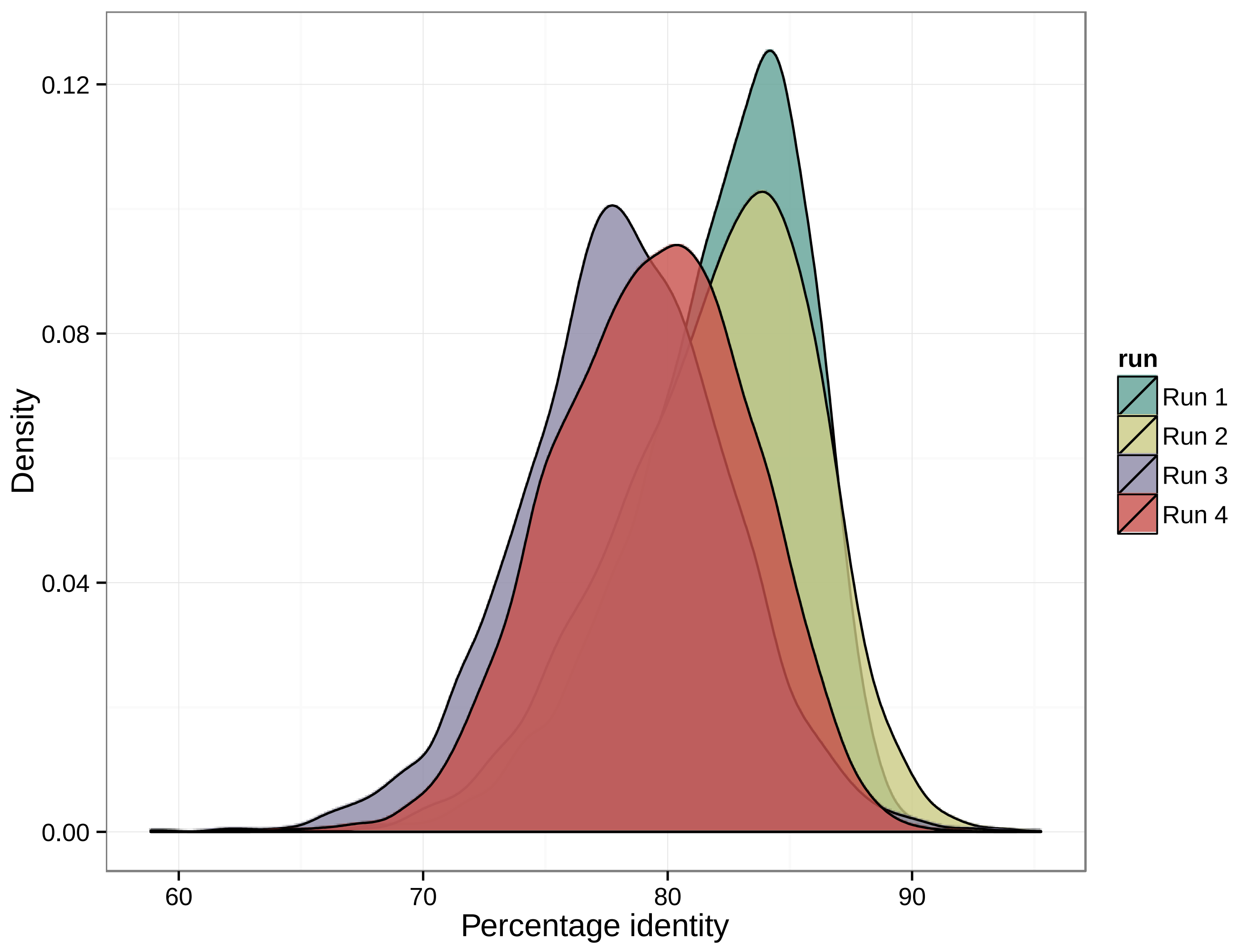
