Briefings in Bioinformatics Advance Access published April 6, 2012

doi:10.1093/bib/bbs015

A survey of error-correction methods for next-generation sequencing

Xiao Yang, Sriram P. Chockalingam and Srinivas Aluru

Submitted: 23rd January 2012; Received (in revised form): 7th March 2012

Abstract

Error Correction is important for most next-generation sequencing applications because highly accurate sequenced reads will likely lead to higher quality results. Many techniques for error correction of sequencing data from next-gen platforms have been developed in the recent years. However, compared with the fast development of sequencing technologies, there is a lack of standardized evaluation procedure for different error-correction methods, making it difficult to assess their relative merits and demerits. In this article, we provide a comprehensive review of many error-correction methods, and establish a common set of benchmark data and evaluation criteria to provide a comparative assessment. We present experimental results on quality, run-time, memory usage and scalability of several error-correction methods. Apart from providing explicit recommendations useful to practitioners, the review serves to identify the current state of the art and promising directions for future research. Availability: All error-correction programs used in this article are downloaded from hosting websites. The evaluation tool kit is publicly available at: http://aluru-sun.ece.iastate.edu/doku.php?id=ecr.

Keywords: *error correction; next-generation sequencing; sequence analysis*

INTRODUCTION

Propelled by ever increasing throughput and decreasing costs, next-generation sequencing (NGS) is continually displacing traditional Sanger approaches in many applications such as resequencing, *de novo* sequencing, metagenomics and gene expression analysis. Many NGS technologies have been developed, including systems currently in wide use such as the Illumina Genome Analyzer and HiSeq platforms, as well as newer offerings from companies such as Ion Torrent and Pacific Biosciences [1]. The properties of major NGS platforms—454 LifeSciences [2], Illumina [3], Applied Biosystems SOLiD [4], Helicos HeliScope [5], Complete Genomics [6], Pacific BioSciences and Ion Torrent—are summarized in Table 1. These properties are current as of this writing (February 2012) and are expected to change frequently.

The growing prominence of NGS platforms and the myriad applications enabled by them have spurred significant research efforts in the development of bioinformatics methods for enabling such applications. Although all NGS experiments ultimately result in DNA sequence data, bioinformatics methods for analyzing the data can be very different based on the target application. For most of these methods, the quality of the results, and sometimes run-time performance or memory usage, can be improved by preprocessing the NGS data set to improve data quality. While this can be done by simple approaches such as trimming based on quality scores etc., which invariably result in loss of information, a class of sophisticated methods emerged that

Corresponding author. Xiao Yang, The Broad institute, 7 Cambridge Center, Cambridge, MA 02142, USA. Tel: +1-617-714-7919; Fax: +1-617-714-8932; E-mail: xiaoyang@broadinstitute.org

Xiao Yang is a Computational Biologist in genome sequencing and analysis program at Broad Institute. He developed two error-correction programs previously.

Sriram P. Chockalingam is a PhD student in Department Computer Science and Engineering, IIT Bombay. He developed a parallel algorithm for next-gen error correction.

Srinivas Aluru is the Mehl Professor of Computer Engineering at Iowa State University. He conducts research in computational genomics, systems biology and high performance computing. He is a Fellow of AAAS and IEEE.

Table 1: Characteristics of major NGS platforms as of February 2012

Company	Platform	Read length (bp)	Throughput & time per run	Technique	Dominant error type
Illumina	HiSeq 2000[a]	36 50 100	105–600 Gb 2–11 days	Reversible terminator	Substitution
Applied Biosystems	5500 SOLiD™ System[b]	356 075	7–9 Gb/day	Sequencing by ligation	–
Complete Genomics		35	–	Ligation based	–
Helicos BioSciences	HeliScope SMS[c]	25–55	21–35 Gb –	**Single molecule sequencing**	Insertion Deletion
454 Life Sciences	GS FLX Titanium XL+[d]	≤1000	700 Mb 23 h	Sequencing by **synthesis**	Insertion Deletion
Ion Torrent	Ion PGM Sequencer 318[e]	>200	>1 Gb 2 h	Ion semiconductor sequencing	Insertion Deletion
Pacific Biosciences	PacBio RS	1 k–10 k	–	**Single molecule sequencing**	Insertion Deletion

[a]HiSeq 2000: Performance and Specifications. http://www.illumina.com/systems/hiseq_2000/performance_specifications.ilmn (11 February 2012, date last accessed). [b]5500 SOLiD™ System: Specifications. http://media.invitrogen.com.edgesuite.net/solid/pdf/COI8235-5500-Series-Spec-Sheet-F2.pdf (11 February 2012, date last accessed). [c]tSMS™ Performance. http://www.helicosbio.com/Technology/TrueSingleMoleculeSequencing/tSMStradePerformance/tabid/151/Default.aspx (11 February 2012, date last accessed). [d]GS FLX Titanium XL+. http://my454.com/products/gs-flx-system/index.asp (11 February 2012, date last accessed). [e]Ion Personal Genome Machine™ Sequencer: Performance. http://www.iontorrent.com/technology-how-does-it-perform/ (11 February 2012, date last accessed).

detect and correct errors in NGS sequence data. There is growing interest in addressing NGS error correction, as evidenced by a million dollar challenge from Ion Torrent [7].

There are mainly three types of biases that lead to errors in NGS data—systematic bias, coverage bias and batch effects, depending on sequencing platform, genome content and experimental variability [8]. Some of these errors are inherent in the starting sequencing library as a result of PCR conditions, enrichment and sequence library preparation methods. In this manuscript, we focus on sequencing errors, which include substitutions, insertions and deletions. In addition, sequencers may output *N* to indicate the confirmed presence of a nucleotide that cannot be called accurately. While substitution errors are dominant in some platforms such as Illumina, in others such as 454 and Ion Torrent, homopolymer and carry-forward errors manifested as insertions and deletions are abundant. Due to the ubiquitous use of Illumina sequencing platforms, most error-correction algorithms so far have targeted substitution errors [9–21]. There is an increasing need to identify and correct insertion and deletion errors owing to the emergence of new platforms such as Ion Torrent; so far, only a few methods [10, 16, 17] can model such errors.

