Stampy: A Statistical Algorithm for Sensitive and Fast Mapping of Illumina Sequence Reads

Supplementary information

Content

1. Algorithm

1.1 Building the hash table

Stampy uses a new open-addressing hashing algorithm to encode the genome (section 1.2). The hash table contains at most 2^{29} long-word (4 byte) entries, each entry consisting of a genomic coordinate and an additional 2 bits of auxiliary data to support fast searches at high load factors. The hash table occupies 2 Gb, and to fit mammalian-size genomes of about $3x10^9$ nt into the table, only every fifth position is entered. This scheme ensures that to store the position, 30 bits are sufficient, leaving room for 2 auxiliary bits required for the fast search algorithm (see section 1.2 for details). At each eligible position, a "hash" is constructed from the 15bp DNA word observed at that position in the reference genome. This is done by first encoding the 15 nucleotides into a 30-bit word, and then dividing out the reverse-complement symmetry by subjecting it to a transformation that maps words related by reverse-complementing to the same 29-bit word, and words that are not so related to distinct 29-bit words. To reduce clustering within the hash table, this word is further pseudo-randomized by multiplying modulo a large prime. As a side effect of this, the words 0x1fffffff and 0x1ffffffe become unused as hashes; these words therefore are available for use as flags (see below). For smaller genomes, a smaller hash table is used, and the resulting hash is further reduced modulo a power of 2 to limit the hash values to the size of the hash table. Note that for any size, different 15-mers (unrelated by reverse complementation) may hash to the same value, although this is extremely unlikely for the full-length hash table.

To avoid entering extremely long hash chains related to repetitive sequence, all positions are first scanned and hashes occurring more than 200 times noted. As candidate mapping positions from such chains are costly to consider, and are unlikely to result in a unique mapping position, their presence is flagged in the hash table by one of the unused words (0x1fffffff), and the actual locations are not entered into the hash table.

To improve search times, hash chains are entered roughly in order of decreasing length – this ensures that longer chains are less frequently interrupted by other elements than shorter ones, reducing search times and optimizing cache use for the longer chains that are relatively frequently accessed. We use a quadratic probing sequence that achieves a good balance of reducing clustering and making reasonably good use of caching[1]. In addition, for the longest chains we use linear probing to further optimize cache usage; because long chains are entered first, the issue of clustering is virtually absent, and cache usage for long chains is particularly poor under a quadratic probing scheme. The presence of the second unused word, 0x1ffffffe,in the primary position of the probe sequence signals the use of linear probing; when the initial position is already occupied the algorithm falls back to quadratic probing.

1.2 An improved hash table supporting fast searches

Because the scanning algorithm spends most of its time looking for 1-difference matches (Section 1.3), many hash lookups performed by the mapper will be unsuccessful. An unsuccessful search in a standard open addressing hash table is slow because the probe sequence is traversed until an empty slot is found, and the density of empty slots is low at high load factors. Stampy uses a hash table that addresses this by adding flag bits indicating whether more entries exist in the chain. A *chain* here is defined as the smallest prefix of a probe sequence that contains all elements inserted in a hash bucket. As this flag bit will only be present for chains that include at least one element, an additional sentinel bit is used to ensure that the search algorithm does not enter empty chains.

The order in which hash slots are probed is given by the probe sequence h(k,i). For standard hash tables this can be an arbitrary function of the object k and probe index i, as long as h(k, .): {0,…,m} → {0,…,m} is a permutation. However, the use of the sentinel bit now requires that the probe sequence be determined by the hash h'(k):=h(k,0), rather than by the object k. Of the standard probe sequences, linear and quadratic probing satisfy this requirement, but double hashing does not. Another choice, which minimizes clustering but has bad cache performance, is random hashing, which in its simplest form is h(k,i) = h'(k) + h''(i) mod n, where h''(i) is a permutation. To balance the opposing needs of low clustering and good cache behaviour, Stampy uses mainly quadratic probing, supplemented by linear probing for long chains, as explained in section 1.1.

The basic algorithms for inserting an object k, and searching for an object k, in a hash table T (assuming a single probe sequence, h, for simplicity) are:

```
Hash-insert(T,k):
        j ← h'(k), i ← 0, last ← -1
        while T[j].obj ≠ NIL:
                if h'(T[j].obj) = h'(k):  last ← j
                i ← i+1; j ← h(k,i)
        T[j].obj ← k
        T[ h'(k) ].present ← True
        if last ≠ -1: T[last].extend = True

Hash-search(T,k):
        if T[h'(k)].present = False: return NIL
        i ← 0; j ← h'(k)
        while True:
                if h'(T[j].obj) = h'(k):
                        if T[j].obj = k: return j
                        if T[j].extend = False: return NIL
                i ← i+1; j ← h(k,i)
```

1.3 Scanning the read

To find candidate locations for a single read, all overlapping 15mers in the read are considered. In addition, every 1-base mismatch ("1-neighbour") is considered. For reads longer than 34 bp, 1-neighbours are considered for a reduced fraction of initial 15-mers; half of them for reads up to 49 bp, to a third for reads of 50bp and above. Simulation experiments showed that this resulted in negligible loss of sensitivity, and a considerable reduction of computational time (data not shown). In addition, for 15mers that contain a single N character, all 4 possibilities for that position are considered, but no other 15mers are. Those 15mers that contain more than a single N character are not considered.

1.4 Searching the hash table

If the initial hash table entry corresponding to the 15mer is flagged by the high-count flag, this 15mer is not further considered. A repeat mask table with read locations that are not scanned for this reason is kept for later use. (More specifically, when the read 15mer is marked as repetitive, the corresponding location is marked with a Phred score 0; for 1-neighbours, it is marked by the Phred quality of the mutated base. When more than one 15mer, mutated or otherwise, is repetitive, the minimum Phred score is used. These values are used to calculate the mapping posterior, for details see section 1.8. A Phred score is a representation of a probability p as an integer, using the formula $-10\log_{10} p$. For non-repetitive 15mers, all positions in the genome that were entered in the hash table and match the 15mer or its reverse complement, together with their orientation, are retrieved.

1.5 Similarity filtering

To avoid excessive numbers of potential candidates, a neighbourhood similarity filtering step is included at this stage. A "fingerprint" is computed from three 4-nucleotide words close to but not overlapping the 15mer, and falling within the read. The fingerprint comprises the counts of A, C and G nucleotides within those 12 positions. The counts of the corresponding positions at the putative genomic location (in the implied orientation) are obtained, and the sum of absolute differences of A, C, G counts, and the implied absolute difference of T counts, is computed. This similarity statistic has the property that it increases by at most 2 with every single-nucleotide change, and every incremental 1bp insertion or deletion. Longer insertions or deletions have the opportunity to cause more drastic changes to this statistic, but this potential problem is mitigated by the fact that most indels are copy number changes of short tandem repeats or homopolymer runs, to which this statistic is relatively insensitive. Only locations for which this fingerprint match value does not exceed a set threshold are considered. This threshold is reduced by 2 for one-mismatch 15mers, so that similar approximate mismatch thresholds are used for the 15+12 nucleotide positions that are considered, independent of whether the original read 15mer or one of its 1-neighbours generated the candidate location.

1.6 SIMD alignment

Locations that pass the similarity filter are then considered for full-length alignment. At this stage the number of candidates often exceeds 1000, so that the use of a highly efficient implementation of the alignment algorithm is imperative. Stampy uses a Single instruction, multiple data (SIMD) implementation of banded affine-gap alignment. The implementation traverses the dynamic programming table diagonally, allowing an optimal exploitation of the parallelism provided by the x86 SIMD instructions. The dynamic programming table is held in registers rather than in memory, so that expensive cache misses are avoided. Since, in the first instance, the complete read is considered to derive from the reference, a Needleman-Wunsch global alignment rather than a Smith-Waterman local alignment is computed. This requires aligning the possibly low-quality end of the read containing relatively many mismatches; to handle such cases appropriately the algorithm computes a mapping quality score in Phred units that accounts for nucleotide quality scores, as well as gap opening and extension. To the probability represented by the base quality Phred scores, a term corresponding to the expected divergence to the reference (by default equivalent to a probability of 10^{-3} per nucleotide) is added, as for alignments no distinction ought to be made between read errors, polymorphisms and substitutions. The algorithm uses a 30bp diameter band, allowing insertions and deletions of up to 15bp to be fully considered at this stage. The limited number of bits available in SIMD registers to accumulate the score puts a limitation on the maximum read length that can be considered in a single alignment; in our implementation, reads of up to 4500 bp can be considered.

1.7 Single-end reads: realignment

Stampy reports the best-matching candidate mapping location, or in case of ties, a deterministic pseudo-random choice. When the process of generating candidate mapping locations described above does not result in any candidate, or when the candidate is judged to be no more similar than a best match for random sequence, no mapping location reported.

The best-matching candidate is re-aligned under the same model as used in the SIMD alignment, but now with a larger band of 60 bp diameter by default, enabling the correct identification of insertions or deletions ('indels') of up to 30 bp in length. When the most likely alignment comes within 10 bp of the edge of the dynamic programming band, the read is again re-aligned with double the bandwidth. In this way alignment in the presence of large indels is improved without a large increase in computational time in the general case.

The alignment algorithm at this stage allows flushing of indels to the leftmost or rightmost possible location; the desired behaviour can be selected by command-line options. When desired alignment posteriors per alignment column can also be computed, through the use of the Forward and Backward algorithms (REF Durbin et al). In certain cases this additional information allows potential indels

at the edge of reads to be identified. The computation of this information is relatively costly and is switched off by default.

1.8 Single-end reads: mapping posterior

Stampy computes the mapping posterior Phred score, or "mapping quality", in the standard Bayesian fashion by computing the posterior probability that the reported location is incorrect, in the first instance through the formula

$$1 - P(\text{read} \mid L_{opt}) \,/\, \Sigma\, P(\text{read} \mid L_i), \qquad (1)$$

where L_{opt} is the maximum likelihood mapping location, and the sum runs over all candidates considered. Because the alignment model considers read errors, single-nucleotide polymorphisms and substitutions, and short indels, this accurately estimates the probability that a read is mapped incorrectly when such errors are caused by (near-)repetitiveness in the genome, in combination with read errors and mutations. The result is approximate because not all possible mapping locations are considered in the sum, and consequently the resulting posterior does not include the possibility that the correct mapping locations was not considered among the candidates L_i. Broadly, this may happen for three reasons: (i) the read contains highly repetitive sequence and its 15mers were not entered into the genome hash table; (ii) the read is of low quality and/or is divergent from the reference by single-nucleotide and short indel mutations, so that every 15mer entered into the hash table has more than 1 nucleotide difference with the read, or (iii) the sequence is not represented in the reference.