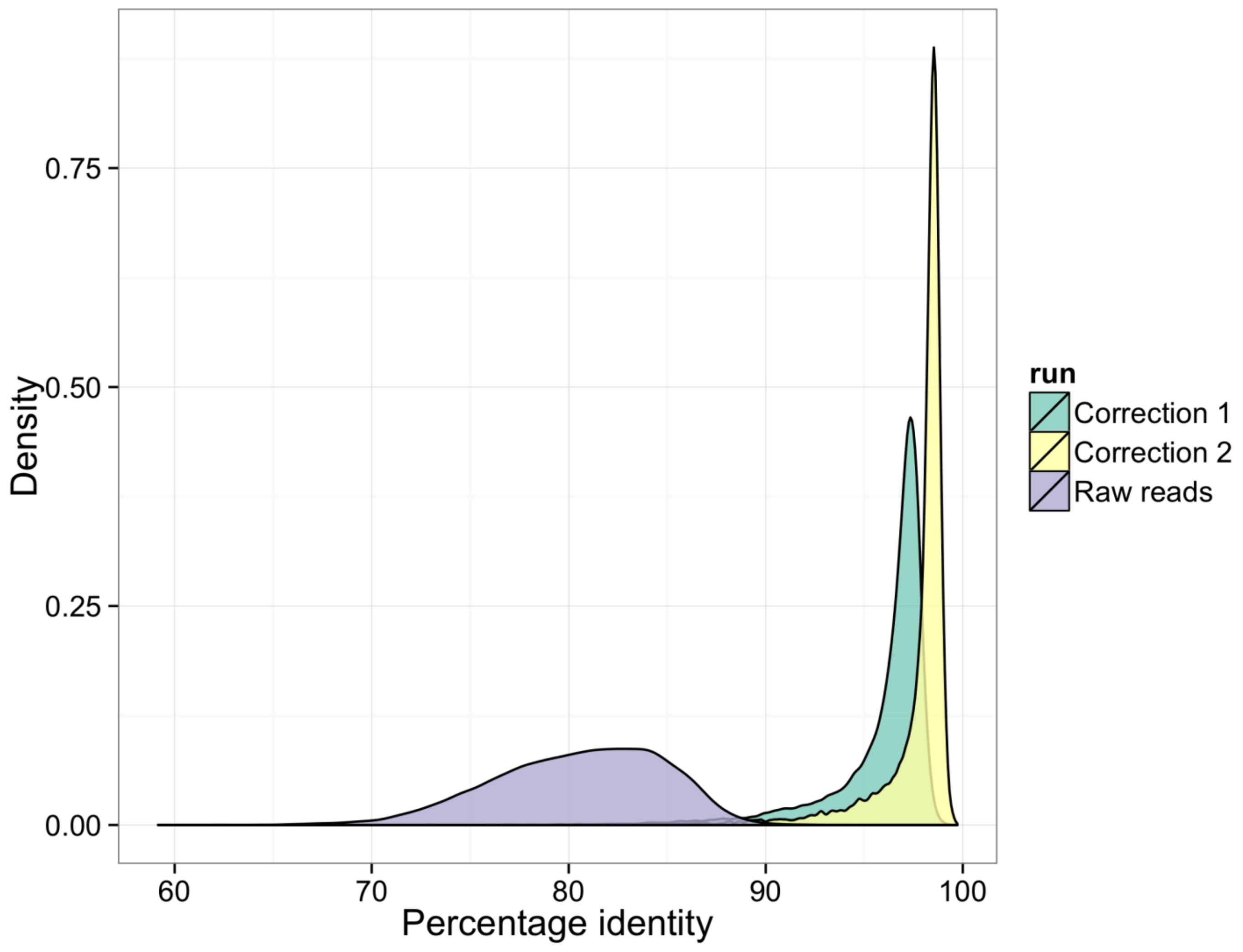


