MeanSCRAPT: an iterative algorithm for clustering 16S rRNA long-read gene datasets

Ipsa Mittra
University of Maryland, College Park
College Park, United States
imittra@umd.edu

Harihara Subrahmaniam Muralidharan
University of Maryland Institute for Advanced Computer Studies
University of Maryland, College Park
College Park, United States

Tu Luan
University of Maryland Institute for Advanced Computer Studies
University of Maryland, College Park
College Park, United States

Mihai Pop
University of Maryland Institute for Advanced Computer Studies
University of Maryland, College Park
College Park, United States

ABSTRACT

Long-read sequencing technologies have furthered genomic studies by providing more in-depth information, and innovations have made this technology more cost-efficient and common. In microbiome research applications, long-read 16S rRNA sequences can potentially provide strain-level community resolution and help identify new taxa through sequence clustering. However, there are currently no existing 16S rRNA long-read sequence clustering algorithms to aid in discovering this information. We propose an iterative sampling-based 16S rRNA long-read gene sequence clustering method that overcomes variability in cluster sequences using a "mean-shifting" strategy. We analyze the iterative clustering process and its methodology to choose clusters and respective representative sequences. We show how this iterative algorithm can cluster sequences within a real human gut microbiome dataset within a reasonable timeframe and memory usage compared to other popular tools. These experiments also compare this algorithm's performance to other clustering and alignment tools: BLAST, MiniMap2, and DNACLUST as it produces less fragmented operational taxonomic units than these other tools. This algorithm is implemented in the software package MeanSCRAPT.

ACM Reference Format:
Ipsa Mittra, Harihara Subrahmaniam Muralidharan, Tu Luan, and Mihai Pop. 2024. MeanSCRAPT: an iterative algorithm for clustering 16S rRNA long-read gene datasets. In . ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The invention of long-read sequencing technology can provide improved de novo assembly, mapping certainty, and reduced amplification bias[1]. Long-read sequencing of the 16S rRNA gene can shed more light on sequence-based bacterial analysis. Sequencing of the full gene has been proven to have the potential to provide taxonomic resolution of bacterial communities at species and strain levels [7]. This technology is also being applied to clinical settings where long-read sequencing has been used to gain greater resolution in identifying unique taxa for various human microbiomes [5]. Although long-read sequencing of the 16S rRNA gene is becoming more cost-efficient and implemented in experimental settings, there are very few bioinformatic tools to analyze this data, including MIrROR [13] and MetaSquare [10]. For 16S rRNA short-reads, sequence clustering tools such as SCRAPT [11], UCLUST [4], CD-HIT [9], and DADA2 [3] characterize diversity within microbial communities. Currently, there are no existing tools for 16S rRNA long-read sequences. Here, we describe the MeanSCRAPT software package, which uses an iterative sampling-based algorithm to cluster 16S rRNA long-read sequences.

The SCRAPT (Sample, Cluster, Recruit AdaPt, and iTerate) software package is a clustering algorithm that uses adaptive sampling to bias clustering towards the most abundant clusters in a dataset. The algorithm begins by taking a sub-sample of the entire dataset and performing an initial clustering on this subset. The representative sequences from each resulting non-singleton cluster are used as "bait" to add other sequences to the already identified clusters in the dataset. During each iteration of the clustering process, SCRAPT uses a mode-shifting technique to change the sequences representative of each cluster. The mode-shifting method ensures that the representative sequences accurately approximate the parent sequence of each cluster from where the corresponding members "evolve." This iterative clustering is shown to have much higher performance over one-shot clustering techniques for various sequence types [11].

To take advantage of this iterative clustering approach for long-read 16S rRNA sequences, we adapt the algorithmic structure of the SCRAPT algorithm. SCRAPT's efficiency derives from its methodology of discovering the largest clusters in the first iterations with the most resources. In the initial iterations, the algorithm prioritizes discovering large clusters, so the later iterations do not have high costs. Additionally, a high level of parallelization is implemented in each iteration of clustering through baiting, shifting, and rebaiting. We apply these characteristics to this new clustering algorithm

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

for long-read sequences. The original SCRAPT algorithm also implements a mode-shifting algorithm for 16S short-read sequences where the sequence with the highest multiplicity is selected as the representative sequence of a cluster. This 'mode' representative sequence is thought to demonstrate the oversampling of the true biological sequence within the cluster. However, this shifting strategy relies on sequences within a cluster having high variation of k-mer abundances. While this applies to short-read 16S rRNA sequences with high abundance variation, it cannot be applied to long-read 16S rRNA sequences as they do not have varied abundance and are generally of lower quality [7]. However, long-read 16S rRNA sequences do have varied k-mer profiles. As a result, a mean-shifting technique can be applied where representative sequences of a cluster are selected by choosing the sequence closest to the 'mean' of the sequences within the cluster.