Stampy's model accounts for all three possibilities. Sections 1.9 and 1.10 each describe the model for (i) and (ii), for short and longer reads respectively. Section 1.11 describes the model for (iii). Adding the probabilities for these three events to the naïve posterior (1) results in the final mapping quality, which is reported as a Phred score capped at 99. In addition, when (iii) is deemed likely, no mapping is reported for that read.

As an example of the situations handled by this model, consider a read from a highly repetitive genomic region. A read error within this read may cause a spurious hit elsewhere in the genome. However, because of the repetitiveness, this location is not deemed to be reliable, since the cumulative likelihood of the repetitive loci (which each require one read error) can overcome the single higher likelihood of the spurious hit. A similar situation occurs when the genome carries a mutation in an otherwise repetitive region; in this case, the read will be mapped correctly, but for the same reason as before, the map will not be deemed to be reliable.

1.9 Single-end reads: failing to find candidates – non-overlapping 15mers

As described above, two reasons for failing to identify the correct candidate locations are that the read is highly repetitive and its 15mers have not been entered into the hash; or that read errors or divergence cause all read 15mers to

be more than 1 mutation removed from the reference 15mer. This section describes an algorithm to estimate the probability of either event occurring.

Because the way the read is scanned changes with the length of the read as described in section 1.3, the model is dependent on the read length. For simplicity only case (1) that 1-neighbours are considered for all 15mers within the read, and case (2) that these are considered for only one-third of the 15mers, are distinguished; the intermediate case in which 1-neighbours of half of the 15mers are considered is approximated by the model for case (1).

First, consider case (2), that 1-neighbours for a third of read 15mers are considered. More precisely, the algorithm considers 1-neighbours for a consecutive block of 5 overlapping 15mers from the read; for the subsequent 10 only the read 15mers are considered, then again for a consecutive block of five 15mers the 1-neighbours are considered, and so on. The contribution to sensitivity of the 10 read 15mers are ignored in the model, as are the effects of indel mutations on sensitivity.

Suppose that the correct mapping location for the whole read is L. Because only 15mers from genomic loci that are multiples of 5 are entered into the genome hash table, only 15mers at read position k such that mod $(L+k) = 0$ have a possibility of matching. From the algorithm described above, and under the stated assumptions, it follows that this happens precisely once in a block of five overlapping 15mers from the read; the next possibility occurs exactly 15 bp further on. This means that the possible matches involve non-overlapping adjacent 15mers. If the offset k were known, the required probability is just the probability that all non-overlapping adjacent read 15mers at offset k have 2 or more mutations, or that the corresponding 15mer in the reference is repetitive and not included in the hash table. The final probability is computed by computing this compound probability for each offset k and taking the Bayesian average using a uniform prior of the possible offsets.

The compound probability is just the product of the probabilities defined above for all 15mers at the offset k. To calculate the relevant probability for one 15mer, write q_i for the error probability of base i in the 15mer (i = 1,…,15). The probability that no mutation (read error or substitution) occurs in the 15mer is

$$z = \Pi_{i=1}^{15} (1-q_i)$$

If we write

$$p_1 = \Sigma_{i=1}^{15} \; q_i \; / \; (1-q_i)$$

then the probability of two or more mutations w.r.t. the reference is $1 - z(1+p_1)$.

It remains to include the probability of missing a candidate because of repetitiveness. If the read 15mer is marked as repetitive in the hash table, the probability is taken to be 1 – this is an approximation, as the possibility of the true reference 15mer not being repetitive, but having been mutated into a highly repetitive one is ignored. If the read 15mer is not repetitive, but one of its 1-neighbours is, then the probability of the true reference 15mer being repetitive is estimated as the probability of the mutation having occurred. While this is not

correct in the strict Bayesian sense, as the prior of the read deriving from a repetitive sequence is not weighed explicitly against the alternative, simulation experiments show that this procedure is effective at reducing false positive rates and estimating well-calibrated priors (see main text, and results below). The data required for this procedure is collected at the scanning stage described in section 1.3.

1.10 Single-end reads: failing to find candidates – overlapping 15mers

This model deals with the case of relatively short reads, in which case 1-neighbours of each overlapping 15mer are considered. The analysis of this situation is complicated because of the dependency structure introduced by the overlapping 15mers, and the non-uniform distribution of read errors described by the quality scores.

We approximate the situation by considering that in a contiguous subsection of the read, the probability of 0, 1, 2, 3 or 4 mutations is known; that 5 mutations or more do not occur; and that conditional on a certain number of mutations occurring, these are distributed uniformly along the subsection. A simulation was used to estimate the probability that, conditional on the length of the subsection and the number of mutations occurring, every overlapping 15mer fully within the read subsection and at the (unspecified) offset *k* overlaps with 2 mutations or more, as only in this case would the correct candidate *not* be considered.

In particular, independently of *k*, for 3 mutations, at least one 15mer is guaranteed to overlap at most 1 mutation in a read of length 34. This can be seen by considering that in a length-34 read, two non-overlapping 15-mers fall wholly within the read for any *k*; and for both these 15-mers to *not* yield the correct candidate, at least 2 mutations are required in each. Considering the algorithm in section 1.3 this means that Stampy is guaranteed to consider the correct mapping location for any read with 3 or fewer mutations in the first 34 bp. For 4 mutations, this is true for reads of 52 bp and longer.

More generally, the probability of not identifying the correct candidate is estimated as follows. First, the probability of precisely *m* mutations among *L* positions, each independently having probability q_i of mutating, is

$$\sum_{1 \le i_1 < \cdots < i_m < L} \left(q_{i_1} \cdots q_{i_m} \times \prod_{i \notin \{i_1, \ldots, i_m\}} (1 - q_i) \right) = \prod_{i=1}^{L} (1 - q_i) \sum_{1 \le i_1 < \cdots < i_m < L} \prod_{k=1}^{m} \frac{q_{i_k}}{1 - q_{i_k}} \qquad (2)$$

The last sum over m-fold products is inefficient to calculate as written, particularly as *m* and *L* increase. However, this sum may be recognized as the elementary *m*-th degree symmetric polynomial e_m in the variables $q_i/(1-q_i)$, $i=1,\ldots,L$. These can be computed from the power sums,

$$p_i = \sum_{k=1}^{15} \left(\frac{q_k}{1 - q_k} \right)^i$$

using Newton's identities:

$$e_1 = p_1$$
$$e_2 = \tfrac{1}{2} p_1^2 - \tfrac{1}{2} p_2$$
$$e_3 = \tfrac{1}{6} p_1^3 - \tfrac{1}{2} p_1 p_2 + \tfrac{1}{3} p_3$$
$$e_4 = \tfrac{1}{24} p_1^4 - \tfrac{1}{4} p_1^2 p_2 + \tfrac{1}{8} p_2^2 + \tfrac{1}{3} p_1 p_3 - \tfrac{1}{4} p_4$$
$$e_5 = \tfrac{1}{120} p_1^5 - \tfrac{1}{12} p_1^3 p_2 + \tfrac{1}{8} p_1 p_2^2 + \tfrac{1}{6} p_1^2 p_3 - \tfrac{1}{6} p_2 p_3 - \tfrac{1}{40} p_1 p_4 + \tfrac{1}{5} p_5$$

Since the p_i can be calculated efficiently, so can $z\, e_m$ in (2), which represents the probability of precisely m mutations occurring among the L positions.

For longer reads, it often occurs that sections are of poor quality. This happens predominantly at the far end of the sequence, but low-quality bases can occur anywhere in the read because of localized issues such as air bubbles in the flow cells, focusing or optical alignment problems. In order not to obtain an overly pessimistic estimate of the probability of missing candidates, particularly for long reads, Stampy considers every 15- to 52-bp subsection of the read, calculates the probabilities of 0 to 5 mutations following the algorithm above, and estimates the probability of missing the true candidate, conditional on finding it in the subsection under consideration. Since the simulation assumes that fewer than 5 mutations occur, no subsections are considered where the probability of 5 or more mutations exceeds 0.1. This cutoff was found to provide accurate estimates in practice (data not shown). The final reported probability is the minimum of these probabilities over all subsections that are considered.

The effect of repetitive 15mers is modelled by adding to q_i the repeat mask probability obtained by the algorithm in section 1.3, similar to the procedure described in section 1.8.

1.11 Single-end reads: random matches

The models for (i) and (ii) are Bayesian models, in the sense that a full description of the process that generated the reads is attempted. However, when a read is not represented in the reference, for instance because it derived from a contaminant, an generative modeling approach fails. Algorithmically, a unique best match may exist, but it will be biologically meaningless.

To identify such cases, a hypothesis-testing stage forms the last part of the mapping quality model. This stage assesses the likelihood that the read is a random sequence, and that any sequence similarity of the candidate mapping results from finding the best match for this random sequence. This question is complicated by the alignment procedure, which considerably increases the chances of finding fairly good matches at random.

Suppose that the best alignment of a read of l nucleotides, n of which align to the reference with m mismatches, includes i insertions and d deletions. The quintuplet (l,n,m,i,d) forms a summary of the complexity of the alignment. To estimate the size of the search space, we need to compute the number of alignments of similar complexity.

Consider the quintuplet to be fixed, and denote the set of all alignments characterized by the quintuplet, at a particular locus and strand x, by A_x. Any alignment $a \in A_x$ implies a particular sequence s_a of n nucleotides at the aligning positions of the read. Denote the set of all (locus, strand) pairs for the reference by G. Suppose that the set $S = \{ s_a \mid a \in A_x , x \in G \}$ may be considered to be drawn uniformly from the set of all sequences of length n. Then, the likelihood of randomly obtaining a match that is as good as the best alignment that was in fact found, is

$$1-\left(1-\frac{1}{4^n}\right)^{|S|} \approx 1-e^{-|S|\times 4^{-n}} \approx |S|\times 4^{-n},$$

since this alignment implies n matches, and $|S|$ loci and alignments have potentially been considered. The second approximation holds only if $|S|4^{-n} \ll 1$. If $|S|4^{-n}$ is of order 1 or greater, the likelihood of finding a random match of comparable quality is nearly 1, under the stated assumptions. As mentioned above, in this case no mapping is reported.

