An Empirical Evaluation of Error Correction Methods and Tools for Next Generation Sequencing Data

Atif Mehmood
Riphah Institute of Computing and Applied Sciences (RICAS)
Riphah International University
Lahore, Pakistan

Javed Ferzund, Muhammad Usman Ali, Abbas Rehman, Shahzad Ahmed
Department of Computer Science
COMSATS Institute of Information Technology
Sahiwal, Pakistan

Imran Ahmad
Riphah Institute of Computing and Applied Sciences (RICAS)
Riphah International University
Lahore, Pakistan

***Abstract*—Next Generation Sequencing (NGS) technologies produce massive amount of low cost data that is very much useful in genomic study and research. However, data produced by NGS is affected by different errors such as substitutions, deletions or insertion. It is essential to differentiate between true biological variants and alterations occurred due to errors for accurate downstream analysis. Many types of methods and tools have been developed for NGS error correction. Some of these methods only correct substitutions errors whereas others correct multi types of data errors. In this article, a comprehensive evaluation of three types of methods (k-spectrum based, Multi-sequencing alignment and Hybrid based) is presented which are implemented and adopted by different tools. Experiments have been conducted to compare the performance based on runtime and error correction rate. Two different computing platforms have been used for the experiments to evaluate effectiveness of runtime and error correction rate. The mission and aim of this comparative evaluation is to provide recommendations for selection of suitable tools to cope with the specific needs of users and practitioners. It has been noticed that k-mer spectrum based methodology generated superior results as compared to other methods. Amongst all the tools being utilized, Racer has shown eminent performance in terms of error correction rate and execution time for both small as well as large data sets. In multisequence alignment based tools, Karect depicts excellent error correction rate whereas Coral shows better execution time for all data sets. In hybrid based tools, Jabba shows better error correction rate and execution time as compared to brownie. Computing platforms mostly affect execution time but have no general effect on error correction rate.**

***Keywords*—Next generation sequencing; bioinformatics; errors; error correction; execution time; k-spectrum; suffix tree based; hybrid based**

I. Introduction

Gigantic amount of data is originated with the help of next generation sequencing technologies at lowest cost and high throughput. As compared to old generation of sequencing data (the first-generation technology) for example Sanger NGS data faces high challenges of error rate. NGS plays a leading role in the discovery of many applications in bioinformatics research and changed the way of genomic research [1]. NGS demands high-power CPU and various algorithms that can work in parallel mode for bioinformatics studies. It also needs the spacious memory and execution time for total data that may cause issues for data management. NGS takes advantage of big data computing infrastructure that divides the memory in clusters and provides the batch queue system which helps to produce large amount of sequencing reads [2]. Errors in sequencing data mainly occur due to the replacement of correct bases with incorrect bases and indels. NGS technologies produce different tools such as Illumina and Solid to induce the substitution error, whereas the Roche 454 and Ion torrent create the insertion and deletion error. Most of the tools and methods focus on removing the substitution errors [3]. There are three types of biases that cause errors in sequencing data: systematic bias, coverage bias and batch effect bias. The rate of error in data is also different for various NGS technologies. It is key step to remove the data error before any analysis can be made. These errors also disturb the accuracy of algorithm therefore it is beneficial to rectify data before analysis to conclude better results in downstream analysis [4].

Correction of sequencing errors is a critical module for bioinformatics discovery. The basic concept behind correcting the sequencing read errors is to correct the erroneous bases. Many error correction tools subjective of different data structures related to various methods have been developed. The error correction methods are classified into four categories:

1) K-spectrum based method such as Quake (2010), Lighter (2014) [5], Reptile (2010) [6], BLESS (2014) [7] hammer (2011), Musket (2013), HECTOR (2014) and RACER (2013). These tools correct the errors on k-mer incidence.

2) Multiple sequence alignment based method such as Karect (2015), Coral (2011) and ECHO (2011).

3) Suffix tree based method such as SHREC (2009) [8].

4) Hybrid based method such as LoRDEC (2014) [9], Jabba (2016) and Brownie (2015).

