Supplemental Material for Fiona: a parallel and automatic strategy for read error correction

Marcel H. Schulz, David Weese, Manuel Holtgrewe, Viktoria Dimitrova, Sijia Niu, Knut Reinert, and Hugues Richard

S1 CHOOSING THE K-MER RANGE

To determine, the best value for k_{min}, we extend a technique proposed by the authors of HiTEC (Ilie *et al.*, 2011), and modify it to account for heterogeneous read lengths by weighting the reads contribution by their individual lengths. Briefly, given the error rate ε and the distribution of read lengths $(\ell_i)_{i=1}^m$, we compute for each possible value of k the two quantities U_k and D_k.

The first, U_k, denotes the expected number of positions where errors are distributed in such a way that they cover all possible seeds of length k (termed uncorrectible in (Ilie *et al.*, 2011)).

$$U_k = \sum_{i=1}^{m} \ell_i \cdot \sum_{e=1}^{\ell_i} f_k(e, \ell_i)\ \varepsilon^e\ (1-\varepsilon)^{(l_i - e)} \quad , \tag{1}$$

where $f_k(e, \ell)$ is the number of sequences with e errors in a read of length ℓ such that any interval of length k contains at least one error. This value can easily be computed using a recurrence formula (Ilie *et al.*, 2011).

The second, D_k, denotes the expected number of positions where a sequencing error in the seed misdirects the genomic location it should be corrected to (termed destructible in (Ilie *et al.*, 2011)).

$$D_k = \sum_{i=1}^{m} \ell_i \cdot (1 - (1-q_k)^{\ell_i - k}) \cdot (1-\varepsilon)^{\ell_i} \quad , \tag{2}$$

and $q_k = (1-(1-\varepsilon)^k)(1-\varepsilon)(1-(1-1/4^k)^n) \cdot 3/4$. We then select k_{min} such that $U_k + D_k$ is minimal. The maximum value k_{max} is set to $k_{\text{min}} + 10$ to provide a sufficiently large interval to detect erroneous k-mers. In order to save computation time we do not use every k value in each round (see Section S2.5).

S2 IMPLEMENTATION DETAILS

Our implementation represents tree nodes by their intervals, where the root corresponds to the whole suffix array. To simulate the top-down traversal we have to determine the children of node's interval which are subintervals. To this end, we conduct binary searches for each character `A,C,G,T,N` following the interval's LCP and determine the LCP of each subinterval. For more details on the suffix tree traversal using the suffix array only, we refer the reader to (Navarro & Baeza-Yates, 2000).

In Fiona the top-down traversal does not exceed a string depth of k_{max}. Hence it suffices to lexicographically sort the suffixes in the suffix array by only their k_{max}-prefixes. Such a *partial suffix array* also called *h-order suffix array* (Larsson & Sadakane, 2007) was also used in (Zhao *et al.*, 2011) and (Siragusa *et al.*, 2013*a*) as an efficient generalized suffix tree implementation for NGS reads.

We discard suffixes of length less than k_{min} and construct the k_{max}-order suffix array in 2 steps. First, we use a hash table to group the suffixes into buckets by their q-prefix for a given $q < k_{\text{min}}$. We then refine the sorting from a prefix of length q to k_{max} using quick sort on each bucket and start the top-down search as described previously. As the refinement and traversal steps are data-independent among the buckets, we parallelized them using OpenMP. For load balancing each thread executes jobs of approximately the same number of suffixes. To further reduce the physical memory consumption of Fiona, we omit to store all suffixes in the hash table at once. Instead, we repeat the construction and traversal m times and in the i^{th} round consider only suffixes with a hash value h of the q-prefix in the interval $[h_{i-1}, h_i)$ $(0 = h_0 < h_1 < \ldots < h_m = 5^q)$. Despite an increase in the running time of a factor m, we can decrease the dominating size of the hash table by roughly the same factor by doing so.

S2.1 Seed extension

To efficiently compute the *right extension* $E(a, b)$ of two strings a and b we use a banded variant of Myers' bitvector algorithm (Myers, 1999) as first used in (Weese *et al.*, 2012) and specialized for seed extension in (Siragusa *et al.*, 2013*a*,*b*). To extend a seed with minimal errors one typically uses dynamic programming (DP). For seed extension with an upper bound of errors it suffices to compute cells within a band (parallelogram) instead of the whole DP matrix. Our banded bitvector approach exploits bit-parallelism of logical and arithmetic operations to compute whole columns of the DP parallelogram which is practically faster than computing the column cells seperately. For more details on the implementation we refer to (Siragusa *et al.*, 2013*b*).

S2.2 Storing potential corrections

Whenever an erroneous node is detected in a traversal, the children of the node are divided into potentially erroneous and correcting nodes according to the number of suffixes in the subtrees. For each erroneous suffix, different possible corrections for reads with this suffix are examined as explained above and shown in Figure 2. A linked list with corrections for each read is kept during the tree traversal. For each read position the best supporting correction is stored in the list. This list can be accessed by all threads, but is locked for other threads whenever one thread is modifying an entry in the list.

S2.3 Parallelized computation of optimal read corrections

After all subtrees have been traversed and potential corrections have been stored in the linked list, the list of correcting operations for each read and each position is iterated in parallel. For each read position in an erroneous read r the optimal set of corrections is computed as explained above.

S2.4 Handling of repeats and `N`s

Reads that are produced by the sequencer may contain the ambiguous base `N`. All previous suffix tree based methods omit reads that contain one or more `N`'s. In Fiona each base position with an `N` is considered erroneous and is attempted to be corrected. In addition, anchors containing `N`'s are neglected for the suffix tree traversal. When computing the overlap error rate to determine the set of correcting reads, `N`'s occurring in correcting reads as counted as errors, while `N`'s in the erroneous read are not.

To reduce false positive error candidates from small variations in repeat regions, we also ignore anchors that are tandem repeats of 1–6 nucleotides with at least 2 repetitions. As a second measure against repeats, we completely neglect mononucleotide q-gram buckets, e.g. `CCCCCC`, and also the most abundant q-gram buckets that in total contain a fraction of ρ of all q-grams, where ρ is the assumed repeat rate ($\rho = 0.01$ by default).

