Improving Metagenomic Assemblies Through Data Partitioning: A GC Content Approach

Fábio Miranda[1(✉)], Cassio Batista[1], Artur Silva[2,3], Jefferson Morais[1], Nelson Neto[1], and Rommel Ramos[1,2,3]

[1] Computer Science Graduate Program, Federal University of Pará, Belém, Brazil
{fabiomm,cassiotb,jmorais,nelsonneto,rommelramos}@ufpa.br
[2] Institute of Biological Sciences, Federal University of Pará, Belém, Brazil
asilva@ufpa.br
[3] Center of Genomics and Systems Biology, Federal University of Pará, Belém, Brazil

Abstract. Assembling metagenomic data sequenced by NGS platforms poses significant computational challenges, especially due to large volumes of data, sequencing errors, and variations in size, complexity, diversity and abundance of organisms present in a given metagenome. To overcome these problems, this work proposes an open-source, bioinformatic tool called GCSplit, which partitions metagenomic sequences into subsets using a computationally inexpensive metric: the GC content. Experiments performed on real data show that preprocessing short reads with GCSplit prior to assembly reduces memory consumption and generates higher quality results, such as an increase in the size of the largest contig and N50 metric, while both the L50 value and the total number of contigs produced in the assembly were reduced. GCSplit is available at https://github.com/mirand863/gcsplit.

Keywords: DNA sequencing · Metagenomics · Data partitioning
Bioinformatic tools · Metagenomic data preprocessing

AQ1

1 Introduction

Metagenomics consists in determining the collective DNA of microorganisms that coexist as communities in a variety of environments, such as soil, sea and even the human body [1–3]. In a sense, the field of metagenomics transcends the traditional study of genes and genomes, because it allows scientists to investigate all the organisms present in a certain community, thus allowing the possibility to infer the consequences of the presence or absence of certain microbes. For example, sequencing the gastrointestinal microbiota enables the understanding of the role played by microbial organisms in the human health [4].

Nevertheless, second generation sequencing technologies—which belong to the Next Generation Sequencing (NGS), and are still the most widespread technology on the market—are unable to completely sequence the individual genome

of each organism that comprises a metagenome. Instead, NGS platforms can sequence only small fragments of DNA from random positions, and the fragments of the different organisms are blended [5]. Hence, one of the fundamental tasks in metagenome analysis is to overlap the short reads in order to obtain longer sequences, denominated *contigs*, with the purpose of reconstructing each individual genome of a metagenome or represent the gene repertoire of a community [6]. This task is referred to as the metagenome assembly problem.

Roughly speaking, metagenome assembly can be done with or without the guidance of a reference genome. The reference assembly can be performed by aligning reads to the genomes of cultivated microbes [7]. However, this method is rather limited because the microbial diversity of most environments extends far beyond what is covered by the reference databases. Consequently, it is necessary to perform *de novo* assembly when reconstructing a metagenome that contains many unknown microorganisms.

Although it seems simple at first glance, the metagenome assembly problem is actually quite complex. Among the several challenges this task arises, there are sequencing errors specific to each platform and the processing of the large volume of data produced by NGS platforms [8]. Moreover, the problem is further complicated by variations on the size of the genomes and also by the complexity, diversity and abundance of each organism present in a microbial community [9]. For these reasons, the metagenome assembly becomes a challenging problem.

To solve all these challenges, either *de novo* assembly can be performed directly by a metagenome assembler, or the short reads can be clustered in advance in order to individually assembly each organism present in the metagenome [10]. The latter approach has the advantage of reducing the computational complexity during the metagenome assembly, because the assembler will process smaller subsets of short reads and, furthermore, it is possible to run the individual assembly of each genome in parallel, since those tasks are independent from each other. The reduction of computational complexity can also be achieved through the previous digital normalization or data partitioning prior to assembly, which reduces the dataset by removing redundant sequences, and divides it into groups of similar reads, respectively [11].

