

CAMP: a modular metagenomics analysis system for integrated multistep data exploration

Lauren Mak^{1,2,*}, Braden Tierney^{2,*†}, Wei Wei^{2,†}, Cynthia Ronkowski³, Rodolfo Brizola Toscan⁴, Berk Turhan¹, Michael Toomey¹, Juan Sebastian Andrade-Martínez¹, Chenlian Fu¹, Alexander G Lucaci², Arthur Henrique Barrios Solano⁵, João Carlos Setubal⁵, James R Henriksen^{6,7}, Sam Zimmerman⁸, Malika Kopbayeva⁹, Anna Noyvert¹⁰, Zana Iwan¹¹, Shraman Kar¹¹, Nikita Nakazawa¹¹, Dmitry Meleshko^{1,2}, Dmytro Horyslavets^{12,13}, Valeriia Kantsypa^{12,14}, Alina Frolova^{12,13}, Andre Kahles¹⁵, David Danko^{16,17}, Eran Elhaik¹⁸, Pawel Labaj⁴, Serghei Mangul^{3,19,20,21}, The International MetaSUB Consortium[‡], Christopher E. Mason^{2,16,22,23,*§}, Iman Hajirasouliha^{10,24,*§}

¹Tri-Institutional Computational Biology & Medicine Program, Weill Cornell Medicine of Cornell University, 10065 NY, United States

²Department of Systems and Computational Biomedicine, Weill Cornell Medicine of Cornell University, 10065 NY, United States

³Department of Clinical Pharmacy, USC Alfred E. Mann School of Pharmacy and Pharmaceutical Sciences, University of Southern California, 90033 CA, United States

⁴Małopolska Centre of Biotechnology, Jagiellonian University, 30-387 Kraków, Poland

⁵Department of Biochemistry, Institute of Chemistry, Universidade de São Paulo, 05508-900 São Paulo, Brazil

⁶Natural Resource Ecology Laboratory, Colorado State University, 80521 CO, United States

⁷Two Frontiers Project, 80521 CO, United States

⁸Broad Institute of MIT and Harvard, 02142 MA, United States

⁹Mathematics Institute, University of Warwick, CV4 7AL Coventry, UK

¹⁰Nazarbayev Intellectual School of Physics and Math, 050000 Almaty, Kazakhstan

¹¹School of Molecular and Theoretical Biology, 51010 Tartu, Estonia

¹²Institute of Molecular Biology and Genetics, NASU, 03680 Kyiv, Ukraine

¹³Kyiv Academic University, 03142 Kyiv, Ukraine

¹⁴Taras Shevchenko National University, 01033 Kyiv, Ukraine

¹⁵ETH Zurich, 8092 Zurich, Switzerland

¹⁶Biotia, 11101 NY, United States

¹⁷GeoSeeq Foundation, 11101 NY, United States

¹⁸Department of Biology, Lund University, 223 62 Sweden

¹⁹Present address: Sage Bionetworks, WA 98121, United States

²⁰Department of Biological and Morphofunctional Sciences, College of Medicine and Biological Sciences, University of Suceava, 720229 Suceava, Romania

²¹Present address: Department of Computers, Informatics and Microelectronics, Technical University of Moldova, 2045 Chisinau, Moldova

²²WorldQuant Initiative for Quantitative Prediction, Weill Cornell Medicine of Cornell University, 10065 NY, United States

²³The Feil Family Brain and Mind Research Institute, Weill Cornell Medicine of Cornell University, 10065 NY, United States

²⁴Englander Institute for Precision Medicine, Weill Cornell Medicine of Cornell University, 10065 NY, United States

*To whom correspondence should be addressed. Email: laurenmak93@gmail.com

Correspondence may also be addressed to Braden Tierney. Email: btt4001@med.cornell.edu

Correspondence may also be addressed to Christopher E. Mason. Email: chm2042@med.cornell.edu

Correspondence may also be addressed to Iman Hajirasouliha. Email: imh2003@med.cornell.edu

†These authors share second authorship.

‡For the complete list of MetaSUB International Consortium members, please see the Supplementary Materials.

§The last two authors should be regarded as Joint Last Authors.

Abstract

Computational analysis of large-scale metagenomics sequencing datasets provides valuable isolate-level taxonomic and functional insights from complex microbial communities. However, the ever-expanding ecosystem of metagenomics-specific methods and file formats makes designing scalable workflows and seamlessly exploring output data increasingly challenging. Although one-click bioinformatics pipelines can help organize

Received: June 18, 2025. Revised: October 9, 2025. Accepted: October 22, 2025

© The Author(s) 2026. Published by Oxford University Press.

This is an Open Access article distributed under the terms of the Creative Commons Attribution-NonCommercial License

(<https://creativecommons.org/licenses/by-nc/4.0/>), which permits non-commercial re-use, distribution, and reproduction in any medium, provided the original work is properly cited. For commercial re-use, please contact reprints@oup.com for reprints and translation rights for reprints. All other

permissions can be obtained through our RightsLink service via the Permissions link on the article page on our site—for further information please contact journals.permissions@oup.com.

these tools into workflows, they face compatibility and maintainability challenges that can prevent replication. To address the gap in easily extensible yet robustly distributable metagenomics workflows, we have developed the Core Analysis Modular Pipeline (CAMP), a module-based metagenomics analysis system written in Snakemake, with a standardized module and directory architecture. Each module can run independently or in sequence to produce target data formats (e.g. short-read preprocessing alone or followed by *de novo* assembly), and provides output summary statistics reports and Jupyter notebook-based visualizations. We applied CAMP to a set of 10 metagenomics samples, demonstrating how a modular analysis system with built-in data visualization facilitates rich seamless communication between outputs from different analytical purposes. The CAMP ecosystem (module template and analysis modules) can be found at <https://github.com/Meta-CAMP>.

Introduction

Metagenomics—the study of microbial communities recovered directly from environmental samples using genomic methods—has emerged over the last two decades to become an important field in bioinformatics. This culture-independent method of investigating microbial communities in their natural contexts has revealed invaluable insights that have greatly enhanced our understanding of the natural world [1–5]. Despite a vast body of literature, a rapidly expanding toolkit, and several established protocols, metagenomics still has a long way to go due to the complex nature of the microbial world and the innate challenges of analysis workflows combined. Since a microbial community typically consists of a complex mixtures of dozens to thousands of species spanning both known and unexplored branches of the tree of life [6, 7], precise and reliable workflows are required to identify these species and decipher their functional roles. In a world today where global-scale sequencing efforts such as Human Microbiome Project [8, 9] and International Metagenomics and Metadesign of Subways and Urban Biomes (MetaSUB) [2] are underway, we have the opportunity to work with extremely rich, deep, and high-dimensional sequencing datasets. There is a field-wide need to develop workflow systems that are straightforward to test, maintain, reuse to generate reproducible results [10].

A typical metagenomic workflow consists of multiple *in silico* steps following sample collection and sequencing to resolve the taxonomic and/or functional composition of the originating community [11]. Over the past decades, a myriad of tools have been developed and are continually developed and updated with the goal of supporting the various components of these analytic goals. Despite the presence of a vast array of open-source tools available, many of them are not accessible online, easy to install, or even installable at all [12]. A widespread problem in bioinformatics is the short life span of tools. Kern *et al.* reported that of the 2396 web tools they surveyed in a 133-day test frame, only 31% of tools always worked, 48.4% worked occasionally, and 20.6% never worked [13].

This issue of fragility of tools cascaded when these tools are organized into workflows, resulting in many dependent points of potential failure at both installation and run times. Although open-source package management systems such as Conda, containerization systems such as Docker and Singularity, and workflow management systems (WMSs) such as Nextflow and Snakemake, have partially addressed these challenges, dependency conflicts and operating incompatibilities are still major problems when multiple tools need to co-exist in the same environment [14, 15]. Besides, workflows that solely rely on containerization for environment management are not widely usable on high-performance compute (HPC) clusters with root-access restrictions.

To streamline the complex process of decision-making, especially for users without extensive computational backgrounds, all-in-one pipelines such as ATLAS [16],

FinnLab/MAG_Snakemake_wf [17], MUFFIN [18], nf-core/mag [19], and nbis-meta [20] have gained popularity with the promise of ease of use, reproducibility, and standardized protocols. These conveniences, however, often come at a cost of flexibility and transparency. Commonly developed and maintained by a single laboratory, these all-in-one workflows tend to have limited end-user customization potential. Easy-to-install workflows with well-chosen default settings can be suitable for researchers familiar with basic Unix commands, but can be analytic black boxes in nature, taking end-users from input to desired output with a one-line command. In this process, the intermediate results are implicitly obscured, preventing the end-user from integrating their domain expertise and allowing manual curation to play a role in data cleaning and analysis. In-depth intermediate results exploration, however, has been shown to be an integral part of ensuring metagenome-assembled genome (MAG) quality, especially in the reconstruction of circularized and (near-)complete microbial isolate genomes [21, 22]. These factors combined left researchers with a world of metagenomic analysis pipelines generating substantially different results even in the same task with the same input [23], a nontrivial obstacle for future metagenomic research.

Specifically, this workflow system should be highly flexible and transparent in terms of tool selection, parameter setting, and intermediate result exploration, while maintaining stability and a reasonable level of user-friendliness. In addition, this system should be comprehensive and scalable, able to cover most, if not all, aspects of metagenomic research needs, and with built-in Slurm integration for parallel processing on HPC systems, instead of focusing only on specific tasks such as MAG reconstruction. As metagenomics is an actively developing field with new tools being actively developed and potentially new analysis components being developed in the future, a future-facing workflow system should have the capacity to smoothly incorporate new tools and components without breaking the overall structure and also provide the environment for different tools to be benchmarked and compared.

With these objectives in mind, we developed CAMP, short for ‘Core Analysis Modular Pipeline’, a Snakemake-based fully modular workflow system designed for core metagenomic analyses with the above-mentioned features. A modular architecture, in contrast to the traditional monolithic design in most one-click pipelines, is one that designs workflows as an interconnected suite of modules, each responsible for one single analytic task (e.g. *de novo* assembly). While the concept of modular structure has been existing and there have been pipelines that claim to have adopted a ‘modular’ architecture, such as nf-core/MAG [19], FinnLab/MAG_Snakemake_wf [17], and MUFFIN [18], these pipelines are only modular insofar as having text. Most of these pipelines offer only limited end-user flexibility, having predefined workflows and tightly coupled configuration schema that need to be adhered to, such as in the case of nf-core/mag [19]. Tweaking of these prede-

finer structures often not possible without the modification of the internal code. In contrast, CAMP is fully modular with lightweight parameter files for each individual module, offering users the maximum freedom in designing their own pipelines. Each CAMP module is fully self-contained while sharing a common command-line interface (CLI) and directory structure, balancing customization with usability.