Different error correction tools and algorithms have evolved with the passage of time possessing better accuracy and minimum execution time. Evaluation of specific tools is a study matter being provided by various educational sources. In this comparative study, three methods and six tools are selected, each pair of tools belonging to each method. Musket and RACER are selected from the K-spectrum based category, Coral and Karect are selected from the multiple sequence alignment categories, and Jabba and Brownie are selected from the Hybrid based category. These tools run on two different computing platforms. This piece of study aims to answer the following questions:

- Do these tools cope with data scalability?
- Does the computing platform affect the performance of tools?
- Which method of error correction is better?
- Which tool outperforms other tools?
- Which tool is better within the same category?
- Which tool has maximum error correction rate?
- Which tool requires minimum execution time on same dataset?

In addition, performance of different tools will be evaluated for different data sets. The rest of the paper is organized as follows: Section 2 describes the related work and Section 3 presents the experimental details. Results are discussed in Section 4 and paper is concluded in Section 5.

II. Related Work

Error correction depends on read coverage and error correction rate of different tools. These tools are based on different approaches and data structures. Three main approaches are used to make error correction tools more efficient such as k-spectrum based, suffix array based and hybrid based approach. Li et al. [5] has developed tool that depends on k-mer spectrum based. The authors used 31-55 k-mer length as well as bloom filter and hash table data structure. The authors focused on the removal of substitution errors. They also checked the trusted regions and extracted the optimal solution by using extension mechanism. In this task of material study, the experimental results are targeted on achieving maximum error correction rate. Heo et al. [7] used the hash table data structure. They used k-spectrum approach for error correction. They determined the solid minimum edit path in between solid k-mer. Using the reverse bloom filter, they changed false positive rate. During k-mer counting Bloocoo used 10 bits for storing solid k-mers. They also described the need for 4 GB memory requirement for human genome correction. Song et al. [5] developed memory efficient tool based on k-mer spectrum. The authors used bloom filter and 23 k-mer length, Sequencing reads were processed in three steps and two bloom filters were used for error correction. In this work, k-mer subsample is obtained using first bloom filter and then test is applied on each read on each position to find solid k-mer. These solid k-mers are stored and second bloom filter is applied. They used greedy approach for error correction which is also used in bless. Lighter corrected substitution errors. They used multiple sequence alignment method and suffix array based data structure. In his paper, two-sided error correction technique was used to correct substitutions errors. Salmela et al. [9] presented hybrid based error correction using de Bruijn graph. They corrected most weak left and right regions by choosing traversal paths in graph. The authors argued that LoRDEC consumes less memory as compared to other tools and error correction rate is 99%. In fiona, used partial suffix array with hierarchical statically method to correct errors in sequencing reads. They used each read r as reference and corrected first overlap reads. It is also able to corrected substitutions errors produced by illumina platform. They argued that fiona can process the data on inexpensive hardware. The authors used the hash table data structure and confusion matrix error model. Their technique is sufficient to correct short reads without using reference genome.

III. Experimental Design

For the experiments, six tools are selected based on their reviews. These tools belong to three methods of error correction. Two different computing platforms are used to run these tools. Four datasets of different sizes are used for the experiments. Details are given below:

A. Tools

A brief description of the selected tools is presented in Table I.

Coral is used for multiple alignments of short reads to correct the error. It is the first approach used for the short-read sequencing. Coral can easily understand and run on the data produced by different NGS technologies. It can also read data coming from single molecule sequencing technology. Coral works by first indexing the reads. All the k-mers that are valuable in total data are indexed two times into forward and reverse directions. After this process the list of k-mers are stored in hash table. Next step after indexing is multiple alignments; every alignment depends on base that being generated from neighborhood based read. This alignment helps to correct the overall data and look over the overlapping k-mer read [10]. After the comparison, the new reads are produced to have minimum error rate. Coral is superior approach as compared to SHREC, and Reptile.

Karect also belongs to the same category as Coral; however, its working differs from it. Karect uses each read as a reference and stores results in partial order graph (POG). It is also used for multiple-alignment. It is able to correct different type of errors and handles data generated by different NGS technologies. It uses less peak memory during data processing. Its performance is outstanding against low-coverage region and high error rate of data. Karect depends on POG that accumulates partial alignment results. Alignment and normalization are performed based on correction reference reads with respect to alignment of each read [11]. **Musket** uses the k-spectrum based method. It provides more accurate results against the correction reads and has the ability to execute the large read length of data and provides high coverage level. It mainly comprises of three techniques; one-sided aggressive correction, two-sided conservative method and voting based refinement method. Its time and space complexity are good for large dataset. When compared to other programs like Reptile, SHREC and Musket, it is three times faster than these tools [12].

