Correcting Illumina data

Michael Molnar and Lucian Ilie

Submitted: 23rd June 2014; Received (in revised form): 2nd August 2014

Abstract

Next-generation sequencing technologies revolutionized the ways in which genetic information is obtained and have opened the door for many essential applications in biomedical sciences. Hundreds of gigabytes of data are being produced, and all applications are affected by the errors in the data. Many programs have been designed to correct these errors, most of them targeting the data produced by the dominant technology of Illumina. We present a thorough comparison of these programs. Both HiSeq and MiSeq types of Illumina data are analyzed, and correcting performance is evaluated as the gain in depth and breadth of coverage, as given by correct reads and k-mers. Time and memory requirements, scalability and parallelism are considered as well. Practical guidelines are provided for the effective use of these tools. We also evaluate the efficiency of the current state-of-the-art programs for correcting Illumina data and provide research directions for further improvement.

Keywords: *DNA sequencing; Illumina data; error correction; coverage depth; coverage breadth*

INTRODUCTION

Research in biomedical sciences has been revolutionized by the possibility to sequence the molecule of DNA. Great demand for rapid and affordable DNA sequencing caused the replacement of the Sanger sequencing method [1] by improved technologies that can produce huge amounts of data at ever decreasing costs. Fierce competition produced a number of sequencing technologies, including Illumina/Solexa, Roche 454, AB SOLiD, Ion Torent, Pacific Biosciences, etc.; see [2] for details.

These high-throughput technologies enable many fundamental applications, including *de novo* genome assembly, genome resequencing, cancer mutation discovery, metagenomics, DNA–protein interaction discovery, personalized medicine, etc. As a result, highly ambitious projects have been started, such as the Genome 10 K Project [3] (www.genome10k.org), whose goal is to obtain the genomes of 10 000 vertebrate species, the 1000 Genomes Project [4] (www.1000genomes.org), which proposes to obtain the genomes of 1000 genetically varying humans, and the Human Microbiome project [5] (commonfund.nih.gov/Hmp), whose aim is to characterize the microbial communities found at several different sites on the human body.

The technology from Illumina is clearly dominating the market, and the datasets produce by the Illumina machines are the focus of this work. Most of the Illumina datasets are currently produced by two machines with different output. The HiSeq machines are designed for high output (up to 1000 GB) to be produced in a longer period (up to 6 days), with paired reads of length up to 125 bp (typically 2×100 bp). The MiSeq machines, on the other hand, are much smaller and cheaper (benchtop platforms) and produce much lower output (up to 15 GB) in a short period (5–55 h) but with longer reads of up to 300 bp (typically 2×250 bp; source: www.illumina.com/systems/sequencing.ilmn).

Both HiSeq and MiSeq datasets have a non-negligible number of errors (see Datasets Section for details), and effective correction of errors is crucial for improving the quality of the data. Therefore, a large number of error-correcting programs have been designed, including Euler [6], SHREC [7], Recount [8], Reptile [9], Quake [10], CUDA [11], HSHREC [12], SOAP [13], HiTEC [14], Coral [15], DecGPU [16], Hammer [17], ECHO [18], PSAEC [19], MyHybrid [20], SGA [21], RACER [22], Musket [23] and BLESS [24]; see also the survey of Yang *et al.* [25]. In addition, many

Corresponding author: Lucian Ilie, Department of Computer Science, University of Western Ontario, London, ON, N6A 5B7, Canada. Tel: +1 519 661 2111 ext. 86848; Fax: +1 519 661 3515; E-mail: ilie@csd.uwo.ca
Michael Molnar is a PhD student in Bioinformatics in the Department of Computer Science, University of Western Ontario, Canada.
Lucian Ilie, PhD, is a full professor in the Department of Computer Science, University of Western Ontario, Canada. His research interests include bioinformatics, algorithms, and stringology.

genome assemblers include their own correction programs.

With so many programs available, it becomes difficult for users to know which ones to use and when. We thoroughly evaluate these programs and provide guidelines for their effective use, as well as point out areas of improvement for further research.

Because of the clear domination of the Illumina platforms, all programs are targeting the correction of errors produced by these machines, which consist essentially of substitutions, that is, the correct base has been erroneously replaced by a different one. Few programs also provide mechanisms to correct indel errors, that is, erroneous insertions or deletions of bases.

The existing evaluation methods in the literature are either incomplete or inadequate, as explained in detail in the Evaluation Methodology section. We use several measures of correctness that provide a comprehensive picture for the performance of the correcting programs. A dataset has two main parameters that can be improved by correction: 'depth of coverage' (the number of times each base is covered on the average) and 'breadth of coverage' (the proportion of the genome that is covered). Also, the reads of a dataset can be used as 'whole reads' (for overlap graph-based assembly [21]) or broken into 'k-mers' (i.e. subsequences of length k, used in deBruijn graph-based assembly [13, 26–29]). Evaluating the gain with respect to each type of coverage using either whole reads or k-mers gives our four measures of correctness: ReadDepthGain, KmerDepthGain, ReadBreadthGain and KmerBreadthGain. Precise definitions of these measures are given in the Evaluation Methodology section.

We use a variety of datasets, both HiSeq and MiSeq, from reference genomes ranging from bacteria to humans. All our datasets are real and have not been altered in any way. They provide not only a good assessment of the actual performance of the correcting programs but also a clear indication of the current state of the art in correcting Illumina data.

We compare seven programs, BLESS, Coral, HiTEC, Musket, RACER, SGA and SHREC, that have performed the best in recent studies [21–24]. We show that there is no single winner. BLESS, Musket, RACER and SGA perform the best overall, each having its own advantages and disadvantages. In spite of much research in this area, significant room for improvement remains with respect to both depth and breadth of coverage. In particular, all programs actually decrease the breadth of coverage by correct k-mers, that is, the KmerBreadthGain is always negative.

Our study is organized as follows. In order, we present the programs to be compared, describe in detail the evaluation methodology, introduce our selection of datasets used for comparison, present and discuss the results of the comparison and conclude with recommendations and directions for future research.

ERROR CORRECTION PROGRAMS

The programs for error correction can be divided into three main categories by their approach [25]. Many programs are based on the k-mer spectrum approach [30], where reads presumed erroneous are changed so that all their k-mers appear frequently in the dataset; these include BLESS, CUDA, DecGPU, Euler, Hammer, Musket, Quake, Recount and Reptile. A second approach relies on counting k-mers, with the assumption that correct ones are much more frequent than those containing errors; SHREC, HiTEC, HSHREC, PSAEC and RACER belong here. The third approach uses multiple sequence alignment, as used by Coral, ECHO and MyHybrid. We refer the reader to the survey of [25] for details.

In our study, we have selected for comparison those programs that performed the best in recent studies: BLESS, Coral, HiTEC, Musket, RACER, SGA and SHREC. All three paradigms are covered by our choice: BLESS, Musket and SGA use k-mer spectrum, HiTEC, RACER and SHREC count k-mers, and Coral relies on multiple alignment. The source code is free for all seven programs and can be found from the links below:

(i) BLESS: sourceforge.net/projects/bless-ec/
(ii) Coral: www.cs.helsinki.fi/u/lmsalmel/coral/
(iii) HiTEC: www.csd.uwo.ca/~ilie/HiTEC/
(iv) Musket: sourceforge.net/projects/musket/
(v) RACER: www.csd.uwo.ca/~ilie/RACER/
(vi) SGA: github.com/jts/sga/
(vii) SHREC: sourceforge.net/projects/shrec-ec/

EVALUATION METHODOLOGY

Evaluating the correcting performance has been inconsistent throughout the literature. Direct evaluation can be done in two ways: using point correction of single base errors [15, 23, 24] or correction of whole reads [7, 14, 22]. Indirect evaluation of correction can be also performed through its impact on genome assembly quality [15, 21, 23, 24]. Furthermore, some papers use both real and simulated datasets [7, 23, 24] while others use only real datasets [14, 15, 21, 22]. We discuss in this section the suitability of these approaches while introducing our four measures of correcting performance evaluation, aiming at providing a comprehensive and accurate picture.