Besides limited modularity, most of these abovementioned pipelines also suffer from a limitation of scope, focusing mainly on solving existing problems with focused goals such as assembly, binning, and profiling, and falls short in large-scale exploratory studies such that of MetaSUB [2]. CAMP, on the other hand, is designed for both current and future studies, both in its current comprehensiveness and in its flexible yet standardized design that allows unlimited number of future modules to be seamlessly incorporated. As an overview, we designed CAMP with the goal to establish a paradigm for next-generation metagenomic research by encapsulating the following features:

- **‘Set menu’-style computing to ‘a la carte’-style study design:** Switching from one-click pipelines to a modular analysis system allows the user to assemble the unique workflow for their specific study by downloading only the necessary modules.
- **Built-in soft pauses at the end of each module:** At the conclusion of each module, or ‘step’ in the workflow, the user can explore various semi-automated visualizations of their analytical results and apply their own knowledge base to enhance downstream analyses. This includes modifying downstream parameters from their default values.
- **Modules as benchmarking and comparison meta-tools:** A module can serve as a benchmarking ‘sandbox,’ akin to the Snakemake pipeline described by [24], where new methods can be seamlessly incorporated into a workflow and subsequently compared to other methods with similar objectives.
- **Compressed summary statistics tables:** The standardized output dataframes can serve as inputs for downstream machine learning ingestion.
- **Semiguide visualizations:** Visualizations are essential for large-scale dataset analysis, hypothesis generation, and reliability assessments. To make effective judgment calls, it is important that understand what CAMP is doing at all intermediate steps.
- **Module template for future expansions:** Each module is based on a standard directory template. New modules for new analysis purposes can be easily set up from scratch using the `cookiecutter` command within a few hours. This process includes creating Conda environments and generating module-specific parameter and resource files.

Methods

Each module is a GitHub repository containing core components organized in a standardized directory structure. This ensures consistency across the system, including the module directory (Box 1), working directory, parameterization, and input/output architectures. The core components include the `Snakefile`, `utils.py`, `ext/` directory, `parameter.yaml`, and `resource.yaml`, which are further customized for the

module’s specific purpose. The design features of the modular system are outlined in Table 1, and available analysis modules are further described in Fig. 1.

Box 1: Standardized module architecture for system-wide navigability and extendability.

`workflow/`: Contains all of the module’s functional code and scripts from external tools that are under an open-source license.

`module_slug.py`: The Click-based command-line interface that wraps a Snakemake-based workflow.

`Snakefile`: Workflow that itself wraps Snakemake rules, which call external algorithms or internal functions (`utils.py`) to process data.

`utils.py`: Internal functions such as dataset ingestion and working directory setup.

`ext/`: External scripts and smaller data files required by algorithms used in the workflow.

`configs/`: Contains all settings files.

`parameter.yaml` and `resource.yaml`: Instead of relying on Snakemake’s dictionary-based system of specifying parameters and resources on the command line, algorithm parameters, and computational resource allocations have been unified into single YAML files. Users can: (i) use the default YAMLs provided, (ii) modify the values in the YAMLs according to their dataset properties, or (iii) make multiple copies of the YAMLs to analyze datasets of different sizes and biological origins.

`samples.csv`: A standardized input file where each row corresponds to a sample. The first column is always the sample’s name, and every column thereafter contains paths to that sample’s data.

`conda/`: Conda environment recipes. The main environment is always called `module_slug.yaml` and should be set up immediately after cloning the module to a local directory. All other Conda environments will be set up the first time the workflow is run into a directory called `conda_envs/` in the module’s top-level directory.

`sbatch/`: A Snakemake profile that enables the module’s rules to be run using the Slurm job scheduler.

`test_data/`: Test data to ensure proper module setup, and an example work directory to demonstrate its structure, and input, intermediate, and output file formats.

`dataviz.ipynb`: Guided and semi-automated data visualization of the module’s final output using Jupyter notebook. Users can apply the built-in graphing functions as is to their own data, or customize as needed. Example visualizations generated using a toy dataset can be found in the Supplementary Materials.

Workflow management system

While Nextflow [26] and Snakemake [27] are both popular choices as WMSs for computational biology analysis, we opted for Snakemake due to its general readability and accessibility, particularly for researchers accustomed to using Python. Snakemake’s syntax is minimal and resembles standard Python, making workflows easy to understand, develop, expand, and troubleshoot. Although working with Snakemake’s wildcards may pose a learning curve, its directives (e.g. output, log) and ability to specify file names explicitly generally result in output directories that are easy to navigate and conducive to workflow reruns. Nextflow also offers many strengths as a WMS, with better support for cloud deployment

Table 1. Each feature in the module system was designed to maximize scalability (in terms of dataset size as well as system distribution), portability (i.e. compatibility with existing analysis environments), ease of use, and analysis transparency and reproducibility

Feature	How does it facilitate best-practices?
(i) Standardized module template	•Ease of use: Every single module in the system has the same structure as described in Box 1, which makes it extremely easy to (i) learn how to use existing modules, (ii) customize existing modules with new tools and rules, and (iii) make completely new modules. – The module template is a Cookiecutter [25] that can be used to start new modules from scratch in a few minutes. •Scalability: Thanks to a fully automated Cookiecutter-based template, the entire system is extremely flexible and can easily integrate developments in the metagenomics field. New modules for custom purposes easily be set up with the template and connected to existing modules in the system. For example, a hybrid short-long read assembly module can use as inputs the output <code>samples.csv</code> files from short- and long-read preprocessing.
(ii) Each module is a self-contained analysis unit	• Soft pauses are built into the end of every module so that the user can visually inspect the intermediate output and apply additional analysis steps that may not have been apparent at the outset. •Portability: Each module can be downloaded and set up individually. For example, if a user just wants to do short-read taxonomic classification, they only need the short-read preprocessing and classification modules and do not need to set up anything MAG-related.
(iii) Click-based Python CLI	• Snakemake’s API is unwieldy for coordinating multiple command-line parameters and computational resources, not to mention restarting and rerunning existing jobs. The intuitive Python-based command-line wrapper internally communicates with Snakemake’s API and allows users to interact with an easily operated interface. •Ease of use: The interface eliminates the need to write long, unwieldy, error-prone Snakemake commands.
(iv) Standardized input–output file format	• The input and output of every module is always in a standardized <code>samples.csv</code> format that encodes the locations of input or output data for each sample in the dataset. •Scalability: This simple file format is the glue that allows modules to be joined together into whatever complexity of workflow the user needs.
(v) Standardized working directory structures	• Ease of use: Similar to the standardized module directory, once the user is familiar with navigating the intermediate and output files of one module, all others are equally navigable. •Transparency and reproducibility: Intermediate and output reports as well as job logs are located in predictable and intuitive directory structures. If jobs fail, finding the error, whether it is due to computational under-resourcing or incorrectly set parameters, is straightforward.
(vi) Summary statistics of module outputs for accessible first-pass interpretations	•Scalability: Large datasets can be difficult to analyze without intuitive summaries. For example, the short-read taxonomic classification module comes with the taxa discovered by two classification algorithms as well as their estimated relative abundances in standardized CSVs for easy comparisons. •Portability, transparency, and reproducibility: By providing summaries at the end of each module, users can document every step of their analysis parsimoniously without having to keep every single large intermediate file.
(vii) Jupyter notebook-based semi-automated visualization of summary statistics	•Ease of use and scalability: While summary statistics are useful for documentation, visualizations are a more intuitive way to understand dataset outputs, especially large datasets with many samples that need to be compared.
(viii) Module-independent files for parameter and computational resource settings	•Scalability: While a default set of parameter and resource values are provided with the sample config YAML, users can set different parameter and resource profiles depending their data. •Portability, transparency, and reproducibility: To share the parameter and resource settings used in their analyses, users need only share these two text files. •Ease of use: Users only have to refer to these two text files to adjust parameters and resources without creating complex data structures on the command line.
(ix) Included test dataset and sample outputs from test dataset	•Portability: The user can verify proper module setup, as well as visually inspect the proper intermediate and output file formats and work directory to gain mechanistic familiarity the purpose of each step in the module.
(x) Conda-based environment setup with integrated YAMLS	•Scalability: Instead of re-installing existing Conda environments into every analysis working directory, Conda environments are built directly into the module directory only once. •Portability and ease of use: Complex conflict-free environments are deployed along with the module workflow code, and can be easily set up using standard Conda commands. •Transparency and reproducibility: Environments are standardized and auto-documented for version consistency.
(xi) Slurm job scheduler integration	•Scalability: Allowing modules to be run with job scheduler assistance helps users with access to HPCs to parallelize running rules within the module.

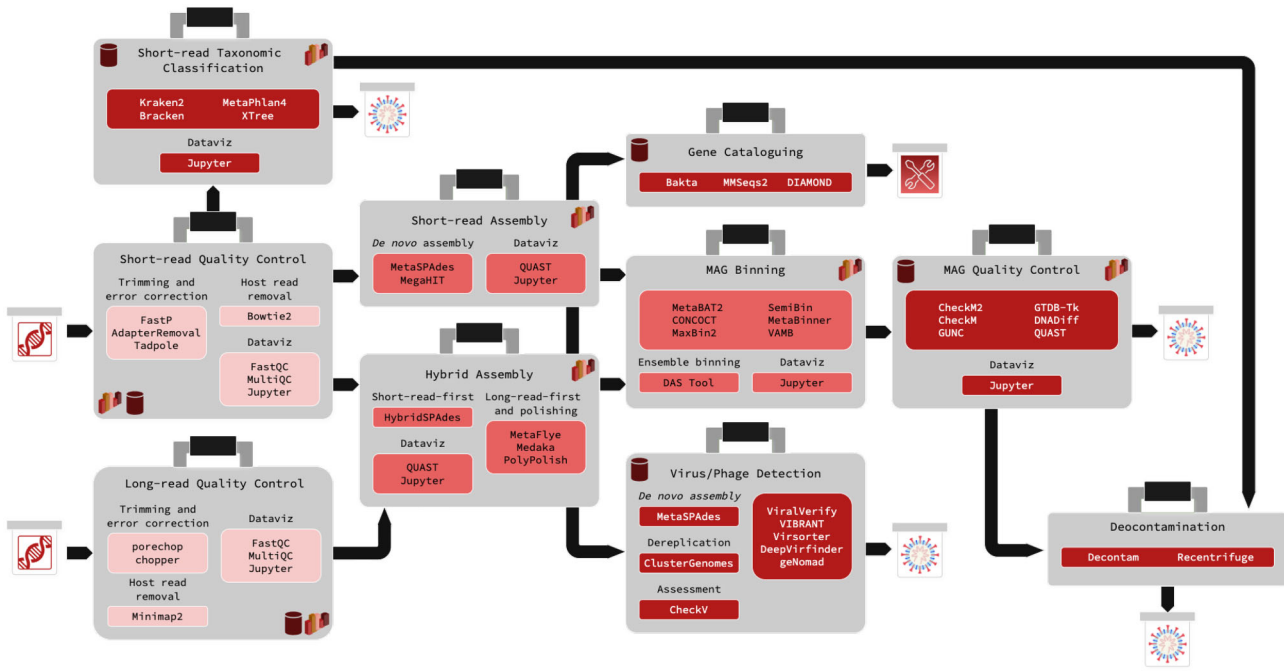


Figure 1. An overview of the available metagenomics analysis modules in the CAMP. All modules share the same internal architecture, but wrap a different set of algorithms (shown to the left of each box) customized to its particular analysis goals. Modules that are typically the beginning of analysis projects contain light red boxes, modules that are typically intermediate steps contain medium red boxes, and modules that are typically terminal analysis steps contain dark red boxes. Typical first inputs are indicated by the boxed DNA symbols, while terminal outputs that report some form of quality metric or taxonomic classification information are indicated by the boxed graphic bacteria. Functional profiling outputs are indicated with a boxed wrench and screwdriver. Modules that contain built-in dataviz notebooks contain a bar graph symbol, and modules that require database downloads contain a cylinder.

natively as well as nf-core, a repository of curated analysis pipelines developed by the community at large [28].