S2.5 Seed sampling

As another means for improving on running time, we implemented a seed sampling heuristic. Instead of examining all branching nodes in every round, we examine only every third node on a tree branch depending on the correction round. More precisely, we considered only seeds of length ℓ, where $\ell \equiv k_{\min} + u - 1 \mod s$. u denotes the number of the current round and s is the sampling rate. By default, Fiona uses a sampling rate of $s = 3$.

S2.6 Complexity

The asymptotic running time of Fiona is dominated by the search for errors and their corrections in a traversal of the suffix tree. The pseudo-code of this search is outlined in Algorithm 1 (notations are the same as in the main text). We omitted some details in the pseudo-code like counting votes, detecting which e acquires the most votes and insertion of the correction in the correction list as they have no influence on the asymptotic running time which is dominated by innermost computation, the seed extension.

Algorithm 1: Fiona($\mathcal{R}$, n, ε)

input : $\mathcal{R}$, the set of m reads of length ℓ
input : n, the genome length
input : ε, the expected read error rate

```
1  foreach suffix tree node αz do
2      if |α| + 1 ∈ [k_min, k_max] and |S(αz)| < cutoff(|α| + 1) then
3          foreach sibling αx of αz do
4              if |S(αx)| ≥ cutoff(|α| + 1) then
5                  foreach (r, i) ∈ S(αz) do
6                      foreach (s, j) ∈ S(αx) do
7                          foreach e ∈ {(1 1), (1 0), (0 1)} do
8                              compute right extension E(r[i + e'_1, |r|], s[j + e'_2, |s|]) with Myers algorithm
9                              ...
10                     ...
```

Worst-case running time. Assume a fixed string depth k and child nodes αx and their siblings αy of parent nodes α with $|\alpha| = k$. Then let $N_{\text{err}}(k)$ denote the number of pairs (r, i) iterated by lines 3 and 5 and let $N_{\text{for}}(k)$ denote the number of pairs (s, j) iterated by lines 4 and 6. As the spectrum of a node is the set of leaves below it and all nodes at a fixed depth k have disjoint sets of leaves, we can derive the following loose upper bounds $N_{\text{err}}(k) < m \cdot \ell$ and $N_{\text{cor}}(k) < m \cdot \ell$. The inner loop over the 3 possible error types (line 7) is executed for the product

of suffixes $N_{\text{err}}(k) \cdot N_{\text{cor}}(k) < m^2 \cdot \ell^2$. We traverse only nodes at depth $k \in [k_{\text{min}}, k_{\text{max}}]$ and as we chose $k_{\text{max}} := k_{\text{min}} + 10$ there are not more than 10 different depths to examine.

The innermost seed extension in line 8 aligns to suffixes of at most $\ell - k_{\text{min}}$ characters. As we limit the number of tolerated errors by $\lfloor \varepsilon \cdot \ell \rfloor$ the extension can be computed by a banded DP algorithm that computes only a band of $\lfloor \varepsilon \cdot \ell + 1 \rfloor$ diagonals and at most $\ell - k_{\text{min}}$ columns of the DP matrix. We use a banded variant of Myers bit-vector algorithm which computes 64 column cells at once and hence needs $\mathcal{O}(\ell \cdot \lfloor \frac{\varepsilon \cdot \ell}{64} \rfloor)$ time.
Thus the overall worst-case running time of one round of Fiona is:

$$\mathcal{O}\left(10 \cdot m^2 \cdot \ell^2 \cdot 3 \cdot \ell \cdot \left\lfloor \frac{\varepsilon \cdot \ell}{64} \right\rfloor\right) \subset \mathcal{O}\left(\varepsilon \cdot m^2 \cdot \ell^4\right) \quad .$$

This time already includes the partial suffix array constructing which requires $\mathcal{O}\left(m \cdot \ell \cdot \log\left(m \cdot \ell\right) \cdot k_{\text{max}}\right)$ time to sort all suffixes by their prefix of length k_{max}.

Expected running time. However this worst case bound does not reflect the complexity of Fiona in practice. So we give the expected running time as well for which we can make some more stringent assumptions. At one string depth k the number of seeds of erroneous reads can be expected to be a ε-fraction of all seeds at one level k, i.e. $N_{\text{err}}(k) < \varepsilon \cdot m \cdot \ell$. For each erroneous seed we expect to find λ_k correct seeds, where $\lambda_k = m \cdot \frac{\ell - k}{n}$ is the expected seed coverage. Hence, the loop in line 7 will be executed $\approx \frac{\varepsilon \cdot m^2 \cdot \ell^2}{n}$ times.

The expected running time of one round of Fiona is:

$$\mathcal{O}\left(10 \cdot \frac{\varepsilon \cdot m^2 \cdot \ell^2}{n} \cdot 3 \cdot \ell \cdot \left\lfloor \frac{\varepsilon \cdot \ell}{64} \right\rfloor \; + \; m \cdot \ell \cdot \log\left(m \cdot \ell\right) \cdot k_{\text{max}}\right) \subset \mathcal{O}\left(\frac{\varepsilon^2 \cdot m^2 \cdot \ell^4}{n} \; + \; m \cdot \ell^2 \cdot \log\left(m \cdot \ell\right)\right) \quad .$$

We can then rewrite the expected running time as a function of genome length and coverage $\lambda = \lambda_1$:

$$\mathcal{O}\big(n \cdot \varepsilon^2 \cdot \lambda^2 \cdot \ell^2 \; + \; n \cdot \lambda \cdot \ell \cdot \log\left(n \cdot \lambda\right)\big)$$

which is linear on genome length and quadratic with coverage. The worst-case and expected case running times were given for one round only. However, the total running time has the same complexity as the automatic round selection will stop after at most a constant number of 6 rounds.

Memory consumption. The data structures required by Fiona are the reads which have $m \cdot \ell$ characters in total, the suffix array with $m \cdot (\ell - k_{\text{min}} + 1)$ entries and the correction list with $\mathcal{O}\left((k_{\text{max}} - k_{\text{min}}) \cdot \varepsilon \cdot m \cdot \ell\right)$ entries in expectation if all errors would be found at each string depth k. Hence, the overall memory consumption is linear in the total length of reads

$$\mathcal{O}(m \cdot \ell) \quad .$$

S3 DETAILS OF EVALUATION SETUP

All experiments presented in this manuscript were run on a computer with an 8-core Intel Xeon X5550 @2.67Ghz processor and 72 GB of RAM running Debian Linux 6.0.6.

