nature biotechnology

Assembling large genomes with single-molecule sequencing and locality-sensitive hashing

Konstantin Berlin[1–3,6], Sergey Koren[4,6], Chen-Shan Chin[5], James P Drake[5], Jane M Landolin[5] & Adam M Phillippy[4]

Long-read, single-molecule real-time (SMRT) sequencing is routinely used to finish microbial genomes, but available assembly methods have not scaled well to larger genomes. We introduce the MinHash Alignment Process (MHAP) for overlapping noisy, long reads using probabilistic, locality-sensitive hashing. Integrating MHAP with the Celera Assembler enabled reference-grade *de novo* assemblies of *Saccharomyces cerevisiae*, *Arabidopsis thaliana*, *Drosophila melanogaster* and a human hydatidiform mole cell line (CHM1) from SMRT sequencing. The resulting assemblies are highly continuous, include fully resolved chromosome arms and close persistent gaps in these reference genomes. Our assembly of *D. melanogaster* revealed previously unknown heterochromatic and telomeric transition sequences, and we assembled low-complexity sequences from CHM1 that fill gaps in the human GRCh38 reference. Using MHAP and the Celera Assembler, single-molecule sequencing can produce *de novo* near-complete eukaryotic assemblies that are 99.99% accurate when compared with available reference genomes.

Genome assembly is the process of reconstructing a genome from a collection of short sequencing reads and is an integral step in any genome project[1,2]. Unlike resequencing projects, *de novo* assembly is performed without the aid of a reference genome; rather, the genome is reconstructed from scratch. An accurate reconstruction is crucial, as both the continuity and base accuracy of an assembly can affect the results of all downstream analyses[3]. However, repetitive sequences make assembly difficult when the repeat length exceeds the read length[4]. Most high-throughput sequencing methods generate sequencing reads of only a few hundred base pairs, which is shorter than most common repeats. Although short reads are sufficient for many analyses, they are not sufficient for resolving major repeat families in either microbial or eukaryotic genomes, leading to fractured and incomplete assemblies[5].

Recent advances in single-molecule sequencing technologies have promised reads hundreds of fold longer than second-generation methods[6,7]. Most notably, Pacific Biosciences' SMRT sequencing was the first commercially available long-read technology[7]. Using a DNA polymerase anchored in a zero-mode waveguide, PacBio SMRT sequencing has delivered usable reads of up to 54 kbp[8]. Preliminary results from Oxford Nanopore suggest that >10 kbp read lengths are possible using MinION nanopore sequencing[9]. Such long read lengths drastically simplify genome assembly by resolving repetitive structures in the assembly graph[8,10]. However, the long reads generated by single-molecule sequencing currently suffer from low accuracy (82–87% PacBio[11], 78–85% MinION[9]), and new algorithms have been needed to compensate for mistakes in the raw sequences[10,12–15]. Although SMRT sequencing may be error-prone, it exhibits less sequencing bias than previous technologies[16,17], and theoretical research has shown that random error can be overcome algorithmically[18]. Thus, by oversampling the genome at sufficient coverage (e.g., 50× of PacBio P5C3), SMRT sequencing can be used to produce highly accurate and continuous assemblies[10,12–15], including automatically finished genomes for most bacteria and archaea[11].

Early assemblies of noisy, long reads have been successful, but have suffered from a substantial computational cost. For example, an initial assembly of *D. melanogaster* from SMRT reads required more than 600,000 CPU hours, when tools available at the time were used—the equivalent of more than 20 days running on a thousand-core computer cluster[19]. Even small bacterial genomes previously required a day to assemble using the HGAP[15] or PBcR[10] assembly pipelines. The primary bottleneck of long-read assembly has been the sensitive all-versus-all alignment required to determine overlapping read pairs. For the *D. melanogaster* SMRT assembly, this overlapping step consumed 95% of the total runtime. Even if the accuracy of long-read technologies improves, all-pairs overlapping will remain a substantial bottleneck in overlap-layout-consensus assembly[1]. Taken together, the computational cost and the comparatively high sequencing costs have prevented widespread application of SMRT sequencing to genomes larger than 100 Mbp. The steadily increasing throughput of the PacBio instrument has begun to address sequencing costs, but the computational cost of assembling larger genomes has remained beyond the reach of most investigators.

To solve the computational problem of long-read assembly, we present a probabilistic algorithm for efficiently detecting overlaps between noisy, long reads. MHAP uses a dimensionality reduction technique named MinHash[20] to create a more compact representation of sequencing reads. Originally developed to determine the similarity of web pages[21], MinHash reduces a text or string to a small set of fingerprints, called a sketch. MinHash sketches have been successfully

[1]Department of Chemistry and Biochemistry, University of Maryland, College Park, Maryland, USA. [2]Institute for Advanced Computer Studies, University of Maryland, College Park, Maryland, USA. [3]Invincea Labs, Arlington, Virginia, USA. [4]National Biodefense Analysis and Countermeasures Center, Frederick, Maryland, USA. [5]Pacific Biosciences of California, Inc., Menlo Park, California, USA. [6]These authors contributed equally to this work. Correspondence should be addressed to S.K. (korens@nbacc.net).

Received 14 August 2014; accepted 8 April 2015; published online 25 May 2015; corrected after print 6 October 2015; doi:10.1038/nbt.3238

applied to document similarity[20], image similarity[22], sequence similarity[23–25] and metagenomic clustering[26]. The approach can also be viewed as a generalization of minimizers[27]. Briefly, to create a sketch for a DNA sequence, one must convert all *k*-mers (also known as, shingles or *q*-grams) to integer fingerprints using multiple, randomized hash functions. For each hash function, only the minimum valued fingerprint, or min-mer, is retained. The collection of min-mers for a sequence makes the sketch (**Fig. 1** and Online Methods). This locality-sensitive hashing allows the Jaccard similarity of two *k*-mer sets to be estimated by simply computing the Hamming distance between their sketches. The resulting estimate is strongly correlated with the number of shared *k*-mers between two sequences (**Supplementary Fig. 1**). Because the sketches are comparatively small, this is a computationally efficient technique for estimating similarity.

RESULTS

MinHash alignment filtering

MHAP uses MinHash sketches for efficient alignment filtering. The time required to hash, index, store and compare *k*-mers is proportional to the sketch size, so it is preferable to keep sketches small. However, using fewer min-mers reduces the sensitivity of the filter. It is possible to use sketches an order of magnitude smaller than the input reads, while maintaining acceptable overlap detection accuracy (**Fig. 2a**,**b**). For human, using a small value of *k* (e.g., 10) increases the number of false matches found, so it is preferable to use the largest value of *k* that maintains sensitivity.

Specifically, 16-mers can effectively detect 2 kbp overlaps from 10 kbp reads simulated from the human genome with an overlap error rate of 30%, so MHAP uses $k = 16$ by default (**Fig. 2b**, **Supplementary Notes 1** and **2** and Online Methods). Sensitivity can be further improved by increasing the sketch size, which reduces the expected error of the Jaccard estimate (**Supplementary Fig. 1**). Additionally, because the error rate of an alignment is roughly additive in the error rate of the two reads, mapping high-error reads to a reference genome is easier than overlapping. For mapping 10 kbp reads to the human genome with a 15% error rate, a sketch of only ~150 16-mers is required to achieve over 80% sensitivity.

The efficiency of MHAP improves with increased read length. **Figure 2c** compares the total number of *k*-mers counted during MHAP overlapping with a direct approach that exactly measures the Jaccard similarity between two reads without using sketches. For a fixed number of total bases sequenced, and a minimum 20% overlap length, the relative number of min-mer comparisons performed by MHAP decays rapidly with increasing read length, because the complexity is governed only by the sketch size (a constant) and the number of reads (which decreases for increasing read length; **Supplementary Note 1** and **Supplementary Table 1**). Thus, the efficiency of MHAP is expected to improve with the increasing read length and accuracy of future long-read sequencing technologies.

MHAP overlapping performance

In addition to being fast, MHAP is also a highly sensitive overlapper. We evaluated the sensitivity and specificity of MHAP versus two other tools designed for SMRT reads, BLASR[28] and DALIGNER[29]. BWA-MEM[30], SNAP[31] and RazerS[32] were also evaluated, but current versions of these algorithms did not reliably detect noisy overlaps between all pairs of reads (**Supplementary Note 3**). The performance of MHAP, BLASR and DALIGNER was evaluated by comparing detected overlaps to true overlaps, which were inferred from mapping reads to reference genomes, and the tools were evaluated using multiple parameter settings and sequencing chemistries (**Table 1**, **Supplementary Tables 2** and **3** and **Supplementary Figs. 2** and **3** and Online Methods).

MHAP sensitivity is tunable based on the size of *k*, the sketch size and the Jaccard similarity threshold. Based on the parameter sweep (**Supplementary Table 2**) and empirical assembly tests, two MHAP parameter settings (fast and sensitive) were chosen that balanced speed with accuracy (**Table 1** and **Supplementary Note 2**). BLASR sensitivity is primarily affected by the *bestn* parameter, which controls how many alignments are reported for each read. The HGAP[15] assembler sets *bestn* equal to the depth of sequencing coverage, but this can result in missed overlaps for repetitive genomes. BLASR runtime and sensitivity was highly genome-dependent and affected by sequence complexity and uneven replicon coverage (**Table 1**). Like BWA-MEM, BLASR was originally designed for mapping to a reference and is not ideally suited for overlapping all pairs of reads. In contrast, MHAP considers all possible alignments; it was consistently accurate across all genomes tested and an order of magnitude faster than BLASR at all levels of sensitivity (**Supplementary Figs. 2** and **3**).