2 MATERIALS AND METHODS

2.1 Algorithm Overview

The MeanSCRAPT algorithm broadly follows the structure of the SCRAPT algorithm (Algorithm 1). The first step of this algorithm is to remove all sequences within the data set of low quality. Next, MeanSCRAPT takes a sub-sample of the entire data set and performs an initial clustering on this subset. The representative sequences from each resulting non-singleton cluster are used as "bait" to add other sequences to the already identified clusters in the dataset. During each iteration of the clustering process, MeanSCRAPT uses a mean-shifting technique to change the sequences representative of each cluster, also known as centroids. This mean-shifting method examines the k-mer counts of each sequence and creates an artificial "average" sequence whose k-mer profile is the mean of the k-mer profiles of all sequences within the cluster. The centroid for this cluster is then chosen from the cluster sequences by determining the sequence with the k-mer profile most similar to the artificial "average sequence. As input, the MeanSCRAPT algorithm takes a set of deduplicated sequences, a percentage identity, iteration count, thread count, a threshold for k-mer quality, and a sampling rate. MeanSCRAPT uses SeqKit [14] to alternate between FASTQ and FASTA formats, performing filtering, deduplication, and subsampling.

2.2 Quality Score Filtering

While long-read sequencing provides more information for taxonomic classification of each sequence , it comes at the cost of higher precision reads. A long-read sequencing microbiome dataset may contain sequences of varying quality. Low-quality reads can affect the accuracy of clustering algorithms as they create unnecessary noise within the dataset. In MeanSCRAPT, this issue is solved by a quality score filtering of sequences. Before clustering, the entire data set has to be filtered to remove low-quality sequences. Each sequence is examined using the following process: for each k-mer, a quality score is calculated using equation 1. The quality score of the whole sequence is taken as the average of the sequence's k-mer quality scores. The sequence is only kept for clustering if its representative quality score is higher or equal to the input quality threshold given. As a result, any predominantly low-quality sequences are removed from the dataset to reduce noise in the following step.

Algorithm 1: SCRAPT for Long-Read Sequences

Input: S, De-duplicated 16S rRNA gene sequence data,

$\propto$, initial sampling rate, $0 \leq \propto \leq 100$

δ, adjustment factor, $0 \leq \delta \leq 100$,

M, sequence to count dictionary,

r, similarity threshold, $0 \leq r \leq 1$,

q, quality score threshold

Output: 16S rRNA gene sequence clusters and singletons

Initialization: Let d_i and σ_i be the number of sequences clustered and the sampling rate in the i^{th} iteration.

1. $\sigma_1 = \propto$, $d_1 = d_0 = 1$, $I = 1$, $S_1 = S$
2. D ← dictionary of kept sequences
3. For all sequences s in S
   a. For all k-mers k in s
   b. *removed* ← 0
      i. *score* ← 0
         1. For base pair b in k
            a. *score* += log (quality score(b))
      ii. If *score* > q
         1. Update D with k
      iii. Else
         1. *removed* += 1
   c. If *removed* >= 0.1 * k-mers in s
      i. Remove s from S
4. **Repeat**
   a. $S_i' \leftarrow Sample(S_i, \lceil |S_i| * \frac{\sigma i}{100} \rceil)$;
   b. (clusters $c_1, c_2, \ldots, c_n$, centroids $k_{c1}, k_{c2}, \ldots, k_{cn}$) ← Alignment Method ($S_i'$, r);
   c. $c_1', c_2', \ldots, c_n' \leftarrow Bait((k_{c1}, k_{c2}, \ldots, k_{cn}), S_i, r)$;
   d. $k_{c1}', k_{c2}', \ldots, k_{cn}' \leftarrow Mean\text{-}shifting\ ((c_1', c_2', \ldots, c_n' \leftarrow), M)$;
   e. $c_1'', c_2'', \ldots, c_n'' \leftarrow Bait((k_{c1}', k_{c2}', \ldots, k_{cn}'), S_i, r)$;
   f. **if** $d_i \leq d_{i-1}$ **then** $\sigma_i = \sigma_{i-1} + (\frac{di-1}{di} - 1) * \delta$;
   g. **else** $\sigma_i = \sigma_{i-1} - (1 - \frac{di-1}{di}) * \delta$;
   h. i = i + 1;
   i. $d_i = \sum_{k=1}^{n} |c_k^R|$;
   j. $S_i \leftarrow S_{i-1} \setminus (s \mid s \in c_i^R, \forall i \in 1..n\}$;
5. **until** *the number of iterations is met, or clusters above a certain size are discovered;*