The main focus of this study is the application of the data partitioning method towards the reduction of computational complexity and the improvement of metagenome assembly. The developed computational approach, denominated GCSplit, uses the nucleotide composition of the reads, i.e., the amount of bases A, G, C and T present on DNA sequences. This decision was based on the fact that different organisms or genes that compose metagenomes have distinct GC content and different GC contents will present coverage variation, a metric used by assemblers to reconstruct the genomes, which in turn affects the k-mer selected to perform the sequence assembly based on NGS reads.

The rest of this paper is structured as follows. Related works on digital normalization and data partitioning are discussed in Sect. 2. Section 3 then presents the proposed algorithm. In Sect. 4, the impact of the new approach

on the performance of the metagenomic assembler metaSPAdes [12] is evaluated through experiments on real data. Finally, Sect. 5 presents the conclusions and plans for future works.

2   Related Work

In the literature there are several studies that attempt to reduce the computational complexity and improve metagenomic assemblies through data preprocessing techniques. The main approaches used are either digital normalization or data partitioning, the latter being the main focus of this article. In this context, the goal of this section is to carry out a bibliographical review of tools that use such methodologies.

Diginorm [13] is a tool that uses the CountMin Sketch data structure to count k-mers, with the purpose of obtaining an estimate of the sequencing coverage; and reducing coverage variation by discarding redundant data. Due to the data structure, this technique keeps a constant memory usage and a linear runtime complexity for the *de novo* assembly in relation to the amount of input data.

Trinity's *in silico* normalization (TIS) [14], which belongs to the Trinity assembler algorithm package, presents an implementation that computes the median k-mer coverage for all reads of a given dataset. If the median coverage is lower than the desired value, all reads are kept. Otherwise, the reads may be kept with a probability that is equal to the ratio of the desired coverage by the median coverage.

NeatFreq [15] clusters and selects short reads based on the median k-mer frequency. However, the main innovation in the work is the inclusion of methods for the use of paired reads alongside with preferential selection of regions with extremely low coverage. The results achieved indicate that the coverage reduction obtained by NeatFreq increased the processing speed and reduced the memory usage during the *de novo* assembly of bacterial genomes.

ORNA [16] presents a novel and interesting approach that normalizes short reads to the minimum necessary amount in order to preserve important k-mers that connect different regions of the assembly graph. The authors treat data normalization as a set multi-cover problem, and they also have proposed a heuristic algorithm. Their results show that a better normalization was achieved with ORNA, when compared with similar tools. Moreover, the size of the datasets was drastically reduced without a significant loss in the quality of the assemblies.

Khmer [17,18] presents a novel data partitioning methodology, in which the main data structure—a probabilistic model called bloom filter—is used to obtain a compact representation for graphs. The authors' implementation can represent each k-mer using only 4 bits, which was the major factor for achieving a forty-fold memory economy while assembling a soil metagenome.

MetaPrep [19] contains efficient implementations for k-mer counting, parallel sorting, and graph connectivity and partitioning. The developed solution was evaluated in a soil metagenome dataset composed of 223 Gigabases (Gb) distributed in 1.13 billion short reads. As a result of the experiment, MetaPrep took only 14 min to process this dataset using just 16 nodes of the NERSC

Edison supercomputer. The authors also assessed how MetaPrep can improve the performance of the metagenomic assembler MEGAHIT.

Latent Strain Analysis [20] is an approach that separates the DNA sequences into partitions considering the biological factor, thus allowing the individual assembly of each genome present in a metagenome. The proposed methodology assumes that the abundance of the genomes present in a sample reflects on their k-mer abundance. The results achieved allowed the partial or almost complete assembly of bacteria whose relative abundance varies to a minimum of 0.00001%.

During the literature review, it was not found any software that performs data partitioning using the information present in the nucleotide composition of short reads sequenced by NGS platforms. Hence, in this work we propose GCSplit, a tool that uses the GC content of the DNA sequences in combination with statistical metrics to partition the dataset. This new approach is promising because it is computationally inexpensive and uses information present in the reads that, as far as we know, has not been used in any other work for data partitioning. Further details about this new algorithm will follow.