TABLE I. ERROR CORRECTION TOOLS

Tools	Methods	Overview of Algorithm	Error Correction Type
Karect	Multiple sequence alignment	Partial order graph is used to accumulate partial alignment results. It considers each read r as reference.	Substitution Insertion Deletion
Coral	Multiple sequence alignment	Correction with alignment uses bases from the error in the correction process. Indexing k-mers that occur in reading are connected with a hash table.	Substitution Insertion
Racer	K-mer based	Racer is linked with k-mer counting program. It also uses 2-bit encoding nucleotide and arbitrary replacement of the unknown position and K-mer stored in the hash table.	Substitution
Musket	K-mer based	It is multi-threaded program, uses a master slave model and demonstrates superior parallel scalability. One sided aggressive and voting based refinement.	Substitution
Jabba	Hybrid based	Pseudo alignment approach with seed and extend method using maximal exact matches. This method corrects third generation reads by mapping on de Bruijn graph.	Substitution Insertion Deletion
Brownie	Hybrid based	It depends on de Bruijn graph and works with the help of Jabba and Karect tool. It also needs extra libraries to run the algorithm.	Substitution

RACER is another efficient tool that shows maximum error correction accuracy, time and space complexity. RACER ignores installation of extra software for processing the data, whereas other tools have two or three extra software libraries to process data. It uses the hash table for storing the k-mer because it introduces 2-bit encoding of nucleotides for random replacement of unknown position. It has the capability to process different data formats such as fastq and fasta data. **Jabba** uses hybrid method to correct the alignment and error in third generation sequencing to map the reads on de Bruijn graph which is made for second next generation sequencing. Seudo alignment approach is mostly used by this tool. Jabba processes the data in two phases: in the first phase smaller k-mer size (K=13) are used, in the second phase results are processed on de Bruijn graph that provide the extra accuracy for given data. Jabba also uses larger k-mer size (k=75) for long reads and thus repeating the entire process. It uses less time as compared to Proovread and time consumption is more like RACER.

Brownie also uses hybrid method and supports Jabba in its methodology and techniques being adopted. It also creates the de Bruijn graph for Jabba as a result the resultant file is stored in Jabba directory and then different commands are applied to find the result of error correction (Releases. biointec/brownie. GitHub). This tool requires three extra libraries for processing the data. It provides exceptional results on small dataset.

B. *Error Correction Methods*

Tools selected for this study implement different error correction methods. These methods are presented in Table 2..

K-spectrum based method decomposes the reads and makes the set of k-mer. Mostly NGS technology introduces substitution error, so k-mer set has small distance to each other if they belong to same genome location. k-spectrum is then constructed using hashing and k-mer frequency is counted to determine the error threshold. During this process, threshold of each type of k-mer (solid k-mers and weak k-mers) is determined. Then both k-mers are compared with the help of bloom filter and results are stored in hash table [1]. These results are converted into the high multiplicity k-mers and algorithm corrects the error in erroneous regions and provides with corrected reads.

TABLE II. ERROR CORRECTION METHODS USED BY THE SELECTED TOOLS

Method Label	Method Type	Tools
M1	K-spectrum based	Musket, Racer
M2	Multi Sequencing Alignment (MSA)	Coral, Karect
M3	Hybrid based	Jabba, Brownie

Multiple sequence alignment (MSA) is used for biological sequences such as protein, DNA and RNA. Two approaches are used for MSA; iterative and progressive. MSA starts working on one sequence and then aligns step by step. Working parameters and steps differ for each type of MSA. In the progressive approach, it starts from much similar sequence and aligns the new sequence to each of the previous sequences. After that it creates the distance matrix and phylogenetic guided tree is created from the matrices. Using the guided tree, it defines the next sequence to be added for alignment and preserves the gap. These steps are repeated until the total data is converted into appropriate alignment. In the iterative approach, it starts the alignment in pair wise grouping. Selection of these pairs depends on the sequence relation on the guided tree. Progressive approach is competitively efficient as compared to the iterative approach.

Hybrid Method is suitable for third generation sequencing that produces large amount of data with high error rate. This respective method uses minimum CPU time for data processing [13].

