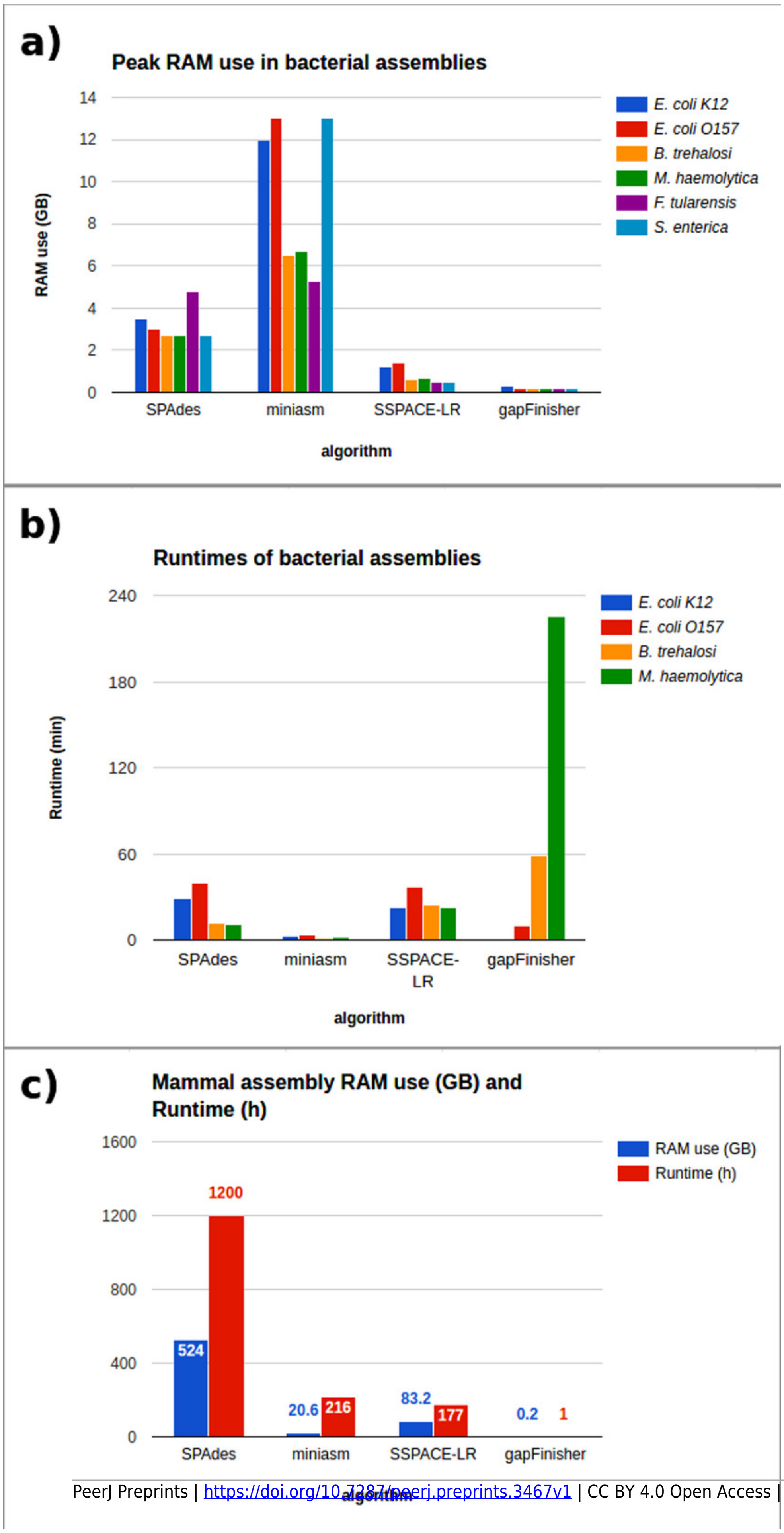


PeerJ Preprints NOT PEER-REVIEWED

Figure 5

Results from the performance benchmark of the assembly, scaffolding and gap filling tools used.

PeerJ Preprints NOT PEER-REVIEWED



PeerJ Preprints NOT PEER-REVIEWED

Table 1(on next page)

Next-generation sequencing read statistics and sequencing coverage for the sample datasets.

PeerJ Preprints NOT PEER-REVIEWED

	Illumina MiSeq paired end reads (100X)			PacBio RS reads (200X)		
Organism	**Total reads**	**Total bases**	**Avg. read length**	**Total reads**	**Total bases**	**Avg. read length**
E. coli K12 MG1655	3,046,358	460,000,058	151	383,482	929,129,994	2,422
E. coli O157:H7	3,794,862	548,505,079	144	403,919	1,100,295,861	2,724
B. trehalosi	1,718,212	249,216,010	145	205,096	499,939,066	2,437
M. haemolytica	1,724,414	249,368,724	144	175,953	531,234,319	3,019
F. tularensis	926,716	199,169,591	214	176,376	399,767,452	2,266
S. enterica	1,943,848	279,774,061	143	394,699	1,000,244,555	2,534
Organism	**Illumina MiSeq reads (25X)**			**PacBio RS reads (50X)**		
Mammal	329,484,322	62,120,890,467	188	17,695,174	146,961,409,902	8,305

PeerJ Preprints NOT PEER-REVIEWED

Table 2(on next page)

De novo assembly, scaffolding and gap filling statistics for the model genomes.