To make our definitions precise, we introduce some notation. Consider a reference genome G of length L. We denote the ith nucleotide by $G[i]$ and the subsequence starting at i and ending at j by $G[i..j]$; we have $G = G[1..L]$. Consider a dataset of N reads $D = \{R_i | 1 \le i \le N\}$. The length of a read R, denoted $|R|$, is the number of nucleotides in R; e.g. $|\texttt{ACCATG}| = 6$.

We shall distinguish a read R from its sequence of nucleotides $\mathrm{seq}(R)$. Different reads can have the same sequence. Denote $\mathrm{seq}(D) = \{\mathrm{seq}(R) | R \in D\}$.

A k-mer is a sequence of nucleotides of length k. To distinguish between k-mers and their sequences, we introduce the notion of 'positional k-mer', that is, the k-mer starting at position j in read R_i, $R_i[j..j+k-1]$, denoted (k, i, j). The set of positional k-mers is $\text{pos-k-mers}(D) = \{(k, i, j) | 1 \le i \le N, 1 \le j \le |R_i| - k + 1\}$. The sequences of positional k-mers are called simply k-mers. The set of k-mers occurring in D (or G) is denoted $\text{k-mer}(D)$ [$\text{k-mer}(G)$].

As an example, the dataset $D = \{\texttt{ACCT}, \texttt{ACCT}, \texttt{GGGG}\}$ contains three reads, two sequences ($\mathrm{seq}(D) = \{\texttt{ACCT}, \texttt{GGGG}\}$), nine positional 2-mers (3 in each read) and four 2-mers ($\text{2-mer}(D) = \{\texttt{AC}, \texttt{CC}, \texttt{CT}, \texttt{GG}\}$).

Coverage

Two parameters characterize a dataset and can be improved by correction: the depth of coverage and breadth of coverage. The 'depth of coverage' is the average number of times each position in the genome has been sequenced, that is, $\frac{1}{L}\sum_{i=1}^{N} |R_i|$. The 'breadth of coverage' is the fraction of the genome that is covered by reads. The evaluating methods we propose quantify the improvement of these two parameters.

To define precisely the breadth of coverage, we need to discuss what it means that a position in the genome is covered by a read or k-mer. Simply being part of an interval representing a read or k-mer is insufficient, as the intervals have to overlap significantly to assemble into longer, more useful, sequences. The overlap can take various values for overlap graph-based assembly, but it is k–1 for deBruijn graph-based assembly. For consistency, we shall always consider this maximum overlap. That means, we consider reads (or k-mers) starting at every position. For reads of length ℓ (in case of k-mers, replace ℓ with k), we define the breadth of coverage as the ratio of the ℓ-mers in the genome that appear in the dataset, that is, $\frac{|\ell\text{-mer}(G) \cap \ell\text{-mer}(D)|}{|\ell\text{-mer}(G)|}$.

Depth of coverage gain

As mentioned above, two main evaluation approaches have been proposed in the literature. One considers the point correction of single errors [15, 23, 24], and the other takes into account only successful correction of entire reads [7, 14, 22]. Counting point corrections of single errors may appear as the natural thing to do; however, it is not necessarily relevant in practice. Consider two scenarios. In the first, five errors are corrected in a read having only five errors. In the second, five errors are corrected in a read having ten errors. While they both count as five corrected errors, the former read becomes error free after correction, thus being more useful in downstream applications.

A read R is called 'correct' if it is found in the reference genome exactly as given, that is, $\mathrm{seq}(R) = \mathrm{seq}(G[i..i+|R|-1])$, for some $1 \le i \le L - |R| + 1$; R is called 'erroneous' otherwise. Based on this definition, we have a binary classifier on D where TP = the number of reads that are erroneous before correction and correct after correction, TN = reads correct both before and after correction, FP = reads correct before and erroneous after correction and FN = reads erroneous both before and after correction. The 'depth of coverage read gain' is then defined as $\textsc{ReadDepthGain} = \frac{TP-FP}{P} = \frac{TP-FP}{TP+FN}$. This means, the total number of correct whole reads gained, $TP - FP$, as a fraction of the total number, $P = TP + FN$, of erroneous reads before correction.

The $\textsc{ReadDepthGain}$ measures the gain in depth of coverage as given by correct reads; however, the impact on the quality of reads in the corrected dataset is inversely proportional to the quality of the reads in the initial dataset, as the proportion of the new correct reads out of the total number of reads is $\frac{TP-FP}{P+N} = \frac{P}{P+N}\textsc{ReadDepthGain}$. To evaluate the impact of correction, we introduce $\textsc{OrigReadDepth} = \frac{N}{P+N}$, that is, the ratio between the number of correct reads in the original dataset and the total number of reads, and $\textsc{CorrReadDepth} = \frac{TP+TN}{P+N}$, that is, the ratio between the number of correct reads in the corrected dataset and the total number of reads. The relation between the original and corrected read depth and the read depth gain is $\textsc{CorrReadDepth} = \textsc{OrigReadDepth} + \frac{P}{P+N}\textsc{ReadDepthGain}$, that can be written as follows:

$$\textsc{CorrReadDepth} = \textsc{OrigReadDepth} + \textsc{ReadDepthGain}(1 - \textsc{OrigReadDepth}). \quad (1)$$

The above reasoning, relying on read counting, works well when all reads have the same length. To accommodate the case when reads have different lengths, we modify the above definitions so that the contribution of each read R toward each of the values TP, TN, FP, FN is $|R|$ instead of 1. This produces the same values as above for equal length reads.

Producing error-free reads is an obvious mark for success, yet we need a more detailed analysis of the errors corrected. When complete correction is not achieved, the percentage of the errors corrected becomes important. However, the ratio of errors corrected is insufficient by itself to characterize the quality of correction. Consider again two scenarios. In both, five errors are corrected inside a read having ten errors, thus yielding the same percentage of errors corrected. Assume, however, that in the first the leftmost five errors are corrected, whereas in the second every other error is corrected. It is likely that the former read will have a longer error-free subsequence than the latter, again making it more useful in applications.

Our main point is that a corrected error becomes useful only if it helps in creating a sufficiently long error-free subsequence, long enough to warrant unique positioning in the reference genome (except for repeats). To account for this, we consider correction of k-mers, where the value of k is chosen as the smallest possible that guarantees, with high probability, unique position in the reference genomes. In all our experiments, we have used $k = 20$, but the evaluation program allows any value between 5 and 32. Note that we consider positional k-mers here.

