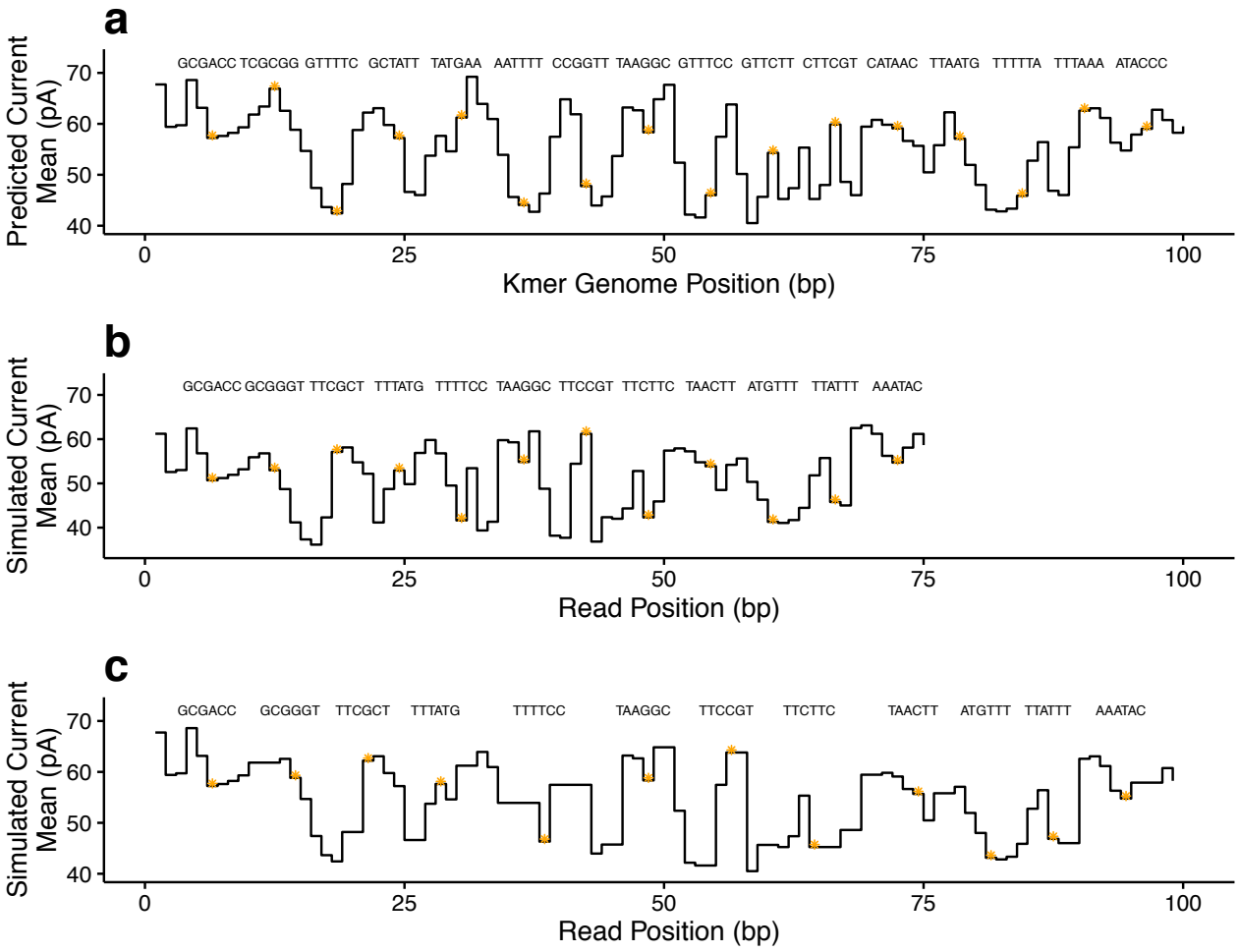
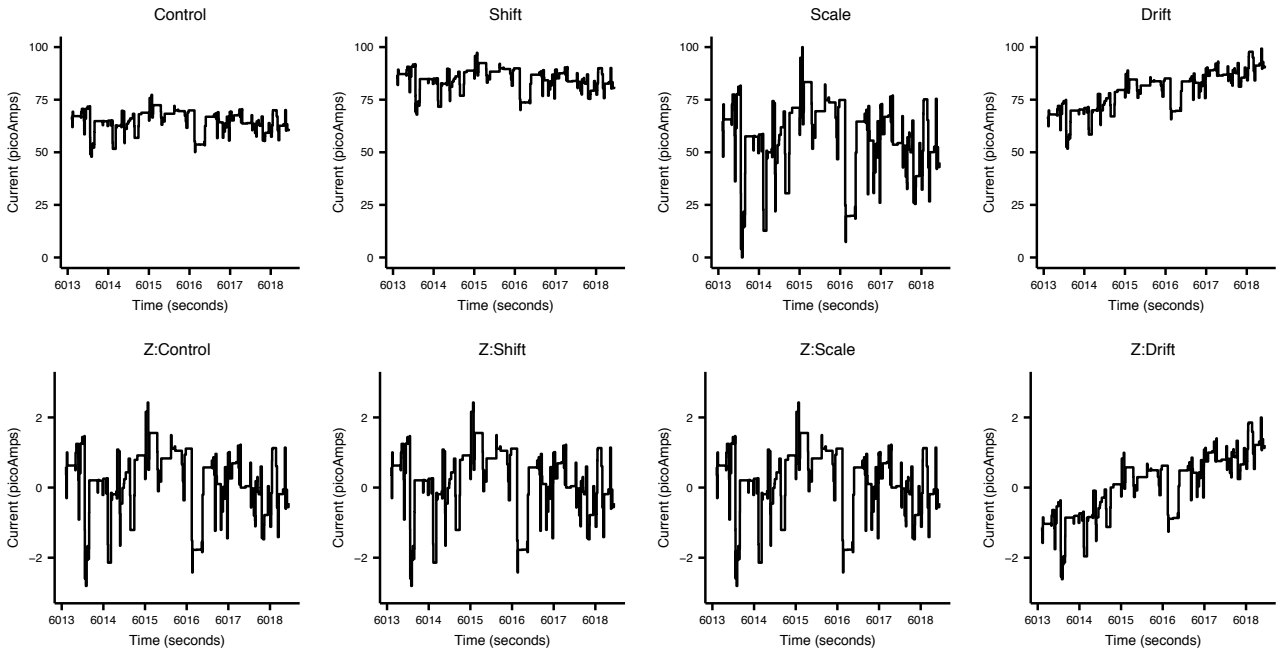
Supplementary Figure 1 - Simulated reference and reads in squiggle space.