Like MHAP, DALIGNER utilizes efficient *k*-mer matching to detect long-read overlaps. Although developed for the Dazzler assembler,

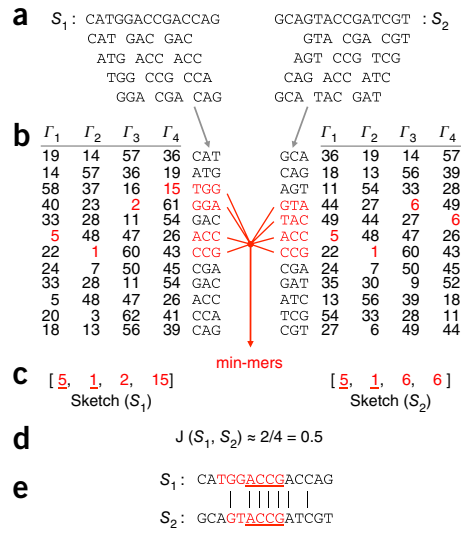

Figure 1 Rapid overlapping of noisy reads using MinHash sketches. (**a**) To create a MinHash sketch of a DNA sequence *S*, we first decomposed the sequence into its constituent *k*-mers. In the example shown, $k = 3$, resulting in 12 *k*-mers each for S_1 and S_2. (**b**) All *k*-mers are then converted to integer fingerprints by multiple hash functions. The number of hash functions determines the resulting sketch size *H*. Here, where $H = 4$, four independent hash sets are generated for each sequence ($\Gamma_{1\ldots H}$). In MHAP, after the initial hash (Γ_1), subsequent fingerprints are generated using an XORShift pseudo-random number generator ($\Gamma_{2\ldots H}$). The *k*-mer generating the minimum value for each hash is referred to as the min-mer for that hash. (**c**) The sketch of a sequence is composed of the ordered set of its *H* min-mer fingerprints, which is much smaller than the set of all *k*-mers. In this example, the sketches of S_1 and S_2 share the same minimum fingerprints (underlined) for Γ_1 and Γ_2. (**d**) The fraction of entries shared between the sketches of two sequences S_1 and S_2 (0.5) serves as an estimate of their true Jaccard similarity (0.22), with the error bound controlled by *H*. In practice, $H \gg 4$ is required to obtain accurate estimates. (**e**) If sufficient similarity is detected between two sketches, the shared min-mers (ACC and CCG in this case) are located in the original sequences and the median difference in their positions is computed to determine the overlap offset (0) for S_1 and S_2.

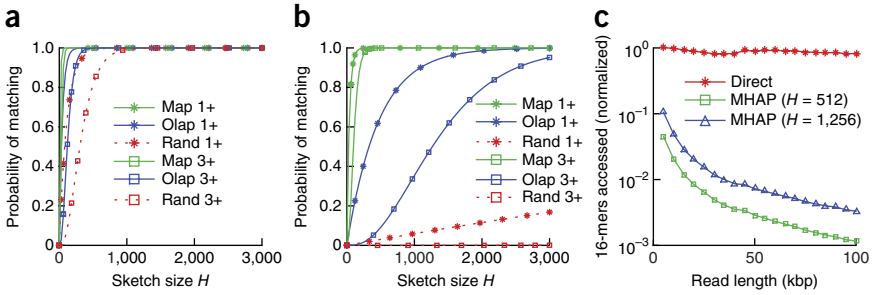

Figure 2 Simulated MHAP performance for various sketch sizes and read lengths. 10 kbp reads were randomly extracted from the human reference genome, and errors were introduced to simulate a SMRT sequencing error model (10% insertion, 2% deletion and 1% substitution)[13]. (**a**) For $k = 10$, it is common to find at least three min-mer matches by chance. (**b**) For $k = 16$, at least three min-mer matches are sufficient to separate random matches from true matches. Three scenarios are shown for ≥1 and ≥3 matching min-mers: unrelated sequences (Rand), reads overlapping by exactly 2 kbp (Olap) and reads mapped to a perfect reference (Map). The expected Jaccard similarity between a pair of random and nonrandom reads was estimated based on 50,000 independent trials and equation (9) in Online Methods. (**c**) The total number of 16-mers processed by MHAP decreases exponentially relative to the direct approach (shown for sketch sizes of 512 and 1,256). The number of accesses is normalized by the maximum value observed during the simulations, and given on a log-scaled *y* axis. Random fluctuations are an artifact of the random read sampling.

which has not been released, DALIGNER can be run independently. The DALIGNER code was under active development and unstable for large genomes at the time of writing, so tests were limited to 1 Gbp of subsampled data for all programs (**Table 1**). In contrast to MHAP's sketch strategy, DALIGNER's direct approach considers all *k*-mers in all reads (e.g., all *k*-mers in **Fig. 1**, rather than the reduced sketch). This means that DALIGNER must rely heavily on filtering repetitive *k*-mers for efficiency. Matching *k*-mer seeds are identified from the repeat-filtered set using a cache-optimized radix sort and merge, and then extended using a linear-time difference algorithm[33] to compute the overlap. On the subsampled data, DALIGNER was the fastest tool tested, with sensitivity similar to MHAP 'sensitive', but DALIGNER dropped an average of 5% in sensitivity for sequences 10 kbp or longer, whereas MHAP maintained sensitivity across all read lengths (**Supplementary Table 3**). Because MHAP computes sketches before overlaps, it pays a higher initialization cost than DALIGNER, which is a large percentage of runtime for the smaller data sets presented in **Table 1**. However, for larger data sets, MHAP's initialization cost is offset by faster lookups and reduced representation size. In addition, the sketching approach does not require aggressive repeat and *k*-mer filtering, allowing MHAP to find overlaps from the most repetitive regions of the genome and enabling streaming computation. With repeat filtering disabled, DALIGNER was unstable and slower than MHAP (**Supplementary Note 3**). The performance of MHAP and DALIGNER is governed by multiple factors, but is comparable according to summary metrics on these subsampled data sets. How well the detected overlaps form an assembly will be an important point of future comparison, once a DALIGNER-based assembler is made available.

SMRT sequencing and assembly

To demonstrate the efficient and complete assembly of large genomes, we integrated MHAP into the Celera Assembler[34] PBcR[10,13] (PacBio corrected reads) hierarchical assembly pipeline and assembled previously released whole-genome SMRT data from *Escherichia coli* K12, *S. cerevisiae* W303, *D. melanogaster* ISO1, *Arabidopsis thaliana* Ler-0[35] and the complete hydatidiform mole CHM1[17] (85×, 117×, 90×, 144× and 54× coverage, respectively; **Supplementary Table 4**).

Hierarchical long-read assembly pipelines like PBcR and HGAP have been shown to outperform hybrid methods for high-coverage (>50×) SMRT sequencing[8,10,11]. These assemblers are similar in design, so only PBcR is considered in this study to isolate the effect of the overlapper. In the first phase of PBcR, similar, potentially overlapping pairs of uncorrected single-molecule reads are found using either BLASR or MHAP (PBcR-BLASR and PBcR-MHAP). The raw reads are then corrected using a consensus of overlapping reads, and the corrected reads are assembled using an updated version of Celera Assembler[13]. For improved consensus quality, the Quiver[15] polishing tool realigns raw reads to the assembly to correct short insertion, deletion and substitution errors. For higher accuracy technologies, such as Illumina Synthetic Long Reads (Moleculo), the correction step can be bypassed and the reads assembled directly.

Table 1 Overlapping sensitivity and specificity for MHAP, BLASR and DALIGNER

Genome	Program	Sensitivity (%)	PPV (%)	Time (CPU h)	Overlap Mem (GB)
E. coli K12	BLASR fast	89	100	21.53	3.71
	BLASR sensitive	95	100	86.84	3.71
	MHAP fast	65	99	2.27	20.14
	MHAP sensitive	85	99	3.61	22.34
	DALIGNER	86	100	0.95	22.16
S. cerevisiae W303	BLASR fast	16	100	71.43	10.85
	BLASR sensitive	35	100	434.30	11.42
	MHAP fast	70	98	18.58	17.69
	MHAP sensitive	83	98	31.44	19.95
	DALIGNER	73	100	10.55	34.04
A. thaliana Ler-0	BLASR fast	2	100	30.92	17.87
	BLASR sensitive	11	100	82.24	11.69
	MHAP fast	69	100	13.34	19.63
	MHAP sensitive	88	100	21.27	22.21
	DALIGNER	86	100	9.67	48.13
D. melanogaster ISO1	BLASR fast	75	100	166.81	23.41
	BLASR sensitive	89	100	878.15	22.89
	MHAP fast	62	99	9.08	23.24
	MHAP sensitive	83	99	15.73	20.61
	DALIGNER	85	100	5.99	34.27
Human CHM1	BLASR fast	41	100	34.20	13.20
	BLASR sensitive	59	100	42.50	9.36
	MHAP fast	64	96	9.37	18.20
	MHAP sensitive	77	97	17.19	21.90
	DALIGNER	82	99	6.78	35.96

Sensitivity, the percentage of true overlaps identified by the program; PPV (positive predictive value), the percentage of true overlaps out of all overlaps reported. Overlaps were considered true if confirmed by reference mapping or by an alignment greater than 70% identity across the length of the overlap (**Supplementary Note 9**). PPV is included to show that MHAP is highly specific, despite not computing a gapped alignment. However, PPV is less critical than sensitivity, as downstream steps of the assembly pipeline can arbitrarily filter low-identity overlaps. In all cases, a maximum of 1 Gbp was randomly selected and mapped to the reference genome to ensure all programs would finish. MHAP was run with parameters $k = 16$, min. matches = 3, min. Jaccard = 0.04 and sketch sizes of 512 (fast) and 1,256 (sensitive). BLASR was run with parameters: minReadLength, 2,000; maxScore, 1,000; maxLCPLength, 16; minMatch, 12; m 4 and *nCandidates/bestn* set to sequencing depth of coverage (fast); and 10× coverage (sensitive). DALIGNER was run with parameters: -d -v -H2000 and all others set to defaults. By default, MHAP filtered *k*-mers representing more than 0.001% of total bases and DALIGNER filtered all *k*-mers occurring more than 100 times. True positives and false negatives were estimated to ± 1% (Online Methods and **Supplementary Note 9**).

