NAR Genomics and Bioinformatics, 2025, **7**, lqaf158
https://doi.org/10.1093/nargab/lqaf158
Pipelines and Workflows for Biological Data Analysis

OXFORD

MeTAline: enabling reproducible and scalable metagenomic analyses

Olfat Khannous-Lleiffe[1,2], Diego Fuentes-Palacios[1,2], Dániel Májer[1,2], Toni Gabaldón [1,2,3,4,*]

[1]Barcelona Supercomputing Centre (BSC-CNS), Plaça Eusebi Güell, 1–3, Barcelona 08034, Spain
[2]Institute for Research in Biomedicine (IRB Barcelona), The Barcelona Institute of Science and Technology, Baldiri Reixac, 10, Barcelona 08028, Spain
[3]Catalan Institution for Research and Advanced Studies (ICREA), Pg. Lluís Companys 23, Barcelona 08010, Spain
[4]Centro de Investigación Biomédica En Red de Enfermedades Infecciosas (CIBERINFEC), Instituto de Salud Carlos III, Madrid 28029, Spain
*To whom correspondence should be addressed. Email: toni.gabaldon@bsc.es

Abstract

The taxonomic and functional characterization of microbial communities inhabiting a given niche can elucidate associations between the microbiota and relevant variables, including health and disease. As compared to metabarcoding, shotgun metagenomic sequencing, which analyzes all DNA present in a sample, offers superior taxonomic resolution and additionally enables the inference of functional capabilities encoded within the microbial community of interest. However, this approach requires the use of diverse computational tools and substantial computational resources. Here, we present MeTAline, a bioinformatics pipeline for the analysis of shotgun metagenomics data. Implemented in Snakemake, MeTAline provides an efficient and reproducible workflow encompassing read trimming and filtering, host read removal, taxonomic classification via both *k*-mer and gene marker-based methodologies, and extensive functional annotation. Containerization in Docker and Singularity ensures ease of installation, portability, and reproducibility. Finally, the pipeline's architecture supports high parallelization, rendering it suitable for both local and high-performance computing environments. MeTAline is freely available at https://github.com/Gabaldonlab/meTAline under an open-source GNU GPL v3.0 license.

Introduction

The advent of next-generation sequencing technologies has revolutionized the study of microbial communities across diverse niches by enabling accurate, culture-independent profiling [1]. In particular, shotgun metagenomic sequencing (metagenomics) provides high-resolution taxonomic identification and relative abundance of microbes within a sample. Additionally, it enables the inference of functions, reactions, and metabolic pathways encoded by the identified sequences, revealing the potential roles of the microbes and the interactions of these microbes within their community and with their surroundings. This dual taxonomic and functional profiling has become a critical analytical step for understanding microbial communities from metagenomics data.

A typical metagenomics analysis for shotgun metagenomic sequencing data includes multiple steps, which use independent tools, including read trimming, quality filtering, host read subtraction, taxonomic classification, and functional annotation. In large-scale studies, these analyses can present important computational challenges [2], which must be addressed by automation through integration of the tools within computational pipelines that enable parallelization.

The rapid growth of metagenomic data has underscored the need for pipelines that are not only scalable and efficient but also reproducible. While significant progress has been made in developing such tools, there is no universal standard, and the user is confronted with multiple choices for both taxonomic or functional profiling. Among the most widely adopted tools for taxonomic profiling are Kraken2 [3] and MetaPhlAn4 [4], each representing two fundamentally distinct assembly-independent approaches: *k*-mer-based or marker-based, respectively, each of which may be best suited to different questions or datasets [5, 6]. A practical metagenomic pipeline should therefore integrate these two commonly used tools to accommodate a variety of analytical needs and preferences.

While assembly-based approaches are valuable for genome reconstruction, assembly-free methods, such as the aforementioned Kraken2 and MetaPhlAn4, provide distinct advantages in metagenomic research. They offer greater efficiency, improved detection of rare taxa even with low coverage [7, 8], and enhanced quantification of microbial diversity, making them an essential tool in studying complex microbial communities.

Regarding functional profiling, most existing tools require nucleotide-to-protein translation prior to database searches [9, 10], which is a time-consuming process. An exception to this is HUMAnN [11], which incorporates nucleotide sequence searches. Additionally, this tool can use previously computed MetaPhlAn4 taxonomic assignments to stratify community functional profiles, indicating which species contribute to each pathway.

Received: May 1, 2025. **Revised:** September 23, 2025. **Accepted:** September 30, 2025

Some web-based metagenomics platforms are available, such as MGnify [12], MG-Rast [13], and Galaxy [14]. These offer graphical user interfaces and multiple tools to analyze metagenomics data, but require data upload and rely on shared external cloud/cluster-based resources, which often results in data size limitations or long job execution times.