2.3 Mean Shifting

During each iteration, cluster representatives are selected using the process of mean-shifting. A hash table was created to associate each possible k-mer with an integer. For each sequence, k-mer counts were stored in a respective dictionary. This dictionary has keys to the integer values of k-mers within the sequence and values of the counts of those k-mers. To determine an artificial "average" k-mer profile, a similar dictionary is created by calculating the average value for each key across all the sequences within the cluster. The set of possible k-mers is calculated, and for each k-mer k in the set, the average count is calculated by summing all k occurrences across clusters in sequence and dividing by the number of sequences within the cluster. The representative sequence for the cluster is then chosen by finding a real sequence within the cluster that has the k-mer profile closest to the artificial representative sequence. This is determined by finding the sequence whose Euclidean distance to the artificial sequence is shortest. This process is given in more detail in Algorithm 2.

Algorithm 2: Mean-Shifting

Input: *S*, De-duplicated 16S rRNA gene sequence data in cluster,

M, sequence to k-mer count dictionary, *k*, size of k-mer

Output: Representative centroid of cluster, *c*

Initialization: Let *d* be the number of sequences within the cluster.

1. *possible_kmers* ← 4^k
2. *sum_of_counts* ← array of 0s, size of *possible_kmers*
3. for sequence *s* in cluster
   a. *curr_kmer_dict* ← dictionary of k-mer counts of sequence *s* where key is k-mer and value is counts of k-mer
   b. for k-mer *k* in curr_kmer_dict
      i. *sum_of_counts*[*k*] ← *sum_of_counts*[*k*] + *curr_kmer_dict*[*k*]
4. *avg_kmer_counts* ← []
5. for k-mer *k* *sum_of_counts*
   a. *avg_kmer_counts*[*k*] ← *sum_of_counts*[*k*] / *d*
6. *smallest_distance* ← 10000
7. *closest_seq* ← ""
8. for sequence *s* in cluster
   a. *curr_kmer_dict* ← dictionary of k-mer counts of sequence *s* where key is k-mer and value is counts of k-mer
   b. *distance* ← 0
   c. for k-mer *k* in *avg_kmer_counts*
      i. *distance* ← *distance* + (*avg_kmer_counts*[*k*] - *curr_kmer_dict*[*k*])^2
   d. *distance* ← $\sqrt{(distance)}$
   e. if *smallest_distance* < *distance*
      i. *smallest_distance* ← *distance*
      ii. *closest_seq* ← *s*
9. return *closest_seq*