Environment management

We have opted not to implement the modular system with Docker [14] or Singularity [15], because many users of HPC clusters, lack root access privileges and thus cannot utilize containerized workflows. Instead, we have included conflict-free recipes in each module directory, each of which sets up a Conda environment directly. This allows the same environment to be used for all datasets processed using the module [29]. To ensure reproducibility and interoperability within the module environment, each Conda recipe has hard-coded algorithm versions and is designed specifically for Unix-based operating systems. In response to Anaconda’s licensing policy as of March 2024, the defaults channel have been phased out of each module’s Conda recipe. This allows users to recreate environments from YAML files without automatically pulling packages from the proprietary channel, facilitating a smoother transition for those shifting to community-supported channels like conda-forge that remain unrestricted.

Modular system use-cases

The primary strengths of a modular system are its flexibility and extensibility, complemented by the interdependency of the tools. This architecture enhances accessibility and simplifies maintenance, enabling a longer lifespan. Such a design is accessible not only to the developers of the module template and core modules but also to anyone with fundamental command-

line, Python, and Conda development skills. Below are a few use cases:

- (1) A user possessing a short-read dataset and seeking to conduct an initial analysis with short-read taxonomic classification can download two modules: ‘short-read preprocessing’ for the removal of low-quality and erroneous bases, and ‘short-read taxonomic classification’ for the generation of a unified report consolidating discovered taxa from two classification algorithms. Simplifying the learning process, the user interfaces with a single CLI, which remains consistent across all modules.
- (2) The user, intending to expand their analysis to include gene annotation, can effortlessly do so by downloading and executing two additional modules: ‘short-read assembly’ and ‘gene cataloguing’. No additional onboarding time is necessary in terms of interface learning. Moreover, the input for short-read assembly remains consistent with that of short-read taxonomic classification, utilizing the `samples.csv` file containing paths to preprocessed short reads.
- (3) The user, aiming to broaden their analysis to include MAG reconstruction, can seamlessly achieve this by downloading the binning and MAG quality-checking modules. Utilizing the output of short-read assembly, specifically the `samples.csv` file containing paths to the *de novo* assemblies, as their input, no additional onboarding time is required to learn new interfaces.
- (4) A user who has developed a new short-read taxonomic classification tool can leverage the existing short-read taxonomic module as a sandbox environment. They

can write a custom Snakemake rule to execute their tool, implement a function to harmonize the output format with the existing merged report, and conduct benchmarking analyses to assess the strengths and weaknesses of their tool.

- (5) A user aiming to study extra-chromosomal elements, given the absence of an existing module for this purpose, can create a new blank module utilizing the Cookiecutter template. They can then populate it with the requisite Snakemake rules and customize the remaining demo configurations to suit their specific objectives. Leveraging the standardized input–output format ensures seamless integration of this new extra-chromosomal elements module with the broader module system, facilitating reuse by both the developer and the wider research community.

Currently available modules

There are ten modules available, and seven are used in this proof-of-concept analysis to accomplish one of three purposes: (i) short-read taxonomic classification, (ii) MAG reconstruction and quality-checking, and (iii) virus and phage inference from short-read sequencing datasets (Fig. 1). There are several modules currently under active development, including Oxford Nanopore long-read preprocessing, marker gene-based operational taxonomic unit (mOTU) detection, and decontamination. The algorithms included for each module were chosen based on previously documented performance in benchmarking studies, as well as code stability and overall workflow compatibility.

General-purpose analysis

Module 1: Short-read preprocessing

Raw sequencing datasets are filtered for low-quality bases, low-complexity regions in reads, extremely short reads, and adapter content using fastp [30]. Reads can optionally be deduplicated. If host read removal is selected, trimmed and filtered reads are mapped using Bowtie2 and Samtools with the ‘very-sensitive’ flag to the host reference genome (here, the human reference genome assembly GRCh38), and mapped reads removed [31, 32]. As a last-pass, BayesHammer or Tadpole is used to correct sequencing errors [33, 34]. FastQC and MultiQC are used to generate overviews (e.g. parameters such as per-base quality scores, sequence duplication levels) of processed dataset quality [35, 36].

Module 2: Nanopore long-read quality control

Raw sequencing datasets are trimmed using PoreChop [37], and then low-quality bases are filtered out using chopper [38]. Host reads are optionally removed using Minimap2 [39]. FastQC and MultiQC are used to generate overviews (e.g. parameters such as per-base quality scores, sequence duplication levels) of processed dataset quality [35, 36].

De novo assembly

Module 3: Short-read assembly

The processed sequencing reads can be assembled using MetaSPAdes (with optional flags for metaviral and/or plasmid assembly also available), MegaHIT, or both [40, 41]. For the purposes of this study, only MetaSPAdes was used. The assembly is subsequently summarized using QUAST [42].

Module 4: Hybrid assembly

For short-read-first hybrid assembly, processed short sequencing reads are assembled with hybridmetaSPAdes, with the draft assembly polished by processed long reads [43]. For long-read-first hybrid assembly, processed long reads are assembled with MetaFlye [44], before being corrected first with the long reads using Medaka [45] and then with short reads using PolyPolish [46]. The assemblies are subsequently summarized and compared using QUAST [42].

MAG inference and quality-checking

Module 5: MAG binning

Processed sequencing reads are mapped back to the *de novo* assembled contigs using Bowtie2 and Samtools. This read coverage information, along with the contig sequences themselves, are used as input for the following binning methods: MetaBAT2, CONCOCT, SemiBin, MaxBin2, VAMB, and MetaBinner [47–52]. The sets of MAGs inferred by each algorithm are used as input for DAS Tool, an ensemble binning methods, to generate a set of consensus MAGs scored based on the presence/absence of single-copy genes (SCGs) [53]. Some of the contig pre-processing scripts were adapted from the MAG Snakemake workflow [17].

Module 6: MAG quality-checking

The consensus refined MAGs are quality-checked using an array of parameters. CheckM2 calculates completeness and contamination based on the annotated gene content of a MAG [54]. CheckM1 calculates per-MAG short-read coverage and strain heterogeneity based on the proportion of presence of multiple copies of single-copy marker genes that pass a high amino acid identity threshold [55]. gunc is also used to assess contamination [56]. MAGs are classified using GTDB-Tk, which relies on approximately calculating average nucleotide identity (ANI) to a database of reference genomes [57]. For MAGs with a species classification, their contig content is compared to the species’ reference genome and genome-based completion, misassembly, and nonalignment statistics calculated using dnadiff and QUAST [42, 58]. Each MAG’s gene content, with an emphasis on transfer RNA (tRNA) and ribosomal RNA (rRNA) genes in accordance with MIMAG genome reporting standards [59], is summarized using prokka [60].

The overall score per MAG was calculated using CheckM2-calculated completeness and contamination, and the contiguity metric N50: completeness – $5 \times$ contamination + $0.5 \times \log(N50)$. This equation was adapted from dRep’s overall score equation, which is based on completeness, contamination, contiguity, strain heterogeneity, and genome size, and [61]. dRep used this score to select a representative genome from a cluster of genomes with similar sequences. Various versions of this equation exist, with most variants setting the coefficients for strain heterogeneity and genome size to 0 [62–64].

Other analysis goals

Module 7: Short-read taxonomic classification

The processed sequencing reads can be classified using MetaPhlan4, Kraken2/Bracken, and/or XTree [65–68]. XTree, formerly referred to as UTree, is additionally included as an experimental short-read classification option, but this study focuses on comparing the taxa discovered by the two pub-

Table 2. A comparison of features found across several recently published metagenomics analysis pipelines and CAMP

Feature	CAMP	ATLAS	MAG Snakemake wf	MUFFIN	nf-core/mag	nbis-meta
Language	Snakemake	Snakemake	Snakemake	Nextflow	Nextflow	Snakemake
Modular design?	+	-	-	-	-	-
Install testing?	+	+	-	+	+	-
Containerized?	-	+	+	+	+	+
Documentation?	++	+	++	++	++++	+
Data visualization	+	-	+	-	-	-
Nanopore long-read QC	+	-	-	+	+	-
Hybrid assembly	+	-	-	+	+	-
Co-assembly and co-binning	-	-	+	-	+	-
MAG binning	6+1	2+1	3+1	3+1	3+1	3
Short-read taxonomic classification	+	-	-	-	+	+
Virus/phage detection	+	-	-	-	+	-

The symbol ‘+’ indicates that the feature has been implemented in the pipeline, with more ‘+’ indicating extensiveness. Conversely, the ‘-’ indicates that the feature is absent. The ‘MAG Binning’ row is formatted as X+Y, where X refers to the number of MAG binning algorithms and Y indicates the presence of a consensus binning tool.

lished field-standard tools—Kraken2 and MetaPhlan4. To estimate the relative abundance of a taxon, MetaPhlan4 calculates marker gene coverage and Bracken calculates the proportion of reads assigned to a taxon with k-mer uniqueness-based scaling [67]. Since each of these output reports are of different formats, the raw reports from each algorithm are standardized in format for easier comparisons downstream.

Module 8: Virus/phage inference

The processed sequencing reads are assembled with MetaSPAdes, and viral contigs are subsequently identified using the output assembly graph and ViralVerify [40, 69]. Contigs containing putative viral genetic material are also identified using VIBRANT, VirSorter2, DeepVirFinder, and geNomad [70–73]. The aggregated lists of contigs from the three inference methods is dereplicated using VirClust [74] and merged with the ViralVerify list, and the overall quality of the putative viruses is assessed using CheckV [75].

Module 9: Gene cataloging

Open reading frames (ORFs) are identified in the *de novo* assembly using Bakta, and clustered using MMSeqs [76, 77]. Genes are identified from these ORFs by alignment to the DIAMOND database to obtain the functional profile of the sample [78].

Module 10: Decontamination

The decontamination module is still under construction. A feature table of relative abundances [e.g. operational taxonomic units (OTUs), taxa, MAGs] is provided to Decontam and Recentrifuge, each of which estimates contamination from feature abundances either within or between samples respectively [79, 80].

Module 11: mOTUs profiling

The long-read profiling module is still under construction. It wraps mOTUs [81], which estimates the relative abundance of taxa from a short- or long-read sequencing dataset, and calls single-nucleotide variants from marker genes.