This article provides a review of error-correction algorithms for NGS data and their comparative evaluation. While the algorithms are documented in individual research papers, each work uses potentially different data sets for experimental evaluation, rendering cross-comparisons difficult. In addition, different papers use different metrics for evaluation, which clouds proper interpretation of results of multiple methods. A key contribution of this work is to establish a common set of benchmarks and evaluation metrics, and experimental evaluation of error-correction programs on these for providing clear information and explicit guidance for the practitioner. The programs are assessed on multiple dimensions including quality, run-time, memory usage, ability to scale and ability to deal with nuances of real data such as quality scores and ambiguous nucleotides.

OVERVIEW OF ERROR-CORRECTION METHODS

Error-correction methods designed so far have mainly targeted haplotype genome sequencing. In this setting, error correction with respect to a specific genomic position can be achieved by laying out all the reads covering the position, and examining the base in that specific position from all these reads. As errors are infrequent and random, reads that contain an error in a specific position can be corrected using the majority of the reads that have this base correctly. This general idea has been implemented in all error-correction algorithms, albeit indirectly. As the source genome is unknown, the reads from the same genomic location are inferred relying on the assumption that they typically share subreads of a fixed length, such as *k*-mers. Some methods [13, 17] further derive multiple sequence alignment (MSA) of reads that share common *k*-mers and seek corrections relying on the MSA, while others [9–12, 14–16, 18–21] correct errors at the level of *k*-mers or variable length subreads. In both cases, genomic repeats and

non-uniform sampling of genome may lead to multiple equally likely correction choices that lead to ambiguity in correction. We classify error-correction methods into three types—*k*-spectrum based, suffix tree/array-based and MSA-based methods.

k-spectrum-based

These methods work by decomposing the reads into the set of all *k*-mers present in them, termed the *k*-specturm [10]. In NGS data sets predominantly containing substitution errors, *k*-mers within a small Hamming distance from each other are likely to belong to the same genomic location. By identifying such a *k*-mer set, alignment is directly achieved without resorting to MSA, and error correction can then be applied by converting each constituent *k*-mer to the consensus. Mis-correction can occur if a *k*-mer set consists of elements sampled from different genomic locations. However, with a reasonably chosen *k* value, the overall correction is expected to be satisfactory.

The idea of correcting errors using *k*-spectrum was first proposed in [10, 22]. In a given data set, a *k*-mer occurring at least *M* times is termed *solid*, and is termed *insolid* otherwise. Reads containing *insolid* *k*-mers are converted to solid ones with a minimum number of edit operations so that they contain only solid *k*-mers post-correction. A dynamic programming solution for this, and a heuristic for scaling to larger data sets, is a built-in component in the short read assembler by Chaisson *et al.* [10]. A similar idea is also used as a stand-alone error-correction component in the SOAPdenovo assembler [23].

Under the same framework, Quake [14] in addition incorporates nucleotide specific miscall rates and works as follows: (i) identify solid and insolid *k*-mers—(a) calculate the weight of a *k*-mer as the weighted sum of all its instances, each given by $\prod_{i=0}^{k-1} p_i$, where p_i is the probability that the *i*-th base is correctly recorded, specified by its quality score; (b) model the weight histogram of solid *k*-mers as a mixture of Gaussian and Zeta distributions, and insolid *k*-mers as Gamma distribution, then choose a cut-off to differentiate them. (ii) Convert each read *r* to be free of insolid *k*-mers—(a) heuristically locate erroneous region in *r* using insolid *k*-mers. If some insolid *k*-mers cover the 3′-end, trimming is applied; (b) greedily correct bases with low quality scores until all *k*-mers are solid, if applicable. Otherwise, *r* is considered not fixable. Quake was applied only to simulated Illumina reads, which were generated with simplification that sequencing errors introduced conform to the probabilities specified by quality values, which is not necessarily accurate in real data sets [24].

Reptile [21] also incorporates quality scores and it further makes use of *k*-mer context to simultaneously improve specificity and sensitivity. Each read is decomposed into a sequence of overlapping substrings, termed tiles, which are corrected sequentially from 5′ to 3′ direction. Since errors are non-uniformly distributed in each read, sometimes it is necessary to explore alternative decomposition of a read in order to limit the maximum number of errors in a tile. Each tile consists of a sequence of (two in current implementation) non-overlapping or overlapping *k*-mers. This captures *k*-mer context information and helps in resolving ambiguities when a *k*-mer aligns equally well to different genomic locations. Given a tile *t* and its observed frequency in the data set, all other tiles that are similar to *t* but occurring with significantly higher frequency than *t* are considered as candidates to which *t* might be converted. To identify these candidates, Reptile searches a Hamming graph, where a node denotes a *k*-mer and an edge connects two nodes with Hamming distance $\leq d$. An efficient bucketing strategy was proposed to replace the direction construction of such a graph that can be memory demanding.

Hammer [25] extends the idea of Hamming graph to use spaced seeds to provide higher sensitivity in identifying clusters of similar *k*-mers. A consensus *k*-mer is generated to denote the error free *k*-mer for each cluster. This is equivalent to correct the counting of *k*-mers in the data set [15, 19, 20].

Suffix tree/array based

Suffix tree/array-based error-correction methods SHREC [18] and HiTEC [12] generalize the *k*-mer-based approach by handling multiple *k* values and the corresponding threshold *M*. The major differences between them include (i) how *M* is derived and (ii) how suffixes in the input are stored. Suffix arrays are more space-efficient than suffix trees, while permitting similar operations.

