Alic, AS.; Ruzafa, D.; Dopazo, J.; Blanquer Espert, I. (2016). Objective review of de novo stand-alone error correction methods for NGS data. Wiley Interdisciplinary Reviews: Computational Molecular Science. 6(2):111-146. doi:10.1002/wcms.1239.

Article type: Advanced reviews

Objective Review of *de Novo* Stand-Alone Error Correction Methods for NGS Data

Article ID

Andy S Alic [1]

Institute of Instrumentation for Molecular Imaging (I3M), Universitat Politècnica de València, Spain

David Ruzafa

Departamento de Quìmica Fìsica e Instituto de Biotecnologìa, Facultad de Ciencias, Universidad de Granada, 18071 Granada, Spain

Joaquin Dopazo

Department of Computational Genomics, Príncipe Felipe Research Centre (CIPF), Valencia, Spain

CIBER de Enfermedades Raras (CIBERER), Valencia, Spain

Functional Genomics Node (INB) at CIPF, Valencia, Spain

Ignacio Blanquer

Institute of Instrumentation for Molecular Imaging (I3M), Universitat Politècnica de València, Spain

Biomedical Imaging Research Group GIBI 2^{30}, Polytechnic University Hospital La Fe, Valencia, Spain

Keywords

NGS, errors, correction, software, review

Abstract

The sequencing market has increased steadily over the last years, with different approaches to read DNA information, prone to different types of errors. Multiple studies demonstrated the impact of sequencing errors on different applications of Next Generation Sequencing (NGS), making error correction a fundamental initial step. Different methods in the literature use different approaches and fit different types of problems. We analysed a number of 50 methods divided into five main approaches (k-spectrum, suffix arrays, multiple sequence alignment, read clustering and probabilistic models). They are not published as a part of a suite (stand-alone) and target raw, unprocessed data without an existing reference genome (*de Novo*). These correctors handle one or more sequencing

[1]Correspondence to: asalic@posgrado.upv.es

technologies using the same or different approaches. They face general challenges (sometimes with specific traits for specific technologies) such as repetitive regions, uncalled bases and ploidy. Even assessing their performance is a challenge in itself because of the approach taken by various authors, the unknown factor (*de Novo*) and the behaviour of the third party tools employed in the benchmarks. This work aims at helping the researcher in the field to advance the state-of-the-art, the educator to have a brief but comprehensive companion and the bioinformatician to choose the right tool for the right job.

The Next Generation Sequencing (NGS) appeared in 2005 and since then its market has increased steadily, with various technologies being developed. The NGS has evolved faster than the the Moore's law in Computer Science, allowing us to sequence and assemble large genomes like the Loblolly Pine with 22 Gb[1] or the Norway Spruce with 20 Gb[2] for a reasonable cost in time and resources. However, there are many other species (e.g. the Amoeba Dubia with a 670 Gb estimated genome size[3], 200x human genome's size) that are still challenging to assemble. The errors introduced by the sequencing process are one of the main reasons NGS data has to be corrected before any further use. Multiple studies have demonstrated the impact of sequencing errors on different applications of NGS, making error correction a fundamental initial step.[4–7] There are many error correction tools in the literature that cope with different technologies and error types. However, to our knowledge, there is no complete, objective review of the modern methods that could help researchers, educators and users at the same time. There are benchmarks summarizing a number of methods, but there is none extensively focusing on the implementation, features and the overall domain (including challenges). Our work synthesises 50 *de Novo* stand-alone error-correction software. The Supplementary Material includes the description of the approach used to search the literature along with the inclusion criteria.

The article continues with the motivation (also containing a brief description of the sequencing technologies and various error sources), followed by a presentation of the correctors. Next, section "Discussion" presents some important points related to challenges faced by correctors and how their performance is assessed in the literature. Finally, we conclude our paper with some general remarks about the current state of the field of the error correction of NGS data.

We also introduce the concept of gradual recommendations. The recommendations are gradual, because they progress with the text and each one is based on the previous information. Section "Error Correction Software" includes general recommendations based on the features presented in table "Software Categories". The recommendations from section "Error correction in real projects" use as foundation the previous ones and extend the suggestions now that the reader has read about some real-world examples. Subsection "Recommendations" from section "Challenges" focuses on proposals taking into account the challenges that the correctors must address. Finally, subsection "Recommendations" from section "Testing" offers advice (now that the reader knows the methods,

where these have been used and what the challenges are) based on real-world performance using different metrics.

Motivation

The market size of the next generation sequencing was estimated at $2.5 billion in 2014.[8] Furthermore, Illumina managed to lower the cost of sequencing with its HiSEQ X to ∼$1000 (in 2015) for the human genome.[9,10] This price is quite an achievement when considering that not so long ago (2000-2003) the draft of the human genome costed about $300 million.[11] Overall, NGS has become widely used by the medical and scientific community not only for the basic biological research, but also in numerous applied fields such as medical diagnosis, forensic biology, virology and biotechnology. These are just a few clear proofs of the increasing importance of NGS in the world (not mentioning the increase in size of the segments, faster sequencing machines and improved quality of the generated data). This quickly evolving and advancing environment facilitated the development of a myriad of methods with different applications for the NGS data. One of the most important steps (usually the first) is the correction of errors, yielding many benefits for the ulterior ones as demonstrated in subsection "Benefits of Error Correction". Our reader may assume that an easy way to deal with the errors is to increase the coverage (i.e. add more sequencing data). While the increase in coverage indeed helps the correction process, there are still many challenges that the correctors must address (especially in *de Novo* sequencing). Furthermore, an increase in coverage comes with an increase in costs, sequencing/processing time and storage requirements.

After an extensive literature search (see approach and details in the Supplementary Material), we selected a number of 50 correctors. As one may expect, there is a tremendous amount of information scattered across these papers. We strive to summarize the deluge of information for an audience from many fields such as bioinformatics, biology, chemistry, computer science and others with an interest in NGS. Our aim is to help the researcher in the field of error correction by grouping the information and synthesizing the existing work. Secondly, our work also comes in handy for educators because it summarizes and presents the key points of the information found in the selected articles. We tried to present the information gradually, without an abrupt and direct presentation of the correctors (our readers are not expected to have an apriori deep knowledge of the domain). Finally, the actual users of the correction software can find Table 5 useful to choose the right tool for their specific requirements. The Supplementary Material contains an additional table with all the testing results from the reviewed articles.

Sequencing Technologies

The DNA sequencing was born in 1977 with the publication of the Sanger method[12]. This method implies a large amount of DNA as template for each read and needs an

independent PCR for each possible nucleotide. The PCRs are produced in presence of four deoxynucleotides and a single dideoxynucleotide, which stops the elongation. Once synthesized, the truncated DNAs are resolved by electrophoresis. During the synthesis reaction, a radioactive nucleotide (usually dATP - Deoxyadenosine triphosphate) is incorporated into the elongating strands that simplifies the determination of the sequence.

NGS methods are more efficient than Sanger sequencing in two different ways. On the one hand, in Sanger sequencing only 1 Kb (max) can be sequenced in a single experiment, whereas NGS is parallel by definition, allowing a throughput of hundreds/thousands of gigabases per run. Note that in this article Kb, Mb and Gb are the acronyms for kilobases, megabases and gigabases. On the other hand, the chemical reactions are usually combined with the signal detection in some versions of NGS, whereas in Sanger sequencing they are two separate processes. Factors like the reduction of time, manpower and reagents in NGS lead to lower costs making it possible to do more repeats than with the Sanger method. This results in a more accurate, reliable sequencing and better coverage. In NGS, the first step is the DNA cleavage into short segments (or reads) with lengths depending on the particular sequencer used.

In this review we focus on NGS technologies based on sequencing by synthesis (SBS), using DNA polymerase/ligase enzymes to generate a complementary strand. As defined by [13], Pacific Biosciences is the only SBS approach which has a real time sequencing strategy. All the others are synchronously controlled as we shall see in the following sub-sections where we present a brief but comprehensive description of the sequencing chemistries of the five aforementioned technologies. Fuller *et al.* also divide the methods in single molecule based (Pacific Biosciences and Oxford Nanopore - not specified in [13]) and ensemble based (Illumina, Roche 454, Ion Torrent - not specified in [13] and SOLiD). The former sequences single molecules of DNA as they are obtained from the source, while the latter relies on the amplification (cloning) of DNA segments before starting the actual sequencing process.

The sequencing technologies explicitly supported by the methods from our review (ordered by the number of correctors supporting them) are: Illumina, Roche 454, Pacific Biosciences, Ion Torrent, Oxford Nanopore and SOLiD. Note that we put a strong accent on the actual DNA "reading" step, because this is the main step where sequencing errors are generated. For further and detailed information regarding the platforms and the entire sequencing process, please, check the Further Reading section where we listed some resources which cover the themes more in depth. Additionally, the interested reader can find more details about chemistry of SBS sequencing in [13].

Table 1 offers some information about a number of well-known sequencers.

Table 1: Information (as of November 2015) about sequencing machines as reported by the vendors themselves; SR stands for single read, PE stands for pair-end reads, MP stands for mate-paired reads, 1D stands for a segmenet read in one direction and 2D stands for a segment read both forward and reverse; Q or QV followed by a number represents the quality score for a base (depending on the reference the score can be in Phred[14] format or not)

Platform	Instrument	Unit	Reads / Unit (single reads)	Avg./Max Read Len. (bp)	Read Type	Quality	Error Type	Info
Illumina	HiSeq X Ten	Lane	375,000,000	150/150	PE	≥75% bp >Q30	mismatch	[15]
	HiSeq 3000/4000	Lane	312,500,000	150/150	PE	≥75% bp >Q30	mismatch	[16]
	HiSeq NextSeq 500 High-Output	Run	400,000,000	150/150	SR/PE	≥75% bp >Q30	mismatch	[17]
	HiSeq NextSeq 500 Mid-Output	Run	130,000,000	150/150	PE	≥75% bp >Q30	mismatch	[17]
	HiSeq 2500 High-Output HISEQ SBS V4	Lane	250,000,000	125/125	SR/PE	≥80% bp >Q30	mismatch	[18]
	HiSeq 2500 High-Output TRUSEQ SBS V3	Lane	186,048,000	100/100	SR/PE	≥80% bp >Q30	mismatch	[18]
	HiSeq 2500 Rapid Run	Lane	150,696,000	250/250	SR/PE	≥75% bp >Q30	mismatch	[18]
	HiScanSQ	Lane	93,024,000	100/100	SR/PE	≥80% bp >Q30	mismatch	[19]
	Genome Analyzer IIx	Lane	42,075,000	150/150	SR/PE	≥80% bp >Q30	mismatch	[20]
	Genome Analyzer IIe	Lane	42,075,000	150/150	SR/PE	≥80% bp >Q30	mismatch	[21]
	Genome Analyzer II	Flow Cell	50,000,000	75/75	SR/PE	≥70% bp >Q30	mismatch	[22]
	Genome Analyzer	Flow Cell	26,000,000	50/50	SR/PE	≥98.5% bp error-free	mismatch	[23]
	MiSeq Reagent Kit v3	Lane	25,000,000	300/300	SR/PE	>75% bp >Q30	mismatch	[24]

Table 1 Continued from previous page

Platform	Instrument	Unit	Reads / Unit (single reads)	Avg./Max Read Len. (bp)	Read Type	Quality	Error Type	Info
	MiSeq Reagent Kit v2	Lane	15,000,000	250/250	SR/PE	>70% bp >Q30	mismatch	[24]
Thermo Fisher Scientific	Ion Proton I	Chip	80,000,000	125[a]/200	SR	NA	indel	[25]
	Ion PGM 318	Chip	4,000,000	363[a]/400	SR	>99% aligned/measured accuracy	indel	[26]
	Ion PGM 316	Chip	2,000,000	333[a]/400	SR	>99% aligned/measured accuracy	indel	[26]
	Ion PGM 314	Chip	400,000	181[a]/400	SR	>99% aligned/measured accuracy	indel	[26]
	SOLiD 5500xl W	Lane	3,200,000,000	75/75	SR/PE/MP	NA	mismatch	[27]
	SOLiD 5500 W	Lane	1,600,000,000	75/75	SR/PE/MP	NA	mismatch	[27]
	SOLiD 5500	Lane	120,000,000	75/75	SR/PE/MP	up to 99.99% ≥QV40	mismatch	[28,29]
	SOLiD 5500xl	Lane	200,000,000	75/75	SR/PE/MP	up to 99.99% ≥QV40	mismatch	[28,29]
PacBio	RS II	SMRT Cell	55,000	18,181[a]/60,000	SR	>99.999% (QV50)	indel	[30]
	RS	SMRT Cell	22,375	4,575/20,848	SR	NA	indel	[31]
Roche 454	GS FLX Titanium XL+	Run	1,000,000	700/1000	SR	99.997% Consensus accuracy	indel	[32]
	GS FLX Titanium XLR70	Run	1,000,000	450/600	SR	99.995% Consensus accuracy	indel	[32]
	Junior	Run	100,000	400/-[b]	SR	99% accuracy at 400bp	indel	[33]
	Junior+	Run	100,000	700/-[b]	SR	99% accuracy at 700bp	indel	[34]
Oxford Nanopore	MinIon	Run	4,400,000	9,545[a,c]/300,000	1D/2D	96% accuracy base calling	indel	[35]
	PromethION	Run	26,000,000	9,846[a,c]/300,000	1D/2D	96% accuracy base calling	indel	[35]

[a]Calculated from max throughput divided by max number of reads,[b]Unspecified max length, [c]Values from Fast Mode

Illumina/Solexa

Figure 1: Main sequencing steps for Illumina.

The Illumina reads receive adapters at their ends. These adapters attach themselves to the respective complementary adapters, with the latter hooked on a board with many variants of (complementary) adapters placed on a solid surface. Next, a segment is cloned by PCR amplification, creating a spot with many copies of the same initial read. The last step before the actual sequencing splits each read in the two complementary strands. Once the board contains only the single stranded reads held in place by adapters, fluorescent labelled, terminated nucleotides and DNA polymerase are added as a mix on the board (Fig. 1, Step 1). Fluorescent bases produce unique colours for each matching base. A polymerase is a protein that rebuilds the double helix starting from a single stranded template. It adds the complementary base for each of the template's composing nucleotides. Due to the terminated property of the free nucleotides added earlier in the mix, the polymerase attach to one and only one base per cycle (Fig. 1, Step 2). The sequencer registers the colour of the latest incorporated nucleotide for each read by taking a snapshot of the board (Fig. 1, Step 3). The process continues with the elimination of the terminator with the fluorescent label and the starting of a new cycle. The number of cycles gives the length of the read, with all reads normally having the same length. Using the snapshots, the sequencer determines the nucleotides composing a read (Fig. 1, Step 4).

Roche 454

As in Illumina's method, the 454 reads pass through a PCR amplification step and bind to adapters for which the complementaries lay hooked on a bead. Roche 454 uses the same fluorescent signalling to read the attached nucleotides. Therefore, the addition of each nucleotide releases a light signal. The main difference consists in the approach taken at each cycle. Instead of adding a solution containing all four fluorescent and terminated bases, the sequencer adds the solution with one and only one type of bases without the terminator. As a result, a variable number of bases can bind on the read at each cycle. The intensity of the signal represents the number of nucleotides added in each cycle. Roche tries to reconstruct entire homopolymers (runs of identical bases) at each cycle to save time. As an example, at Step 2 from Fig. 1, the sequencer adds two Cytosines (instead of one like in Illumina's case) in the depicted cycle. Generally, the sequences generated by 454 instruments have different lengths, because different numbers of bases are incorporated at each cycle.

Ion Torrent/PGM

Unlike Illumina and 454, Ion torrent sequencing is not based on the detection of optical signals. Instead, it takes advantage of the release of protons (H^+) following the addition

of deoxynucleotides to the DNA strands by the DNA polymerases. The fluctuation in the pH of the solution can be easily measured and, using its level of acidity, the instrument can determine how many bases have been attached in each cycle. The bases are identified as in the case of 454.