Comparison with other workflows

To further delineate what sets CAMP apart, we compare CAMP with five other pipelines commonly used in metagenomics analysis: ATLAS [16], FinnLab/MAG_Snakemake_wf [17], MUFFIN [18], nf-core/mag [19], and nbis-meta [20] (Ta-

ble 2). It is worth pointing out that these pipelines, though used in an array of modern metagenomic analysis procedures, are mostly MAG-focused, as many of the names suggest. CAMP offers a distinctly different philosophy of project design, which envisions a comprehensive and extensible system capable of supporting all core and extant analyses in current and future metagenomic research.

For example, aside from CAMP, only nf-core/mag offers both short-read taxonomic classification and virus/phage detection—two additional analysis results unrelated to MAG binning and quality-checking [19]. Even with the respect to core MAG-related tasks, however, we have demonstrated our commitment to building modules that are easily extensible. CAMP incorporates six different binning algorithms, which is far and away the maximum of three incorporated by other workflows (Table 2). In addition to the classic options such as MetaBAT2 [47], CONCOCT [48], and MaxBin2 [51] which are commonly used in other pipelines, the binning module includes newer machine learning-based binning tools such as VAMB [50] and SemiBin [49]. In the ‘Results’ section below, we further demonstrate why diversity on perspective in MAG binning is necessary. Different binning strategies return different numbers of MAG bins, many of which are of questionable quality that would not have been identified using the classic options alone. For large-scale studies that want to balance MAG quality with speed, CAMP goes above and beyond to provide the most robust analysis results.

Results

Proof-of-concept and data

To demonstrate CAMP’s applicability for large-scale metagenomics analysis and inter-sample comparison, we have applied seven modules to a randomly selected set of 10 samples from the International Metagenomics and Metadesign of Subways and Urban Biomes (MetaSUB) dataset, which consisted (at time of [2]’s publication) of 4728 samples of microbiomes collected from urban transit system surfaces between 2015–2017. Sample collection, metagenomic DNA extraction, and sequencing information can be found in [2] and the [GeoSeq repository](#). The raw sequencing data can be found under the SRA accession ID [PRJNA732392](#), as well as the [GeoSeq repository](#). The 10 randomly chosen samples were collected from subway train and transit station surfaces from three of the four boroughs of New York that are

Table 3. Ten randomly chosen urban microbiome samples from the MetaSUB New York dataset along with metadata such as their dates and locations of origin

Sample ID	Full sample name	Sampling time	Train line	Transit station (borough)	Swabbed surface	Num. read pairs
0	haib17CEM5106_ HCY5JCCXY_SL269539	Winter 2014	D	NA	Subway train seat	5933 694
1	haib17CEM5106_ HCV72CCXY_SL269726	Winter 2014	6	59 St. (Manhattan)	NA	2002 519
2	haib17CEM5106_ HCVMTCCXY_SL269540	Winter 2014	D	NA	Subway train seat	2864 273
3	haib17CEM5106_ HCV72CCXY_SL269724	Winter 2014	6	Grand Central (Manhattan)	NA	1814 764
4	haib17DB4959_ H3MGVCCXY_SL259896	June 2017 (City Sampling Day)	NA	Van Siclen Av. (Brooklyn)	Transit station bench	8285 289
5	haib17CEM5106_ HCV72CCXY_SL269741	Winter 2014	6	NA	Subway train ceiling handrail	3864 641
6	haib17CEM5106_ HCY5JCCXY_SL269601	Winter 2014	NA	36th St. (Queens)	Transit station turnstile	3760 124
7	haib17CEM4890_ H2NYMCCXY_SL254808	June 2016 (City Sampling Day)	NA	Halsey St. (Brooklyn)	Transit station turnstile	23 076 691
8	haib17DB4959_ H3MGVCCXY_SL259845	June 2017 (City Sampling Day)	NA	Flushing-Main St. (Queens)	Transit station escalator handle	5349 073
9	haib17CEM4890_ HMC MJCCXY_SL335779	June 2016 (City Sampling Day)	NA	33rd St. (Manhattan)	Transit station kiosk	4834 819

connected by subway transit lines- Manhattan, Queens, and Brooklyn (no samples from the Bronx were in the subset, Table 3). Most of the samples were obtained from transit station surfaces in the winter of 2014, and generally contain between 2 and 6 million paired-end reads. With the exception of the workflow figure, all other data-based graphics in this manuscript were generated with the Jupyter notebooks built into each module, which users can also apply to summarize their data visually. Example visualizations generated using a toy dataset can be found in the Supplementary Materials.

Short-read quality control

Between 66.6%–93.8% of short sequencing reads and 65.3%–89.8% of sequencing bases were retained after low-quality base-trimming, adapter trimming, host (human genome GRCh version 38) genome removal, and error correction steps were applied to each of the 10 samples (Fig. 2 and Supplementary Table S1). The largest reduction in dataset size occurred at the *fastp* low-quality read removal step, with host read removal by *Tadpole* also eliminating some sequence content (Fig. 2A). No reads and bases were lost at the adapter step despite adapters being present in the pre-flight MultiQC check, indicating that all of the adapter bases were trimmed by *fastp*.

After preprocessing, most samples passed most MultiQC quality metrics. For the reverse reads of samples 8–9 and samples 5–7, there were outstanding warnings for the per-tile sequencing quality (example in Supplementary Fig. S1A) and per-base sequence content (example in Supplementary Fig. S1B). Upon checking the FastQC plots for samples 8–9, in both cases, the tile failed at one position in particular (towards the end), not across all bases (Supplementary Fig. S1A). Since

the average per-base quality is generally high, no further trimming was pursued. For samples 5–7, the ACGT content was uneven for the first 10 bases (Supplementary Fig. S1B). It has been demonstrated previously in RNASeq datasets that 5' nucleotide biases can be caused by steps in library preparation [82]. Since all adapters were removed and overall sequencing quality was high, no further trimming was pursued.

Short-read assembly

The distribution of assembly sizes mostly follows the distribution of short-read dataset size, with the exception of sample 4 (Figs 3A and 2B). Although the size of sample 7 is nearly $3\times$ larger than sample 4 (in terms of both numbers of reads and bases), sample 4 has the largest assembly by far in terms of both numbers of contigs and bases (Fig. 3). MultiQC reported that samples 4 and 7 contained 7.4% and 10.9% duplicated reads overall, which accounts minimally for the difference. Sample 4 may contain a much lower proportion of PCR duplicates. As shown later by both short-read taxonomic classification and MAG binning, sample 4 does not contain appreciably fewer species than sample 7. The largest, average, and median contig sizes are similar, though there is sizable variance in terms of assembly contiguity as measured by N50 (Supplementary Table S2).

The vast majority of contigs in all *de novo* assemblies are extremely small and fall below the size cutoffs of most MAG binning algorithms (two examples in Table 4). Smaller contigs are more likely to contain low-quality bases and have irregular coverage [83], and are also more likely to originate from low-abundance organisms [84]. To retain only high-quality sequencing information for binning, we selected a MAG binning minimum contig cutoff of 2500 base-pairs. Only a small frac-

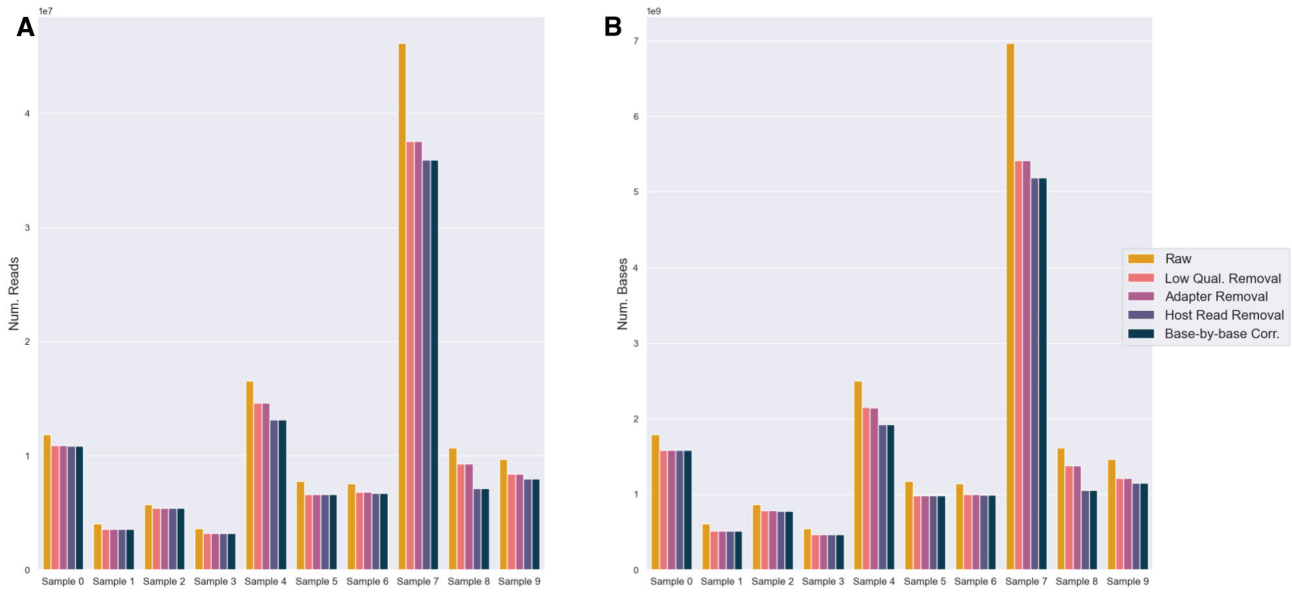


Figure 2. Quality control and preprocessing of short-read sequencing data. Counts of (A) reads and (B) bases retained after each preprocessing step across all samples. The steps include low-quality base trimming, adapter removal, host genome removal, and error correction.

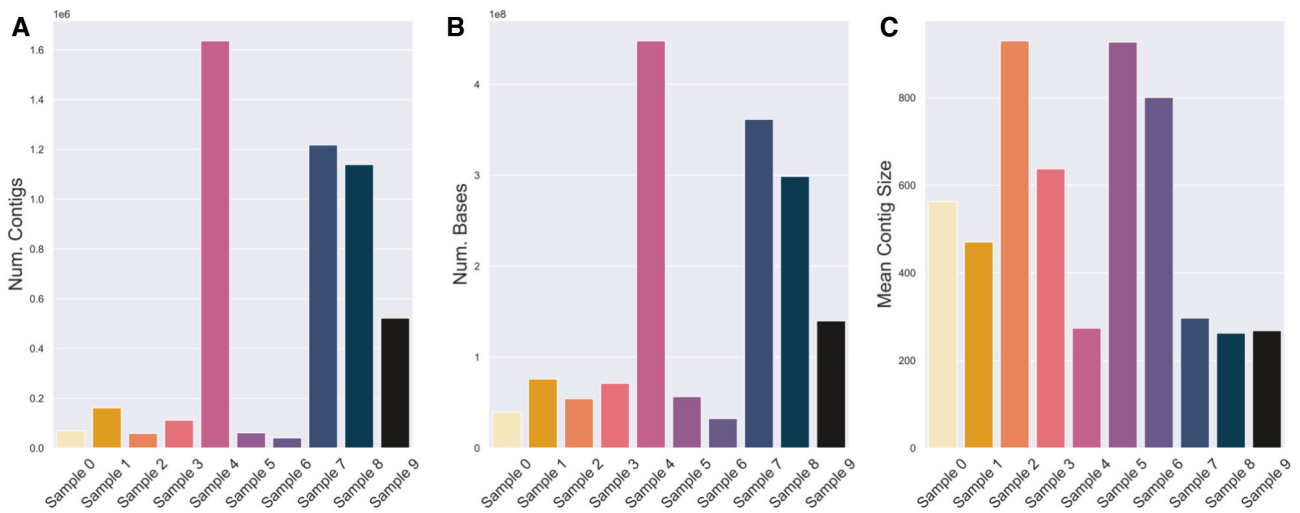


Figure 3. *De novo* assembly sizes generally correlate with short-read sequencing dataset sizes. (A) The number of contigs and (B) number of sequencing bases in each sample's assembly, as well as the (C) mean contig size.