In response to these challenges and to provide a flexible tool for reproducible analysis, we present MeTAline v1.2, a modular pipeline for metagenomic analysis. MeTAline is implemented in the Snakemake workflow management system, seamlessly integrating software to carry out the entire analytical process, from initial read trimming and filtering, through host read subtraction, to taxonomic classification and functional profiling. The modular structure of MeTAline enables researchers to design different analytical workflows including either Kraken2 (*k*-mer-based classification) or MetaPhlAn4 (clade-specific maker-based). This ensures compatibility with widely accepted pipelines and facilitates taylor-made designs to accommodate diverse research purposes.

MeTAline is available via Docker and Singularity containers, thereby ensuring reproducibility and smooth portability. Finally, the parallelizable architecture of MeTAline makes it suitable for both local and high-performance computing (HPC) environments, allowing proper scalability to meet the demands of large datasets.

Materials and methods

Design

MeTAline v1.2. (Fig. 1) has been implemented in Snakemake (v. 9.6.0) [15] and consists of five main rules: *trimming*, *host_depletion*, *kmer_taxonomy*, *RAnalysis*, and *BioBakery*. The *trimming* rule is essential to ensure high-quality data and is based on trimmomatic (v. 0.39) [16], which removes low-quality bases and adapter sequences from the sequencing reads. This rule also includes a quality assessment of the reads by fastqc (v. 0.12.1) [17] and an additional step (concat_reads) that merges the trimmed reads of the same sample from different lanes into a single file for further processing. This step is necessary for samples sequenced across multiple lanes, a common practice used to avoid lane-specific biases.The *host_depletion* rule is automatically activated when a reference genome is provided as an input and removes host reads by aligning the trimmed reads to the reference using hisat2 (v.2.2.1) [18]. This step outputs a fastq file containing only the unmapped reads, which are subsequently used for the taxonomy assignment. If no reference genome is indicated, all trimmed reads are used for the taxonomy assignment. The *kmer_taxonomy* rule runs Kraken2 (v. 2.1.3) to generate a report with the *k*-mer-based classification using a specific database. Additionally, this rule includes subrules that allow the user to visualize taxonomic data in Krona (v. 2.8.1) and extract reads of interest such as unclassified reads (the default option). The *RAnalysis* rule converts Kraken2 reports to Biom format, which is then further transformed into the phyloseq (v. 1.34) [19] format to ensure compatibility with standard microbiome analysis tools. This step also provides subroutines that run R scripts for calculating alpha diversity metrics and visualizing the relative abundances of the top 25 taxa at the phylum level. The *BioBakery* rule conducts both taxonomic and functional profiling using gene marker-based tools, specifically MetaPhlAn4 (v. 4.1.1) for taxonomic profiling and HUMAnN (v. 3.9) [11] for functional profiling.

Implementation details

MeTAline uses snakemake under the hood using a .json configuration file, which the user has to generate first with a command indicating different arguments (Table 1) such as the extension of the fastq file, the base directory where output folders will be created, the source of the databases, among other parameters that are necessary to execute the pipeline rules. To parallelize meTAline for use in an HPC infrastructure, another python-based command is provided, metaline-prepare-greasy-array-job, to ease the setup; a tutorial of how to use it is described (https://github.com/Gabaldonlab/meTAline/wiki/Parallelize-meTAline-in-HPC-environment) in our github repository.

A single "input-output" command line interface design ensures simplicity, modularity, and composability, enabling snakemake native features such as the use of profiles as well as external and complementary tools such as xargs, GNU Parallel, or Greasy, to efficiently handle parallel execution without adding complexity to the core logic or restricting flexibility. While Snakemake provides built-in features for parallelization, we also acknowledge the power of complementary Unix tools such as GNU, Parallel, and Greasy. These tools often offer fine control, efficient resource management, and advanced options such as load balancing, making them especially useful for high-throughput or I/O-heavy tasks. In contrast, Snakemake's strength lies in portability, reproducibility, and seamless integration with clusters and cloud environments, capabilities that go beyond what traditional command line tools are designed to provide.

As shown in Table 1 and in Fig. 1, different databases are required by some of the rules: Kraken2 db by *kmer_taxonomy*, MetaPhlAn4 database, and nucleotide, and protein databases by *Biobakery*, for both nucleotide and translation searches, respectively. These databases can be downloaded from the corresponding original repository of each tool. Kraken2 also offers the option to customize the database to be used, with detailed instructions available in its manual: https://github.com/DerrickWood/kraken2/wiki/Manual, which we highly recommend. Our Github contains a selection of downloadable test databases, a test sample input (*test_input* directory containing raw data, both forward and reverse reads), and the corresponding expected output directories and files to facilitate the trial of MeTAline (see Quick Start and Try-Out).

The pipeline is available in the containerized environments Docker and Singularity, which simplifies installation and ensures consistent execution across different computing environments. These containers encapsulate all required software dependencies, eliminating the need for complex manual setups and making the pipeline readily accessible to users. A prebuilt singularity image is provided for immediate use, but detailed instructions to build containers from scratch are also available in MeTAline's GitHub repository. While the current release relies on a single, easy-to-use container that packages snakemake within it, we note that the workflow is compatible with a modular, per-rule containerization strategy, which has been considered for future improvements. Including Snakemake in the container ensures consistency with the tested workflow, simplifies setup, and helps avoid discrepancies across environments.