Abi SOLiD

In this case, the reads are used to prepare clonal bead populations.[36] Instead of binding one nucleotide or a homopolymer per cycle (like the previously described sequencers), ABi uses fluorescently labelled di-nucleotide probes. Instead of individual bases, SOLiD encodes the transition between bases. At each ligation step, four DNA primers attached to different fluorescent dyes (out of the 16 DNA possibilities) are added to the reads that match to the complementary DNA primers on the bead. Next, the fluorescent part is read and afterwards cleaved from the probe. The sequencer repeats this cycle of ligation/reading/cleavage as many times as needed in order to obtain a read of a certain length. In each cycle two positions of every five are determined. Once the sequencer has executed enough cycles, it resets the template with the primer going one position backward by removing one base of the primer. In order to determine the complete sequence of the read SOLiD sequencers perform this resetting step five times. As the primer is moved one base backward, the sequencer reads each base twice, improving the robustness.

Pacific Biosciences

Pacific Biosciences uses a single-molecule real time (SMRT) sequencing approach.[37] It employs the same fluorescent labelling as the previous technologies, but instead of executing cycles of incorporating nucleotides and taking snapshots, it detects the signals in real time, as they are emitted when the incorporations occur (using a zero-mode waveguide system[38]). Like Illumina, Pacific Biosciences uses all four bases at the same time floating in the mix. It has a bead with many wells having a diameter between 70 and 100 nm, lower than the wavelength of the visible light. Due to the physical properties of the fluorescent additives, light is needed in order to make the fluorescent dye glow. Owing to this requirement, the bottom of the wells is illuminated, but due to their very small diameter, the light intensity decays exponentially along the wells, creating a shadow zone. The wells have a DNA polymerase attached to their bottoms (the illuminated zone) which rebuilds the DNA complementary strand of the DNA segments floating in the mix. Each time a nucleotide is added, the fluorescent dye is cleaved. As a result, two consecutive signals do not overlap because the sensors only record the non-cleaved fluorescent dyes as the previous ones move up in the well into the shadow zone. This approach does not require cycles, because each polymerase works independently of the others.

Oxford Nanopore

As its name suggests, this sequencing technology uses very small nanopores (allowing one nucleotide at a time) to read the DNA sequences.[39] The idea of using nanopores to decipher the DNA's code has been around since 1989, but it was not viable, until 2010 when a DNA polymerase capable of attracting the DNA to the nanopore was discovered[40]. The main components of this technology are the protein nanopores, resembling those found on the cellular membrane. These structures are inserted into an electrical resistant artificial membrane onto which an electrical potential is applied. Owing to the flowing of the potential only through the aperture of the nanopore, any molecule passing through generates a variation in the current, resulting in a specific signature. Using the previously described process, the sequencer is able to decode the DNA (or RNA or proteins). In order to get the DNA segments to pass through the hole, the segments are mixed with copies of a carrier enzyme. These carrier enzymes attach to the DNA strands. They are pulled to a nanopore where the DNA is unzipped (if necessary) and the resulting single strand passes through the aperture, producing variations in the potential. One interesting feature is the capability to sequence both strands of the DNA segment using the same nanopore, generating the so-called 2D reads (5'-3' and its complementary 3'-5' strands linked together). In order to do this, the DNA must have a hairpin structure at the end to keep the two templates together after unzipping. This way, once the first strand has passed through the hole, the complementary one is pulled through.

Errors in NGS

There are four types of basic sequencing errors: insertion, deletions, mismatches and uncalled/unknown bases (or Ns).[41] The differences in the sequencing process of the aforementioned technologies lead to different types of errors. Table 1 lists the predominant error type for each NGS technology. This constitutes an important factor when choosing the values of the parameters for the correctors. Mismatches are prevalent in Illumina and SoLID, while indels constitute the main error type in Roche 454, Ion Torrent, Pacific Biosciences and Oxford Nanopore.[5,42–46] More details about the types of errors in the aforementioned technologies and practical experiments appear in[5,44,47]. The ensemble-based methods are prone to pre-sequencing errors (generated by the library preparation method and the choice of primers)[48], unless PCR-free kits are used[49–51].

Owing to its one nucleotide incorporation per cycle, the Illumina sequencers avoid insertions and deletions almost entirely.[5,44] Sleep *et al.*[52] describe the substitution errors for various Illumina sequencers. They found that the percentage of error increases towards the 3' end of the reads[7,53] because of a phenomenon called dephasing/phasing. It is caused by the fact that an error generated at one cycle affects the next cycles, hence the increased number of errors towards the end.[52] This same phenomenon is the main cause for the limited length of the reads in all ensemble-based SBS methods where some strands in a group of clones may fall behind resulting in a de-synchronization of the emissions of each clone in a group.[13] Another important cause of sequencing

errors is the crosstalk arising due to the overlap of dye emission frequencies. The Illumina Genome Analyzer® uses a red laser to read A and C and a green laser to read G and T. As a result, the Genome Analyzer® produces many substitution G $\rightarrow$ T and C $\rightarrow$ A.[7,54] Related to the previous cause for the MiSeq® sequencer, Schirmer *et al.* determined that A $\leftarrow\rightarrow$ C substitution errors appeared more often than G $\leftarrow\rightarrow$ T (red laser/filtering problem). They also studied the relation between the position of bases in a read and the quality scores for Illumina MiSeq data. Generally, errors occurring between the start and the middle of the reads had much higher quality scores, than those in the second half of a read. Furthermore, the authors found several 3-mer motifs usually preceding substitutions and indels, resulting from the selection of primers and library design. The estimated error rate for Illumina sequencers is between 1 and 2.5%.[7,55,56]

Following the brief description of the Roche 454's sequencing approach, it becomes clear that, analogous to Illumina, some nucleotides are misclassified. Furthermore, the exact length of the homopolymers cannot be exactly determined each time[43,57], with the sequencer introducing:

- insertions (when recorded homopolymers are longer than real ones)
- deletions (when recorded homopolymers are shorter than the real ones)

Luo *et al.*[58] demonstrate the relation between homopolymers and their length, where pyrosequencing (Roche 454 FLX Titanium®) loses accuracy as the length of the homopolymer increases. Gilles *et al.*[59] found a chemistry-related source of errors termed the CAFIE effect (carry forward and incomplete extension). Carry forward is generated by the inability to fully clean a well (unincorporated nucleotides are not removed) after a cycle. As a result, during the next base flow nucleotides are prematurely incorporated to specific sequence combinations, hence generating noise. The incomplete extension effect appears when some DNA strands on a bead miss the nucleotides incorporation at a certain flow cycle. They must wait for the next flow cycles, but they are already out-of-phase with the other strands. The Roche 454 GS Junior® has an indel error rate of 0.38 per 100 bases.[6] Gilles *et al.* report a mean error rate of 1.07%.

The Ion Torrent sequencing approach has indels as dominant error type.[46] Bragg *et al.* also observe that insertions appear more often than deletions and that (in contrast to Illumina) indels are an order of magnitude more likely to be generated. Due to the similarity in the sequencing idea between Ion Torrent and 454, it becomes clear that homopolymers pose a problem for this technology too.[6] The sequencing accuracy of the reads steadily decreased from their start to their end.[6,46] Loman *et al.* observed that in comparison to 454 GS Junior®, the Ion Torrent PGM® is less accurate when dealing with homopolymers (accuracy of 60% for homopolymers with six or more bases). Furthermore, for homopolymers shorter than two bases, insertion is the main error type, but the situation changes with the increase in length of the homopolymers where deletions become the norm.[46] For long homopolymers (more than 14 bases), Ion Torrent does not generate reads at all.[60] The same study reinforces the problem with homopolymers by mentioning the inability to predict the correct number of bases for

homopolymers longer than eight bases. The observed error rate is 1.78% (all types of errors) in[60], between 1.68% and 4.86% (all types of errors, depending on the used kit) with 96-97% of them being homopolymers errors in[46] and 1.5 indels per 100 bases in[6].

Pacific Biosciences generates longer reads than other sequencing technologies, but the error rate is still high.[61] The errors seem to be uniformly distributed and independent of the sequence context.[62] The same authors and[63] suggest that Pacific Biosciences is more susceptible to insertions than to deletions. Currently, the error rate for Pacific Biosciences is between 15% and 20%.[63,64]

Oxford Nanopore is an emergent technology, generating long reads with a small and portable device (the MinION®[65]). It is still in development, but there are some publications studying the sequencing results[66,67]. The accuracy is still low, with insertion as the predominant type of error[42]. Goodwin *et al.* report a very high error rate, between 25% and 40%.

GC Content

It is widely accepted that extreme base composition of some regions poses a problem for sequencing technologies.[5] For example the GC content (rich and poor regions) is often a source of bias and unevenness in coverage. The coverage is an extremely important aspect of the NGS, as it is needed to successfully process the output data as we discuss in section "Low-Coverage Regions and Uniformity". The problem is even more important as the bias can be introduced during the library preparation step, before the actual sequencing process.[5] This holds true for ensemble-based SBS technologies where the amplification step (emulsion PCR or bridge amplification) generates (much) lower coverage on the very GC-rich and GC-poor regions[5]. Quail *et al.* consider that this problem appears for Ion Torrent due to its double amplification step (library and template). They managed to lower the bias by using the Kapa HiFi enzyme for the fragment amplification. Furthermore, the bias can be eliminated by using PCR-free preparation kits for Illumina[49], Ion Torrent[50] and 454[51].

Ross *et al.*[5] provide an excellent measurement of the biased caused by the GC regions. They use the genomes of four species as the correct and trusted source and compare it with the data generated by Illumina MiSeq®, Ion Torrent PGM®, Pacific Biosciences RS® and Complete Genomics®. Fig. 3 and fig. 4 from the aforementioned article depict the strong variation introduced by the these GC extreme zones. Pacific Biosciences sequencing seems to obtain better sequencing results, because of its lack of amplification before sequencing[5], but bias still plagues this technology (slight but noticeable) when faced with genomes with GC-rich regions like S. Aureus.[60] The Pacific Biosciences RS®, like the Illumina MiSeq® and Ion Torrent PGM®, is also susceptible to dissociation of fragment ends in adapter ligation.[5] High and low GC content seems to influence Oxford Nanopore too as the coverage is more variable than in zones with a 20%-60% GC content. As a result, this extreme GC content partially motivates the lack of coverage for certain regions in the genome.[68]

Benefits of Error Correction

The most important application of error correction is in the field of genome assembly where the input data is corrected before the actual assembly. Many error correction publications include tests with assemblers and real data. Various assembly metrics demonstrate the need of error correction to generate meaningful assembly output (see subsection "Assembly" from section "Testing").[69]

A second application is re-sequencing, where multiple samples from an organism with an already known genome are sequenced. The main purpose of this operation is to compare the variability among different genomes from the same species. Another purpose is the comparison of datasets from the same organism sequenced using different technologies or sample preparation procedures. Re-sequencing indirectly uses the same metrics like gain and accuracy which compare the corrected reads against a reference genome.[4,70]

Thirdly, the authors in[71–74] stress the impact of error correction on short reads aligners. Errors are dangerous because they can cause an aligner to miss the real locus of a read in the reference genome. Furthermore, in the case of repetitive regions, a faulty read from a unique path in the genome can end up in multiple locations, provided it matches the repetitive region due to the errors.

Another affected application of NGS is the detection of SNPs. Normally, an aligner maps the reads against a reference genome to search for variants, but the errors in the reads can be misleading, increasing the total number of differences.[7] Furthermore, as the distribution of SNPs is not uniform, a region can have a high density of SNPs. Errors have a higher impact in these areas.

Additionally, there are other steps following sequencing that can benefit from error correction (e.g. identification of copy number variation or chromosomal rearrangement).[4] In conclusion almost any possible operation on NGS data benefit from the corrected input. Section "Error correction in real projects" lists a many real studies that used the correctors included in this review.

Error Correction Software

The following subsections deal with the correctors included in this review. The exact target of a corrector is not specified in most papers, the authors normally specifying DNA reads. The benchmarks performed in the same articles contain only datasets from whole genome sequencing (WGS) projects. One exception is **PAGANtec**[75] which works with transcriptome assemblies.

Technology support

Illumina is the market leader, with a 70% market share.[8] The majority of software in our review support Illumina (and in some cases other technologies at the same time),

fact that reinforces the status of the aforementioned company. The second major player is Roche with its 454 line of sequencers which, despite its shutdown of its technology in 2013, is still widely used (officially supported until 2016).[76] As a matter of fact, **Karect**, one of the most recent error correctors (2015), handles indels errors from 454. We can see an increase on the support of Pacific Biosciences, but all the current correctors rely on an additional dataset on a different technology to perform the correction. Ion Torrent is not widely supported as of now, but since the prevalent errors for this technology are indels[5], the tools handling indels should also work with it. Finally, there is only one program that targets SOLiD color-space data, namely **HSHREC**.

In our review, we have found programs supporting more than one technology. Table 2 enumerates the technologies supported by all correctors (column "*Tech*"). Fig. 2 depicts the categories in which the correctors fall. All but one of the tools supporting only one technology work with Illumina and only target mismatch errors. **Hector** is the exception to the above rule, designed only for 454 reads, supporting indels. All Pacific Biosciences software focus only on Pacific Biosciences, but they use Illumina/454 reads for the cross-correction, therefore they are classified in a separate group.

There are several software tools handling multiple technologies which can tackle all types of errors. Our reader can determine the support for different types of errors by consulting table 2, columns "*N*" and "*Indel*". All programs support mismatches, therefore it is not mentioned in the aforementioned table. Some programs like **HSHREC** treat all datasets in the same manner with no special handling for different technologies (albeit **HSHREC**) has a special version which can correct colour space reads, as a different executable program). We included it in the first category because the base space version does not have a target technology.

The software with different profiles can be further divided in software using the same correction method for all technologies, but setting different values for parameters, and software with internal algorithmic modifications for a certain technology. For the first group, **Coral** is a perfect example since it uses the same algorithm to correct both Illumina and 454, but in case of Illumina the algorithm sets very high values for gaps, forcing mismatches only. **Blue** on the other hand has a flag for 454 to enable searching for homopolymers errors. **Karect** has generic support for multiple sequencing technologies, running in two modes, with indels or without.

A new approach is the cross-correction using a high-quality short reads dataset to correct a dataset having (much) longer, lower-quality reads. There are correctors targeting the very long reads (**LoRDEC**, **proovread**, **Jabba** and **LSC**) produced by Pacific Biosciences, 454 (**Blue**) and Oxford Nanopore (**Nanocorr**).

Figure 2: Classification using the technology support among correctors; Letters between paranthesises on the leaves used to group the algorithms in Table 2.

The latest review including software supporting indels is from 2013[77] and it does not include the latest additions to indel-aware software, like **Blue**, **Fiona**, **Pollux** or

Karect. The authors of the review stressed the need for better software solutions with indels support, as the results of the existing algorithms at that time (**HSHREC** and **Coral**) were not comparable to the Illumina-specific solutions.

Software Categories

We clustered the algorithms according to their core functionality, extending the work in[77,78]. Due to the length limitation of the article, the individual description of the correctors is located in the Supplementary Material. Table 2 summarizes some important features of the analysed software. We explain the information on the columns in the following sections.

The **K**-**S**pectrum **B**ased (**ksb**) software corrects the reads employing the k-mer spectrum.[79] A k-mer is a segment from a read with k-bases. The set of all k-mers of a read is generated by using a sliding window of dimension k. At each step, the window is shifted by one element and "the visible" segment of the read is added to the spectrum set. This is by far the most popular approach, used by 28 out of 50 correctors. Generally, the applications use the k-mer spectrum (Fig. 3) to decide whether a k-mer is correct or not. The error-free k-mers are those appearing in a number of reads entering a predefined distribution (a Gaussian in our example). Roughly speaking, k-mers appearing in a small number of reads are considered erroneous, since the coverage is not uniform, the k-mers in the low coverage areas are under-represented (more information about k-mers in section "K-mer")

Figure 3: Typical distribution of k-mers used by **ksb** correctors; Vertical axis shows the number of k-mers which appear in the number of reads displayed on the horizontal axis; First peak corresponds to erroneous k-mers which appear only in a few reads; Correct k-mers typically exist in a number of reads close to the coverage; K-mers found in many reads (right part of the spectrum) typically correspond to repetitive regions.