Table 4. With a minimum contig size threshold of 2500 bp, a tiny minority of contigs—between 0.1% and 0.5%—will be kept for MAG binning

Sample	Min. contig size	Num. contigs (prop.)	Num. bases (prop.)
4	0 bp	1636 542 (1.000)	447 963 811 (1.000)
4	2500 bp	1872 (0.001)	10 832 446 (0.024)
4	50 000 bp	12 (0.000)	1000 212 (0.002)
4	100 000 bp	3 (0.000)	342 030 (0.001)
7	0 bp	1217 385 (1.000)	361 500 456 (1.000)
7	2500 bp	6605 (0.005)	53 796 944 (0.149)
7	50 000 bp	94 (0.000)	11 802 085 (0.033)
7	100 000 bp	41 (0.000)	8342 312 (0.023)

tion of the assemblies are of contigs larger than 50 kbp (0.2% at maximum), which is $\sim 1\%$ of the size of the average bacterial genome [85].

MAG binning

Aside from VAMB and samples 7–9, the number of MAGs inferred by each binner (including the MAG inference aggregator and ensemble binner DAS Tool) was generally within a factor of two of each other (Fig. 4A). In general, the more MAGs inferred by a binner, the smaller the inferred MAGs on average in terms of the numbers of sequencing bases (Fig. 4A and C). The three binner that do not use single-copy marker genes to estimate the number of MAGs—MetaBAT2, CONCOCT, and SemiBin—tend to infer a larger number of MAGs (Fig. 4A) [47–53]. SemiBin only uses single-copy marker genes to further partition MAGs with an average of > 1 copy per gene [49]. While MaxBin2 and MetaBinner generally inferred the same number of MAGs, MaxBin2 MAGs generally contain many more contigs but are similar in overall size (as counted by base pairs). This can be explained by MetaBinner's stringent post-clustering contig reassignment step; MetaBin-

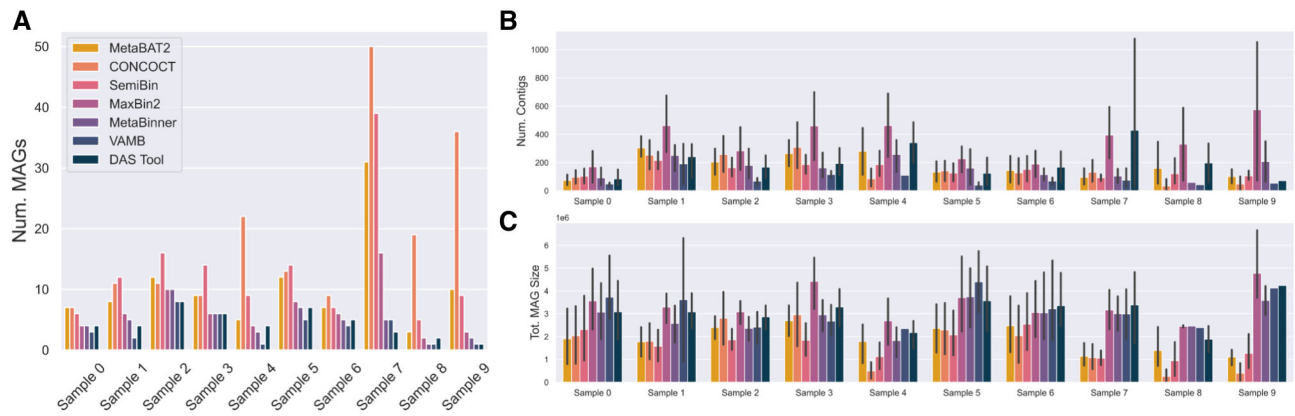


Figure 4. Most MAG binning algorithms infer a consistent number of MAGs, with the exceptions of samples 4, and 7–9 and VAMB. **(A)** Number of MAGs inferred by each binning algorithm across samples. **(B)** The numbers of contigs per. **(C)** Total MAG size inferred for each sample and comparison across binners (bars indicate standard deviation).

Table 5. The number of MAGs inferred by the six binning algorithms and ensemble-aggregated by DAS Tool, as well as the number of MAGs that can be classified by GTDB-Tk

Sample	Num. DAS tool MAGs	Num. classifiable MAGs	Num. species-classifiable MAGs
0	4	4	3 (75%)
1	4	4	4 (100%)
2	8	8	4 (50%)
3	6	6	5 (83.3%)
4	4	4	3 (75%)
5	7	7	5 (71.4%)
6	5	5	5 (100%)
7	3	3	1 (33.3%)
8	2	2	2 (100%)
9	1	1	1 (100%)

ner uses a MetaWRAP-like consensus method to remove contigs that are not robustly associated with cluster centroids [52].

Despite having the largest assembly, only four MAGs were inferred from sample 4 by DAS Tool (Table 5). Furthermore, the size of the MAGs is ~2 Mbp, which is smaller than the average bacterial genome (Fig. 4C) [85]. The largest number of MAGs was inferred from sample 2 (Table 5), which was the third smallest dataset and assembly (by the number of bases, Figs 2B and 3B). However, sample 2 also had the largest average contig size, so more contigs were included in the binning process after applying a minimum size threshold of 2500 bp (Fig. 3C).

Aside from VAMB, DAS Tool generally inferred the fewest MAGs, though they are usually the largest and similar to the average size of a bacterial genome (Fig. 4C) [85]. In terms of overall assembly utilization, DAS Tool MAGs contained between 91.3% and 99.7% of usable bases in contigs >2500 bp.

MAG QC

Most of the inferred MAGs (27/44, 61.3%) across all samples were high-quality, according to a slightly modified version of the MIMAG standards naming conventions (Fig. 5R) [59]. MAGs that are high-quality but missing the requisite rRNA and tRNA genes are still called high-quality, and those with are called ‘draft’ instead. There were no MAGs that con-

tained all of the rRNA and tRNA genes required for ‘draft’ status.

Similarly, most of the MAGs (34/44) had CSSs that fell under the GUNC threshold for allowable taxonomic chimerism. Of the low-quality 5/44 (9.2%) MAGs, 2/5 had low CheckM2-estimated contamination and gunc-estimated CSS and were still classifiable to the species level. The remaining low-quality high contamination MAG (CheckM2: 10.81, gunc: 0.97) was classified to the genus *Pseudomonas*. Given the high CSS, the MAG may be a chimeric bin of contigs from multiple genera in the *Pseudomonadaceae* family. CheckM2-estimated completeness was $\leq 50\%$ for only a single MAG, and contamination was ≥ 10 for three (Fig. 5B). Of the species-classifiable MAGs, contamination was at maximum 5, but some had high CSS, up to 1.

Recall that only a small fraction of contigs were larger than 50 kbp (Table 4). The average size of the contigs in the MAGs of samples 4 and 7 are 9.7 and 18.3 kbp, respectively (Fig. 4C). The median NA50 of the classifiable MAGs are 8.5 and 35.3 kbp, respectively (Fig. 5M), indicating that most of the assemble-able and mappable content of a MAG is concentrated within a small number of large contigs. The rest of the associated reference genome may be difficult to assemble and may have different sequence properties than the rest of the contigs. The use of BLAST or a taxonomic classifier (e.g. Kraken2, MetaPhlan4, CAT) [65, 67, 86] to classify short contigs may be more appropriate, as they may be small fragments from low-abundant species too poorly covered by sequence content to bin into MAGs.

The median relative abundance of each MAG per sample is generally between 10% and 20%, with half of the distributions having some right skew i.e. there are a few MAGs where the sample relative abundance is much higher than the average (Fig. 5E), which generally agrees with the expected log-normal distribution of bacterial relative abundances in a sample [87, 88].

Completeness is highly correlated with QUAST-estimated genome fraction (the fraction of the reference genome that the MAG aligned to) (Fig. 6A). However, contamination and CSS do not correlate with the proportion of unaligned sequence material (Fig. 6B and C). Though both CheckM2 and gunc use a MAG’s protein content to estimate contamination, they differ in terms of application and thus interpretation. CheckM2 uses a neural network trained from al-