The formula above assumes that every alignment is equally probable; more specifically, that for a random read, mismatches; insertion starts and ends; and deletions all occur uniformly along the read. We in addition assume that deletions can have any size uniformly from 0 to a set maximum, here 30. To calculate |S|, note that there are m mismatches to be distributed among n aligning locations; $2i+d$ positions are to be chosen within the read of length l at which to start or end an alignment, or place a deletion; and d deletion lengths to be chosen; in addition the mismatching nucleotide at m position is to be chosen. In approximation, the total number of possibilities thus becomes,

$$2g \times 3^m \binom{n}{m} \binom{l}{2i+d} 30^d << 4^n$$

where g is the size of the reference, and the factor 2 accounts for the choice of strand. This formula is approximate as it does not consider the fact that deletions cannot occur between insertion starts and ends; in practice such situations arise most often for incorrectly mapped reads, in which case an overestimation is conservative.

Note that when insertions run off either end of the sequence, one less insertion start or end position needs to be chosen, and $2i+d$ should be replaced by $2i+d-1$ in the formula above. A similar consideration does not apply to deletions as they cannot occur at the read boundaries.

Nucleotides are not uniformly chosen from the 4 possibilities, but rather often are biased somewhat towards A/T or G/C. To account for this, in the formulas above we use b (and b-1) instead of 4 (and 3), where b is somewhat less than 4 to account for any G/C bias, using the formula $b = \exp(-f \log \tfrac{1}{2}f - (1-f) \log \tfrac{1}{2}(1-f))$, where f is the G+C content. For instance, for f=0.35, a value b=3.8 is used.

1.12 Paired-end reads: paired-end candidates

The paired-end pathway follows the single-end one up and including the point of calculating the single-end mapping quality, for each of the reads independently.

If no candidates were obtained for both reads, the paired-end read is reported as unmapped.

When the best locations for the single reads are close together on the genome (defined as an implied insert size within 4 standard deviations of the mean), the resulting paired-end mapping positions are considered. If in addition both single reads map sufficiently uniquely (each with a posterior probability of less than 1% of having been mapped incorrectly due to near-repetitiveness), and in addition the estimated probability of not having found the correct candidate is sufficiently low for both (again less than 1%), the paired-end mapping location is reported.

When any of these conditions are not met, Stampy creates a shortlist of pairs of mapping locations. From the candidate locations for each member of the read pair, the locations that together constitute 99.9% of the single-read posterior mapping probability are extracted, up to a maximum of 20 locations, and subject to a minimum of 3. For each of these locations, the mate is aligned against the reference around the location implied by the library insert size distribution, plus or minus 4 standard deviations. The alignment model used is as for single-end reads, but contains an additional term modeling the likelihood of the implied insert size, which helps to disambiguate the mapping position of locally repetitive reads. In addition to this list of novel pairs, the pairing of the top-scoring single-end mapping locations is added. When these are not close together, a score corresponding to the prior probability of the physical insert overlapping the breakpoint of structural variation, which could give rise to such configurations, is added. This prior probability is user-specified, and is 3×10^{-6} (Phred 55) by default. (Here, 'close' is defined as the distance at which the likelihood of the insert size under the insert size model, approximated by a Gaussian distribution, becomes less than the prior probability for a structural variant.)

The posterior mapping quality is calculated as the product of the single-end mapping qualities in case of the top-scoring single-end hits being selected as the pair, or the single-end posterior of the anchoring read in other cases.

2. Performance on simulated data

2.1 Setup of the simulation experiment

To assess the performance of Stampy under different conditions, and to compare this with the performance of other read mapper programmes, a simulation pipeline was developed.

In short, reads were generated by taking sequence data from randomly chosen positions in the human genome. Read errors were introduced according to the empirical read error distribution, and additional changes were added to this to simulate SNPs and indels.

The empirical read error distribution was obtained from two sets of paired-end Illumina data from the 1000 Genomes Pilot project 1. To ensure that mapping in the presence of read errors was sufficiently well assessed, these were chosen to have error rates at the lower 25% quartile within their read length category (accessions SRR003994, sample NA19239, 36 bp; and SRR005802, sample NA18520, 72 bp).

The read error distribution was calculated by first mapping the actual reads to the human reference using Stampy, and tabulating base match and mismatch counts stratified by read, cycle (position within the read), reported quality score, and the number of preceding mismatches. In this way, the reported quality scores were recalibrated, taking account of position within the read and dependencies between errors within a read. The use of Stampy at this stage does not cause bias in favour of Stampy in the simulations. No known SNPs or indels were removed from the actual reads prior to tabulating mismatches; in this way, the estimated error rates will slightly exceed the true error rates, but this is not expected to affect the conclusions of the study.

The recalibration tables were used to subsequently introduce errors in sequence data taken from known paired positions, chosen uniformly on the genome and at a distance of 250 ± 25 bp apart to simulate a distribution of insert sizes.

Several sets of reads were generated, each with a different mutation profile in addition to read errors. These categories comprise: -

(i) single-nucleotide changes drawn from a Poisson distribution with rate 10^{-3}, to simulate human SNPs ["SNP0.001"; one category];

(ii) one, two or three single-nucleotide changes added to each read ["snp1", "snp2", "snp3"; three categories]

(iii) single-nucleotide changes added drawn from a Poisson distribution with rate 0, 1%,… up to 15%, simulating substitutions in a divergent species [sixteen categories];

(iv) a single short insertion or deletion added in one read or read pair, with length drawn uniformly from [1..30], and placed uniformly into the read, and using random sequence in the case of insertions ["indel30", one category].

(v) a single large (20 kb) deletion affecting one read or read pair. The location of the deletion was chosen uniformly in one of the reads. This category simulates reads overlapping large deletions ["largedeletion", one category]

(vi) a single large insertion affecting one read or read pair. This was implemented by choosing an insertion location uniformly in one of the reads; the sequence to either the left or the right of this point was then replaced by random sequence. This category is a simple simulation of reads overlapping structural variation breakpoints ["largeinsertion", one category].

For each category, 4 data sets were created: either with 36 bp and 72 bp reads, and either single reads or read pairs. Each data set contained 1 million reads or read pairs.

The simulation code also generates a read label which encodes the true position and orientation, the true insert size, and any indel mutation (position and size). Read errors and single-nucleotide mutations were not recorded in the read label.

A pipeline was written for the process of simulating reads, mapping these with each of Stampy (using BWA as pre-mapper), BWA, Maq, Eland and Novoalign; converting the output to SAM format where required; adding back reads, if any, that were not reported; and collecting statistics including sensitivity (recall rate) for various thresholds of reported mapping quality values, mapping quality calibration, and mapping and indel recall sensitivities conditional on indels of a particular size being present in the read.

Reads (read pairs) were considered to be mapped correctly if their position (both their positions) coincided with the true position, as encoded in the read label. We consider the main aim of a mapper to infer the read's correct genomic locus, and consider the ability to correctly identify indels through inferring the corresponding gap in the read alignment to be desirable but not essential. This led us to consider reads to be mapped correctly if any of its nucleotides were aligned to the correct reference nucleotide, as encoded in the true alignment provided in the label. In practice this means that, when a read contains a single indel, either the sequence to the left or to the right of the indel needs to overlap the correct reference sequence in order for the read to be considered correctly mapped.

2.2 Simulating read data

The code to simulate reads is part of the Stampy program. The following table lists the command line options that were used to generate the reads. Read lengths and base qualities were not generated, but taken from existing reads.

Category	Options
snp0.001	--substitutionrate=0.001
snp1	--simulate-numsubstitutions=1
snp2	--simulate-numsubstitutions=2
snp3	--simulate-numsubstitutions=3
divN, N=0…0.15	--substitutionrate=N
indel30	--simulate-minindellen=-30 --simulate-maxindellen=30
largedeletion	--simulate-minindellen=-20000 --simulate-maxindellen=-20000
largeinsertion	--simulate-minindellen=20000 --simulate-maxindellen=20000

In addition, the options `--insertsize=250 --insertsd=25` were used throughout to set the insert size distribution.

Prior to generating the simulated reads, the empirical error distribution of a set of actual Illumina reads were obtained using the recalibration algorithm (option `-R`), which generates `.recaldata` files. Reads were then simulated using the `–S` option. This also generates labels that include the position and (where appropriate) mutation information.

After mapping, the mapped reads in `.SAM` format were supplemented with input reads that were filtered out by the mapper. Various statistics were then collected by a script that is included in Stampy, accessible through the `–P` (parse) option.

2.3 Assessing mapping quality calibration

Mapper programmes differ considerably in the spectrum of reported mapping Q values. This makes it difficult to meaningfully plot the mapping quality calibration, since a naïve binning scheme may put very different numbers of data points in any particular bin across the different mapping programmes, causing large sampling errors in the estimated Q scores. Instead we followed a scheme where total and incorrect mapping counts were sorted by decreasing reported mapping Q score, and accumulated until the accumulated theoretical expected incorrect mapping count exceeded 3.0. The ratio of this expected count over the total mapping count, was then plotted against the ratio of the observed count over the same total, both on a logarithmic (Phred) scale. After each point the accumulated counts were reset and the process proceeded down the list. In this way, every point plotted has roughly equal (maximum) sampling errors associated to them, independently of the spectrum of reported Q scores.

2.4 Choice of read mappers for comparison

We used the following criteria to select read mappers for comparison to Stampy. First, the mapper should be able to map Illumina reads of 36-72 bp, both single- and paired-end. Second, it should produce SAM or BAM files, or output for which conversion tools to either format were readily available. Third, it should produce mapping quality information, which is essential for SNP and indel callers. Fourth, the read mapper should be able to run on the hardware at our disposal (a dual quad-core Linux cluster, with 16 and 32 Gb of memory per node). Fifth, the mapper should be reasonably popular choice. Finally, we did not strive for completeness, and used the first five criteria as necessary but not sufficient conditions; rather we tried to cover at least the most popular, the fastest, and the most sensitive algorithms.

Based on these criteria we chose Maq, BWA, Eland, and Novoalign.