Suffix **T**rie/**A**rray **B**ased (**stab**) generally build a suffix structure with the common parts of the reads. These correctors try to locate inconsistencies in their path, while exploring the trie/array. Fig. 4 depicts an example where a low frequency of a divergent suffix signals a possible error case. Normally the reads on a trie follow the same path, but it happens to diverge at some point. A corrector has to decide if the split is an error or not. Fig. 4 a) depicts a divergence point (different nucleotide) where the frequency of one of the resulting paths is very low ($<<$ k/2) and the bases of this path after the divergence point till the leaves are exactly the same as for the path with the frequency $\lesssim$ k/2, hence it is an error. For the other case where the frequencies are the same (k/4), a SNP (Single Nucleotide Polymorphism) causes the divergence. Otherwise, the common path till the divergence point is a repetitive region in the genome, followed by the unique region for each path. The trie in Fig. 4 b) with the erroneous base in bold and italic exemplifies the branching caused by an error. The \$ symbol marks the end of a suffix (a standard way of depicting suffix tries). It is crystal clear that due

to the low frequency of the suffixes containing the bad base, a corrector can isolate the error and can take a valid decision given enough coverage. Given the shortness of the reads in our example we consider one base to be sufficient proof of inclusion on one branch or another. As a result, for the suffix A***A***A\$ the third base will match with its counterpart from the suffix AGA\$ (the first base is the same since we are talking about the same family of suffixes), therefore ***A*** should be G. Next, the branch TA***A***A\$ triggers a warning for the corrector due to its low frequency of its sub-branch ***A***A\$. The problematic base is again surrounded by bases that match a sibling path (i.e. TA***A***A can be converted to TAGA, with the latter having a higher frequency), therefore it is safe to assume that ***A*** is in fact G. Finally, after analysing these cases, a corrector can support its decision by detecting the relation between the suffixes A***A***A\$ and TA***A***A\$ where the former is in fact included in the latter and the corrections on both sides have an even higher degree of validity when taken together.

Figure 4: Suffix trie example; a) An error on the rightmost path results in branch having a very low frequency ($<<$ k/2) compared with its sibling branch ($\lesssim$ k/2); b) Example of a trie for a very short genome with read TA***A***A having an error on its third position

Multiple **S**equence **A**lignment **B**ased (**msab**) software focuses on aligning the reads to identify the overlap between them (see Fig. 5). The methods use different algorithms (like Needleman-Wunsch in **Coral**[73]) to build a consensus from a set of reads that are likely to fit together. Generally, these methods cluster together a number of related reads (e.g. those having at least one k-mer in common, like **Coral**), which may belong to (as the corrector may wrongly include reads from other regions) the same genomic locus. Reads containing k-mers appearing in multiple loci or erroneous k-mers matching wrong locations will normally fail in the alignment process. Being part of the same region, a **msab** corrector can generate a multiple sequence alignment and try to determine and fix the anomalies in the resulting consensus. The example Fig. 4 b) demonstrates that given sufficient coverage, a corrector is able to group a number of reads, isolate the erroneous bases and make a decision if possible. In the above example, we are able to take a decision in every case. On the contrary, if we have a mismatch between the first position of the first read and the fifth position of the third read, the decision is not straightforward any more (if possible altogether). For instance, if instead of (A,A) - the correct version - the pair would be (A,N), the corrector would have to either ignore it or apply some kind of heuristic. An example of approach would be to convert N to A, since A is a valid nucleotide. On the other hand, this approach could be rendered useless by the use of quality scores where N has a very high score compared to A (the previously considered valid base may not be so valid after all). In this case it is up to the corrector to take the appropriate action using different approaches and the context of the problem.

Read **C**lustering **B**ased (**rcb**) methods use different clustering methods to group reads which fit together. This group resembles the **msab** one, but the algorithms in it do not generate an alignment, but search for reads that are similar and choose a consensus which is the correct form for all these similar reads. Fig. 6 shows a central read (having

Figure 5: a) Multiple sequence alignment of reads versus the (prospective) reference genome; b) Example of four read with the common k-mer "TTACGAA" and the four basic types of errors.

the most common part with all the others) and its satellites. For simplicity, we only exemplify the one difference case. In our example all four satellites have one distinct nucleotide each. A corrector should group them together as they present a high degree of similarity, hence they are in fact clones of the same read, but with errors. Fig. 6 b) is an example (extracted from the bibliography [74]) where the consensus read is the one with the highest frequency. The other reads differ from the main read by just one nucleotide and also have much lower frequencies. Please keep in mind that the algorithms included here do not perform a multiple sequence alignment to determine the correct read, they just group them by differences and search for a valid consensus, the error-free existing read.

Figure 6: a) Clustering approach for one reference read and four related having one difference each; b) Real example with the main read market in bold and the satellites aligned and with the different locus market with bold and italic.

Probabilistic **M**odels **B**ased (**pmb**) methods use the Expectation Maximization (EM) algorithm to determine the correct base at each position by calculating the likelihood of the existing variants at that specific position. Basically, the problem of error correction boils down to selecting the right nucleotide at a certain position where two or more reads overlap and there is more than one choice. The pmb software base their approach on the fact that this problem has unknown parameters (unobserved component), in this case the correct base. As a result, using the existing input data (observed component) and maybe more information (like the error rate), they try to generate a model (after multiple iterations over the same data). This model can say with a certain degree of trustiness of the correction of a certain nucleotide. The EM alternates between two steps, the E (guessing the probability) and the M (re-estimating the model parameters using the new probability), until it converges to the desired model. Figure 7 presents the basic algorithm. For an extensive explanation of the EM algorithm, the reader should check the article of Do and Batzoglou from [80]. Different algorithms in this category use different position comparison methods (position part of k-mer or read) and convergence points.

Figure 7: The EM algorithm initializes the probabilities of the bases before entering the loop where it alternates between E-step and M-step; Once the convergence threshold has been reached the method exits and enters the correction stage; The capital P represents the probability for a base to be the real one;

Please note that many methods can be included in more than one category. For instance, **Coral**[73] is listed in this review in the category of msb algorithms[77,78,81], but it also uses the k-mer spectrum to determine the related reads. The same case arises with **Premier**[82] and **Premier Turbo**[83] which use k-mers to update the probabilities for the variants on a position.

Table 2: The software and important features support; The columns have the following meaning: Par. Tech- the parallel technology (if any), k- whether or not a software make use of k-mers, Qual.- support for quality scores, N- support for uncalled bases, Indel- support for indels, Var. L.- support for variable length reads, Hzyg.- support for heterozygosity, Rep.- support for repetitive regions, Trim.- whether or not the algorithm incorporates trimming, Tech. Supp - the categories in which a software fits from Fig. 2

Type	Name	Par. Tech	Lang.	k	Qual.	Tech	N	Indel	Var. L.	Hzyg.	Rep.	Trim.	Tech. Supp
ksb	**CUDA-EC**[56]	CUDA	c++	y	n	Illum.	NA	NA	NA	n	n	y	a
ksb	**Reptile**[84]	OMP	Perl/c++	y	y	Illum.	→A	n	NA	n	n	n	a
ksb	**Quake**[7]	OMP	c++	y	y	Illum.	NA	n	NA	y	y	y	a
ksb	**Edar**[85]	NA	NA	y	n	All	NA	y	y	n	y	n	a
ksb	**Hammer**[86]	-	c++	y	y	Illum.	y	n	y	n	n	n	a
ksb	**REDEEM**[87]	-	c++	y	n	Illum.	n	n	y	n	y	n	a
ksb	**DecGPU**[88]	CUDA/MPI	c++	y	n	Illum.	NA	NA	NA	n	n	y	a
ksb	**CUDA-EC2**[89]	CUDA	NA	y	y	Illum.	NA	n	y	n	n	y	a
ksb	**Qamar**[45]	PThr.	c++	y	n	Sanger/Illum.	y	n	y	n	n	n	d
ksb	**Parallel Reptile**[90]	MPI	c++	y	y	Illum.	→A	n	NA	n	n	n	a
ksb	**BayesHammer**[91]	OMP	c++, python	y	y	Illum.	n	n	NA	n	y	y	a
ksb	**QuorUM**[92]	PThr.	c++	y	y	Illum.	NA	n	y	n	n	y	a
ksb	**RACER**[93]	OMP	c++	y	n	Illum.	→A	n	NA	NA	n	n	a
ksb	**Musket**[71]	PThr./OMP	c++	y	n	Illum.	y	n	NA	n	y	n	a
ksb	**Hector**[43]	OMP	c++	y	n	454	y	y	y	n	n	n	b

Table 2 Continued from previous page

Type	Name	Par. Tech	Lang.	k	Qual.	Tech	N	Indel	Var. L.	Hzyg.	Rep.	Trim.	Tech. Supp
ksb	**Lighter**[94]	PThr.	c++	y	y	Illum.	y	n	y	n	n	n	a
ksb	**HErCoOl**[95]	Threads	java	y	n	454/Ion	NA	y	y	n	n	n	d
ksb	**Trowel**[96]	Boost	c++	y	y	Illum.	NA	n	NA	n	y	n	a
ksb	**LoRDEC**[62]	PThr.	c++	y	n	PacBio	NA	y	y	n	y	y	a
ksb	**BLESS**[97]	-	c++	y	n	Illum.	n	n	n	n	n	n	a
ksb	**Blue**[98]	Threads	c#	y	y	Illum./454	y	y	y	n	y	n	c,e
ksb	**BFC**[99]	PThr.	C	y	y	Illum.	NA	n	y	y	n	n	a
ksb	**Scribble**[100]	NA	NA	y	n	Illum.	n	n	y	n	y	n	a
ksb	**PAGANtec**[75]	OpenMP	C++	y	n	Illum.	n	n	y	n	n	n	a
ksb	**ACE**[101]	OpenMP	C++	y	n	Illum.	y	n	n	y	n	n	a
ksb	**FADE**[102]	FPGA	Verilog	y	n	Illum.	y	n	n	n	n	n	a
ksb	**Pollux**[103]	-	C	y	n	Illum./Ion/454	y	y	y	n	n	n	d
ksb	Gu *et al.*[104]	NA	C++	y	n	Illum.	y	NA	NA	NA	NA	n	a
ksb	**Jabba**[105]	NA	C++	y	n	PacBio	n	y	y	y	n	y	c
stab	**SHREC**[55]	Threads	java	n	n	Illum.	n	n	n	n	n	n	a
stab	**HSHREC**[106]	Threads	java	n	n	All	y	y	y	y	n	n	d
stab	**PSAEC**[107]	PThr.	c/c++	n	n	Illum.	n	n	n	n	n	n	a
stab	**HiTEC**[4]	-	c/c++	n	n	All Subst.	n	n	n	n	y	n	a
stab	**MyHybrid**[70]	NA	NA	n		All	y	y	y	n	y	n	d
stab	**Pluribus**[108]	NA	NA	n	n	All	n	y	y	n	n	n	d
stab	**Fiona**[41]	OMP	c++	y	n	Illum./454/Ion	y	y	y	n	y	n	e
msab	**ECHO**[109]	Python/PThr.	Python/c++	y	y	Illum.	y	y	y	n	y	n	a
msab	**Coral**[73]	OMP	c	y	y	Illum./454	y	y	y	n	n	n	f
msab	**LSC**[64]	NA	Python	n	b	PacBios	y	y	y	n	n	y	c
msab	**CloudRS**[110]	Hadoop	Java	y	y	Illum.	NA	n	NA	NA	y	n	a
msab	Chung *et al.*[111]	Hadoop	Java	y	y	Illum.	n	n	n	NA	y	n	a

Table 2 Continued from previous page

Type	Name	Par. Tech	Lang.	k	Qual.	Tech	N	Indel	Var. L.	Hzyg.	Rep.	Trim.	Tech. Supp
msab	**proovread**[63]	Grid	Perl	n	y	PacBio	NA	y	y	NA	y	y	c
msab	**Nanocorr**[68]	Grid	Python	n	n	Nanopore	y	y	y	n	n	n	c
msab	**Karect**[112]	PThr.	C++	y	y	Illum./454/Ion	y	y	y	y	y	y	d
rcb	**FreClu**[74]	-	java	n	n	Illum.	NA	n	n	n	n	n	a
rcb	Sleep *et al.*[52]	-	NA	n	n	Illum.	n	n	y	n	n	y	a
pmb	**RECOUNT**[53]	NA	c++	n	y	Illum.	NA	NA	NA	n	n	n	a
pmb	**Premier**[82]	NA	NA	y	y	Illum.	n	n	NA	n	n	n	a
pmb	**Premier Turbo**[83]	NA	NA	y	y	Illum.	n	n	NA	n	n	n	a
pmb	**kGEM**[113]	NA	NA	n	NA	All	y	y	y	n	n	n	d

Recommendations

From table 2, it is clear that depending on the nature of the project some programs are better than others. For Illumina projects almost all correctors can be used. Although, as we shall see in the coming sections (mostly in section "Testing", the Illumina-only correctors offer a better performance on Illumina data when compared to multi-technology software.

From a computational resources point of view, the correctors written in a low-level language like C++ should be used. One must take this last advice with a grain of salt as the performance is highly dependent on the quality of the code and the algorithms used. Another very important aspect is the multi-core and multi-computer support. Nowadays, even the mobile phones are multi-core and the speed of CPUs has hit a hard limit, therefore any piece of software capable of scaling on multiple cores should be preferred over the others. This scalable applications are very useful when the time frame is very short. Furthermore, the same programs win when testing multiple combinations of parameters at the same time and running multiple instances of a single threaded program is not an option (e.g. when the user has to run the next instance of the program with a combination of parameters based on a previous run). Lastly, the multi-threaded programs would normally consume less memory than multiple single-threaded instances running at the same time. The best example for this last observation is an OpenMP corrector creating a k-mer spectrum. An optimized multi-threaded program would create just one structure to keep the k-mers and their count and it would allow thread-safe access to the structure. An optimized single-threaded program using the same mechanism would avoid the locks but for multiple instances, the same structure would be replicated as many times as instances are running.

Error correction in real projects

The software presented so far was used in many research projects. This section is the extension of the motivation, where we present instances where scientist successfully employed error correctors. Hopefully, this section opens the door for further usage in comparable or new cases. We concerted our efforts to find real life biological projects where the utility of correctors is demonstrated by practical use. In contradiction to what we found in the articles accompanying each corrector, some real life projects use the correctors for additional applications like mithocondrial genome correction [114] and RNA-seq [115–118]

Table 3: Work using the correctors included in the current review

Year	Study Description	Where
	Quake/Illumina	
2012	Exomes comparison of C. Carpio & D. Rerio	[119]
2012	GAGE, evaluation of genome assemblies/algorithms	[69]

2013	Salinarchaeum HArcht-Bsk1T genome	[120]
2013	L. Arenae draft genome	[121]
2013	C. Sinensis draft genome	[122]
2013	Genetic variants in C. Sinensis	[123]
2013	Genome-wide mutations in diploid yeast	[124]
2014	Results of correction of heterozygous NGS	[125]
2014	O. Sativa de novo assemblies, novel gene space aus/indica	[126]
2014	Study of hydrocarbons production from fatty acids in Cyanobacteria	[127]
2014	Genetic diversity in P. Pacificus from population-scale resequencing	[128]
2014	Genetic parameters estimation and response to selection in breeding program of M. Galloprovincialis	[129]
2014	Metagenomic characterization of C. Defluviicoccus tetraformis	[130]
2014	Prediction of antibiotic resistance by gene expression profiles	[131]
2014	Methicillin resistance in S. Aureus	[132]
2014	De novo creation of repeat libraries from whole-genome NGS reads	[133]
2014	Aerobic fungal degradation of cellulose	[134]
2014	B. Tryoni draft genome	[135]
2014	Genome reorganization	[136]
2015	P. Vulgata/P. Lamarcki draft genomes	[137]
2015	The domestic dromedary genome	[138]
2015	The brown kiwi genome	[139]
2015	Comparative Genomics of S. Pyogenes M1	[140]
2015	Approach for Identification and Characterization of Foodborne Pathogens	[141]
2015	P. Glaucus complete mitochondrial genome	[114]
2015	Mechanisms for Speciation and Caterpillar Chemical Defense	[142]

BayesHammer/Illumina

Year	Study Description	Where
2014	GABenchToB, assembly benchmark for bacteria genomes	[143]
2014	C. Burnetii genome	[144]
2014	P. Atrosepticum genome	[145]
2014	Hidden diversity in honey bee gut symbionts	[146]
2014	S. Lemnae draft genome	[147]
2015	Discovering Natural Products from Cyanobacteria	[148]
2015	Characterize the metabolism of M. Thiooxydans L4 in the marine environment	[149]
2015	Utilization of alginate and other algal polysaccharides by marine Alteromonas macleodii ecotypes	[150]
2015	Genome-Wide Re-distribution in Active Yeast Genes	[151]
2015	Study of the metabolome of M. Producens JHB	[152]