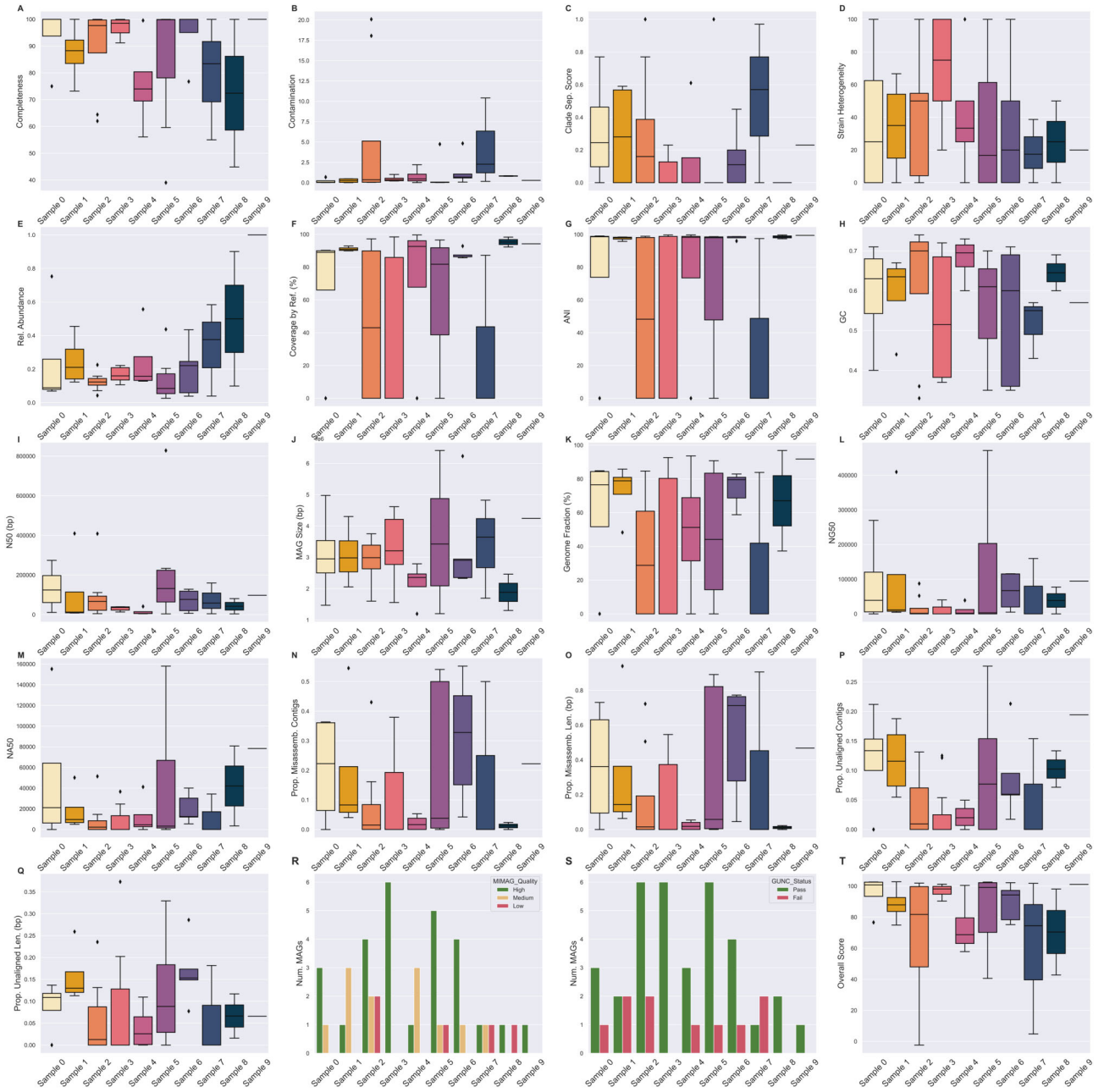


Figure 5. Quality assessment metrics of DAS Tool-inferred MAGs across all samples. The median of plot's metric across all of the MAGs inferred from that sample is indicated with a line across the box. **(A)** Completeness, **(B)** contamination, **(H)** GC, **(I)** N50, and **(J)** MAG size were reported by CheckM2. **(C)** Clade separation score (CSS) was reported by gunc. **(D)** Strain heterogeneity and **(E)** relative abundance were reported by and calculated from CheckM1 respectively. **(F)** MAG coverage by the GTDB-Tk classified reference genome and **(G)** ANI to that genome were reported by dnadiff. Metrics **(K–Q)** were reported by QUAST comparing each MAG to the GTDB-Tk classified reference genome. **(T)** was calculated from the MAG's completeness, contamination, and N50.

most protein-annotated 5000 RefSeq bacterial genomes [54]. gunc classifies the candidate proteins and measures the taxonomic entropy within a contig and the MAG overall [56]. If a MAG is high in CheckM2-estimated contamination but low in gunc-estimated CSS, then the MAG's extraneous sequence content likely belongs to the same or a related taxon but still contains extra protein. The MAG may be a chimera of multiple highly related species or strains. Conversely, if a MAG is low in CheckM2-estimated contamination but high in gunc-estimated CSS, the extraneous sequence content may be nonredundant proteins from more distantly related taxa

that nevertheless aligns to the reference genome (e.g. different genera). At least for this pilot dataset, there seems to be a stronger correlation between the CSS and the proportion unaligned bases than CheckM2-estimated contamination (Fig. 6B and C), potentially indicating that there is more inter-species chimerism within a MAG. Both metrics only consider the over- or under-abundance of protein content. If there are contaminant nonprotein-coding sequences, or sufficient synonymous mutation-based variation, that unalignable sequence material would not be considered by either metric. In contrast, strain heterogeneity—which measures the proportion

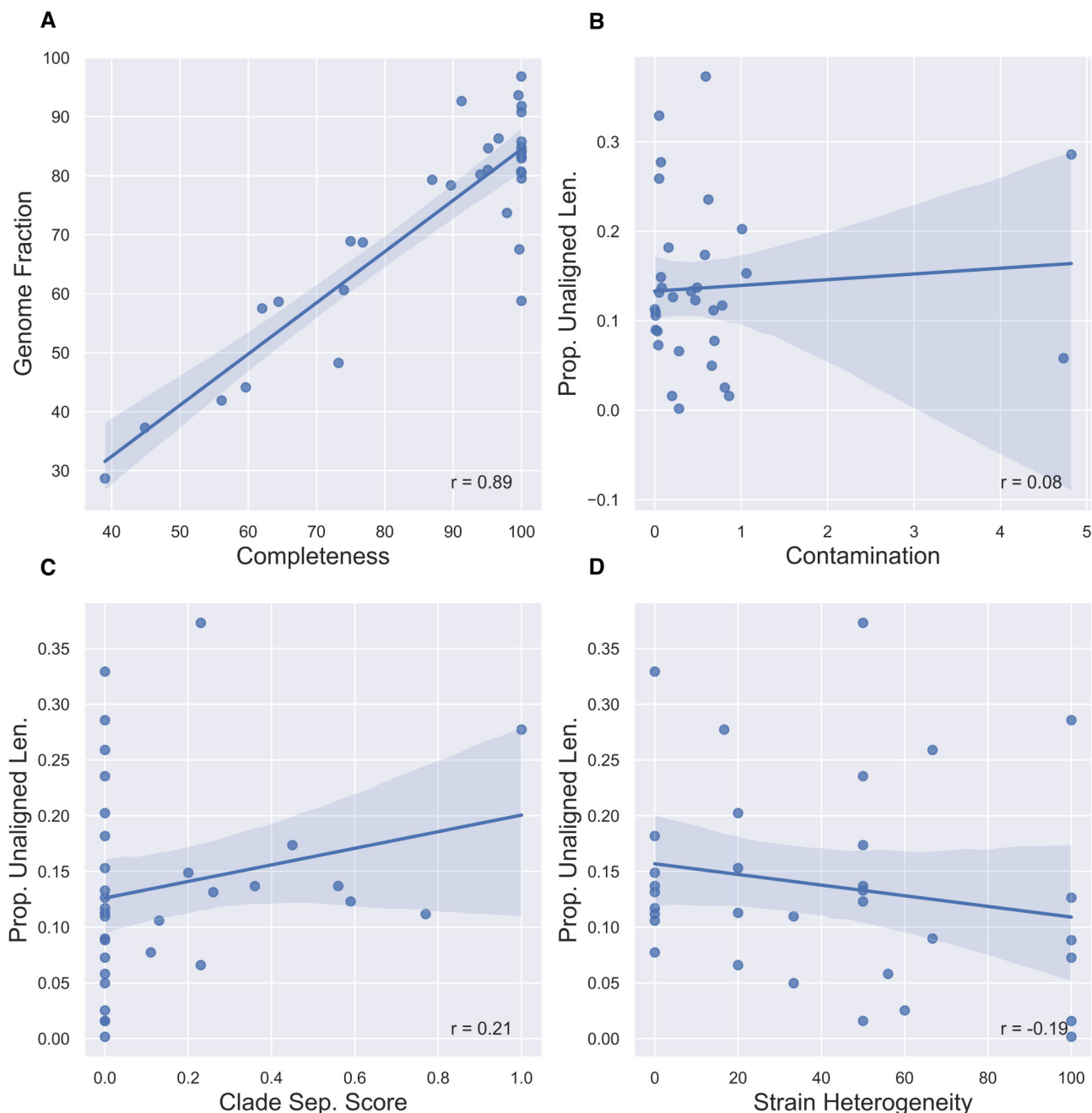


Figure 6. Reference-free metrics are sometimes correlated with reference-based metrics. **(A)** Completeness is highly correlated with QUAST-estimated genome fraction (the fraction of the reference genome that the MAG aligned to). **(B, C)** Contamination and CSS do not correlate with the proportion of unaligned sequence material. **(D)** Strain heterogeneity is mildly inversely correlated with the proportion of unaligned material.

of multiple copies of SCGs with high minimum amino acid similarity—naturally does not correlate with unaligned material, which by definition does not pass a sequence similarity threshold (Fig. 6D).

Consistently, a larger proportion of the MAG is alignable to the reference genome it is classified as, rather than the converse (Supplementary Fig. S2). Much of the reference genome material is missing in the MAG, which is corroborated by the median sizes of the MAGs—3 Mbp, which is small for a bacterial genome (Figs 4C and 5J) [85]. Since the median ANI between a MAG and the reference genome of its classified species (if available) was 97.34 (Fig. 5G), the inferred MAGs are likely a specific substrain of the associated species. For example, the

most abundant MAG in sample 4 was classified as *Cutibacterium acnes* and had an average coverage of 55.1X. A total of 93.7% of the species reference genome was present in the MAG, and there were 573 positions with alternate alleles relative to the reference. Within the MAGs themselves, there was generally some degree of strain heterogeneity (Fig. 5D).

The overall score is a summary statistic that takes into account gene content completeness, extra-lineage gene contamination, and overall sequence contiguity (2). While each MAG's overall score is overall correlated with the MIMAG quality (Kruskal–Wallis $H = 34.364$, $P = 3.45 \times 10^{-8}$), it is surprisingly not correlated with whether or not the MAG is classified at the species level by GTDB-Tk (Kruskal–Wallis

$H = 0.0046$, $P = 0.946$). The latter could be due to the fact that all MAGs were classified to the genus level and 75% of the MAGs were classifiable at the species level. Ergo, the overall score and MIMAG quality are high enough for both classified and unclassified that they are not likely to be distinguishing factors. Similarly, MIMAG quality is not correlated with species classification (χ^2 -square = 3.822, $P = 0.148$). However, it is correlated with size (Kruskal–Wallis $H = 7.291$, $P = 0.026$) and even more strongly with N50 (Kruskal–Wallis $H = 22.751$, $P = 1.147 \times 10^{-5}$).

Comparing minimum contig size thresholds on MAG quality

To determine if the minimum contig size threshold had appreciable impacts on MAG recovery and quality, we also binned samples 3 and 7 (the smallest and largest short-read sequencing datasets, respectively) with a minimum size threshold of 1000 bp instead of 2500 bp. The results were highly dataset-dependent, and displayed no clear trends in terms of overall MAG quality (Supplementary Fig. S3 and Table 3). For maximally robust MAGs, it may be beneficial to bin assemblies with multiple minimum contig size thresholds, and then use an ensemble or dereplication tool like DAS Tool or dRep to find the most robust contig clusterings.

As expected, the sample 7 MAGs had lower CheckM2-estimated completeness and contamination when binned with a 2500 bp threshold than 1000 bp. Conversely, the sample 3 MAGs were higher in both quality metrics with a threshold of 2500 bp (Supplementary Table S3). If the species-classified MAGs are considered alone, the average genome fraction (as measured by QUAST, represents the fraction of the associated reference genome found in the MAG) is higher with the 2500 bp cutoff, despite having the same average MAG sizes (Supplementary Fig. S3H and I). The MAGs assembled with the 1000 bp threshold also had proportionally more unaligned contigs and bases (Supplementary Fig. S3N and O).