The Bowtie program, is both fast and popular, however it does not produce mapping quality scores. The BFAST program was considered, but we decided not to include it because of its high memory usage. The Mosaik suite of programs could have been included, but we felt that the mappers chosen already covered the spectrum of sensitivity and speed sufficiently.

The following software versions were used:

Stampy v1.0

BWA 0.5.6 (r1303)

Maq v0.7.1

ELAND_standalone.pl v 1.3 2009/11/13 (part of CASAVA 1.6.0a11)

Novoalign V2.05.16

3. Performance on real data

3.1 Re-mapping 1000 Genomes Data

To assess the performance on real data we re-mapped human genomic data from the 1000 Genomes Project, Pilot 1, using Stampy and BWA. To limit the use of computational resources, we chose two individuals from the Pilot 1 (low coverage) data. As this data has been generated while the sequencing platforms have undergone considerable development, we made sure not to choose among individuals with only short read length (36 bp) and/or only single-end reads. From those that included reads of 50bp or over, comprised at least 50% paired-end Illumina data, we chose the individual NA18510, which has a coverage closest to median (4.62 X). For this individual 16 51bp paired-end lanes of Illumina data were available. For the other sample we chose NA18520, which was also used to generate the empirical error distribution for 72bp reads (see Section 2). For this individual 7 lanes of 76 bp paired-end Illumina data were available. Table **S1** below lists the input data that was considered.

To assess performance, we mapped each read of a mate pair independently and calculated the proportion of pairs that ended up in consistent locations. Mapping locations were considered to be consistent if they were within 10kb of each other; while this is generous for genomic DNA, it does capture a fraction of transcriptome-derived read pairs that map across introns; the expected false positive rate from using a generous window size of 10kb is negligible (a naïve estimate is 1.5e-6).

This performance indicator estimates the frequency at which single-end reads are both, independently, mapped correctly. This does not test the part of the algorithm that uses mate pair information to rescue reads that otherwise could not have been mapped. It also does not give a direct estimate of the proportion of single-end reads that can be mapped correctly, since both reads of a mate pair are required to be correctly mapped.

File name
NA18510/sequence_read/SRR005804_1.filt.fastq
NA18510/sequence_read/SRR005804_2.filt.fastq
NA18510/sequence_read/SRR005805_2.filt.fastq
NA18510/sequence_read/SRR005806_1.filt.fastq
NA18510/sequence_read/SRR005806_2.filt.fastq
NA18510/sequence_read/SRR005807_1.filt.fastq
NA18510/sequence_read/SRR005807_2.filt.fastq
NA18510/sequence_read/SRR005808_1.filt.fastq
NA18510/sequence_read/SRR005808_2.filt.fastq
NA18510/sequence_read/SRR005809_1.filt.fastq
NA18510/sequence_read/SRR005809_2.filt.fastq
NA18510/sequence_read/SRR005810_1.filt.fastq
NA18510/sequence_read/SRR005810_2.filt.fastq
NA18510/sequence_read/SRR005810.filt.fastq
NA18510/sequence_read/SRR011049_1.filt.fastq
NA18510/sequence_read/SRR011049_2.filt.fastq
NA18510/sequence_read/SRR011050_1.filt.fastq
NA18510/sequence_read/SRR011050_2.filt.fastq
NA18510/sequence_read/SRR011051_1.filt.fastq
NA18510/sequence_read/SRR011051_2.filt.fastq
NA18510/sequence_read/SRR011052_1.filt.fastq
NA18510/sequence_read/SRR011052_2.filt.fastq
NA18510/sequence_read/SRR011060_1.filt.fastq
NA18510/sequence_read/SRR011060_2.filt.fastq
NA18510/sequence_read/SRR011062_1.filt.fastq
NA18510/sequence_read/SRR011062_2.filt.fastq
NA18510/sequence_read/SRR011063_1.filt.fastq
NA18510/sequence_read/SRR011063_2.filt.fastq
NA18510/sequence_read/SRR011064_1.filt.fastq
NA18510/sequence_read/SRR011064_2.filt.fastq
NA18520/sequence_read/SRR005797_1.filt.fastq
NA18520/sequence_read/SRR005797_2.filt.fastq
NA18520/sequence_read/SRR005798_1.filt.fastq
NA18520/sequence_read/SRR005798_2.filt.fastq
NA18520/sequence_read/SRR005799_1.filt.fastq
NA18520/sequence_read/SRR005799_2.filt.fastq
NA18520/sequence_read/SRR005800_1.filt.fastq
NA18520/sequence_read/SRR005800_2.filt.fastq
NA18520/sequence_read/SRR005801_1.filt.fastq
NA18520/sequence_read/SRR005801_2.filt.fastq
NA18520/sequence_read/SRR005802_1.filt.fastq
NA18520/sequence_read/SRR005802_2.filt.fastq
NA18520/sequence_read/SRR005803_1.filt.fastq
NA18520/sequence_read/SRR005803_2.filt.fastq

Table S1. Data files from 1000 Genomes Pilot 1, for the comparative mapping experiment.

3.2 Mapping divergent mouse data

To assess the ability of the different aligners to map reads to a divergent reference, we used *Mus Spretus* genomic reads that were kindly made available to us by Dr. David Adams (Sanger Institute, Hinxton). The speciation time of this

subspecies is estimated at 1.1 Mya[2], and the divergence to the C57BL/6J mouse reference sequence is 2% [3].

3.3 Mapping reads from mRNA transcript data

We used a single lane of Illumina paired-end reads from an mRNA-seq experiment conducted by Dr Ioannis Ragoussis (Wellcome Trust Centre for Human Genetics, Oxford). The mRNA data for this sample was obtained from MCF-7 cancer cell lines, and prepared following the standard Illumina mRNA-seq protocol. This data set is available upon request.

3.4 Allele biases

To assess the presence of allele biases due to systematic mapping problems at polymorphic indel sites, we first identified high-confidence heterozygous sites from indel calls made by the 1000 Genomes Project, on the father-mother-child trio NA12891-NA12892-NA12878. Such sites were defined as those that were homozygous for the reference and the alternative allele in the parents. Only sites where the genotype quality of all three individuals were above Phred score 40 were considered; and the indel call quality itself was required to be above Phred score 100. In total 10553 sites were considered.

We then used three BAM files of the child data (NA12878) that were produced by the project (mapped by Maq and BWA respectively), and a BAM file produced from the same data using Stampy.

To calculate the number of reads supporting either the reference or the indel allele, we considered reads in which the called indel would fall within 10 bp from either end. Reads that contained no indel within 10bp of the called position were classed as reference reads; otherwise, the read was classed as supporting the indel. We then calculated the ratio (reads supporting the alternative) / (all reads) for all sites, and computed the cumulative distribution. This was done separately for insertions and deletions.

4. Simulation results

4.1. Recall rates

	Single-end 36bp	Paired-end 36bp	Single-end 72bp	Paired-end 72bp
BWA	3.2	4.9	3.2	3.4
Stampy	13.7	26.8	10.7	14.6
Stampy (standalone)	30.3	50.8	22.6	31.2
Eland	40.2	41.6	20.7	19.4
MAQ	55.8	51.8	29.1	24.8
Novoalign	46.9	79.6	81.1	61.6

Table S2. Runtime in CPU hours per gigabase of sequence data, for the five mappers considered, on the "snp0.001" dataset. "Stampy (standalone)" refers to Stampy without BWA as a first stage.

Data set	Programme	36bp,SE	36bp,PE	72bp,SE	72bp,PE
snp0.001	**stampy**	82.3	94.5	87	96.2
	novoalign	86.5	96.5	90.7	97.4
	bwa	69.9	82.8	74	87.1
	eland	71.3	70.2	71.8	76.5
	maq	85	92.7	83.9	91.9
snp1	**stampy**	77.1	92.4	85.7	95.7
	novoalign	80.5	95.7	89.3	97
	bwa	53.3	67.4	69.7	84
	eland	59.8	56.3	69.2	74.4
	maq	78.8	87.4	80.3	89.8
snp2	**stampy**	68.4	88.3	84.1	95.1
	novoalign	61.5	93.5	87.5	96.4
	bwa	27.3	37.9	62.6	78.4
	eland	38.5	34.5	67.3	71.8
	maq	62.9	72.6	72.5	84.7
snp3	**stampy**	54.3	79.7	82.3	94.2
	novoalign	18.5	83.7	85.4	95.5
	bwa	0.9	0.9	50.2	67.8
	eland	6	5.7	65	68.7
	maq	16.2	22.6	49.2	65
largedeletion	**stampy**	39.6	70.2	71.1	88
	novoalign	15.6	68.1	35.6	69.1
	bwa	8.1	35.7	16.6	45.4
	eland	12.5	41.4	33	67.9
	maq	16.3	39	14.6	33.5
largeinsertion	**stampy**	19.7	59.9	39	72.7
	novoalign	7.6	61.2	17.6	59
	bwa	3.9	47.3	8.3	64.9
	eland	6.1	36.4	16.4	55.4
	maq	8	48.8	7.2	50.4

Table S3. Recall rates for various data sets (first column) and five aligners (second column), separated by read length and single-end or paired. Rates are given as the proportion of reads mapped to the correct position or positions, as a percentage of all input reads.

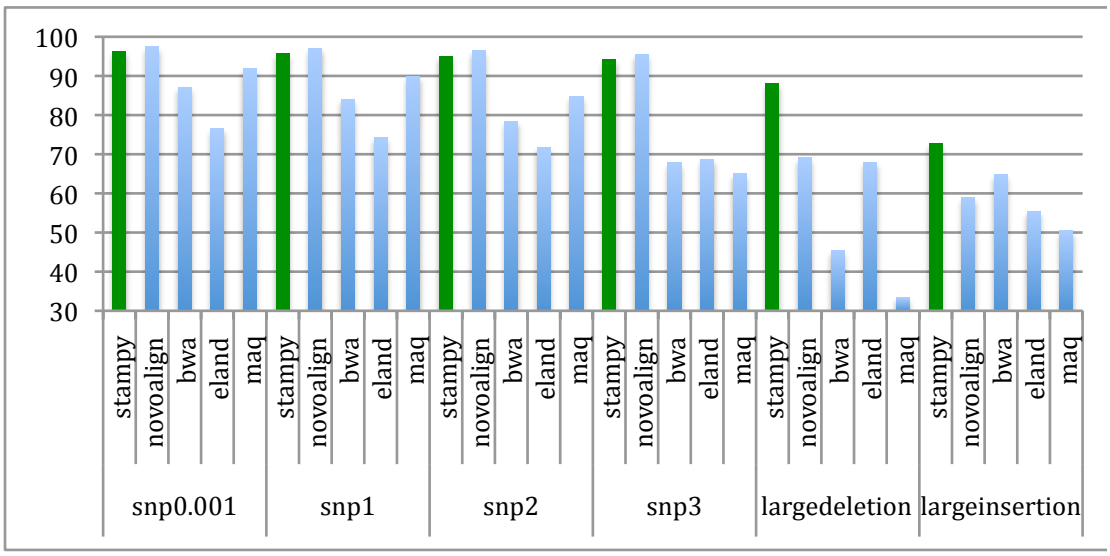

Figure S1. Recall rates for several data sets and five aligners, for 72 bp paired-end reads (final column of Table S2).