SHREC constructs a generalized suffix trie. For each internal node *u*, the concatenation of edge labels from the root to *u* spells a substring s_u that occurs in the input, and the number of times s_u occurs equals to the number of leaves of the subtree rooted at *u*. The expected occurrence of s_u is assumed to be an outcome of Bernoulli trials, and computed

analytically. Then, if the observed occurrence of s_u is smaller than α unit of variance, the last base of s_u is considered as an error. In order to identify the potential correction for s_u, the subtree rooted at u is compared with the subtree rooted at v, where u and v are siblings and s_v is error free. If the two subtrees are identical, then u can be merged to v and relevant changes are made for all reads that were previously leaves under u. Otherwise, there might be more than one error in a read containing s_u. Then, every read r under u is individually inspected. Multiple errors can be corrected by repeating the above procedure.

Built upon SHREC, Hybrid-SHREC [16] further captures insertion/deletion errors. In accordance with the above notation, if the last base of s_u is an insertion, then in the suffix trie, u should be compared with the siblings of the parent of u; on the other hand, if a deletion occurs, u should be compared with the children of its sibling. This extension is able to capture up to one insertion/deletion error in a read per given iteration.

HiTEC uses suffix array and is based on the following ideas: assume in read r, the substring $r[i, i+k]$ contains exactly one sequencing error at base $r[i+k]$. Meanwhile, if there exists a substring s such that $|s| = k+1$, $s\ [0,\ k-1] = r[i,\ i+k-1]$, $s[k] \neq r[i+k]$ and s occurs over M times in R, then $s[k]$ is likely to be the real base mis-recorded as $r[i+k]$. In other words, any erroneous base can be corrected by HiTEC only if this base is preceded by an error free k-mer. Thus, the goal is to derive proper values of k and M so that a satisfactory error-correction result can be achieved. HiTEC chooses k to minimize the total number of bases that cannot be corrected and the bases that can be potentially wrongly corrected. And similar to SHREC, M is chosen empirically based on the expected and observed k-mer occurrences.

MSA based

Coral [17] identifies reads colocated on the unknown reference genome by using k-mers as seeds. First, a hash table is created recording all the reads to which each k-mer belongs. Second, each read r is taken as the reference, then any read r' that shares at least one k-mer with r is identified and aligned with r using Needleman–Wunsch algorithm. A consensus sequence is created after alignment, which is taken as the new reference to be aligned with the remaining reads. This approach can be applied to both 454 reads and Illumina reads by controlling alignment penalties for edit operations. Insertion and deletion are considered in the former case, whereas in the latter case, only substitution is allowed. Finally, after MSA is generated, corrections are applied if the number of reads involved in the MSA is neither too large nor too small, and the largest edit distance between any constituent read and the consensus of the MSA is relatively small. These are controlled by user specified parameters.

Another extended MSA method was proposed in ECHO [13], which consists of two major stages: neighbor finding and maximum a posteriori error correction. In the first stage, read adjacencies with respect to the reference genome are identified in the same manner as the 'overlap' step in the traditional overlap-layout-consensus based Sanger assembler. But, ECHO considers only substitutions. First, both the input reads and their reverse complimentary strands are recorded in a k-mer hash table in the form of key-value pair: $\langle x,(r,p)\rangle$, where k-mer x starts at position p on read r. Similar to Coral, k-mers that occur with a high frequency in the data set are ignored. Next, for each k-mer x, for each pair of reads that contain x, pairwise alignment is applied and considered as valid only if the minimum overlap is $\geq \omega$ and the maximum error rate in the aligned region is $\leq \varepsilon$. The parameters k, ω and ε are chosen automatically: k is empirically set to be $\lfloor l/6 \rfloor$. To select ω and ε, it is assumed that reads are uniformly sampled from the reference genome and the coverage follows Poisson distribution. The threshold values are found using a grid search method within the parameter space of (ω, ε). For each selected pair of $(\omega^\star, \varepsilon^\star)$, neighbor finding method is applied to a subset of input data that are randomly sampled. Then the observed coverage histogram is generated for this partial data and is approximated by a closest Poisson distribution. The values of (ω, ε) are chosen such that the difference between the observed coverage and the corresponding Poisson distribution is minimized. In the second stage, a position dependent substitution matrix $\mathbf{M} = \{M_1, M_2 \ldots\}$ is estimated using an expectation maximization algorithm, where M_i is a 4×4 matrix recording substitution error rate between any two bases in {A, C, G, T} at position i. Then, for each alignment position derived from the first stage, the consensus base is calculated to be the maximum a posteriori estimate using the base information in the alignment column and the corresponding quality score information.

BENCHMARK DATA SETS AND EVALUATION METHODOLOGY

We selected data sets from real experiments instead of simulations, including a diversity of read length, genomic coverage and error rates. All data are from resequencing experiments where the genome sequence is known *a priori*, which is needed for computing the accuracy of error correction. Quality, run time and memory usage are measured for each method. By chronological order of their release dates, the programs evaluated are—Hybrid-SHREC (HSHREC for short) [16], Reptile [21] (version 1.1), Quake [14] (version 0.3), SOAPdenovo Corrector [23] (version 1.05, SOAPec for short), HiTEC [12], ECHO [13] (version 1.10) and Coral [17] (version 1.3).

Details of the chosen data sets are summarized in Table 2. Besides two bacterial genomes that were typically used in error-correction studies, we include one data set from *Saccharomyces cerevisiae* and one from *Drosophila melanogaster* since there is an increasing need to apply error correction to larger eukaryotic genomes. In addition to Illumina data used to assess all the programs, we include 454 and Ion Torrent data to evaluate HSHREC and Coral, which are the only programs capable of handling insertion and deletion errors. Notably, *D7* consists of reads of different lengths, which is currently a typical sequencing practice for large genomes.