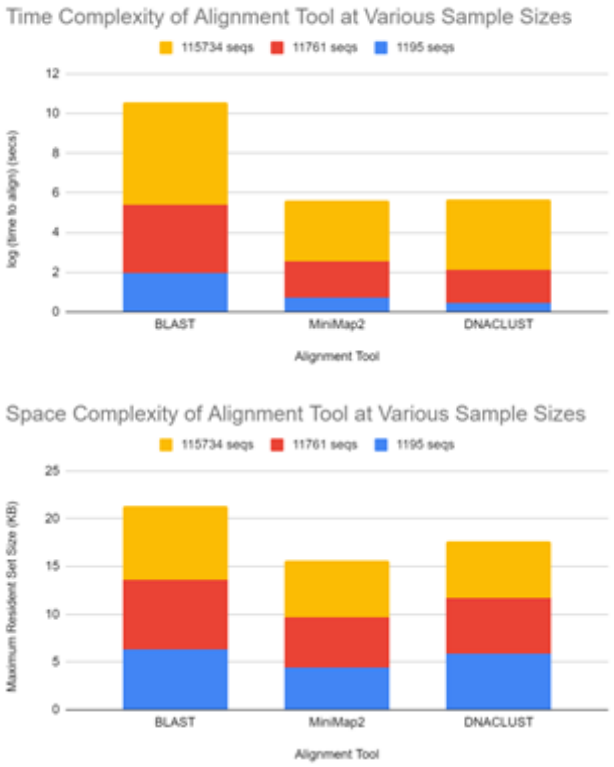

Figure 1: Samples of size 1195, 11761, and 115734 sequences were run on BLAST, MiniMap2, and DNACLUST, and a) time usage and b) memory usage was calculated.

2.4 Alignment within Clustering Iteration

In the original SCRAPT algorithm, a portion of unclustered sequences is sampled and clustered with DNACLUST [6]. The non-singleton clusters are analyzed using mean-shifting, discussed later, to determine representative sequences for each cluster. These representative sequences for each cluster of the subsample are then used to recruit other sequences from the original dataset to be added to the current clusters. As DNACLUST is used to cluster millions of short, highly similar DNA sequences, it was appropriate for clustering short-read 16S rRNA sequences. However, as the characteristics of long-read 16S rRNA sequences - longer length and higher variability - are significantly different from the ideal input for DNACLUST, we explored using MiniMap2 [8] and BLAST [2] as alignment tools that would aid in creating clusters for each iteration. To determine the best clustering kernel for MeanSCRAPT, each alignment tool was tested for memory and time usage efficiency and cluster fragmentation. Random samples of sequences were taken from a dataset of 16S rRNA human gut microbiome long-read sequences [12] at sampling rates of 0.1%, 1%, and 10% of the full dataset. Each subsample was used as input for BLAST, MiniMap2, and DNACLUST to create either clusters or pairwise alignments. DNACLUST was able to cluster these subsamples directly. As BLAST and MiniMap2 return pairwise alignments, these alignments were post-processed to create clusters and identify centroids. Additionally, for each alignment command, the time and memory usage were recorded (Figures 1a and 1b).

3 RESULTS

3.1 Human Gut Microbiome Dataset

A distribution plot was created to determine the sequence length characteristic of the original human gut microbiome 16S rRNA dataset (Figure 2a). The distribution of all sequence lengths is bimodal, with peaks at sequence lengths of approximately 500 and 1500 base pairs , approximately at the half and full counts of base pairs of the 16S rRNA gene. There are also sequences greater than 1600 base pairs, the length of the 16S rRNA gene, found within this dataset. From analyses of the individual files and the distribution plot, it was determined that the original dataset contained both long-read and short-read sequences. After filtering for sequences of over two thousand base pairs and calculating quality scores for all the sequences within this dataset, a distribution plot of the quality scores was created (Figure 2b). This is a normalized and cumulative distribution plot where we can see that approximately half of the sequences have a calculated score below 2.42, and ten percent of the sequences have quality scores under 2.35.

3.2 MeanSCRAPT performs better than other tools

MeanSCRAPT's performance was compared to other popular tools, all run using 32 threads with a maximum memory allocation of 256