The precise definition follows. A positional k-mer (k, i, j) is called 'correct' if it can be found in the reference genome, that is, $\mathrm{seq}(R_i[j..j+k-1]) = \mathrm{seq}(G[\ell..i+k-1])$, for some $1 \le \ell \le L-k+1$; the positional k-mer (k, i, j) is called 'erroneous' otherwise. A binary classifier on pos-k-mers(D), similar to the one for reads is obtained; TP is the number of positional k-mers that were erroneous before correction and correct afterward, etc. The 'depth of coverage positional k-mer gain' is defined as $\textsc{KmerDepthGain} = \frac{TP-FP}{TP+FN}$. The $\textsc{KmerDepthGain}$ quantifies the gain in depth of coverage as given by correct positional k-mers. As we did for whole reads, to evaluate the impact of correction on the quality of the k-mers, we introduce $\textsc{OrigKmerDepth} = \frac{N}{P+N}$ and $\textsc{CorrKmerDepth} = \frac{TP+TN}{P+N}$ that again satisfy (1) with the appropriate modifications:

$$\textsc{CorrKmerDepth} = \textsc{OrigKmerDepth} + \textsc{KmerDepthGain}(1 - \textsc{OrigKmerDepth}).$$

As previously mentioned, the two measures introduced so far, using whole reads and whole k-mers, are well related to the two main approaches to genome assembly: overlap graph-based assembly, which requires correct whole reads, and de Bruijn graph-based assembly, which uses correct whole k-mers.

Breadth of coverage gain

We have shown above that complete correction of whole reads, as well as of whole positional k-mers, is a good indication of the correcting performance from the depth of coverage point of view. For our other parameter, breadth of coverage, we need to consider read sequences and k-mers. Consider again two scenarios. Assume we have two reads, R_1 and R_2, R_1 having twenty copies, ten of which erroneous, and R_2 having ten copies, five of which erroneous. In the first scenario, a program P_1 corrects all copies of R_1 but destroys all copies of R_2. In the second, a program P_2 corrects two copies of R_1 and one copy of R_2, without destroying anything. We have that $\textsc{ReadDepthGain}(P_1) = 0.33$ and $\textsc{ReadDepthGain}(P_2) = 0.20$. Therefore, P_1's correction is judged superior. However, P_2 preserves copies of both reads, thus producing a dataset of superior breadth of coverage of the reference genome. We introduce two measures of the gain in breadth coverage to cover this aspect, complementing the previous ones. We handle the k-mer case first.

We note first that the depth of coverage is a property of the dataset, whereas the breadth of coverage is a property of the genome. Therefore, we shall define a binary classification on k-mer(G). A k-mer K of the genome is called 'covered' if $K \in$ k-mer(D).

The k-mer is 'not covered' otherwise. *TP* becomes the number of k-mers that are not covered before correction but covered afterward, etc. The new measure is KmerBreadthGain $= \frac{TP-FP}{TP+FN}$, and the impact on the breadth of coverage is assessed using OrigKmerBreadth and CorrKmerBreadth defined as before and satisfying (1).

For reads, if all reads have the same length ℓ, then $\mathrm{seq}(D) = \ell\text{-mer}(D)$, and the definition of ReadBreadthGain is similar with the above KmerBreadthGain, with k replaced by ℓ. If the reads do not have the same length, then *TP* is the number of elements of $\mathrm{seq}(D)$ that did not occur in G before but did afterward, *FP* represents the opposite and *TN* counts those that appeared both before and after. For *FN*, we use $|\ell\text{-mer}(G)|$, where ℓ is the weighted average read length. As above, we also introduce OrigReadBreadth and CorrReadBreadth satisfying (1).

This way, KmerBreadthGain measures the gain in breadth of coverage as given by correct k-mers, and ReadBreadthGain measures the gain in breadth of coverage as given by correct read sequences.

Unaltered real datasets

The first thing to note is that correction is much easier for simulated datasets [7, 23, 24], giving the false impression that a high percentage of errors (often more than 99%) can be corrected. As this does not happen in reality, artificial datasets are not relevant for practice, and we consider only real datasets, described in detail in the Datasets section.

In addition to our arguments above, evaluation of point correction of single errors cannot be performed for real datasets, as it is not known where the errors are or how they should be corrected. To surmount this obstacle, in the literature, datasets are sometimes mapped to the reference genome using read aligners, the place and correct values of erroneous positions being deduced from the information in the genome [15, 23, 24]. However, this procedure has a severe downside: many reads cannot be mapped at all or are mapped ambiguously. Therefore, they have to be removed from the dataset. As these are the most difficult reads to correct, their elimination produces a different, much easier to correct, dataset. Comparison using this modified dataset is no longer relevant for the performance on the real one. In our study, no dataset has been altered in any way.

To conclude this section, let us further note that we use only measures that are directly related to error correction, as opposed to indirect indicators such as the improvement in genome assembly quality or alignment of reads [15, 21, 23, 24]. Assemblers usually have their own correcting steps, and it is unclear how they interfere with the correcting programs under evaluation. Aligners as well have their own procedure that again interacts differently with various programs. Both are expected to improve with good correction of errors; however, we preferred not to use the amount of improvement for performance evaluation.

DATASETS

As mentioned in the previous section, we have used only real datasets that have not been altered in any way. Currently, the main platforms from Illumina are HiSeq and MiSeq, with different parameters. HiSeq has high output and low error rates. MiSeq is a bench top machine, much cheaper, with much lower output but also lower production time, and significantly higher error rates. Correcting errors in datasets produced by the two machines poses different challenges, and for that reason we shall compare the two separately.

We have included 13 HiSeq and 9 MiSeq datasets, all recently produced, from a wide variety of reference genomes and different coverage levels. All details are given in Table 1. The datasets in each part are sorted by the total number of base pairs. Included in the comparison, for the first time, are three whole human datasets, H11–13. The datasets H1, H4, H5 and H11 are from HiSeq 2500 machines, and the rest from HiSeq 2000.

The quality of the original datasets can be seen in Tables 2 and 3, in the columns labeled 'Original' (the other columns are discussed in the Results section.) The original quality is evaluated with respect to each of the four measures, as given by OrigReadDepth, OrigKmerDepth, OrigReadBreadth and OrigKmerBreadth. It can be seen that a significant ratio of reads and k-mers contain errors, making correction an essential step in improving the quality of the data. HiSeq data have fewer errors, yet up to 60% of the reads can be erroneous. This percentage can be as high as 99% in MiSeq data. The percentage of erroneous 20-mers is lower, but it can still reach 30% for HiSeq and 60% for MiSeq data. For M1, M5, M8 and M9, almost all reads are erroneous.

Table 1: The HiSeq and MiSeq datasets used in our study

Dataset	Organism	Accession number	Read length	Number of reads	Total bp	Depth of coverage	Reference genome	Genome length
HiSeq datasets								
H1	*Mycobacterium tuberculosis*	ERR400373	151	2 092 946	316 034 846	72	NC.000962.3	4 411 532
H2	*Salmonella enterica*	ERR230402	100	3 257 972	325 797 200	67	NC.011083.1	4 888 768
H3	*Saccharomyces cerevisiae*	ERR422544	100	4 776 774	477 677 400	40	R64-1-1	12 071 326
H4	*Legionella pneumophila*	SRR801797	100	8 850 220	885 022 000	260	NC.002942.5	3 397 754
H5	*Escherichia coli*	SRR1191655	101	11 726 414	1 184 367 814	255	NC.000913.2	4 639 675
H6	*Escherichia coli*	SRR490124	100	21 553 358	2 155 335 800	465	NC.000913.2	4 639 675
H7	*Caenorhabditis elegans*	SRX218989	100	31 642 176	3 164 217 600	32	WS222	100 286 070
H8	*Caenorhabditis elegans*	SRR543736	101	57 721 732	5 829 894 932	58	WS222	100 286 070
H9	*Drosophila melanogaster*	SRR823377	100	63 014 762	6 301 476 200	52	Release 5	120 381 546
H10	*Drosophila melanogaster*	SRR988075	101	75 938 276	7 669 765 876	64	Release 5	120 381 546
H11	*Homo sapiens*	ERX069715	100–102	1 357 751 670	137 132 918 670	43	Build 38	3 209 286 105
H12	*Homo sapiens*	ERX069504	100–102	1 637 816 924	165 419 509 324	52	Build 38	3 209 286 105
H13	*Homo sapiens*	ERX069505	101	1 708 169 546	172 525 124 146	54	Build 38	3 209 286 105
MiSeq datasets								
M1	*Escherichia coli*	SRR519926	251	801 192	201 099 192	43	NC.000913.2	4 639 675
M2	*Mycobacterium tuberculosis*	SRR1200797	50–250	1 482 716	348 224 181	79	NC.000962.3	4 411 532
M3	*Salmonella enterica*	SRR1203044	35–250	1 784 756	433 166 399	89	NC.011083.1	4 888 768
M4	*Salmonella enterica*	SRR1206093	35–251	1 977 970	472 256 906	97	NC.011083.1	4 888 768
M5	*Listeria monocytogenes*	SRR1198952	35–251	2 177 790	507 711 040	171	NC.017546.1	2 973 801
M6	*Pseudomonas syringae*	SRR1119292	35–251	2 576 622	639 853 726	105	NC.007005.1	6 093 698
M7	*Bifidobacterium dentium*	SRR1151311	35–251	3 926 618	984 280 778	373	NC.013714.1	2 636 367
M8	*Escherichia coli*	SRR522163	251	11 181 452	2 806 544 452	605	NC.000913.2	4 639 675
M9	*Orientia tsutsugamushi*	SRR1202083	301	10 315 434	3 104 945 634	1460	NC.009488.1	2 127 051