S3.1 Data sets and genomes used

Table S1. Identifiers and sources of the used reference sequences

organism	accession	genome length
B. pertussis 18323	NC_018518.1	4 043 846
E. coli K-21	NC_000913.2	4 639 675
E. coli O104:H4	NC_018658.1	5 273 097
H. sapiens	GRCh37	2 861 343 787
P. syringae	NC_007005.1	6 093 698
P. falciparum 3D7	ASM276v1	23 264 338
S. cerevisae	NCBI release 54	12 156 676
S. aureus LGA251	NC_017348, NC_017349	2 799 725
C. elegans	ENSEMBL release 60	100 286 070
D. melanogaster	flybase r5.29	120 381 546

Table S2. The experimental read data sets used for the evaluation

organism	accession	avg. length	read count	coverage	Gbp
B.pertussis	ERR161541[2]	142 bp	2 464 690	85x	0.3
C. elegans	SRR443373[I]	100 bp	29 657 035	30x	3
D. melanogaster	SRR492060[I]	76 bp	51 727 822	28x	3.4
D. melanogaster	SRX016210[1]	544 bp	4 692 486	18x	2.2
E. coli K-12	ERR022075[I]	100 bp	22 720 100	490x	2.3
E. coli K-12	ERR022075[I]	100 bp	1 378 122	30x	0.14
E. coli K-12	SRR000868[1]	253 bp	230 517	13x	0.06
E. coli K-12	ERR039477[2]	92 bp	390 976	8x	0.04
E. coli K-12	SRR611140[2]	162 bp	4 669 065	163x	0.8
E. coli K-12	SRR620425[2]	170 bp	4 237 734	156x	0.7
E. coli O104:H4	SRR254209[2]	178 bp	977 971	32x	0.2
H. sapiens	SRR1238539[2]	177 bp	186 132 134	11x	31.5
P. falciparum	ERR161543[2]	154 bp	1 959 564	13x	0.3
P. syringae	ERR005143[I]	36 bp	14 204 532	42x	0.26
S. aureus	ERR236069[2]	228 bp	1 338 465	109x	0.31
S. aureus	SRR070596[1]	514 bp	185 384	34x	0.1
S. cerevisae	SRR031259[I]	36 bp	7 485 708	22x	0.27
S. cerevisae	SRX039441[1]	274 bp	690 237	16x	0.19

[1] 454. [2] Ion Torrent,[I] Illumina.

S3.2 Performance evaluation

Gain Computation with compute_gain. For the evaluation of read correction quality, the metric *gain* has been established in (Yang *et al.*, 2010) and (Yang *et al.*, 2013) as a good summary of both sensitivity and precision. We use the gain metric for edit distance: For both 454 and IonTorrent data, the uncorrected reads are mapped against the known reference genome using BWA-SW (Li & Durbin, 2010) with default parameters. BWA-SW yields the best local match in a SAM file which is used as the predicted true origin in the rest of the evaluation.

For the evaluation, we developed a tool called `compute_gain` which employs multi-core parallelism and is included in the Fiona distribution. This tool reads the SAM file generated by BWA-SW and the corrected FASTQ file that the correction tool produces. The full uncorrected and corrected read sequences are then aligned around the predicted origin position using a banded DP alignment algorithm

from (Döring *et al.*, 2008). The gain can be computed by $(b - a)/b$ where a and b are the sums over the number of errors after and before correction over all reads. When more errors are introduced than corrected over all the reads, the gain takes a negative value.

Handling Local Alignments for 454 and IonTorrent Data. Since BWA-SW yields local alignments, we use a similar scheme for filtering BWA-SW matches as in (Yang *et al.*, 2013): Reads where fewer than 30 bases are aligned by the read mapper are ignored. We also ignore reads aligning with more than 10 errors or with an overall error rate of more than 20 %.

Besides the gain, we also compute the base error rate which is the total number of edit distance errors divided by the total number of bases.

Sensitivity and Specificity Computation. Computing other metrics such as sensitivity and specificity values requires that false positive correction, true positive corrections etc. can be defined and computed in an unique way. As we assess the true originating position of the reads using the best alignment to the genome, we have to avoid changes in the alignment before and after correction which would create ambiguities. There is not such a problem when using Hamming distances, as each aligned base of the read is compared to the same exact position on the reference genome before and after correction.

For edit distance, however, positions of gaps in each read alignment can change before and after correction. For instance, even a substitution of a bp can change the position it aligns to on the reference genome. There, read bases untouched during read correction and that previously aligned without an error, might after correction align with a mismatch, even though their status was unchanged.

In their study, (Yang *et al.*, 2013) propose an approximation of the true/false positive/negative counts using a scheme that we describe below. Our program `compute_gain` also allows for such a computation but because of the limitations on corrections using edit distance operations, we only give these metrics in the supplement.

The approach to compute the metrics for sensitivity and specificity and the counts required for this computation are as follow. Collect a quadruple (p, o, c, w) for each erroneous alignment base where p is the reference position, o is an offset in gaps in the reference, c is the correct base (reference base, can be $-$ for gaps in the reference) and w is the wrong base in the read. The sets B and A contain the errors **b**efore and **a**fter the correction. The true positives can now be computed as $|B \setminus A|$, false positives as $|A \setminus B|$, and the number of false negatives as $|A \cap B|$. The number of true negatives is the number of reference positions before correction that are not in A nor B.