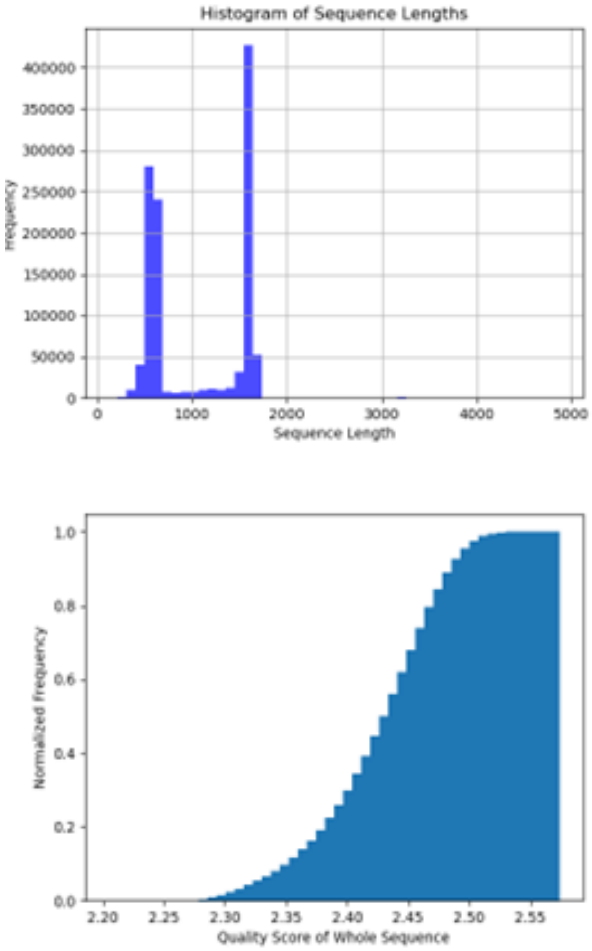

Figure 2: Samples of size 1195, 11761, and 115734 sequences were run on BLAST, MiniMap2, and DNACLUST, and a) time usage and b) memory usage was calculated.

GB on the same machine to find clusters with a similarity threshold of 80 percent. MeanSCRAPT experiments were run with default inputs of a sampling rate of 0.1 percent and 50 iterations.

Figure 3 compares clusters created by MeanSCRAPT, DNACLUST, MiniMap2, and BLAST using a fragmentation curve. Fragmentation is computed by ordering all clustering in decreasing size and measuring, for every size x, the total number of sequences contained in clusters of size x or greater. A fragmentation curve is created by plotting the fragmentation for all cluster sizes. On the fragmentation curve , the clustering method, whose curve is at the top, produces less fragmented clusters than all other tools. For this dataset, we see that the MeanSCRAPT clustering tool produces the least fragmented clusters at the topmost curve in the figure. In comparison, BLAST and MiniMap2 produce extremely fragmented clusters. MiniMap2 determines very small clusters, as seen by the clump of singletons in the top right corner of the graph. BLAST has

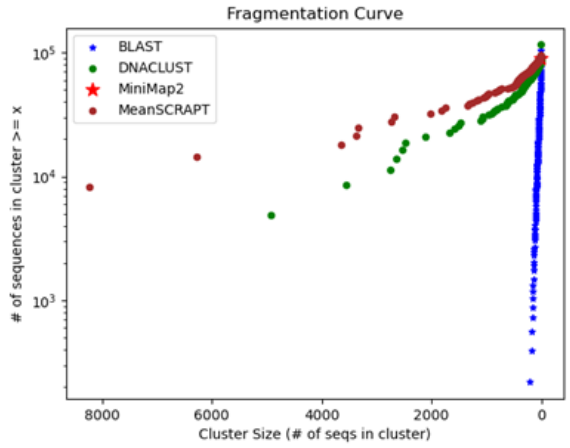

Figure 3: Fragmentation curves for the toy dataset of the human gut microbiome dataset for similarity of 0.80

only a slightly higher variation in cluster sizes, as most of the cluster sizes are still singletons. DNACLUST produces less fragmented clusters but still has more fragmentation than MeanSCRAPT.