Supplementary Figure 1

Kernel density plot showing the accuracy of reads from the four individual MinION runs used to generate the *de novo* assembly.

The mean accuracy varies from 78.2% (run 3) to 82.2% (run 1).



Supplementary Figure 2

Kernel density plot demonstrating the raw nanopore read accuracy and effect of two rounds of error correction on accuracy.

The mauve area represents uncorrected sequencing reads, where the green area shows the improvement in accuracy after the first round of correction and the yellow shows improvement from the second round of correction. Further rounds of correction did not improve the accuracy further.

```
merSize = 14
cgwErrorRate = 0.25
ovlCorrBatchSize = 100000
ovlHashBlockLength = 50000000
frgCorrThreads = 4
frgMinLen = 1000
merylMemory = 16384
cnsMinFrags = 7500
ovlStoreMemory = 16384
ovlMinLen = 40
merylThreads = 96
ovlErrorRate = 0.04
ovlThreads = 16
ovlRefBlockSize = 100000000
frgCorrBatchSize = 100000
unitigger = bogart
ovlHashBits = 25
ovlConcurrency = 48
cnsConcurrency = 24
cnsErrorRate = 0.25
```

Supplementary Figure 3

Spec file for Celera Assembler.

A complete bacterial genome assembled *de novo* using only nanopore sequencing data

Nicholas J. Loman
Joshua Quick
Institute of Microbiology and Infection, University of Birmingham

Jared T. Simpson
Ontario Institute for Cancer Research

1 Supplementary Note

1.1 Signal-level Consensus Algorithm

The electrical current signal sampled by the MinION inherently contains more information than the base-called reads. We sought to use this information when computing the final consensus sequence of our assembly. In this section we describe our consensus algorithm. We first give a brief overview of the algorithm. We start with our initial draft assembly, G, and progressively modify it by making small localized changes. We assess whether the modified sequence, G', increases the probability of the electrical current data for a set of reads. If the probability of the signal data increases we accept G' and iterate. To make these calculations computationally feasible, this process is run over short segments of the genome and seeded by reads aligned to G. At the core of this algorithm is calculating the probability of the raw signal data emitted by the MinION given a proposed consensus sequence. We first describe our probabilistic model of the data, then describe how we use this model to compute the consensus sequence for the assembly.

Signal-level MinION Data

The MinION continuously samples electrical current flow across the pore. When DNA is detected in the pore, these current samples are turned into *events* by a feature detection algorithm provided by the instrument's software. The feature detection algorithm is designed to find stepwise changes in the current signal that indicate that the DNA sequence has moved through the pore. For each event the feature detection algorithm records the mean of the current samples (in picoamps), its standard deviation and the total duration of the event (in seconds). Simulated idealized data to illustrate these concepts is shown in Figure S1 and Table S1.

We only use the mean current level in our model but remark that the duration and standard deviation are sources of information that should be considered in future work. When we refer to individual events e_i below we are referring to the mean current level component of the event only. For the purpose of our consensus algorithm, we consider a MinION read to be a sequence of events $(e_1, e_2, ..., e_n)$.

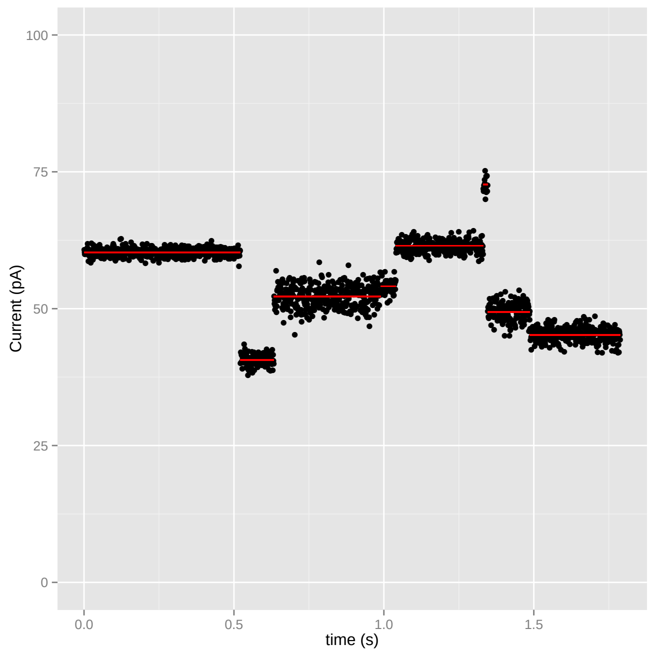


Figure S1: Simulated ideal signal data to illustrate the data that is input into our model. The black points are sampled current levels at a given time. The red lines are the events detected by a feature detection algorithm, which partitions the samples into discrete segments.

Modeling Events

The fundamental principle of nanopore-based sequencing is that the amount of current flow through the pore depends on the sequence of DNA that resides in the pore [3, 5]. The MinION model assumes that the current level measured for each event is drawn from a Gaussian distribution with mean and variance dependent on the 5-mer sequence that is in the pore when the samples are taken. Oxford Nanopore provides the parameters for the Gaussians for all 5-mers in the FAST5 files describing each read. We refer to this collection of parameters as $\mathbf{\Theta}$. The probability of observing event e_i given that 5-mer k is the true sequence in the pore is $P(e_i|k, \mathbf{\Theta}) = P(e_i|k, \mu_k, \sigma_k) = \mathcal{N}(\mu_k, \sigma_k^2)$.

Event	mean current (pA)	current std dev	duration (s)
1	60.3	0.7	0.521
2	40.6	1.0	0.112
3	52.2	2.0	0.356
4	54.1	1.2	0.051
5	61.5	1.0	0.291
6	72.7	1.5	0.015
7	49.4	1.5	0.141
8	45.2	1.2	0.301

Table S1: Events detected from the idealized example shown in Figure S1.