Accession numbers are included for the datasets and for the corresponding reference genomes. The datasets in each group are sorted by the total number of base pairs.

The breadth of coverage depends on both the quality of the dataset and the depth of coverage. The OrigReadBreadth is not expected to be very high but the OrigKmerBreadth is. This happens for the HiSeq data, where most datasets have OrigKmerBreadth more than 90%, but not for MiSeq, where we detect a wide range, from 50% to 100%.

RESULTS

We present our results in Tables 2–8. All numbers presented have been multiplied by 100 for readability. Tables 4–8 are presented as heat maps, with darker colors meaning better results. Complete information is provided in the Supplementary Material (Supplementary_File_1.xlsx). Not all datasets could be run by all programs, the reasons being given in the caption of Table 4.

We have run all programs on the same DELL PowerEdge R820 computer with 32 cores Intel Xeon at 2.2 GHz and 1 TB of RAM, running Linux Red Hat, CentOS 6.3. All programs have been tested in parallel, except BLESS and HiTEC, which do not have parallel modes. All programs were run with default parameters as indicated in their manuals, as this is the only way they can be used in real tests, when reference genomes are not available. The precise commands used to run all the programs are given in the Supplementary Material (Supplementary_File_2.pdf).

Comparison on HiSeq data

The error correction comparison on HiSeq data is presented in Table 4 for the ReadDepthGain and KmerDepthGain measures and in Table 5 for ReadBreadthGain and KmerBreadthGain measures. Only Musket, RACER and SGA could run the human datasets; therefore, we provide two averages, one for the small- and medium-sized datasets H1-10 (denoted Avg_{10}) and the other for all 13 datasets ($\mathrm{Avg}_{\mathrm{all}}$). The averages for HiTEC are very high because HiTEC could run only H1-6. SHREC also failed to run five datasets. Under these conditions, whenever necessary, we used head-to-head comparison to rank the programs.

Table 2: Improvement of the quality of HiSeq data

Dataset	(Orig | Corr) ReadDepth			(Orig | Corr) KmerDepth			(Orig | Corr) ReadBreadth			(Orig | Corr) KmerBreadth		
	Original	Best correction	Program	Original	Best correction	Program	Original	Best correction	Program	Original	Best correction	Program
H1	80.15	92.40	BLESS	93.80	96.98	Musket	30.36	34.29	BLESS	98.54	98.51	Coral
H2	81.93	89.94	BLESS	92.99	95.31	HiTEC	39.68	42.42	BLESS	93.90	93.80	SGA
H3	82.01	88.15	RACER	90.76	92.54	RACER	26.46	28.14	BLESS	98.26	98.24	SGA
H4	66.73	97.43	RACER	88.41	98.36	HiTEC	80.38	90.09	BLESS	98.94	98.94	Coral
H5	47.83	54.98	BLESS	71.53	74.17	HiTEC	50.59	52.88	BLESS	84.08	82.52	SGA
H6	50.56	93.16	BLESS	86.16	97.62	HiTEC	79.67	96.51	BLESS	99.96	99.96	Coral
H7	68.13	84.98	SGA	90.98	95.25	RACER	18.08	21.92	BLESS	96.38	96.31	SGA
H8	69.70	77.81	RACER	78.96	81.47	RACER	31.34	34.13	BLESS	99.45	99.43	SGA
H9	46.23	55.98	BLESS	77.33	81.00	RACER	19.28	22.92	BLESS	93.19	92.80	SGA
H10	40.07	55.73	BLESS	75.36	80.20	Musket	19.97	26.36	BLESS	95.41	95.11	SGA
H11	49.17	80.51	SGA	94.23	96.43	RACER	19.79	30.16	SGA	98.86	98.81	SGA
H12	49.09	82.41	SGA	94.21	96.67	RACER	23.23	35.63	SGA	98.96	98.91	SGA
H13	78.90	84.78	SGA	95.08	96.83	RACER	35.05	37.12	SGA	98.54	98.48	SGA

The quality of the original data is presented in the 'Original' columns as given by OrigReadDepth, OrigKmerDepth, OrigReadBreadth and OrigKmerBreadth. The quality of the corrected data is presented in the 'Best correction' columns where CorrReadDepth, CorrKmerDepth, CorrReadBreadth and CorrKmerBreadth are given for the best correction obtained in each case, obtained by the program indicated in the corresponding 'Program' column.

Table 3: Improvement of the quality of MiSeq data

Dataset	(Orig | Corr) ReadDepth			(Orig | Corr) KmerDepth			(Orig | Corr) ReadBreadth			(Orig | Corr) KmerBreadth		
	Original	Best correction	Program	Original	Best correction	Program	Original	Best corr.	Program	Original	Best corr.	Program
M1	1.46	27.38	SGA	50.57	69.07	RACER	0.25	6.98	BLESS	100.00	100.00	Coral/SGA
M2	64.92	85.29	RACER	93.21	96.57	RACER	18.27	23.43	RACER	98.60	98.60	Coral
M3	23.21	33.26	RACER	72.79	77.97	RACER	5.78	7.61	RACER	83.16	83.16	Coral
M4	52.60	88.13	RACER	84.77	93.71	RACER	14.15	20.79	RACER	97.51	97.51	Coral
M5	2.35	3.21	RACER	41.01	43.36	RACER	0.87	0.96	RACER	50.16	49.69	SGA
M6	11.40	25.99	RACER	67.48	74.34	RACER	3.62	6.97	RACER	78.50	78.50	Coral
M7	69.35	95.74	RACER	94.36	98.06	RACER	58.28	70.13	RACER	99.99	99.99	Coral
M8	2.71	39.47	HiTEC	62.11	80.16	HiTEC	5.92	69.85	BLESS	100.00	100.00	HiTEC
M9	0.29	0.68	RACER	42.90	46.73	HiTEC	1.01	1.96	BLESS	74.95	74.95	Coral

The quality of the original data is presented in the 'Original' columns as given by OrigReadDepth, OrigKmerDepth, OrigReadBreadth and OrigKmerBreadth. The quality of the corrected data is presented in the 'Best correction' columns where CorrReadDepth, CorrKmerDepth, CorrReadBreadth and CorrKmerBreadth are given for the best correction obtained in each case, obtained by the program indicated in the corresponding 'Program' column.

