Illumina TruSeq synthetic long-reads empower *de novo* assembly and resolve complex, highly repetitive transposable elements

Rajiv C. McCoy[1], Ryan W. Taylor[1], Timothy A. Blauwkamp[2], Joanna L. Kelley[3], Michael Kertesz[4], Dmitry Pushkarev[5], Dmitri A. Petrov*[1] and Anna-Sophie Fiston-Lavier*[1,6]

[1]Department of Biology, Stanford University, Stanford, California 94305, USA
[2]Illumina Inc., San Diego, California 92122, USA
[3]School of Biological Sciences, Washington State University, Pullman, Washington 99164, USA
[4]Department of Bioengineering, Stanford University, Stanford, California 94035, USA
[5]Department of Physics, Stanford University, Stanford, California 94035, USA
[6]Institut des Sciences de l'Evolution-Montpellier, Montpellier, Cedex 5, France
Corresponding authors: Rajiv C. McCoy rmccoy@stanford.edu
Dmitri Petrov dpetrov@stanford.edu, and Anna-Sophie Fiston-Lavier asfiston@univ-montp2.fr
*DAP and ASFL are joint senior authors on this work.

Supplemental materials

Generation of TruSeq synthetic long-reads from short read data

Short Read Pre-Processing

Prior to the assembly of the long reads, the short reads in every well are pre-filtered to correct for errors which could lead to misassemblies. Reads that do not have a sufficient stretch of high-quality bases are filtered. Low-quality ends of remaining bases are trimmed (hard-clipped). Read pairs that appear to read through one another, and thus potentially contain adapter sequence on the 3 end(s) of one or both reads, are modified as follows. The first read is trimmed of bases that appear to extend beyond the second read, and the second read is discarded, resulting in an unpaired read that should have had any 3 adapter sequence clipped off. If the trimmed reads in a pair are shorter than 30bp, the pair is discarded. If one read in a pair is shorter than 30bp, and the second read longer than 50bp, the longer read is kept. Adapter sequences are removed and the end-marker sequences identified and trimmed, and reads containing end-marker sequences are tagged for downstream use in the pipeline.

Assembly of Contigs

The assembly module consists of several steps: digital normalization, read error correction, graph construction, and clean-up using paired end reads. These steps are described in more detail in the following sections.

Digital Normalization

Due to bias introduced during PCR, the read coverage among input fragments in the sample can vary greatly. In order to normalize coverage variation across fragments (which improves the accuracy of the assembly as well as the computational performance of the algorithm), digital normalization methods outlined by Brown et al (Brown et al., 2012) are used. The digital normalization process smooths out highly biased sequence coverage by removing specific over-represented sequences. Coverage is normalized such that the highest coverage fragments are approximately 40x.

Error Correction

Following digital normalization, an error correction step is performed using an overlap- based method. The aim of this step is to correct PCR and sequencing artifacts which introduce false base substitutions or indels. At a high level, it operates as follows. An index of all k-mers of length 31 in the reads is constructed (the k-mer hash). For each read, k-mers in the read are compared to the index to find the set of reads which share the same k-mer. Matches to candidate overlapping reads are extended using semi-banded global alignment, and those which have a match length of at least 31 bases and share 95% identity, are retained. Multiple sequence alignment (MSA) of the set of overlapping reads is performed. Using both the base quality scores of the reads and the results of the MSA, a consensus sequence for the read is generated.

Graph Construction

The main assembly step is performed using the String Graph Assembler (SGA) (Simpson and Durbin, 2012), which is an overlap-based assembly method. In the first stage, SGA uses a k-mer overlap size of 31 to create a graph with reads as vertices and k-mer overlaps as edges.

After the construction of an initial graph, the next step of the algorithm is to clean the graph and remove spurious edges using several heuristics. The algorithm requires that paths in the graph are supported by paired-end reads. It checks for the existence of a path linking the two reads of a read pair within the expected insert size distribution (500 bp by default). Any edges in the graph which do not support read pairs are removed. In addition, tips and bubbles in the read graph, which normally occur during de novo assembly, are cleaned up using standard graph cleaning methods.

Scaffolding Contigs to Assemble Long Reads

The next stage in the pipeline is scaffolding, the goal of which is to use paired-end information to place and orient the contigs generated in the previous step and fill in gaps between contigs. The method employed in the long reads pipeline is based on the scaffolding method employed in the original SGA assembler, and the user is referred to Simpson and Durbin (2012) for further details.