Reptile/Illumina

2014	Decrypting cryptobiosis-analyzing anhydrobiosis using transcriptome sequencing	[115]
2015	SNP genotyping and population genomics from expressed sequences	[153]
	HSHREC/Illumina,454	
2014	Decrypting cryptobiosis-analyzing anhydrobiosis using transcriptome sequencing	[115]
	BLESS/Illumina	
2014	Transcriptome, sequence polymorphism, and natural selection in P. Eremicus	[116]
	Blue/Illumina	
2014	S. Scitamineum genome	[154]
	Coral/Ion Torrent[a], Illumina[b], 454[c]	
2014	GABenchToB, assembly benchmark for bacteria genomes[a]	[143]
2014	Global gene expression in the exocarp of developing P. Avium L.[b]	[117]
2015	Comparative genomics/gene expression applied on P. Xuthus and P. Machaon genomes[c]	[155]
	DecGPU/Illumina	
2013	Genomic analysis of S. Dulcamara	[156]
	Echo/Illumina	
2012	Pipeline for small RNA-seq data analysis	[118]
2014	Results of correction of heterozygous NGS	[125]
2014	Assembly/annotation for T. Pratense	[157]
	Freclu/Illumina	
2011	MicroRNA-mediated gene regulation role	[158]
2011	Purification of monocyte subsets from H. Sapiens blood and their transcriptomes analysis	[159]
2013	Identification of functional cis-regulatory elements	[160]
	Hector/454	
2015	Triple-negative breast cancers in patients with no BRCA1 or BRCA2 mutation	[161]
	Lordec/PacBio,Illumina	

2015	De novo tandem repeat detection using short&long reads	[162]
	LSC/PacBio,Illumina	
2014	D. Officinale genome and genes analysis	[163]
2015	Detect fusion genes, determine fusion sites and identify and quantify fusion isoforms	[164]
2015	S. Miltiorrhiza transcriptome and tanshinone biosynthesis insights	[164]
	proovread/PacBio,Illumina	
2015	Characterization of venom toxin-encoding genes in E. Coloratus	[165]
	QuorUM/Illumina	
2014	P. Taeda reference genome	[1]
	RECOUNT/Illumina	
2012	Brain tumor glioblastoma-derived neural stem cells transcriptome analysis	[166]

Recommendations

Continuing the discussion from subsection "Recommendations", section "Error Correction Software", we can now see the role of correctors in real projects. **Quake**[7] is the most used corrector as it targets the sequencing technology with the largest market share. Furthermore, as we shall see in section "Testing", it has a good level of correction. As a general rule, Illumina data should be handled by Illumina-only correctors since they should be better tuned for the technology than their general counterparts. There are exceptions like **Coral**[73] that is used for both Illumina and 454. The same software is utilized with Ion Torrent for which there is a generic support. Generally, the software supporting all types of errors can be used with unsupported technologies, but the user must understand that the result might not be what expected. In the above case, Ion Torrent is somewhat similar to 454, hence **Coral** works. In the case of datasets from multiple technologies, one can use more than one corrector for each technology as Wang *et al.* did[115]. Another possibility is to use a cross-corrector like **Blue**, **LorDEC**, **proovread** and **LSC** where instead of stacking up all the reads from multiple technologies, one can use one technology to correct the other. The correctors are used in many types of projects with the obviously most targeted being assembly. Variant calling and different transcriptome studies are also very common in the existing projects. A very important fact emerging from table 3 is the range of genomes data tackled with the stand-alone correctors. The size and complexity ranges from bacteria (S. Pyogenes) to mammals (Dromedary) and plants (Loblolly Pine).

Discussion

In this section of the paper, we discuss the most important topics for the error correction state of the art. It starts with the general problem of errors in NGS data, followed by the key features of the methods and ending with the main testing approaches.

Challenges

Data Preparation and Post-processing Steps

There are cases in which the input data must pass through some additional pre-processing steps like the conversion to a certain format. **Blue** performs a preparation step to generate the k-mer spectrum. **Reptile** has a pre-processing step, to separate the reads from their quality scores and to filter reads containing ambiguous characters. **CloudRS** converts the FASTQ input file to a specific format. Furthermore, it must upload the converted input to the Hadoop cluster and download the result locally when the job has finished. **HSHREC** generates the corrected output files without the initial descriptions of the reads. Some tools may need this information for further processing like SolexaQA++[167] which generates statistics from multiple technologies. The dataset requires a post-processing step (that the user must implement) to restore the initial information. Secondly, **HSHREC** generate two files, one containing all corrected reads and the other the skipped reads. Generally, the output of the error correction tools is FASTA/FASTQ and it does not require any explicit processing.

K-mer

K-mers Handling: Many methods base their decision on k-mers and apply different techniques to deal with the memory limitation and CPU requirements. **BLESS** uses the hard-drive to store the k-mers during the counting.[168,169] **RACER** encodes the bases in a k-mer as a 2-bit representation to save memory. A newer version of **Quake** integrates Jellyfish[170] to count k-mers instead of its own implementation to stay competitive against the more recent algorithms. It also provides a distributed approach for those cases in which the local memory is not enough to handle the k-mers. To speed up the k-mer spectrum generation,[90] implements a parallel counter.[88] use GPGPUs to generate the k-mer spectrum. A rare feature is the support for variable k-mers for grouping reads as in[110], where the corrector uses a wildcard based k-mer.

K-mer Size Selection: The value of k is extremely important.[7,82] A too low value for k would result in many k-mers appearing in most reads, thus joining in groups reads without any real relation. On the contrary, very large values would generate too many unique k-mers, which also have a higher probability of including more errors, therefore introducing noise into the grouping. Long k-mers may also require more RAM memory.

The analysed methods set the size of the k-mer by: accepting a user value, having a fixed default value and/or performing automatic selection. **Hammer** requires the user to set it. **Quake** and **ECHO** define a formula to determine the optimum value as presented in Table 4. **Reptile** considers $10 \leq k \leq 16$ enough for microbial genomes, fitting the spectrum in less than 4 GB of RAM. **Coral** uses a default value of 21 for the k-mer. **Hector** and **Musket** require both the k-mer length and the total estimated number of k-mers from all reads. **HiTEC** and **Fiona** automatically identify the optimum k-mer length at each step. **CloudRS** stacks reads using a wildcard based 25-mer and, later in the correction procedure, a fixed 24-mer. The k-mer size should be odd in order to avoid palindromic k-mers.[171] The software using k-mers in their pipeline are market in table 2, column "*k*".

Table 4 Formulas to determine the k-mer size for non-automatic k-mer determination; N = Genome length, l = read length; p = probability that a random k-mer appears in a random string of length N, using the alphabet {A, C, G, T}; N_s = number of unique solid k-mers as reported by BLESS

Formula	Where
$k = \log_4 200N$	[7,70]
$k \geq \lceil \log_4 N \rceil$	[73]
$4^k > N$	[84]
$k = \lfloor l/6 \rfloor$	[109]
$k = \log_4 2Np^{-1}$; $p = 10^{-4}$	[95]
$N_s/4^k \leq 0.0001$	[97]

K-mer Distribution: Software relying on correct and erroneous k-mers tries to fit the k-mer spectrum on a certain distribution. The correctors compute the histogram with the frequencies for each k-mer in the set of reads. A valid estimation tries to model the initial, complex distribution as a combination of multiple, simpler distributions.[7] **Quake** divides the solid k-mer distribution in a combination of a normal and a Zeta distribution and it considers (like **BLESS**) the weak k-mers to follow a Gamma distribution. **Lighter** assumes a Poisson distribution, like **Fiona**. **REDEEM** models the k-mer distribution as a Multinomial one. For 454, **Hector** encodes homopolymers using the base and the multiplicity. The authors observe that the distribution of the original reads tends to be unimodal. With the encoding applied in the homopolymers space, the distribution of the homopolymers spectrum is analogous to the one (bimodal) obtained by **Musket** in base space. The same authors conclude that, generally, the homopolymers spectra are bimodal.

Coverage Cut-off and K-mer Distance: K-mer based error correction methods can cut the k-mer histogram to remove k-mers with too high or too low frequencies, which normally reduces the noise caused by highly-repetitive regions or singular errors. Some algorithms automatically compute the best cut-off value, allowing users to override this value. **Quake** uses the Broyden-Fletcher-Goldfarb-Shanno method to calculate the histogram cut-off, but this method fails when the curve of the distribution is not smooth enough.[71] The authors of **Musket** empirically determined that the lowest count for a k-mer around the valley can be a good cut point. **Musket** also has an option to use a user-provided value. **Trowel** uses a different approach by using the contiguity of high-quality-scores bases instead of the coverage. It also expands the trusted k-mer set, adding new k-mers after they are corrected. **QuorUM** bases its decision on the quality of bases from k-mers, therefore all bases in a solid k-mer must have a quality greater than a threshold. **RACER** uses an internal threshold to deem a k-mer followed or preceded by a certain base, either as solid or weak. **Hammer** and **Reptile** create a Hamming graph for the array of all k-mers, locating groups of similar k-mers that only differ in a few positions, and then collapsing all those k-mers into a consensus k-mer. To improve memory consumption, **Lighter** uses a random method to decide whether to store or not a k-mer, assuming that a correct k-mer appears multiple time in a dataset, thus the chance of being selected is high.

Repetitive Regions

In general, the problem of repetitive regions cannot be tackled by considering individual reads or k-mers in isolation. Some argue that in the case of highly repetitive genomes, a sequencing error has a greater probability to change a solid k-mer to another solid k-mer.[7] They calculated the percentage of all one base mutations for a k-mer k that will convert k into a sequence which also exists in the genome. The results show a 2.25% for E. Coli and 13.8% for human chromosome 1, with a 15-mer and a 18-mer respectively. Increasing the k-mer length up to 19 did not significantly change the result, dropping the percentage to 11.1% for the H. Sapiens' first chromosome. The different percentages obtained for the two organisms result from the higher complexity of the human genome. **REDEEM** was specifically designed to handle repeats. The main problem with repetitive regions is the similarity of two sequences which are reside on different loci of the genome. A corrector may try to convert them to a consensus, hence destroying the existing valid zones. These wrongly fixed reads would prevent an assembler from correctly composing the real genome (or make it generate chimeric assemblies). It is a real problem for highly-repetitive genomes (plants).[77] Furthermore, misreads in repetitive regions can cause an abnormal high frequency of a k-mer[172] which could result in an erroneous classification as solid by some correctors. The methods based on multiple sequence alignment are more resilient to challenges posed by repeats, although they do not totally solve the problem.[109] The relationships between reads can tackle to some extent some small repetitive regions because of the higher length of the analysed strings compared to k-mers. Moreover, a read from a repetitive region has a higher probability to enter the right group provided it shows enough dissimilarity with other repetitive regions. **Fiona** implements a filter to remove

suffixes with an unreasonable high frequency and supports tandem repeats. **Blue** addresses the problem of repeats by evaluating alternative fixes for a read (it works in the case of significant differences among reads). **proovread** takes into account the loci on the long reads where large blocks of short reads map, collecting many reads. In contrast, non-repetitive regions may not even participate in alignments, because of the uniqueness resulted from the Pacific Biosciences's high error rate.[110] makes use of a high frequency k-mer filter to avoid stacking reads from repetitive regions. Column "*Rep*" from table 2 contains the support for repetitive regions in the correctors.

Ploidy

A corrector must distinguish between errors and variants. Since most error correctors where tested on bacterial genomes, the information on the behaviour of most tools are restricted to the haploid case. The support for heterozygosity is stated on column "*Hzy.*" on table 2. Authors of[52] apply a smoothing technique to avoid removing zones with only biological variants. They build a tree using the frequency of reads, and consider a true variant as a sequence appearing with a high enough frequency compared with the parent sequence. Their decision is based on the fact that the frequencies of a variant should be much higher than the ones from the sequencing errors. The authors of **ECHO** explain a modification to support a diploid genome with homozygous and heterozygous genotypes. Their approach is to consider a uniform distribution over all possible genotypes. They skip a correction if the estimated coverage is much greater than the expected coverage at an analysed locus.

Read Trimming and Splitting

To avoid the propagation of errors to the next steps, the correctors may eliminate bases from both ends of a read, which can be considered a complementary method to reduce errors.[106] Users should take great care in using trimming, because it can heavily influence the next steps like assembly, where the final result may become fragmented[92] as the assembler is not able to find proper overlaps among reads. However, not trimming faulty bases may result in erroneous assemblies.[69] Some authors[56,88,173] try to fix the read first and when this is not possible, they trim it. If the result remains unsatisfactory, they discard the read. Another approach is to pre-process the reads and cleave the suffixes and prefixes having low quality scores to decrease the number of false positives in the k-mer spectrum.[89] One must be careful with the quality scores and take into account that the accuracy of the quality scores depend upon library preparation method as demonstrated for Illumina MiSeq data[48]. When dealing with very long reads like Pacific Biosciences with a high percentage of errors[62], along with considering trimming their ends, an additional approach is to split the long reads into smaller, high quality segments. The authors of[85] do not correct the spurious bases, opting instead to split the read at the locus of the error and to remove the faulty base. They argue that for a further assembly step based on k-mers such as Velvet[171], their method should not pose any problems.

Unknown/Uncalled Bases

The unknown bases (denoted by N in Illumina and Roche 454 sequencing) are one of the four basic types of errors. Schirmer *et al.* concludes that this type of error does not occur randomly as supported by their non-uniform distribution. Column "*N*" from table 2 indicate whether the correctors support Ns or not. In some cases[4,52,55,106], the authors prefer to exclude all reads which contain one or more uncalled bases. Other programs (such as **LSC**) eliminate the reads that have a frequency of Ns above a threshold. A different strategy is to convert each unknown to a real base, like **RACER**, **Reptile** and **Parallel Reptile**. Tools such as[73,98] tackle the unknown bases in the correction process. Some authors fail to mention the support for uncalled bases in their papers, therefore it is up to the user to experiment. Since the unknown bases are a type of error[41], the authors should make it clear whether their software supports them or not. In[98], the authors put together a table with a selection of correcting software and their features including handling of uncalled bases.

Low-Coverage Regions and Uniformity

Illumina sequencing generally has a higher average coverage than other platforms[174], but their short size may not be suited for phylogenetic profiling when a high resolution is required. However, many studies[54,175–177] have found that GC-poor and GC-rich regions have low coverage or even no coverage at all. An error corrector must consider these platform-specific shortcomings to increase sensitivity and specificity.[7]

The authors of **MyHybrid** and **Coral** state that correction methods expects the coverage to be relatively high and use that multiplicity for a meaningful decision. Therefore, they cannot do much for reads from low-coverage regions. The methods that use a threshold for weak/solid k-mers will work if the coverage is high enough or uniform, but they will end up destroying the low-coverage regions (**QuorUM** tries to avoid this problem). **Edar** takes into account the bias introduced by GC regions when calculating the k-mer coverage, by actively considering the GC content of the k-mer. For a reliable result, the authors recommend using a reference genome to accurately calculate the coverage. While many authors do not state the minimum required coverage for a successful correction, **ECHO**'s paper specifies a coverage of 15 or higher. **Hammer** and **BayesHammer** are specifically designed for error correction without uniformity assumptions. Due to uniformity, some authors admit their algorithm's limit like in case of **Reptile** where a non-uniform coverage and the existence of more than one acceptable tile force the algorithm to skip a correction decision. The authors of **Blue** mention the caveat of a simple k-mer cut-off due to uniformity which can result in the rejection of correct k-mers in low-coverage regions and the acceptance of erroneous k-mers in very high-coverage regions. **Fiona** detects erroneous k-mers by calculating the expected coverage for each k-mer, given a uniform sampling of genomic positions. It uses a hierarchical statistical model to describe the expected coverage distribution of k-mers based on library preparation and sequencing.

Parameters

All the methods rely on specific thresholds, lengths, ratios or probabilities to drive their correction. As the methods evolve, they tend to move the burden of choosing the best suited values for their parameter from the user to the program itself.

Generally, the k-mer size has a default (user adjustable) value. However, some correctors like **Quake** require an explicit value from the user, but offer a formula to determine it. **Coral** has a default value, but it also proposes a formula in order to obtain the best results.