C. Datasets

In this study, four different datasets are used that are generated by the Illumina sequencing machine. A detailed description of the datasets is presented in Table III. These datasets are selected on the basis of varying attributes such as read length, number of reads and size. Some of these datasets were previously used in correction studies. The accession numbers provide the complete details about datasets.

All the datasets are available on National Center for Biotechnology www.ncbi.org.

D. Platforms for Running the Tools

Two different computing platforms are used to evaluate the tools. A brief description of the used platforms is presented in Table IV.

Machine 1 has 2.10GHz CPU (Intel i3) with 8 GB main memory. Operating system is Ubuntu Linux version 14.04 and compiler was g++. Machine 2 has different specification such as 3.10GHz CPU (Intel i7) and 16 GB RAM. Both machines used the same version of Ubuntu Linux and compiler.

TABLE III. DATASETS USED FOR THE EVALUATION OF SELECTED TOOLS

Dataset	Species	Sequencing Platform	Accession Number	No of Bases	Size
D1	S.aureus	Illumina	SRR022868	3100M	2.3Gb
D2	S.aureus	Illumina	SRR022865	821.2M	692.1Mb
D3	Escherichia coli	Illumina	SRR022918	677.2M	386Mb
D4	C.elegans	Illumina	SRR065887	316.5M	207.7Mb

TABLE IV. LATFORMS USED TO RUN THE TOOLS

Machine Label	Processor	Installed memory	System type	Operating system	Compiler
Machine1	Core(i3) 2.10GHz	8GB	64 bit	Ubuntu Linux version 14.04	g++
Machine2	Core(i7) 3.10GHz	16GB	64 bit	Ubuntu Linux version 14.04	g++

IV. RESULTS

On Machine 1, an experiment was conducted to evaluate the error correction rate. The results are shown in Figure 1. The tools comparison shows that Musket, Racer, Coral, Karect, Jabba and Brownie are best performers on data structures D2, D1, D4, D1, D4 and D3 respectively. If we take into account the data sets, for D1 and D2 Racer, for D3 Racer and Brownie produced best results and for D4 JABBA produced best results. The result from overall perspective depicts that, Racer has shown consistent performance in terms of error correction rate on all data sets. However, on the largest dataset JABBA outperforms other tools for error correction rate. On average basis for error correction rate, Coral and Musket show middle level performance, respectively.

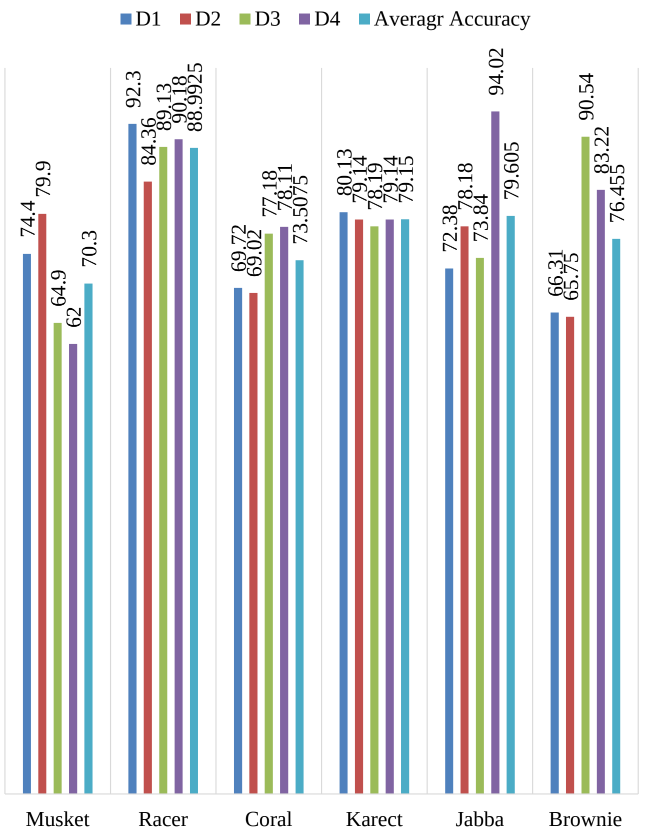

Fig. 1. Error correction rate on Machine 1.