4.2 ROC curves

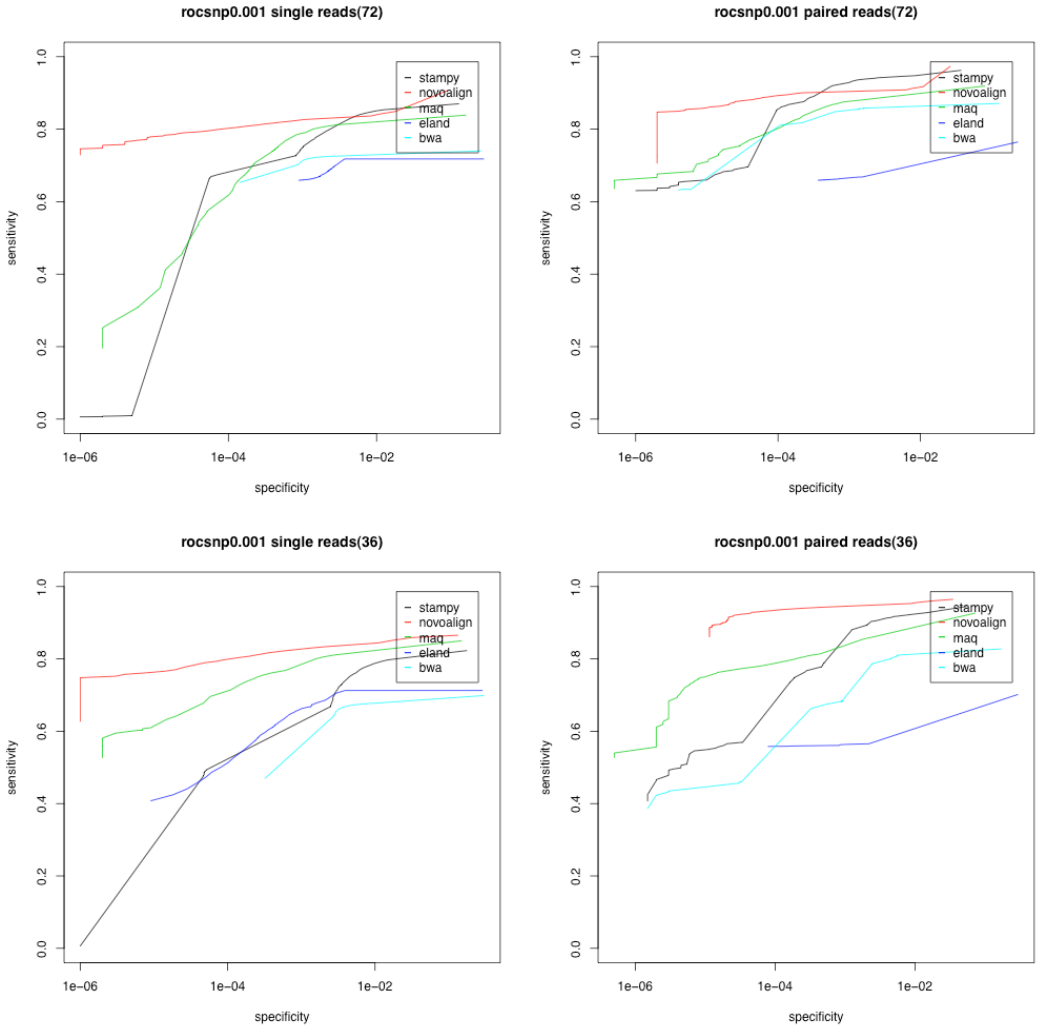

Figure S2. ROC curves for mapping reads with 0.1% SNPs, and no indels.

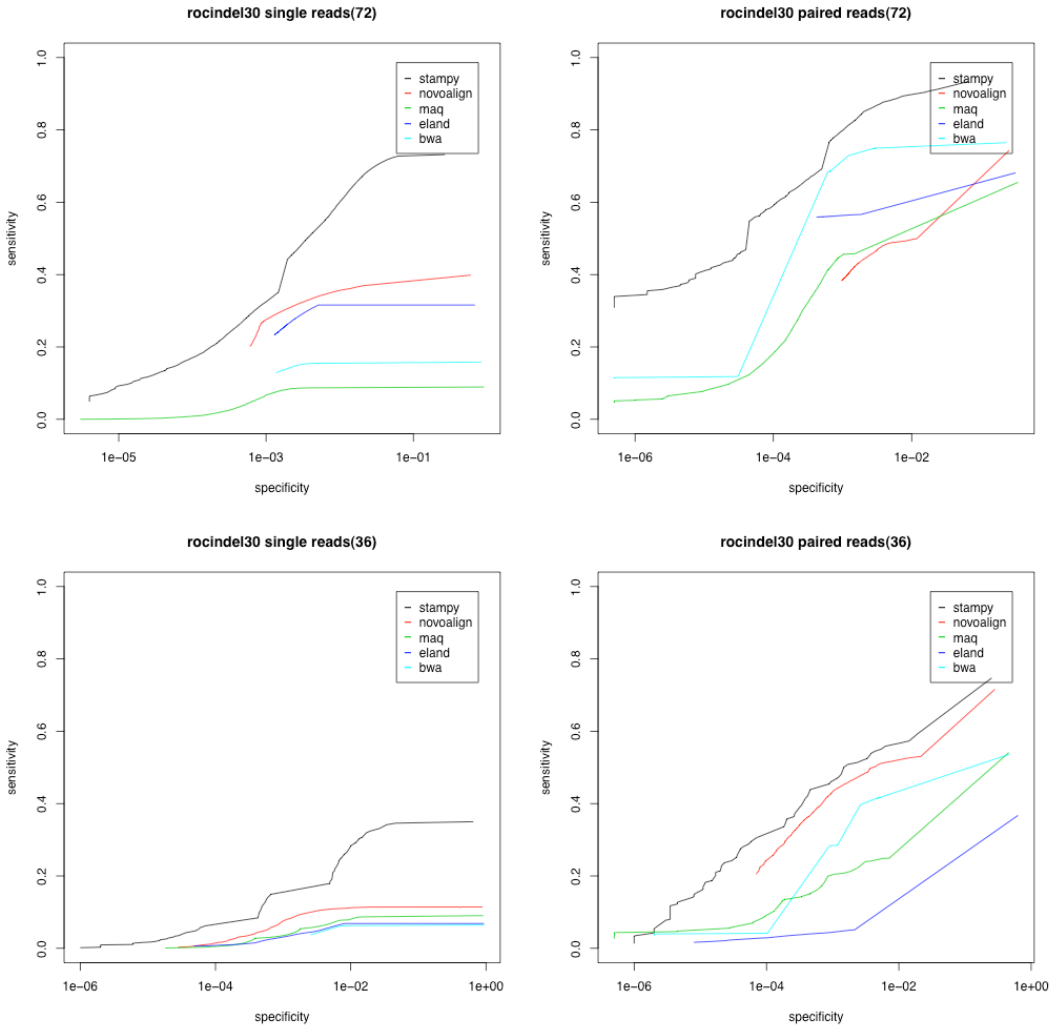

Figure S3. ROC curves for mapping reads with a single indel per read or read pair.

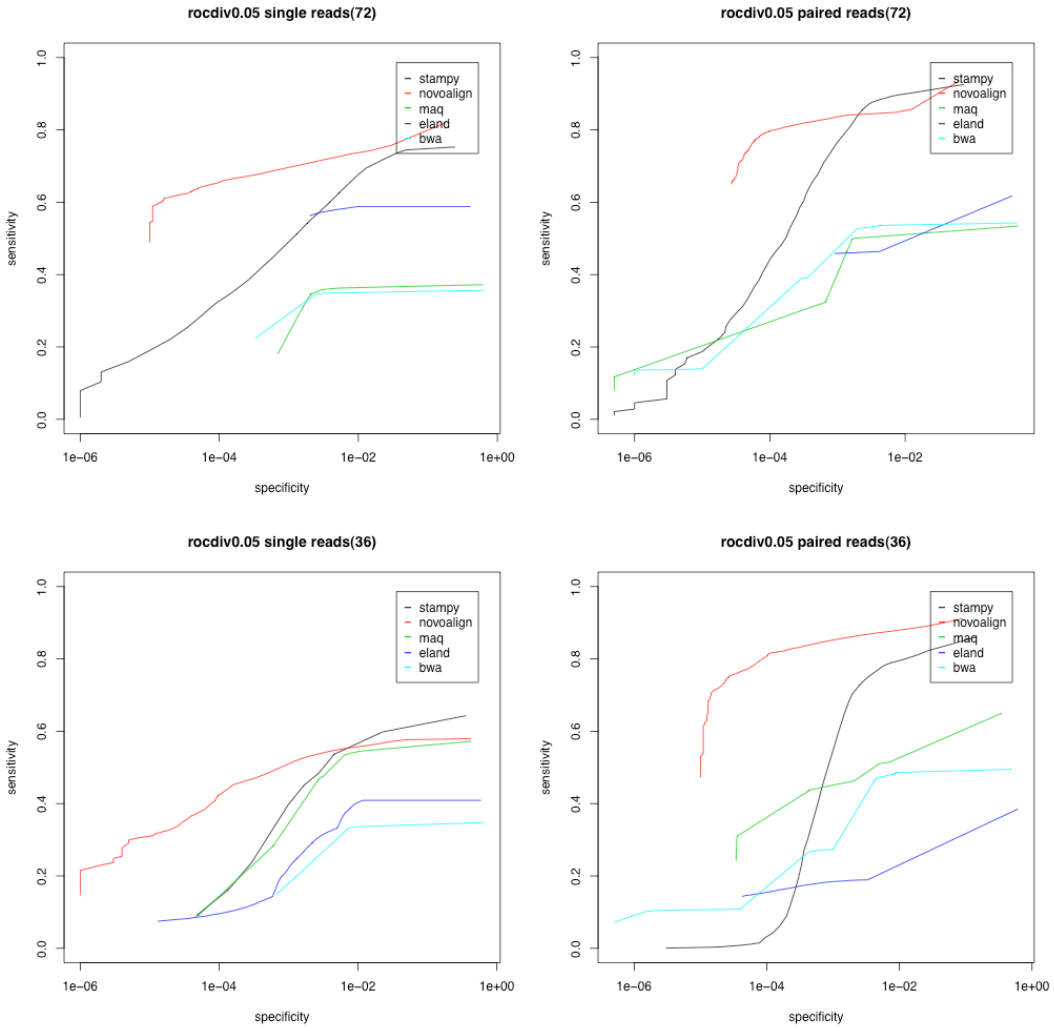

Figure S4. ROC curves for mapping reads at 5% divergence from the reference.