Consider the following example for a case where the algorithm described above does not count the true/false positives/negative corrections correctly. The read `ACGCTCTACG` aligns against the reference `ACGATCTGTACG` with edit distance 3. The read error correction method corrects the read to `ACGCTGTACG` (replacing the third `C` from the left by a `G`). This makes the read align against the reference with edit distance 2.

```
before correction                    after correction

               11                                  11
pos  012345678901                        012345678901
        *   **                              **
REF  acgatctgtacg                    REF  acgatctgtacg
     |||.|||  |||                         |||  |||||||
READ acgctct--acg                    READ acg--ctgtacg
          ^
          `-- change to ``g''

  edit distance = 3                      edit distance = 2
```

Since the edit distance alignment before correction was ambiguous, the alignment errors are not correlated correctly. The algorithm above counts one uncorrected base (false negative) at position 3, one introduced error (false positive) at position 4, and two corrected errors at position 7 and 8 (true positives). However, the corrected base was the C previously aligned to position 5 and after correction aligned to position 7 as G. Thus, there actually is one corrected error, two uncorrected errors, and no introduced error.

Note that the alignment algorithm that we used for edit distance place gaps at the rightmost possible position. Any dynamic programming (DP) alignment algorithm implementation has to decide how to handle ambiguous traces through the DP matrix, effectively placing gaps at the leftmost or rightmost possible position. Using the reverse complement for both read and reference in the example above gives an example for incorrect counts for a DP alignment algorithm placing gaps at the leftmost position.

S4 ROBUSTNESS EVALUATION

Table S3 shows the robustness of Fiona with respect to the expected error rate. The parameter of ϵ was varied between 2 % and 10 % when correcting the 13x *E. coli* and the 34x *S.aureus* data sets with varying coverage.

S5 PROGRAM VERSION AND PARAMETRIZATION DETAILS

For all programs the gain analysis was done on the complete read set, even if some programs discarded reads from their output.

Table S3. Results of robustness evaluation

			best	fraction of best gain value with $\varepsilon =$								
	data set		gain	2 %	3 %	4 %	5 %	6 %	7 %	8 %	9 %	10 %
E.coli	SRR000868	13x	90.5	95.52	98.75	99.92	100.00	99.60	92.58	92.64	91.80	87.73
S.aureus	SRR070596	34x	74.9	87.15	89.39	92.74	93.30	100.00	85.47	85.47	85.47	84.92

Fraction of the best gain for different given values for the per-base error rate ε. Fiona was run one *E. coli* and one *S.aureus* data set with different values of the per-base error rate between 2 % and 10 %.

S5.1 Fiona

Fiona was used in version 0.2. The values chosen automatically by Fiona for $k_{\min}$ were 14 for the *E. coli* data sets, 15 for *S. cerevisae*, 16 for *RAL399-2L*, and 17 for *C. elegans* and *D. melanogaster*. We used the parameter `-t 8` for eight threads, and pass the genome length using `-g GLEN`. A q-gram of length 10 was hard-coded into Fiona for benchmarking (and can be changed after recompiling). Fiona uses a default error rate of 0.05 and Fiona-H for Illumina data uses a default of 0.01. Example:

```
fiona -nt 8 -g 1000 IN.fq OUT.fa.
```

S5.2 Allpaths-LG

Allpaths-LG was used in release 44994. We ran the read correction program `ErrorCorrectReads.pl` with the settings `THREADS=8 PHRED_ENCODING=33 REMOVE_DODGY_READS=0 UNPAIRED_READS_IN=IN.fq READS_OUT=OUT.fq`. The output was post-processed to add back the reads that were removed by the read correction routine although `REMOVE_DODGY_READS=0` was set. Example:

```
ErrorCorrectReads.pl THREADS=8 PHRED_ENCODING=33 REMOVE_DODGY_READS=0 UNPAIRED_READS_IN=IN.fq
READS_OUT=OUT.fq
```

S5.3 Coral

Coral was used in version 1.4. It was run with default parameters, i.e. `-fq` for the input FASTQ files, `-o` for the output file, and `-454` for both 454 and IonTorrend data. For some 454 and Ion Torrent data, Coral's parameters had to be adjusted to allow higher error rates to yield better gain (as instructed by the Coral authors).

We ran Coral with default parameters (an error rate of 7 % or `-e 0.07`). We also tried to increase the error rate in Coral to 10 %, 15 %, 20 %, and 25 %. The best variant for each data set is given as Coral*. We stopped Coral after 24 hours on the 18x *D. melanogaster* data set with `-e 0.20` and `-e 0.25`.

Coral uses 8 threads by default. Example:

```
coral -fq IN.fq -o OUT.fa -454 -e 0.10.
```

S5.4 HybridShrec

HybridShrec was used in version 1.0 (BaseSpaceShrec). We ran HybridShrec in two versions. The variant HybridShrec was run with default parameters as instructed by the authors. For most data sets, the program did not run through but exited with the suggestion to lower the strictness parameter `-s`. Because the achieved gain did not correlate with the strictness parameter value, we ran it with strictness values set from 2 to 7 and reported the best achieved gain.

The default setting for levels to use is 14 to 17. The variant HybridShrec^F used the same levels as fiona. This variant was run with strictness values 2 to 7 as well. Example:

```
java -Xmx40960m Shrec -n 5348428 -s 7 IN.fa
```

S6 RUNNING TIME AND MEMORY CONSUMPTION

Table S4 shows the running time and memory consumption of the evaluated tools on 454 and IonTorrent data. Table S5 shows the same metrics for Illumina data.

Table S4. Running time and memory consumption for 454 (top) and IonTorrent (bottom) experiments

data set		Gbp	Allpaths-LG time	Allpaths-LG mem	Coral time	Coral mem	Coral* time	Coral* mem	Fiona time	Fiona mem	HybridShrec* time	HybridShrec* mem	HybridShrec^F time	HybridShrec^F mem
D. melanogaster	18x	2.2	**145.0**	**11**	496.1	59	1414.1	60	240.7	18	333.2	41	499.5	42
E. coli K-12	13x	0.06	1.0	**0**	**0.8**	3	0.9	3	2.5	1	4.8	5	5.0	12
S. aureus	34x	0.1	**3.0**	**1**	5.5	5	112.2	5	12.3	1	12.0	14	13.6	15
S. cerevisae	16x	0.19	**6.5**	**1**	7.1	5	19.6	5	13.1	2	22.5	15	30.5	15
B. pertussis	85x	0.3	**6.0**	**2**	13.5	9	81.2	9	32.0	3	58.3	17	54.0	21
E. coli K-12	8x	0.04	**2.6**	**0**	3.4	3	4.4	3	3.1	1	7.1	5	9.2	8
E. coli K-12	163x	0.8	**14.2**	**4**	243.0	13	373.8	13	118.3	9	111.3	19	160.2	6
E. coli K-12	156x	0.7	**15.0**	7	249.1	12	290.1	12	49.2	8	111.4	18	111.0	**6**
E. coli O104:H4	32x	0.2	**3.8**	**1**	5.3	8	12.6	8	15.2	2	21.7	15	28.7	16
H. sapiens[1]	11x	31.5	**572.8**	**129**		—[2]		—[2]	1187.1	244		—[2]		—[2]
P. falciparum	13x	0.3	**5.6**	**1**	11.0	11	24.8	11	20.5	3	38.9	16	49.7	20
S. aureus	109x	0.31	**4.4**	**1**	12.0	13	175.8	13	43.7	3	51.1	18	53.8	29