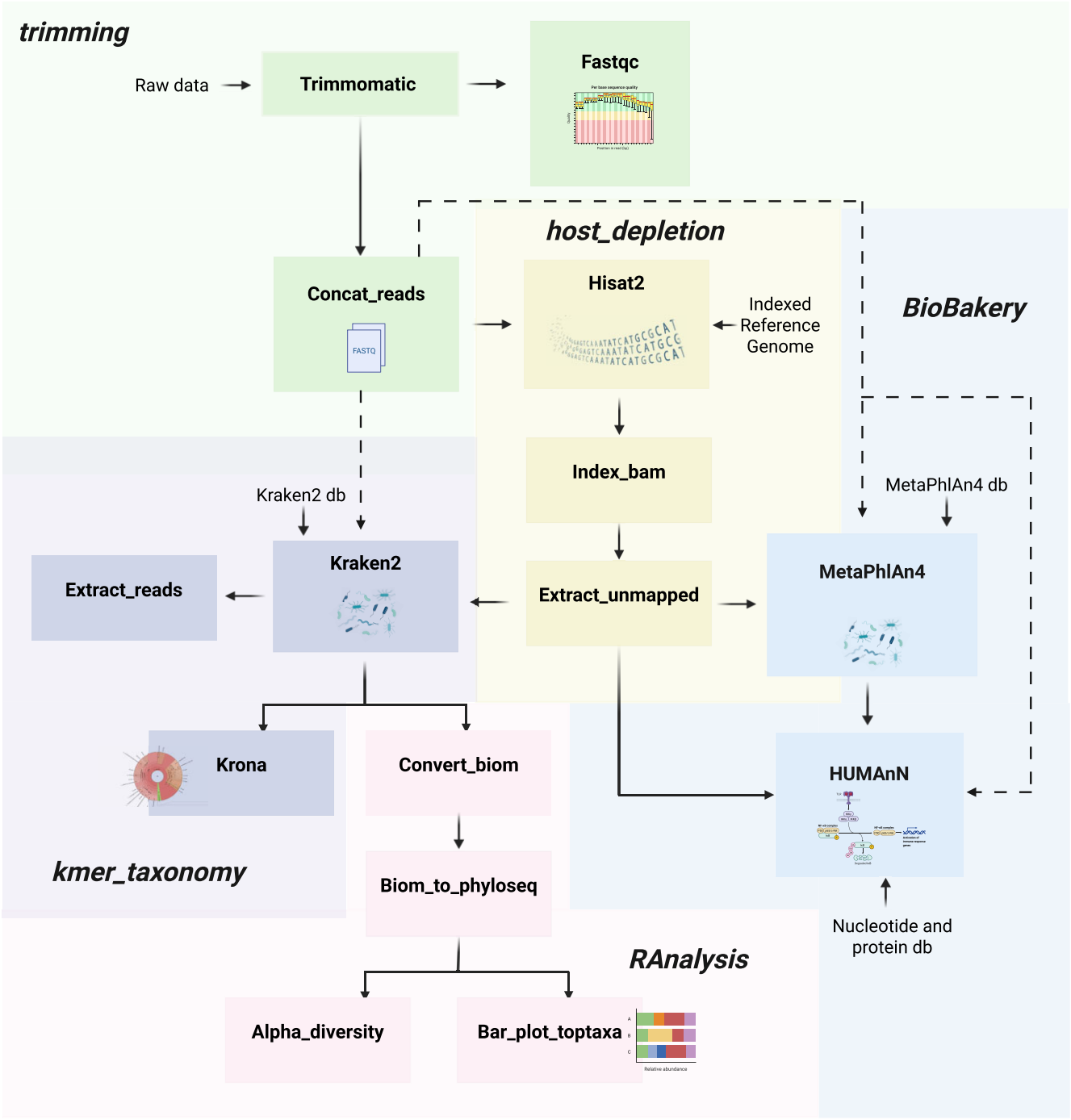

Figure 1. Schematic view of the different rules included in the meTAline pipeline, each shaded color represents one of the rules: *trimming* (green), *host_depletion* (yellow), *kmer_taxonomy* (purple), *RAnalysis* (pink), and *BioBakery* (blue), comprising different steps indicated in boxes. Dash arrows indicate an alternative workflow in the absence of a reference genome. If a reference genome is not provided, the concatenated trimmed reads will be used as input for the next rules. Alternatively, if a reference genome is provided, these rules will be using the unmapped reads to the host genome as input. Created in BioRender: Gabaldón, T. (2025) https://BioRender.com/r1mmfgp.