To measure the effect of SMRT sequencing coverage on overlapping and assembly, we randomly sampled and assembled the *S. cerevisiae* W303 genome using 25–100× coverage (**Supplementary Note 4** and **Supplementary Fig. 4**). At coverage less than ~70×, the assembly benefited from increased overlap sensitivity (MHAP sensitive), but for coverage more than ~70×, the increased redundancy compensated for lower sensitivity (MHAP fast). The precise coverage where higher sensitivity is preferred depends on the genome size, repeat complexity, and sequencing error rate, but for all genomes considered here, MHAP fast produced larger contigs than BLASR. By default, PBcR-MHAP will automatically enable sensitive overlapping and correction when coverage is less than 50×.

Table 2 details the assembly performance of PBcR-MHAP and PBcR-BLASR on PacBio SMRT sequencing data. Owing to its recent release and ongoing integration with Dazzler, no DALIGNER-based assemblies were available for comparison. Assembly continuity is measured using the traditional NG50 metric (half the genome size is contained in contigs of length *N* or greater) and an "assembly performance" metric defined by Lee *et al.*[8] as the NG50 contig length divided by the NG50 length of the reference segments (observed vs. idealized NG50). In all cases compared, MHAP overlapping produced a comparable or improved assembly in less time than BLASR. For bacterial genomes, a complete assembly from long reads now requires roughly the same compute time as generating an incomplete assembly from short reads (**Supplementary Note 5**, **Supplementary Table 5** and **Supplementary Fig. 5**). For all eukaryotic genomes tested, except CHM1, the MHAP assemblies rival the continuity of current reference genomes.

For *S. cerevisiae* W303, the MHAP assembly assembles 12 out of 16 chromosomes without gaps, approaches perfect continuity and represents the most continuous assembly of *S. cerevisiae* W303 to date, including both hybrid and reference-assisted assemblies[8,36] (**Supplementary Fig. 6**). Greater performance gains were observed for the larger, more complex genomes of *A. thaliana* Ler-0 and *D. melanogaster* ISO1 (**Table 2**). Compared to the *A. thaliana* Col-0 version 10 reference[37], which has undergone continued improvement since its initial sequencing in 2000, the MHAP Ler-0 assembly is more continuous and contains an average of five fewer gaps per chromosome (**Supplementary Fig. 7**). Similarly, the MHAP *D. melanogaster* assembly achieves excellent continuity and widespread agreement with the version 5 reference[38] (**Supplementary Table 5** and **Supplementary Fig. 8**), as well as an ~600-fold speedup versus BLASR. Despite the original *Drosophila* Genome Project's 2 kbp, 10 kbp and bacterial artificial chromosome (BAC) inserts, which provide long-range structural information, the contig NG50 of the MHAP SMRT assembly is greater than the scaffold NG50 of the original Sanger assembly (21 Mbp vs. 14 Mbp)[34], demonstrating that sufficient coverage of long reads can independently resolve repeats in eukaryotic genomes. For example, each autosomal chromosome arm (2L, 2R, 3L, 3R, 4) is spanned by just five MHAP contigs, on average, and chromosome arm 3L is fully spanned by a single 25 Mbp contig (**Fig. 3**). This assembly potentially resolves 52 of 124 (42%) gaps in the version 5 reference that have persisted for over a decade of finishing (**Supplementary Note 5** and **Supplementary Table 6**). Half of these putative gap closures match the estimated gap sizes in the reference, but require further validation to confirm. Lastly, this assembly required less than 4 days on a single 16-core computer, demonstrating that reference-grade assembly of 100 Mbp eukaryotes is now possible without computing clusters.

De novo human assembly using long reads

The human genome has long been regarded as the pinnacle of whole-genome shotgun sequence assembly[39] and is the most widely studied genome with enormous resources dedicated to sequencing,

Table 2 Continuity and performance of long-read assembly with MHAP

Genome	Pipeline	Assembly Size	Contigs	NG50 (AP)	Ctg/Chr	Olap h	Contig h	Quiver h
E. coli K12	SPAdes[60] Illumina	4,596,097	166	133,063 (3%)	166	–	3.6[a]	–
	SPAdes[60] Hybrid	4,664,071	63	1,443,745 (31%)	63	–	4.9[a]	–
	PBcR-BLASR	4,661,585	1	4,661,585 (100%)	1	23	23	6.6
	PBcR-MHAP	4,651,226	1	4,651,226 (100%)	1	2.5 (9×)	2.1 (11×)	6.6
S. cerevisiae W303	S228C Reference[51]	12,157,105	17	924,431 (100%)	1	–	–	–
	W303 Reference[36]	11,886,100	359	261,861 (28%)	20	–	–	–
	PBcR-BLASR	12,294,493	23	818,023 (89%)	2	150	71	66
	PBcR-MHAP	12,255,862	21	818,260 (89%)	2	20 (8×)	7.5 (9×)	66
A. thaliana Ler-0	TAIR10 (ref. 37)	119,482,035	102	11,194,537 (100%)	15	–	–	–
	HGAP[15]ASM[61]	130,857,836	545	6,720,323 (60%)	16	–	–	–
	PBcR-MHAP	120,486,579	38	11,164,124 (100%)	8	1,700	200	420
D. melanogaster ISO1	Ref v5 (ref. 38)	162,469,346	37,647	21,485,538 (100%)	2	–	–	–
	Ref v1 (ref. 46)	114,201,575	1,450	201,052 (1%)	243	–	–	–
	PBcR-BLASR	138,364,521	128	15,297,019 (71%)	11	610,000	21,000	500
	PBcR-MHAP	143,328,915	132	20,985,587 (98%)	11	890 (670×)	170 (120X)	500
Human CHM1	Ref 38	3,049,316,025	1,386	56,413,054 (100%)	30	–	–	–
	Ref 28 (ref. 40)	2,852,886,468	48,081	455,244 (1%)	1121	–	–	–
	Illumina[42]	2,827,635,806	40,824	129,173 (0%)	1530	–	–	–
	PBcR-MHAP	2,816,715,694	5,202	1,857,788 (3%)	165	82,000	33,000	6,000
	PBcR-MHAP (s)	2,828,300,545	3,434	4,320,471 (8%)	88	220,000	40,000	6,000

[a]For SPAdes, this represents the entire assembly time.

Assembly, total number of base pairs in all contigs (only contigs containing at least 50 reads are included in all PBcR results); Contigs, number of contigs >200 bp; NG50, *N* such that 50% of the genome is contained in contigs of length ≥*N* where the genome size is set to reference length, excluding unknown sequences; AP, assembly performance, is the contig NG50 divided by the NG50 of the reference segments[8]. The genome sizes were estimated from the reference to be 4,639,675 for *E. coli* K12; 12,157,105 for *S. cerevisiae* W303; 119,482,035 for *A. thaliana* Ler-0; 129,663,327 for *D. melanogaster* ISO1; and 3,101,804,741 for human; Ctg/Chr, average number of contigs >200 bp per chromosome, excluding unassigned scaffolds. Averages for assemblies are based on alignments to the reference genome, whereas averages for reference genomes (and NG50) are based on splitting at three or more consecutive Ns. Olap h, CPU hours to compute initial overlaps using BLASR or MHAP (with fold speedup versus BLASR, where applicable); Contig h, CPU hours to correct reads and assemble contigs after initial overlapping; Quiver h, CPU hours to polish the assembly with Quiver. All timing was performed on AMD 2.4GHz 6136 processors, and overlapping steps were constrained to 32 GB of RAM per computational node. BLASR was run with default PBcR parameters, which automatically sets *nCandidates/bestn* such that 99% of reads will find all their overlaps (1.5C for *E. coli*, 1.5C for *S. cerevisiae*, and 35C for *D. melanogaster*). All MHAP assemblies were run with MHAP fast, except CHM1, which was also run with the more sensitive parameter settings (s) due to the relatively lower depth of coverage.