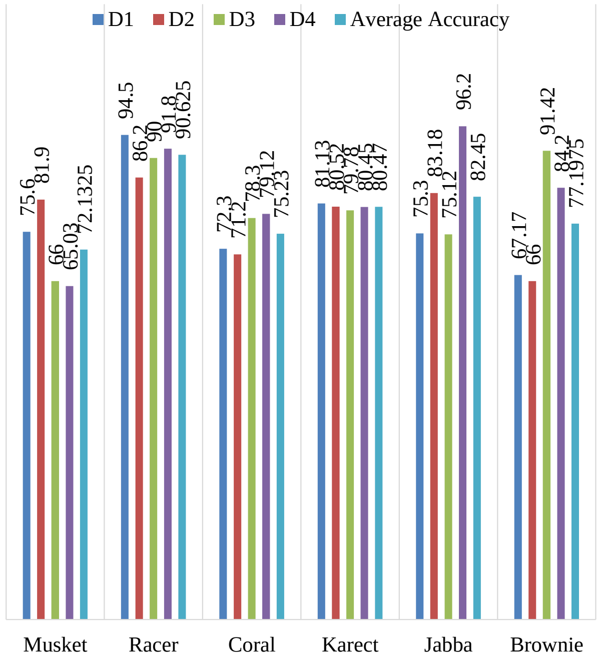

Fig. 2. Error correction rate on Machine 2.

On Machine 2, identical experiment was conducted to evaluate the error correction rate. The results are shown in Fig. 2. Comparison of various tools shows that Musket, Racer, Coral, Karect, Jabba and Brownie performed best on D2, D1, D4, D1, D4 and D3 data sets respectively. Analysis of data structures shows that for D1 and D2 RACER produced best results, for D3 Racer and Brownie produced best results and for D4 JABBA produced best results as compared to other tools. If we look at overall results, Racer has shown consistent performance in terms of error correction rate on all data sets. However, on the largest dataset JABBA is the winner for error correction rate. If we consider the average error correction rate, Coral and Musket are poor performers respectively. So, improvement in processing speed and memory does not affect the error correction rate.

On Machine 1, another experiment was conducted to evaluate the execution time required to process the data for error correction. The results are shown in Fig. 3. The comparative analysis of tools shows that, Musket, Racer, Coral, Karect, Jabba and Brownie best performed on D4, D1, D3, D1, D3, and D3 respectively. Consideration of data structures shows us that for D1 and D2 Racer produced best results, for D3 and D4 Brownie produced best results. From overall perspective, Racer has shown consistent performance in terms of time on all data sets. However, on the largest dataset Racer is at its peak of performance for execution time. On low to average basis Karect and Musket are poor performers respectively. So, enhancement in processing speed and memory reduces the execution time required for error correction. The difference is evident in the case of D4 which is the largest data set used in this study.

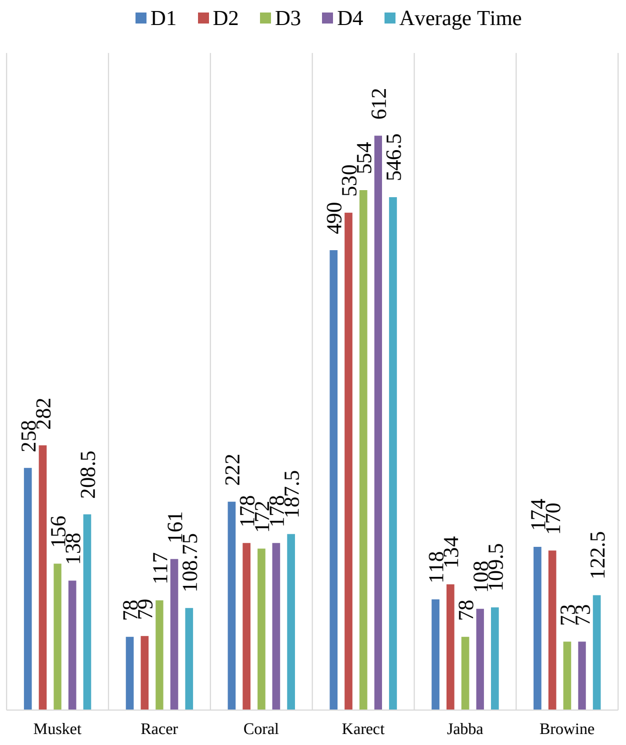

Fig. 3. Execution time on Machine 1.