Errors are identified by using a mapping program to align reads to the reference genome. Only uniquely mapped reads are considered, with errors given by differences with the reference genome. While eliminating unmapped reads and multiply mapped reads will bias results, quite likely by underestimating the error rate, it is not possible to include them in the analysis as the true bases remain unknown. Read mapping is an active area of research and many tools have been developed (e.g. [26–28]). We select the well regarded state of the art mapping program BWA [28] for aligning Illumina reads. To demonstrate mapping program bias is negligible, we also report results using RMAP [26] for data set *D3*. For 454 (*D4*) and Ion Torrent (*D8*) reads, we use Mosaik [29] and TMAP [30], respectively, as each is designed specifically for the underlying platform.

Using the above strategy, we first characterize the errors present in the benchmark data sets (Table 3). Illumina read mapping is carried out using BWA, which is run with the default parameters for indexing the reference genome (index -a bwtsw). The maximum allowable edit distance (parameter '-n') is set to be 4 for *D1*, 10 for *D2*, and 6 for the remaining data sets. This is to account for differences in read lengths. TMAP performs alignment in two stages, using different algorithms in the second stage for aligning reads left over in the first stage. We ran TMAP with the option 'mapall -g 3 map1 MAP2 MAP3', enabling it to align using BWA [31] in the first stage, and BWA-SW [28], SSAHA [32] in the second stage. The other parameters are set to default. We postpone the discussion of parameter selection for Mosaik until later (Table 4).

Table 2: Sequence data sets

Data set	Sequence read archive accession number	Platform	Reference genome	Genome length	NCBI reference sequence accession number	Read length (bp)	Number of reads
D1	SRX000429	Illumina	*E. coli*	4 639 675	NC_000913	2 × 36	20 816 448
D2	ERA000206	Illumina	*E. coli*	4 639 675	NC_000913	2 × 100	28 428 648
D3	SRR022918	Illumina	*E. coli*	4 639 675	NC_000913	2 × 47	14 275 243
D4	SRR000868	454	*E. coli*	4 639 675	NC_000913	37–385	230 517
D5	SRX100885	*Illumina*	*S. cerevisiae*	12 071 326	PRJNA128	2 × 76	52 061 664
D6	SRR022866	Illumina	*S. aureus*	2 820 462	NC_003923	2 × 76	25 169 557
D7	SRX023452 SRX006152 SRX006151 SRX015867	Illumina	*D. melanogaster*	119 029 979	–	2 × {45 75 95}	98 391 467
D8	ERR039477	Ion Torrent	*E. coli*	4 639 675	NC_000913	16–107	390 975

The sixth column specifies the corresponding reference genomes that are obtained from NCBI website except for *D7*, which is obtained from BCM Drosophila database (http://www.hgsc.bcm.tmc.edu/project-species-i-DGRP_lines.hgsc), line RAL-391 (freeze1 July 2010). This genome consists of five chromosomes (Line391_Chr2L, Line391_Chr2R, Line391_Chr3L, Line391_Chr3R and Line391_ChrX), the concatenation of which forms the reference genome. The *S. cerevisiae* reference genome of *D5* is a concatenation of 16 chromosomes.

Table 3: Mapping statistics

Data set	Number of reads	No. of reads containing *N*s	No. multi-mapped & unmappable reads	Coverage	Error rate before correction (%)
D1	20 816 448	107 740	795 924	160.7	0.51
D2	28 428 648	1 795 045	1 398 918	574	1.01
D3	14 275 243	7 155 638	3 248 669	72.1	1.54
D4	230 517	8141	4575	11.6	0.26
D5	52 061 664	1 455 543	11 759 905	244	0.68
D6	25 169 557	1 036 879	10 320 864	650.3	1.55
D7	98 391 467	2 716 595	22 660 379	56.3	1.15
D8	390 975	0	16 422	7.8	1.28

The number of reads that contain one or more *N*s is shown in the third column. Since such reads cannot be handled by some methods such as HiTEC, they are removed prior to evaluation and for coverage calculation (column five). The fourth column specifies the number of reads that are mapped to multiple locations or those that cannot be mapped within a specified edit distance.

Table 4: Alignment results of *D4* using different penalty values of Mosaik

Experiment	Mismatch (% length)	*mr*	*mp*	*go*	*ge*	Number of soft-clipped bases	Number of unmapped reads	Error rate before correction (%)
E1	5	10	9	15	6.6	51 256	3211	0.30
E2 (default)	15	10	9	15	6.6	60 700	1641	0.33
E3	2	10	9	10	6.6	9259	10 982	0.26
E4	3	10	9	8	6.6	8931	10 955	0.27
E5	5	10	9	8	6.6	11 940	3076	0.35

mr, match reward; *mp*, mismatch penalty; *go*, gap open penalty; *ge*, gap extension penalty

We use the following measures for each program—number of erroneous bases identified and successfully corrected (true positives, TP), correct bases wrongly identified as errors and changed (false positives, FP), and erroneous bases that were either uncorrected or falsely corrected (false negatives, FN). We report sensitivity and specificity for each program. Then, we combine these into the gain metric [21], defined by $\text{gain} = (\text{TP} - \text{FP})/(\text{TP} + \text{FN})$, which is the percentage of errors removed from the data set by the error-correction program. A negative gain value indicates that more errors have been introduced due to false corrections, which is not captured by measures such as sensitivity and specificity. We disregarded other measures used in the literature if they cannot be directly tied to accuracy improvements. For instance, [16] measures whether the percentage of reads that can be uniquely aligned to the reference genome increased after applying error correction. However, converting every ambiguously mapped read to one specific unrelated read known with high confidence will always guarantee the best results. Another measure used is to compare N50 values from assembly program by assembling post-error-correction data versus assembling the original data set [17]. While this is useful to show assemblers benefit from error-correction program, this cannot be used to determine accuracy of the error-correction program because of interference from built in error-correction component of assembly program.