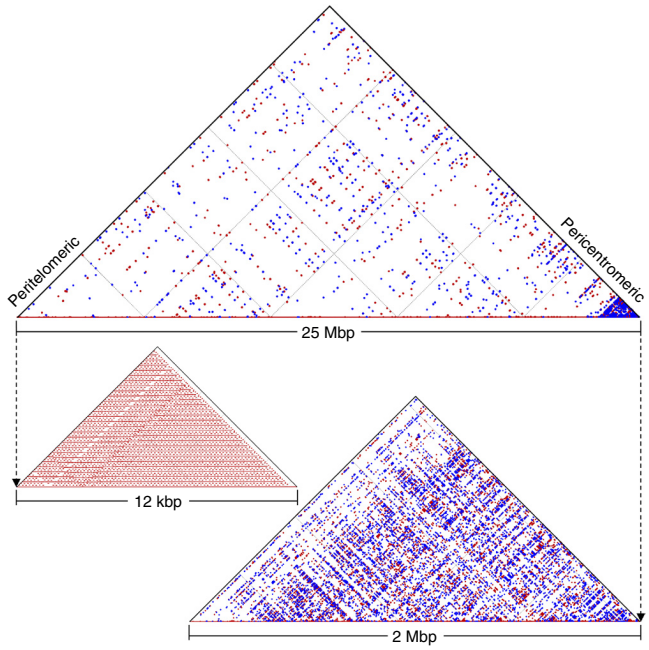

Figure 3 Single-contig assembly of *D. melanogaster* chromosome arm 3L. A single ~25 Mbp contig from the MHAP *D. melanogaster* assembly covers the full euchromatic region of chromosome arm 3L. (Top) All 100-bp exact repeats across the length of the 3L assembly are shown using a self-alignment dotplot. Red dots indicate forward repeats, and blue dots inverted repeats. Points nearer to the hypotenuse indicate repeat copies nearer to each other in the genome. (Bottom left) All 20 bp exact repeats are shown for the first 12 kbp of the assembly illustrating a peritelomeric tandem repeat. (Bottom right) All 20 bp exact repeats are shown for the last 2 Mbp of the assembly, which is composed of an elevated repeat density, characteristic of the pericentromeric region.

assembling and finishing[40–42]. As a final test of MHAP, we assembled 54× SMRT sequencing reads from the essentially haploid CHM1 genome[17], and compared our assembly to the GRCh38 human reference, as well as to two alternate assemblies of CHM1—a reference-guided 100× Illumina assembly[42] and a patched version of GRCh37 generated by iterative mapping and assembly of the same SMRT data analyzed here[17]. The CHM1 genome lacks allelic variation, simplifying assembly, but the genome size and repeat content remain challenging.

The contig NG50 of the MHAP CHM1 assembly is an order of magnitude larger than both the Illumina CHM1 assembly and early BAC-based Sanger assemblies of the human genome (**Fig. 4**, **Table 2** and **Supplementary Figs. 9–11**). As predicted by the *S. cerevisiae* coverage analysis, the sensitive MHAP parameters produced a better assembly than the fast MHAP parameters for this depth of coverage. Based on a comparison to GRCh38, the MHAP assembly averages just 88 contigs per chromosome and potentially resolves 51 of 819 (6%) annotated reference gaps (**Fig. 4**, **Supplementary Note 5** and **Supplementary Table 6**). Of these putative gap closures, 16 match the estimated gap sizes in the reference, but require further validation to confirm. By the same analysis, the patched GRCh37 reference created by Chaisson *et al.*[17] potentially closes 43 GRCh38 gaps, and the Illumina-based assembly closes 3 (not unexpected, as information from this project has already been incorporated into GRCh38). One example of an historically difficult region to assemble is the major histocompatibility complex (MHC), which has an important role in immunity[43]. In contrast to the Illumina assembly, which breaks the MHC region into over 60 contigs, MHAP assembles 97% of this region into just two contigs. Compared to the Illumina assembly, a total of 21 (5%) additional MHC genes are correctly reconstructed into a single piece by the MHAP assembly.

An alignment dotplot shows general agreement between the MHAP assembly and the GRCh38 reference (**Supplementary Fig. 12**). However, thousands of known structural variants unique to CHM1 (ref. 17) precluded a reference-based validation of our assembly using GRCh38. For reference-free validation, we mapped complementary Illumina data from the same CHM1 sample[17] to the MHAP assembly and quantified the number of base discrepancies. By this measure, we estimated the average consensus accuracy to be 99.99% across 94% of the assembled bases covered by mapped Illumina pairs (**Supplementary Note 6**). For additional validation, we replicated the approach of Chaisson *et al.*[17] and compared the assembly to 16 finished CHM1tert[44] BAC clones. These 16 BACs were selected from unduplicated regions of the genome and validated to ensure accurate mapping and quality estimation (Mark Chaisson, personal communication). The MHAP assembly structurally agrees with these BACs and matches the 99.97% identity previously reported[17] (**Supplementary Note 6**). This 0.02% reduction in identity is likely due to sequence composition, with 72% of SMRT consensus errors in these regions localized to homopolymers[17]. Extending this analysis to all 102 BACs generated by Chaisson *et al.*[17], including a tiling of the complex 10q11.23 region, lowered average BAC alignment identity to 99.66% (**Supplementary Note 6**). However, this identity figure is uncertain, because mapping error and potential errors in the BACs themselves confound validation. Confident assembly and validation of such regions remain an open problem, as evidenced by the seven gaps and multiple misassemblies in the GRCh37 assembly of the 10q11.23 region[17]. Because sequencing read-length determines the complexity of the assembly problem[10], the accurate reconstruction of these highly repetitive regions will likely require longer sequencing read-lengths than are currently available.

Assembly validation and repeat resolution

To validate all assemblies, we compared each to the closest available reference genome using dnadiff[45] (**Supplementary Note 5** and **Supplementary Table 5**). With some minor exceptions, all assemblies are structurally concordant with the reference sequences (**Supplementary Figs. 5–8** and **12**). However, only the *E. coli* and *D. melanogaster* data sets were generated from the same strain as the reference, and detailed validation was limited to these genomes. In particular, the *D. melanogaster* SMRT reads were sequenced from the same sub-line of ISO1 used by the *Drosophila* Genome Project since 2000 (ref. 46), providing the opportunity to thoroughly validate SMRT sequencing of a eukaryotic genome.

Supplementary Table 5 provides GAGE[47] accuracy metrics for the *E. coli* and *D. melanogaster* assemblies. Both MHAP assemblies achieve a minimum of 99.99% accuracy, corresponding to a Phred Quality Value[48] (QV) of 40 and are more accurate and complete than the BLASR-based assemblies. These accuracy estimates count all consensus discrepancies as errors, including heterozygous variants, and therefore represent a lower bound on accuracy. Quiver provides an additional QV10 improvement in accuracy with the cost of added runtime (**Table 2**).

We further analyzed the completeness of the *D. melanogaster* gene sequences in the MHAP assembly (**Supplementary Note 7**). We mapped 17,294 annotated genes (full-length, including introns) to the MHAP assembly, identifying a total of 16,776 (97%) genes contained

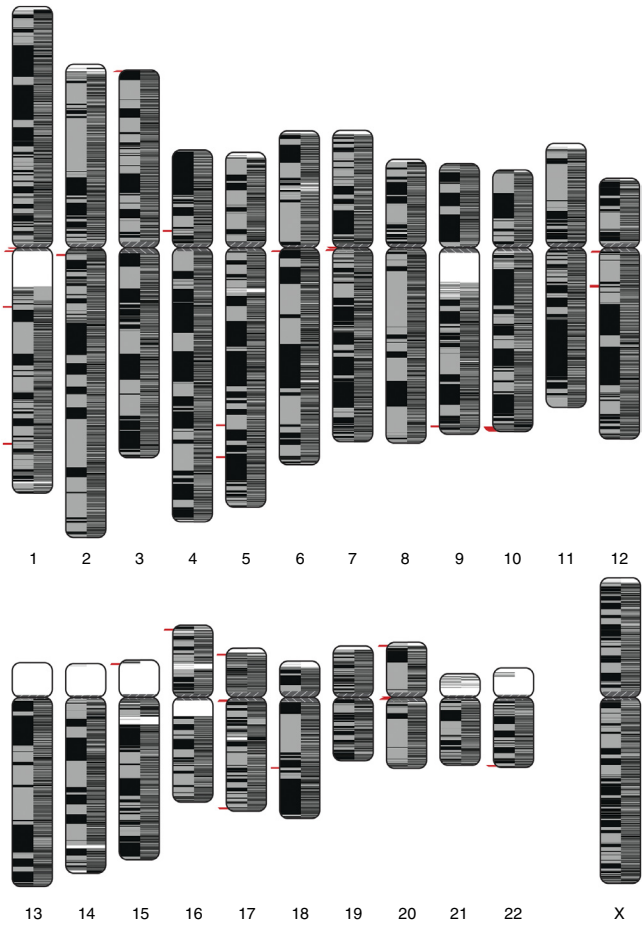

Figure 4 Continuity and putative GRCh38 gap closures of the CHM1 assembly. Human chromosomes are painted with assembled CHM1 contigs using the colored Chromosomes package[59]. Alternating shades indicate adjacent contigs, so each vertical transition from gray to black represents a contig boundary or alignment breakpoint. The left half of each chromosome shows the MHAP assembly of the SMRT data set and the right half shows the Illumina-based assembly[42]. The SMRT assembly is considerably more continuous, with an average of less than 100 contigs per chromosome. Putative GRCh38 gap closures are shown as red dashes next to the spanned gap position. Most SMRT closures fall near the telomeres and centromeres. The three gaps spanned by the Illumina assembly are not associated with the primary chromosomes and cannot be displayed. Tiling gaps, in white, coincide with missing assembly sequence and/or regions of uncharacterized reference sequence (i.e., long stretches of *N*'s) to which no contigs could be mapped. A full list of putative gap closures is given in **Supplementary Table 6**.

in a single alignment to a single contig (vs. 16,189 for BLASR). Of these, 16,751 were reconstructed at over 99% identity and 14,824 were reconstructed with perfect identity (vs. 12,591 for BLASR).