The competition is very close for all four measures. For ReadDepthGain, BLESS is first for H1–10 and RACER is first overall. HiTEC, SGA, Musket and SHREC are close behind. For human datasets, SGA is first, RACER second and Musket third.

For KmerDepthGain, HiTEC is first, followed by RACER, BLESS, Musket and SHREC, all very close. However, of these programs, HiTEC could run the fewest datasets, and only RACER and Musket could run all datasets. For human datasets, the order changes: RACER is first, Musket is second and SGA third.

In the case of ReadBreadthGain, BLESS is first, followed closely by RACER, HiTEC, SHREC, Musket and SGA. For human datasets, SGA is first, RACER second and Musket third. For KmerBreadthGain, SGA is first, followed by Coral, BLESS, RACER and HiTEC. For human datasets, SGA is first, Musket second and RACER third. All programs actually decrease the k-mer breadth of coverage.

Table 4: Comparison on HiSeq datasets for ReadDepthGain and KmerDepthGain. Darker colors indicate better results

Data	BLESS	Coral	HiTEC	Musket	RACER	SGA	SHREC
ReadDepthGain							
H1	61.72	51.85	58.41	58.30	59.05	51.16	51.28
H2	44.30	41.16	43.37	42.77	42.72	40.14	42.64
H3	34.07	28.41	33.42	33.09	34.13	32.73	31.56
H4	91.64	55.66	92.01	86.56	92.27	86.94	88.90
H5	13.52	12.32	13.70	13.08	13.35	12.35	(e)
H6	86.16	4.37	84.46	79.08	82.84	55.60	65.96
H7	51.65	46.49	(d)	41.56	52.43	52.89	(f)
H8	25.14	24.21	(d)	23.68	26.77	25.97	−7.12
H9	18.14	10.96	(d)	16.79	17.91	17.03	14.80
H10	26.13	24.10	(d)	26.04	25.92	25.63	24.44
H11	(a)	(c)	(a)	53.28	57.49	61.66	(f)
H12	(a)	(c)	(a)	55.67	61.30	65.46	(e)
H13	(b)	(c)	(c)	21.25	26.53	27.86	(e)
Avg$_{10}$	45.25	29.95	54.23	42.10	44.74	40.04	39.06
Avg$_{all}$	45.25	29.95	54.23	42.40	45.59	42.72	39.06
KmerDepthGain							
H1	51.14	34.08	51.00	51.28	51.11	34.20	41.84
H2	31.36	26.92	33.03	30.03	31.18	26.29	32.45
H3	17.86	14.45	18.22	18.55	19.26	15.96	16.79
H4	83.20	46.92	85.83	80.72	84.82	75.42	82.91
H5	8.79	7.13	9.27	8.41	8.74	7.14	(e)
H6	79.52	2.92	82.83	72.04	78.45	41.68	67.72
H7	42.21	36.79	(d)	38.92	47.35	42.46	(f)
H8	9.92	8.51	(d)	10.72	11.92	8.95	2.01
H9	14.43	7.81	(d)	15.40	16.19	12.90	14.24
H10	18.24	16.60	(d)	19.64	19.55	17.55	18.62
H11	(a)	(c)	(a)	36.73	38.11	31.79	(f)
H12	(a)	(c)	(a)	37.73	42.42	35.93	(e)
H13	(b)	(c)	(c)	31.38	35.49	28.26	(e)
Avg$_{10}$	35.67	20.21	46.70	34.57	36.86	28.25	34.57
Avg$_{all}$	35.67	20.21	46.70	34.73	37.28	29.12	34.57

The meaning of the letters in parentheses, explaining why a program could not be run for a dataset, is: (a) reads with different lengths; (b) longer than 7 days to run; (c) out of space; (d) segmentation fault; (e) java.lang.NegativeArraySizeException; (f) java.lang.ArrayIndexOutOfBoundsException.

Table 5: Comparison on HiSeq datasets for ReadBreadthGain and KmerBreadthGain. Darker colors indicate better results

Data	BLESS	Coral	HiTEC	Musket	RACER	SGA	SHREC
ReadBreadthGain							
H1	5.66	4.42	5.00	4.99	5.04	4.36	4.37
H2	4.54	4.15	4.19	4.35	4.32	4.06	4.11
H3	2.28	1.88	2.16	2.13	2.23	2.13	2.03
H4	49.49	31.05	44.96	47.05	49.15	47.10	43.63
H5	4.63	4.09	4.35	4.37	4.43	4.08	(e)
H6	82.87	1.32	80.77	77.59	79.37	59.38	68.67
H7	4.69	4.10	(d)	3.63	4.62	4.63	(f)
H8	4.06	3.63	(d)	3.43	4.06	3.96	−2.96
H9	4.50	2.49	(d)	3.71	4.13	3.95	3.26
H10	7.98	7.12	(d)	7.60	7.82	7.77	7.16
H11	(a)	(c)	(a)	11.17	12.05	12.92	(f)
H12	(a)	(c)	(a)	13.76	15.10	16.15	(e)
H13	(b)	(c)	(c)	2.42	3.02	3.18	(e)
Avg$_{10}$	17.07	6.43	23.57	15.89	16.52	14.14	16.28
Avg$_{all}$	17.07	6.43	23.57	14.32	15.03	13.36	16.28
KmerBreadthGain							
H1	−3.08	−1.74	−6.22	−4.02	−3.00	−2.09	−4.16
H2	−1.91	−1.92	−1.90	−1.85	−1.89	−1.73	−1.89
H3	−1.07	−5.09	−1.26	−0.93	−1.12	−0.88	−1.02
H4	−0.36	−1.65	−1.72	−1.72	−1.72	−1.73	−1.72
H5	−11.60	−9.98	−11.75	−11.36	−11.69	−9.83	(e)
H6	−2.20	0.00	−4.92	−10.63	−2.88	−0.21	−5.71
H7	−5.10	−2.68	(d)	−13.46	−7.69	−1.99	(f)
H8	−38.17	−10.96	(d)	−180.53	−86.13	−2.84	−285.54
H9	−30.27	−16.51	(d)	−33.87	−17.15	−5.70	−33.27
H10	−24.62	−39.15	(d)	−30.01	−19.22	−6.49	−33.34
H11	(a)	(c)	(a)	−6.30	−6.06	−4.89	(f)
H12	(a)	(c)	(a)	−7.15	−7.15	−5.63	(e)
H13	(b)	(c)	(c)	−5.00	−5.60	−4.32	(e)
Avg$_{10}$	−11.84	−8.97	−4.63	−28.84	−15.25	−3.35	−45.83
Avg$_{all}$	−11.84	−8.97	−4.63	−23.60	−13.18	−3.72	−45.83

The meaning of the letters in parentheses, explaining why a program could not be run for a dataset, is: (a) reads with different lengths; (b) longer than 7 days to run; (c) out of space; (d) segmentation fault; (e) java.lang.NegativeArraySizeException; (f) java.lang.ArrayIndexOutOfBoundsException.

Table 2 gives the top program for each particular test. This table is dominated by BLESS, RACER and SGA.

Comparison on MiSeq data

The error correction comparison on MiSeq data is presented in Table 6 for all four measures, ReadDepthGain, KmerDepthGain, ReadBreadthGain and KmerBreadthGain, in order. BLESS and HiTEC could not run six of the nine datasets, M2–7, because they contain reads of different lengths. Musket performance is clearly the lowest, which is unexpected in view of its good results on HiSeq data. The *TP* values of Musket are several orders of magnitude lower than those of the best programs; they are equal to zero for four datasets. Essentially, Musket left the MiSeq datasets untouched.