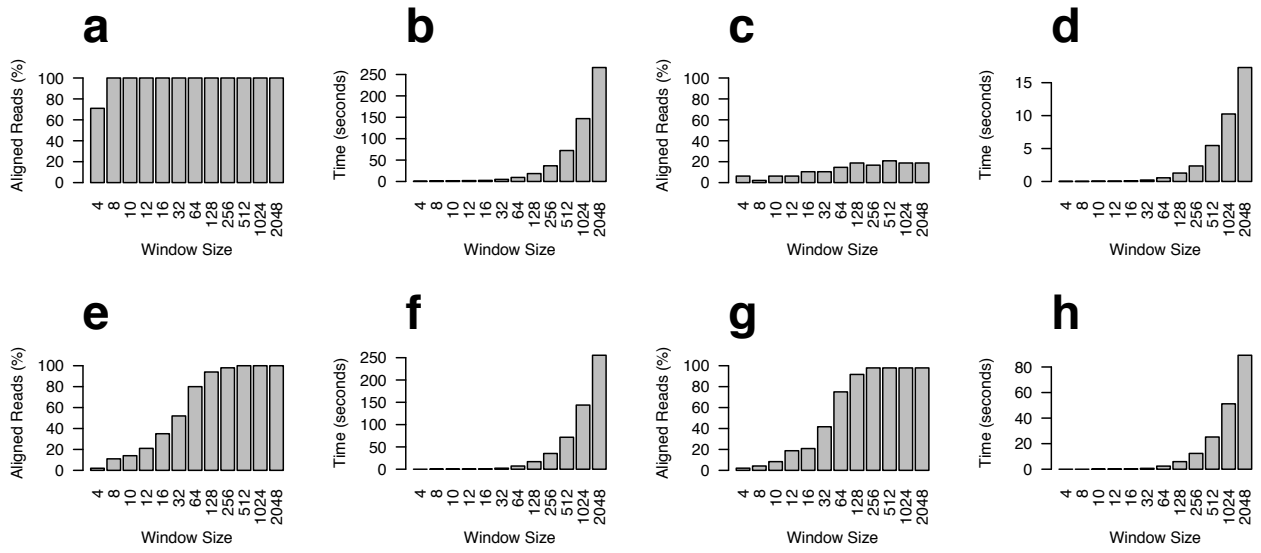
(**a**) shows a model squiggle inferred from the first 100 bp of bacteriophage lambda. Illustrative kmers are shown above asterisked events in the squiggle. (**b)** an example read derived from the same 100 bp region as in (a) but incorporating shift, scale and drift, along with randomly skipped kmers. (**c**) shows this same read, but stretched in the time axis to map directly to the original reference. Comparing (**a**) with (**b**) reveals the requirement for an algorithm such as DTW for comparing a read to reference.

Supplementary Figure 2 - Illustrating the variability within reads and the consequences of z-score normalisation.



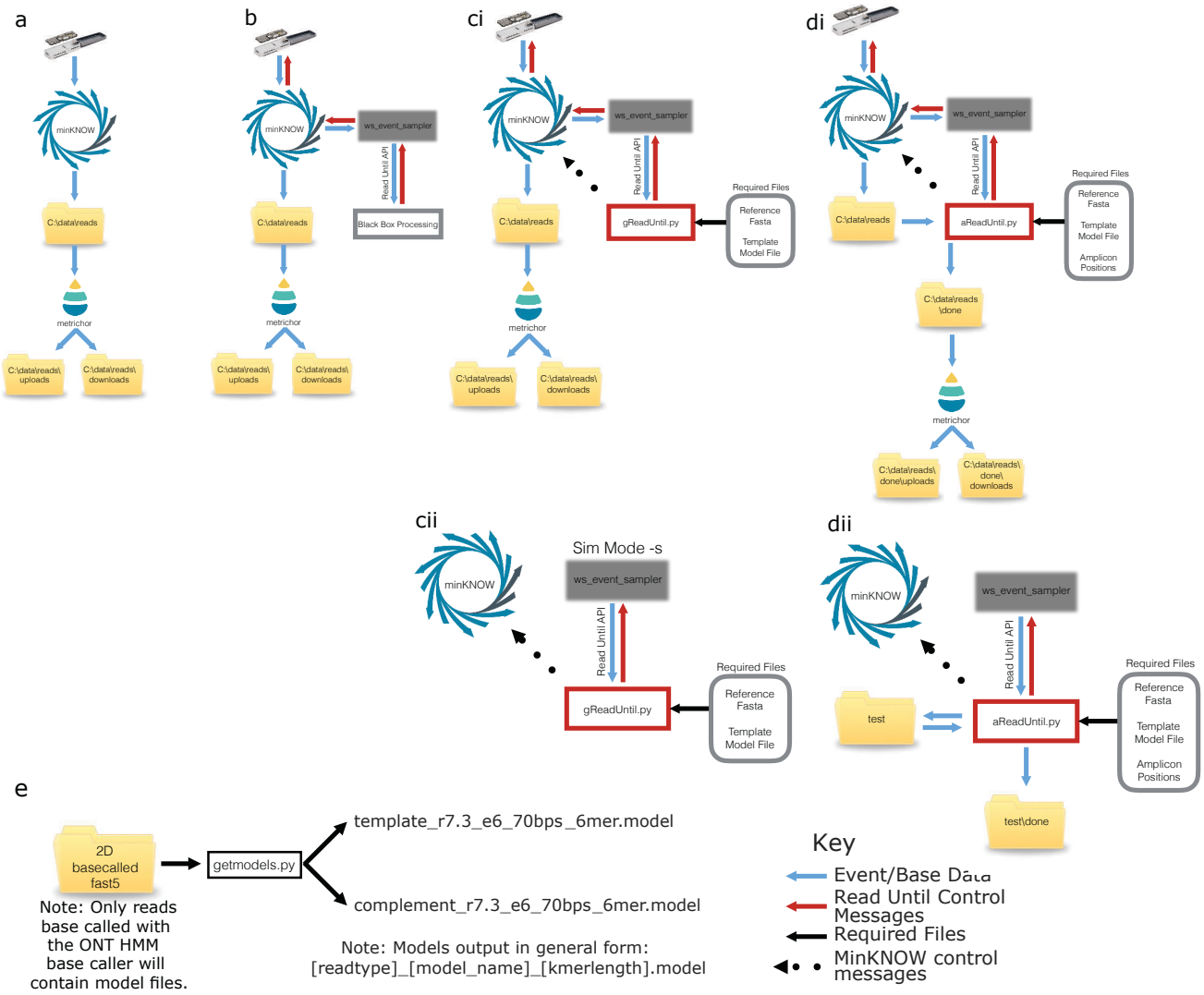
Top row illustrates a read as predicted from the model file and then this same read with shift, scale and drift simulated. Shift represents a global movement in the mean value of the squiggle, scale is a change in the variance and drift a change in the mean current over time. Bottom row shows this same read Z-score normalised. As can be seen, z-score normalisation corrects for shift and scale and reads map back to the control sample after normalisation whereas the drift parameter is not corrected. Note that the drift applied here is exaggerated for illustrative purposes.