We denote an error e in a read by a 4-tuple (g,i,t,w) where g is the genomic position of e, t and w are the true and wrong bases at g, respectively, and i indicates e represents i-th inserted base after position g. Therefore, for a substitution error, t, $w \in \{\text{A,C,G,T}\}$ and $i = 0$; for a deletion error, $t \in \{\text{A,C,G,T}\}$, $w = -$ denoting a missing base, and $i = 0$; for an insertion error, $t = -$, $w \in \{\text{A,C,G,T}\}$, and $i \neq 0$. An attempted error correction can also be described by the same 4-tuple, with the interpretation that base w in the read is being replaced with base t.

To calculate gain, we need the set of real sequencing errors E_m and the set of attempted corrections E_c made for each read r using any error-correction program. Alas, the former can only be approximated by read mapping, and the latter is usually untraceable because most programs only provide corrected

reads as output. We thus infer *TP*, *FP* and *FN* as follows:

Case 1: only substitution errors are targeted for correction. Let r be an original read and r_c be the read post-correction. Derive E_m by mapping r to the reference genome and recording differences. E_c can be derived from bases that differ when calculating Hamming distance between r and r_c. Then, calculate $TP = |\ E_c \cap E_m\ |$, $FP = |\ E_c \setminus E_m\ |$, $FN = |\ E_m \setminus E_c\ |$.

Case 2: Indel errors are also targeted for correction. E_m is calculated as before but the method to calculate E_c in case 1 is no longer valid, even in the case where $|r_c| = |r|$. For example, let $r =$ TTTAAT TCAGGTAT and $r_c =$ TTAATTCAGGTATT with a Hamming distance of 8 between them. We over-counted *FP* by 6 if the true E_c consists of an insertion and a deletion error at the beginning and the end of the read, respectively, as shown below:

r TTTAATTCAGGTAT -
r_c - TTAATTCAGGTATT

Hence, instead of directly calculating E_c, we measure E_r, the set of errors remaining in r_c, by applying global alignment between r_c and the genomic region where r was mapped to, and recording the differences in the alignment. Accordingly, calculate $TP = |\ E_m \setminus E_r\ |$, $FP = |\ E_r \setminus E_m\ |$ and $FN = |\ E_r \cap E_m\ |$. In practice, we used banded alignment to speed up the process, because it gives a comparable results compared with the global alignment but is significantly faster.

RESULTS AND DISCUSSION

All experiments were carried out on an HP ProLiant DL580G7 server, which has four quad-core Intel Xeon E7520 (1.8 GHz) processors, 128 GB main memory and disk capacity of 370 GB. The server is running 64-bit Ubuntu 11.04 OS. Parameters for each program were chosen based on guidance provided by the developers. HSHREC was run with default parameters since instructions for choosing parameters is not given. For Reptile, the parameters (k, step) are set to be (11, 11) for bacterial genomes, (12, 12) for *S. cerevisiae* and (14, 10) for *D. melanogaster*, where $4^k \geq$ genome length. In *D5* and *D7*, 'QThreshold' is set to be the minimum quality value in the data set, since it is unclear how to choose a proper value for this data set. The remaining parameters are set by default as per the instructions in the *README* file. Quake is run with $k = 15$ for *Escherichia coli* and $k = 17$ for *D. melanogaster*, as suggested by the program manual. HiTEC 64-bit version was used for all the experiments since the 32-bit version was not able to run on some of the data. SOAPec is run in two steps: (i) 'KmerFreq –i lst –o freq –f 2', where lst is a file listing the names of fasta files, and freq is the output file recording 17-mer frequency, and (ii) 'Corrector –i lst –r freq –f 2'. Two parameters are required to be set manually for HiTEC: reference genome length and error rate. The former was set to be the real reference genome length according to Table 2, and the latter was set to be 2%, an upper bound of the average error rate in Table 3. Other parameters are selected automatically by the program. ECHO was run with the default parameters since it has a mechanism for automatically choosing parameters. Coral also has a mechanism to select parameters automatically, therefore, it was run with default parameters by setting sequencing platforms to '-illumina' for Illumina data sets, and '−454' for 454 and Ion Torrent data sets.

Error-correction results for Illumia data sets are summarized in Table 5, and for 454 and Ion Torrent data sets in Table 6. For Illumina data sets, HSHREC, Reptile, SOAPec and Coral were able to generate complete results. ECHO produced a significant amount of intermediate data and failed to yield any results for *D1* after running for over two days. Similar behavior was observed for *D2*, *D5* and *D6*. However, for data set *D7*, ECHO was running for more than one day but the job had to be killed due to large size of temporary files it generated (over 130 GB) saturated the hard drive. Quake failed on *D2*, *D5* and *D6*, with an error message indicating that the data set has insufficient coverage over the reference genome. Nonetheless, all three data sets have ultra-high coverage. HiTEC failed with 'segmentation fault' error for data sets *D5* and *D6*. In addition, it was not able to run on *D7*, where the reads have different length. For 454 and Ion Torrent data sets, both HSHREC and Coral generate complete results. For each data set, the best result of different methods is shown in bold for 'Gain', 'Run-time' and 'Memory' values. In the following, we discuss the results on Illumina data sets separately from 454 and Ion Torrent data sets.