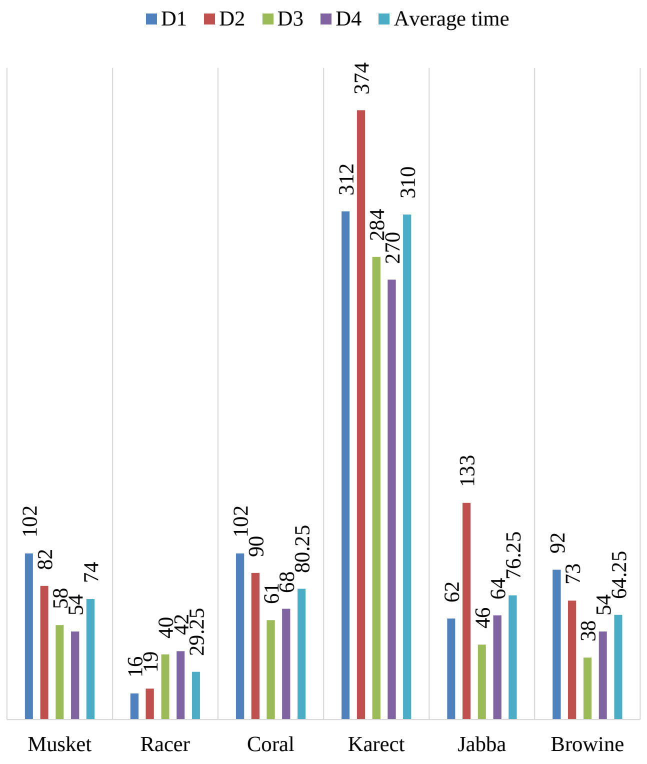

Fig. 4. Execution time on Machine 2.

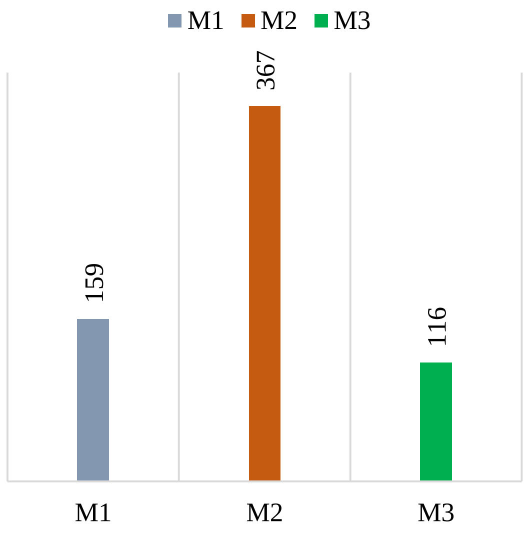

Fig. 5. Execution time on Machine 1 for three methods.

On Machine 2, same experiment was conducted to evaluate the time required to process the data for error correction. The results are shown in Fig. 4.. Figure illustrates that, Musket, Racer, Coral, Karect, Jabba and Brownie are best performers on D4, D1, D3, D4, D3 and D3, respectively. Looking on structures of data sets, it is obvious from figure that for D1 and D2 Racer produced best results, for D3 Brownie produced best results and for D4 Racer produced best results with Brownie and Musket at second position. Overall results show that Racer has shown consistent performance in terms of execution time on all data sets. However, on the largest dataset Racer acts as best performer for execution time. On low to average basis, Coral and Karect are poor performers, respectively. On Machine 1, Browine used the minimum execution time, whereas on Machine 2, Racer used the minimum execution time for D4.

On Machine 1, average execution time was calculated for each method. The results are shown in Fig. 5.. If we consider the methods, M3 (Hybrid) based tools produced best results for all data sets as compared to M1 (K-spectrum) based tools. Whereas, M2 (Multi Sequencing Alignment) based tools performed poorly.

On Machine 2, average execution time was also calculated for each method. The results are shown in Fig. 6. If we take into account various methods, M1 (K-spectrum) based tools produced best results for all data sets, as compared to M3 (Hybrid) based tools. Whereas, M2 (Multi Sequencing Alignment) based tools performed poorly. So, improvement in processing speed and memory affects the execution time required by different methods. K-Spectrum based tools perform better on high performance machines, whereas Hybrid based tools can produce better results even on lower specification machines.