As opposed to the HiSeq case, the competition for the coverage depth is not very close. For ReadDepthGain, RACER is first, followed by SGA. We note good performance for BLESS and HiTEC on the three datasets they could run. For KmerDepthGain, RACER is first again with no other program close to it. However, HiTEC is first on two out of three datasets that it could run.

For ReadBreadthGain, RACER is first again but with SGA and SHREC following closely. However, BLESS is the best for all three datasets it could run, with HiTEC second for two of those. For KmerBreadthGain, Coral is the best with SGA coming in second place. Again, all programs decrease the k-mer breadth of coverage. We have not

Table 6: Comparison on MiSeq datasets. Darker colors indicate better results

Data	BLESS	Coral	HiTEC	Musket	RACER	SGA	SHREC
ReadDepthGain							
M1	22.04	0.06	25.27	0.00	26.10	26.30	13.38
M2	(a)	30.65	(a)	−0.02	58.06	55.10	53.43
M3	(a)	10.23	(a)	0.06	13.09	11.56	−12.36
M4	(a)	61.32	(a)	0.70	74.96	66.16	−39.56
M5	(a)	0.42	(a)	0.06	0.88	0.64	−1.27
M6	(a)	13.03	(a)	0.00	16.46	14.60	−1.31
M7	(a)	77.71	(a)	0.00	86.11	67.74	−33.41
M8	33.61	0.01	37.78	0.00	20.19	22.72	19.95
M9	0.37	0.00	0.39	0.00	0.39	0.18	0.33
Avg	18.67	21.49	21.15	0.09	32.92	29.44	−0.09
KmerDepthGain							
M1	21.22	0.04	36.29	0.00	37.42	9.50	23.81
M2	(a)	22.32	(a)	−0.16	49.48	41.22	42.80
M3	(a)	11.78	(a)	0.07	19.05	12.20	8.93
M4	(a)	34.36	(a)	0.46	58.72	33.33	22.66
M5	(a)	2.81	(a)	0.12	4.00	2.81	−1.72
M6	(a)	11.40	(a)	0.01	21.09	11.28	16.42
M7	(a)	48.40	(a)	0.00	65.59	38.27	59.46
M8	32.74	0.91	47.65	0.00	30.67	11.82	35.55
M9	4.28	0.61	6.71	0.00	6.19	1.61	5.74
Avg	19.42	14.74	30.22	0.06	32.47	18.00	23.74
ReadBreadthGain							
M1	6.75	0.01	4.22	0.00	4.36	4.39	2.26
M2	(a)	3.38	(a)	−0.02	6.31	5.99	5.81
M3	(a)	1.50	(a)	0.00	1.94	1.70	1.70
M4	(a)	6.29	(a)	0.02	7.73	6.81	6.67
M5	(a)	0.07	(a)	0.01	0.10	0.09	0.09
M6	(a)	2.76	(a)	0.00	3.47	3.14	2.57
M7	(a)	25.48	(a)	0.00	28.41	22.60	26.72
M8	67.96	0.01	54.46	0.00	33.65	37.29	33.82
M9	0.97	0.00	0.47	0.00	0.46	0.28	0.42
Avg	25.22	4.39	19.72	0.00	9.60	9.14	8.90
KmerBreadthGain							
M1	−187.97	0.00	−1206.02	0.00	−12.78	0.00	−110.53
M2	(a)	−1.57	(a)	0.01	−6.60	−3.98	−6.41
M3	(a)	−2.61	(a)	−0.01	−3.76	−2.91	−3.61
M4	(a)	−0.35	(a)	0.00	−0.60	−0.36	−0.91
M5	(a)	−7.49	(a)	−0.95	−12.43	−6.56	−11.71
M6	(a)	−3.06	(a)	0.00	−5.12	−3.33	−4.77
M7	(a)	−0.51	(a)	0.00	−6.15	−1.54	−6.67
M8	−6.67	0.00	5.00	0.00	−5.00	−0.83	−2.50
M9	−39.71	−1.70	−40.69	0.00	−42.11	−15.28	−37.07
Avg	−78.12	−1.92	−413.90	−0.11	−10.51	−3.87	−20.46

The meaning of the letters in parentheses, explaining why a program could not be run for a dataset, is: (a) reads with different lengths; (b) longer than 7 days to run; (c) out of space; (d) segmentation fault; (e) java.lang.NegativeArraySizeException; (f) java.lang.ArrayIndexOutOfBoundsException.

considered the performance of Musket. It has the smallest amount of decrease but that is because it leaves the initial datasets largely unchanged and it would be misleading to consider its performance the best.

Table 3 gives the top program for each particular test. This table is dominated by RACER.

Table 7: Time and memory for HiSeq datasets. Darker colors indicate better results

Data	BLESS	Coral	HiTEC	Musket	RACER	SGA	SHREC
Time (s/Mb)							
H1	8.59	1.43	5.41	0.32	0.15	0.59	1.80
H2	7.73	1.40	3.82	0.25	0.30	0.57	1.06
H3	4.13	0.84	5.01	0.20	0.15	0.52	0.87
H4	8.44	2.42	6.79	0.29	0.37	0.76	1.41
H5	8.04	2.91	6.74	0.21	0.30	0.62	(e)
H6	8.28	2.17	6.81	0.51	0.33	0.74	1.82
H7	12.38	1.79	(d)	0.29	0.30	0.74	(f)
H8	12.86	1.42	(d)	0.27	0.28	0.74	2.13
H9	12.28	1.80	(d)	0.46	0.36	0.78	2.03
H10	14.54	1.60	(d)	0.33	0.32	0.79	2.03
H11	(a)	(c)	(a)	0.29	0.80	1.08	(f)
H12	(a)	(c)	(a)	0.31	0.50	1.06	(e)
H13	(b)	(c)	(c)	0.27	0.49	1.03	(e)
Avg 10	9.73	1.78	5.76	0.31	0.28	0.69	1.64
Avg all	9.73	1.78	5.76	0.31	0.36	0.77	1.64
Space (MB/Mb)							
H1	0.04	226.91	18.94	9.74	9.79	9.06	3172.65
H2	0.04	219.17	19.05	9.36	9.68	8.79	3077.78
H3	0.05	153.14	19.96	6.40	6.94	6.05	2099.05
H4	0.02	90.31	10.60	3.45	4.37	3.45	1132.93
H5	0.01	68.19	7.67	2.58	3.09	2.63	(e)
H6	0.02	45.34	5.01	1.43	2.15	1.61	465.23
H7	0.05	33.31	(d)	0.99	2.78	1.19	(f)
H8	0.03	23.88	(d)	0.81	2.50	0.81	171.99
H9	0.03	23.10	(d)	0.75	2.82	0.78	159.14
H10	0.03	21.40	(d)	0.83	2.38	0.70	130.76
H11	(a)	(c)	(a)	0.44	1.69	0.36	(f)
H12	(a)	(c)	(a)	0.41	2.36	0.35	(e)
H13	(b)	(c)	(c)	0.35	1.38	0.34	(e)
Avg 10	0.03	90.48	13.54	3.63	4.65	3.51	1301.19
Avg all	0.03	90.48	13.54	2.89	4.00	2.78	1301.19

Time is given in seconds and memory in megabytes, both per input mega base pair. The meaning of the letters in parentheses, explaining why a program could not be run for a dataset, is: (a) reads with different lengths; (b) longer than 7 days to run; (c) out of space; (d) segmentation fault; (e) java.lang.NegativeArraySizeException; (f) java.lang.ArrayIndexOutOfBoundsException.