Table 5: Experimental results for Illumina data sets

Data set	Method	Specificity	Sensitivity	Gain	Runtime (min)	Memory (GB)	Error rate after correction (%)
D1	HSHREC	0.9913	0.6262	−2.8864	153.23	14.3	0.97
	Reptile	0.9999	0.9341	**0.9334**	**23.32**	**3.7**	0.03
	Quake	0.9998	0.8167	0.7840	38.88	4.1	0.11
	SOAPec	0.9989	0.8099	0.5982	25.51	21.1	0.20
	HiTEC	0.9996	0.9299	0.8467	143.13	13.0	0.08
	ECHO	–	–	–	–	–	–
	Coral	0.9999	0.5835	0.5794	36.19	7.5	0.21
D2	HSHREC	0.9930	0.2852	−1.6063	779.15	29.9	0.96
	Reptile	0.9999	0.9137	0.9101	225.43	19.1	0.09
	SOAPec	0.9991	0.3380	0.2512	**124.30**	23.3	0.75
	Quake; ECHO	–	–	–	–	–	–
	HiTEC	0.9999	0.9557	**0.9498**	683.87	**9.8**	0.05
	Coral	0.9999	0.1134	0.1127	450.29	30.0	0.90
D3	HSHREC (B)	0.9866	0.6355	−0.3166	74.13	12.6	1.83
	HSHREC (R)	0.9860	0.6256	−0.2690			1.96
	Reptile (B)	0.9999	0.8535	0.8521	71.64	3.4	0.29
	Reptile (R)	0.9999	0.7927	0.7910			0.45
	SOAPec (B)	0.9958	0.5861	0.3667	24.08	19.9	1.23
	SOAPec (R)	0.9955	0.5447	0.3436			1.41
	Quake (B)	0.9990	0.3648	0.3134	80.43	2.0	1.34
	Quake (R)	0.9990	0.332	0.2856			1.54
	HiTEC (B)	0.9997	0.9466	**0.9291**	59.19	**6.2**	0.14
	HiTEC (R)	0.9995	0.9345	**0.9122**			0.19
	ECHO (B)	0.9999	0.9091	0.9076	304.21	16.0	0.18
	ECHO (R)	0.9999	0.8986	0.8973			0.22
	Coral (B)	0.9999	0.0003	0.0029	**8.32**	7.7	1.94
	Coral (R)	0.9999	0.0003	0.0027			2.15
D5	HSHREC	0.9773	0.3635	−2.4975	1027.48	30.1	2.75
	Reptile	0.9999	0.2278	**0.2243**	165.88	**4.12**	0.61
	SOAPec	0.9995	0.1871	0.1257	**135.51**	26.8	0.69
	Quake; HiTEC; ECHO	–	–	–	–	–	–
	Coral	0.9999	0.0709	0.0678	544.31	34.5	0.73
D6	HSHREC	0.9688	0.2248	−4.2402	813.65	30.2	3.64
	Reptile	0.9999	0.6194	**0.6131**	295.94	**13.1**	0.60
	SOAPec	0.9989	0.3216	0.2571	**153.67**	22.5	1.15
	Quake; HiTEC; ECHO	–	–	–	–	–	–
	Coral	0.9999	0.0276	0.0256	210.17	40.0	1.51
D7	HSHREC	0.9258	0.5702	−5.8509	2562.88	30.3	7.82
	Reptile	0.9999	0.6745	**0.6678**	532.76	24.0	0.38
	SOAPec	0.9983	0.5225	0.3779	415.03	36.8	0.72
	Quake	0.9939	0.3993	−0.1262	**222.35**	**12.1**	1.30
	HiTEC; ECHO	–	–	–	–	–	–
	Coral	0.9999	0.4532	0.4492	373.19	30.0	0.63

If a method fails to process a particular data set, corresponding entries are denoted by '–'s. All data sets are evaluated against BWA alignments. For *D3*, (B)—Compared with alignments from BWA (R)—Compared with alignments from RMAP.

Illumina

Only four programs—HSHREC, Reptile, SOAPec and Coral—succeeded in generating results for all data sets. Reptile consistently produced superior gain which is generally a factor of two better than SOAPec, which itself has better gain values compared with the other two. SOAPec often uses significantly more memory than Reptile. Among those data sets where results can be generated for all methods under comparison, Reptile, HiTEC and ECHO yield comparable gain values that are superior to other methods. HiTEC and ECHO have automated parameter selection for many of the program parameters, compared with the manual default parameter selection mechanism in Reptile. Reptile, on the other hand, has a better scalability to larger data sets. An improvement could be achieved by including in Reptile an automated parameter selection method such as those used in HiTEC or ECHO. The current version of HiTEC does not handle '*N*'

Table 6: Experimental results for 454/ion torrent data sets

Data set	Method	Specificity	Sensitivity	Gain	Runtime (min)	Memory (GB)	Error rate after correction (%)
D4 (k = 10)							
D4/E1	HSHREC	0.9983	0.9063	0.2937	16.72	9.9	0.20
	Coral	0.9986	0.9732	**0.4976**	**2.57**	**2.4**	0.15
D4/E2	HSHREC	0.9979	0.8730	0.2242	16.72	9.9	0.25
	Coral	0.9983	0.9432	**0.4252**	**2.57**	**2.4**	0.19
D4/E3	HSHREC	0.9989	0.9137	0.4709	16.72	9.9	0.13
	Coral	0.9993	0.9520	**0.6984**	**2.57**	**2.4**	0.08
D4/E4	HSHREC	0.9989	0.9130	0.4737	16.72	9.9	0.13
	Coral	0.9993	0.9520	**0.7029**	**2.57**	**2.4**	0.08
D4/E5	HSHREC	0.9986	0.8689	0.4511	16.72	9.9	0.19
	Coral	0.9992	0.9512	**0.7346**	**2.57**	**2.4**	0.09
D4 (k = 20)							
D4/E1	HSHREC	0.9983	0.9063	0.2931	16.72	9.9	0.20
	Coral	0.9986	0.9732	**0.4921**	**2.57**	**2.4**	0.15
D4/E2	HSHREC	0.9978	0.8736	0.1859	16.72	9.9	0.26
	Coral	0.9981	0.9437	**0.3716**	**2.57**	**2.4**	0.21
D4/E3	HSHREC	0.9989	0.9137	0.4709	16.72	9.9	0.13
	Coral	0.9993	0.9520	**0.6985**	**2.57**	**2.4**	0.08
D4/E4	HSHREC	0.9989	0.9130	0.4736	16.72	9.9	0.13
	Coral	0.9993	0.9520	**0.7029**	**2.57**	**2.4**	0.08
D4/E5	HSHREC	0.9986	0.8689	0.4511	16.72	9.9	0.19
	Coral	0.9992	0.9512	**0.7330**	**2.57**	**2.4**	0.10
D8							
D8/with all mapped reads	HSHREC	0.9959	0.7567	0.3999	8.72	10.21	0.68
	Coral	0.9971	0.7361	**0.5155**	**1.13**	**2.24**	0.62
D8/excluding reads with >10 errors	HSHREC	0.9968	0.7763	0.4502	8.72	10.21	0.53
	Coral	0.9982	0.7706	**0.6047**	**1.13**	**2.24**	0.43