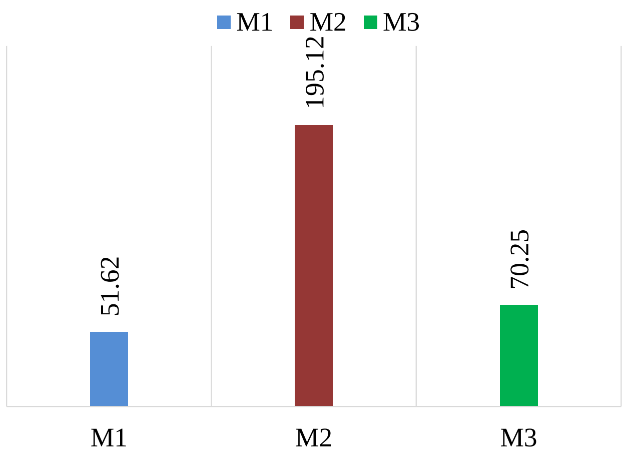

Fig. 6. Execution time on Machine 2 for three methods.

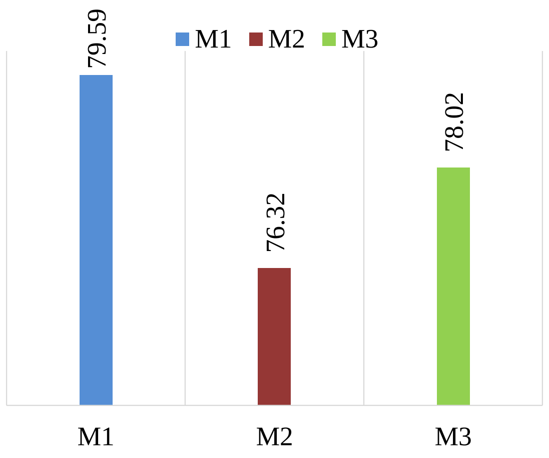

Fig. 7. Error correction rate on Machine 1 for three methods.

On Machine 1, average error correction rate was calculated for each method. The results are shown in Fig. 7.. If we consider the methods, M1 (K-spectrum) based tools produced best results for all data sets, with M3 (Hybrid) based tools at second position. Whereas, M2 (Multi Sequencing. Alignment) based tools performed poor.

On Machine 2, average error correction rate was also calculated for each method. The results are shown in Fig. 8. If we consider the methods, M1 (K-spectrum) based tools produced best results for all data sets, with M3 (Hybrid) based tools at second position. Whereas, M2 (Multi Sequencing Alignment) based tools performed poorly. So, improvement in processing speed and memory does not affect the error correction rate of different methods.

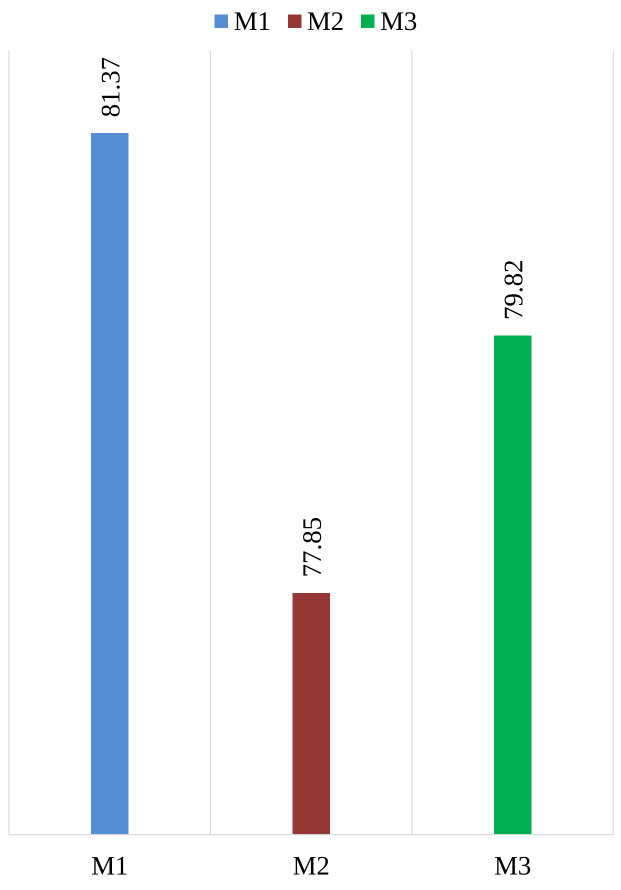

Fig. 8. Error correction rate on Machine 2 for three methods.

