Metapath: Long read metagenome assembler based on random kmer sizes and overlap graphs

Juan Fernando Meza, Juan David Marmolejo & Nathalia Portilla

Biological Context

Metagenomics is defined as the direct genetic analysis of genomes within an environmental sample and has played a pivotal role in driving significant advancements in microbial ecology (Thomas, Gilbert, Meyer, 2012). Generally, metagenomics workflows include six principal steps: DNA extraction, construction of libraries (labeling sample genetic material), sequencing (obtention of genetic sequences at reads level), read quality analysis (filter for the integrity of reads), assembly (join of different reads until tentative genomes organism differentiation), and functional-taxonomy analysis (taxonomy assignation of genomes and comparative analysis). The workflow of tools in metagenomics can be altered by the choice of sequencing technology, with the emergence of long-read platforms challenging the longstanding dominance of short-read sequencing, which, for a considerable period, faced limitations such as GC bias from PCR amplification and constraints in sequencing long repeats and structural variants crucial for Metagenome-Assembled Genomes (MAGs) identification (Free, 2023).

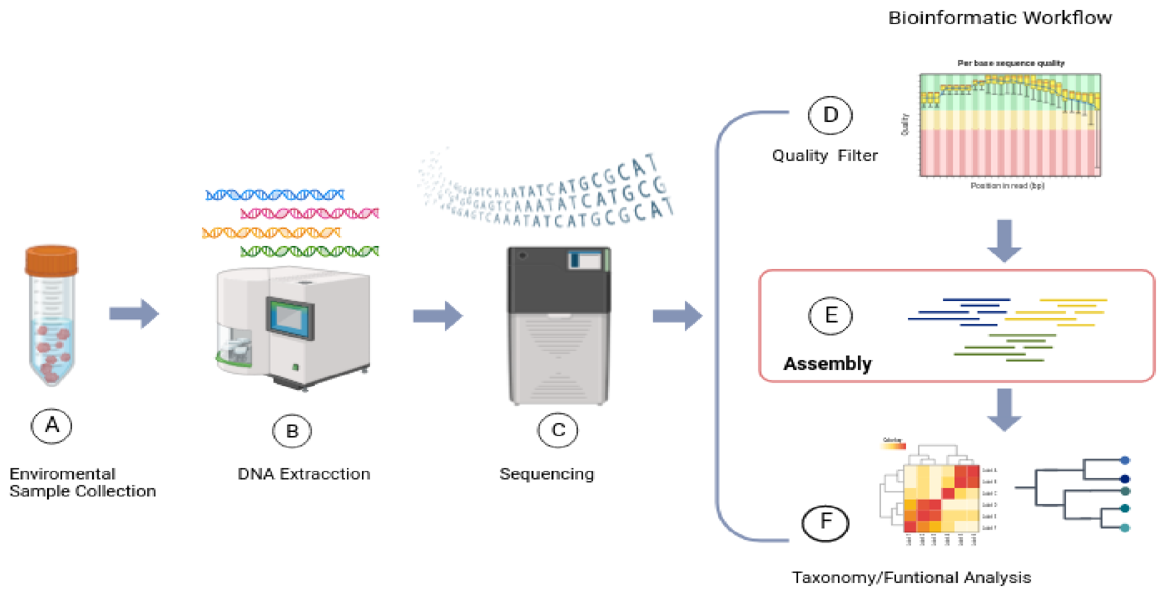

Figure 1. Metagenomics general flux

Motivation

Long-read sequencing platforms enhance genome assembly by improving contiguity and completeness, which is particularly a critical aspect in intricate communities characterized by high composition diversity and intra-species heterogeneity (Xie et al., 2020). Despite the acknowledged benefits, long-read technologies, exemplified by PacBio, exhibit lower throughput, accuracy, and higher costs compared to short-read sequencing. PacBio, with an error rate of 10-15%, faces challenges, especially concerning low-abundant organisms. To address this, PacBio employs circular consensus sequences (CCS), involving multiple passes over a circular DNA template, with a coverage of 15 passes estimating 99% accuracy (Xie et al., 2020).