Supplementary Figure 3 - Parameter scans to determine the efficiency of matching reads to a reference using Dynamic Time Warping.



The top row of panels (**a**), (**b**), (**c**) and (**d**) present results for un-normalised data, the bottom row (**e**), (**f**), (**g**) and (**h**) show results for Z-score normalised data. Percentage of correct alignments attained using windows sizes ranging from 4-2048 events are shown in panels (**a**), (**c**), (**e**), and (**g**). The total time taken to attain these correct alignments, for this range of windows sizes, are shown in panels (**b**), (**d**), (**f**) and (**h**). For the four panels on the left, (**a**), (**b**), (**e**) and (**f**), 100 synthetic reads taken from a synthetic genome (approximating a phage *Lambda* size genome by values drawn from a random uniform distribution) was tested. For the panels on the right 48 real minION reads from amplicon 3 of phage *Lambda*, were used. Panels (**a**) and (**e**) show, as expected, that applying Z-score normalisation to the synthetic data reduces the accuracy of mapping for extremely short reads. For real phage *Lambda* reads, panels (**c**) and (**g**), a dramatic improvement is observed when normalisations applied, panel (**g**). By a window size of 256 events, mapping is almost optimal. Placing 48 real event squiggles of length 256 took 15 seconds in total, panel (**h**), giving an average of 0.3 seconds per placement.

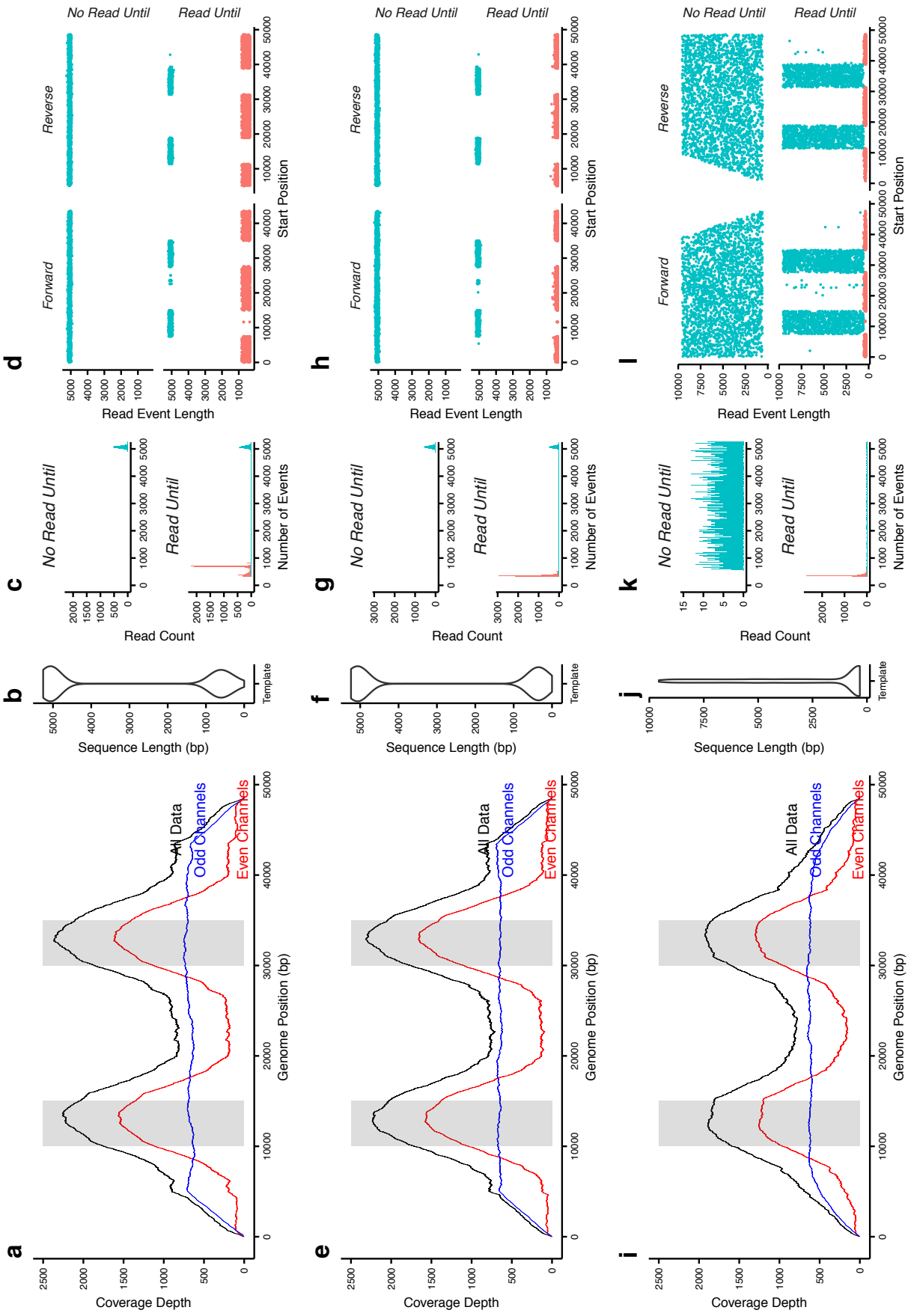
Supplementary Figure 4- 'Read Until' Work Flows.