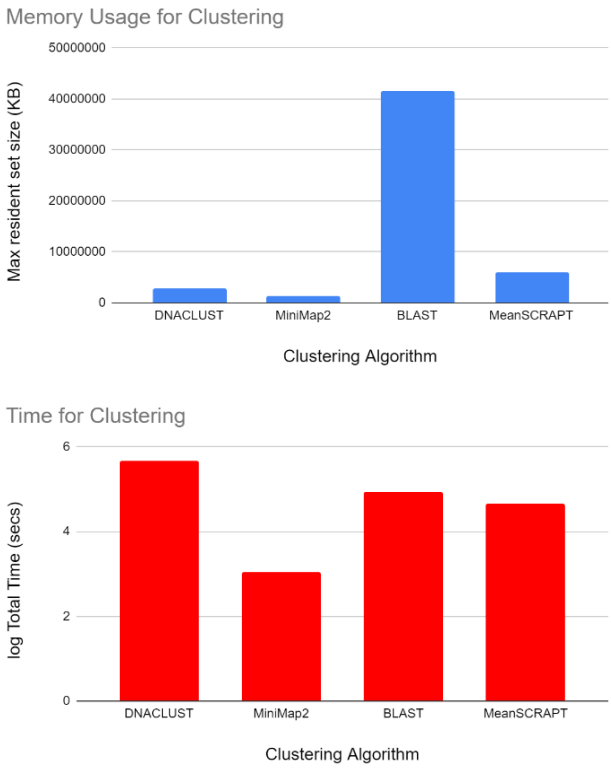

Figure 4: a) time usage and b) memory usage calculated for DNACLUST, MiniMap2, BLAST, and MeanSCRAPT.

MeanSCRAPT has similar or better memory usage than the other clustering tools, as seen in Figure 4a. DNACLUST, MiniMap2, and MeanSCRAPT all have a maximum resident set size of under 10 GB, while BLAST has a maximum resident size of over 40 GB. MeanSCRAPT also takes less time than most of the alternate clustering approaches. Figure 4b illustrates time usage for these algorithms, calculated as the sum of usage and system time. For BLAST and MiniMap2, this time is determined by the sum of creating the pairwise alignments and creating the clusters from these alignments. Although MiniMap2 takes the least time, MeanSCRAPT has the second lowest run time. This additional time is due to the preprocessing required by MeanSCRAPT, but this cost pays for the better clusters this algorithm produces.

4 DISCUSSION

In this paper, we describe the MeanSCRAPT tool, which uses an iterative clustering process that implements mean-shifting and is applied to 16S rRNA long-read sequences. We show that this approach has a much better performance compared to alternate one-shot clustering methods such as DNACLUST, MiniMap2, and BLAST.

The fragmentation comparison between these methods demonstrates that MeanSCRAPT has created the least fragmented clusters, likely due to its mean-shifting algorithm and iterative process. BLAST and MiniMap2 are not direct clustering algorithms. Instead, they create pairwise alignments from which clusters are created using sequence similarity and implementing greedy clustering. As a result, most of these clusters are singletons since very few sequences produced alignments where the sequence similarity was over the given threshold between unique sequences. In comparison to DNACLUST, MeanSCRAPT produces clusters of much greater size.

In addition to less fragmented clusters, MeanSCRAPT has relatively better time and memory usage than other approaches. MiniMap2 was the only tool that had less memory usage than MeanSCRAPT, but the overwhelming majority of clusters it found were singletons. MiniMap2 and DNACLUST had less time to run compared to MeanSCRAPT. However, these better performances in time and memory are most likely due to the preprocessing MeanSCRAPT requires. Before starting iterative clustering, MeanSCRAPT has to calculate k-mer counts and store using hashing for each retained sequence to implement mean-shifting during each iteration. Calculating and storing this information takes the bulk of time and memory for this tool. This cost pays for the better clustering performance of MeanSCRAPT.

MeanSCRAPT is only tested on a small dataset of approximately 100,000 sequences. In the future, we will continue testing this algorithm on the human gut microbiome dataset of over a million sequences. A similar study will be performed on this whole dataset to compare MeanSCRAPT to DNACLUST, BLAST, and MiniMap2. Additionally, we will test this tool at various similarity thresholds and sampling rates. The current study assesses these tools' performances at 80 percent similarity. Initially, these tools were attempted to be used at a 90 percent and over similarity threshold ; however, almost all clusters produced were singletons. This is likely due to the high variation between these long-read sequences. By testing at different similarity thresholds, especially at higher thresholds, we can assess how cluster, time, and memory performance are affected. Additionally, we should test how these metrics are affected using various sampling rates.

As MeanSCRAPT does not have the best memory usage within this study, we can improve this metric by converting this software package from Python to C++. This conversion will produce lower memory costs for calculating k-mer counts and the representative centroids of each cluster. We will also test this tool on other long-read 16S rRNA datasets.