In terms of recovered species, the same set of taxa were recovered in both 1000 and 2500 bp binning attempts of sample 3. The MAGs classified as *D. sp003523545* and *M. sp001878835* were larger with the 2500 bp threshold (Supplementary Table S3).

However, the 1000 and 2500 bp attempts at MAG inference for sample 7 returned slightly different sets of taxa (Supplementary Table S3). Specifically, the 2500 bp attempt at MAG inference did not recover *Chimaeribacter coloradensis* specifically, only a MAG that was classified as *Chimaeribacter* at the genus level (Supplementary Table S3). The 1000 bp attempt at MAG inference recovered both the species *C. coloradensis* as well as a separate *Chimaeribacter*-only MAG, although the MAG was larger than the reference genome size (5.6 versus 4.7 Mbp, respectively), and the contamination and CSS were high (22.66, 0.52). The *Pseudomonas B*-classified MAG in the 2500 bp attempt was larger than the potential analog in the 1000 bp attempt, which was classified at a higher resolution as *P. B. luteola*. However, the completeness of the 2500 bp attempt MAG was low, and contamination/CSS were high. As such, the MAG inferred with the 2500 bp size threshold is likely a chimera of several *Pseudomonas* species, as noted previously.

Short-read taxonomic classification

The short-read taxonomic classification Jupyter notebook semi-automatically compares the 30 technical replicates

Table 6. The relative abundance estimates for the six most high-abundance species discovered by Kraken2–Bracken and/or MetaPhlan4 can vary by a factor of 10

Species	Kraken2/Bracken	MetaPhlan4
<i>Acinetobacter lwoffii</i>	0.039822	0.382999
<i>Acinetobacter radioresistens</i>	0.004897	0.052929
<i>Acinetobacter schindleri</i>	0.181852	0.159433
<i>Acinetobacter variabilis</i>	0.006946	0.069463
<i>Kocuria rosea</i>	0.001764	0.005049
<i>Mixta calida</i>	0.010038	0.101190

(10 samples $\times$ 2 classifiers) using 2 alpha diversity metrics [taxonomic richness (i.e. number of taxa and Shannon entropy)] and 2 beta diversity metrics (Bray–Curtis dissimilarity and Jaccard distance) at different taxonomic ranks: phylum, class, order, family, genus, and species.

Despite discovering fewer unique taxa at every taxonomic rank overall, MetaPhlan4 almost always discovered more taxa than Kraken2–Bracken (Fig. 7). When comparing shared taxa across all 20 technical replicates, both the Bray–Curtis and Jaccard distances between each replicate’s species profile indicates that there is almost total assortment of species by classifier (Fig. 8A and C). A total of 137 unique species were discovered across both classifiers, with the vast majority (599/630, 95.1%) found by a single classifier. The vast majority—131/137 species (95.6%)—was discovered by only one of the classifiers. Even for the 6/137 (4.4%) species discovered by both, the estimated relative abundance of the universally discovered species varied widely (Table 6). At the higher rank of phylum, there is less distinctive classifier-specific assortment, but the taxonomic profiles of the same sample generated by different classifiers tend not to cluster together (Fig. 8B and D). There are three phyla present in a majority of classifier-sample replicates: Proteobacteria, Actinobacteria, and Firmicutes (Fig. 9), which were previously noted as the most prevalent three phyla by the original MetaSUB publication [2]. The relative abundance accounted for by all of these phyla (per sample) are 0.941 (± 0.081) ranging from [0.443, 1].

We conducted a small case study on the species discovered by the short-read classifiers in sample 7, and found that each classifier’s associated database contains some but not all of the species found in the others (Supplementary Materials). Unless custom databases are built from the same reference genome source, or multiple tools are used, it is difficult to compare classification strategies.

Comparing short-read discovered and MAG species

The number of unique species discovered across both short-read taxonomic classifiers far outnumbers the number of MAGs in each sample (Fig. 7A versus Table 5). If we consider a more stringent filter and only consider species that have been discovered in ≥ 10 classifier-sample replicates with a minimum relative abundance of at least 0.01, then the number of ‘verified’ species is much closer to the number of MAGs. However, the species discovered in by the short-read classifiers generally do not match the species classified by the MAGs (Table 7). By virtue of how ‘verified’ was defined, the 16 species are common to all 10 samples—6 having been discovered by both Kraken2–Bracken and MetaPhlan4, and 10 having been discovered by only Kraken2–Bracken in all 10 samples. When comparing highly prevalent short-read-discovered species to inferred MAGs, there is in fact more diversity in the num-

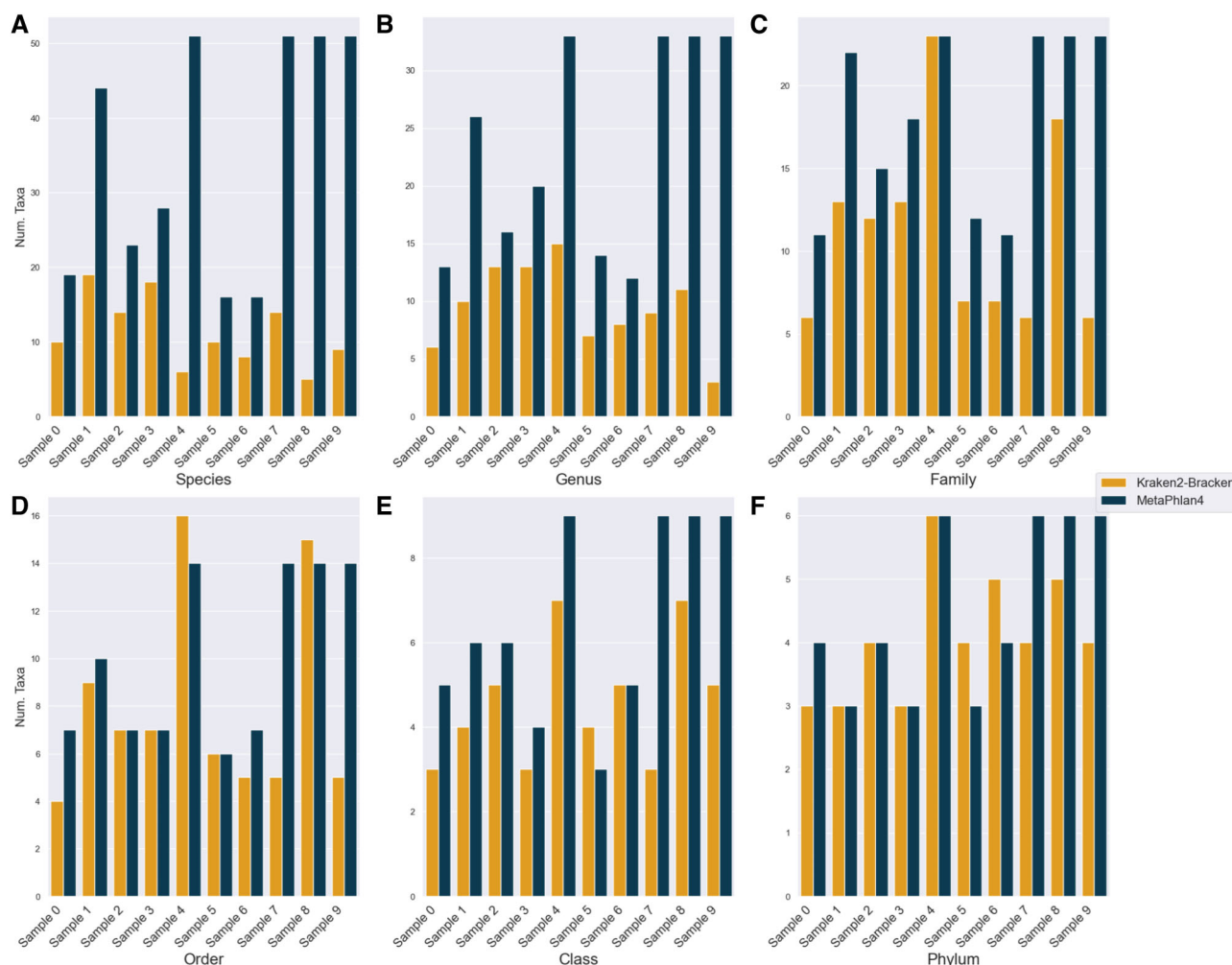


Figure 7. Each classifier detects different numbers of taxa across all ranks. (A) Species. (B) Genus. (C) Family. (D) Order. (E) Class. (F) Phylum.

ber of MAG species discovered across all 10 samples (30) as well as more diversity between samples (Table 7). Of the species in Table 7, there are 9 ‘verified’ short-read and 17 MAG species that do not appear in the list of 75 most abundant taxa (i.e. the ‘core’ and part of the ‘sub-core’ global urban microbiome) in [2]. The most abundant bacterial species from the original study—*C. acnes*—appears in the 3/10 samples (Table 7) [2].

While there are likely more species per sample than binning algorithms are inferring, it is likely that the undiscovered species were not complete or stable enough for DAS Tool to confidently associate sets of contigs with each other.

Virus and phage inference

A small fraction of the total number of contigs are flagged by the virus/phage-inference algorithms as potentially containing viral or phage sequence matter (Supplementary Table S6). However, of those contigs, only around 10% of the flagged contigs are actually classifiable at the species i.e. align to a virus or phage reference genome in RefSeq.

The largest number of virus/phage-flagged contigs was found in sample 7, which had the largest number of reads and assembly in terms of bases (Supplementary Table S6). CheckV quality does not map well to RefSeq classifiability; only 17/26, 65.4% of CheckV-assessed complete viruses were

species-classifiable. At best, the average CheckV-assessed completeness was on average 38.2% in sample 3, which contained 2044 algorithm flagged contigs (the second-most, next to sample 7) (Supplementary Table S7).

There are 332 unique species-classifiable viruses, with the largest number in the sample with the largest number of reads, assembled bases, flagged contigs, and Blast-aligned contigs. Of these, 259/332 (78.0%) are unique to a single sample. The same virus or phage species was typically discovered multiple times in the same sample (Table 8). Taking the low completeness per contig into account, each contig could either be fragments of the same isolate, or there could be viral/phage strain diversity in the sample despite the low (‘verified’) bacterial diversity; 11/332 (3.3%) of species were found in 4+ samples (Fig. 10). Of these, 10/11 (90.9%) aligned to bacteriophage reference genomes.