PeerJ Preprints NOT PEER-REVIEWED

Organism	Tool	Sequences Expected	Final	Sum (bp)	N50 length	Gaps (bp)	Gap %	LCBs*	Indels	Inversions
E. coli K12 MG1655	SPAdes	1	35	4,661,027	4,640,853	0	0.00%	4	0	2
	SSPACE-LR	1	34	4,661,028	4,641,005	1	0.00%	4	0	2
	gapFinisher			4,661,028	4,641,005	1	0.00%			
	miniasm	1	**1**	4,793,967	4,793,967	0	0.00%	4	0	2
	SSPACE-LR	-	-	-	-	-		-	-	-
	gapFinisher			-	-	-				
E. coli O157:H7	SPAdes	10	87	5,547,646	3,323,349	3	0.00%	27	5	5
	SSPACE-LR	10	50	5,568,199	3,323,349	18486	0.33%	23	4	3
	gapFinisher			5,568,974	3,323,349	**5750**	0.10%			
	miniasm	10	25	5,898,494	537,223	0	0.00%	27	7	7
	SSPACE-LR	10	16	5,908,008	61,209	9514	0.16%	23	6	5
	gapFinisher			5,907,537	61,209	**2495**	0.04%			
B. trehalosi	SPAdes	1	51	2,376,258	274,711	2	0.00%	30	7	8
	SSPACE-LR	1	12	2,401,287	438,635	2804	0.12%	21	6	3
	gapFinisher			2,401,265	438,599	**401**	0.02%			
	miniasm	1	17	2,510,680	221,473	0	0.00%	30	3	7
	SSPACE-LR	1	10	2,520,563	377,524	9883	0.39%	21	4	5
	gapFinisher			2,521,341	37,752	**4920**	0.20%			
M. haemolytica	SPAdes	1	112	2,664,209	101,958	35	0.00%	40	5	7
	SSPACE-LR	1	17	2,718,326	1,073,880	13504	0.50%	13	4	2
	gapFinisher			2,717,906	1,073,740	**4498**	0.17%			
	miniasm	1	10	2,926,783	378,549	0	0.00%	40	4	1
	SSPACE-LR	1	8	2,928,560	378,549	1777	0.06%	13	4	1
	gapFinisher			2,928,834	378,549	**1155**	0.04%			
F. tularensis	SPAdes	3	135	1,807,729	25,688	0	0.00%	80	12	18
	SSPACE-LR	3	58	1,855,045	56,838	23176	1.25%	42	9	9
	gapFinisher			1,851,864	56,791	**11254**	0.61%			
	miniasm	3	20	2,000,228	15,305	0	0.00%	80	5	4
	SSPACE-LR	3	9	2,021,978	426,098	21750	1.08%	42	4	4
	gapFinisher			2,021,618	425,969	**16843**	0.83%			
S. enterica	SPAdes	4	217	4,982,997	153,597	655	0.01%	49	4	10
	SSPACE-LR	4	94	5,026,381	1,020,795	10050	0.20%	27	3	6
	gapFinisher			5,028,882	1,020,937	**2917**	0.06%			
	miniasm	4	16	5,373,212	735,723	0	0.00%	49	5	2
	SSPACE-LR	4	10	5,384,667	874,322	11455	0.21%	27	5	2
	gapFinisher			5,384,057	874,322	**4641**	0.09%			
Mammal	SPAdes	Unkn.	236592	2,253,617,865	19,739	0	0.00%	-	-	-
	SSPACE-LR	Unkn.	10143	2,462,623,627	599,108	10136364	0.41%	-	-	-
	gapFinisher			2,466,785,189	601,444	**6945295**	0.28%			
	miniasm	Unkn.	1314	2,460,097,408	8,668,858	0	0.00%	-	-	-

PeerJ Preprints NOT PEER-REVIEWED

S-LR / 1st	Unkn.	1115	2,460,626,045	9,381,548	528637	0.02%	-	-	-
gapF / 1st			2,460,674,964	9,381,548	**351878**	0.01%			
S-LR / 2nd	Unkn.	1008	2,460,965,525	9,562,827	642439	0.03%	-	-	-
gapF /2nd			2,460,993,827	9,562,827	**616617**	0.03%			

*LCB = Locally Collinear Block of progressiveMauve (27) alignment.