Results and discussion

Feature-based comparison of meTAline with existing pipelines

A comparison of MeTAline features with those of other popular available tools is provided in Table 2. Taxprofiler [20] emerges as the most similar tool. Taxprofiler is a nextflow pipeline for assembly-free taxonomy profiling, offering multiple profiling tools and supporting key functionalities such as read filtering, quality assessment, host depletion, taxonomy assignment, and reproducibility.

As compared to MeTAline, however, taxprofiler has some shortcomings. One important limitation is that taxprofiler does not include functional profiling, which is a fundamental step for a complete metagenomics analysis. Although the inclusion of up to 10 alternative profiling tools may be considered an advantage as it provides flexibility, it may be also

Table 1. Arguments and input parameters provided to create the config file

Argument	Details	Default
–config-file	Configuration JSON file to be generated, defining parameters, settings, etc.	config.json
–version	Pipeline version	1
–extension	Extension of the raw reads files	fastq.gz
–sample-barcode	Identification of the sample in the output files	None
–basedir	Base directory in which the pipeline will create output	None
–reads-directory	Directory where the fastqs reads are stored	None
–fastq-prefix	Wildcard parameter corresponding to the basename of the fastq files	None
–prefix-fr	Specify the naming convention on raw reads used to distinguish forward and reverse reads.	_
–reference-genome	Indexed reference genome path (e.g reference/Human_index)	None
–krakendb	Kraken2 database	None
–taxid	Selects the taxid used for extracting reads of interest after kraken2 assignment. You can add more than one using space as the separator	0 (unclassified reads)
–metaphlan-db	Path where the metaphlan4 database is located	None
–metaphlan-index	Index of the metaphlan4 database	None
–protein-db	Humann database to do the translation search (by default this is by-passed)	None
–n-db	Humann database to do the nucleotide search (based on already built annotations	None

For more details of programme parameters (such as for Trimmomatic and fastqc), revise carefully the metaline-generate-config command.

Table 2. Comparison of MeTAline with different available metagenomics pipelines

	MeTAline	Taxprofiler	SqueezeMeta	MetaWRAP	MG-Rast
Input	Raw reads	Raw reads	Assembly	Raw reads	Raw reads
Assembly-based analysis	No	No	Yes	Yes	No
Read filtering/quality assessment	Yes	Yes	Yes	Yes	Yes
Host depletion	Yes	Yes	No	Yes	
Taxonomy assignment	Yes	Yes	Yes	Yes	Yes
Functional profiling	Yes	No	Yes	Yes	Yes
Reproducibility	SingularityDocker	SingularityDocker	Conda	CondaDocker	Web-based
Local installation	Yes	Yes	Yes	Yes	No
HPC suitability	Yes	Yes	No	Yes	No
Visualization support	Yes	Yes	Yes	Yes	Yes

problematic for less-experienced users that might be confused. Importantly, some of the included tools may produce suboptimal results according to recent analysis, for instance mOTUs [21] is less sensitive than MetaPhlAn4, which is the recommended marker gene-based tool, and Bracken has been shown to result in increased rates of false positives [6, 21]. Finally, taxprofiler's MetaPhlan4 does not include the viral profiling option.

MeTAline-based analysis of shotgun metagenomics samples

To illustrate the use of MeTAline, we ran it on data from a previously published study (PRJEB10878) [22] consisting of shotgun metagenomics data from fecal samples of 74 individuals with colorectal cancer individuals and 54 healthy controls.

We executed meTAline for these 128 samples in a highly parallelized mode using greasy arrays managed by Slurm (see our GitHub tutorial titled "Parallelize meTAline in HPC environment" for setup instructions). In this use case, we launched an array of 32 greasy jobs, each composed of 4 tasks, corresponding to 4 individual samples. Each task was allocated 24 CPUs and executed on high-memory nodes due to the requirements of our Kraken2 database. The kraken2 database used for this run was a customized kraken2 database of ∼135 GB, which is a niche-specific database for the gut microbiome that builds on an in-house curated collection of 2110 fungal reference genomes, the human telomere-to-telomere (T2T) reference genome, and previously published curated databases for gut-specific bacterial and archeal genomes [23].

A key advantage of our approach, as mentioned before, is the use of a single input–output pipeline that not only simplifies workflow management but also significantly improves parallelization efficiency. As a result, the average runtime per greasy job (processing four samples) was 37 min and 1 s, with the longest execution time recorded at 44 min and 14 s.

For each individual sample, MeTAline generates eight output directories, each containing specific data files, as summarized in Table 3. The first directory, TRIMMOMATIC, contains the .fastq files produced after the trimming and filtering steps. These include the paired trimmed reads, unpaired trimmed reads, and, when applicable, a concatenated file. If concatenation is not performed, this file will be identical to the paired trimmed reads and is used in subsequent steps. This directory also includes a quality assessment report, generated by Fastqc.

