

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

Rajiv C. McCoy¹, Ryan W. Taylor¹, Timothy A. Blauwkamp², Joanna L. Kelley³, Michael Kertesz⁴, Dmitry Pushkarev⁵, Dmitri A. Petrov*¹ and Anna-Sophie Fiston-Lavier*^{1,6}

¹Department of Biology, Stanford University, Stanford, California 94305, USA

²Illumina Inc., San Diego, California 92122, USA

³School of Biological Sciences, Washington State University, Pullman, Washington 99164, USA

⁴Department of Bioengineering, Stanford University, Stanford, California 94035, USA

⁵Department of Physics, Stanford University, Stanford, California 94035, USA

⁶Institut 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.

1 Supplemental materials

2 Generation of TruSeq synthetic long-reads from short read data

3 Short Read Pre-Processing

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

14 Assembly of Contigs

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

17 **Digital Normalization**

18 Due to bias introduced during PCR, the read coverage among input fragments in the sample can vary greatly.
19 In order to normalize coverage variation across fragments (which improves the accuracy of the assembly as well
20 as the computational performance of the algorithm), digital normalization methods outlined by Brown et al.
21 (2012) are used. The digital normalization process smooths out highly biased sequence coverage by removing
22 specific over-represented sequences. Coverage is normalized such that the highest coverage fragments are
23 approximately 40×.

24 **Error Correction**

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

33 **Graph Construction**

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

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

43 **Scaffolding Contigs to Assemble Long Reads**

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

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

54 **Gap Filling**

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

60 **Assembly QC and Correction**

61 The final stage of the analysis pipeline involves verification of the scaffolds and error correction. The short
62 read data is again aligned against the scaffolds generated in the previous step using BWA aligner (Li and
63 Durbin, 2009). Based on the alignments, the scaffolds are corrected for single-nucleotide errors and broken
64 into smaller scaffolds should there be only partial alignment support. Quality scores for the final long reads
65 are also estimated from the alignments.

66 **Breaking Scaffolds**

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

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

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

74 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
75 there are only few reads or none, the scaffold for the region is left as it is.

76 Q-scores

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

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

88 Pre-assembly quality control

89 Assessment of contamination

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

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

104 **Genome assembly from TruSeq synthetic long-reads**

105 **Assembly with the Celera Assembler**

106 The following Celera Assembler parameters are roughly based on those recommended for PacBio consensus-
107 corrected reads: [http://sourceforge.net/apps/mediawiki/wgs-assembler/index.php?title=PBcR#Assembly_](http://sourceforge.net/apps/mediawiki/wgs-assembler/index.php?title=PBcR#Assembly_of_Corrected_Sequences)
108 [of_Corrected_Sequences](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
109 use a greater k-mer size and k-mer threshold to increase specificity and reduce the number of false joins
110 (which could generate chimeric sequences).

```
111 unitigger=bogart  
112 merSize=31  
113 merThreshold=auto*2  
114 ovlMinLen=800  
115 obtErrorRate=0.03  
116 obtErrorLimit=4.5  
117 ovlErrorRate=0.03  
118 utgErrorRate=0.015  
119 utgGraphErrorRate=0.015  
120 utgGraphErrorLimit=0  
121 utgMergeErrorRate=0.03  
122 utgMergeErrorLimit=0
```

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

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

132 **Contig merging with Minimus2**

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

```
139 REFCOUNT= 0  
140 MINID    = 99.9  
141 OVERLAP  = 800  
142 MAXTRIM  = 1000  
143 WIGGLE   = 15  
144 CONSERR  = 0.01
```

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

149 **Assembly assessment with NUCmer alignment**

150 Alignment of assembled contigs to the high quality reference genome was performed with NUCmer (version
151 3.23) (Delcher et al., 2002; Kurtz et al., 2004), and the resulting alignment file was filtered according to
152 guidelines described in the documentation: <http://mummer.sourceforge.net/manual/#mappingdraft>.

```
153 nucmer ref.fasta qry.fasta
```

```
154
```

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

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

```
157 show-coords -THrcl out.q.delta | \
```

```
158     awk '{if ($7>99 && $5>1000) print $12"\t"$1"\t"$2"\t"$13"\t"$11}' > nucmer.bed
```

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

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

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

```
163
```

```
164 for i in X 2L 2R 3L 3R 4 XHet 2LHet 2RHet 3LHet 3RHet YHet M U
```

```
165 do
```

```
166     echo $i
```

```
167     # count the alignments
```

```
168     cat nucmer.bed | awk -v i=$i '{if ($1==i) print}' | cut -f4 | sort | uniq | wc -l
```

```
169
```

```
170     # count the gaps
```

```
171     bedtools complement -g reference.genome -i nucmer.merge.bed > nucmer.complement.bed
```

```
172     cat nucmer.complement.bed | awk -v i=$i '{if ($1==i) print}' | wc -l
```

```
173
```

```
174     # sum the total aligned length
```

```
175     cat nucmer.merge.bed | awk -v i=$i '{if ($1==i) print $3-$2}' | \
```

```
176         awk '{sum+=$1} END {print sum}'
```

```
177     printf "\n\n"
```

```
178 done
```

```
179
```

180 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:

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

189 References

- 190 Altschul, S. F., Madden, T. L., Schäffer, A. A., Zhang, J., Zhang, Z., Miller, W., and Lipman, D. J. (1997).
191 Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids*
192 *Research* **25**:3389–3402.
- 193 Brown, C. T., Howe, A., Zhang, Q., Pyrkosz, A. B., and Brom, T. H. (2012). A Reference-Free Algorithm
194 for Computational Normalization of Shotgun Sequencing Data. *arXiv.org* .
- 195 Delcher, A. L., Phillippy, A., Carlton, J., and Salzberg, S. L. (2002). Fast algorithms for large-scale genome
196 alignment and comparison. *Nucleic Acids Research* **30**:2478–2483.
- 197 Kurtz, S., Phillippy, A., Delcher, A. L., Smoot, M., Shumway, M., Antonescu, C., and Salzberg, S. L. (2004).
198 Versatile and open software for comparing large genomes. *Genome Biology* **5**:R12.
- 199 Li, H. and Durbin, R. (2009). Fast and accurate short read alignment with Burrows-Wheeler transform.
200 *Bioinformatics* **25**:1754–1760.
- 201 Quinlan, A. R. and Hall, I. M. (2010). BEDTools: a flexible suite of utilities for comparing genomic features.
202 *Bioinformatics* **26**:841–842.
- 203 Simpson, J. T. and Durbin, R. (2012). Efficient de novo assembly of large genomes using compressed data
204 structures. *Genome Research* **22**:549–556.