In the broader context, long-read metagenomics has significantly elevated the capacity to recover high-quality Metagenome-Assembled Genomes (MAGs). Implementation of combined techniques, such as Hi-C and de novo DNA modification detection, holds promise for achieving strain-level differentiation, particularly in identifying extrachromosomal elements like plasmids and phages (Albertsen, 2023). The recent enhancements in high-molecular-weight DNA extraction protocols have facilitated the sequencing of complex metagenomes through extended read lengths and comprehensive coverage.

Although the use of long reads is gaining acceptance in metagenomic pipelines, the field still lacks dedicated long-read metagenomic assemblers designed to address specific challenges, including variable coverage in species composition, the presence of long intra-genomic/inter-genomic repeats, and inter/intra species heterogeneity.

Metagenomic assemblers due to the pairwise read nature of short-reads NGS (Next Generation Sequencing) platforms, break down reads into k-mers by resolving computational resources, some of the genomic contexts are lost and the selection of a specific k-mer size can alter assembly between runs, to resolve this implementation of iterative Bruijn graphs combined with k-mer sizes are used on the metagenome assembler but there is not an integrative range of k-mer between tools (New & Brito, 2020). On another hand, efforts on metagenomic assemblers like Metafly use a solid k-mer selection, through the calculation of the k-mer frequency of all reads, the selection by a threshold of these k-mers. To resolve the necessity for repeat identification, Metafly constructed a disjoint graph as a draft to represent arbitrary paths, based on these graphs a repeat graph is constructed collapsing each family of long repeats as a single path in which edges are classified as unique or repetitive, this classification leads contiguity as is the biggest problem for the presence of mosaic repeats (multiple edges), using read-paths reads traversers, all in the beginning are unique and determinate single successors and predecessors, if there are multiple the edge is classified as repeat, but variable coverage of metagenomic reads generate bias.

metaFlye algorithm

Metaflye is a computational method developed by Kolmogorov, Rayko, Polevikov & Pevzner (2020) for fast and long-read assembly of metagenomes using repetitive graphs. Compared to other long-read assembly tools such as FALCON, Canu, miniasm and wtdbg2; Metafly outperforms them in terms of contiguity (in terms of NG50 statistics) and runtimes. The Metafly tool also confirms that long-read assemblers significantly outperform short-read assemblers concerning complete gene, plasmid, and viral 16S RNA sequencing (Kolmogorov et al., 2020).

The need to implement a new algorithm (metaFlye) for the assembly of long sequences arises from Flye that can assemble these sequences (figure 2). This algorithm selects solid K-mers (high-frequency K-mers in the read set) to approximate the set of genomic K-mers. It then uses the solid K-mers to efficiently detect overlapping reads and combine them into "disjointings" that become different paths in the assembly graph (Kolmogorov et al., 2020). However, this approach would not be suitable for reads from a metagenome because it discriminates against low abundance species by having a smaller number of solid K-mers to assemble (Kolmogorov et al., 2019; Kolmogorov et al., 2020). This limitation in metagenomes is addressed by a new approach implemented in Metafly that combines global K-mer counting with analysis of local K-mer distributions to ensure a fair selection of solid K-mers to represent low abundance species (Kolmogorov et al., 2020). Therefore, we outline the MetaFlye workflow that addresses this limitation through two consecutive steps: identification of solid K-mers in the metagenome and identification of repeats in the assemblage graph.