3   The Proposed Algorithm

GCSplit was implemented in C++ in order to facilitate the communication with the library that performs the parallelization of the critical sections of the algorithm. The software KmerStream [21] and metaSPAdes [12], which are automatically executed to estimate the best values of k-mer and to assembly the metagenome, respectively, are dependencies of the proposed algorithm. The object-oriented programming paradigm was used in order to simplify an eventual addition of either new assemblers or k-mer estimation programs in the future, since one would only need to implement new classes to interact with the desired programs. Figure 1 summarizes the main steps of the developed algorithm.
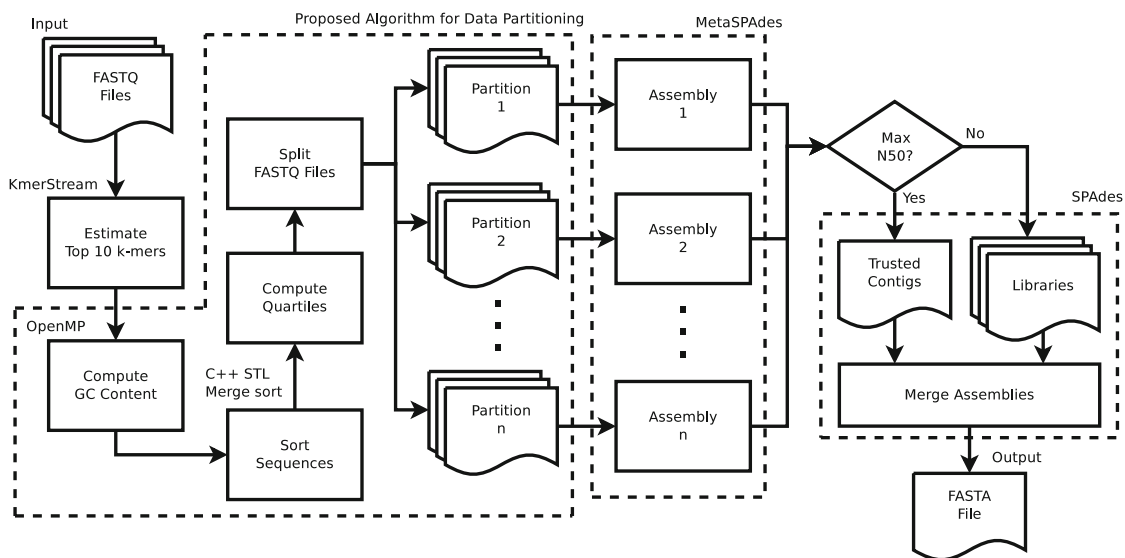

Fig. 1. GCSplit algorithm overview.

As input, GCSplit takes two paired FASTQ files, both containing the metagenome's sequences. The files are automatically passed to KmerStream, which estimates the best k-mer values for the assembly. Next, an algorithm was developed within GCSplit in order to partition the datasets, as shown in the pseudo-code of Algorithm 1. More specifically, the algorithm computes the GC content from all reads, making use of OpenMP library to do so in parallel. C++ STL library, on the other hand, has a parallel implementation of the merge sort algorithm, which is used to sort the sequences according to their GC content in ascending order.

Algorithm 1. Pseudo-code to partition the dataset

    Input   : Paired reads vectors $L_i(x)$ and $R_i(x)$, number of partitions n
    Output : n sets of paired FASTQ files

1 $s \leftarrow |L|$;
2   **for** $i \leftarrow 0$ **to** $s - 1$ **do**
3 $l \leftarrow |L_i|$;
4 $L_i.gc = 0$;
5   **for** $j \leftarrow 0$ **to** $l - 1$ **do**
6   **if** $L_i[j] =$ 'C' **or** $L_i[j] =$ 'G' **then**
7 $L_i.gc = L_i.gc + 1$;