In brief, scaffolding is accomplished by re-aligning the input short reads to the contigs using BWA aligner (Li and Durbin, 2009), and using the paired-end alignments to infer scaffold structure. The link between two contigs is made when 2 or more paired reads map such that read 1 from a read pair maps to one contig and read 2 from the same read pair maps to the other. The orientation of the contigs relative to one another is also inferred from the orientation of the read-pairs. In addition, the end-marker sequences are used to help guide and constrain the construction of our scaffold graph

Gap Filling

The next step of this module is to fill in scaffold gaps where possible in order to resolve repeats. In this step, we use the input short reads, making use of the FM index computed during the contig assembly. We begin by finding the highest scoring read which matches the end of one of the contigs, and continue to chain together reads iteratively. If a chain is found that overlaps another contig in the same scaffold, the consensus is retained and the gap filled with this sequence.

Assembly QC and Correction

The final stage of the analysis pipeline involves verification of the scaffolds and error correction. The short read data is again aligned against the scaffolds generated in the previous step using BWA aligner (3). Based on the alignments, the scaffolds are corrected for single-nucleotide errors and broken into smaller scaffolds should there be only partial alignment support. Quality scores for the final long reads are also estimated from the alignments.

Breaking Scaffolds

The short reads used during the Long Reads assembly are aligned to the scaffolds. The alignments are searched for read pairs in which one read aligns and the other one does not. Unaligned reads are re-aligned, and reads that are overlapping or running into scaffold gaps are counted and computed. In order to determine

whether or not to break a scaffold gap, Illumina computes the following formula:

```
   sqrt(0.3+(reads aligning to mid point of gap on fwd strand)*(0.3+
(reads aligning to mid point of gap on rev strand)))/(total
      number of reads in gap)
```

If this ratio is smaller than 0.1, the gap is left as it is; if it is larger, the scaffold is broken at this gap. If there are only few reads or none, the scaffold for the region is left as it is.

Q-scores

From the alignments of short reads to the scaffolds, a pileup file is generated which provides the base quality scores of the aligned reads at each position in a scaffold. The quality score at each scaffold position is then estimated from the read base qualities as follows:

- Remove Ns and indels from the pile-up.
- If coverage > 5 and all nucleotides at this position agree and set Q-score to max of pileup.
- If $< 5\%$ mismatches or > 3 matches, set Q-score to mean of pileup.
- If all of the above steps fail, look at the most frequently occurring nucleotide in the pileup and the second most frequent one. Compute the posterior probability of most frequent base given the quality scores. This includes some correction factors from a PCR error rate model. Do the same for the second most frequent nucleotide. Choose the nucleotide with the highest posterior probability and compute the q-score from this probability.

Pre-assembly quality control

Assessment of contamination

We assessed the degree of contamination with BLASTN (Altschul et al., 1997) by searching against the NCBI nucleotide database (see Methods). The degree of contamination in the TruSeq synthetic long-read libraries was low, with 99.8% (953,797) of reads having top hits to *D. melanogaster* reference sequences. We note that the number of synthetic long-reads with top BLASTN hits to *D. melanogaster* is lower than the number that map to the reference genome with BWA-MEM for several reasons. First, a small number of reads derived from regions of extremely low divergence erroneously map to other *Drosophila* species. Second,

the "Uextra" scaffolds likely contain some contamination from other species as described in the release notes: `http://www.fruitfly.org/data/sequence/README.RELEASE5`. Finally, for a very small number of reads, large proportions of the reads lengths are clipped by BWA-MEM with only small subsequences that align. Based on the BLASTN results, the most abundant contaminant reads had top matches to known symbionts of *D. melanogaster*, including acetic acid bacteria from the genera *Gluconacetobacter*, *Gluconobacter*, and *Acetobacter* (Table S2). Because contamination was extremely rare and because we could not exclude that sequences with no BLAST hits may correspond to fly-derived sequences not previously assembled in the reference genome, we included all sequences in downstream analyses.

Genome assembly from TruSeq synthetic long-reads

Assembly with the Celera Assembler

The following Celera Assembler parameters are roughly based on those recommended for PacBio consensus-corrected reads: `http://sourceforge.net/apps/mediawiki/wgs-assembler/index.php?title=PBcR#Assembly_of_Corrected_Sequences`. Based on our goal of assembling separate copies of TEs, however, we elected to use a greater k-mer size and k-mer threshold to increase specificity and reduce the number of false joins (which could generate chimeric sequences).

```
unitigger=bogart
merSize=31
merThreshold=auto*2
ovlMinLen=800
obtErrorRate=0.03
obtErrorLimit=4.5
ovlErrorRate=0.03
utgErrorRate=0.015
utgGraphErrorRate=0.015
utgGraphErrorLimit=0
utgMergeErrorRate=0.03
utgMergeErrorLimit=0
```

The bogart unitigger, which is recommended for Illumina data or Illumina data in combination with other data types, and is also employed in the PacBio corrected read assembly pipeline. We required overlap of

at least 800 bp in order to merge across reads, a parameter that further increases overlap specificity. Error rates are set substantially lower than the default options, given the low observed rate of mismatches to the reference genome in the TruSeq synthetic long reads as well as the fact that we sequenced a highly inbred strain of *D. melanogaster*. These parameters are intentionally conservative to avoid the erroneous merging of contigs at identical repeats. Modifications to these parameters may increase overlap sensitivity and achieve greater contig lengths, but likely at the expense of mis-assembly. Assembly for species with higher rates of polymorphism would require error rates to be set higher to avoid separate assembly of individual haplotypes.