Example workflows for read until. (**a**) Illustrating the components required for the standard ONT workflow. (**b**) 'Read Until' is implemented by running the ws_event_sampler utility. This provides raw read data and receives instructions to reject reads from specific channels. ws_event_sampler ensures that reads which are rejected have not already naturally completed. (**ci**) Illustrates the communication and data flow when running the gReadUntil.py script. Note that this script requires a model file and reference fasta to function. (**cii**) Illustrating the same process but using simulated rather than real data. (**di**) Illustrating the more complex workflow for Amplicon Balancing with 'Read Until' where the script is monitoring both reads streamed via the ws_event_sampler utility

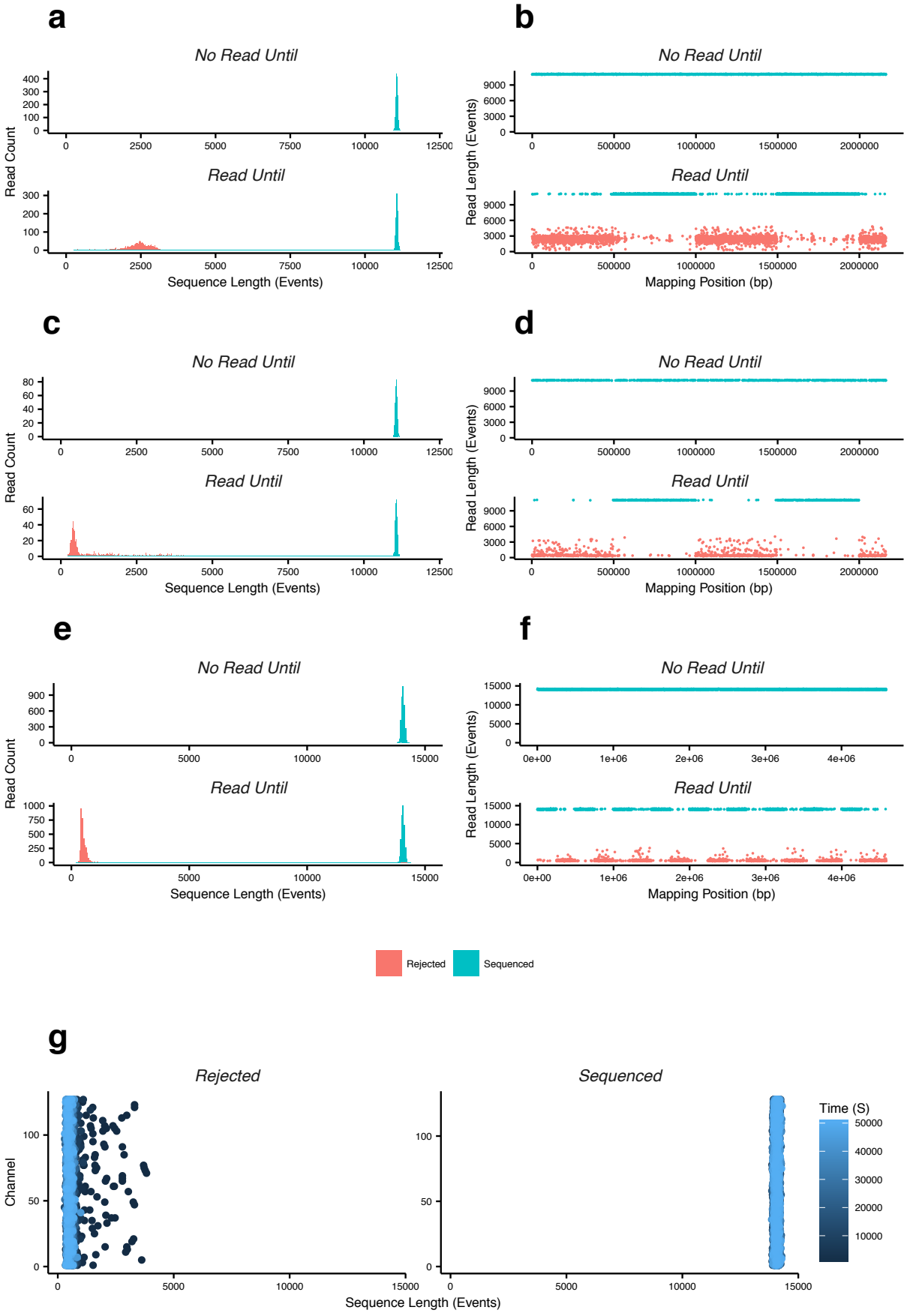
and reads as they are written to disk. (**d**ii) When run in simulation mode Amplicon Balancing also simulates the production of read data used to track the output of the scripts. (**e**) The naming conventions used by the get_models.py script.

Supplementary Figure 5- Simulations of 'Read Until' sequencing selecting regions from genomes.