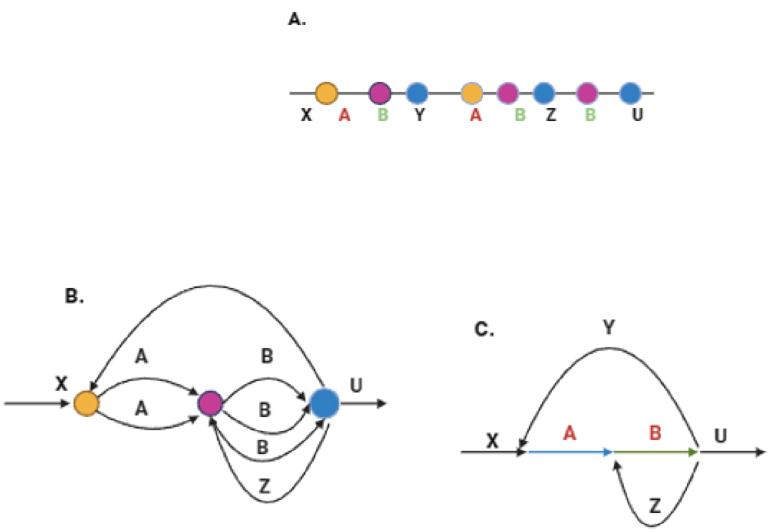

Figure 2. A repeating graph is constructed from local alignments within an XABYABZBU genome. The alignment paths for all local self-alignments are represented as a repeating graph. Obtained from (Kolmogorov et al., 2019).

Identification of solid K-mers in metagenomes: It starts with counting K-mers in all reads. Given an error rate per nucleotide ε in reads the tool estimates the probability that a K-mer in a read is error free. This approach is based on a Poisson error distribution model and takes into account the per-nucleotide error rate ε in reads.

$$E = e^{-k\varepsilon}$$

E: Probability that a K-mer in a reading is error free.
e: Base of natural logarithm (exponential probability that a K-mer is error-free).
k: Size of the K-mer
ε: Error rate per nucleotide in reads.

Therefore, the expected number of solid K-mers in a read is E * |read| and for each read a frequency threshold f is selected such that there are at least E * |read| K-mers in the read with a frequency at least f and indexes the K-mers above this threshold a hash table (Kolmogorov et al., 2020).

Repeat detection: metaFlye classifies each edge of the metagenome assembly graph as unique (its sequence appears only once in a single genome) or repetitive (the edge sequence appears several times in a single genome or is shared by several genomes) (Figure 3). For this the tool employs the read paths along a specific edge (Kolmogorov et al., 2020). The next unique edge along this path is defined as the successor of the current edge. A set of all read paths along an edge can establish one or several successors. In order to address chimeric reads, metaFlye filters out successors backed by less than MaxSucc / delta supporting reads, where MaxSucc represents the maximum number of supporting reads for a successor and delta is a threshold (default value of delta=5)(Kolmogorov et al., 2020). If an edge has multiple successors or predecessors, it is classified as repetitive. This process is repeated iteratively over the entire set of edges until no more edges are classified as repetitive.

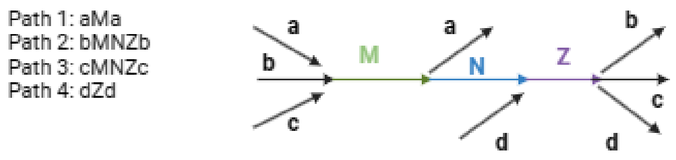

Figure 3. Mosaic repeat in graph adapted from Kolmogorov et al., 2020.

Metapath algorithm

Metapath leverages PacBio Hi-Fi long-read data, generating random k-mers within a specified range—typically assumed to be between 15 to 17, reported as the optimal size for PacBio. These k-mers serve as nodes in the construction of an overlap graph, where each edge is assigned the number of overlapping base pairs between corresponding nodes. Subsequently, a random pair of initial and final nodes is identified, characterized by exclusive outputs and inputs, respectively, with at least one path connecting them. Multiple paths are discovered for this node pair, and the algorithm selects the path maximizing total overlap. To prevent redundancy, the chosen paths are removed from the graph, and the process is iterated. The algorithm concludes when the paths reach 5% of the longest path, at which point the data is exported. This workflow is shown in Figure 3.