A Probabilistic Model of MinION Events

We want to calculate the probability of observing an event sequence given a known DNA sequence S, $P(e_1, e_2, ..., e_n|S, \mathbf{\Theta})$. To do this, we represent the known DNA sequence S as its constituent 5-mers, $S = (s_1, s_2, ..., s_m)$. The simplest generative model of the data is where the sequencer moves stepwise from 5-mer to 5-mer, and emits a single event from each 5-mer. This situation is depicted in Figure S2.

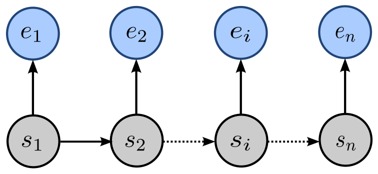


Figure S2: A simple model where each 5-mer (grey circles) emits a single event (blue circles).

In this situation, where there is a perfect one-to-one match between events and 5-mers, calculating the probability of the data is straightforward if we assume independence between events:

$$P(e_1, e_2, ..., e_n|s_1, s_2, ..., s_n, \mathbf{\Theta}) = \prod_{i=1}^{n} P(e_i|s_i, \mu_{s_i}, \sigma_{s_i}) \tag{1}$$

In reality however, we need to account for two types of event detection errors: *skipped* 5-mers, where the system does not detect an event for a 5-mer that transited the pore, and *split events*, where the signal for a single 5-mer is emitted multiple times, presumably because of transient noise that exceeds the event detection threshold.

In this situation, we no longer know which 5-mer emitted each event. To provide motivation for our model we will describe an example from the more complex generative process. Let $S =$ (`ACGTA`, `CGTAA`, `GTAAC`, `TAACT`) be the 5-mers of the known DNA sequence. The process starts at the first 5-mer of S, ACGTA, and emits an event drawn from $\mathcal{N}(\mu_{\texttt{ACGTA}}, \sigma^2_{\texttt{ACGTA}})$. We then choose to stay at ACGTA, move to the next 5-mer, CGTAA, or skip CGTAA and move directly to GTAAC. Once a move has been made we can emit another event drawn from the appropriate Gaussian and repeat until we reach the end of the sequence. We can model this process using a hidden Markov model with the structure shown in Figure S3.

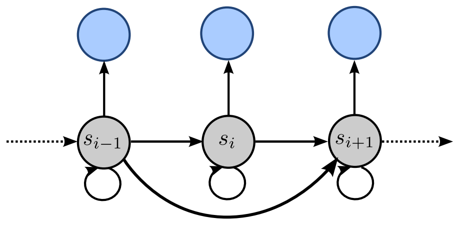


Figure S3: A hidden Markov model allowing self-transitions and jumps over states. This allows us to model split and skipped events.

We will treat the path through the model as the hidden state of the system, π, where each entry is a pair $\pi_i = (i, s_j)$ which denotes that at step i the system emitted event i from 5-mer s_j. We denote the transition probabilities with $P(\pi_i|\pi_{i-1}, S)$ and for convenience define an initial transition $P(\pi_1 = (1, s_1)|\pi_0, S) = 1$. We can then calculate the joint probability of a path and the data:

$$P(\pi, e_1, e_2, ..., e_n|S, \mathbf{\Theta}) = \prod_{i=1}^{n} P(e_i|\pi_i, \mu_{s_i}, \sigma_{s_i})P(\pi_i|\pi_{i-1}, S) \tag{2}$$

We can calculate the probability of the data given S by summing over all possible paths using the Forward algorithm:

$$P(e_1, e_2, ..., e_n|S, \mathbf{\Theta}) = \sum_{\pi} P(\pi, e_1, e_2, ..., e_n|S, \mathbf{\Theta}) \tag{3}$$

The structure of the model and transition probabilities are implied by the 5-mer sequence S as described later.

Profile HMM of Events and 5-mer in the Pore

The model drawn above only allows single 5-mers to be skipped. We could add transitions between more distant 5-mers to allow longer jumps but this increases the computational cost to compute $P(e_1, e_2, ..., e_n|S, \mathbf{\Theta})$ [2]. To allow a similar description of the data in a more computationally convienient structure we can convert this model into a standard Profile HMM. The Profile HMM we use is shown in Figure S4.

The Profile HMM expands the state space to include explicit states for extra events emitted by any 5-mer (E states) and silent states that allow 5-mers to be visited without observing an event (K states). This model has a repeating block structure with 3 states for every 5-mer in S. Each state in a block is indexed by the block's corresponding 5-mer.