On the basis of the above findings, answers to the questions formulated in the Introduction section are presented below:

Do these tools cope with data scalability?

No, all tools cannot handle data scalability because the performance of these tools degrades with increase in data size. However, Racer and JABBA perform equally well on small and large datasets.

Does the computing platform affect the performance of tools?

Yes, the computing platform affects the execution time required to process data. However, it does not affect the error correction rate.

Which method of error correction is better?

K-spectrum based method produced best results with Hybrid based method at second position.

Which tool outperforms other tools?

Racer has outperformed other tools both in execution time and error correction rate. JABBA is the second-best performer.

Which tool is better with in the same category?

Racer is better than Musket in k-spectrum based category. Karect is better than Coral in terms of error correction rate, whereas Coral is better than Karect in terms of execution time in the Multiple Sequence Alignment based category. Jabba is better than Brownie in the hybrid based category.

V. Conclusions and Future Work

Among the three methods studied, k-spectrum based method generated good results as compared to other methods. Racer can perform well in error correction rate and time execution on small as well as large data sets. In multi sequence alignment based tools, Karect performed better in error correction rate whereas Coral performed better in execution time for all data sets. Jabba performs well in error correction rate and time execution; however, brownie provided good results in terms of execution time on Machine 2. These tools depend on hybrid based method. Computing platform has effect on execution time but has not significant effect on error correction rate. In future, we want to evaluate tools that can process large datasets in shorter time.

References

[1] Isaac Akogwu, Nan Wang, Chaoyang Zhang, and Ping Gong, "A comparative study of k-spectrum-based error correction methods for next-generation sequencing data analysis," Human Genomics, pp. 49-59, 2016.

[2] Xiao Yang, Sriram , Chockalingam , and Srinivas Aluru, "A survey of error-correction methods for next-generation sequencing," *BRIEFINGS IN BIOINFORMATICS*, vol. 14, pp. 56-66, 2012.

[3] Leena Salmela and Jan Schröder, "Correcting errors in short reads by multiple alignments," *bioinformatics*, vol. 27, pp. 1455-1461, 2011.

[4] Margaret A Taub, Hector Corrada Bravo, and Rafael A Irizarry, "Overcoming bias and systematic errors in next," *genome medicine*, pp. 1-5, 2010.

[5] Li Song, Liliana Florea, and Ben Langmead, "Lighter: fast and memory-efficient sequencing error correction without counting," *Genome Biology*, pp. 1-13, 2014.

[6] Xiao Yang, Karin S Dorman, and Srinivas Aluru, "Reptile: representative tiling for short read error correction," *bioinformatics*, vol. 26, pp. 2526-2533, 2010.

[7] Yun Heo, Xiao Long Wu, Deming Chen, Jian Ma, and Wen Mei Hwu, "BLESS: Bloom filter-based error correction solution for high-throughput sequencing reads," *bioinformatics*, vol. 30, pp. 1354-1362, 2014.

[8] Jan Schröder, Heiko Schröde, Simon J Puglisi, and Ranjan Sinha, "SHREC: a short-read error correction method," *bioinformatics*, vol. 25, pp. 217-2163, 2009.

[9] Leena Salmela and Eric Rivals, "LoRDEC: accurate and efficient long read error correction," *Bioinformatics*, pp. 3506-3514, 2014.

[10] Leena Salmela and Jan Schröder, "Correcting errors in short reads by multiple alignments," *bioinformatics*, vol. 27, pp. 1455-1461, 2011.

[11] Amin Allam, Panos Kalnis, and Victor Solovyev, "accurate correction of substitution,insertion and deletion errors for next-generation sequencing data," *bioinformatics*, pp. 3421-3428, 2015.

[12] Yongchao Liu, Jan Schro der, and Bertil Schmidt, "Musket: a multistage k-mer spectrum-based error corrector for Illumina sequence data," *bioinformatics*, vol. 29, pp. 308-315, 2013.

[13] Giles Miclotte et al., "Jabba: hybrid error correction for long sequencing reads," *Algorithms Mol Biol*, pp. 1-12, 2016.