From our resource tests, it was determined that the optimal number of k-mers per read for computational resource optimization is approximately 25 k-mers. A higher number of k-mers could lead to longer waiting times during graph construction. The graph construction was performed using the NetworkX package (Hagberg, Schult and Swart, 2008) , which not only facilitates graph construction but also enables path searching. However, maximizing overlap with this library poses an NP-complete problem, and the volume of generated data could exponentially increase waiting times. Utilizing a specific algorithm for a Directed Acyclic Graph (DAG) is not feasible since we encountered graphs in our data that could be cyclic. For this reason, the decision was made to identify random paths, assuming they might represent an estimate of paths with an average total overlap length.

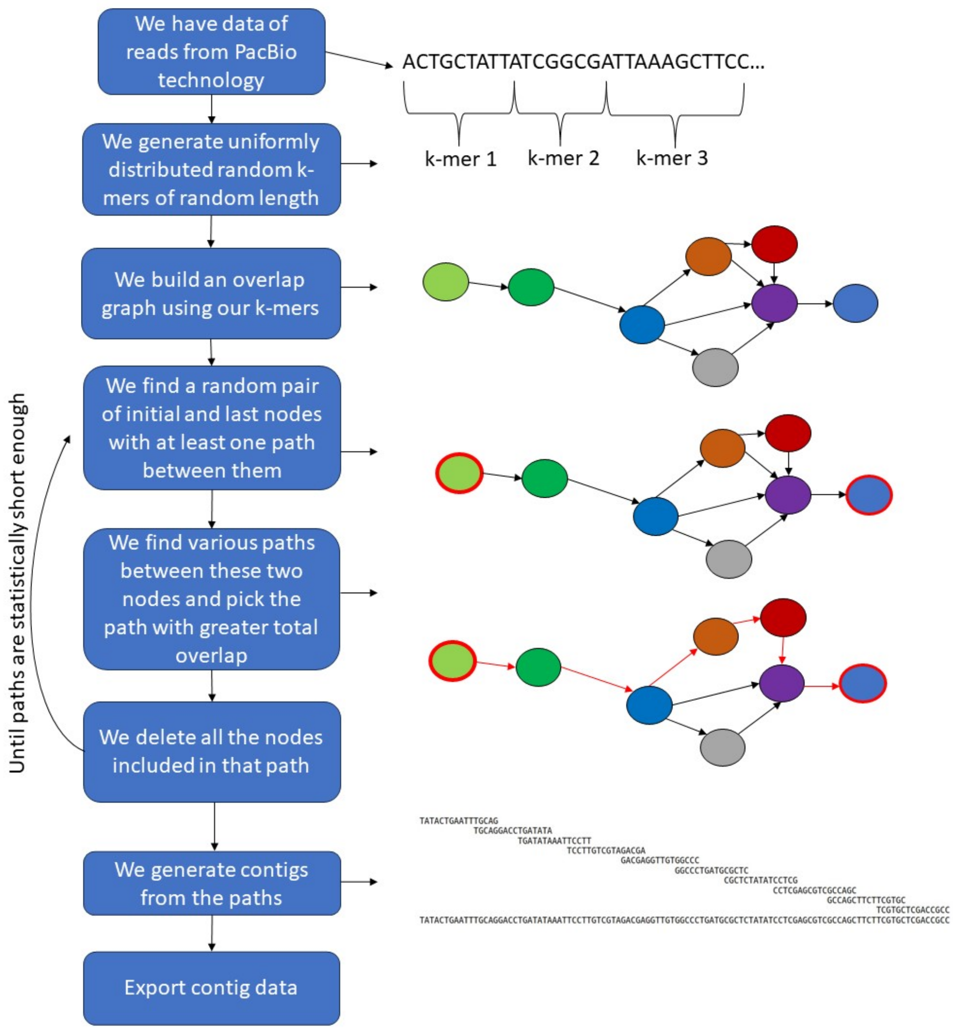
We have data of reads from PacBio technology
ACTGCTATTATCGGCGATTAAAGCTTCC...
k-mer 1
k-mer 2
k-mer 3
We generate uniformly distributed random k-mers of random length
We build an overlap graph using our k-mers
We find a random pair of initial and last nodes with at least one path between them
We find various paths between these two nodes and pick the path with greater total overlap
We delete all the nodes included in that path
Until paths are statistically short enough
We generate contigs from the paths
TATACTGAATTTGCAG
TGCAGGACCTGATATA
TGATATAAATTCCTT
TCCTTGTCGTAGACGA
GACGAGGTTGTGGCCC
GGCCCTGATGCGCTC
CGCTCTATATCCTCG
CCTCGAGCGTCGCCAGC
GCCAGCTTCTTCGTGC
TCGTGCTCGACCGCC
TATACTGAATTTGCAGGACCTGATATAAATTCCTTGTCGTAGACGAGGTTGTGGCCCTGATGCGCTCTATATCCTCGAGCGTCGCCAGCTTCTTCGTGCTCGACCGCC
Export contig data