5. Mapping quality calibration

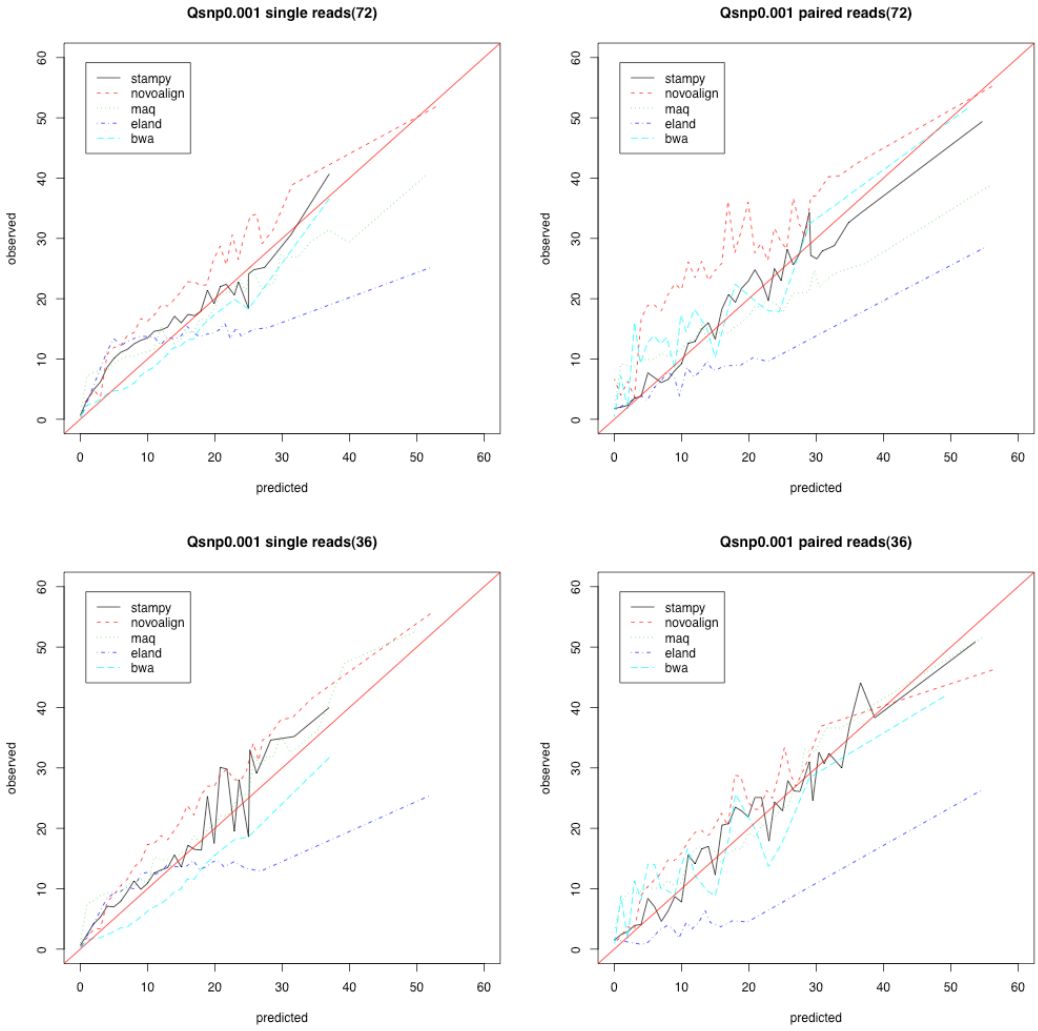

Figure S5. Mapping quality calibration, for reads with 0.1% SNPs.

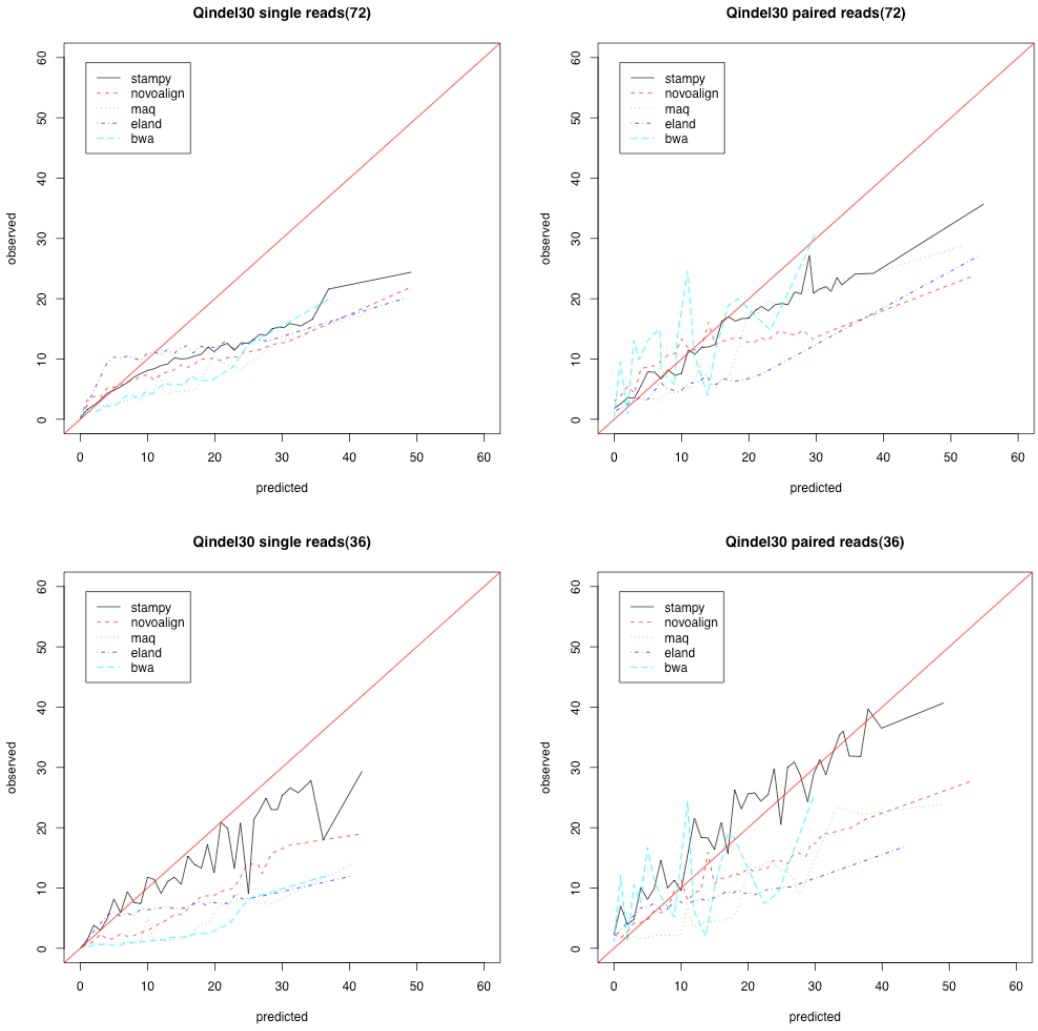

Figure S6. Mapping quality calibration, for reads with one indel per read or read pair.