The technology flag can explicitly set the source technology. For example, **Coral** has a flag for "Illumina/454" and **Fiona** another one for "Illumina/Ion Torrent/454", which helps the software to decide the best approach for correction. Others have a flag which enable targeting the errors specific to a specific platform, i.e. homopolymers errors in 454 with the "hp" flag in[98]. **Karect** can run with indels support (Ion Torrent, Roche 454) or without (Illumina).

Parameter selection automatic/manual: A manual method to select parameters requires the user to try different values for different parameters to obtain the best results. On the other hand, an automatic method would prevent the user to provide additional valuable information to infer the best actions the algorithm has to take during the correction process. We must distinguish between automatic determination of the best value for the dataset/ analysed case and the default value of a parameter (deemed by the authors to be an acceptable value). The two programs supporting full automatic parameter value selection are **HiTEC** and **Fiona**. Note that **HiTEC** needs the length of the genome and the percentage of errors as input, but these two parameters remain the same for a certain dataset.

Single Threaded vs Parallel

Generally, the programs tested in this review support parallel processing using multiple threads. There are methods, like **BLESS**, that can compete against multi-threaded software due to their approach, despite being single threaded . Other methods like **Reptile** have been updated to run on multiple CPUs[90], using the same initial correction mechanism. The parallel implementations are a normal trend as both the NGS data size and the length of the reads increase.

A distinction must be made between those programs being natively parallel (they internally split the jobs between multiple workers) like **Coral** and **HSHREC** and those that have no parallel implementation but their input can be divided in chunks and multiple processes can be launched on different fragments of the initial dataset like in the case of **proovread**.

Parallel Technology: The reader can check the parallel technology used by a corrector on column "*Par. Tech.*" from table 2. Most of the parallel implementations use OpenMP to distribute the workload among the threads. **DecGPU** and **Parallel Reptile**

support distributed-memory computing by MPI. The distributed-memory model requires a more complex programming and configuration, but enables gathering a larger number of RAM memory. This advantage may enable tackling larger-scale correction problems which cannot be addressed on a single node. Other methods, such as **Quake**, use parallelism only for counting k-mers. Furthermore, **proovread** and **Nanocorr** can run on multi-core desktops and distributed clusters using queuing engines like SGE (commercial[2]) and SLURM[178]. As of now, there are two fully distributed methods using the Map-Reduce (Hadoop) paradigm.[110,111] An interesting addition to the field is **FADE**[102], the FPGA error corrector which unleashes the massive parallelism available on FPGA devices to tackle the error correction.

Operating System and Programming Language

The resource consumption is a problem because of the continuous growth of the size of the NGS data. Owing to this, we observe that the majority of authors chose a low-level language like C or C++ to implement their solution. An interesting trend is the use of of C++ over C in writing the software, with just one program (from those being available online for us to analyse), **Coral**, being implemented in pure C. The C# implementation of **Blue** obtains the best performance when compared against other algorithms on Illumina and Roche 454. Even though C# is not considered to be a high performance language, **Blue** performs really well against the rest of the algorithms. Some correctors appear to be implemented in Perl and/or Python. These are often scripts that are used to execute third party software. In case of **LCS**, the authors offer a software wrapper written in Perl that uses an external aligner to map the short reads against the long ones. **Nanocorr** is a python wrapper for BLAST and pbdagcon [3]. For the reader's convenience, we list in table 2 the programming language of choice for each corrector.

Overall, the correctors should work on the three most important desktop/server operating systems: Windows, Linux and Mac OS. Some authors[94] mention the supported platform in their papers. Generally, the authors prefer to support Unix flavours and to distribute the source code and the instructions to build it. Furthermore, the tests for the majority of the works were done under Linux, with some authors also using Windows.[52] Besides PC based methods, we included a corrector which runs on FPGAs - **FADE** - (even though with the help of a computer that handles the data transfer and storage).

Recommendations

Depending upon the preprocessing type, a corrector that can separate the steps of the correction can save a lot of time when processing big datasets. For instance, **Blue** can generate the k-mer table in a separate process. In its case, the advantage is twofold. Firstly, it can use the same k-mer table for multiple runs with the different combinations

[2] Available at `http://www.univa.com/products/grid-engine.php`
[3] Available at `https://github.com/PacificBiosciences/pbdagcon`

of values for the majority of parameters that are not involved in the histogram's generation. Secondly, it makes cross-correction possible since the k-mer table can be generated from one technology that can be used to correct data from a different one. Another important aspect is the k-mer size selection. Any program able to determine the size of k-mers automatically (**HiTEC**) or use variable sizes (**Fiona**) is recommended over those with user-defined only k-mer size selection. Next recommendation is to select a software like **Fiona** or **Blue** that consider repetitive regions as they may avoid altering similar zones with SNPs. The unknown bases support depends upon the used sequencing technology as some technologies do not produce this type of error. Furthermore, in case of extreme necessity, one can easily write a script that can convert the unknowns into random or specific nucleotides. The low-coverage issue must always be a top priority since some algorithms can skip those zones because of the limited information. Finally, the user must be careful with the trimming and the splitting of the reads. In the former case, a corrector (like **Quake** and **DecGPU**) may trim a read if the correction is not possible. If result is then fed to an assembler, the correction may negatively influence the overlap detection. **Edar** applies a distinct correction mechanism by cleaving the reads. This approach may be detrimental for any further step because a lot of information is lost. The reads become much smaller and the relation between segments part of the same read is lost forever.

Testing

This section focuses on the testing part included on the analysed papers. We extracted all datasets which we could identify in the papers along with the results provided by the authors. Due to space limitation, we split the datasets in two categories. Table 5 lists all those datasets appearing in[77,78]. The rest of the datasets (second category) along with the benchmark information are located in the Supplementary Material. The gain metric for both categories is calculated (by the authors or by the reviewers) using the formula from[77]. The reader must be careful though, because there is no standard way to count TP, FP and FN. As a result, the numbers given by the authors and the reviewers must be taken with a grain of salt. The reliable gain appears on the same column (same review, the authors used the same testing approach for all software). The hardware configurations used by the correctors' authors and in[78] as more sections related to testing are located in the Supplementary Material.

Methods

Some metrics are general and do not focus on certain type of error correction mechanism. For instance, the sensitivity and specificity appear in k-mer-spectrum methods like **Reptile**, suffix trie/array methods like **SHREC** and MSA methods like **Coral**. Sensitivity, specificity, gain and genome assembly statistics are the most widespread metrics.

Simply counting the mapping reads that did not map before the correction and do so after it, can prove the effectiveness of a corrector.[53,64,106] However, the differences

obtained heavily depend on the aligner's parameters. For example, the authors of[77] test with different values for the aligner, albeit only for datasets with indels, given the complexity introduced by these types of errors.

In the case of artificial datasets, it is possible to report quite reasonable the error rate before and after correction.[85,109] On the other hand, the exact error rate for real data can only be estimated.[44]

Table 5: Testing result from benchmarks and original papers; **Mem/Rt/Gain** - memory in GB/runtime in mins (num of threads in parentheses for **Rt O**)/gain as percentage reported in the original paper (represented by **O**) and in the survey by[77] (represented by **S**), **Gain R** - gain(percentage) reported by[78], **# Rds** - number of reads, **# Bs** - number of bases; The programs that didn't run successfully are market by "-*"

Dataset	Genome	Tech	# Rds	# Bs	Software	Mem O	Mem S	Rt O (Num Th)	Rt S	Gain O	Gain S	Gain R
SRX000429	E. Coli 36	Illum	10.4	749.4	SHREC	-	-	-	-	-	-	0.75
					HSHREC	-	14.3	-	153.23	-	-2.886	0.989
					HiTEC	-	13	-	143.13	0.861	0.846	0.952
					DecGPU	-	-	-	-	-	-	0.772
					Coral	-	7.5	-	36.19	-	0.579	0.857
					ECHO	-	-*	-	-*	-	-*	0.715
					Hammer	-	-	-	-	-	-	0.944
					Quake	-	4.1	-	38.88	-	0.784	-*
					REDEEM(i)	9	-	120(1)	-	-	-	0.926
					RACER(d)	1.1	-	15.7(1)	-	0.908	-	-
					Trowel	3.5	-	2.9(32)	-	0.765	-	-
					CloudRS	-	-	-	-	0.207	-	-
					Reptile(e)	1.1	3.7	47.4d 149.4d2(1)	23.32	0.757d1 0.802d2	0.933	-0.047
					Par. Reptile(e)	-	-	0.18d1 1.38d2(512)	-	0.757d1 0.802d2	-	-
SRR001665_1(j)	E. Coli K12 MG1655	Illum.	10.4	749.4	SHREC	-	-	-	-	-	-	0.822

Table 5 Continued from previous page

Dataset	Genome	Tech	# Rds	# Bs	Software	Mem O	Mem S	Rt O (Num Th)	Rt S	Gain O	Gain S	Gain R
					HSHREC	-	-	-	-	-	-	0.989
					HiTEC	-	-	-	-	0.857	-	0.795
					DecGPU(b)	-	-	12.31(4)	-	-	-	0.999
					Coral	-	-	-	-	-	-	0.999
					ECHO	-	-	-	-	-	-	0.999
					Hammer	-	-	-	-	-	-	0.668
					REDEEM	-	-	-	-	-	-	0.231
					BLESS	0.01	-	23(1)	-	0.967	-	-
					Qamar(f)	3.5	-	38.85(-)	-	0.913	-	-
					Reptile(e)	0.84	-	21d1 73.8d2(1)	-	0.652d1 0.709d2	-	0.998
					FADE	-	-	0.51(FPGA)	-	0.923	-	-
					SHREC	-	-	-	-	-	-	0.89
					HiTEC	-	-	-	-	-	-	0.324
					DecGPU	-	-	-	-	-	-	0.548
					Coral	-	-	-	-	-	-	0.816
					ECHO	-	-	-	-	-	-	0.711
SRR006332	Acinetob. ADP1	Illum.	18.13	652.7	Hammer	-	-	-	-	-	-	0.984
					Reptile(e)	2.2	-	99.6d1(1)	-	0.632d1	-	0.982
					Quake	-	-	-	-	-	-	0.926
					CUDA-EC	-	-	-	-	-	-	0.953
					REDEEM	-	-	-	-	-	-	0.595
					CloudRS	-	-	-	-	0.265	-	-

Table 5 Continued from previous page

Dataset	Genome	Tech	# Rds	# Bs	Software	Mem O	Mem S	Rt O (Num Th)	Rt S	Gain O	Gain S	Gain R
NC_005966	Acinetob. ADP1	Illum.	4	144	SHREC	-	-	-	-	-	-	0.997
					HSHREC	-	-	-	-	-	-	0.995
					HiTEC	-	-	-	-	-	-	0.994
					DecGPU	-	-	-	-	-	-	0.995
					Coral	-	-	-	-	-	-	0.997
					ECHO	-	-	-	-	-	-	0.993
					Hammer	-	-	-	-	-	-	0.997
					Reptile(e)	0.66	-	15.6d1(1)	-	0.599d1	-	0.995
					Quake	-	-	-	-	-	-	0.993
					CUDA-EC	-	-	-	-	-	-	0.995
ERA000206	E. Coli K12 MG1655	Illum.	14.21	2800	HSHREC	-	29.9	-	779.15	-	-1.606	-
					Reptile	-	19.1	-	225.43	-	0.91	-
					HiTEC	-	9.8	-	683.87	0.874	0.949	-
					Coral	-	30	-	450.29	-	0.112	-
					Blue(a)	1.6+0.4	-	9.2+14.4(8)	-	-	-	-
SRR022918	E. Coli 47	Illum.	7.2	677.2	HSHREC	-	12.6	-	74.13	-	-0.316B -0.269R	-
					Reptile(e)	1.9	3.4	56.4d1(1)	71.64	0.381d1	0.852B 0.791R	-
					Quake	-	2	-	80.43	-	0.313B 0.285R	-
					HiTEC	-	6.2	-	59.19	-	0.929B 0.912R	-

Table 5 Continued from previous page

Dataset	Genome	Tech	# Rds	# Bs	Software	Mem O	Mem S	Rt O (Num Th)	Rt S	Gain O	Gain S	Gain R
					ECHO	-	16	-	304.21	-	0.907B 0.897R	-
					Coral	2.6	7.7	8.12(4)	8.32	0.974	0.002B 0.002R	-
					RACER(d)	1.5	-	13.65(1)	-	0.826	-	-
					Trowel	4.5	-	3.7(32)	-	0.447	-	-
					Fiona	1	-	2.5(8)	-	0.768	-	-
					Hector(c)	-	-	3(2)	-	0.882	-	-
SRR000868	E. Coli UTI89	454	0.4	118.2	HSHREC(h)	-	9.9	-	16.72	-	0.293E1 0.224E2 0.470E3 0.473E4 0.451E5	-
					Coral(h)	1.8	2.4	5.3(4)	2.57	0.857	0.497E1 0.425E2 0.698E3 0.702E4 0.734E5	-
SRX100885	S. Cerevisae	Illum.	26.03	4000	HSHREC	-	30.1	-	1027.48	-	-2.497	-
					Reptile	-	4.12	-	165.88	-	0.224	-
					Coral	-	34.5	-	544.31	-	0.067	-
					RACER(d)	3.2	-	198.31(1)	-	0.146	-	-
					CloudRS	-	-	-	-	0.779	-	-
					Chung *et al.*	-	-	153.48(80)	-	-	-	-

Table 5 Continued from previous page

Dataset	Genome	Tech	# Rds	# Bs	Software	Mem O	Mem S	Rt O (Num Th)	Rt S	Gain O	Gain S	Gain R
SRR022866	S. Aureus USA300	Illum.	12.77	1900	HSHREC	-	30.2	-	813.65	-	-4.24	-
					Reptile	-	13.1	-	295.94	-	0.613	-
					Coral	4	40	80(4)	210.17	0.974	0.025	-
					RACER(d)	2.1	-	81.68(1)	-	0.47	-	-
					Trowel	5.7	-	6.4(32)	-	0.447	-	-
					CloudRS	-	-	-	-	0.335	-	-
					Chung *et al.*	-	-	63.45(80)	-	-	-	-
SRX023452 SRX006152 SRX006151	D. Melanog.	Illum.	18.96 9.46 20.8	3600 1400 1900	HSHREC	-	30.3	-	2562.88	-	-5.85	-
					Reptile	-	24	-	532.76	-	0.667	-
					Quake	-	12.1	-	222.35	-	-0.126	-
					Coral	-	30	-	373.19	-	0.449	-
					RACER(d)	39.4	-	501.95(1)	-	0.57	-	-
					Par. Reptile(e)	-	-	2.82d1 412.98d2(512)	-	-	-	-
ERR039477	E. Coli DH10B	Ion	0.39	36	Fiona	1	-	3.1(8)	-	0.905	-	-
					HSHREC(g)	-	10.21	-	8.72	-	0.399T1 0.450T2	-
					Coral(g)	-	2.24	-	1.13	-	0.515T1 0.604T2	-

Table 5 Continued from previous page

Dataset	Genome	Tech	# Rds	# Bs	Software	Mem O	Mem S	Rt O (Num Th)	Rt S	Gain O	Gain S	Gain R

(a) Resources for preparing the dataset and correcting the data;
(b) The result for 1xGPU execution, for more threads and GPUs please refer to [88];
(c) Approximate values for runtime extracted from the chart in the paper for the original results;
(d) Results for the datasets filtered using BWA;
(e) Runtime for Hamming dist (1 and 2) between two k-mers;
(f) Gain calculated by us using the values for TP, FP, FN given by the authors and the formula in [77];
(g) Profiles: T1 - With all mapped reads; T2 - excluding reads with more than 10 errors;
(h) Different profiles for the aligner and a k-mer length of 10; The parameters set for E(1-5) can be found in [77];
(i) Dataset not explicitly pinpointed, we inferred the value;
(j) SRR001665_1 is the same as SRX000429, but only the forward direction is used;

Gain/Specificity/Sensitivity

The gain (G), specificity (SP) and sensitivity (SE) metrics (for formulae see Equation 1) seem to become the de-facto on error correction. SP and SE first appeared in[55]. The gain[84], represents the percentage of eliminated errors. They are all based on counting:

- TP (true positives) existing errors that are corrected.
- TN (true negatives) correct bases left unmodified.
- FP (false positives) correct bases that are wrongly considered being faulty.
- FN (false negatives) erroneous bases left unmodified.

$$G = \frac{TP - FP}{TP + FN}, SE = \frac{TP}{TP + FN}, SP = \frac{TN}{TN + FP}. \quad (1)$$

There are differences in how the authors of each tool compute TP/TN/FP/FN. For **Reptile**, they compute the errors at base level, while for **SHREC** and **RACER**, they count the errors at reads level (a read is either error-free or erroneous, without considering the number bad base). The lack of a standard approach on counting the errors leads to some serious inconsistencies in the results published in the literature by the same tool in different benchmarks even using the same dataset and formula. For example, in[77], **Coral** obtains a score of 0.002 for an Illumina dataset (SRR022918), while with the same dataset, it scores 0.97 in its own paper. There is no doubt that **Coral** is a good corrector (as demonstrated by[41]) and even in the aforementioned survey it performs really good on datasets with indels. The problem lies in the different approach in performing the tests, which is not infallible. Moreover, the previous difference in score may arise just by changing the way to prepare the datasets before correction. Even though the approach is the same (filter non and multi mapping reads), the aligner can make a difference too. Salmela *et al.*[73] use Soap, while Yang *et al.*[77] use BWA.[179,180] Table 5 contains the results obtained by some correctors in[77,78] on a number of datasets. The different results obtained in the original article versus the surveys can be explained by the difference in dataset preparation, FP/TP/FN/TN counting and maybe different versions of the tested programs. The Supplementary Material contains the results

Assembly

The majority of the recent publications include some information about the assembly performance. Many list the N50 metric and the contigs count, but there are variations. Salzberg *et al.*[69] define N50 value as "the size of the smallest contig (or scaffold) such that 50% of the genome is contained in contigs of size N50 or larger". A contig is a multiple sequence alignment of reads represented as a consensus while the scaffold is a list of contigs that defines their order, orientation and the length of the gaps between them.[181] **Pluribus**' paper provides the number of nodes in the Bruijn graph generated

by Velvet, which give a measure of the fragmentation of the assembly. For **QuorUM**, the E-size statistics[69] complements the N50 value.