A path π through this model consists of states that emit events (incrementing the event index, states M and E) and states that move from 5-mer to 5-mer (incrementing the 5-mer index, states M and K). Each entry in the path π is again represented as a pair $\pi_k = (i, X)$ where i is the index of the last event emitted by the path and $X \in \{M_{s_1}, E_{s_1}, K_{s_1}, ..., M_{s_m}, E_{s_m}, K_{s_m}\}$. When describing transition probabilities we will use the notation $t(M_{s_{j-1}} \rightarrow M_{s_j})$ to mean $P(\pi_k = (i, M_{s_j}) | \pi_{k-1} = (i-1, M_{s_{j-1}}))$ and so on.

To calculate the probability $P(e_1, e_2, ..., e_n | S, \mathbf{\Theta})$ we again use the Forward algorithm which fills in a dynamic programming matrix of size $3mn$ to sum over all paths through the model. We refer the reader to section 5.4 of Durbin et al. [2] for the recursive definition of the dynamic programming matrix for a Profile HMM.

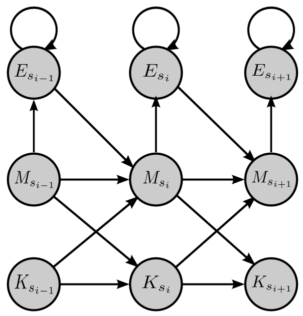


Figure S4: The state structure of the Profile Hidden Markov Model we use to calculate the probability of a sequence of events given a known sequence.

Transition Probabilities

The transition probabilities of the model are crucial to capturing the behaviour of the pore. Most importantly, we must accurately model the probability of not observing a signal for a particular 5-mer. If the signals for two adjacent 5-mers, s_{i-1} and s_i, are similar the feature detection algorithm may not register a change in current. To make this explicit, consider the situation where the sequenced DNA is `GAAAAAAC`. There is no chance to register the `AAAAA` $\rightarrow$ `AAAAA` transition as the current signals for these 5-mers are identical by definition. Therefore the probability of an event sequence generated by S given two candidate sequences $S_1 =$ `GAAAAAAC` and $S_2 =$ `GAAAAAC`, which only differ by the undetectable length of the poly-A run, should be very similar, $P(e_1, e_2, ..., e_n | S_1, \mathbf{\Theta}) \approx P(e_1, e_2, ..., e_n | S_2, \mathbf{\Theta})$.

We model this by making the transition probability to K states a function of the absolute difference between expected current for the adjacent 5-mers

$t(M_{s_{i-1}} \to K_{s_i}) = t(K_{s_{i-1}} \to K_{s_i}) = f(|\mu_{s_{i-1}} - \mu_{s_i}|)$. We use a lookup table for f that was calculated from a sample of reads matched to an earlier draft assembly. The lookup table stores probabilities in bins of 0.5 pA.

While the majority of skipped 5-mers are due to signal similarity, we also observe skips of 5-mers whose signal differs substantially from the preceding 5-mer. These occurrences are presumably due to events that were too short to reliably detect and hence discarded by the feature detection algorithm.

The probability of transitioning to an E state is independent of the 5-mer. Similar to the f table, we learned the probability of emitting an additional event p_{me} from a sample of reads. We calculate $t(M_{s_{i-1}} \to E_{s_{i-1}})$ using p_{me} and the probability of not transitioning to the K state:

$$t(M_{s_{i-1}} \to E_{s_{i-1}}) = p_{me}(1 - t(M_{s_{i-1}} \to K_{s_i}))$$

We also estimated the probability of staying in the E state, p_{ee}, from the same sample of reads and set $t(E_{s_{i-1}} \to E_{s_{i-1}}) = p_{ee}$. Later we describe how these transition probabilities are trained to capture the characteristics of individual MinION reads. The complete set of transition probabilities for our model is as follows:

$$\begin{aligned}
&t(M_{s_{i-1}} \to K_{s_i}) = f(|\mu_{s_{i-1}} - \mu_{s_i}|)\\
&t(M_{s_{i-1}} \to E_{s_{i-1}}) = p_{me}(1 - t(M_{s_{i-1}} \to K_{s_i}))\\
&t(M_{s_{i-1}} \to M_{s_i}) = 1 - t(M_{s_{i-1}} \to K_{s_i}) - t(M_{s_{i-1}} \to E_{s_{i-1}})\\
&t(E_{s_{i-1}} \to E_{s_{i-1}}) = p_{ee}\\
&t(E_{s_{i-1}} \to M_{s_i}) = 1 - t(E_{s_{i-1}} \to E_{s_{i-1}})\\
&t(K_{s_{i-1}} \to K_{s_i}) = f(|\mu_{s_{i-1}} - \mu_{s_i}|)\\
&t(K_{s_{i-1}} \to M_{s_i}) = 1 - t(K_{s_{i-1}} \to K_{s_i})
\end{aligned}$$