characters that frequently occur in NGS data nor does it handle data sets with different read length. To overcome these limitations, the use of the suffix array data-structure in HiTEC needs to be improved. ECHO, on the other hand, relies on MSA for error correction, which achieves high gain values, but with a trade-off of large run-time and memory usage. Although Coral, also a MSA-based method, has much less run-time and memory usage compared with ECHO, its performance is significantly worse compared with ECHO when applied to Illumina reads. It is worthwhile to note that Quake utilizes an aggressive strategy to trim or discard reads, which may lead to a significant percentage of data loss. For instance, in *D3*, it discards 3 126 742 reads and trimmed another 200 117 reads to a shorter length. Although if we disregard all these reads, the gain value improves significantly to 0.8.

As an illustration that different read mapping tools can produce a consistent view of error-correction program, we also used RMAP in our evaluation for *D3*. Indeed, RMAP produced different results from BWA, but the relative gain values of different methods remained the same.

454 and Ion Torrent

Mappers often perform soft clipping, which trims ends of the reads containing low quality bases. With default parameters, Mosaik soft-clipped 60 700 bases, which is 30% of the total number of errors, thus excluding a large portion of the reads. For evaluation of error correction, we aim to have alignments that reduce both the number of soft-clipped bases and unmapped reads, without distorting the error rate of the data set. Mosaik's default parameters are such that the gap-opening penalty is much higher than the mismatch penalty. Since 454 predominantly makes insertions/deletion errors, allowing the gap open penalty closer to mismatch penalty could avoid alignments with too many mismatches. We tuned down the maximum mismatch value, which by default equals to 15% of the read length. We conducted experiments varying different alignment parameters—% read length of mismatch allowed, match reward, mismatch penalty, gap opening penalty and gap extension penalty. The number of soft-clipped bases, unmapped reads and the error rate for different choices of parameters are shown in

Table 4, and the corresponding results are shown in Table 6. Evaluation with two different alignment bands are shown as $k = 10$ and $k = 20$ in the table. For the Ion Torrent data set *D8*, TMAP is able to produce alignments with no soft clipping, but the side effect of such a mapping is that some of the alignments have larger edit distances. For *D8*, TMAP generated an alignment of maximum edit distance of 25, and 5603 reads were aligned with edit distance >10. Table 6 shows the results for *D8* for the cases where we include all mapped reads, and when we ignore the reads with aligned edit distances >10 to adjust for excessive corrections. The performance of Coral is consistently better than HSHREC in both data sets tested. However, neither of these methods explicitly models homopolymer or carry-forward errors in such types of data.

In summary, for Illumina read correction, Reptile, HiTEC and ECHO are generally more accurate compared with other methods. Parameter selection in HiTEC and ECHO involves less manual work compared with Reptile. HiTEC is limited by its inability to handle variable read lengths, and ambiguous bases in the reads recorded as '*N*'. Compared with ECHO, HiTEC and Reptile have a better scalability to larger genomes. Currently, HSHREC and Coral are the only programs that can handle insertion/deletion errors, with Coral providing better results among the two. However, their performance is not on par with substitution-error-based error-correction methods and further work is needed.

CONCLUSIONS AND FUTURE DIRECTIONS

Owing to the predominant use of Illumina sequencers, most error-correction programs to date have focused only on substitution errors due to the resulting computational advantages. Although 454 sequencers have been around for a long time, their long reads and low error rates make the use of error-correction program much less important. This has changed with the introduction of Ion Torrent, which produces short reads and predominantly makes insertion and deletion errors. Current methods that handle indels do not achieve as good results as those targeting Illumina, and there is need for improved algorithms here.

Further research on error-correction algorithms is needed due to many current deficiencies: automated choice of parameters sensitive to data set being processed is important to avoid the user inadvertently choosing wrong parameters. Existing stand-alone error-correction programs target haplotype genome sequencing. Differentiating polymorphisms from sequencing errors, especially when polymorphism rate is comparable to error rate, is a challenging problem; also, extending error correction to non-uniformly sampled data sets and metagenomic data sets are important. None of the current methods handle paired read information, which can be useful to error correct reads that come from repeat regions. There is need for improving run-time and memory footprint of the algorithms in light of the factor of 10 throughput advances per year that propelled some NGS systems into the range of billions of reads per experiment. A very important unaddressed problem is correcting hybrid data sets of sequences derived from multiple platforms.

Key Points

- Error correction to improve sequence accuracy is important to many applications of NGS.
- This review paper provides a comprehensive assessment of many error correction algorithms using a common set of benchmarks and assessment criteria to enable practitioners to choose the right method targeted to their application needs.
- Further progress is needed to handle large genomes and larger datasets, to handle insertion and deletion errors, to correct hybrid datasets from multiple next generation platforms, and to develop error correction methods for datasets in population studies.