While N50 and the maximum contig length measure the quality of error correction and give valuable feedback over the correction, the authors of[98] state that these metrics are not always accurate. This mainly happens because the assemblers can generate chimeric contigs in overcorrected datasets. In any case, the correctness of an assembly is hard to verify.[182] As a consequence, more refined assembly evaluation approaches are considered (e.g. the Mauve Assembly Metrics[183] for **Blue**). **BLESS** uses several assemblers from two different categories (de Bruijn and string-graph based). This approach increases the reliability of the capability to produce valid results that do not fit a certain type of assembler (or worse, a certain assembler) and to prove the corrector's generality. Furthermore, the same authors do not just provide the value of N50, they also assess the quality of the assembly, using GAGE[69], as the authors of **Musket**.

For **BLESS**, the authors chose only artificial datasets to demonstrate the capabilities of their implementation. **DecGPU** contains the assembly information only for their corrector, not the other correctors in their benchmark. Salmela[73] eliminates the trimmed reads generated by **Quake**, because the Illumina-only assembler Edena[184] can only handle reads with the same length.

Our reader can find the assembly results (where available in the original paper) in the table with the performance assessment from the Supplementary Material.

Genomes Used for Testing

A recent review[81] tests seven correctors on three large genomes (H. Sapiens, D. Melanogaster and C. Elegans) among others. The datasets for the aforementioned species are very large, between 31 million and 1.7 billion reads. As discussed in[109], the more complex, diploid genomes bring up the problem of heterozygosity and how to discriminate between true variants and sequencing errors. The repetitive regions also pose a problem to the corrector as mentioned by the authors of **Musket** and **HiTEC**. We can see a clear focus on the human genome, as the largest and most complex datasets for benchmarking come from this organism (for **BLESS**, **Blue** and **Fiona**).

Real vs. Artificial Datasets

We discern three situations for the datasets used in benchmarks: correctors tested only on simulated datasets, only on real data or both types of datasets. The main reasons stated to avoid artificial datasets are the lack of simulators capable of producing meaningful data and the non-existence of some real challenges that only appear in already existing real data[73,84]. To generate the artificial data, the authors of **BLESS** used simLibrary and simNGS[185], for **Lighter** and **Hector** - Mason[186] and for **Pluribus** - ART[187]. Pbsim[188] is cited in[63], but the authors did not use it, as they preferred to perform their tests on real data only. Correction software from 2013 onwards are tested mainly on

data generated with dedicated software, opposed to previous use of in-house mechanisms. Some authors test their work on existing artificial datasets, e.g.[172] for **Edar** and[189] for **Qamar**.

Resource Consumption

Despite that early methods were not explicitly targeting a reduction in the requirements on CPU and memory consumption, currently all the methods try to address this aspect. Some of the first stand alone error correction methods were developed in Java, whilst the latest prefer C/C++. On the other side, **Blue** (2014) runs on the Microsoft .Net® platform while offering a very good performance. Many authors do not specify the exact method of determining the resource consumption. The authors of[62,73] use the Unix time command.

There are multiple approaches to measure memory consumption. This is especially troublesome when testing programs in C/C++ against the ones in Java. For the latter, the simplest way is to measure the memory used by a process, but one must be careful with the memory allocation of the VM. In case of very small datasets, the overhead added by the VM can give a skewed view of the real behaviour. Even the memory usage of native programs is hard to assess under Linux given the fact that there are two main memory types: virtual memory and Resident Set Size. Most of the papers do not state how memory is measured, making comparisons difficult. We also observed a lack of information regarding the number of threads used to asses the performance in some cases. For example,[55] state the use of multi-cores, but do to mention how many threads were actually used in their test. The scalability of the program, i.e. how the software behaves with an increasing number of threads is often not evaluated. Some authors like[98] present additional test cases in which they only asses their program, usually for very large datasets. Table 5 contains the results for a number of correctors. For reader's convenience, we also included the results obtained in the original articles. One can see some differences in the memory and time obtained on the same dataset. This is not necessarily a sign of overinflated results in the original work, but more of a difference in the testing system and how the authors prepared the data for correction. Furthermore, there is no clear indication of the version of the program used in the original benchmark and survey benchmark respectively. The aforementioned table is a guideline for the interested user that can help him/her choose the right toll given the target hardware.

Recommendations

Using the three benchmark reviews cited earlier, we can see some correctors that emerge as the winners. Molnar and Ilie[81] consider **BLESS**, **Musket**, **RACER** and **SGA** the best choices for HiSeq data. The last three were also able to handle H. Sapiens datasets with over 1.3 billion reads of 100-103bp on a Dell computer with 32 cores and 1TB RAM. For MiSeq, **RACER** wins in three from a total of four tests. Tahir *et al.*[78] recommend **HiTEC**, **ECHO** and **DecGPU**. In their opinion, the first two have

a plus with their automatic parameter selection that can optimize performance. Finally, Yang *et al.*[77] obtain good results with **Reptile** and **HiTEC** for Illumina data. Unfortunately, **HiTEC** fails to run for three of seven datasets. This review also includes datasets from Roche 454 and Ion Torrent, where **Coral** wins in all cases against **HSHREC**. The two aforementioned correctors supporting indels do not obtain a high gain for Illumina datasets, making them inferior to Illumina-only correctors.

An important advice for the reader is to consider more than one metric when selecting a program. We recommend that (s)he should consider not only gain, sensitivity and specificity, but also other metrics like genome assembly and short read alignment. Another important aspect is the type of datasets used in testing. A corrector able to handle heterozygous organisms such as H. Sapiens (like **Blue**, **Fiona** and **BFC**) should perform pretty well with other complex organisms. From a resource consumption perspective, **BLESS** uses the least memory, but it is single threaded and quite slow since it uses the disk. **Blue** on the other hand obtains some very good results in its own publication, offering a trade-off between memory and CPU consumption.

Conclusion

As the sequencing market share suggests, Illumina has become an important player in the industry. Our review strongly supports this claim based upon the general support for this technology, with almost all programs supporting either only Illumina or Illumina plus additional sequencing technologies. Pacific Biosciences and Oxford Nanopore technologies with their (very) long reads gave birth to a new trend. This trend requires the evolution of error correction techniques to support longer reads and to deal with the high error rate these technologies currently have. Overall we can see improvements in the area of error correction for different technologies as the newest methods are both resource efficient and offer a very good correction. A reliable and structured way to measure the accuracy is also very important.

There is room for further improvement especially on the biological aspects of the correction. Here, we refer to concepts from biology like ploidy, heterozygosity and repetitive regions not the more computer science oriented concepts like the memory consumption, genome representation on two bits per base and multi-core support. Now that the error correction field has been sufficiently explored, the newer methods improve over the existing ones. A not so favourable trend is the non-existence of some mature-enough methods that are constantly enriched with new features, as in other related fields like assembly (Mira[190]) or short sequence alignment (BWA[180]).

Acknowledgement

We want to thank our colleague Eloy Romero Alcale who has provided valuable advice regarding the structure of the document.

Funding

This work was supported by Generalitat Valenciana [GRISOLIA/2013/013 to A.A.].

References

1. Zimin A, Stevens KA, Crepeau MW, Holtz-Morris A, Koriabine M, Marçais G, et al. Sequencing and assembly of the 22-Gb loblolly pine genome. Genetics. 2014;196(3):875–890.

2. Nystedt B, Street NR, Wetterbom A, Zuccolo A, Lin YC, Scofield DG, et al. The Norway spruce genome sequence and conifer genome evolution. Nature. 2013;497(7451):579–584.

3. Friz CT. The biochemical composition of the free-living Amoebae¡ i¿ Chaos chaos, amoeba dubia¡/i¿ and¡ i¿ Amoeba proteus¡/i¿. Comparative biochemistry and physiology. 1968;26(1):81–90.

4. Ilie L, Fazayeli F, Ilie S. HiTEC: accurate error correction in high-throughput sequencing data. Bioinformatics. 2011;27(3):295–302.

5. Ross MG, Russ C, Costello M, Hollinger A, Lennon NJ, Hegarty R, et al. Characterizing and measuring bias in sequence data. Genome Biol. 2013;14(5):R51.

6. Loman NJ, Misra RV, Dallman TJ, Constantinidou C, Gharbia SE, Wain J, et al. Performance comparison of benchtop high-throughput sequencing platforms. Nature biotechnology. 2012;30(5):434–439.

7. Kelley DR, Schatz MC, Salzberg SL. Quake: quality-aware detection and correction of sequencing errors. Genome Biol. 2010;11(11):R116.

8. marketsandmarkets com. Next Generation Sequencing (NGS) Market by Platforms (Illumina HiSeq, MiSeq, HiSeqX Ten, NextSeq 500,Thermo Fisher Ion Proton/PGM), Bioinformatics (Exome Sequencing, RNA-Seq, ChIP-Seq), Technology (SBS, SMRT) & by Application (Diagnostics, Personalized Medicine) Global Forecast to 2020; 2014. Accessed: 2014-09-09. .

9. Illumina. Human whole-genome sequencing power; 2015. Available from: `http://www.illumina.com/systems/hiseq-x-sequencing-system.html`.

10. Check Hayden E. Is the $1,000 genome for real?; 2015. Available from: `http://www.nature.com/news/is-the-1-000-genome-for-real-1.14530`.

11. Institute NHGR. International Human Genome Sequencing Consortium Announces ”Working Draft” of Human Genome; 2000. Available from: `https://www.genome.gov/10001457`.

12. Sanger F, Nicklen S, Coulson AR. DNA sequencing with chain-terminating inhibitors. Proceedings of the National Academy of Sciences. 1977;74(12):5463–5467.

13. Fuller CW, Middendorf LR, Benner SA, Church GM, Harris T, Huang X, et al. The challenges of sequencing by synthesis. Nature biotechnology. 2009;27(11):1013–1023.

14. Cock PJ, Fields CJ, Goto N, Heuer ML, Rice PM. The Sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants. Nucleic acids research. 2010;38(6):1767–1771.

15. Illumina. Specification Sheet: Illumina HiSeq X Series; 2015. Available from: http://www.illumina.com/documents/products/datasheets/datasheet-hiseq-x-ten.pdf.

16. Illumina. Specification Sheet: HiSeq 3000/HiSeq 4000 Sequencing Systems; 2015. Available from: https://www.illumina.com/content/dam/illumina-marketing/documents/products/datasheets/hiseq-3000-4000-specification-sheet-770-2014-057.pdf.

17. Illumina. Specification Sheet: NextSeq Series Specifications; 2015. Available from: http://www.illumina.com/content/dam/illumina-marketing/documents/products/datasheets/datasheet-nextseq-500.pdf.

18. Illumina. HiSeq 2500 Specifications; 2015. Available from: http://www.illumina.com/systems/hiseq_2500_1500/performance_specifications.html.

19. Illumina. Specification Sheet: HiScan SQ System; 2012. Available from: http://www.illumina.com/documents/products/datasheets/datasheet_hiscansq.pdf.

20. Illumina. Specification Sheet: Genome AnalyzerIIx System; 2011. Available from: https://support.illumina.com/content/dam/illumina-marketing/documents/products/datasheets/datasheet_genome_analyzeriix.pdf.

21. Illumina. Specification Sheet: Genome AnalyzerIIe System; 2010. Available from: http://www.illumina.com/documents/products/datasheets/datasheet_genome_analyzer_IIe.pdf.

22. Illumina. Specification Sheet: Genome AnalyzerII System; 2009. Available from: http://tucf.org/htseq_GenomeAnalyzer_SpecSheet.pdf.

23. Illumina. Specification Sheet: Genome Analyzer System; 2007. Available from: http://www.geneworks.com.au/library/GenomeAnalyzer_SpecSheet.pdf.

24. Illumina. Specification Sheet: MiSeq System; 2015. Available from: http://www.illumina.com/content/dam/illumina-marketing/documents/products/datasheets/datasheet_miseq.pdf.

25. Scientific TF. Ion Proton System Specifications; 2015. Available from: http://www.thermofisher.com/es/en/home/life-science/sequencing/next-generation-sequencing/ion-torrent-next-generation-sequencing-workflow/ion-torrent-next-generation-sequencing-run-sequence/ion-proton-system-for-next-generation-sequencing/ion-proton-system-specifications.html.

26. Scientific TF. Ion PGM System Specifications; 2015. Available from: http://www.thermofisher.com/es/en/home/life-science/sequencing/next-generation-sequencing/ion-torrent-next-generation-sequencing-workflow/ion-torrent-next-generation-sequencing-run-sequence/ion-pgm-system-for-next-generation-sequencing/ion-pgm-system-specifications.html.

27. Scientific TF. 5500 W & 5500xl W Series; 2015. Available from: https://www.thermofisher.com/es/en/home/life-science/sequencing/next-generation-sequencing/solid-next-generation-sequencing/solid-next-generation-sequencing-systems-r
html.

28. Scientific TF. 5500 & 5500xl Series; 2013. Available from: http://www.appliedbiosystems.com/absite/us/en/home/applications-technologies/solid-next-generation-sequencing/next-generation-systems.html.

29. Scientific TF. SOLiD System accuracy with the Exact Call Chemistry module; 2011. Available from: https://tools.thermofisher.com/content/sfs/brochures/cms_091372.pdf.

30. Biosciences P. PacBio RS II; 2015. Available from: http://files.pacb.com/pdf/PacBio_RS_II_Brochure.pdf.

31. Biosciences P. PacBio RS; 2015. Available from: http://files.pacb.com/pdf/PacBio_RS_Brochure.pdf.

32. 454 R. GS FLX+ System; 2015. Available from: http://454.com/products/gs-flx-system/index.asp.

33. 454 R. GS Junior System; 2015. Available from: http://454.com/products/gs-junior-system/index.asp.

34. 454 R. GS Junior+ System; 2015. Available from: http://454.com/products/gs-junior-plus-system/index.asp.

35. Technologies ON. Specification MinION & PromethION; 2015. Available from: https://www.nanoporetech.com/community/specifications.

36. Biosystems A. Overview of SOLiD Sequencing Chemistry; 2013. Available from: http://www.appliedbiosystems.com/absite/us/en/home/applications-technologies/solid-next-generation-sequencing/next-generation-systems/solid-sequencing-chemistry.html.

37. Biosciences P. How does SMRT work?; 2014. Available from: http://www.pacificbiosciences.com/products/smrt-technology/.