Because repeats represent the greatest challenge to assembly, we also analyzed the completeness of *D. melanogaster* transposable element (TE) families. TEs in *D. melanogaster* represent a large fraction of annotated repeats and vary over a wide range of sizes and levels of sequence diversity[49]. To assess TE resolution in our assembly, we replicated the analysis of a recent study[50] that assembled *D. melanogaster* from Illumina Synthetic Long Reads (Moleculo) using Celera Assembler. The SMRT assembly is hundreds of fold more continuous than the Moleculo assembly (NG50 21 Mbp vs. 70 kbp), partially owing to higher sequencing coverage (90× vs. 34×) but primarily because of longer read length and reduced sequencing bias. Of 5,425 annotated TE elements in the euchromatic arms[49], 5,274 (97%) are contained in a single contig by the MHAP assembly, and the majority (4,984) aligned perfectly to the reference. For the highly abundant *roo* TE family, 97% (134 of 138) of copies were resolved, with 93 at 100% identity. This is in contrast to the results of McCoy *et al.*[50], where only 5.2% of *roo* copies were resolved using Moleculo. For the *juan* family, with less than 0.01% divergence between copies, all 11 copies were reconstructed at perfect identity by the SMRT assembly. We conclude that the high error rate of SMRT sequencing does not prohibit the accurate reconstruction of TE sequences, even from highly abundant TE families with many identical copies interspersed throughout the genome.

Improved telomere assemblies

Because SMRT sequencing generates long reads without the need for cloning or amplification, it is possible to better reconstruct the repeat-rich heterochromatic regions of eukaryotic chromosomes. This is a distinct advantage compared to previous sequencing methods, for which heterochromatin sequencing was thought to be impossible because of cloning biases or short read lengths. As proof of principle, we evaluated the ability of long-read sequencing to reconstruct heterochromatic sequences in the telomeric regions of *S. cerevisiae*[51] and *D. melanogaster*[38] (**Supplementary Note 8**). Telomeres play important roles in chromosome replication in all eukaryotic genomes, and in humans their loss has been associated with disease[52]. However, these sequences are typically missing from *de novo* assemblies in eukaryotes. For example, it was not until 6 years after the initial shotgun assembly of *D. melanogaster* that the reference genome began to include telomeric sequence[53].

We mapped known *S. cerevisiae* telomeric repeats to the MHAP *S. cerevisiae* W303 assembly (**Supplementary Note 8**) and identified nine chromosomes where a single contig included an alignment comprising at least 50% of the terminal telomeric repeat on both the left and right ends. This indicates that a majority of the 16 chromosomes were completely resolved from telomere to telomere. In the remaining cases, four chromosomes were composed of more than one contig containing the telomeres, and two chromosomes did not extend into the telomeres (or the telomeric sequence did not match the reference). This is a substantial improvement compared with the current *S. cerevisiase* W303 genome[36], in which only one chromosome is spanned from end to end and only five chromosome ends have been annotated.

In contrast to the simple telomeric repeats of *S. cerevisiae* and other eukaryotes, *D. melanogaster* telomeres are composed of head-to-tail arrays of three specialized retrotransposable elements (*Het-A*, *TART* and *Tahre*) and clusters of telomere-associated sequences (TASs)[53]. Because telomeric TEs preferentially transpose to chromosome ends, they are virtually absent from the euchromatic regions[53]. By mapping repeat families from RepBase[54] and a recent *D. melanogaster* repeat study[55], we identified repeat arrays characteristic of *D. melanogaster* telomeres (**Supplementary Note 8**). A total of 24 telomeric contigs are present in the MHAP assembly. One contig, corresponding to chromosome 2R, contains both subtelomeric (*HetRp_DM*) and telomeric sequence, fully capturing the transition from euchromatin to heterochromatin. This contig extends 80 kbp beyond the end of the reference assembly, and represents the first time the full telomeric transition sequence has been identified for this chromosome. Chromosome arms 2L, 3R and X also have large contigs containing telomeric repeats extending past the end of the current reference sequence, indicating additional regions of the reference that could be improved by the *de novo* MHAP assembly.

DISCUSSION

We show in this manuscript that it is possible to assemble large genomes from noisy, long reads. As was previously demonstrated for microbial genomes, assembly of eukaryotic genomes using PacBio SMRT sequencing can produce reference-grade genomes. In the best cases, entire chromosome arms assemble into single-piece assemblies from telomere to centromere. As required, base quality can be further improved by polishing with a complementary short-read sequencing technology. For the few remaining gaps, long-read assemblies could be paired with super-long linking information as generated by optical[56] or chromatin interaction maps[57,58]. These complementary scaffolding approaches could be used to span centromeres, resolve entire chromosomes and phase haplotypes to produce truly complete assemblies.

Our results indicate that probabilistic alignment methods are well suited to address the read length and error rate of single-molecule sequencing. Several strategies have previously been developed to find similarities in high-dimensional data, and chief among them are probabilistic dimensionality-reduction approaches[20]. Such algorithms trade the guaranteed accuracy of a deterministic method for a much faster solution with bounded error. By producing high-quality assemblies in a fraction of the time, MHAP demonstrates that this tradeoff is acceptable for the overlapping problem, where sequencing redundancy compensates for increased variance in overlap estimates. Further, the long reads produced by single-molecule sequencing allow for a coarser estimate of similarity than traditional dynamic programming.

MHAP can serve as a drop-in replacement for current overlapping methods. Although only haploid or inbred genomes were assembled in this study, the sensitivity of MHAP is well suited for outbred genomes. However, assembling structurally divergent alleles from sequence overlaps remains a considerable challenge for all assemblers. Preliminary support is included in Quiver for heterozygous variant calling, but automated haplotype phasing remains an area for future improvement. Also, MHAP currently reports only the boundaries of an overlap, not a gapped alignment. This is sufficient for PBcR, which computes gapped alignments during the correction phase of the algorithm, but for assemblers without a correction step, the addition of a fast alignment strategy may be required. Complementary concepts from MHAP and DALIGNER could also prove useful, as sketching is compatible with DALIGNER's sort-and-merge approach, and MHAP, which is currently implemented in Java, could benefit from DALIGNER's efficient C implementation and data structures. MHAP runtime would also benefit from DALIGNER's aggressive repeat filtering, but the consequences of this strategy on downstream assembly quality are unclear.

In addition to SMRT sequencing, MHAP is likely to be suitable for assembling nanopore sequences[6], which are expected to have similar read length and error characteristics. MinHash sketches could also be applied to the problems of reference alignment, sequence clustering and alignment-free distance estimation. Future work includes evaluating the applicability of MinHash to these other areas. Fast and sensitive methods are needed not just for long-read overlapping, but to address the ever-expanding scale of genomic data for all applications.

METHODS

Methods and any associated references are available in the online version of the paper.

Accession codes. Assemblies are available from GenBank: *E. coli*: CP009685; *S. cerevisae*: GCA_000773925; *A. thaliana*: JSAD00000000; *D. melanogaster*: GCA_000778455; *H. sapiens*: GCA_000772585. The latest version of Celera Assembler (including PBcR-MHAP) is available from http://wgs-assembler.sourceforge.net. All source codes for MHAP and the analyses presented here are available from https://github.com/marbl/MHAP. The software and data used for this manuscript (including the AWS tutorial) are available from http://www.cbcb.umd.edu/software/PBcR/MHAP and **Supplementary Software 1**).

Note: Any Supplementary Information and Source Data files are available in the online version of the paper.

ACKNOWLEDGMENTS

We are indebted to C. Bergman of the University of Manchester for his considered advice throughout this project and editing of an early version of this manuscript. We also thank Pacific Biosciences and all those involved in generating and freely releasing the data analyzed here. The contributions of S.K. and A.M.P. were funded under Agreement No. HSHQDC-07-C-00020 awarded by the Department of Homeland Security Science and Technology Directorate (DHS/S&T) for the management and operation of the National Biodefense Analysis and Countermeasures Center (NBACC), a Federally Funded Research and Development Center. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the US Department of Homeland Security. In no event shall the DHS, NBACC or Battelle National Biodefense Institute (BNBI) have any responsibility or liability for any use, misuse, inability to use, or reliance upon the information contained herein. The Department of Homeland Security does not endorse any products or commercial services mentioned in this publication.

AUTHOR CONTRIBUTIONS

K.B. and S.K. conceived, designed and implemented the MHAP algorithm. C.S.C. and J.P.D. conceived, designed and implemented the consensus algorithms. S.K. ran and analyzed the genome assemblies. J.M.L. coordinated data release and assisted with pipeline executions. C.S.C. and S.K. performed cloud-computing experiments. K.B., S.K. and A.M.P. drafted the manuscript. A.M.P. coordinated the project. All authors read and approved the final manuscript.

COMPETING FINANCIAL INTERESTS

The authors declare competing financial interests: details are available in the online version of the paper.