Figure 4. Scheme for the algorithm's workflow.

Given the random sampling nature of our graph paths, variations in lengths are expected for each run. By running our algorithm multiple times and ranking the path lengths, we observe distributions as illustrated in Figure 5. In this Figure it is evident that, even when in average the path lengths have the expected tendency, but also it is possible that for certain runs of the algorithm the path lengths overlap.

Focusing solely on the analysis of averages, there is a possibility that we could discern between the sizes of genomes present in the input sample, where the variance in the distribution could serve as an error in the average data. In the data used for algorithm evaluation (see formalization of the computational problem), we employed a limited number of reads, so confirming this hypothesis would depend on conducting an assessment with a higher number of reads.

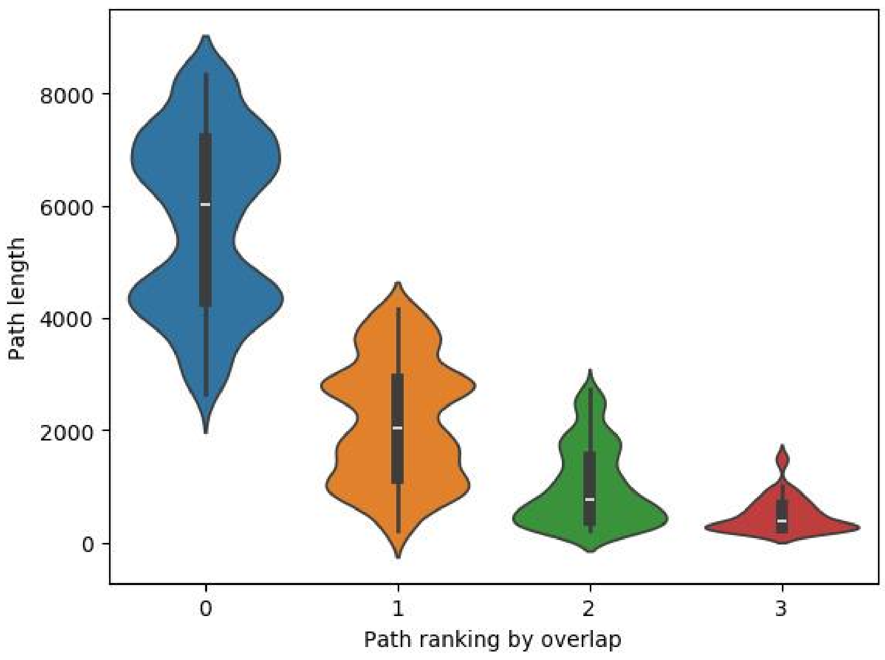

Figure 5. Distribution of path lengths: The longest paths are represented in blue, while the second, third, and fourth longest paths are depicted in orange, green, and red, respectively.

On the other hand, Figure 6 illustrates the distributions of k-mer overlaps for each identified path that will form a contig. The general trend indicates that the frequency appears to decrease as k-mer overlapping increases; however, the distributions, at least for some paths, seem to exhibit a bimodal behavior. There is a possibility that this behavior is associated with biological patterns, such as high repetitive regions; nonetheless, systematic comparisons are necessary to distinguish this behavior from patterns that may arise from the assembly process.