38. Zhu P, Craighead HG. Zero-mode waveguides for single-molecule analysis. Annual review of biophysics. 2012;41:269–293.

39. Technologies ON. DNA: Nanopore sequencing; 2015. Available from: https://nanoporetech.com/applications/dna-nanopore-sequencing.

40. Lieberman KR, Cherf GM, Doody MJ, Olasagasti F, Kolodji Y, Akeson M. Processive replication of single DNA molecules in a nanopore catalyzed by phi29 DNA polymerase. Journal of the American Chemical Society. 2010;132(50):17961–17972.

41. Schulz MH, Weese D, Holtgrewe M, Dimitrova V, Niu S, Reinert K, et al. Fiona: a parallel and automatic strategy for read error correction. Bioinformatics. 2014;30(17):i356–i363.

42. Mikheyev AS, Tin MM. A first look at the Oxford Nanopore MinION sequencer. Molecular ecology resources. 2014;14(6):1097–1102.

43. Wirawan A, Harris RS, Liu Y, Schmidt B, Schröder J. HECTOR: a parallel multistage homopolymer spectrum based error corrector for 454 sequencing data. BMC bioinformatics. 2014;15(1):131.

44. Wang XV, Blades N, Ding J, Sultana R, Parmigiani G. Estimation of sequencing error rates in short reads. BMC bioinformatics. 2012;13(1):185.

45. Sahli M, Shibuya T. Qamar–A More Accurate DNA Sequencing Error Correcting Algorithm. International Proceedings of Chemical, Biological & Environmental Engineering. 2012;31.

46. Bragg LM, Stone G, Butler MK, Hugenholtz P, Tyson GW. Shining a light on dark sequencing: characterising errors in Ion Torrent PGM data. PLoS computational biology. 2013;9(4):e1003031.

47. Nakamura K, Oshima T, Morimoto T, Ikeda S, Yoshikawa H, Shiwa Y, et al. Sequence-specific error profile of Illumina sequencers. Nucleic acids research. 2011;p. gkr344.

48. Schirmer M, Ijaz UZ, D'Amore R, Hall N, Sloan WT, Quince C. Insight into biases and sequencing errors for amplicon sequencing with the Illumina MiSeq platform. Nucleic acids research. 2015;p. gku1341.

49. Illumina. TruSeq DNA PCR-Free Sample Preparation Kit; 2013. Available from: http://www.appliedbiosystems.com/absite/us/en/home/applications-technologies/solid-next-generation-sequencing/next-generation-systems/solid-sequencing-chemistry.html.

50. Biosciences S. Accel-NGS DNA Library Kit for Ion Torrent; 2015. Available from: http://www.swiftbiosci.com/products/accel-ngs-dna-library-kit.

51. Gen G. Fast, PCR free DNA library construction for 454; 2015. Available from: http://resource.jerei.com/10896/13041512044595_0.pdf.

52. Sleep JA, Schreiber AW, Baumann U. Sequencing error correction without a reference genome. BMC bioinformatics. 2013;14(1):367.

53. Wijaya E, Frith MC, Suzuki Y, Horton P. Recount: expectation maximization based error correction tool for next generation sequencing data. In: Genome Inform. vol. 23. World Scientific; 2009. p. 189–201.

54. Dohm JC, Lottaz C, Borodina T, Himmelbauer H. Substantial biases in ultra-short read data sets from high-throughput DNA sequencing. Nucleic acids research. 2008;36(16):e105–e105.

55. Schröder J, Schröder H, Puglisi SJ, Sinha R, Schmidt B. SHREC: a short-read error correction method. Bioinformatics. 2009;25(17):2157–2163.

56. Shi H, Schmidt B, Liu W, Müller-Wittig W. A parallel algorithm for error correction in high-throughput short-read data on CUDA-enabled graphics hardware. Journal of Computational Biology. 2010;17(4):603–615.

57. Shao W, Boltz VF, Spindler JE, Kearney MF, Maldarelli F, Mellors JW, et al. Analysis of 454 sequencing error rate, error sources, and artifact recombination for detection of Low-frequency drug resistance mutations in HIV-1 DNA. Retrovirology. 2013;10(1):18.

58. Luo C, Tsementzi D, Kyrpides N, Read T, Konstantinidis KT. Direct comparisons of Illumina vs. Roche 454 sequencing technologies on the same microbial community DNA sample. PloS one. 2012;7(2):e30087.

59. Gilles A, Meglécz E, Pech N, Ferreira S, Malausa T, Martin JF. Accuracy and quality assessment of 454 GS-FLX Titanium pyrosequencing. BMC genomics. 2011;12(1):245.

60. Quail MA, Smith M, Coupland P, Otto TD, Harris SR, Connor TR, et al. A tale of three next generation sequencing platforms: comparison of Ion Torrent, Pacific Biosciences and Illumina MiSeq sequencers. BMC genomics. 2012;13(1):341.

61. Koren S, Schatz MC, Walenz BP, Martin J, Howard JT, Ganapathy G, et al. Hybrid error correction and de novo assembly of single-molecule sequencing reads. Nature biotechnology. 2012;30(7):693–700.

62. Salmela L, Rivals E. LoRDEC: accurate and efficient long read error correction. Bioinformatics. 2014;p. btu538.

63. Hackl T, Hedrich R, Schultz J, Förster F. proovread: large-scale high accuracy PacBio correction through iterative short read consensus. Bioinformatics. 2014;p. btu392.

64. Au KF, Underwood JG, Lee L, Wong WH. Improving PacBio long read accuracy by short read alignment. PLoS One. 2012;7(10):e46679.

65. Technologies ON. The MinION device: a miniaturised sensing; 2014. Available from: https://nanoporetech.com/technology/the-minion-device-a-miniaturised-sensing-system/the-minion-device-a-miniaturised-sensing-system.

66. Loman NJ, Quinlan AR. Poretools: a toolkit for analyzing nanopore sequence data. Bioinformatics. 2014;30(23):3399–3401.

67. Watson M, Thomson M, Risse J, Talbot R, Santoyo-Lopez J, Gharbi K, et al. poRe: an R package for the visualization and analysis of nanopore sequencing data. Bioinformatics. 2014;p. btu590.

68. Goodwin S, Gurtowski J, Ethe-Sayers S, Deshpande P, Schatz M, McCombie WR. Oxford Nanopore Sequencing and de novo Assembly of a Eukaryotic Genome. bioRxiv. 2015;p. 013490.

69. Salzberg SL, Phillippy AM, Zimin A, Puiu D, Magoc T, Koren S, et al. GAGE: A critical evaluation of genome assemblies and assembly algorithms. Genome research. 2012;22(3):557–567.

70. Zhao Z, Yin J, Li Y, Xiong W, Zhan Y. An efficient hybrid approach to correcting errors in short reads. Modeling Decision for Artificial Intelligence. 2011;p. 198–210.

71. Liu Y, Schröder J, Schmidt B. Musket: a multistage k-mer spectrum-based error corrector for Illumina sequence data. Bioinformatics. 2013;29(3):308–315.

72. Liu Y, Schmidt B, Maskell DL. CUSHAW: a CUDA compatible short read aligner to large genomes based on the Burrows–Wheeler transform. Bioinformatics. 2012;28(14):1830–1837.

73. Salmela L, Schröder J. Correcting errors in short reads by multiple alignments. Bioinformatics. 2011;27(11):1455–1461.

74. Qu W, Hashimoto Si, Morishita S. Efficient frequency-based de novo short-read clustering for error trimming in next-generation sequencing. Genome research. 2009;19(7):1309–1315.

75. Joppich M, Schmidl D, Bolger AM, Kuhlen T, Usadel B. PAGANtec: OpenMP Parallel Error Correction for Next-Generation Sequencing Data. In: OpenMP: Heterogenous Execution and Data Movements. Springer; 2015. p. 3–17.

76. Genomeweb. Roche Shutting Down 454 Sequencing Business; 2013. Available from: http://www.genomeweb.com/sequencing/roche-shutting-down-454-sequencing-business.

77. Yang X, Chockalingam SP, Aluru S. A survey of error-correction methods for next-generation sequencing. Briefings in bioinformatics. 2013;14(1):56–66.

78. Tahir M, Sardaraz M, Ikram AA, Bajwa H. Review of Genome Sequence Short Read Error Correction Algorithms. American Journal of Bioinformatics Research. 2013;3(1):1–9.

79. Pevzner PA, Tang H, Waterman MS. An Eulerian path approach to DNA fragment assembly. Proceedings of the National Academy of Sciences. 2001;98(17):9748–9753.

80. Do CB, Batzoglou S. What is the expectation maximization algorithm? Nature biotechnology. 2008;26(8):897–900.

81. Molnar M, Ilie L. Correcting Illumina data. Briefings in Bioinformatics. 2014;p. bbu029.

82. Yin X, Song Z, Dorman K, Ramamoorthy A. PREMIERPRobabilistic error-correction using Markov inference in errored reads. In: Information Theory Proceedings (ISIT), 2013 IEEE International Symposium on. IEEE; 2013. p. 1626–1630.

83. Yin X, Song Z, Dorman K, Ramamoorthy A. PREMIER Turbo: Probabilistic error-correction using Markov inference in errored reads using the turbo principle. In: Global Conference on Signal and Information Processing (GlobalSIP), 2013 IEEE. IEEE; 2013. p. 73–76.

84. Yang X, Dorman KS, Aluru S. Reptile: representative tiling for short read error correction. Bioinformatics. 2010;26(20):2526–2533.

85. Zhao X, Palmer LE, Bolanos R, Mircean C, Fasulo D, Wittenberg GM. EDAR: an efficient error detection and removal algorithm for next generation sequencing data. Journal of computational biology. 2010;17(11):1549–1560.

86. Medvedev P, Scott E, Kakaradov B, Pevzner P. Error correction of high-throughput sequencing datasets with non-uniform coverage. Bioinformatics. 2011;27(13):i137–i141.

87. Yang X, Aluru S, Dorman KS. Repeat-aware modeling and correction of short read errors. BMC bioinformatics. 2011;12(Suppl 1):S52.

88. 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.

89. Shi H, Schmidt B, Liu W, Müller-Wittig W. Quality-score guided error correction for short-read sequencing data using CUDA. Procedia Computer Science. 2012;1(1):1129–1138.

90. Shah AR, Chockalingam S, Aluru S. A parallel algorithm for spectrum-based short read error correction. In: Parallel & Distributed Processing Symposium (IPDPS), 2012 IEEE 26th International. IEEE; 2012. p. 60–70.

91. Nikolenko SI, Korobeynikov AI, Alekseyev MA. BayesHammer: Bayesian clustering for error correction in single-cell sequencing. BMC genomics. 2013;14(Suppl 1):S7.

92. Marçais G, Yorke JA, Zimin A. QuorUM: an error corrector for Illumina reads. arXiv preprint arXiv:13073515. 2013;.

93. Ilie L, Molnar M. RACER: Rapid and accurate correction of errors in reads. Bioinformatics. 2013;p. btt407.

94. Song L, Florea L, Langmead B. Lighter: fast and memory-efficient error correction without counting. bioRxiv. 2014;.

95. Milicchio F, Prosperi MC. HErCoOl: High-throughput Error Correction by Oligomers. In: Computer-Based Medical Systems (CBMS), 2014 IEEE 27th International Symposium on. IEEE; 2014. p. 227–232.

96. Lim EC, Müller J, Hagmann J, Henz SR, Kim ST, Weigel D. Trowel: a fast and accurate error correction module for Illumina sequencing reads. Bioinformatics. 2014;p. btu513.

97. Heo Y, Wu XL, Chen D, Ma J, Hwu WM. BLESS: Bloom filter-based error correction solution for high-throughput sequencing reads. Bioinformatics. 2014;p. btu030.

98. Greenfield P, Duesing K, Papanicolaou A, Bauer DC. Blue: correcting sequencing errors using consensus and context. Bioinformatics. 2014;p. btu368.

99. Li H. BFC: correcting Illumina sequencing errors. Bioinformatics. 2015;p. btv290.

100. Duma D, Cordero F, Beccuti M, Ciardo G, Close TJ, Lonardi S. Scrible: Ultra-Accurate Error-Correction of Pooled Sequenced Reads. In: Algorithms in Bioinformatics. Springer; 2015. p. 162–174.

101. Sheikhizadeh S, de Ridder D. ACE: Accurate Correction of Errors using K-mer tries. Bioinformatics. 2015;p. btv332.

102. Ramachandran A, Heo Y, Hwu Wm, Ma J, Chen D. FPGA accelerated DNA error correction. In: Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition. EDA Consortium; 2015. p. 1371–1376.

103. Marinier E, Brown DG, McConkey BJ. Pollux: platform independent error correction of single and mixed genomes. BMC bioinformatics. 2015;16(1):10.

104. Gu Y, Liu X, Zhu Q, Dong Y, Brown CT, Pramanik S. A new method for DNA sequencing error verification and correction via an on-disk index tree. In: Proceedings of the 6th ACM Conference on Bioinformatics, Computational Biology and Health Informatics. ACM; 2015. p. 503–504.

105. Miclotte G, Heydari M, Demeester P, Audenaert P, Fostier J. Jabba: Hybrid Error Correction for Long Sequencing Reads Using Maximal Exact Matches. In: Algorithms in Bioinformatics. Springer; 2015. p. 175–188.

106. Salmela L. Correction of sequencing errors in a mixed set of reads. Bioinformatics. 2010;26(10):1284–1290.

107. Zhao Z, Yin J, Zhan Y, Xiong W, Li Y, Liu F. PSAEC: an improved algorithm for short read error correction using partial suffix arrays. Frontiers in Algorithmics and Algorithmic Aspects in Information and Management. 2011;p. 220–232.

108. Savel DM, LaFramboise T, Grama A, Koyutürk M. Suffix-Tree Based Error Correction of NGS Reads Using Multiple Manifestations of an Error. In: Proceedings of the International Conference on Bioinformatics, Computational Biology and Biomedical Informatics. ACM; 2013. p. 351.

109. Kao WC, Chan AH, Song YS. ECHO: a reference-free short-read error correction algorithm. Genome research. 2011;21(7):1181–1192.

110. Chen CC, Chang YJ, Chung WC, Lee DT, Ho JM. CloudRS: An error correction algorithm of high-throughput sequencing data based on scalable framework. In: Big Data, 2013 IEEE International Conference on. IEEE; 2013. p. 717–722.

111. Chung WC, Chang YJ, Lee D, Ho JM. Using geometric structures to improve the error correction algorithm of high-throughput sequencing data on MapReduce framework. In: Big Data (Big Data), 2014 IEEE International Conference on. IEEE; 2014. p. 784–789.

112. Allam A, Kalnis P, Solovyev V. Karect: accurate correction of substitution, insertion and deletion errors for next-generation sequencing data. Bioinformatics. 2015;31(21):3421–3428.

113. Artyomenko A, Mancuso N, Skums P, Mandoiu I, Zelikovsky A. kGEM: An Expectation Maximization Error Correction Algorithm for Next Generation Sequencing of Amplicon-based Data. In: Proc. International Symposium Bioinformatics Research and Applications; 2013. .

114. Shen J, Cong Q, Grishin NV. The complete mitochondrial genome of Papilio glaucus and its phylogenetic implications. Meta gene. 2015;5:68–83.

115. Wang C, Grohme MA, Mali B, Schill RO, Frohme M. Towards decrypting cryptobiosis-analyzing anhydrobiosis in the tardigrade milnesium tardigradum using transcriptome sequencing. PloS one. 2014;9(3).

116. MacManes MD, Eisen MB. Characterization of the transcriptome, nucleotide sequence polymorphism, and natural selection in the desert adapted mouse Peromyscus eremicus. PeerJ. 2014;2:e642.

117. Alkio M, Jonas U, Declercq M, Van Nocker S, Knoche M. Transcriptional dynamics of the developing sweet cherry (Prunus avium L.) fruit: sequencing, annotation and expression profiling of exocarp-associated genes. Horticulture Research. 2014;1.

118. Gupta V, Markmann K, Pedersen CN, Stougaard J, Andersen SU. shortran: a pipeline for small RNA-seq data analysis. Bioinformatics. 2012;28(20):2698–2700.

119. Henkel CV, Dirks RP, Jansen HJ, Forlenza M, Wiegertjes GF, Howe K, et al. Comparison of the exomes of common carp (Cyprinus carpio) and zebrafish (Danio rerio). Zebrafish. 2012;9(2):59–67.