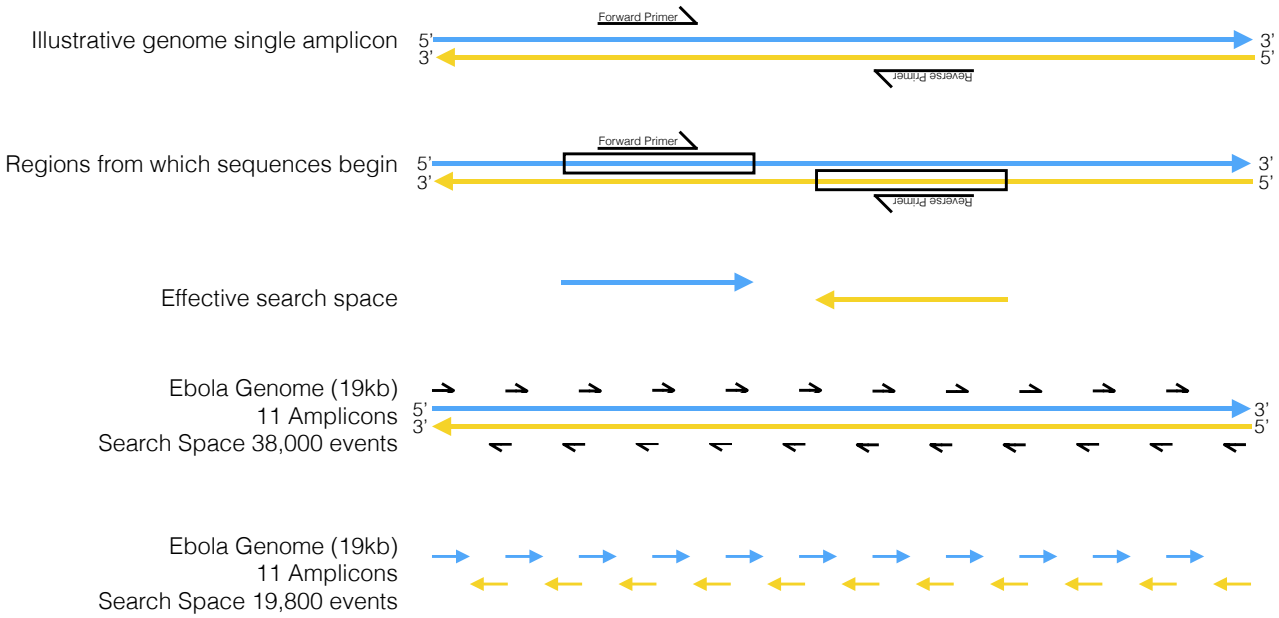
Further analysis of genomic selection with read until. ws_event_simulator.exe was used to simulate template only reads from the lambda genome. The simulator generates data at 30 bases per second and we determined the improvement on read until efficiency of both more timely rejection messages and the effects of variation in library length. Note that when simulating reads, the simulator is generating 'events' - i.e current measurements, not bases. (**a-d**) shows a library of average length 5kb selecting reads from 10-15 Kb and 30-35 Kb. With rejection messages sent every 15 seconds, reads are rejected after around 750 bases. The effect of enrichment is not significantly improved by sending rejection messages every 1s (compare **a**,**e**) although reads are rejected more rapidly around 500 bases in length. Introducing significant variation into library read lengths (**i-l**) has a greater effect on the observed enrichment illustrating again the importance of using the longest read lengths possible to maximise the benefits of 'Read Until'.

Supplementary Figure 6 - Simulating 'Read Until' on larger references.



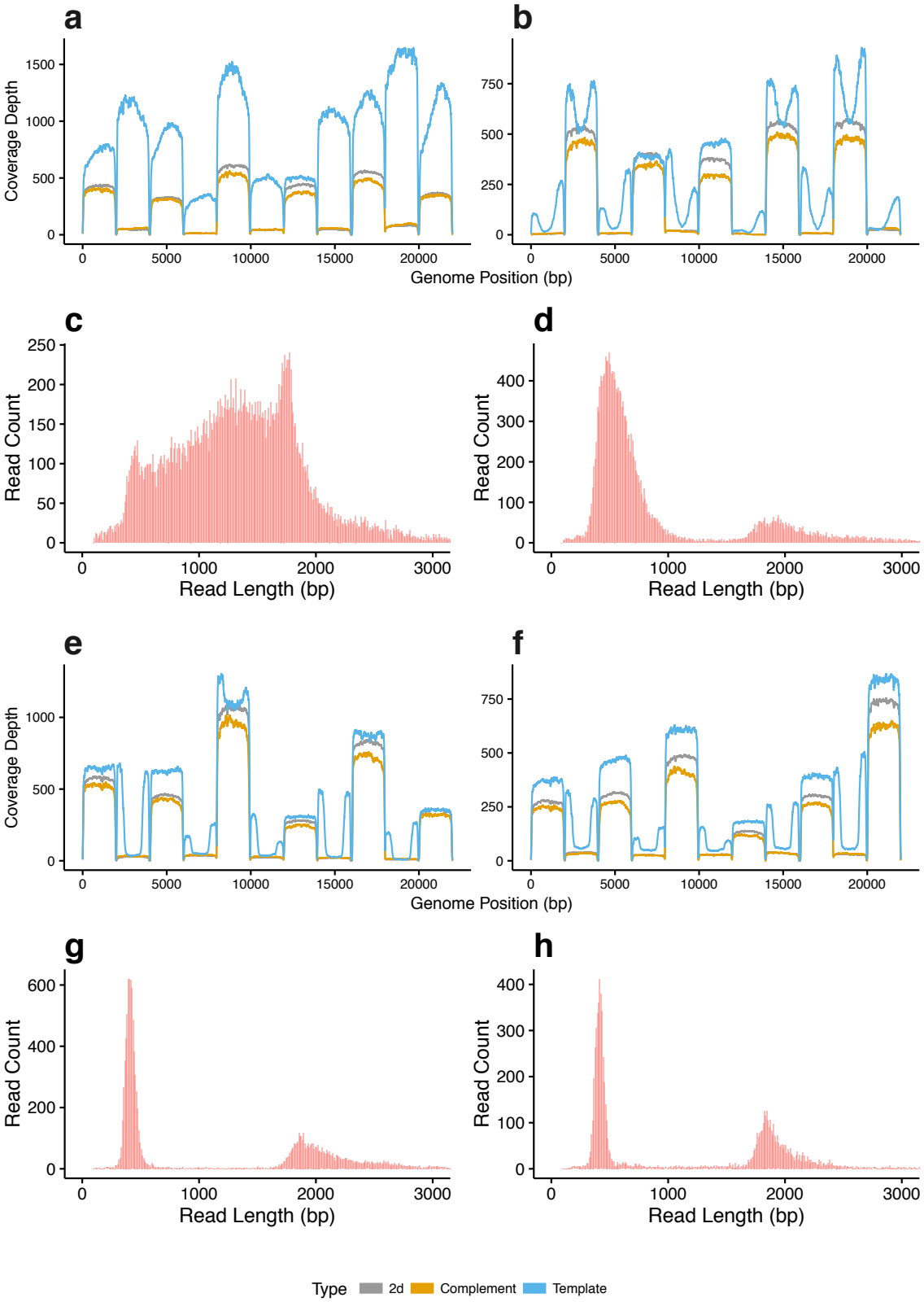
Simulations of selective sequencing. (**a,b**) Simulation of selective sequencing using 8 cores on a 2Mb genome running 'read until' on even numbered channels from 512 channels (i.e rejecting reads on 256 channels). Note that reads are rejected after around 2,500 events (**a**). (**c,d**) Reducing the number of channels to 256 increases the speed at which reads can be rejected to approximately 750 events. (**e,f**) Further reductions to 128 channels (i.e 64 channels actively selecting reads) gives very rapid read rejection across a 4.5 Mb genome. This clearly illustrates the requirement for significant parallel computation. (**g**) Individual reads lengths plotted against channel. Points are coloured according to the time the read began. Note that later reads are rejected in a more timely fashion than earlier reads, likely a consequence of the desynchronisation of channels in the simulation over time. We simulate long 1d reads (i.e template only) here of length 12-14 kb , corresponding to a 2d library of approximately 7kb average read length.