Emission Distributions

To complete the model we need emission distributions for each state. The emission distribution for the M state are the Gaussians described above. The emission distribution for the E states are nearly the same however we noted higher variance for the duplicated events so we scale the standard deviation by a factor of $v = 1.75$. As described above, no events are emitted from the K states.

$$P(e_i|\pi_k = (i, M_{s_j})) = \mathcal{N}(\mu_{s_j}, \sigma^2_{s_j}) \tag{4}$$

$$P(e_i|\pi_k = (i, E_{s_j})) = \mathcal{N}(\mu_{s_j}, (v\sigma_{s_j})^2) \tag{5}$$

This completes the description of our HMM.

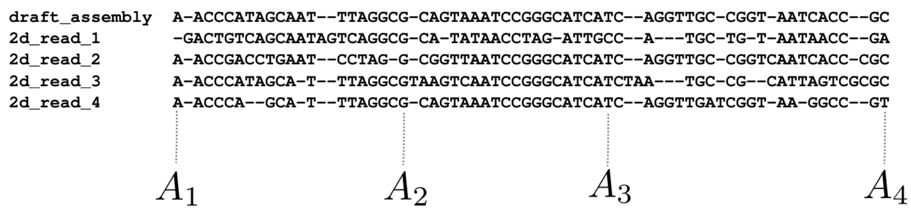
A_1	A_2
(2D_read_1, template, 1872)	(2D_read_1, template, 1896)
(2D_read_1, complement, 7076)	(2D_read_1, complement, 7063)
(2D_read_2, template, 3011)	(2D_read_1, template, 3040)
(2D_read_2, complement, 12072)	(2D_read_1, complement, 12051)

Table S2: An example of the first two reads for the first two anchors shown in Figure S5.

1.2 Computing the Consensus Sequence

Our consensus pipeline begins by mapping the uncorrected 2D MinION reads to the draft assembly, G. To do this we use `bwa mem` with option `-x ont2d` [4]. After the read mapping step, we segment the genome into 10 kilobase segments which overlap by 200bp. For each 10kb segment we extract the subsequence of each read that maps to the segment. The 10kb segment of the the draft genome, and the reads aligned to the segment, are the input into our consensus algorithm.

We select an *anchor* every 50bp of the draft assembly's sequence for the segment. These anchor points define a mapping between bases in the draft assembly and the current signal events for each strand (template and complement) of the aligned 2D MinION reads. Later, when evaluating the probability of the data in our HMM we treat each strand of a 2D read independently. An anchor A_i is a set of tuples specifying a MinION read identifier, a strand identifier and the index of the nearest event for the base aligned to the draft assembly. We use the anchor positions to extract subsequences of the draft assembly, the 2D MinION reads and their template and complement events to input into the HMM. An example is in Figure S5 and Table S2.



```
draft_assembly  A-ACCCATAGCAAT--TTAGGCG-CAGTAAATCCGGGCATCATC--AGGTTGC-CGGT-AATCACC--GC
2d_read_1       -GACTGTCAGCAATAGTCAGGCG-CA-TATAACCTAG-ATTGCC--A---TGC-TG-T-AATAACC--GA
2d_read_2       A-ACCGACCTGAAT--CCTAG-G-CGGTTAATCCGGGCATCATC--AGGTTGC-CGGTCAATCACC-CGC
2d_read_3       A-ACCCATAGCA-T--TTAGGCGTAAGTCAATCCGGGCATCATCTAA---TGC-CG--CATTAGTCGCGC
2d_read_4       A-ACCCA--GCA-T--TTAGGCG-CAGTAAATCCGGGCATCATC--AGGTTGATCGGT-AA-GGCC--GT
```

Figure S5: Anchor positions store mappings between bases and MinION events for every 2D read mapped to a segment. In this example, anchors are selected every 20bp along the draft assembly row.