8 $L_i.gc = \dfrac{L_i.gc \times 100}{l}$;
9 $L_i.pair \leftarrow R_i$;   `// Creates a pointer from `L_i` to `R_i
10 $\text{sort}(L, 0, s - 1)$;   `// merge sort`
11 $p \leftarrow \dfrac{s}{n}$;
12 $r \leftarrow s \bmod n$;
13 $x \leftarrow 0$;
14 $y \leftarrow 0$;
15 **for** $i \leftarrow 0$ **to** n **do**
16   **if** $r > 0$ **then**
17 $y \leftarrow y + p + 1$;
18 $r \leftarrow r - 1$;
19   **else**
20 $y \leftarrow y + p$;
21   createPartition(L, x, y - 1, string(i));
22 $x \leftarrow y$;

The proposed algorithm then calculates the approximate number of reads p that go into each partition n and also divides the dataset into n subsets/partitions based on this value. The number of partitions "n"—which is expected to be much smaller than the number of reads "s", i.e., $n << s$—is defined by the user because the ideal value may vary for each dataset. It is important to notice that this partitioning method keeps the reads paired, which is crucial in obtaining assemblies with the highest quality possible.

Procedure. createPartition(L, b, e, i)

Input : dataset vector $L_i(x)$, the beginning of the partition b, the end of the
partition e, the index of the partition i

Output : Two paired FASTQ files with reads from the selected range

1 OutputStream $r_1 \leftarrow$ New OutputStream("partition" + i + "_R1.fastq");

2 OutputStream $r_2 \leftarrow$ New OutputStream("partition" + i + "_R2.fastq");

3 **for** $j \leftarrow b$ **to** e **do**

4 $r_1 \leftarrow L_j$;

5 $r_2 \leftarrow L_j.pair$;

6 r_1.close();

7 r_2.close();

The division of the data into subsets is performed by a procedure called createPartition, whose implementation has four instructions and a single loop. Therefore, the asymptotic analysis of the second task results in a complexity of $\mathcal{O}(e - b)$, where "b" is the beginning and "e" the end of the partition.

Regarding Algorithm 1, the computation of the number of reads $|L|$ takes $\mathcal{O}(s)$; sorting with merge sort takes $\Theta(s \log s)$; dividing the dataset into "n" partitions takes $\mathcal{O}(n(b - e))$; and computing the GC content of "s" reads with length "l" takes $\mathcal{O}(sl)$ due to the nested loop ranging from lines 2 to 9. Therefore, we can conclude that the proposed partitioning algorithm has an asymptotic complexity of $\mathcal{O}(sl)$, since "s" is normally much greater than "n" ($s >> n$) and, thus, $s \times l > s \log s > n \times (b - e)$ in the worst case.

The created partitions are individually assembled by metaSPAdes. Then, another assembly is performed with SPAdes [22] to concatenate the n previous assemblies, where the assembly result that gets the highest N50 is used as input in the trusted contigs parameter and the remaining are used as single libraries. SPAdes was used in this final step because metaSPAdes does not accept multiple libraries as input yet. Ultimately, the final output is a FASTA file that contains a high quality assembly.

4   Evaluation

4.1   Datasets and Platform

In order to evaluate GCSplit, analysis were conducted in three real metagenomic datasets, whose samples were collected from the following environments: from a moose rumen, from hot springs at headwaters and a sewage treatment plant. The datasets used in the experiments are listed in Table 1.

The moose rumen sample was collected in Växsjö, Sweden. This sample was sequenced with the Illumina HiSeq 2500 platform and contains 25, 680, 424 paired short reads with 101 base pairs (bp). These short reads can be obtained from the Sequence Read Archive (SRA) database through the accession number ERR1278073.

Table 1. Dataset description

ID	Dataset	Read count R ($\times 10^6$)	Size M (Gbp)	SRA accession number
MR	Moose rumen	25.6	5.2	ERR1278073
HS	Hot springs	7.4	3.7	ERR975001
STP	Sewage treatment plant	21.3	6.4	SRR2107212, SRR2107213

The hot springs samples were collected in the years 2014 and 2015 at the headwaters of Little Hot Creek, located in the Long Valley Caldera, near Mammoth Lake in California, United States [23]. Sequencing was conducted using one of the following Illumina platforms: MiSeq PE250, MiSeq PE300 or HiSeq Rapid PE250. The insert size used has approximately 400 bp and MiSeq runs were prepared using the Agilent SureSelect kit, while HiSeq PE250 samples were prepared using the Nextera XT library preparation kit. These data, which contain 7, 422, 611 short reads of 250 base pairs can be obtained via SRA accession number ERR975001.