Supplementary Figure 7 - Reducing the search space for amplicon sequencing.



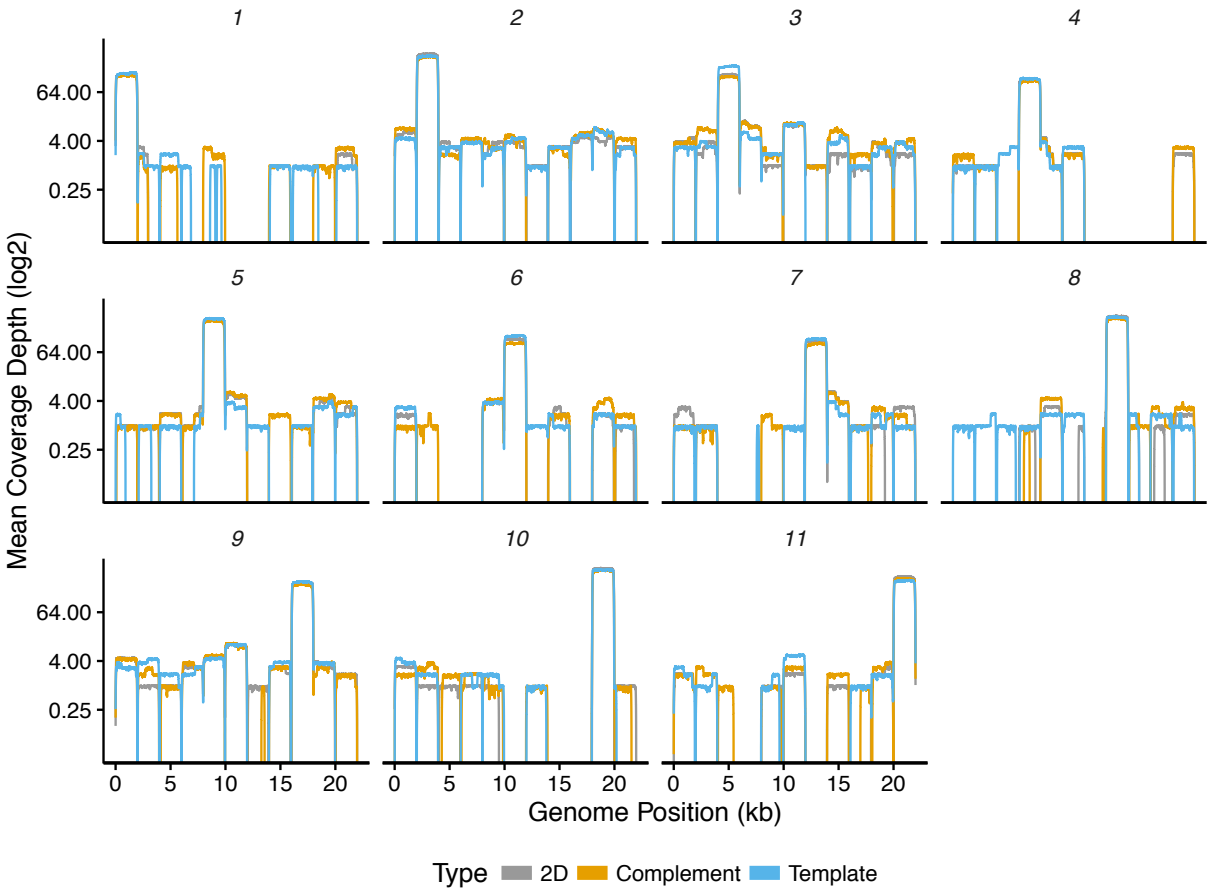
Illustrating the reduction in search space possible when using an amplicon based sequencing approach. Only regions overlapping the forward and reverse start sites of each amplicon need be retained in the final reference for matching.

Supplementary Figure 8 - Repeating odd numbered amplicon sequencing.