1. Miller, J.R., Koren, S. & Sutton, G. Assembly algorithms for next-generation sequencing data. *Genomics* **95**, 315–327 (2010).
2. Nagarajan, N. & Pop, M. Sequence assembly demystified. *Nat. Rev. Genet.* **14**, 157–167 (2013).
3. Denton, J.F. *et al.* Extensive error in the number of genes inferred from draft genome assemblies. *PLOS Comput. Biol.* **10**, e1003998 (2014).
4. Ukkonen, E. Approximate string-matching with q-grams and maximal matches. *Theor. Comput. Sci.* **92**, 191–211 (1992).
5. Schatz, M.C., Delcher, A.L. & Salzberg, S.L. Assembly of large genomes using second-generation sequencing. *Genome Res.* **20**, 1165–1173 (2010).
6. Clarke, J. *et al.* Continuous base identification for single-molecule nanopore DNA sequencing. *Nat. Nanotechnol.* **4**, 265–270 (2009).
7. Eid, J. *et al.* Real-time DNA sequencing from single polymerase molecules. *Science* **323**, 133–138 (2009).
8. Lee, H. *et al.* Error correction and assembly complexity of single molecule sequencing reads. *bioRxiv* doi:10.1101/006395 (2014).
9. Quick, J., Quinlan, A.R. & Loman, N.J. A reference bacterial genome dataset generated on the MinION portable single-molecule nanopore sequencer. *GigaScience* **3**, 22 (2014).
10. Koren, S. *et al.* Reducing assembly complexity of microbial genomes with single-molecule sequencing. *Genome Biol.* **14**, R101 (2013).
11. Koren, S. & Phillippy, A.M. One chromosome, one contig: complete microbial genomes from long-read sequencing and assembly. *Curr. Opin. Microbiol.* **23**, 110–120 (2015).
12. English, A.C. *et al.* Mind the gap: upgrading genomes with Pacific Biosciences RS long-read sequencing technology. *PLoS ONE* **7**, e47768 (2012).
13. Koren, S. *et al.* Hybrid error correction and *de novo* assembly of single-molecule sequencing reads. *Nat. Biotechnol.* **30**, 693–700 (2012).
14. Ribeiro, F.J. *et al.* Finished bacterial genomes from shotgun sequence data. *Genome Res.* **22**, 2270–2277 (2012).
15. Chin, C.S. *et al.* Nonhybrid, finished microbial genome assemblies from long-read SMRT sequencing data. *Nat. Methods* **10**, 563–569 (2013).

16. Ross, M.G. *et al.* Characterizing and measuring bias in sequence data. *Genome Biol.* **14**, R51 (2013).
17. Chaisson, M.J. *et al.* Resolving the complexity of the human genome using single-molecule sequencing. *Nature* **517**, 608–611 (2014).
18. Lam, K.K., Khalak, A. & Tse, D. Near-optimal assembly for shotgun sequencing with noisy reads. *BMC Bioinformatics* **15** (suppl. 9), S4 (2014).
19. PacBio. Data Release: Preliminary de novo Haploid and Diploid Assemblies of *Drosophila melanogaster* http://blog.pacificbiosciences.com/2014/01/data-release-preliminary-de-novo.html (2014).
20. Broder, A.Z. On the resemblance and containment of documents. *Compression and Complexity of Sequences 1997. Proceedings* 21–29 (1997).
21. Broder, A.Z. Identifying and filtering near-duplicate documents. *Combinatorial pattern matching* 1–10 (2000).
22. Chum, O., Philbin, J. & Zisserman, A. Near duplicate image detection: min-Hash and tf-idf weighting. *British Machine Vision Conference* **810**, 812–815 (2008).
23. Buhler, J. Efficient large-scale sequence comparison by locality-sensitive hashing. *Bioinformatics* **17**, 419–428 (2001).
24. Narayanan, M. & Karp, R.M. Gapped local similarity search with provable guarantees. *Algorithms Bioinform.* **3240**, 74–86 (2004).
25. Yang, X. *et al. De novo* assembly of highly diverse viral populations. *BMC Genomics* **13**, 475 (2012).
26. Rasheed, Z. & Rangwala, H. Mc-minh: Metagenome clustering using minwise based hashing. *SIAM International Conference in Data Mining* (2013).
27. Roberts, M., Hayes, W., Hunt, B.R., Mount, S.M. & Yorke, J.A. Reducing storage requirements for biological sequence comparison. *Bioinformatics* **20**, 3363–3369 (2004).
28. Chaisson, M.J. & Tesler, G. Mapping single molecule sequencing reads using basic local alignment with successive refinement (BLASR): application and theory. *BMC Bioinformatics* **13**, 238 (2012).
29. Myers, G. Efficient local alignment discovery amongst noisy long reads. *Algorithms Bioinform.* **8701**, 52–67 (2014).
30. Li, H. Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM. *arXiv preprint arXiv:1303.3997* (2013).
31. Zaharia, M. *et al.* Faster and more accurate sequence alignment with SNAP. *arXiv preprint arXiv:1111.5572* (2011).
32. Weese, D., Holtgrewe, M. & Reinert, K. RazerS 3: faster, fully sensitive read mapping. *Bioinformatics* **28**, 2592–2599 (2012).
33. Myers, E.W. AnO(ND) difference algorithm and its variations. *Algorithmica* **1**, 251–266 (1986).
34. Myers, E.W.A. Whole-genome assembly of *Drosophila. Science* **287**, 2196–2204 (2000).
35. Kim, K.E. *et al.* Long-read, whole-genome shotgun sequence data for five model organisms. *Scientific Data* **1**, 140045 (2014).
36. Ralser, M. *et al.* The *Saccharomyces cerevisiae* W303–K6001 cross-platform genome sequence: insights into ancestry and physiology of a laboratory mutt. *Open Biol.* **2**, 120093 (2012).
37. Arabidopsis Genome Initiative. Analysis of the genome sequence of the flowering plant *Arabidopsis* thaliana. *Nature* **408**, 796–815 (2000).
38. Hoskins, R.A. *et al.* Sequence finishing and mapping of *Drosophila melanogaster* heterochromatin. *Science* **316**, 1625–1628 (2007).
39. Weber, J.L. & Myers, E.W. Human whole-genome shotgun sequencing. *Genome Res.* **7**, 401–409 (1997).
40. Lander, E.S. *et al.* Initial sequencing and analysis of the human genome. *Nature* **409**, 860–921 (2001).
41. Venter, J.C. *et al.* The sequence of the human genome. *Science* **291**, 1304–1351 (2001).
42. Steinberg, K.M. *et al.* Single haplotype assembly of the human genome from a hydatidiform mole. *Genome Res.* **24**, 2066–2076 (2014).
43. The MHC Sequencing Consortium. Complete sequence and gene map of a human major histocompatibility complex. *Nature* **401**, 921–923 (1999).
44. Huddleston, J. *et al.* Reconstructing complex regions of genomes using long-read sequencing technology. *Genome Res.* **24**, 688–696 (2014).
45. Phillippy, A.M., Schatz, M.C. & Pop, M. Genome assembly forensics: finding the elusive mis-assembly. *Genome Biol.* **9**, R55 (2008).
46. Adams, M.D. *et al.* The genome sequence of *Drosophila melanogaster. Science* **287**, 2185–2195 (2000).
47. Salzberg, S.L. *et al.* GAGE: a critical evaluation of genome assemblies and assembly algorithms. *Genome Res.* **22**, 557–567 (2012).
48. Ewing, B., Hillier, L., Wendl, M.C. & Green, P. Base-calling of automated sequencer traces using phred. I. Accuracy assessment. *Genome Res.* **8**, 175–185 (1998).
49. Kaminker, J.S. *et al.* The transposable elements of the *Drosophila melanogaster* euchromatin: a genomics perspective. *Genome Biol.* **3**, research0084 (2002).
50. McCoy, R.C. *et al.* Illumina TruSeq synthetic long-reads empower *de novo* assembly and resolve complex, highly-repetitive transposable elements. *PLoS ONE* **9**, e106689 (2014).
51. Mewes, H.W. *et al.* Overview of the yeast genome. *Nature* **387**, 7–65 (1997).
52. Blasco, M.A. Telomeres and human disease: ageing, cancer and beyond. *Nat. Rev. Genet.* **6**, 611–622 (2005).
53. George, J.A., DeBaryshe, P.G., Traverse, K.L., Celniker, S.E. & Pardue, M.L. Genomic organization of the *Drosophila* telomere retrotransposable elements. *Genome Res.* **16**, 1231–1240 (2006).
54. Jurka, J. *et al.* Repbase Update, a database of eukaryotic repetitive elements. *Cytogenet. Genome Res.* **110**, 462–467 (2005).
55. Koch, P., Platzer, M. & Downie, B.R. RepARK–*de novo* creation of repeat libraries from whole-genome NGS reads. *Nucleic Acids Res.* **42**, e80 (2014).
56. Schwartz, D.C. *et al.* Ordered restriction maps of *Saccharomyces cerevisiae* chromosomes constructed by optical mapping. *Science* **262**, 110–114 (1993).
57. Burton, J.N. *et al.* Chromosome-scale scaffolding of de novo genome assemblies based on chromatin interactions. *Nat. Biotechnol.* **31**, 1119–1125 (2013).
58. Kaplan, N. & Dekker, J. High-throughput genome scaffolding from *in vivo* DNA interaction frequency. *Nat. Biotechnol.* **31**, 1143–1147 (2013).
59. Böhringer, S., Gödde, R., Böhringer, D., Schulte, T. & Epplen, J.T. A software package for drawing ideograms automatically. *Online J. Bioinform.* **1**, 51–61 (2002).
60. Bankevich, A. *et al.* SPAdes: a new genome assembly algorithm and its applications to single-cell sequencing. *J. Comput. Biol.* **19**, 455–477 (2012).
61. PacBio DevNet. Pacific Biosciences DevNet Datasets https://github.com/PacificBiosciences/DevNet/wiki/Datasets (2014).

ONLINE METHODS

Background. An overlap is typically defined as a maximally scoring alignment between two strings that allows arbitrary orientation and offset of the reads. For two reads S_1 and S_2, both of length L, the fastest method for determining their optimal alignment is Smith-Waterman (SW) dynamic programming, which has a computational complexity of $O(L^2)$[62]. Thus, to naively find all overlapping pairs of N reads would take $O(N^2L^2)$.

In order to reduce the number of pairs that need to be directly evaluated by SW, each read can be broken into a set of k-length substrings (k-mers) by sliding a k-sized window along the length of the read. Thus, a read S of length L is represented by a set of $K = L - k + 1$ k-mers. The k-mers of S can be indexed in a hash table (or suffixes of S in a suffix array) in $O(NK)$ time. This index can be used to filter out potentially nonmatching pairs faster than a direct $O(N^2)$ comparison. To compare two reads, the number of shared k-mers is computed by counting how many k-mers in a query read match k-mers in other reads[28,32,34,63]. The overall time, when comparing against multiple reads, can be improved by using a suffix array to index all existing reads[28]. Overlapping algorithms designed for high-identity reads, such as the original Celera Assembler overlapper[34], may trigger a banded SW search based on a single shared k-mer. However, using any of these approaches, the complexity is proportional to the read length.