The sludge sample was collected in a municipal sewage treatment plant located in Argentina [24]. This sample was split in two technical replicates and sequenced with the platform Illumina HiSeq 1500 in 150 bp short reads. This dataset can be downloaded via SRA accession numbers SRR2107212 and SRR2107213, which contain 8, 899, 734 and 12, 406, 582 short reads, respectively.

The assemblies were performed in a cluster that runs the GNU/Linux x64 operating system (openSUSE), and contains 4 nodes with 64 cores and 512 Gigabytes (GB) of Random Access Memory (RAM).

4.2 Results

Table 2 shows the assembly quality results produced by metaSPAdes without preprocessing and after preprocessing with GCSplit using four partitions for the metagenomes collected from the Moose Rumen (MR), the Hot Springs (HS) and the Sewage Treatment Plant (STP). The best 10 k-mers estimated by KmerStream were used in those assemblies. The statistics were computed with the tool QUAST [25], while peak memory usage was extracted from the assemblers' log. The best results are highlighted in bold.

For the MR dataset, the amount of contigs was drastically reduced from 175, 761 to 572 when preprocessing the data with GCSplit. The N50 value increased from 2, 004 bp in the assembly without preprocessing to 164, 782 bp after preprocessing, while the L50 value decreased from 23, 843 to 6 contigs. This implies that to reach the N50 value of 164, 782 bp, we need to sum the length of only 6 contigs, whereas without GCSplit 23, 843 contigs were necessary to reach a much smaller N50 value, which means that the assembly with metaSPAdes alone contains more fragmented sequences overall. There was also a reduction of 13 GB in memory consumption during the assembly with GCSplit.

Table 2. Assembly quality comparison with four partitions

MetaSPAdes assembly statistics							
Dataset	Preproc.?	#Contigs	Largest contig	Total length (Mbp)	N50	L50	Memory peak (GB)
MR	Yes	**572**	**791, 884**	4.3	**164, 782**	**6**	21
	No	175, 761	408, 484	**276.1**	2, 004	23, 843	34
HS	Yes	**98**	**193, 569**	3.1	**109, 605**	**11**	21
	No	26, 656	109, 705	**28.7**	1, 108	4, 795	45
STP	Yes	**1, 562**	**1, 020, 504**	19.6	**85, 650**	**59**	33
	No	385, 566	340, 186	**463.8**	1, 312	71, 126	76

Moreover, the largest contig produced in the MR assembly after preprocessing the dataset with GCSplit increased from 408, 484 bp to 791, 884 bp. However, the total length of the assembly generated after applying GCSplit to the data dropped from 276.1 Mbp to 4.3 Mbp. The merging strategy adopted using SPAdes may be one of the reasons for this reduction because it breaks down the contigs into smaller sequences of length k (k-mers), but other possibilities are under investigation.

The assembly of the HS sample also showed improvements with the usage of GCSplit. Memory peak dropped from 45 GB to 21 GB on partitioned data. Furthermore, the amount of contigs decreased from 26, 656 in the assembly with metaSPAdes alone to 98 contigs after preprocessing with GCSplit, representing a 99% reduction. On the other hand, the N50 value increased significantly, yielding a 9792% growth after partitioning, which is excellent for gene prediction. Moreover, the L50 value reduced about 99% with the GC content partitioning.

In the HS dataset, the size of the largest contig produced in the assembly after preprocessing was closer to the assembly with metaSPAdes alone, containing 193, 569 bp and 109, 705 bp, respectively. The total length of the assembly also experienced a less dramatical decrease, going from 28.7 Mbp in the traditional assembly to 3.1 Mbp after using GCSplit.

For the STP sample, there was a 56% economy in memory usage with prior data partitioning. Additionally, the amount of contigs drastically reduced from 385, 566 to 1, 562 contigs when the assembly was performed after preprocessing the sequences with GCSplit. The N50 value raised from 1, 312 bp in the assembly with metaSPAdes alone to 85, 650 bp after partitioning, while the L50 value decreased from 71, 126 to 59 contigs with the aid of GCSplit.