The second directory, BAM, remains empty unless the user enables the host depletion module by providing a reference genome. When activated, this folder contains the .bam alignment file, its corresponding index file (.bai), and a compressed

Table 3. Example of output folders resulting in one run of the MeTAline pipeline

TRIMMOMATIC	ERR1018192.1.fastq.gz ERR1018192_1_paired.fq.gz ERR1018192_1_unpaired.fq.gz ERR1018192.2.fastq.gz ERR1018192_2_paired.fq.gz ERR1018192_2_unpaired.fq.gz ERR1018192_qc ERR1018192.1_fastqc.html ERR1018192.1_fastqc.zip ERR1018192.2_fastqc.html ERR1018192.2_fastqc.zip
BAM	ERR1018192.hisat2.bam ERR1018192.hisat2.bam.bai ERR1018192.unmapped.fastq.gz
KRAKEN_ASSIGN	ERR1018192.Kraken2.biom ERR1018192.kraken2.report ERR1018192.kraken2.txt
KRONA_HTML	ERR1018192.html ERR1018192.krona
EXTRACTED_FASTA	ERR10181921.fastq ERR1018192.1.fastq.gz ERR10181922.fastq ERR1018192.2.fastq.gz
R_ANALYSIS	ERR1018192_alpha_div.csv ERR1018192_Barplot_phyla.jpeg ERR1018192.rds
METAPHLAN4	ERR1018192 ERR1018192.unmapped_genefamilies.tsv ERR1018192.unmapped_humann_temp ERR1018192.unmapped_bowtie2_aligned.sam ERR1018192.unmapped_bowtie2_aligned.tsv ERR1018192.unmapped_bowtie2_index.1.bt2 ERR1018192.unmapped_bowtie2_index.2.bt2 ERR1018192.unmapped_bowtie2_index.3.bt2 ERR1018192.unmapped_bowtie2_index.4.bt2 ERR1018192.unmapped_bowtie2_index.rev.1.bt2 ERR1018192.unmapped_bowtie2_index.rev.2.bt2 ERR1018192.unmapped_bowtie2_unaligned.fa ERR1018192.unmapped_custom_chocophlan_database.ffn ERR1018192.unmapped.log ERR1018192.unmapped_pathabundance.tsv ERR1018192.unmapped_pathcoverage.tsv ERR1018192.bz2 ERR1018192_profiled.txt ERR1018192sam.bz2 ERR1018192.vsc.txt
Benchmark	ERR1018192.alignment.benchmark.txt ERR1018192.alpha_div.benchmark.txt ERR1018192.barplot.benchmark.txt ERR1018192.concat.benchmark.txt ERR1018192.conversion_bio.benchmark.txt ERR1018192.conversion_bio_to_phyloseq.benchmark.txt ERR1018192.ERR1018192.trimming.benchmark.txt ERR1018192.extraction_unmapped_reads.benchmark.txt ERR1018192.extract_reads.benchmark.txt ERR1018192.fastqc.benchmark.txt ERR1018192.humann.benchmark.txt ERR1018192.kraken2.benchmark.txt ERR1018192.krona.benchmark.txt ERR1018192.metaphlan4.benchmark.txt plots cpu_time_s.svg io_read_mb.svg io_write_mb.svg max_memory_rss_mb.svg max_proportional_memory_pss_mb.svg max_unique_memory_uss_mb.svg
	max_virtual_memory_vms_mb.svg mean_cpu_load.svg report.html runtime_s.svg