120. Dominova I, Sorokin D, Kublanov I, Patrushev M, Toshchakov S. Complete genome sequence of Salinarchaeum sp. strain HArcht-Bsk1T, isolated from hypersaline Lake Baskunchak, Russia. Genome announcements. 2013;1(4):e00505–13.

121. Riedel T, Fiebig A, Petersen J, Gronow S, Kyrpides NC, Göker M, et al. Genome sequence of the Litoreibacter arenae type strain (DSM 19593T), a member of the Roseobacter clade isolated from sea sand. Standards in genomic sciences. 2013;9(1):117.

122. Xu Q, Chen LL, Ruan X, Chen D, Zhu A, Chen C, et al. The draft genome of sweet orange (Citrus sinensis). Nature genetics. 2013;45(1):59–66.

123. Jiao WB, Huang D, Xing F, Hu Y, Deng XX, Xu Q, et al. Genome-wide characterization and expression analysis of genetic variants in sweet orange. The Plant Journal. 2013;75(6):954–964.

124. Lada AG, Stepchenkova EI, Waisertreiger I, Noskov VN, Dhar A, Eudy JD, et al. Genome-wide mutation avalanches induced in diploid yeast cells by a base analog or an APOBEC deaminase. PLoS Genet. 2013;9(9):e1003736.

125. Fujimoto MS, Bodily PM, Okuda N, Clement MJ, Snell Q. Effects of error-correction of heterozygous next-generation sequencing data. BMC bioinformatics. 2014;15(Suppl 7):S3.

126. Schatz MC, Maron LG, Stein JC, Wences AH, Gurtowski J, Biggers E, et al. Whole genome de novo assemblies of three divergent strains of rice, Oryza sativa, document novel gene space of aus and indica. Genome biology. 2014;15(11):506.

127. Coates RC, Podell S, Korobeynikov A, Lapidus A, Pevzner P, Sherman DH, et al. Characterization of cyanobacterial hydrocarbon composition and distribution of biosynthetic pathways. PloS one. 2014;9(1).

128. Rödelsperger C, Neher RA, Weller AM, Eberhardt G, Witte H, Mayer WE, et al. Characterization of genetic diversity in the nematode Pristionchus pacificus from population-scale resequencing data. Genetics. 2014;196(4):1153–1165.

129. Nguyen TT, Hayes BJ, Ingram BA. Genetic parameters and response to selection in blue mussel (Mytilus galloprovincialis) using a SNP-based pedigree. Aquaculture. 2014;420:295–301.

130. Nobu MK, Tamaki H, Kubota K, Liu WT. Metagenomic characterization of Candidatus Defluviicoccus tetraformis strain TFO71, a tetrad-forming organism, predominant in an anaerobic–aerobic membrane bioreactor with deteriorated biological phosphorus removal. Environmental microbiology. 2014;16(9):2739–2751.

131. Suzuki S, Horinouchi T, Furusawa C. Prediction of antibiotic resistance by gene expression profiles. Nature communications. 2014;5.

132. Hill-Cawthorne GA, Hudson LO, El Ghany MFA, Piepenburg O, Nair M, Dodgson A, et al. Recombinations in Staphylococcal Cassette Chromosome ¡italic¿mec¡/italic¿ Elements Compromise the Molecular Detection of Methicillin Resistance in ¡italic¿Staphylococcus aureus¡/italic¿. PLoS ONE. 2014 06;9(6).

133. Koch P, Platzer M, Downie BR. RepARK-de novo creation of repeat libraries from whole-genome NGS reads. Nucleic acids research. 2014;p. gku210.

134. Busk PK, Lange M, Pilgaard B, Lange L. Several genes encoding enzymes with the same activity are necessary for aerobic fungal degradation of cellulose in nature. PloS one. 2014;9(12):e114138.

135. Gilchrist AS, Shearman DC, Frommer M, Raphael KA, Deshpande NP, Wilkins MR, et al. The draft genome of the pest tephritid fruit fly Bactrocera tryoni: resources for the genomic analysis of hybridising species. BMC genomics. 2014;15(1):1153.

136. Burke GR, Walden KKO, Whitfield JB, Robertson HM, Strand MR. Widespread Genome Reorganization of an Obligate Virus Mutualist. PLoS Genet. 2014 09;10(9).

137. Kenny NJ, Namigai EK, Marlétaz F, Hui JH, Shimeld SM. Draft genome assemblies and predicted microRNA complements of the intertidal lophotrochozoans Patella vulgata (Mollusca, Patellogastropoda) and Spirobranchus (Pomatoceros) lamarcki (Annelida, Serpulida). Marine genomics. 2015;.

138. Fitak RR, Mohandesan E, Corander J, Burger PA. The de novo genome assembly and annotation of a female domestic dromedary of North African origin. Molecular ecology resources. 2015;.

139. Le Duc D, Renaud G, Krishnan A, Almén MS, Huynen L, Prohaska SJ, et al. Kiwi genome provides insights into evolution of a nocturnal lifestyle. Genome biology. 2015;16(1):1–15.

140. Fiebig A, Loof TG, Babbar A, Itzek A, Koehorst JJ, Schaap PJ, et al. Comparative Genomics of Streptococcus pyogenes M1 isolates differing in virulence and propensity to cause systemic infection in mice. International Journal of Medical Microbiology. 2015;305(6):532–543.

141. Lambert D, Carrillo CD, Koziol AG, Manninger P, Blais BW. GeneSippr: A Rapid Whole-Genome Approach for the Identification and Characterization of Foodborne Pathogens such as Priority Shiga Toxigenic ¡italic¿Escherichia coli¡/italic¿. PLoS ONE. 2015 04;10(4).

142. Cong Q, Borek D, Otwinowski Z, Grishin NV. Tiger Swallowtail Genome Reveals Mechanisms for Speciation and Caterpillar Chemical Defense. Cell reports. 2015;10(6):910–919.

143. Jnemann S, Prior K, Albersmeier A, Albaum S, Kalinowski J, Goesmann A, et al. GABenchToB: A Genome Assembly Benchmark Tuned on Bacteria and Benchtop Sequencers. PLoS ONE. 2014 09;9(9).

144. Walter MC, Öhrman C, Myrtennäs K, Sjödin A, Byström M, Larsson P, et al. Genome sequence of Coxiella burnetii strain Namibia. Standards in genomic sciences. 2014;9:22.

145. Nikolaichik Y, Gorshkov V, Gogolev Y, Valentovich L, Evtushenkov A. Genome sequence of Pectobacterium atrosepticum strain 21A. Genome announcements. 2014;2(5).

146. Engel P, Stepanauskas R, Moran NA. Hidden Diversity in Honey Bee Gut Symbionts Detected by Single-Cell Genomics. PLoS Genet. 2014 09;10(9).

147. Aeschlimann SH, Jönsson F, Postberg J, Stover NA, Petera RL, Lipps HJ, et al. The draft assembly of the radically organized Stylonychia lemnae macronuclear genome. Genome biology and evolution. 2014;6(7):1707–1723.

148. Kleigrewe K, Almaliti J, Tian IY, Kinnel RB, Korobeynikov A, Monroe EA, et al. Combining Mass Spectrometric Metabolic Profiling with Genomic Analysis: A Powerful Approach for Discovering Natural Products from Cyanobacteria. Journal of natural products. 2015;78(7):1671–1682.

149. Grob C, Taubert M, Howat AM, Burns OJ, Dixon JL, Richnow HH, et al. Combining metagenomics with metaproteomics and stable isotope probing reveals metabolic pathways used by a naturally occurring marine methylotroph. Environmental microbiology. 2015;.

150. Neumann AM, Balmonte JP, Berger M, Giebel HA, Arnosti C, Voget S, et al. Different utilization of alginate and other algal polysaccharides by marine Alteromonas macleodii ecotypes. Environmental microbiology. 2015;.

151. Lada AG, Kliver SF, Dhar A, Polev DE, Masharsky AE, Rogozin IB, et al. Disruption of Transcriptional Coactivator Sub1 Leads to Genome-Wide Redistribution of Clustered Mutations Induced by APOBEC in Active Yeast Genes. PLoS Genet. 2015 05;11(5).

152. Boudreau PD, Monroe EA, Mehrotra S, Desfor S, Korobeynikov A, Sherman DH, et al. Expanding the Described Metabolome of the Marine Cyanobacterium Moorea producens JHB through Orthogonal Natural Products Workflows. PloS one. 2015;10(7).

153. De Wit P, Pespeni MH, Palumbi SR. SNP genotyping and population genomics from expressed sequences–current advances and future possibilities. Molecular ecology. 2015;24(10):2310–2323.

154. Taniguti LM, Schaker PD, Benevenuto J, Peters LP, Carvalho G, Palhares A, et al. Complete genome sequence of Sporisorium scitamineum and biotrophic interaction transcriptome with sugarcane. PloS one. 2015;10(6):e0129318.

155. Li X, Fan D, Zhang W, Liu G, Zhang L, Zhao L, et al. Outbred genome sequencing and CRISPR/Cas9 gene editing in butterflies. Nature communications. 2015;6.

156. DAgostino N, Golas T, Van de Geest H, Bombarely A, Dawood T, Zethof J, et al. Genomic analysis of the native European Solanum species, S. dulcamara. BMC genomics. 2013;14(1):356.

157. Ištvánek J, Jaroš M, Křenek A, Řepková J. Genome assembly and annotation for red clover (Trifolium pratense; Fabaceae). American journal of botany. 2014;101(2):327–337.

158. Yang R, Dai Z, Chen S, Chen L. MicroRNA-mediated gene regulation plays a minor role in the transcriptomic plasticity of cold-acclimated zebrafish brain tissue. BMC genomics. 2011;12(1):605.

159. Zawada AM, Rogacev KS, Rotter B, Winter P, Marell RR, Fliser D, et al. SuperSAGE evidence for CD14++ CD16+ monocytes as a third monocyte subset. Blood. 2011;118(12):e50–e61.

160. Roccaro M, Ahmadinejad N, Colby T, Somssich IE. Identification of functional cis-regulatory elements by sequential enrichment from a randomized synthetic DNA library. BMC plant biology. 2013;13(1):164.

161. Ollier M, Radosevic-Robin N, Kwiatkowski F, Ponelle F, Viala S, Privat M, et al. DNA repair genes implicated in triple negative familial non-BRCA1/2 breast cancer predisposition. American journal of cancer research. 2015;5(7):2113.

162. Fertin G, Jean G, Radulescu A, Rusu I. Hybrid de novo tandem repeat detection using short and long reads. BMC medical genomics. 2015;8(Suppl 3):S5.

163. Yan L, Wang X, Liu H, Tian Y, Lian J, Yang R, et al. The Genome of Dendrobium officinale Illuminates the Biology of the Important Traditional Chinese Orchid Herb. Molecular plant. 2014;.

164. Weirather JL, Afshar PT, Clark TA, Tseng E, Powers LS, Underwood J, et al. Characterization of fusion genes and the significantly expressed fusion isoforms in breast cancer by hybrid sequencing. Nucleic Acids Research. 2015;p. 1.

165. Mulley JF, Hargreaves AD. Snake venom gland cDNA sequencing using the Oxford Nanopore MinION portable DNA sequencer. bioRxiv. 2015;p. 025148.

166. Engström PG, Tommei D, Stricker SH, Ender C, Pollard SM, Bertone P. Digital transcriptome profiling of normal and glioblastoma-derived neural stem cells identifies genes associated with patient survival. Genome medicine. 2012;4(10):1–20.

167. Cox MP, Peterson DA, Biggs PJ. SolexaQA: At-a-glance quality assessment of Illumina second-generation sequencing data. BMC bioinformatics. 2010;11(1):485.

168. Rizk G, Lavenier D, Chikhi R. DSK: k-mer counting with very low memory usage. Bioinformatics. 2013;p. btt020.

169. Deorowicz S, Debudaj-Grabysz A, Grabowski S. Disk-based k-mer counting on a PC. BMC bioinformatics. 2013;14(1):160.

170. Marçais G, Kingsford C. A fast, lock-free approach for efficient parallel counting of occurrences of k-mers. Bioinformatics. 2011;27(6):764–770.

171. Zerbino DR, Birney E. Velvet: algorithms for de novo short read assembly using de Bruijn graphs. Genome research. 2008;18(5):821–829.

172. Chaisson M, Pevzner P, Tang H. Fragment assembly with short reads. Bioinformatics. 2004;20(13):2067–2074.

173. Shi H, Schmidt B, Liu W, Muller-Wittig W. Accelerating error correction in high-throughput short-read DNA sequencing data with CUDA. In: Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on. IEEE; 2009. p. 1–8.

174. Fichot EB, Norman RS. Microbial phylogenetic profiling with the Pacific Biosciences sequencing platform. Microbiome. 2013;1(1):10.

175. Bentley DR, Balasubramanian S, Swerdlow HP, Smith GP, Milton J, Brown CG, et al. Accurate whole human genome sequencing using reversible terminator chemistry. Nature. 2008;456(7218):53–59.

176. Hillier LW, Marth GT, Quinlan AR, Dooling D, Fewell G, Barnett D, et al. Whole-genome sequencing and variant discovery in C. elegans. Nature methods. 2008;5(2):183–188.

177. Kozarewa I, Ning Z, Quail MA, Sanders MJ, Berriman M, Turner DJ. Amplification-free Illumina sequencing-library preparation facilitates improved mapping and assembly of (G+ C)-biased genomes. Nature methods. 2009;6(4):291–295.

178. Yoo AB, Jette MA, Grondona M. SLURM: Simple linux utility for resource management. In: Job Scheduling Strategies for Parallel Processing. Springer; 2003. p. 44–60.

179. Li R, Li Y, Kristiansen K, Wang J. SOAP: short oligonucleotide alignment program. Bioinformatics. 2008;24(5):713–714.

180. Li H, Durbin R. Fast and accurate short read alignment with Burrows–Wheeler transform. Bioinformatics. 2009;25(14):1754–1760.

181. Miller JR, Koren S, Sutton G. Assembly algorithms for next-generation sequencing data. Genomics. 2010;95(6):315–327.

182. Bradnam KR, Fass JN, Alexandrov A, Baranay P, Bechner M, Birol I, et al. Assemblathon 2: evaluating de novo methods of genome assembly in three vertebrate species. GigaScience. 2013;2(1):1–31.

183. Darling AE, Tritt A, Eisen JA, Facciotti MT. Mauve assembly metrics. Bioinformatics. 2011;27(19):2756–2757.

184. Hernandez D, François P, Farinelli L, Østerås M, Schrenzel J. De novo bacterial genome sequencing: millions of very short reads assembled on a desktop computer. Genome research. 2008;18(5):802–809.

185. Massingham T, Goldman N. simNGS and simLibrary–software for simulating next-gen sequencing data; 2012.

186. Holtgrewe M. Mason–a read simulator for second generation sequencing data. Technical Report FU Berlin. 2010;.

187. Huang W, Li L, Myers JR, Marth GT. ART: a next-generation sequencing read simulator. Bioinformatics. 2012;28(4):593–594.

188. Ono Y, Asai K, Hamada M. PBSIM: PacBio reads simulatortoward accurate genome assembly. Bioinformatics. 2013;29(1):119–121.

189. Dohm JC, Lottaz C, Borodina T, Himmelbauer H. SHARCGS, a fast and highly accurate short-read assembly algorithm for de novo genomic sequencing. Genome research. 2007;17(11):1697–1706.

190. Chevreux B, Wetter T, Suhai S. Genome sequence assembly using trace signals and additional sequence information. In: German Conference on Bioinformatics; 1999. p. 45–56.

Further Reading

Rodrìguez-Ezpeleta N., Hackenberg M. and Aransay AM., eds. Bioinformatics for high throughput sequencing. Springer Science & Business Media, 2011.

http://www.ebi.ac.uk/training/online/course/ebi-next-generation-sequencing-practical-course (accessed January 18 2015)

https://nanoporetech.com/applications/publications (accessed January 18 2015)

Cross-References

DNA base pairs, Similarity searching, Machine learning methods

Supplementary Information

"Supplementary_Material.pdf" - *Additional information related to Error Correction, more testing results and how and on which organisms/datasets each algorithm was tested. It can be accessed by anyone (no sign in or Google account needed) at* `https://drive.google.com/open?id=0B_SKSUhe17RVaTBIcmJncFctU0E`