Furthermore, the largest contig produced in the STP assembly after preprocessing the dataset with GCSplit increased from 340, 186 bp to 1, 020, 504 bp. Conversely, the total length of the assembly generated after applying GCSplit to the data declined from 463.8 Mbp to 19.6 Mbp. This result shows that despite the amount of data has decreased, the N50 value improved and this can favor gene prediction in later analysis.

Another experiment was carried out to assess whether different numbers of partitions affect the computational complexity and quality of the assembly. Table 3 shows the results produced by metaSPAdes after preprocessing with GCSplit using different amounts of partition for the metagenome collected from the Hot Springs (HS).

Table 3. Assembly quality comparison for different number of partitions

MetaSPAdes assembly statistics							
Dataset	#Part.	#Contigs	Largest contig	Total length (Mbp)	N50	L50	Memory peak (GB)
HS	2	205	**349, 825**	**4.7**	50, 407	22	31
	4	82	193, 547	2.7	110, 167	10	21
	8	**66**	283, 547	2.5	**123, 532**	**6**	**19**

The results in Table 3 show that larger values of partition reduce memory consumption. However, their total length represents about half of the data produced when the dataset is divided into only two partitions, which may indicate a significant loss in gene representativity. Therefore, gene prediction analyses are necessary to validate whether the largest amount of data (observed in the assembly of less partitioned data) is proportional to the amount of predicted Open Reading Frames (ORFs) or to products identified when performing a search by homology in public databases. All things considered, the number of partitions is a flexible parameter, so the user can evaluate several options and identify the best, since variations in different organisms can occur due to the GC content of the genome.

5 Conclusion

In this work, we developed a new bioinformatic tool called GCSplit, which partitions metagenomic data into subsets using a computationally inexpensive metric: the GC content of the sequences. GCSplit has been implemented in C++ as an open-source program, which is freely available at the following repository on GitHub: https://github.com/mirand863/gcsplit. GCSplit requires GCC version 4.4.7 or higher, the library OpenMP and the software KmerStream and metaSPAdes.

Empirical results showed that applying GCSplit to the data before assembling reduces memory consumption and generates higher quality results, such as an increase in the size of the largest contig and N50 metric, while both the L50 value and the total number of contigs generated in the assembly were reduced.

Although larger number of partitions produced less data, it is important to notice that the next analysis performed after assembly is gene prediction, where larger sequences are more likely to have genes predicted, as opposed to

fragmented assemblies such as those carried out with metaSPAdes alone, which contain smaller N50 and larger amounts of bp. Moreover, metagenome binning can be favored by the generation of less fragmented results due to higher values of N50.

As future work, we expect to perform gene prediction to assess whether the contigs produced in the assembly with GCSplit contain meaningful information. Additionally, we also plan to test the application of GCSplit in eukaryotic datasets, and test the Overlap–Layout–Consensus (OLC) approach in the merging step. The experiments that would allow the comparison of GCSplit with other algorithms specialized in either digital normalization or data partitioning could not be completed by the submission deadline of this article.