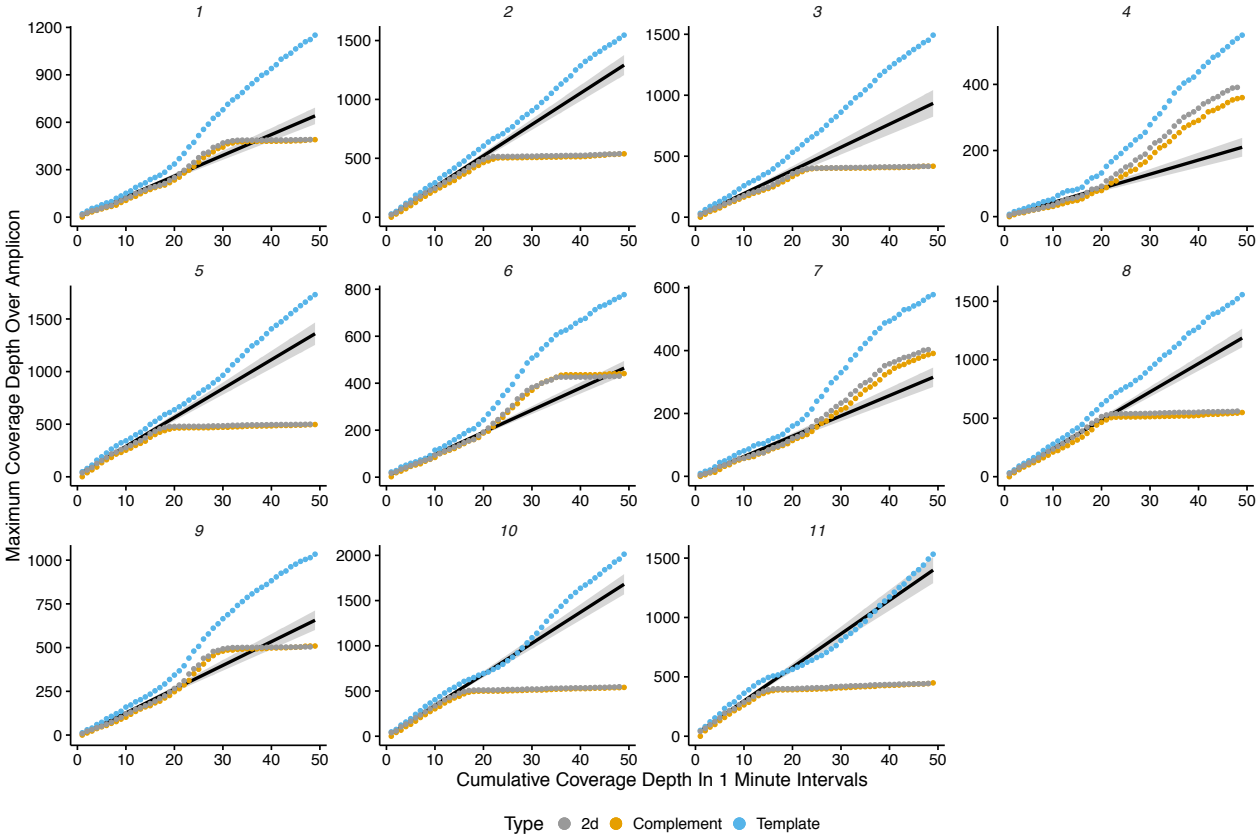
Repeating the sequencing of odd numbered amplicons. (**a**) and (**b**) Data as in Figure 2(b,c). (**c**) Histogram of template read lengths from A showing that reads were rejected before the end of the template sequence (at 2kb) but without a clear separation between sequenced and rejected reads. (**d**) Histogram of template read lengths from B showing a clear separation between rejected short reads and the naturally completing long reads. (**e,f**) Two subsequent repeats of the experiment shown in (**a**) with the characteristic template peaks on rejected amplicons. (**g,h**) Histogram of template read lengths from (**e**,**f**) showing rejection patterns matching (**d**).

Supplementary Figure 9 - Specificity of amplicon identification.



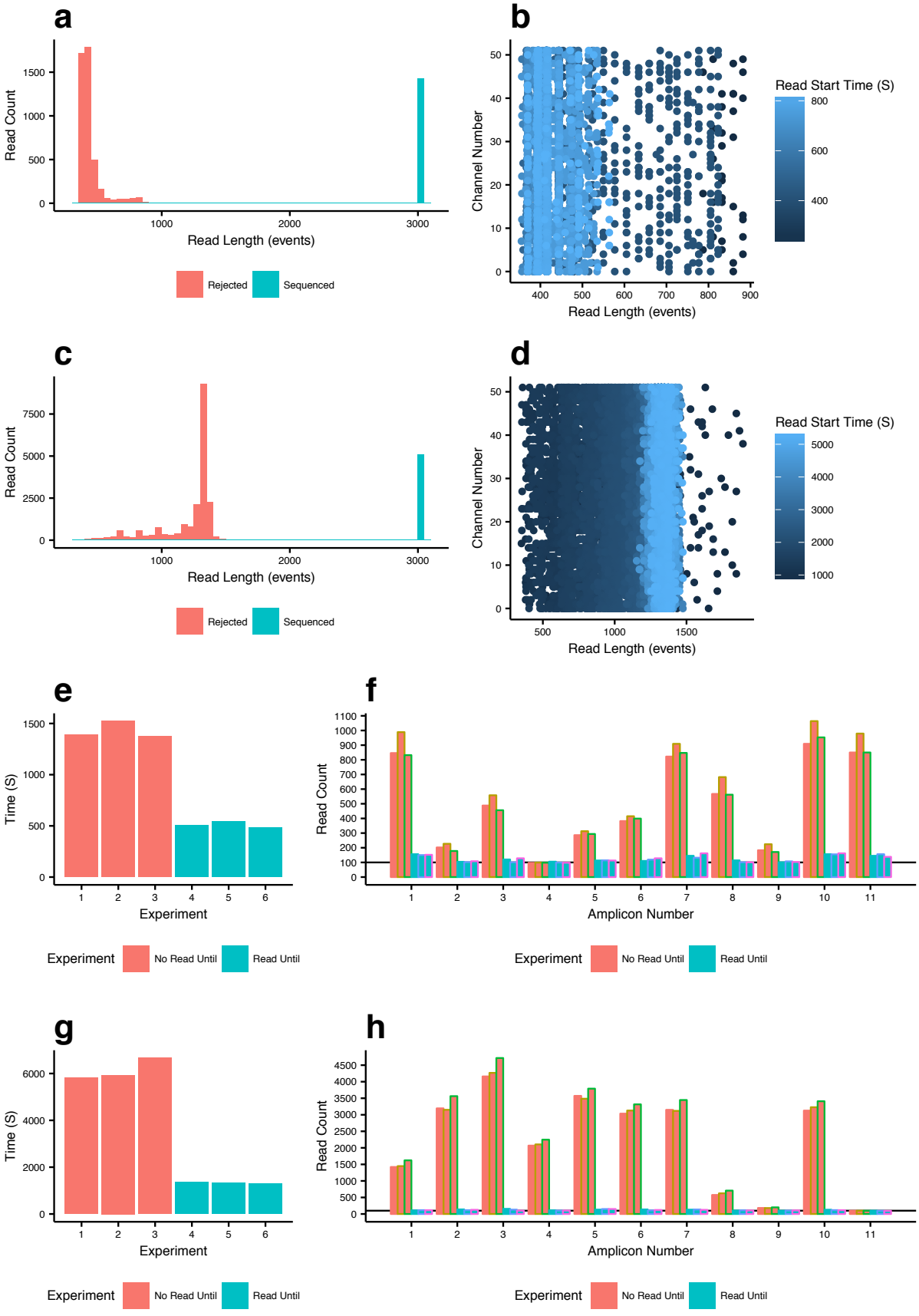
Post processing of individual reads to determine mapping specificity of reads to each amplicon. The first 250 events from each read (offset by 50 events) are mapped to the reference by DTW. Reads are classified according to their proximity to the known primer sites for each amplicon taking in to account the strand of the match. After this classification sequence data is mapped to the reference using BWA and coverage plots generated across the genome. Note the log2 scale.