Training the HMM Transitions

Prior to computing the consenseus sequence of a segment, we adjust the transition probabilities for both strands of the MinION reads. We do this by aligning the events for each read strand to the draft assembly sequence using the Viterbi algorithm, guided by the anchor positions. For each read, we count the number of transitions between the states of the Profile HMM and update the f table, p_{me} and p_{ee} accordingly. As a read may not have have enough events to estimate the f table across the full range of values we use pseudocounts from the original f table to initialize this update.

Consensus Algorithm

The consensus algorithm recomputes the sequence of the draft assembly between anchors A_i and A_{i+2}. After the new consensus sequence is computed, we update the sequence of the assembly and the event-to-assembly mappings for A_{i+1}. This update step uses the Viterbi algorithm to align events to the new consensus sequence. The purpose of this pattern - jumping over an anchor and realigning - is to progressively improve the quality of the event-to-assembly alignments recorded in the anchors.

Let $\mathcal{D} = \{(e_{i,1}, e_{i+1,1}, ..., e_{j,1}), ..., (e_{i,r}, e_{i+1,r}, ..., e_{j,r})\}$ be the set of event subsequences for the r strands of the 2D MinION reads that are anchored at both A_i and A_{i+2}. Our goal is to compute a consensus sequence C that maximizes the probability of the data $\mathcal{D}$. We do this by initializing C to the sequence of the draft assembly between the two anchors. We then propose a set of candidate sequences $\mathcal{C}$ which are derived from the current consensus C. We use our HMM to select the sequence that maximizes the probability of the data:

$$C' = \operatorname*{argmax}_{S \in \mathcal{C}} P(\mathcal{D}|S) \tag{6}$$

where

$$P(\mathcal{D}|S) = \prod_{k=1}^{r} P(e_{i,k}, e_{i+1,k}, ..., e_{j,k}|S, \mathbf{\Theta}) \tag{7}$$

The current consensus sequence is always included in the set of candidates. If the current consensus sequence is selected by equation 6, the process stops.

The set of all possible candidate sequences between A_i and A_{i+2} is too large to test every possibility. Instead, we use two methods to propose small sets of candidate sequences derived from the current consensus sequence C. The first method makes edits to C based on sampling alternative substrings from the reads. The second method, inspired by the Quiver algorithm [1], generates all strings within edit distance of one of C. This pair of methods, the first allowing large changes that must be observed by at least one read, the second exhaustively trying all one base edits, balances between compute time and thoroughly exploring the space of candidate consensus sequences.

The first method, which we call the `block replacement` algorithm, computes the longest common subsequence of 5-mers between C and an aligned 2D read, R. This algorithm partitions C and R into contiguous regions where they share matching 5-mers and regions where they do not. For each pair of consecutive matching regions, we replace the sequence in C with the sequence in R and add this derived sequence to the set $\mathcal{C}$. This algorithm generates a new candidate sequence for each pair of matching regions between C and R and is run for all 2D reads that are aligned between the anchors.

The second method, which we call the `mutation` algorithm, simply makes all one base substitutions, one base insertions and one base deletions to C to generate the candidate set $\mathcal{C}$.

These algorithms are run iteratively. We initialize the current consensus C to the draft genome sequence between the pair of anchors. We then generate candidate set $\mathcal{C}$ using the `block replacement` algorithm. We select the best sequence C' using equation 6. If C' is not C, we set $C \leftarrow C'$ and repeat the process. Once no more improvements to the consensus sequence are made, we run this procedure again except using the `mutation` algorithm to generate $\mathcal{C}$. Once this procedure converges we update the assembly to contain the final consensus sequence. We then update the event alignments for A_{i+1} and move on to the next pair of anchors. This process continues until the entire 10kbp sequence has been processed. Once all 10kbp segments have been processed, which occurs in parallel, we merge the segments together at the overlapping 200bp ends.

Culling Unlikely Candidate Sequences

While we restrict our candidate sequences to those that are close to the current consensus C, we still generate many sequence that are unlikely to be the true sequence of the genome between a pair of anchors. To avoid spending computation time fully evaluating equation 6 for these sequences we periodically cull unlikely sequences from $\mathcal{C}$. We do this by progressively evaluating equation 6 using subsets of k elements of the full data $\mathcal{D}$. We will refer to these subsets as $\mathcal{D}_k$.

To retain a sequence S in $\mathcal{C}$, we require that either $k/5$ event sequences in $\mathcal{D}_k$ have greater probability given S than the current consensus sequence C or that $P(\mathcal{D}_k|S)/P(\mathcal{D}_k|C) > e^{-30}$. If both of these conditions fail, we remove S from $\mathcal{C}$.