For reasons that will become apparent below, we focus our discussion on the hash table indexing approach, though similar reasoning applies to the suffix array approaches. In the case of hash table indexing, all k-mers are first hashed into an integer fingerprint for easier indexing and faster comparison. We define this set of $K = L - k + 1$ integers as $\Gamma(S)$, where $|\Gamma(S)| = K$ is the set's cardinality. Here we assume that the integer size is large enough that a chance of a random collision is negligible. The integers from all strings are hashed into a table such that each hash table bucket will contain a list of all string indices that contain that specific integer fingerprint. In order to find all the strings that have at least w shared k-mers with S, each element in $\Gamma(S)$ is found in the hash table, and a count table is maintained for any read that has at least one matching k-mer. Only the strings in the count table that have $\geq w$ counts are returned. Observe that the time to maintain the count table is directly related to the sum of all counts in the count table and is an additional cost to the hash table lookup of all k-mers.

We refer to the above as the "direct" method and demonstrate how using noisy, long reads can rapidly degrade its performance. Although straightforward, this approach is actively used in efficient alignment algorithms for short-read sequencing[31] and is a viable alternative to other indexed structures like suffix arrays[64]. As an alternative, we present a probabilistic dimensionality reduction and filtering algorithm that theoretically improves runtime and storage requirements over direct methods or suffix-array approaches such as BLASR[28].

Matching k-mer probabilities. Given two random reads generated from a finite alphabet set Σ (e.g., {A,C,G,T}) of length L and an integer $k \leq L$, each read contains K k-mer fingerprints, where the expected number of fingerprints that are shared by these two reads, $E[X_r]$, is equal to the probability that a random k-mer is in the first read,

$$P_{rand} = 1 - (1 - |\Sigma|^{-k})^K \tag{1}$$

multiplied by the number of k-mers in the second read,

$$E[X_r] = P_{rand}K \tag{2}$$

Similarly, the expected number of k-mers that are shared by two reads of arbitrary length within an overlapping $M \leq K$ size k-mer region, $E[X_c]$, corrupted by ε error at each position, is related to the probability that an overlapping k-mer is not corrupted in both reads,

$$P_{ovl} \approx \left[(1-\varepsilon)^2 + \varepsilon^2 \frac{1}{|\Sigma| - 1}\right]^k \tag{3}$$

(assuming substitution errors, for insertions or deletions the probability is slightly different, but on the same order), multiplied by the length of the overlap

$$\begin{aligned} E[X_c] &= (P_{ovl} + P_{rand} - P_{ovl}P_{rand})M + P_{rand}(L - M) \\ &= (P_{ovl} - P_{ovl}P_{rand})M + P_{rand}L \end{aligned} \tag{4}$$

The sensitivity and accuracy of using k-mer counts for filtering overlapping pairs, using any data structure, is dependent on how accurately one can classify if a k-mer count between any two reads is coming from either X_r or X_c.

Given the small DNA alphabet, from equation (2) and (4), we observe that: (i) for large L the value of $E[X_r]$ starts to approach that of $E[X_c]$, and determining if two reads are actually overlapping (or random matches) based on the k-mer count becomes more error-prone; (ii) the number of k-mer hash lookups required for each read grows with L; (iii) the percentage of actual reads that collide for each specific k-mer lookup grows as $1-(1-4^{-k})^L$. So, for example, whereas for $K = 200$, one expects on average that approximately 0.02% of total sequences match any given 10-mer, for $K = 50{,}000$ this grows to approximately 5% of total reads, with the probability of having a 10-mer match for every read at least once, when performing K 10-mer lookups, ≈100%. Unless k is large enough for a specific L, the computational complexity of looking up a length L read in a data structure is not $O(K)$, but rather $O(KN)$. In other words, the computational cost is related to the time it takes to maintain the count table, rather than the overhead of lookups in the data structure storing the k-mers. Therefore, the resulting number of operations required for an all-to-all lookup is $O(KN^2)$, with the constant decreasing exponentially with k. Importantly, the maximum k is limited by the error rate of the reads and the size of the overlap. Thus, ε, L and M bound the performance of an all-to-all lookup, regardless of the data structure used. Below we will demonstrate that when using MinHash sketches, instead of a full k-mer set representation of S, we significantly decrease K, and by extension the complexity of the computation (**Fig. 2a**).

Jaccard similarity. Jaccard similarity is a measurement related to the actual k-mer count, defined as

$$J(S_1, S_2) = \frac{|\Gamma(S_1) \cap \Gamma(S_2)|}{|\Gamma(S_1) \cup \Gamma(S_2)|} \tag{5}$$

Assuming that the number of repeating k-mers (or hash value collisions) is negligible, so both reads have K k-mers,

$$J(S_1, S_2) = \frac{w}{|\Gamma(S_1)| + |\Gamma(S_2)| - w} = \frac{w}{2K - w} \tag{6}$$

where w is the number of k-mer matches between S_1 and S_2. Assuming k is large enough and ε is small enough so that the probability of a random match is sufficiently small, the Jaccard similarity between two overlapping reads is proportional to the percentage of sequence length that is shared between the two reads, and thus independent of L. However, this is a simplification for the purpose of analysis, because in practice the majority of false k-mer collisions are due to repetitive sequence, not hash value collisions. Nonetheless, for highly dissimilar matches the Jaccard similarity approaches $w/(2K)$ and for highly similar matches it approaches 1.

The Jaccard similarity between two L-sized reads, as well as k-mer match count, can be directly computed in $O(L \log L)$ time by using a "sort-merge" algorithm, where we first sort $\Gamma(S)$ of the two reads and then count the number of matching k-mers by performing a merge operation. DALIGNER[29] uses a radix sort for a similar sort-merge operation. For multiple comparisons the cost of sorting is amortized, so a direct-read comparison complexity drops to $O(L)$ time.

Modified sort-merge algorithm. During read overlapping we are looking for similarity in the overlapping region, rather than the averaged similarity over the full length of the reads, as would be measured by the Jaccard similarity. In addition, assembly algorithms require the approximate position and size of the overlap region for building the read layout.

In order to compute the overlap score and region, we modify the basic sort-merge algorithm by also storing the indices of the k-mer position along with the k-mer fingerprint. In case a k-mer in S_1 matches multiple k-mers in S_2, we store only the match with the earliest position in the read, thus guaranteeing O(L) operations. The relative offset of the sequences S_1 and S_2 is estimated by the median difference in the positions of matching k-mers. The k-mer counting algorithm is then run again, constrained to the overlapping region, in order to get the final k-mer count, and the bounds of the overlap are computed from the positions of matching k-mers by taking a uniformly minimum-variance unbiased (UMVU) estimator assuming k-mer matches come from a continuous uniform distribution with unknown start and end points[65]. Based on the matching k-mer count w' in the overlap and overlap size M, we define the overlap similarity as

$$Sim(S_1,S_2) = \frac{w'}{M} \tag{7}$$

This approach reduces the complexity of computing the similarity between two reads from O(L^2) to O(L). Additionally, the O(L) runtime is asymptotically faster than the downstream steps, so filtering false matches early does not asymptotically increase the computational complexity of assembly.

MinHash. Even with the above improvements for computing sequence similarity, we are still left with the problem of computing it for all possible pairs. It is possible to use a hash table or suffix array to accelerate the algorithm[28]. However, we are still faced with a rapid decay in computational efficiency for large L (see above). To significantly decrease the number of table lookups per read, as well as the time it takes to build the lookup table, MHAP uses a probabilistic dimensionality reduction approach called MinHash[20]. The efficiency of MinHash versus a direct computation of Jaccard similarity comes from reducing a read from K integer fingerprints, to some smaller, random, possibly repeating vector of H fingerprints, where $H << K$. We refer to this compressed representation of the string as a sketch. As we only consider H values for any given read, the storage and read lookup cost decrease proportionally with the size of the sketch (**Fig. 2c**).

Observe that the probability of $\Gamma(S_1)$ and $\Gamma(S_2)$ having the same minimum value (or *min-mer*) is equal to the probability that a k-mer in at least one of the strings also exists in both strings. This is equal to the Jaccard similarity J(S_1,S_2)[20]:

$$P[\min\Gamma(S_1) = \min\Gamma(S_2)] = J(S_1,S_2) = \frac{|\Gamma(S_1)\cap\Gamma(S_2)|}{|\Gamma(S_1)\cup\Gamma(S_2)|} \tag{8}$$

Therefore, given H differently seeded hash functions, the probability that the number of min-mers shared by S_1 and S_2 is at least x, can be computed using the cumulative density function of the binomial distribution,

$$f(x;H;p) = 1-\sum_{i=0}^{x-1}\binom{H}{i}p^i(1-p)^{H-i} \tag{9}$$

where p is the Jaccard similarity of the two sequences (equation (8)). Plots for the percentage of read matches that share at least x min-mers, as a function of number of hashes H, are shown in **Figure 2**. The sketch of S is a vector **s** of H min-mers, where each min-mer s_i is the minimum hash value of $\Gamma_i(S)$.