Contig merging with Minimus2

NUCmer (Delcher et al., 2002; Kurtz et al., 2004) alignment to the reference genome revealed that in some cases, the Celera Assembler produced contigs with ends with long stretches (>1 Kbp) of perfect sequence identity. As we demonstrated in the main text, many of these cases represent regions of low coverage in synthetic long reads, where data were insufficient to support a join. We therefore used the simple overlap-based assembler Minimus2 to generate supercontigs from the contigs output by Celera. The parameters used for this assembly were:

```
REFCOUNT= 0
MINID   = 99.9
OVERLAP = 800
MAXTRIM = 1000
WIGGLE  = 15
CONSERR = 0.01
```

The parameter REFCOUNT=0 means that the assembler performs all vs. all alignment of the contigs, rather than merging two separate assemblies (a common application of Minimus2). We required a stringent sequence identity of 99.9% with at least 800 bp of overlap at the contig ends to allow a join, thereby avoiding false contig joins.

Assembly assessment with NUCmer alignment

Alignment of assembled contigs to the high quality reference genome was performed with NUCmer (version 3.23) (Delcher et al., 2002; Kurtz et al., 2004), and the resulting alignment file was filtered according to guidelines described in the documentation: `http://mummer.sourceforge.net/manual/#mappingdraft`.

```
nucmer ref.fasta qry.fasta

delta-filter -q out.delta > out.q.delta
```

We required alignments to have at least 99% identity to the reference for at least 1000 bp.

```
show-coords -THrcl out.q.delta | \
     awk '{if ($7>99 && $5>1000) print $12"\t"$1"\t"$2"\t"$13"\t"$11}' > nucmer.bed
```

We then used BEDTools (version 2.19.1) (Quinlan and Hall, 2010) to merge across perfectly adjacent or partially overlapping alignments.

```
bedtools merge -i nucmer.bed > nucmer.merge.bed
```

Alignment statistics reported in Table 2 were then produced as follows:

```
for i in X 2L 2R 3L 3R 4 XHet 2LHet 2RHet 3LHet 3RHet YHet M U
do
     echo $i
     # count the alignments
     cat nucmer.bed | awk -v i=$i '{if ($1==i) print}' | cut -f4 | sort | uniq | wc -l

     # count the gaps
 bedtools complement -g reference.genome -i nucmer.merge.bed > nucmer.complement.bed
 cat nucmer.complement.bed | awk -v i=$i '{if ($1==i) print}' | wc -l

     # sum the total aligned length
     cat nucmer.merge.bed | awk -v i=$i '{if ($1==i) print $3-$2}' | \
          awk '{sum+=$1} END {print sum}'
     printf "\n\n"
done
```

The same alignment file (.delta) is also analyzed to define the search space for TEs and genes: `https://github.com/rmccoy7541/assess-assembly`. The steps in the pipeline are as follows:

- Map contigs to the reference genome with NUCmer, extracting only the optimal mapping of each contig to one position in the reference.
- Check whether both the start and end boundary of the gene or TE fall within the same aligned contig.
- If so, perform local alignment between the reference sequence of the gene or TE and the corresponding aligned sequence.
- Calculate the percent identity and the proportion of the gene or TE's length that was assembled and aligned.

References

Altschul, S. F., Madden, T. L., Schäffer, A. A., Zhang, J., Zhang, Z., Miller, W., and Lipman, D. J. (1997). Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research* **25**:3389–3402.

Brown, C. T., Howe, A., Zhang, Q., Pyrkosz, A. B., and Brom, T. H. (2012). A Reference-Free Algorithm for Computational Normalization of Shotgun Sequencing Data. *arXiv.org* .

Delcher, A. L., Phillippy, A., Carlton, J., and Salzberg, S. L. (2002). Fast algorithms for large-scale genome alignment and comparison. *Nucleic Acids Research* **30**:2478–2483.

Kurtz, S., Phillippy, A., Delcher, A. L., Smoot, M., Shumway, M., Antonescu, C., and Salzberg, S. L. (2004). Versatile and open software for comparing large genomes. *Genome Biology* **5**:R12.

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

Quinlan, A. R. and Hall, I. M. (2010). BEDTools: a flexible suite of utilities for comparing genomic features. *Bioinformatics* **26**:841–842.

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