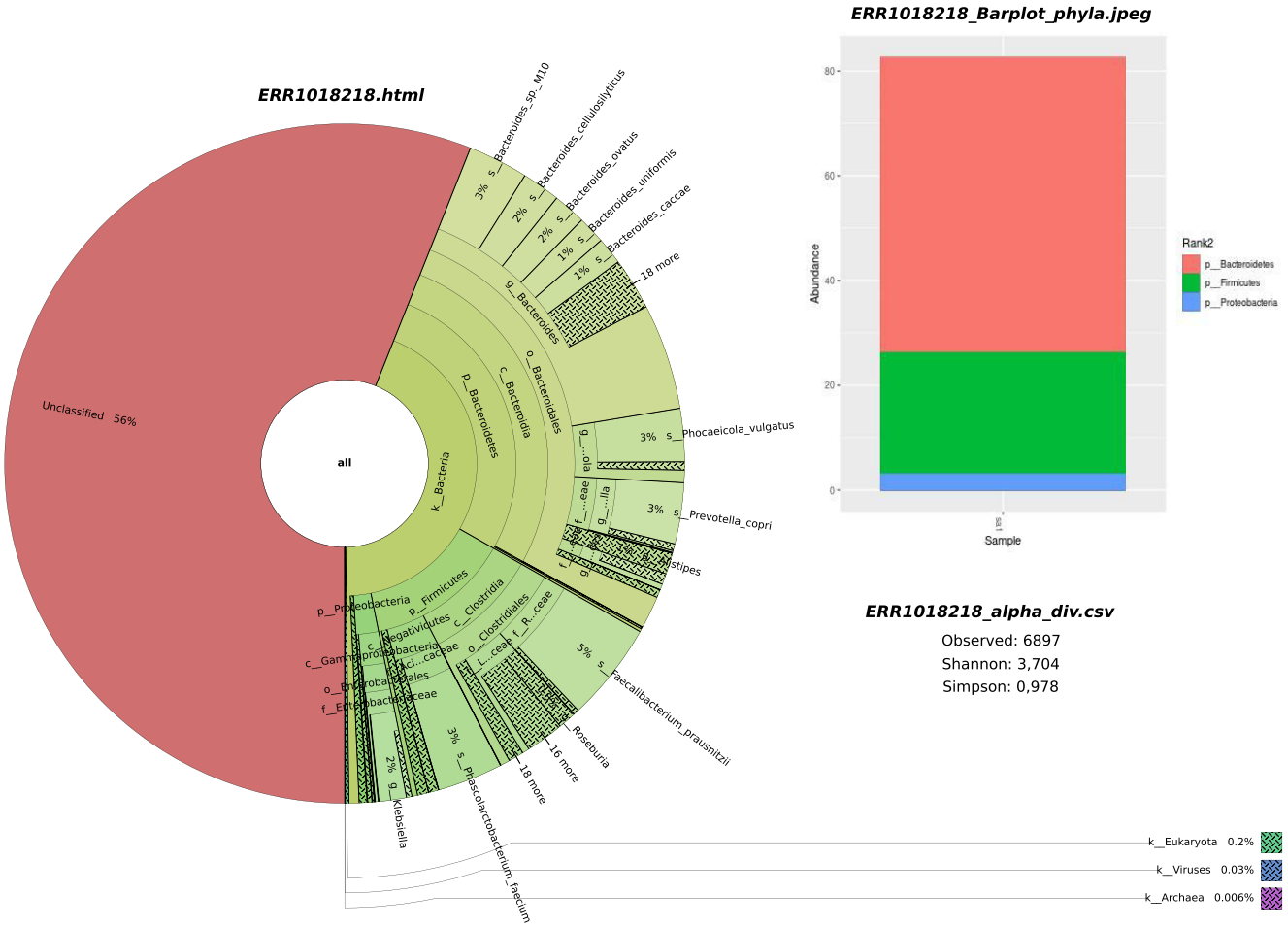

Figure 2. Visualization of some of the output files obtained with the meTAline pipeline for the kraken2 assignment. At the left, an .html file containing a krona representation. At the right, a bar plot representing the phyla of the top most represented taxa, and below that alpha diversity metrics that are reported in a csv file: Observed index, Shannon, and Simpson indices, without any filtering.

fastq file with the unmapped reads (those that did not align to the host reference).

The KRAKEN2 directory includes three output files: a summary report of the taxonomic classification, the detailed Kraken2 assignment file, and a biom file. The biom file can be directly converted into a phyloseq object for downstream analysis. One example of such downstream processing is located in the R_ANALYSIS directory, which contains an .rds file with the phyloseq object, a .csv file with alpha diversity metrics (Observed, Shannon, and Simpson indices), and a barplot depicting the top 25 most abundant phyla.

Additionally, the KRONA_HTML directory provides a Krona-based interactive visualization of the full Kraken2 taxonomic assignment. This folder includes both the final html output and the intermediate Krona input file. Another output folder, EXTRACTED_FASTA, contains fastq files of reads extracted by specific taxonomic identifiers. By default, this extraction targets unclassified reads, although it can be customized to allow further exploration of taxa of interest.

The *BioBakery* module generates the METAPHLAN4 directory, which contains microbial and viral profiling outputs. This includes a general taxonomic profiling file and a second file specific to viral assignment (with the .vsc.txt extension). This directory also includes intermediate alignment files, which can be useful for further analyses such as strain-level profiling. Within the same directory, a subfolder dedicated to human functional profiling contains three tab-delimited text files: genefamilies.tsv (gene family abundances), pathabundance.tsv (pathway abundances), and pathcoverage.tsv (presence or absence of specific pathways). A temporary folder stores additional intermediate files that detail the full functional profiling workflow.

A benchmark directory includes a summary of the resources used per rule, including metrics such as the running time and CPU usage over time, which are also plotted in a subfolder called plots and that the user can include in a report html, after the completion of the run. An example of command to create the report is found in our github repository https://github.com/Gabaldonlab/meTAline/wiki/Reporting-meTAline.

Illustrative examples of some of the outputs from the kraken2 profiling from one sample from the use-case study is shown in Fig. 2.

Finally, we compared meTAline with the most similar available tool, Taxprofiler (Table 2), using the full use-case dataset (total of 128 samples). As previously described, meTAline was able to efficiently execute the analysis in parallel into an array of greasy jobs, each processing four samples in parallel. In contrast, when running taxprofiler in batch mode, the total execution time was 12 h, 12 min, and 59 s. Of the total subtasks, 772 completed successfully, while 125 failed, primarily during

the Kraken2 assignment step (because of a memory issue). Of note, in this case we also used the kraken2 custom database of ∼135 GB. These failures occurred despite allocating 24 CPUs per task and using high-memory nodes (same characteristics of the meTAline job), suggesting taxprofiler may be less stable under memory-intensive conditions, at least when using the standard batch mode options. The html report of this run is shown as Supplementary Data S1. Using small (minikraken2 database, 8 GB) and middle size (Humgut database, 28 GB) [24] databases, the completion of taxprofiler on the analysis of the 128 samples was successful.