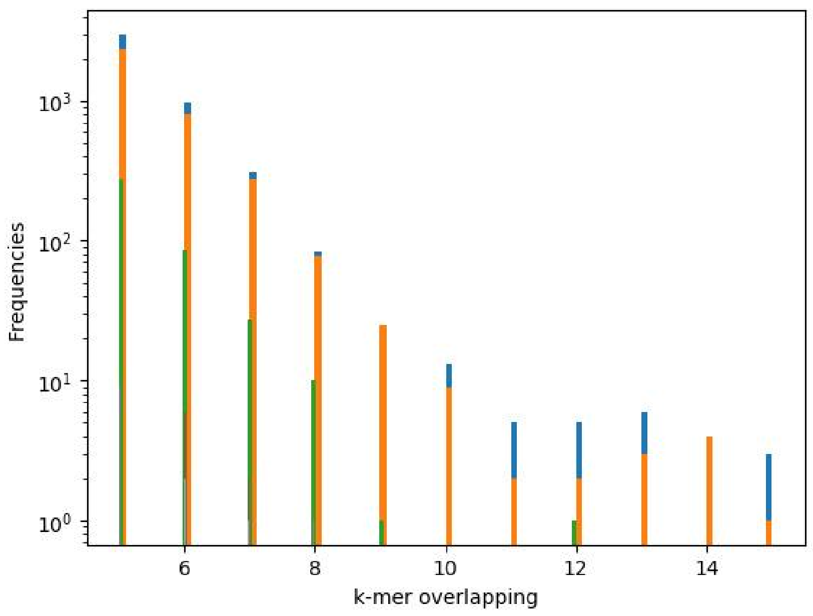

Figure 6. k-mer overlapping distributions for each found path. Each color represents a different path.

Computational Problem Formalization

The input consists of a sequence fastq file which corresponds to the total metagenome to all the reads to analyze in the sample as a requirement for the metaphat . Part of the setup raw data preparation for the algorithm performance was made by generating a mock community, we downloaded from NCBI the complete fasta genomes of three bacteria (NZ_CP017786.1 *Bacillus xiamenensis* with a genome size of 3.6 Mb, AP009049.1 *Clostridium kluyveri* with a genome size of 4 Mb*,* NC_000913.3 *Escherichia coli* with a genome size of 4.6 Mb*)* and using NGSEP software and class Simple Reads Simulator for generate 1000 reads with size of 25000 pb in order to generated a fastq that emulates a metagenome on a PacBio HiFi plataform, the result fastq then act as a entry in our algorthim. The metapath consists in the export of a txt per contig generated (sequence of tentative genome assemblies from the metagenome).

Benchmarking

Overall metafly assembler can construct 14 contigs while metapath generates a total of 7 contigs. For evaluate the completness of assembly we use BUSCO, in the case of Metafly we obtenined 30 complete BUSCOS, 29 complete and single-copy BUSCOS, 1 complere and duplicated BUSCOS, 1 fragmented BUSCOS, 93 Missing BUSCOS and 124 Total BUSCO groups searched. In another hand. In the case of Metaphat we obtained 0 complete BUSCOS, 0 comple and single-copy BUSCOS, 0 complete and duplicated buscos and 0 fragmented buscos, with 93 missing BUSCOS and 124 Total BUSCOS (Simao *et al*., 2015). These results suggest that the Metafly assembly exhibits a higher level of completeness, with a notable number of complete and single-copy BUSCOs. In contrast, the Metapath assembly shows no complete BUSCOs. Further investigation into the missing BUSCOs for Metapath is recommended to provide a comprehensive evaluation. Additionally, examining other metrics such as N50 and total assembly size can contribute to a more thorough assessment of the assemblies, but this is just possible with a complete set of reads, our test dataset just consists of 10.000 kmers from the mock community fastq simulator, with a bigger and better representation of reads can improve the assembly, this data is also seen on the figure 7 in which the majority of the BUSCOS for the metapath algorithm are categorized as missing and for metafly 50% constis of complete and single copy BUSCOS and the other 25% consists on complete and duplicated BUSCOS.