REFERENCES

[1] S. L. Amarasinghe, S. Su, X. Dong, L. Zappia, M. E. Ritchie, and Q. Gouil. 2020. Opportunities and challenges in long-read sequencing data analysis. *Genome Biology* 21, 1 (2020). https://doi.org/10.1186/s13059-020-1935-5

[2] G. M. Boratyn, C. Camacho, P. S. Cooper, G. Coulouris, A. Fong, N. Ma, T. L. Madden, W. T. Matten, S. D. McGinnis, Y. Merezhuk, Y. Raytselis, E. W. Sayers, T. Tao, J. Ye, and I. Zaretskaya. 2013. BLAST: A more efficient report with usability improvements. *Nucleic Acids Research* 41, W1 (2013). https://doi.org/10.1093/nar/gkt282

[3] B. J. Callahan, P. J. McMurdie, M. J. Rosen, A. W. Han, A. J. Johnson, and S. P. Holmes. 2016. Dada2: High-resolution sample inference from Illumina Amplicon Data. *Nature Methods* 13, 7 (2016), 581–583. https://doi.org/10.1038/nmeth.3869

[4] R. C. Edgar. 2010. Search and clustering orders of magnitude faster than blast. *Bioinformatics* 26, 19 (2010), 2460–2461. https://doi.org/10.1093/bioinformatics/btq461

[5] J. L. Gehrig, D. M. Portik, M. D. Driscoll, E. Jackson, S. Chakraborty, D. Gratalo, M. Ashby, and R. Valladares. 2022. Finding the right fit: Evaluation of short-read and long-read sequencing approaches to maximize the utility of clinical microbiome data. *Microbial Genomics* 8, 3 (2022). https://doi.org/10.1099/mgen.0.000794

[6] M. Ghodsi, B. Liu, and M. Pop. 2011. DNACLUST: Accurate and efficient clustering of phylogenetic marker genes. *BMC Bioinformatics* 12, 1 (2011). https://doi.org/10.1186/1471-2105-12-271

[7] J. S. Johnson, D. J. Spakowicz, B.-Y. Hong, L. M. Petersen, P. Demkowicz, L. Chen, S. R. Leopold, B. M. Hanson, H. O. Agresta, M. Gerstein, E. Sodergren, and G. M. Weinstock. 2019. Evaluation of 16S rrna gene sequencing for species and strain-level microbiome analysis. *Nature Communications* 10, 1 (2019). https://doi.org/10.1038/s41467-019-13036-1

[8] H. Li. 2018. Minimap2: Pairwise alignment for nucleotide sequences. *Bioinformatics* 34, 18 (2018), 3094–3100. https://doi.org/10.1093/bioinformatics/bty191

[9] W. Li and A. Godzik. 2006. CD-hit: A fast program for clustering and comparing large sets of protein or nucleotide sequences. *Bioinformatics* 22, 13 (2006), 1658–1659. https://doi.org/10.1093/bioinformatics/btl158

[10] C.-C. Liao, P.-Y. Fu, C.-W. Huang, C.-H. Chuang, Y. Yen, C.-Y. Lin, and S.-H. Chen. 2022. MetaSquare: An integrated metadatabase of 16S rrna gene amplicon for microbiome taxonomic classification. *Bioinformatics* 38, 10 (2022), 2930–2931. https://doi.org/10.1093/bioinformatics/btac184

[11] T. Luan, H. S. Muralidharan, M. Alshehri, I. Mittra, and M. Pop. 2023. SCRAPT: an iterative algorithm for clustering large 16S rRNA gene data sets. *Nucleic Acids Research* 51, 8 (2023). https://doi.org/10.1093/nar/gkad158

[12] Y Matsuo, S Komiya, Y Yasumizu, and et al. 2021. Full-length 16S rRNA gene amplicon analysis of human gut microbiota using MinION™ nanopore sequencing confers species-level resolution. *BMC Microbiology* 21, 1 (2021), 35. https://doi.org/10.1186/s12866-021-02094-5

[13] D Seol, J S Lim, S Sung, Y H Lee, M Jeong, S Cho, W Kwak, and H Kim. 2022. Microbial identification using rrna operon region: Database and tool for Metataxonomics with long-read sequence. *Microbiology Spectrum* 10, 2 (2022). https://doi.org/10.1128/spectrum.02017-21

[14] W Shen, S Le, Y Li, and F Hu. 2016. SeqKit: A cross-platform and Ultrafast Toolkit for FASTA/Q file manipulation. *PLOS ONE* 11, 10 (2016). https://doi.org/10.1371/journal.pone.0163962