Acknowledgements

The authors thank the three anonymous reviewers for comments that helped in improving the manuscript, and also Ankita Mawandia for processing a subset of data.

FUNDING

This work is supported in part by the US National Science Foundation under DMS-1120597, and a Department of Science and Technology Swarnajayanti Fellowship from the Government of India.

References

1. Korlach J, Bjornson KP, Chaudhuri BP, *et al*. Real-time DNA sequencing from single polymerase molecules. *Methods Enzymol* 2010;**472**:431–55.
2. Margulies M, Egholm M, Altman WE, *et al*. Genome sequencing in microfabricated high-density picolitre reactors. *Nature* 2005;**437**:376–80.

3. Bentley DR, Balasubramanian S, Swerdlow HP, *et al*. Accurate whole human genome sequencing using reversible terminator chemistry. *Nature* 2008;**456**:53–9.
4. McKernan KJ, Peckham HE, Costa GL, *et al*. Sequence and structural variation in a human genome uncovered by short-read, massively parallel ligation sequencing using two-base encoding. *Genome Res* 2009;**19**:152741.
5. Shendure J, Ji H. Next-generation DNA sequencing. *Nat Biotechnol* 2008;**26**:1135–45.
6. Drmanac R, Sparks AB, Callow MJ, *et al*. Human genome sequencing using unchained base reads on self-assembling DNA nanoarrays. *Science* 2010;**327**:78–81.
7. Challenges LG. http://www.lifetechnologies.com/global/en/home/new-ideas/grand-challenges.html (9 December 2011, date last accessed).
8. Taub MA, Corrada Bravo H, Irizarry RA. Overcoming bias and systematic errors in next generation sequencing data. *Genome Med* 2010;**2**:87.
9. Chaisson MJ, Brinza D, Pevzner PA. De novo fragment assembly with short mate-paired reads: does the read length matter? *Genome Res* 2009;**19**:336–46.
10. Chaisson M, Pevzner P, Tang H. Fragment assembly with short reads. *Bioinformatics* 2004;**20**:2067–74.
11. Chin FYL, Leung HCM, Li W-L, *et al*. Finding optimal threshold for correction error reads in DNA assembling. *BMC Bioinformatics* 2009;**10(Suppl 1)**:S15.
12. Ilie L, Fazayeli F, Ilie S. HiTEC: accurate error correction in high-throughput sequencing data. *Bioinformatics* 2011;**27**:295–302.
13. Kao W-C, Chan AH, Song YS. ECHO: a reference-free short-read error correction algorithm. *Genome Res* 2011;**21**:1181–92.
14. Kelley DR, Schatz MC, Salzberg SL. Quake: quality-aware detection and correction of sequencing errors. *Genome Biol* 2010;**11**:R116.
15. Qu W, Hashimoto S-I, Morishita S. Efficient frequency-based de novo short-read clustering for error trimming in next-generation sequencing. *Genome Res* 2009;**19**:1309–15.
16. Salmela L. Correction of sequencing errors in a mixed set of reads. *Bioinformatics* 2010;**26**:1284–90.
17. Salmela L, Schroder J. Correcting errors in short reads by multiple alignments. *Bioinformatics* 2011;**27**:1455–61.
18. Schroder J, Schroder H, Puglisi SJ, *et al*. SHREC: a short-read error correction method. *Bioinformatics* 2009;**25**:2157–63.
19. Wijaya E, Frith MC, Suzuki Y, *et al*. Recount: expectation maximization based error correction tool for next generation sequencing data. *Genome Inform Int Conf Genome Inform* 2009;**23**:189–201.
20. Yang X, Aluru S, Dorman KS. Repeat-aware modeling and correction of short read errors. *BMC Bioinformatics* 2011;**12(Suppl 1)**:S52.
21. Yang X, Dorman KS, Aluru S. Reptile: representative tiling for short read error correction. *Bioinformatics* 2010;**26**:2526–33.
22. Pevzner PA, Tang H, Waterman MS. An Eulerian path approach to DNA fragment assembly. *Proc Natl Acad Sci USA* 2001;**98**:9748–53.
23. Li R, Zhu H, Ruan J, *et al*. De novo assembly of human genomes with massively parallel short read sequencing. *Genome Res* 2010;**20**:265–72.
24. Dohm JC, Lottaz C, Borodina T, *et al*. Substantial biases in ultra-short read data sets from high-throughput DNA sequencing. *Nucleic Acids Res* 2008;**36**:e105.
25. Medvedev P, Scott E, Kakaradov B, *et al*. Error correction of high-throughput sequencing datasets with non-uniform coverage. *Bioinformatics* 2011;**27**:i137–41.
26. Smith AD, Chung W-Y, Hodges E, *et al*. Updates to the RMAP short-read mapping software. *Bioinformatics* 2009;**25**:2841–2.
27. Langmead B, Trapnell C, Pop M, *et al*. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biol* 2009;**10**:R25.
28. Li H, Durbin R. Fast and accurate long-read alignment with Burrows-Wheeler transform. *Bioinformatics* 2010;**26**:589–95.
29. Mosaik Assembler Program. http://bioinformatics.bc.edu/marthlab/Mosaik (9 December 2011, date last accessed).
30. TMAP: The Torrent Mapping Alignment Program for Ion Torrent. http://lifetech-it.hosted.jivesoftware.com/docs/DOC-1487 (9 December 2011, date last accessed).
31. Li H, Durbin R. Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics* 2009;**25**:1754–60.
32. Ning Z, Cox AJ, Mullikin JC. SSAHA: a fast search method for large DNA databases. *Genome Res* 2001;**11**:1725–9.