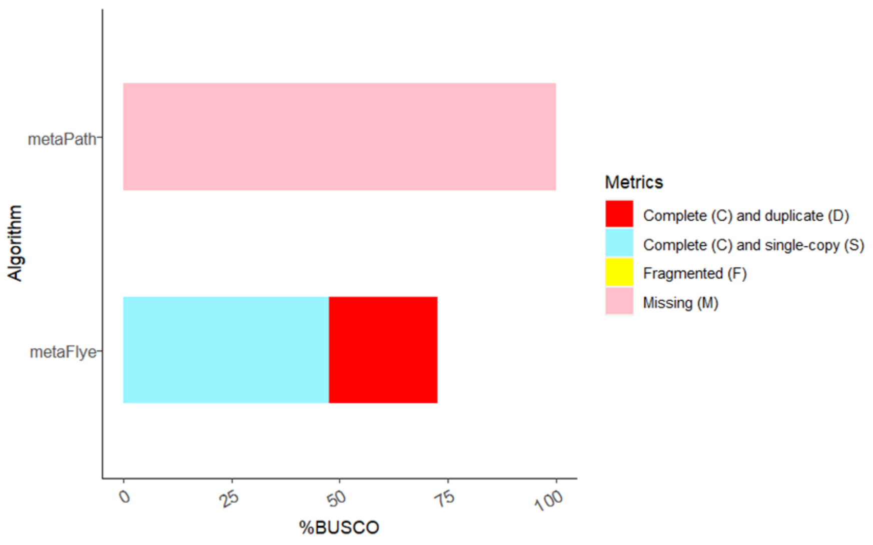

Figure 7. BUSCO results for Metafly and Metapath.

The obtained results are due to handling a limited number of K-mers, specifically 10,000 selected from those generated from metaPath, and a reduced number of reads. This is necessary for testing the algorithm's performance and obtaining future results with longer reads.

Github repository
https://github.com/portillanath/metaphat

References

Albertsen, M. Long-read metagenomics paves the way toward a complete microbial tree of life. *Nat Methods* 20, 30–31 (2023). https://doi.org/10.1038/s41592-022-01726-6

Free, T. (2023). Long-read sequencing for the metagenomic analysis of microbiomes.

Hagberg, A. A., Schult, D. A., & Swart, P. J. (2008). Exploring network structure, dynamics, and function using NetworkX. In G. Varoquaux, T. Vaught, & J. Millman (Eds.), Proceedings of the 7th Python in Science Conference (SciPy2008) (pp. 11–15). Pasadena, CA, USA.

Kolmogorov, M., Bickhart, D. M., Behsaz, B., Gurevich, A., Rayko, M., Shin, S. B., ... & Pevzner, P. A. (2020). metaFlye: scalable long-read metagenome assembly using repeat graphs. *Nature Methods*, *17*(11), 1103-1110.

Kolmogorov, M., Yuan, J., Lin, Y., & Pevzner, P. A. (2019). Assembly of long, error-prone reads using repeat graphs. *Nature biotechnology*, *37*(5), 540-546.

New, F. N., & Brito, I. L. (2020). What is metagenomics teaching us, and what is missed?. *Annual Review of Microbiology*, *74*, 117-135.

Simão, F. A., Waterhouse, R. M., Ioannidis, P., Kriventseva, E. V., & Zdobnov, E. M. (2015). BUSCO: assessing genome assembly and annotation completeness with single-copy orthologs. *Bioinformatics*, *31*(19), 3210-3212.

Thomas, T., Gilbert, J., & Meyer, F. (2012). Metagenomics - a guide from sampling to data analysis. In Microbial Informatics and Experimentation (Vol. 2, Issue 1). Springer Science and Business Media LLC. https://doi.org/10.1186/2042-5783-2-3

Xie, H., Yang, C., Sun, Y., Igarashi, Y., Jin, T., & Luo, F. (2020). PacBio long reads improve metagenomic assemblies, gene catalogs, and genome binning. *Frontiers in Genetics*, *11*, 516269.