Time and space

The time and peak memory usages are shown in Table 7 for HiSeq data and Table 8 for MiSeq data. To be able to present an easily readable comparison, we give the time in seconds and memory in megabytes, both per mega base pairs of input data. The actual time and memory values are presented in Supplementary Material (Supplementary_File_1.xlsx). As mentioned above, BLESS and HiTEC do not provide a parallel mode, so we give their time and memory for running in serial mode.

Because of the size of the HiSeq datasets, the time and space requirements are more important in this case. For the datasets H1-10, RACER is the fastest, followed closely by Musket and, at some distance, by

Table 8: Time and memory for MiSeq datasets. Darker colors indicate better results

Data	BLESS	Coral	HiTEC	Musket	RACER	SGA	SHREC
Time (s/Mb)							
M1	12.41	1.03	5.58	0.08	0.85	1.32	3.40
M2	(a)	1.74	(a)	0.18	0.18	0.58	0.74
M3	(a)	15.15	(a)	0.10	0.33	0.72	1.38
M4	(a)	2.09	(a)	0.10	0.37	0.79	1.51
M5	(a)	3.03	(a)	0.12	0.12	0.40	1.33
M6	(a)	8.45	(a)	0.09	0.35	0.81	1.53
M7	(a)	9.94	(a)	0.07	0.30	0.71	1.05
M8	16.90	2.03	6.35	0.08	0.45	1.26	5.12
M9	14.40	1.84	7.07	0.07	0.34	1.21	2.56
Avg	14.57	5.03	6.34	0.10	0.37	0.87	2.07
Space (MB/Mb)							
M1	0.12	369.66	20.31	6.97	30.63	14.31	4985.94
M2	(a)	206.49	(a)	8.95	7.93	8.20	2879.37
M3	(a)	162.12	(a)	6.19	8.46	6.65	2314.74
M4	(a)	159.98	(a)	5.68	7.64	6.17	2123.14
M5	(a)	146.36	(a)	6.14	6.64	5.87	1974.88
M6	(a)	121.54	(a)	4.88	6.11	4.49	1567.03
M7	(a)	80.53	(a)	3.17	3.18	3.02	1018.68
M8	0.06	48.79	4.17	0.56	2.53	1.36	357.28
M9	0.02	33.87	3.27	0.52	1.37	1.15	322.93
Avg	0.07	147.71	9.25	4.78	8.28	5.69	1949.33

Time is given in seconds and memory in megabytes, both per input mega base pair. The meaning of the letters in parentheses, explaining why a program could not be run for a dataset, is: (a) reads with different lengths; (b) longer than 7 days to run; (c) out of space; (d) segmentation fault; (e) java.lang.NegativeArraySizeException; (f) java.lang.ArrayIndexOutOfBoundsException.

SGA. When considering all datasets, Musket comes first and RACER comes second. BLESS is by far the slowest, not only because of running in serial mode, but also due to spending a lot of time reading/writing files, to reduce memory consumption. Because of this, BLESS was not able to correct the human dataset H13 after running for 7 days. Also, BLESS took longer to correct the H10 dataset than the time Musket or RACER required for any of the human datasets.

The peak memory of BLESS is by far the lowest, two orders of magnitude lower than second best. SGA is second followed closely by Musket and RACER. Owing to high space requirements, Coral, SHREC and, to a lesser degree, HiTEC cannot run large datasets.

For MiSeq data, the space is usually not an issue, due to smaller size of the datasets. The order is similar to that for HiSeq data, with BLESS two orders of magnitude ahead, then Musket, SGA and RACER. Speed is more important, as the MiSeq machines have a much lower cycle time. The order is Musket, RACER and SGA. The time taken by BLESS to correct M8 and M9 is about half a day, comparable with one machine cycle.

Improvement of the data

The actual improvement for each HiSeq dataset is presented in Table 2 and for each MiSeq dataset in Table 3, where the quality of both the original dataset and the best correction obtained with respect to each of the four measures is presented. The best result is given in each case, produced by the program mentioned. Each table is divided into four parts, one for each of our measures. The connection between Tables 2 and 3 and the gain in Tables 4–6 is given by the equation (1). For instance, consider the dataset H1 in the first part of Table 2. We have $\textsc{OrigReadDepth} = 80.15\%$ and, from Table 4, $\textsc{ReadDepthGain} = 61.72\%$, which adds $\textsc{ReadDepthGain}(1 - \textsc{OrigReadDepth}) = 12.25\%$ correct reads to obtain, using (1), $\textsc{CorrReadDepth} = 92.40\%$, where the best gain was obtained by BLESS.

For HiSeq data, the improvement is significant for the reads, for both depth and breadth of coverage. On the average, the read depth of coverage is increased by >17.5% and the read breadth of coverage by about 6%. Dramatic improvements are sometimes obtained, of 42.6% depth increase and 16.8% breadth increase for H6. The first two human datasets have been quite significantly improved as well.

The numbers are expected to be lower for k-mers; the k-mer coverage depth is increased on the average by 4%, however, the k-mer breadth of coverage is unexpectedly decreased by 0.2% on the average. The depth increase can go as high as 11.5% (for H6), while the breadth decrease can be as high as 1.6% (for H5).

For MiSeq data, similar trends are identified. Owing to higher read length, the absolute increase in read coverage depth is slightly lower, 13% on the average, while the read coverage breadth increases by 7.7% on the average. High variability allows for dramatic improvement; for example, in the case of M1, the corrected dataset contains 18.7 times more correct reads than the original and the read coverage breadth is increased 27.5 times. Each of M2, M4 and M6–8 datasets has been much improved through correction.

The k-mer coverage depth increases 5.4%, while the k-mer coverage breadth decreases again by 0.37%. Note the significant decrease for M5, by 3.27%.

CONCLUSION AND FURTHER RESEARCH

Concerning the ranking of the correcting programs, it is clear that there is no single winner. For HiSeq data, BLESS, Musket, RACER and SGA are the best programs. BLESS has often good gain and the lowest memory requirements, but it has the longest running time, does not run in parallel and cannot run datasets with different read lengths. Musket comes rarely on top, but it is often not far from the best and it runs fast with low memory requirements. RACER has often the highest gain and speed but uses a fairly large amount of memory for human datasets. SGA has one of the top gains and low space requirements, but it is slower than RACER and Musket. Using any of the four programs will provide good results. Only Musket, RACER and SGA can run human datasets.

For MiSeq data, RACER is the best in three of the four measures, read coverage (depth and breadth) and k-mer coverage depth. For the fourth, k-mer coverage breadth, SGA provides the smallest decrease. BLESS and HiTEC produce good results whenever they can run.

Looking from the data point of view, the overall improvement of the datasets is quite important, in spite of the slight decrease in k-mer coverage breadth, making the correction of crucial importance in using both HiSeq and MiSeq data. Room for improvement remains in all aspects. The read coverage depth can be theoretically more than doubled for HiSeq data and increased over 5 times for MiSeq data. The k-mer coverage depth can be increased 3 times for HiSeq data and 6 times for MiSeq data. The increase in coverage breadth depends not only on the quality of the correction but also on the coverage depth, so it is more difficult to assess how much room for improvement there is. However, one goal is to reduce the decrease in k-mer coverage breadth exhibited by all programs. A goal that may be hard to achieve is to design programs that improve the current state of the art with respect to all four measures simultaneously. If this cannot be achieved, then the future programs may become more specialized by targeting improvements of only some aspects of the original datasets.