References

1. Vogel, T.M., Simonet, P., Jansson, J.K., et al.: TerraGenome: a consortium for the sequencing of a soil metagenome. Nat. Rev. Microbiol. **7**, 252 (2009)
2. Venter, J.C., Remington, K., Heidelberg, J.F., et al.: Environmental genome shotgun sequencing of the Sargasso Sea. Science **304**, 66–74 (2004)
3. Qin, J., Li, R., Raes, J., et al.: A human gut microbial gene catalogue established by metagenomic sequencing. Nature **464**, 59–65 (2010)
4. Turnbaugh, P.J., Ley, R.E., Hamady, M., et al.: The human microbiome project: exploring the microbial part of ourselves in a changing world. Nature **449**, 804–810 (2007)
5. Namiki, T., Hachiya, T., Tanaka, H., et al.: MetaVelvet: an extension of Velvet assembler to De Novo metagenome assembly from short sequence reads. Nucleic Acids Res. **40**, e155 (2012)
6. Rodrigue, S., Materna, A.C., Timberlake, S., et al.: Unlocking short read sequencing for metagenomics. PLoS ONE **5**, e11840 (2010)
7. Nielsen, H.B., Almeida, M., Juncker, A.S., et al.: Identification and assembly of genomes and genetic elements in complex metagenomic samples without using reference genomes. Nat. Biotechnol. **32**, 822–828 (2014)
8. Wojcieszek, M., Pawełkowicz, M., Nowak, R., et al.: Genomes correction and assembling: present methods and tools. In: SPIE Proceedings, vol. 9290, p. 92901X (2014)
9. Charuvaka, A., Rangwala, H.: Evaluation of short read metagenomic assembly. BMC Genom. **12**, S8 (2011)
10. Rasheed, Z., Rangwala, H.: Mc-MinH: metagenome clustering using minwise based hashing. In: SIAM International Conference in Data Mining, pp. 677–685 (2013)
11. Howe, A.C., Jansson, J.K., Malfatti, S.A., et al.: Tackling soil diversity with the assembly of large, complex metagenomes. Proc. Natl. Acad. Sci. **111**, 4904–4909 (2014)
12. Nurk, S., Meleshko, D., Korobeynikov, A., et al.: metaSPAdes: a new versatile metagenomic assembler. Genome Res. **27**, 824–834 (2017)

13. Brown, C.T., Howe, A., Zhang, Q., et al.: A reference-free algorithm for computational normalization of shotgun sequencing data. arXiv:1203.4802 (2012)
14. Haas, B.J., Papanicolaou, A., Yassour, M., et al.: De Novo transcript sequence reconstruction from RNA-seq using the Trinity platform for reference generation and analysis. Nat. Protoc. **8**, 1494–1512 (2013)
15. McCorrison, J.M., Venepally, P., Singh, I., et al.: NeatFreq: reference-free data reduction and coverage normalization for De Novo sequence assembly. BMC bioinform. **15**, 357 (2014)
16. Durai, D.A., Schulz, M.H.: In-silico read normalization using set multi-cover optimization. bioRxiv:133579 (2017)
17. Pell, J., Hintze, A., Canino-Koning, R., et al.: Scaling metagenome sequence assembly with probabilistic de Bruijn graphs. Proc. Natl. Acad. Sci. **109**, 13272–13277 (2012)
18. Crusoe, M.R., Alameldin, H.F., Awad, S., et al.: The khmer software package: enabling efficient nucleotide sequence analysis. F1000Research **4**, 900 (2015)
19. Rengasamy, V., Medvedev, P., Madduri, K.: Parallel and memory-efficient preprocessing for metagenome assembly. In: IPDPSW, pp. 283–292 (2017)
20. Cleary, B., Brito, I.L., Huang, K., et al.: Detection of low-abundance bacterial strains in metagenomic datasets by eigengenome partitioning. Nat. Biotechnol. **33**, 1053–1060 (2015)
21. Melsted, P., Halldórsson, B.V.: KmerStream: streaming algorithms for k-mer abundance estimation. Bioinformatics **30**, 3541–3547 (2014)
22. Bankevich, A., Nurk, S., Antipov, D., et al.: SPAdes: a new genome assembly algorithm and its applications to single-cell sequencing. J. Comput. Biol. **19**, 455–477 (2012)
23. Stamps, B.W., Corsetti, F.A., Spear, J.R., et al.: Draft genome of a novel Chlorobi member assembled by tetranucleotide binning of a hot spring metagenome. Genome Announc. **2**, e00897–e00914 (2014)
24. Ibarbalz, F.M., Orellana, E., Figuerola, E.L., et al.: Shotgun metagenomic profiles have a high capacity to discriminate samples of activated sludge according to wastewater type. Appl. Environ. Microbiol. **82**, 5186–5196 (2016)
25. Gurevich, A., Saveliev, V., Vyahhi, N., et al.: QUAST: quality assessment tool for genome assemblies. Bioinformatics **29**, 1072–1075 (2013)