MHAP implementation details. The MHAP algorithm is a two-stage filter that combines the ideas described above. First it filters reads based on shared min-mer counts using MinHash. Then, for all reads that pass the MinHash filter, it again filters matches based on the k-mer count for the overlapping region using the sort-merge algorithm to obtain a more accurate similarity estimate. For accuracy, the value of k for the second filter can be smaller than the value for the MinHash filter, as we only perform this operation on read pairs that passed the MinHash filter. We refer to the first filter k-mer size as k_1 and the second as k_2.

The first filter generates H fingerprints for all k_1-mers of a read using the MurmurHash3 hash function implementation in the Guava library (for larger k values a rolling hash can be used). Next, we use the MurmurHash3 fingerprint as a seed into a computationally efficient XORShift random number generator to generate H random fingerprints. Highly repetitive k-mers, comprising over 0.001% of the total bases, can be computed beforehand and ignored. The fingerprint needed for the second filter is computed using the same MurmurHash3, and the fingerprints are sorted and stored in a data structure associated with each S, along with the H min-mers.

The hth min-mer for each read is hashed into the hth hash table, which is shared between all reads. The hash tables are maintained in memory, unless there are too many reads. In that case, the computation is treated as a parallel all-to-all computation, where only a subset of reads are hashed and compared to all reads streamed directly from the drive. This subset-to-all comparison is repeated for all nonintersecting subsets, either in parallel or serially, depending on the computation resources available.

For each read we find all the reads that are similar by looking up the H min-mers in the hash tables. The hth min-mer is looked for in the hth hash table, and the counts are aggregated into one list. Any read that has at least x min-mers in common with the lookup read is propagated to the second-stage filter. The second-stage filter is implemented as described above, but to account for the high Indel rate exhibited by SMRT sequencing, the overlap region is extended by 30% on either side before counting all k-mer matches in the overlap. After counting, the bounds of the overlap are computed using the UMVU estimators, as before.

Since MHAP was designed for complex genomes with large N and many repeats, k_1 is set to 16 to decrease the number of false positives in each hash table bucket. k = 16 is the largest value that can be effectively hashed into a 32-bit fingerprint while providing good sensitivity (**Supplementary Table 2**). Much larger k-mer values would significantly degrade sensitivity, while slightly larger values would double memory usage (requiring a 64-bit fingerprint), without providing significant improvements. Two sketch sizes were chosen (fast = 512 or sensitive = 1,256) based on **Supplementary Table 2**, to achieve sensitivities required for assembly. MHAP fast was used for all assemblies presented here, except CHM1, and is suitable for most scenarios. MHAP sensitive improves assembly performance, at the expense of speed, when coverage is low. For the second stage filter, k_2 was empirically set to 12 to avoid false-positive matches in 100 kbp reads. Using these settings for SMRT sequencing reads, approximately 40% of overlaps detected using the first filter also pass the second filter. All MHAP parameters are adjustable, and tuning them for specific SMRT chemistries or different sequencing technologies could further improve performance. Further guidance for turning MHAP parameters is given in **Supplementary Note 2**.

Evaluating sensitivity and specificity. Sensitivity and specificity were evaluated based on overlaps inferred from reference mapping. An automated script was used to evaluate MHAP, BLASR and DALIGNER results. To avoid an all-versus-all comparison of reads, random sampling was used to estimate performance. For sensitivity, a random sequence was selected and all sequences with an overlap above a minimum length threshold (2 kbp) were extracted from the reference matches. Any missing overlap was considered a false negative whereas an existing overlap was a true positive. To compute positive predictive value (PPV), a random overlap above the minimum length threshold was evaluated by comparing it to the reference mapping. To address repeat-induced overlaps, if no reference mapping was found for an overlap, a full local alignment was performed on the overlapping region using Smith-Waterman with a match/mis-match penalty of +1/−1. The overlap was then marked true if the resulting alignment matched ≥70% of the all bases across the predicted overlapping region. The PPV was computed as the number of true overlaps divided by the number of overlaps evaluated. The sample size required to estimate sensitivity, PPV and specificity to ± 1 was calculated using Clopper-Pearson (**Supplementary Note 9**).

Correcting noisy reads. Noisy reads can be accurately corrected using a multiple alignment of reads sampled from the same region of the genome so long as the sequencing errors are random and independent between reads[13,18]. The correction process consists of two general steps, building a multiple read alignment (MSA) and deciding the correct bases from the MSA. This strategy was first used to correct noisy, long reads using accurate, short reads[13]. With the invention of new consensus algorithms, this hierarchical approach has also been successfully applied to the long reads themselves[15]. The PBcR

doi:10.1038/nbt.3238

pipeline supports two consensus modules: a slower but more accurate method called PBDAGCon and a faster but less robust method called FalconSense. PBDAGCon, described previously[15], uses a directed acyclic graph (DAG) to construct a partial order alignment[66]. Finding an optimized path through the DAG generates a robust consensus. Alternatively, the FalconSense algorithm accelerates consensus generation by aligning reads to a template sequence that is the target of correction[33]. Individual matches and indels are then tagged and sorted to determine a consensus sequence with high support[67] (**Supplementary Note 10** and **Supplementary Fig. 13**). This read correction algorithm was used for all assemblies presented here, except for CHM1, for which PBDAGCon performed better correction due to the lower coverage and higher error rate of the raw data.

Assembling corrected reads. The Celera Assembler PBcR pipeline is a hierarchical, nonhybrid assembly pipeline[11] that corrects and assembles raw PacBio SMRT sequences. By default, the longest 40× of data is corrected using all provided sequences. During the correction phase, either MHAP or BLASR was used (as indicated) to compute overlaps of the uncorrected reads. After read correction, the longest 25× of corrected reads were assembled using Celera Assembler 8.2. Default PBcR parameters were used for all assemblies, except CHM1, which used more sensitive MHAP parameter settings for the initial overlapping, PBDAGCon for read correction and relaxed error rates for Celera Assembler. The spec files and commands used for all assemblies are reproduced in **Supplementary Note 11**. Unless otherwise noted, all PBcR assemblies were polished using Quiver to optimize consensus accuracy.

Assembly validation. All assembled genomes were aligned to their respective reference using Nucmer[68] and validated using the original GAGE[47] scripts. To accelerate the alignment of large genomes, Nucmer defaults were modified to increase the minimum seed size and cluster size, as well as to use only unique seeds in the reference genome. Resulting alignments and validation statistics are reported in **Supplementary Figs. 5–8**, **12** and **Supplementary Table 5**. To estimate QV for *E. coli* and *D. melanogaster* before and after Quiver polishing, assemblies were mapped as above and single-nucleotide polymorphism (SNP) counts tabulated from show-snps in the dnadiff[45] package. For *H. sapiens*, the QV was estimated using both SNP and Indel variants identified by mapping Illumina short-read sequences to the assembly (**Supplementary Note 6**).

Sequencing cost and assembly using cloud computing. Assuming the most current P6C4 SMRT sequencing and a conservative per-cell throughput of 750 Mbp, sequencing *D. melanogaster* to 100× coverage would require 10 µg of DNA, a single sequencing library and approximately 20 SMRTcells. This equates to a sequencing cost of ~$7,000 (https://dugsim.net/) for the *D. melanogaster* genome, with costs scaling linearly for larger genomes.

To expand access to the PBcR-MHAP assembly pipeline, we have provided a free public AWS EC2 Amazon Machine Instance (AMI) as well an Amazon Elastic Block Storage (EBS) snapshot with the sequencing data and example assemblies of *E. coli, S. cerevisiae* and *D. melanogaster*. Supporting documentation for how to reproduce the assemblies presented here is available from http://www.cbcb.umd.edu/software/PBcR/MHAP/aws.html. A nonexpert user was able to reproduce the MHAP *D. melanogaster* assembly (without Quiver) in less than 10 h at a cost of <$300. Quiver polishing this assembly would roughly double the cost. Allocating additional compute nodes, which would marginally increase costs, could further reduce assembly time. For *E. coli*, the total cost of assembly and Quiver polishing is less than $2.

62. Smith, T.F. & Waterman, M.S. Identification of common molecular subsequences. *J. Mol. Biol.* **147**, 195–197 (1981).
63. Rasmussen, K.R., Stoye, J. & Myers, E.W. Efficient q-gram filters for finding all epsilon-matches over a given length. *J. Comput. Biol.* **13**, 296–308 (2006).
64. Manber, U. & Myers, G. Suffix arrays: a new method for on-line string searches. *SIAM J. Comput.* **22.5**, 935–348 (1993).
65. Cheng, R.C.H. & Amin, N.A.K. Estimating parameters in continuous univariate distributions with a shifted origin. *J. R. Stat. Soc., B* **45**, 394–403 (1983).
66. Lee, C., Grasso, C. & Sharlow, M.F. Multiple sequence alignment using partial order graphs. *Bioinformatics* **18**, 452–464 (2002).
67. Anson, E.L. & Myers, E.W. ReAligner: a program for refining DNA sequence multi-alignments. *J. Comput. Biol.* **4**, 369–383 (1997).
68. Kurtz, S. *et al*. Versatile and open software for comparing large genomes. *Genome Biol.* **5**, R12 (2004).

Corrigendum: Assembling large genomes with single-molecule sequencing and locality-sensitive hashing

Konstantin Berlin, Sergey Koren, Chen-Shan Chin, James P Drake, Jane M Landolin & Adam M Phillippy
***Nat. Biotechnol.* 33, 623–630 (2015); published online 25 May 2015; corrected after print 6 October 2015**

In the version of this article initially published, equation 9 appeared incorrectly as:

$$f(x;H,p) = 1 - \left[\binom{H}{x-1} p^{x-1}(1-p)^{H-x}\right]$$

The equation has been corrected in the HTML and PDF versions of the article.