A different and important aspect that requires further investigation is that of biological significance of the correction. Important information such as heterozygosity or single nucleotide polymorphism should not be destroyed by correction. Also, we have seen that all programs reduce the k-mer breadth of coverage. Further investigation is necessary to determine how important the lost information is.

SOFTWARE AVAILABILITY

The source code of the programs we used for evaluation is freely available at http://www.csd.uwo.ca/~ilie/CorrectingIlluminaData/. The first is `readSearch`, which computes ReadDepthGain and ReadBreadthGain as well as any related information, such as OrigReadDepth, OrigReadBreadth, CorrReadDepth, CorrReadBreadth. It uses a suffix array [31] built on the reference genome, using the libdivsufsort library of Yuta Mori (code.google.com/p/libdivsufsort/) and the longest common prefix array computed using the algorithm of [32]. The second program, `kmerSearch`, computes KmerDepthGain and KmerBreadthGain as well as any related information, such as OrigKmerDepth, OrigKmerBreadth, CorrKmerDepth and CorrKmerBreadth. It is implemented using the same strategy as our RACER software [22].

SUPPLEMENTARY DATA

Supplementary data are available online at http://bib.oxfordjournals.org/.

Key Points

- Error correction is essential for improving Illumina sequencing data before using it in applications.
- Existing correcting programs are effective, recovering an important amount of erroneous data, yet significant room for improvement remains.
- Recommended programs:
  - HiSeq data: BLESS, Musket, RACER and SGA.
  - MiSeq data: RACER.
  - Human data: Musket, RACER and SGA.

Acknowledgement

The evaluation has been performed using the facilities of the Shared Hierarchical Academic Research Computing Network (SHARCNET: www.sharcnet.ca) and Compute/Calcul Canada.

AUTHOR CONTRIBUTIONS

L.I. designed the study and the evaluation methodology and wrote the article. M.M. selected and downloaded the datasets, implemented the evaluation programs, installed the correcting programs and performed all tests.

FUNDING

This work was supported by The Natural Sciences and Engineering Research Council of Canada (NSERC) to L.I.

References

1. Sanger F, Nicklen S, Coulson AR. DNA sequencing with chain-terminating inhibitors. *Proc Natl Acad Sci USA* 1977; **74**(12):5463–7.
2. Metzker ML. Sequencing technologies–the next generation. *Nat Rev Genet* 2010;**11**(1):31–46.
3. Haussler D, O'Brien SJ, Ryder OA, *et al*. Genome 10K: a proposal to obtain whole-genome sequence for 10,000 vertebrate species. *J. Hered* 2009;**100**(6):659–74.
4. Siva N. 1000 Genomes project. *Nat Biotech* 2008;**26**(3):256.
5. Turnbaugh PJ, Ley RE, Hamady M, *et al*. The human microbiome project. *Nature* 2007;**449**(7164):804–10.
6. Chaisson M, Pevzner P, Tang H. Fragment assembly with short reads. *Bioinformatics* 2004;**20**:2067–4.
7. Schröder J, Schröder H, Puglisi SJ, *et al*. SHREC: a short-read error correction method. *Bioinformatics* 2009;**25**(17): 2157–63.
8. Wijaya E, Frith MC, Suzuki Y, *et al*. Recount: expectation maximization based error correction tool for next generation sequencing data. *Genome Inform* 2009;**23**:189–201.
9. Yang X, Dorman KS, Aluru S. Reptile: representative tiling for short read error correction. *Bioinformatics* 2010;**26**(20): 2526–33.
10. Kelley DR, Schatz MC, Salzberg SL. Quake: quality-aware detection and correction of sequencing errors. *Genome Biol* 2010;**11**:R116.
11. Shi H, Schmidt B, Liu W, *et al*. A parallel algorithm for error correction in high-throughput short-read data on CUDA-Enabled graphics hardware. *J Comput Biol* 2010; **17**(4):603–15.
12. Salmela L. Correction of sequencing errors in a mixed set of reads. *Bioinformatics* 2010;**26**(10):1284–90.
13. 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.
14. Ilie L, Fazayeli F, Ilie S. HiTEC: accurate error correction in high-throughput sequencing data. *Bioinformatics* 2011;**27**(3): 295–302.
15. Salmela L, Schröder J. Correcting errors in short reads by multiple alignments. *Bioinformatics* 2011;**27**(11):1455–61.
16. Liu Y, Schmidt B, Maskell DL. DecGPU: distributed error correction on massively parallel graphics processing units using CUDA and MPI. *BMC Bioinformatics* 2011; **12**(1):85.
17. Medvedev P, Scott E, Kakaradov B, *et al*. Error correction of high-throughput sequencing datasets with non-uniform coverage. *Bioinformatics* 2011;**27**(13):i137–141.
18. Kao W-C, Chan AH, Song YS. ECHO: a reference-free short-read error correction algorithm. *Genome Res* 2011;**21**: 1181–92.
19. Zhao Z, Yin J, Zhan Y, *et al*. PSAEC: an improved algorithm for short read error correction using partial suffix arrays. In: Atallah M, Li XL, Zhu B (eds). *Frontiers in Algorithmics and Algorithmic Aspects in Information and Management, volume 6681 of Lecture Notes in Computer Science*. Berlin/Heidelberg: Springer, 2011;220–32.
20. Zhao Z, Yin J, Li Y, *et al*. An efficient hybrid approach to correcting errors in short reads. In: Torra V, Narakawa Y, Yin J, Long J (eds). *Modeling Decision for Artificial Intelligence, volume 6820 of Lecture Notes in Computer Science*. Berlin/Heidelberg: Springer, 2011;198–210.
21. Simpson JT, Durbin R. Efficient de novo assembly of large genomes using compressed data structures. *Genome Res* 2012;**22**:549–56.
22. Ilie L, Molnar M. RACER: rapid and accurate correction of errors in reads. *Bioinformatics* 2013;**29**(19): 2490–3.
23. Liu Y, Schröder J, Schmidt B. Musket: a multistage k-mer spectrum-based error corrector for Illumina sequence data. *Bioinformatics* 2013;**29**(3):308–15.
24. Heo Y, Wu X-L, Chen D, *et al*. BLESS: bloom-filter-based error correction solution for high-throughput sequencing reads. *Bioinformatics* 2014;**30**:1354–62.
25. Yang X, Chockalingam SP, Aluru S. A survey of error-correction methods for next-generation sequencing. *Brief Bioinform* 2013;**14**:56–66.
26. Butler J, MacCallum I, Kleber M, *et al*. ALLPATHS: De novo assembly of whole-genome shotgun microreads. *Genome Res* 2008;**18**:810–20.
27. Chaisson MJ, Pevzner PA. Short read fragment assembly of bacterial genomes. *Genome Res* 2008;**18**:324–30.
28. Zerbino DR, Birney E. Velvet: algorithms for de novo short read assembly using de bruijn graphs. *Genome Res* 2008;**18**(5):821–9.
29. Simpson JT, Wong K, Jackman SD, *et al*. ABySS: a parallel assembler for short read sequence data. *Genome Res* 2009;**19**: 1117–23.
30. Pevzner PA, Tang H, Waterman MS. An eulerian path approach to DNA fragment assembly. *Proce Natl Acad Sci USA* 2001;**98**(17):9748–53.
31. Manber U, Myers G. Suffix arrays: a new method for on-line string searches. *SIAM J Comput* 1993;**22**(5): 935–48.
32. Kasai T, Lee G, Arimura H, *et al*. Linear-time longest-common-prefix computation in suffix arrays and its applications. In: *Combinatorial Pattern Matching*. Berlin Heidelberg: Springer, 2001;181–92.