[1] The programs were run on machine with 16 physical and 32 virtual cores and 370 GB of RAM. [2] Out of memory. — Time (in minutes and fractions thereof) and memory (in GB, rounded to the next GB) for the read correction runs from Table 1. For each data set, the results with the lowest running time and memory are given in bold. The results are separated by sequencing technology, the 454 results are above the IonTorrent results.

Table S5. Running time and memory consumption for Illumina experiments

data set		Allpaths-LG time	Allpaths-LG mem	Coral time	Coral mem	ECHO time	ECHO mem	Fiona-H time	Fiona-H mem	HiTEC time	HiTEC mem	Quake time	Quake mem
C. elegans	30x	**61.73**	**13.31**	162.53	49.33		—[2]	445.06	22.27		—[1]	13.79	16.10
D. melanogaster	5x	**9.64**	**3.22**	21.78	17.53	307.59	15.89	31.19	6.24	82.65	13.68	26.37	11.88
D. melanogaster	28x	83.07	15.54	172.36	54.41		—[2]	108.12	33.55		—[1]	**29.81**	**12.37**
E. coli K-12	30x	3.57	0.73	4.44	3.54	67.58	4.11	**2.72**	1.68	13.34	2.58	4.01	**0.15**
E. coli K-12	490x	51.47	9.07	261.33	25.55		—[1]	56.43	16.85	277.32	9.84	**42.72**	**5.80**
P. syringae	21x	2.17	**0.59**	**1.71**	3.71	55.45	2.86	2.73	1.63	11.74	2.53	6.57	1.05
S. cerevisae	22x	**4.00**	1.15	9.75	5.21	102.57	15.14	4.64	3.21	28.27	4.96	8.73	**0.64**

Time (in minutes and fractions thereof) and memory (in GB, rounded to the next GB) for the read correction runs from Table 3. For each data set, the results with the lowest running time and memory are given in bold. [1] The program crashed. [2] The program ran too long.

S7 SENSITIVITY AND SPECIFICITY RESULTS

Table S6 shows the sensitivity and specificity results as well as the number of true and false positives of the evaluated tools on 454 and IonTorrent data. Table S7 shows the same metrics for Illumina data.

Table S6. Sensitivity and specificity results on 454 (top) and IonTorrent data (bottom)

		Allpaths-LG				Coral				Coral*			
data set		sens.	spec.	TP	FP	sens.	spec.	TP	FP	sens.	spec.	TP	FP
D. melanogaster	18x	9.5	100.0	431,174	26,579	45.2	99.9	2,061,475	292,128	68.0	99.8	3,097,777	667,718
E. coli K-12	13x	56.2	100.0	156,100	4,884	59.4	99.9	165,031	34,931	76.4	99.9	212,002	68,012
S. aureus	34x	27.2	99.9	65	8	0.0	100.0	0	0	92.9	99.7	222	43
S. cerevisae	16x	19.9	100.0	92,094	6,799	0.7	100.0	3,165	583	11.1	99.9	51,326	37,492
B. pertussis	85x	41.5	100.0	1,343,905	43,636	0.0	100.0	711	104	45.2	99.5	1,464,858	473,514
E. coli K-12	163x	18.3	100.0	1,880,350	236,298	73.8	99.8	7,590,588	1,450,793	98.5	99.6	10,132,261	2,550,490
E. coli K-12	156x	41.3	99.9	3,188,726	717,730	92.5	99.7	7,147,691	2,429,213	99.1	99.7	7,653,796	1,882,367
E. coli O104:H4	32x	40.8	100.0	180,646	1,217	0.0	100.0	19	9	49.2	99.2	218,026	68,296
H. sapiens	11x	14.7	99.9	66,887,396	14,645,519				—-				—-
P. falciparum 3D7	13x	22.5	100.0	106,451	5,160	0.0	100.0	177	33	46.5	98.9	220,013	101,915
S. aureus	109x	16.1	100.0	130,036	9,786	0.3	100.0	2,503	542	73.8	99.4	596,332	136,676
E. coli	8x	31.7	100.0	117,828	3,880	62.2	99.9	231,115	47,576	82.2	99.8	305,096	68,200

		Fiona				HybridShrec*				HybridShrec^F			
data set		sens.	spec.	TP	FP	sens.	spec.	TP	FP	sens.	spec.	TP	FP
D. melanogaster	18x	67.9	100.0	3,097,392	151,321	48.3	96.3	2,200,434	14,943,564	41.5	100.0	1,892,893	152,683
E. coli K-12	13x	92.8	100.0	257,697	6,379	58.8	99.9	163,164	47,089	43.0	100.0	119,441	7,667
S. aureus	34x	35.2	100.0	84	7	40.6	99.5	97	74	22.2	100.0	53	4
S. cerevisae	16x	40.1	100.0	185,299	18,633	28.5	99.8	131,782	106,449	25.1	100.0	116,000	9,119
B. pertussis	85x	77.8	99.8	2,521,238	161,532	30.3	91.1	981,897	8,611,563	7.2	99.4	232,814	546,348
E. coli K-12	163x	84.6	100.0	8,695,573	341,386	4.2	99.7	427,098	2,370,597	0.0	100.0	0	0
E. coli K-12	156x	76.6	100.0	5,915,443	193,441	4.7	99.7	359,316	2,225,252	0.0	100.0	0	0
E. coli O104:H4	32x	76.1	99.7	337,107	30,122	20.7	99.8	91,465	17,273	20.7	99.8	91,484	17,289
H. sapiens	11x	52.0	99.9	236,114,218	24,304,808				—-				—-
P. falciparum 3D7	13x	69.7	99.2	329,796	73,538	31.4	96.0	148,631	391,500	15.3	99.7	72,315	32,056
S. aureus	109x	67.8	99.9	547,443	22,926	3.4	99.5	27,286	135,833	1.6	100.0	12,617	10,013
E. coli	8x	84.3	99.9	313,159	26,809	60.4	99.8	224,310	140,519	37.8	100.0	75,576	13,199

Sensitivity (*sens.*) and specificity (*spec.*) are given in percent. Also, the table shows true positives (*TP*) and false positives (*FP*). See Table S4 for the reasons of missing values.