We also compared the two pipelines on a single-sample run. While taxprofiler was ∼30% faster than MeTAline (23 min versus 32 min), it consumed significantly more computational resources, both in terms of CPU usage and memory demand, consistent with observations from the full dataset, described above (see Supplementary Fig. S1). The job files used in both cases are shown as Supplementary material, Text S1 (MeTAline) and Text S2 (Taxprofiler) as well as the html reports as Supplementary Data S2 (meTAline) and Data S3 (taxprofiler).

As mentioned, all the comparisons were performed by using a large customized Kraken2 database (B-GUT, 135 Gb) which requires a high computational resources. When running the pipeline for the same single sample, but using the minikraken2 database provided in the test datasets (8 GB) the comparison shows similar CPU and memory consumption (see Supplementary Fig. S2). The html reports for both pipeline runs are provided as Supplementary Data S4 (meTAline) and S5 (taxprofiler). Similar behavior was also observed running the pipeline with a middle size database (28 GB), shown in Supplementary Fig. S3).

These results suggest practical implications for large-scale studies: although taxprofiler may offer slightly faster runtimes for individual samples, meTAline provides a more scalable, resource-efficient, and better solution for high-throughput metagenomic workflows in HPC environments, especially when working with large datasets and large databases.

Conclusions

MeTAline presents a portable and reproducible pipeline for comprehensive metagenomic analysis. This streamlined workflow encompasses quality control, host DNA removal, taxonomic classification, and functional profiling of short-read sequencing data. Furthermore, MeTAline integrates essential features such as basic data visualization and key metric calculations, establishing it as a powerful tool for metagenomic studies.

Acknowledgements

We acknowledge Martina Cardinali for her assistance in running the nextflow taxprofiler pipeline for benchmark proposes, and also Anastasia Sukhorukova and Hrant Hovhannisyan for testing and giving us feedback on the pipeline. Figure 1 was created with BioRender.com.

Author contributions: Olfat Khannous-Lleiffe (Conceptualization [equal], Formal analysis [equal], Investigation [equal], Methodology [lead], Software [lead], Visualization [lead], Writing – original draft [lead]), Diego Fuentes-Palacios (Software [supporting], Writing – review & editing [supporting]), Dániel Májer (Software [supporting]), and Toni Gabaldón (Conceptualization [equal], Funding acquisition [lead], Investigation [equal], Project administration [equal], Supervision [lead], Writing – original draft [equal], Writing – review & editing [lead])

Supplementary data

Supplementary data is available at NAR Genomics & Bioinformatics online.

Conflict of interest

None declared.

Funding

TG group acknowledges support from the Spanish Ministry of Science and Innovation (grant numbers PID2021-126067NB-I00, CPP2021-008552, PCI2022-135066-2, PLEC2023-010225, and PDC2022-133266-I00), cofounded by ERDF "A way of making Europe," as well as support from the Catalan Research Agency (AGAUR) (grant number SGR01551); Gordon and Betty Moore Foundation (grant number GBMF9742); "La Caixa" Foundation (grant number LCF/PR/HR21/00737), Fundació La Marató de TV3 (202328-31), AECC (PRYGN234923GABA), and Instituto de Salud Carlos III (IMPACT grant IMP/00019 and CIBERINFEC CB21/13/00061-ISCIII-SGEFI/ERDF). O.K.L. is supported by the Formación de profesorado universitario (FPU) program from the Spanish Ministerio de Universidades (FPU2020-02907).

Data availability

Pipeline code, usage documentation, parameter descriptions, and example outputs are available on Zenodo (https://doi.org/10.5281/zenodo.15683600) and GitHub (https://github.com/Gabaldonlab/meTAline).

Pre-built Singularity and Docker images are available in GitHub (https://github.com/gabaldonlab/metaline/releases/latest), DockerHub (https://hub.docker.com/repository/docker/cgenomics/metaline), and Sylabs (https://cloud.sylabs.io/library/cgenomics/cgenomics/metaline/metaline).