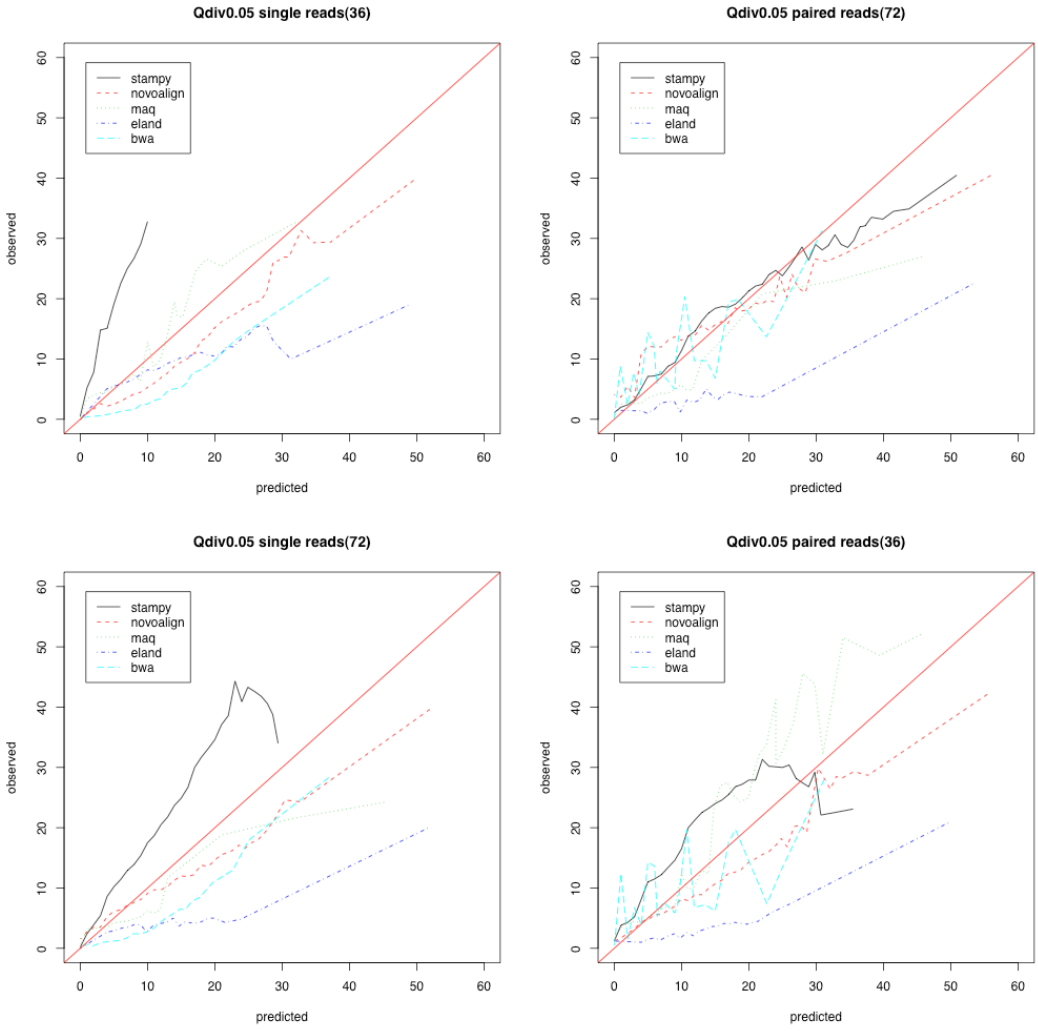

Figure S7. Mapping quality calibration, for reads at 5% divergence from the reference.

4.3. Mapping sensitivity in the presence of indels

These graphs show the proportion of reads that get mapped to the correct location, whether or not the correct indel is indentified. Reads are deemed to be mapped correctly if the inferred read alignment shares an alignment column with the true alignment.

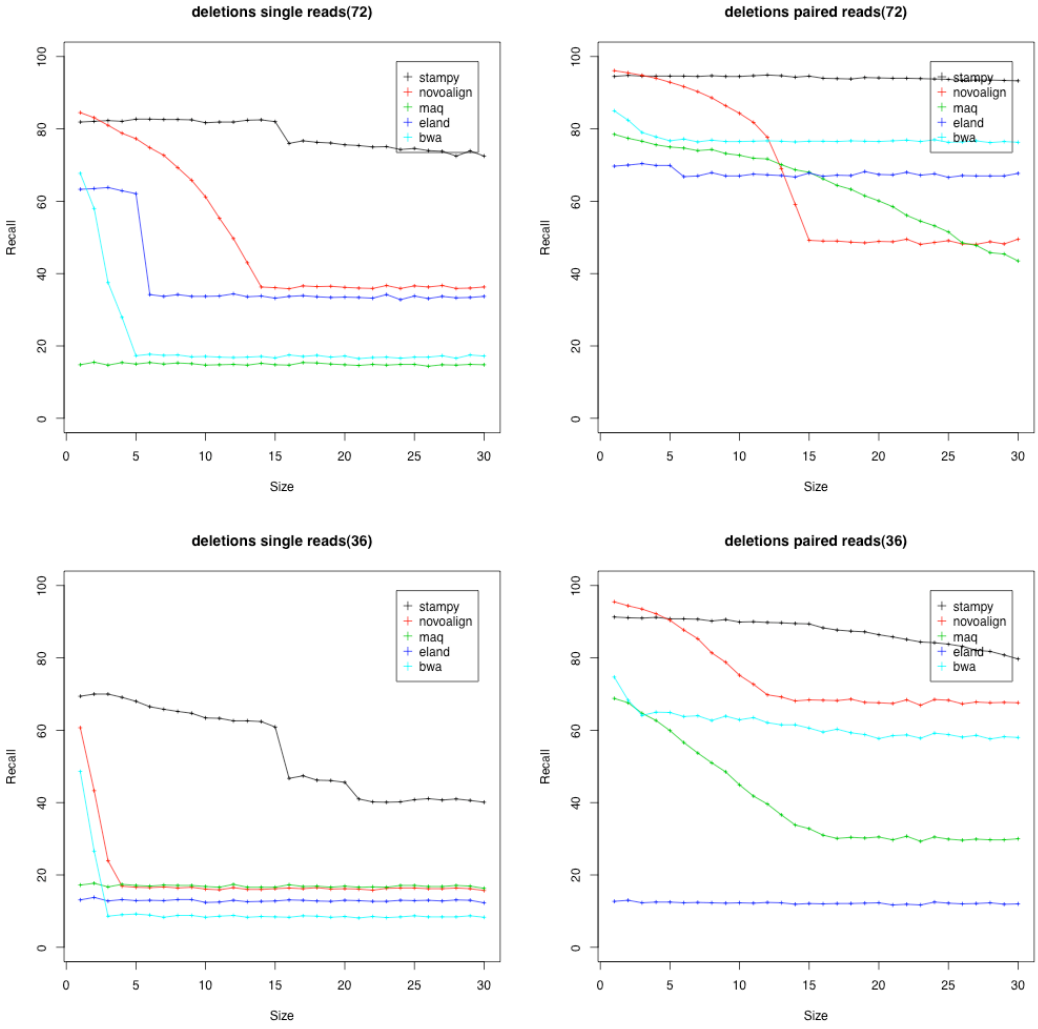

Figure S8. Mapping recall rates by deletion length, for reads with one deletion per read or read pair.

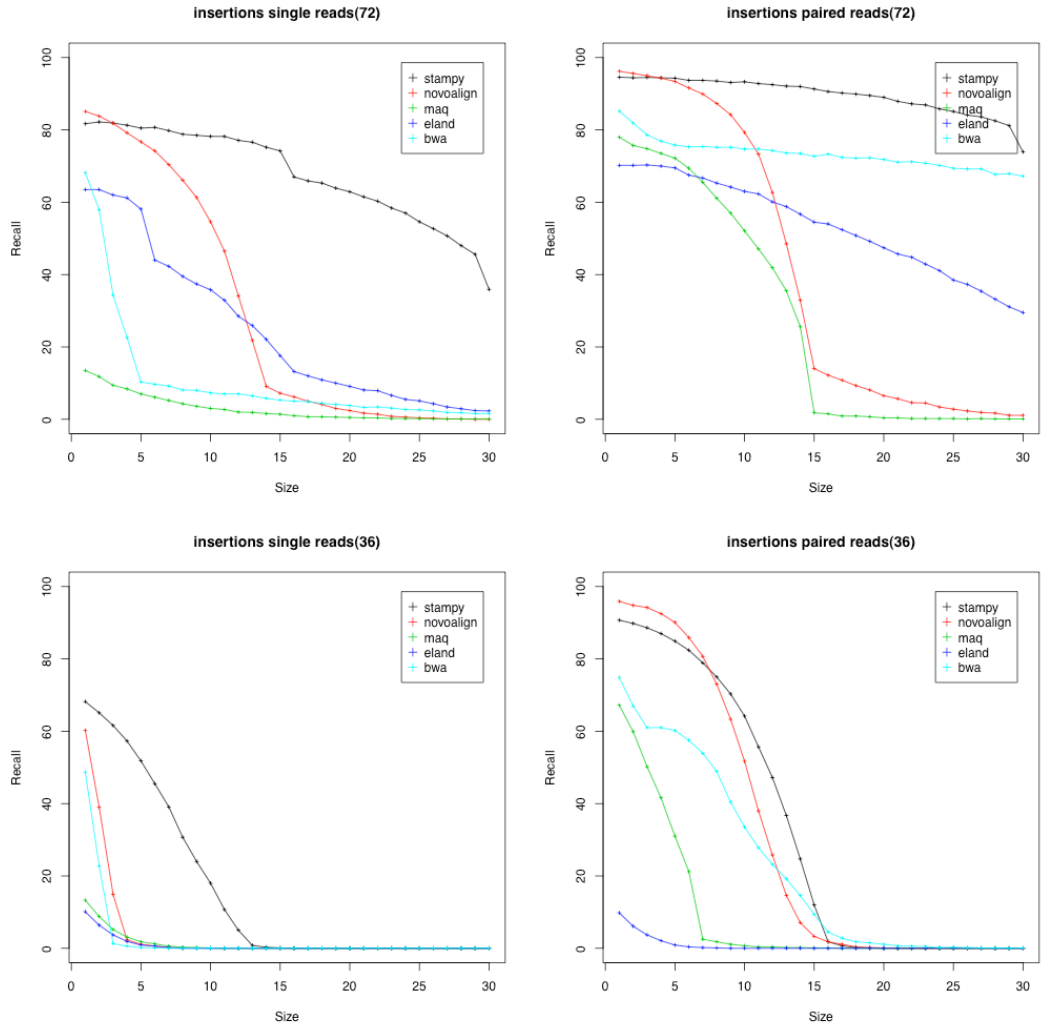

Figure S9. Mapping recall rates by insertion length, for reads with one insertion per read or read pair.