Table S7. Sensitivity and specificity results on Illumina data

		Allpaths-LG				Coral				ECHO			
data set		sens.	spec.	TP	FP	sens.	spec.	TP	FP	sens.	spec.	TP	FP
C. elegans	30x	29.8	100.0	1,847,743	84,120	35.9	99.8	2,224,305	2,453,521				—-
D. melanogaster	5x	34.1	100.0	2,472,286	94,512	52.7	99.9	3,820,682	412,417	44.3	100.0	3,207,820	146,833
D. melanogaster	28x	33.6	100.0	11,676,976	516,280	33.3	99.9	5,416,789	1,678,511				—-
E. coli K-12	30x	99.4	100.0	277,941	353	99.1	100.0	276,949	21,455	93.0	100.0	259,843	5,117
E. coli K-12	490x	98.3	100.0	4,532,069	3,803	97.3	100.0	4,483,687	58,367				—-
P. syringae	42x	75.4	100.0	524,048	6,626	84.7	100.0	589,084	35,909	92.2	100.0	640,955	7,567
S. cerevisae	22x	58.1	100.0	587,218	20,427	60.9	99.9	615,641	162,862	35.2	100.0	355,528	28,853

		Fiona				HiTEC				Quake			
data set		sens.	spec.	TP	FP	sens.	spec.	TP	FP	sens.	spec.	TP	FP
C. elegans	30x	30.1	100.0	1,865,313	300,769				—-	18.2	100.0	1,126,131	71,476
D. melanogaster	5x	53.3	100.0	3,861,950	218,559	40.6	99.8	2,939,537	1,207,761	47.7	100.0	3,455,254	51,454
D. melanogaster	28x	34.3	100.0	11,911,334	1,012,857				—-	68.9	100.0	10,236,997	133,916
E. coli K-12	30x	98.7	100.0	276,003	2,291	94.4	100.0	263,847	3,353	95.6	100.0	267,312	175
E. coli K-12	490x	99.2	100.0	4,571,973	26,034	94.8	100.0	4,369,759	45,666	88.2	100.0	4,065,202	4,742
P. syringae	42x	95.5	100.0	663,786	28,732	94.5	99.9	657,018	118,634	81.2	100.0	564,406	10,534
S. cerevisae	22x	78.0	99.9	788,641	195,792	72.4	99.6	731,798	964,394	53.8	100.0	543,952	16,552

Sensitivity (*sens.*) and specificity (*spec.*) are given in percent. Also, the table shows true positives (*TP*) and false positives (*FP*). See Table S5 for the reasons of missing values.

S8 FULL CORAL RESULTS

Table S8 and Table S9 show the full results of all Coral runs in terms of quality and resource consumption. We stopped the Coral runs after 24 h.

Table S8. Full gain and error rate results for Coral with different value for the `-e` parameter

data set		original e-rate	Coral -e 0.07 e-rate	Coral -e 0.07 gain	Coral -e0.10 e-rate	Coral -e0.10 gain	Coral -e0.15 e-rate	Coral -e0.15 gain	Coral -e0.20 e-rate	Coral -e0.20 gain	Coral -e0.25 e-rate	Coral -e0.25 gain
B. pertussis	85x	3.71	3.71	0.02	3.71	0.06	3.69	0.40	3.55	4.18	**2.57**	**30.60**
D. melanogaster	18x	1.17	0.72	38.81	0.58	50.68	**0.55**	**53.30**	–[1]	–[1]	–[1]	–[1]
E. coli K-12	13x	1.06	0.54	49.42	0.43	59.25	**0.38**	**63.79**	0.39	63.28	0.41	61.46
E. coli K-12	8x	0.62	0.33	46.86	**0.30**	**51.86**	0.32	48.07	0.36	42.47	0.40	35.72
E. coli K-12	162x	1.46	0.59	59.70	0.41	71.62	**0.38**	**73.72**	0.39	73.11	0.42	71.37
E. coli K-12	156x	1.11	0.43	61.07	0.32	71.37	**0.28**	**74.70**	0.29	73.60	0.32	70.59
E. coli O104:H4	32x	5.19	5.19	0.00	5.19	0.03	5.15	0.66	4.80	7.44	**3.44**	**33.82**
P. falciparum 3D7	13x	5.06	5.05	0.03	5.05	0.17	4.97	1.60	4.57	9.65	**3.80**	**24.94**
S. aureus	109x	3.32	3.32	0.24	3.29	1.08	3.03	8.90	1.92	42.38	**1.44**	**56.91**
S. aureus	34x	1.76	1.76	0.00	1.73	2.09	1.51	14.23	1.00	43.51	**0.44**	**74.90**
S. cerevisae	16x	0.95	0.95	0.56	0.94	1.06	0.93	2.41	**0.92**	**2.99**	1.01	-5.84

The values selected for Coral* are highlighted in bold. — [1] Coral did not finish within 24 h.