References

1. Simon C, Daniel R. Metagenomic analyses: past and future trends. *Appl Environ Microbiol* 2011;77:1153–61. https://doi.org/10.1128/AEM.02345-10
2. Bharti R, Grimm DG. Current challenges and best-practice protocols for microbiome analysis. *Brief Bioinform* 2021;22:178–93. https://doi.org/10.1093/bib/bbz155
3. Wood DE, Lu J, Langmead B. Improved metagenomic analysis with Kraken 2. *Genome Biol* 2019;20:257. https://doi.org/10.1186/s13059-019-1891-0
4. Blanco-Míguez A, Beghini F, Cumbo F *et al*. Extending and improving metagenomic taxonomic profiling with uncharacterized species using MetaPhlAn 4. *Nat Biotechnol* 2023;41:1633–44. https://doi.org/10.1038/s41587-023-01688-w
5. Wright RJ, Comeau AM, Langille MGI. From defaults to databases: parameter and database choice dramatically impact the performance of metagenomic taxonomic classification tools. *Microb Genom* 2023;9:000949.
6. Pusadkar V, Azad RK. Benchmarking metagenomic classifiers on simulated ancient and modern metagenomic data. *Microorganisms* 2023;11:2478. https://doi.org/10.3390/microorganisms11102478

7. Sarmashghi S, Bohmann K, P Gilbert MT *et al.* Skmer: assembly-free and alignment-free sample identification using genome skims. *Genome Biol* 2019;20:34. https://doi.org/10.1186/s13059-019-1632-4
8. Quince C, Walker AW, Simpson JT *et al.* Shotgun metagenomics, from sampling to analysis. *Nat Biotechnol* 2017;35:833–44. https://doi.org/10.1038/nbt.3935
9. Bağcı C, Patz S, Huson DH. DIAMOND+MEGAN: fast and easy taxonomic and functional analysis of short and long microbiome sequences. *Curr Protoc* 2021;1:e59.
10. Silva GGZ, Green KT, Dutilh BE *et al.* SUPER-FOCUS: a tool for agile functional analysis of shotgun metagenomic data. *Bioinformatics* 2016;32:354–61. https://doi.org/10.1093/bioinformatics/btv584
11. Beghini F, McIver LJ, Blanco-Míguez A *et al.* Integrating taxonomic, functional, and strain-level profiling of diverse microbial communities with bioBakery 3. *eLife* 2021;10:e65088. https://doi.org/10.7554/eLife.65088
12. Richardson L, Allen B, Baldi G *et al.* MGnify: the microbiome sequence data analysis resource in 2023. *Nucleic Acids Res* 2023;51:D753–9. https://doi.org/10.1093/nar/gkac1080
13. Keegan KP, Glass EM, Meyer F. MG-RAST, a metagenomics service for analysis of microbial community structure and function. *Methods Mol Biol* 2016;1399:207–33.
14. Community G. The Galaxy platform for accessible, reproducible, and collaborative data analyses: 2024 update. *Nucleic Acids Res* 2024;52:W83–94. https://doi.org/10.1093/nar/gkae410
15. Mölder F, Jablonski KP, Letcher B *et al.* Sustainable data analysis with Snakemake. *F1000Res* 2021;10:33.
16. Bolger AM, Lohse M, Usadel B. Trimmomatic: a flexible trimmer for Illumina sequence data. *Bioinformatics* 2014;30:2114–20. https://doi.org/10.1093/bioinformatics/btu170
17. Babraham Bioinformatics. FastQC—a quality control tool for high throughput sequence data. https://www.bioinformatics.babraham.ac.uk/projects/fastqc/ (2 October 2025, date last accessed).
18. Kim D, Paggi JM, Park C *et al.* Graph-based genome alignment and genotyping with HISAT2 and HISAT-genotype. *Nat Biotechnol* 2019;37:907–15. https://doi.org/10.1038/s41587-019-0201-4
19. McMurdie PJ, Holmes S. phyloseq: an R package for reproducible interactive analysis and graphics of microbiome census data. *PLoS One* 2013;8:e61217. https://doi.org/10.1371/journal.pone.0061217
20. Yates F, J.A. S, S. A-L *et al.* nf-core/taxprofiler: v1.2.3—Bouncy Basenji Patch. v1.2.4, Zenodo, https://zenodo.org/records/16985546, 28 August 2025.
21. Irankhah L, Khorsand B, Naghibzadeh M *et al.* Analyzing the performance of short-read classification tools on metagenomic samples toward proper diagnosis of diseases. *J Bioinform Comput Biol* 2024;22:2450012. https://doi.org/10.1142/S0219720024500124
22. Yu J, Feng Q, Wong SH *et al.* Metagenomic analysis of faecal microbiome as a tool towards targeted non-invasive biomarkers for colorectal cancer. *Gut* 2017;66:70–8. https://doi.org/10.1136/gutjnl-2015-309800
23. Khannous-Lleiffe O, Gabaldón T. B-GUT reference genome database improves biomarker discovery and fungal identification in gut metagenomes. Research Square, https://doi.org/10.21203/rs.3.rs-6766778/v1, 26 June 2025, preprint: not peer reviewed.
24. Hiseni P, Rudi K, Wilson RC *et al.* . HumGut: a comprehensive human gut prokaryotic genomes collection filtered by metagenome data. *Microbiome* 2021;9:165. https://doi.org/10.1186/s40168-021-01114-w

Received: May 1, 2025. **Revised:** September 23, 2025. **Accepted:** September 30, 2025