4.4. Indel indentification rate

These graphs show the proportion of reads that get mapped to the correct location, and align with an indel of the correct length. To account for ambiguous placement of indels, it is not required that the indel is located at the "true" position

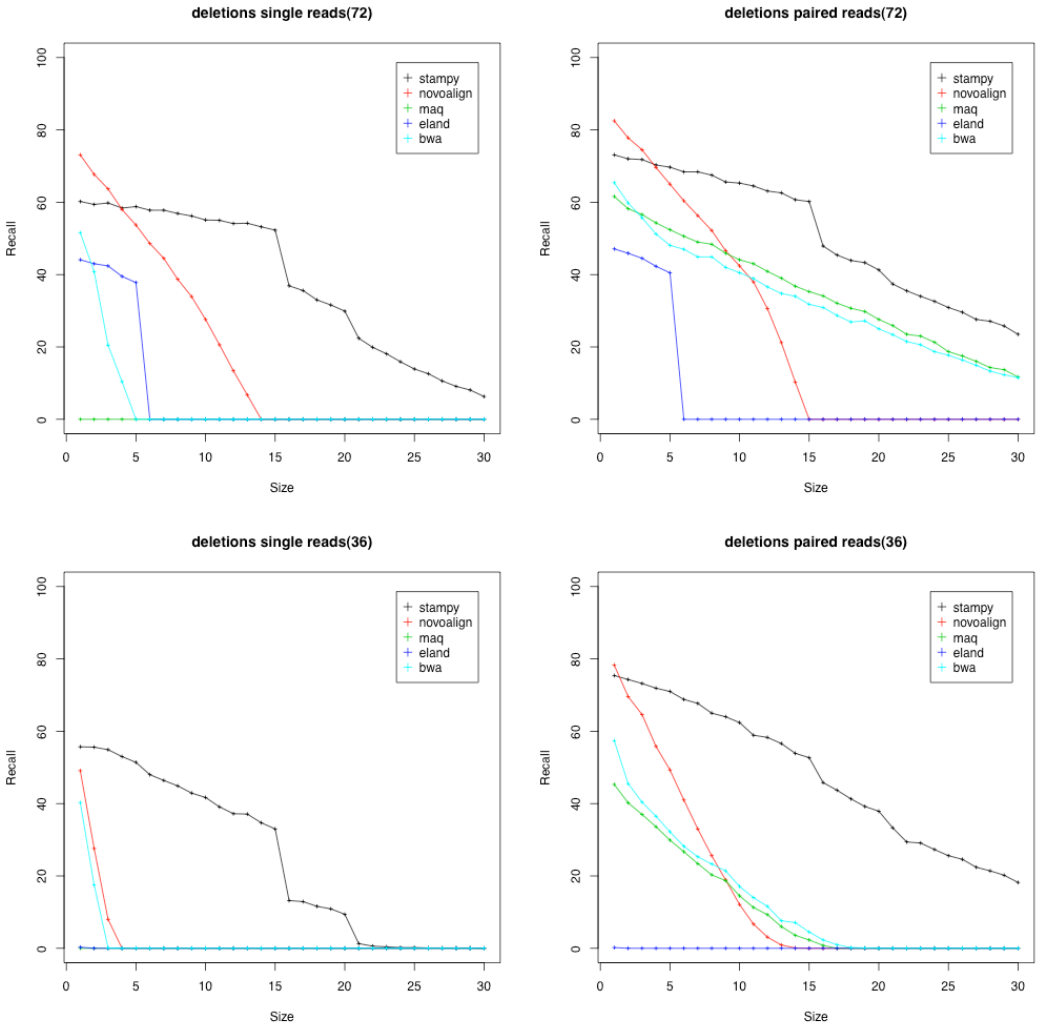

Figure S10. Indel recall rates by deletion length, for reads with one deletion per read or read pair.

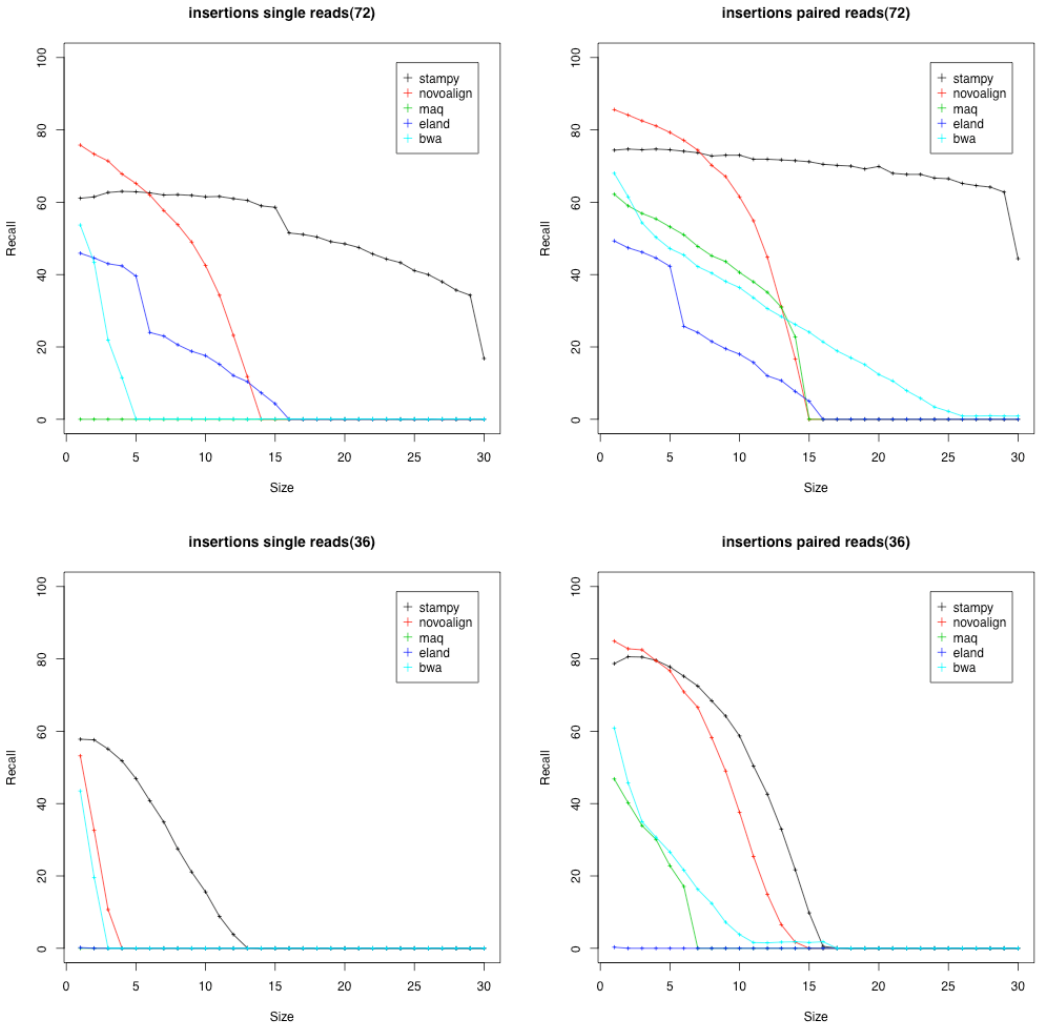

Figure S11. Indel recall rates by insertion length, for reads with one insertion per read or read pair.

4.5. Recall rate as function of divergence

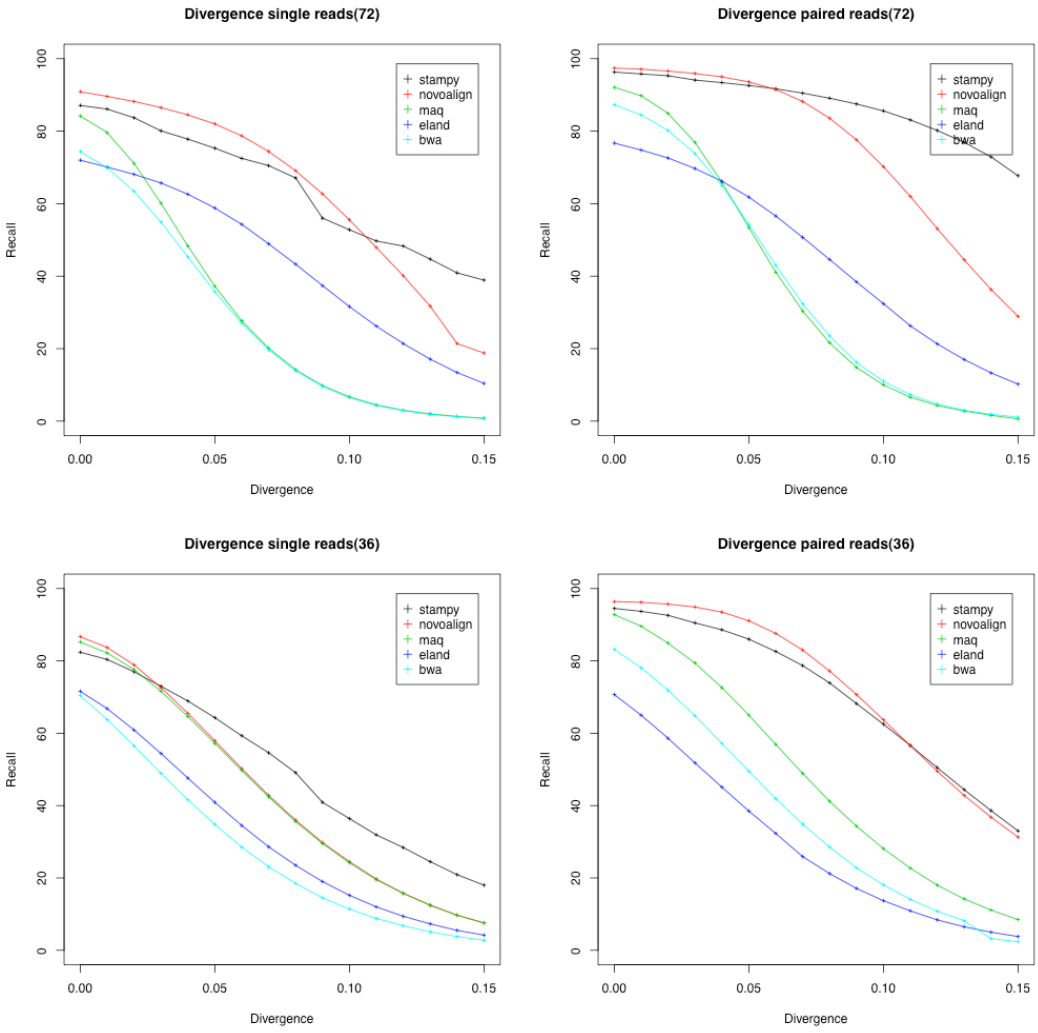

Figure S12. Mapping recall rates by divergence to the reference.

References

1 Cormen, T. H., Leiserson, C.E., Rivest, R.L. and Stein, C. *Introduction to Algorithms*. (MIT Press, 2001).

2 She, J. X., Bonhomme, F., Boursot, P., Thaler, L. & Catzeflis, F. Molecular Phylogenies in the Genus Mus - Comparative-Analysis of Electrophoretic, Scndna Hybridization, and Mtdna Rflp Data. *Biol J Linn Soc* **41**, 83-103 (1990).

3 Zhang, J. *et al.* SNPdetector: a software tool for sensitive and accurate SNP detection. *PLoS Comput Biol* **1**, e53, doi:10.1371/journal.pcbi.0010053 (2005).