Table S9. Full running time (in minutes) and memory consumption (in GB) results for Coral with different values for the `-e` parameter

data set		Coral -e 0.07 time	Coral -e 0.07 mem	Coral -e0.10 time	Coral -e0.10 mem	Coral -e0.15 time	Coral -e0.15 mem	Coral -e0.20 time	Coral -e0.20 mem	Coral -e0.25 time	Coral -e0.25 mem
B. pertussis	85x	9.19	24	9.19	24	9.19	24	9.23	24	**9.39**	**24**
D. melanogaster	18x	59.13	73	59.41	73	**59.62**	**73**	–[1]	–[1]	–[1]	–[1]
E. coli K-12	13x	2.61	17	2.61	17	**2.61**	**17**	2.61	17	2.62	17
E. coli K-12	8x	2.86	18	**2.86**	**18**	2.87	18	2.87	17	2.87	17
E. coli K-12	162x	13.21	28	13.21	28	**13.21**	**28**	13.21	28	13.21	28
E. coli K-12	156x	11.96	27	11.95	27	**11.94**	**27**	11.94	27	11.94	27
E. coli O104:H4	32x	8.24	23	8.24	23	8.25	23	8.27	23	**8.42**	**23**
P. falciparum 3D7	13x	11.02	26	11.02	26	11.03	26	11.08	26	**11.35**	**26**
S. aureus	109x	12.92	28	12.93	28	13.00	28	13.23	28	**13.26**	**28**
S. aureus	34x	4.72	19	4.73	19	4.79	19	4.86	19	**4.85**	**19**
S. cerevisae	16x	5.06	20	5.06	20	5.07	20	**5.10**	**20**	5.18	20

The values selected for Coral* are highlighted in bold. — [1] Coral did not finish within 24 h.

S9 FULL HYBRIDSHREC RESULTS

Table S10 and Table S11 show the full results of all HybridShrec (HS) runs in terms of quality and resource consumption. Some values are missing because HS did not work with the given strictness value or the program terminated.

Table S10. Full gain and error rate results for HybridShrec (HS) with different values for the strictness (`-s`) parameter.

		original	HS*		HS^F		HS*		HS^F		HS*		HS^F		HS*		HS^F		HS*		HS^F		HS*		HS^F	
			-s 2		-s 2		-s 3		-s 3		-s 4		-s 4		-s 5		-s 5		-s 6		-s 6		-s 7		-s 7	
data set		e-rate	e-rate	gain	e-rate	gain	e-rate	gain	e-rate	gain	e-rate	gain	e-rate	gain	e-rate	gain	e-rate	gain	e-rate	gain	e-rate	gain	e-rate	gain	e-rate	gain
B. pertussis	85x	3.71	–[1]	–[1]	4.07	-9.68	–[1]	–[1]	4.11	-10.89	–[1]	–[1]	4.15	-11.83	–[1]	–[1]	4.17	-12.61	–[1]	–[1]	4.20	-13.22	**12.44**	**-235.48**	4.18	-12.78
D. melanogaster	18x	1.17	13.89	-1,086.21	1.15	1.89	14.17	-1,109.80	1.08	7.94	10.72	-813.56	–[1]	–[1]	**4.46**	**-279.51**	**0.73**	**38.17**	–[1]	–[1]	–[1]	–[1]	–[1]	–[1]	–[1]	–[1]
E. coli K-12	8x	1.06	1.20	-12.98	0.81	23.52	**0.64**	**40.05**	**0.70**	**34.28**	–[1]	–[1]	–[1]	–[1]	–[1]	–[1]	–[1]	–[1]	–[1]	–[1]	–[1]	–[1]	–[1]	–[1]	–[1]	–[1]
E. coli K-12	13x	0.62	1.20	-93.69	0.55	10.50	0.97	-56.04	0.46	25.55	**0.36**	**41.81**	**0.37**	**40.26**	–[1]	–[1]	–[1]	–[1]	–[1]	–[1]	–[1]	–[1]	–[1]	–[1]	–[1]	–[1]
E. coli K-12	163x	1.46	**1.73**	**-18.90**	**1.46**	**0.00**	1.81	-23.85	1.46	0.00	1.86	-27.80	1.46	0.00	1.90	-30.45	1.46	0.00	1.94	-33.01	1.46	0.00	2.07	-41.73	1.46	0.00
E. coli K-12	156x	1.11	**1.38**	**-24.15**	**1.11**	**0.00**	1.45	-30.62	1.11	0.00	1.50	-35.16	1.11	0.00	1.53	-37.76	1.11	0.00	1.56	-40.71	1.11	0.00	1.68	-51.48	1.11	0.00
E. coli O104:H4	32x	5.19	5.40	-3.98	5.16	0.51	5.44	-4.82	5.16	0.61	5.56	-7.11	5.15	0.77	5.79	-11.56	5.12	1.35	6.08	-17.35	4.83	6.87	**4.39**	**15.36**	**4.31**	**16.76**
P. falciparum 3D7	13x	5.06	13.27	-162.26	5.14	-1.76	11.38	-124.62	4.87	3.54	**7.67**	**-51.29**	**4.63**	**8.50**	–[1]	–[1]	–[1]	–[1]	–[1]	–[1]	–[1]	–[1]	–[1]	–[1]	–[1]	–[1]
S. aureus	109x	3.32	**3.77**	**-13.44**	**3.31**	**0.32**	**3.89**	-17.02	3.32	0.04	4.04	-21.57	3.33	-0.33	4.26	-28.01	3.35	-0.88	4.55	-36.89	3.38	-1.64	5.00	-50.38	3.41	-2.79
S. aureus	34x	1.76	2.26	-28.03	1.68	4.60	2.28	-29.71	1.70	3.77	2.64	-49.79	1.70	3.35	3.11	-76.57	1.72	2.51	2.50	-41.84	1.53	13.39	**1.59**	**9.62**	**1.40**	**20.50**
S. cerevisae	16x	0.95	2.57	-169.97	0.92	3.67	3.17	-233.71	0.88	7.09	2.06	-116.27	0.76	19.85	**0.90**	**5.48**	**0.73**	**23.11**	–[1]	–[1]	–[1]	–[1]	–[1]	–[1]	–[1]	–[1]

The values selected for HybridShrec* and HybridShrec^F are highlighted in bold. — [1] HybridShrec could not run for the given value of `-s`.

Table S11. Full running time (in minutes) and memory consumption (in GB) results for HybridShrec (HS) with different values for the strictness (`-s`) parameter.