These checks are performed after processing 5 event sequences, eg for subsets $\mathcal{D}_5, \mathcal{D}_{10}, ...$ The conservative threshold that we set on the ratio between probabilities is to avoid culling the true sequence from $\mathcal{C}$ at the cost of retaining some unlikely sequences.

Event Preprocessing

The events recorded for a read may differ from the reference pore model that Oxford Nanopore has provided. To account for this the MinION's software calculates parameters to globally shift and scale the mean and variance of the provided 5-mer model to better match the observed data. Additionally a correction is applied to the event signals to correct for a tendency of the signal to drift over time. We apply these corrections, shifts and scales to the pore model and events prior to input into our HMM.

Read Filtering

During development we noticed that some MinION reads contained observations that were outside of the expected range of signals. To discard these reads rather than having the erroneous observations dominate the probability in equation 6, we filtered the input events by computing the probability of the data given the original draft reference sequence and discarding any strands whose log-scaled probability divided by the number of events is greater than 3.5. Better quality control of the input data is an area of future improvement.

References

[1] Chen-Shan Chin, David H Alexander, Patrick Marks, Aaron A Klammer, James Drake, Cheryl Heiner, Alicia Clum, Alex Copeland, John Huddleston, Evan E Eichler, et al. Nonhybrid, finished microbial genome assemblies from long-read smrt sequencing data. *Nature methods*, 10(6):563–569, 2013.

[2] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids.* Cambridge University Press, 1998.

[3] JohnJ. Kasianowicz, Eric Brandin, Daniel Branton, and DavidW. Deamer. Characterization of individual polynucleotide molecules using a membranechannel. *Proceedings of the National Academy of Sciences*, 93(24):13770–13773, 1996.

[4] Heng Li. Aligning sequence reads, clone sequences and assembly contigs with bwa-mem. *arXiv preprint arXiv:1303.3997*, 2013.

[5] David Stoddart, Andrew J. Heron, Ellina Mikhailova, Giovanni Maglia, and Hagan Bayley. Single-nucleotide discrimination in immobilized dna oligonucleotides with a biological nanopore. *Proceedings of the National Academy of Sciences*, 106(19):7702–7707, 2009.

Supplementary Tables

Supplementary Table 1: Oxford Nanopore MinION datasets used in this study.

Description	Pass 2D	Mean length	Accession
R7.3 chemistry, Metrichor 1.9, SQK-MAP-003	8451	6183	ERX708228
R7.3 chemistry, Metrichor 1.9, SQK-MAP-004	2855	5604	ERX708229
R7.3 chemistry, Metrichor 1.9, SQK-MAP-004	5639	3885	ERX708230
R7.3 chemistry, Metrichor 1.9, SQK-MAP-004	5325	8158	ERX708231

Supplementary Table 2: Results from QUAST when comparing the draft circular contig ("circular_draft") and the polished circular contig ("circular_polished") against the *E. coli* K12 MG1655 reference genome (accession `NC_000913.3`).

Assembly	circular_draft	circular_polished
# contigs ($\geq$ 0 bp)	1	1
# contigs ($\geq$ 1000 bp)	1	1
Total length ($\geq$ 0 bp)	4593166	**4638068**
Total length ($\geq$ 1000 bp)	4593166	**4638068**
# contigs	1	1
Largest contig	4593166	**4638068**
Total length	4593166	**4638068**
Reference length	4641652	4641652
GC (%)	51.07	50.94
Reference GC (%)	50.79	50.79
N50	4593166	**4638068**
NG50	4593166	**4638068**
N75	4593166	**4638068**
NG75	4593166	**4638068**
L50	1	1
LG50	1	1
L75	1	1
LG75	1	1

# misassemblies	2	2
# misassembled contigs	1	1
Misassembled contigs length	**4593166**	4638068
# local misassemblies	3	**2**
# unaligned contigs	0 + 0 part	0 + 0 part
Unaligned length	0	0
Genome fraction (%)	100.000	100.000
Duplication ratio	**0.990**	0.999
# N's per 100 kbp	**0.04**	1.14
# mismatches per 100 kbp	80.27	**25.90**
# indels per 100 kbp	921.05	**371.44**
Largest alignment	2739620	**2765784**
NA50	2739620	**2765784**
NGA50	2739620	**2765784**
NA75	1852782	**1871514**
NGA75	1852782	**1871514**
LA50	1	1
LGA50	1	1
LA75	2	2
LGA75	2	2