Supplementary Figure 10 - Cumulative read counts for each amplicon when normalising with 'Read Until'.



Number of reads mapping to each amplicon over time when running 'Read Until' to normalise amplicon coverage. The black lines represent a loess fit to the first 15 minutes of 2D data collected for each amplicon and provides an estimate of 2D coverage depth in the absence of 'Read Until'. Rejected reads are still written to disk and appear as short template reads which can still be mapped to individual amplicons. Thus template read counts continue to accrue even after reads are being rejected by 'read until'.

Supplementary Figure 11 - Simulating amplicon sequencing with 'Read Until'.



Simulated sequencing on amplicons to explore the benefits of running 'read until'. (**a**) Simulated data for sequencing 11 synthetic 3Kb amplicons. The histogram reveals the majority of rejected reads to be less than 500 events in length. (**b**) Individual reads plotted by length against the channel from which they originate. Reads are rejected faster over time as a consequence of the channels of the simulated flowcell becoming less synchronised over time. (**c**) Repeating this experiment with 50 amplicons rather than 11 shows that the time taken to reject reads increases such that the rejected amplicon length is greater than 1000 events in length. (**d**) With more amplicons, the benefit of desynchronisation of the flow cell over time is lost. Now reads end up taking almost half their length to be rejected. (**e**) 3 simulations of normalisation of reads with 10 fold variance in the input library measuring the time taken to reach 100x coverage without and with read until. The time saving with 'read until' is readily visible. (**f**) Coverage plots for the amplicons as simulated in (**e**). (**g**) As (**e**), but now the input library has a 50 fold variation in amplicon concentration resulting in greater sequencing efficiency as a consequence of 'read until'. (**h**) Coverage plots for amplicons as simulated in (**g**).

Supplementary Tables

Table 1 - Read Data Sets - Available from http://www.ebi.ac.uk/ena/data/view/PRJEB12567

Experiment	Figure	Run Name	Date	Accession	File Name
Genome Selection 30 b/s 10-15 kb, 30-35 kb	2A	Ru7 Lambda	1 July 2015	ERS1051237	GenomeRU1.tar.gz
Genome Selection 70 b/s 10-15 kb, 35-40 kb	2B	RU9 Lambda	28 October 2015	ERS1051238	GenomeRU2.tar.gz
Genome Selection 70 b/s 15-25 kb	2C	Ru11 Lambda All Channels	29 October 2015	ERS1051239	GenomeRU3.tar.gz
Amplicon Sequencing No 'Read Until'	3A	RU24 Lambda PCR	5 October 2015	ERS1051240	AmpliconOddEvenControl.tar.gz
Amplicon Sequencing ODD amplicons only	3B	RU21 UDUD…	4 October 2015	ERS1051241	AmpliconOddReadUntil.tar.gz
Amplicon Sequencing EVEN amplicons only	3C	RU22 DUDU…	4 October 2015	ERS1051242	AmpliconEvenReadUntil.tar.gz
Amplicon Sequencing Control No 'Read Until'	4A Library Control	Default Sample	22 December 2015	ERS1051243	BalanceControl.tar.gz
Amplicon Sequencing Balancing	4A Normalised	Lambda 200xRU	22 December 2015	ERS1051244	Balanced.tar.gz

Table 2 - Example reference to squiggle conversion for six kmers.

Reference Position	Kmer	Mean Current
1	ATGCG	64.0277060402
2	TGCGC	74.1940612355
3	GCGCG	72.5600759653
4	CGCGT	76.3160881774
5	GCGTA	66.6657142728
6	CGTAG	61.7330951621

Table 3 - Primer Design for Lambda Amplicons.

Amplicon	Start	Sequence	End	Sequence
1	52	GCGTTTCCGTTCTTCTTCGT	1980	CAATCAGCCAGCTTTCCTCA
2	2065	GGAATGGTGCAGAAATGTCG	3965	TAATTCCGGGAAAGCTGCTC
3	4070	TTCTGTGCTGGCTGGAAGAG	5989	GAACGTGCCGGACTTGTAGA
4	6059	ACGGCAATCAGCATCGTTTA	7981	GCCATACATCCCCCTTTCAG
5	8012	CCTCAGCCGTATCAGCAAAA	9947	TTCATTCATGGACGGCATCT
6	10008	GCTGAAAACGTGGTGTACCG	11963	GGCAGCATGAGCACTGTCTT
7	12006	GCCGGAGCAAATGAGAAAAT	13941	GTTGGCCAGCATGATACGTC
8	14011	CAGCCAACGTCCGATATCAC	15945	CCAGCCACCGTTACGTTGTA
9	16076	ACCAGCTGCAGAACAAAACG	17987	CCGAGATGGGATTCGGTTAT
10	18022	GATAACGCCAGCAGACTGGA	19972	AGTTCAAGACGACGCAGCAC
11	20053	GTGCATCAGCTGCTCAGGTC	21979	ATGCCATTATGCAAGCCTCA

Table 4 - Amplicon Copy Number for Supplementary Fig. 11.

Amplicon	10 Fold Variance (Fig S10E,F)	50 Fold Variance (Fig S10G,H)
1	18	60
2	4	142
3	10	194
4	2	92
5	6	156
6	8	138
7	18	146
8	12	28
9	4	8
10	20	142
11	18	4