		HS*		HS^F		HS*		HS^F		HS*		HS^F		HS*		HS^F		HS*		HS^F		HS*		HS^F	
		-s 2		-s 2		-s 3		-s 3		-s 4		-s 4		-s 5		-s 5		-s 6		-s 6		-s 7		-s 7	
data set		time	mem	time	mem	time	mem	time	mem	time	mem	time	mem	time	mem	time	mem	time	mem	time	mem	time	mem	time	mem
B. pertussis	85x	–[1]		**54.01**	**21**	–[1]		55.24	20	–[1]		53.71	19	–[1]		53.58	19	–[1]		54.27	20	**58.31**	**17**	54.42	19
D. melanogaster	18x	387.56	42	549.71	42	352.42	41	537.02	42	352.44	41	–[1]		333.22	41	499.52	42	–[1]		–[1]		–[1]		–[1]	
E. coli K-12	8x	6.72	5	5.57	12	**4.81**	**5**	**5.04**	**12**	–[1]		–[1]		–[1]		–[1]		–[1]		–[1]		–[1]		–[1]	
E. coli K-12	13x	11.46	7	11.75	12	6.95	5	11.03	12	**7.08**	**5**	**9.16**	**8**	–[1]		–[1]		–[1]		–[1]		–[1]		–[1]	
E. coli K-12	163x	**111.33**	**19**	**160.21**	**6**	110.03	18	116.05	7	108.12	19	115.84	7	106.06	18	115.29	7	109.48	18	115.73	7	106.34	18	116.51	7
E. coli K-12	156x	**111.36**	**18**	**110.96**	**6**	103.76	18	110.35	6	104.94	18	110.93	6	104.55	18	110.40	6	107.65	17	110.46	6	104.38	17	111.18	6
E. coli O104:H4	32x	27.01	15	28.11	17	23.94	15	27.55	17	24.48	15	28.81	17	22.51	15	27.76	16	23.44	15	28.55	16	**21.72**	**15**	**28.70**	**16**
P. falciparum 3D7	13x	46.10	15	50.40	21	46.50	15	50.38	20	**38.93**	**16**	**49.66**	**20**	–[1]		–[1]		–[1]		–[1]		–[1]		–[1]	
S. aureus	109x	**51.13**	**18**	**53.84**	**29**	48.43	18	55.05	26	48.87	18	54.83	26	48.36	18	54.47	24	48.37	18	53.43	26	44.37	18	54.31	26
S. aureus	34x	14.69	14	15.48	15	12.68	14	15.75	14	12.05	14	15.92	11	11.94	14	15.64	15	12.01	14	13.57	15	**12.01**	**14**	**13.59**	**15**
S. cerevisae	16x	28.96	15	29.42	15	28.58	15	30.02	16	28.95	15	29.68	15	**22.46**	**15**	**30.52**	**15**	–[1]		–[1]		–[1]		–[1]	

The values selected for HybridShrec* and HybridShrec^F are highlighted in bold. — [1] HybridShrec could not run for the given value of `-s`.

S10 NUMBER OF BASES REMOVED BY QUAKE

In Table S12, we report the number of removed bases and percentage of removed bases by Quake. Note that this is only due to clipped away bases. Any read removed by Quake as erroneous was inserted back from the original result.

Table S12. Number of bases in the data sets, bases in Quake's result, bases removed by Quake, and percentage of bases that Quake removed.

data set		original bases	Quake bases	Quake removed	Quake removed %
C. elegans	30x	2 995 360 535	2 602 871 767	392 488 768	13.10
D. melanogaster	5x	719 242 720	682 934 709	36 308 011	5.05
D. melanogaster	28x	3 983 042 294	3 870 735 864	112 306 430	2.82
E. coli	30x	139 190 322	135 027 481	4 162 841	2.99
E. coli	490x	2 294 730 100	2 226 186 200	68 543 900	2.99
P. syringae	41x	525 567 684	131 242 129	394 325 555	75.03
S. cerevisae	22x	276 971 196	276 903 093	68 103	0.02

REFERENCES

Döring, A. *et al.* (2008) SeqAn an efficient, generic C++ library for sequence analysis. *BMC Bioinf.,* **9**, 11.

Ilie, L. *et al.* (2011) HiTEC: accurate error correction in high-throughput sequencing data. *Bioinformatics,* **27** (3), 295–302.

Larsson, N. J. & Sadakane, K. (2007) Faster suffix sorting. *Theoretical Computer Science,* **387** (3), 258–272.

Li, H. & Durbin, R. (2010) Fast and accurate long-read alignment with burrows-wheeler transform. *Bioinformatics,* **26** (5), 589–595.

Myers, E. W. (1999) A fast bit-vector algorithm for approximate string matching based on dynamic programming. *J. ACM,* **46** (3), 395–415.

Navarro, G. & Baeza-Yates, R. (2000) A hybrid indexing method for approximate string matching. *J. Discrete Alg.,* **1** (1), 205–239.

Siragusa, E., Weese, D. & Reinert, K. (2013*a*) Fast and accurate read mapping with approximate seeds and multiple backtracking. *Nucleic Acids Res.,* **41** (7), e78.

Siragusa, E., Weese, D. & Reinert, K. (2013*b*) Scalable string similarity search/join with approximate seeds and multiple backtracking. In *Proceedings of the Joint EDBT/ICDT 2013 Workshops* EDBT '13 pp. 370–374 ACM, New York, NY, USA.

Weese, D., Holtgrewe, M. & Reinert, K. (2012) RazerS 3: faster, fully sensitive read mapping. *Bioinformatics,* **28** (20), 2592–2599.

Yang, X., *et al.* (2013) A survey of error-correction methods for next-generation sequencing. *Brief. Bioinform.,* **14** (1), 56–66.

Yang, X. *et al.* (2010) Reptile: representative tiling for short read error correction. *Bioinformatics,* **26** (20), 2526–2533.

Zhao, Z., Yin, J., Zhan, Y., Xiong, W., Li, Y. & Liu, F. (2011) PSAEC: an improved algorithm for short read error correction using partial suffix arrays. In *Proceedings of the 5th Joint International Frontiers in Algorithmics, and 7th International Conference on Algorithmic Aspects in Information and Management* FAW-AAIM'11 pp. 220–232 Springer-Verlag, Berlin, Heidelberg.