Comparing virus-phage and MAG classifications

To maintain consistency from the short-read taxonomy section, we selected sample 4 for a case study. When we cross-referenced algorithm-flagged contigs with contig assignments to MAGs, we found that 20/44 (45.5%) candidate virus/phage contigs are found in MAGs. Only 1/44 (2.3%) contigs is labeled as a provirus by CheckV—Bixzunavirus I3. The contig is associated with a MAG, but MAG was classified as *M. luteus*

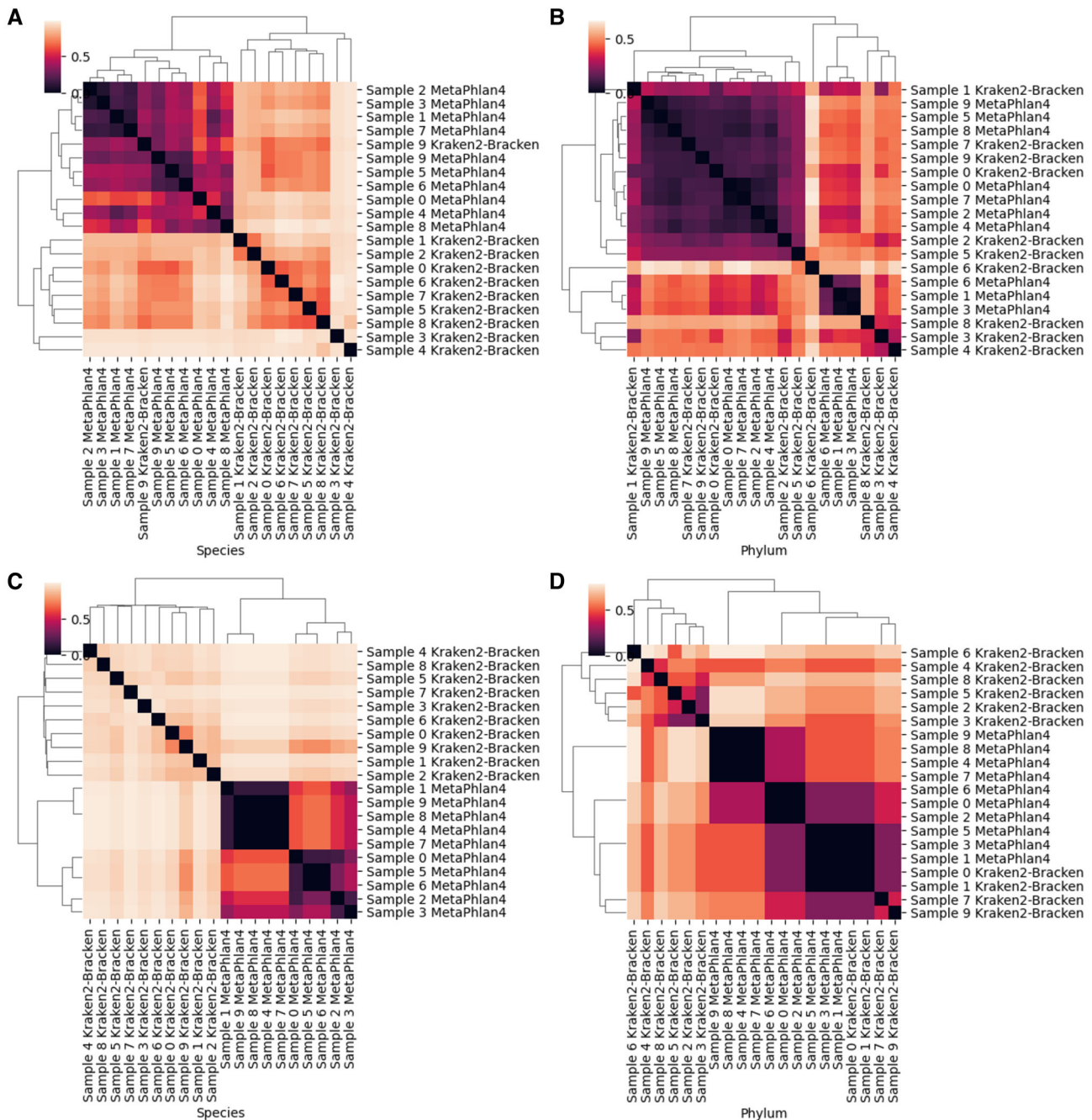


Figure 8. Taxonomic profiles are classifier-specific, not sample-specific, at the taxonomic rank of species but do not present any discernible patterns at the rank of phylum. Bray-Curtis and Jaccard distances between each replicate's species (A, C) and phylum (B, D) profiles.

and is not of the genus *Mycobacterium*, which is the phages' known host [89].

As with the above example, most of the phage-MAG associations are not within the narrow host range of phages. Most phages can only infect several strains of bacteria from the same species, though there are documented cases of phages infecting bacteria from different genera [90]. Of the nonphage contigs (8/44, 18.2%), none are associated with MAGs except 3/4 Pandoraviruses, which are oddly associated with *M. luteus*.

Gene cataloguing

There are 407 098 individual gene annotations found across all contigs across all samples with an average of 3.974 ± 28.98

(range: [1, 3642]). The median number of genes annotated per contig is 1, which agrees with the shortness of most contigs (per-sample averages are 262–930 bp). The vast majority of contigs are extremely short—>99.5% are <2500 bp.

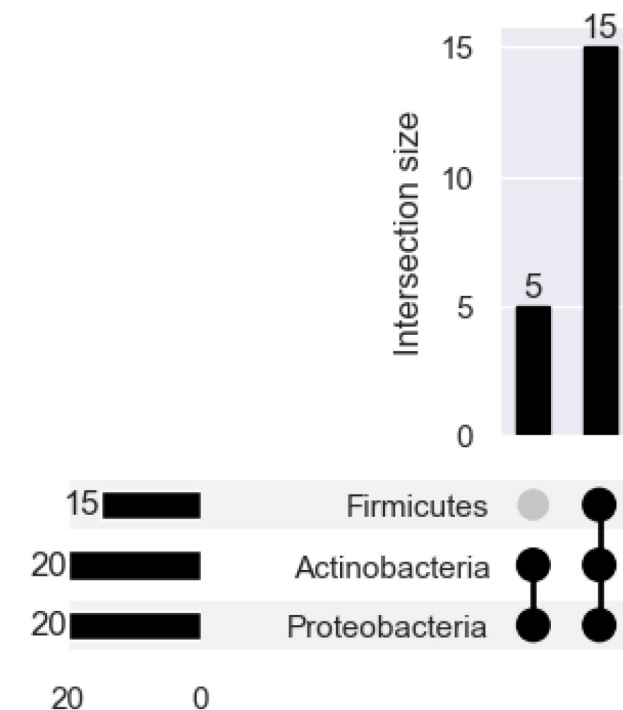
The vast majority of genes cannot be assigned to a cluster of orthologous genes (COG), which are families of orthologous protein-coding genes (Supplementary Table S8). Generally, the larger the dataset and/or assembly, the more COG representatives are found in the assembly. Samples 4, 7, and 8 do not have gene annotations because the associated Bakta jobs did not complete after a week of running the bakta command, likely because the assemblies were too large to completely annotate. Most genes, if they are assigned to a COG, belong to the metabolism category (Fig. 11A). However, the

Table 7. Most ‘verified’ (i.e. discovered in at least 10 classifier-sample replicates, minimum relative abundance of 0.01) short-read species do not overlap with MAG species, and vice versa

Sample	Short-read species	MAG species
Sample 0		<i>Agrobacterium pusense</i> , <i>Corynebacterium variabile</i> , <i>Aerococcus</i> sp002440555
Sample 1		<i>Stenotrophomonas maltophilia</i> N, <i>Psychrobacter</i> sp001652315, <i>Brevibacterium iodinum</i> , <i>Timonella senegalensis</i>
Sample 2		<i>Micrococcus luteus</i> , <i>Epilithonimonas</i> sp002391865, <i>Staphylococcus aureus</i> , <i>Kocuria salsicia</i>
Sample 3		<i>Dysgonomonas</i> sp003523545, <i>Enterococcus faecalis</i> , <i>Leuconostoc mesenteroides</i> , <i>Pseudomonas A stutzeri</i> , <i>Microbacterium</i> sp001878835
Sample 4	<i>A. schindleri</i> , <i>A. lwoffii</i> , <i>A. variabilis</i> , <i>A. radioresistens</i> , <i>M. calida</i> , <i>K. rosea</i> , <i>Weissella confusa</i> , <i>Moraxella osloensis</i> , <i>GGB24656 SGB36527</i> , <i>Kocuria indica</i> , <i>Enterobacter</i> sp DE0047, <i>Pantoea</i> sp PSNIH1, <i>C. acnes</i> , <i>Pseudomonas luteola</i> , <i>Rothia terrae</i> , <i>Staphylococcus hominis</i>	<i>Deinococcus radiopugnans</i> , <i>C. acnes</i> , <i>M. luteus</i>
Sample 5		<i>Microbacterium foliorum</i> A, <i>Corynebacterium ammoniagenes</i> , <i>Carnobacterium A jeotgali</i> , <i>Microbacterium paraoxydans</i> , <i>Rhodococcus erythropolis</i> D
Sample 6		<i>Microbacterium saccharophilum</i> , <i>Epilithonimonas</i> sp002391865, <i>Lactococcus lactis</i> , <i>Pseudomonas E rhodesiae</i> , <i>Dermacoccus nishinomiyaensis</i>
Sample 7		<i>Leuconostoc lactis</i>
Sample 8		<i>C. acnes</i> , <i>Actinomyces oris</i> A
Sample 9		<i>M. calida</i>

Table 8. Each virus or phage species discovered in a sample is represented by on average > 1 contig in that sample

Sample	Num. viruses/phages	Avg. num. contigs
Sample 0	15	1.40
Sample 1	45	1.71
Sample 2	60	1.82
Sample 3	61	3.64
Sample 4	36	1.22
Sample 5	16	1.63
Sample 6	37	2.95
Sample 7	130	3.32
Sample 8	8	1.38
Sample 9	37	2.19

**Figure 9.** There are three phyla present in a majority of all of the 10 samples: Proteobacteria, Actinobacteria, and Firmicutes.

COG with the largest number of genes is that of translation and ribosomal structure (Fig. 11C). There were very few genes annotated in energy production and conversion, extracellular structures, and cytoskeleton COGs (Fig. 11). It is unknown whether urban microbiome assemblies simply contain very few genes of those functions, or are they difficult to assemble or annotate (e.g. not conserved enough or difficult to find via HMM). There were not many cell motility-annotated genes either, which is interesting given one would expect to find many free-living species in the urban milieu.

Computational resources used

If executed as an end-to-end workflow, all modules in our pipeline can be used to efficiently process sequencing data on an HPC cluster. The workflow was designed to optimize resource utilization and minimize processing time (Table 9). Our configuration allowed us to process large-scale metagenomic datasets efficiently, enabling comprehensive analysis of microbial communities across multiple samples. The modular nature of our workflow allows for scalability and adaptability to different computational environments, from local workstations to cloud-based infrastructures.

Discussion

In this paper, we showcase CAMP, a fully modular system comprised of core metagenomic analysis tools, and demonstrated its utility in current and future metagenomic research through a pilot-size case study with real metagenomics data. CAMP is by no means ‘yet another metagenomic pipeline’, but a future-facing infrastructure built with the hope of laying the foundation for future metagenomic research. It is thoughtfully designed with a comprehensive set of tools covering almost all aspects of core metagenomics analyses, from preprocessing to

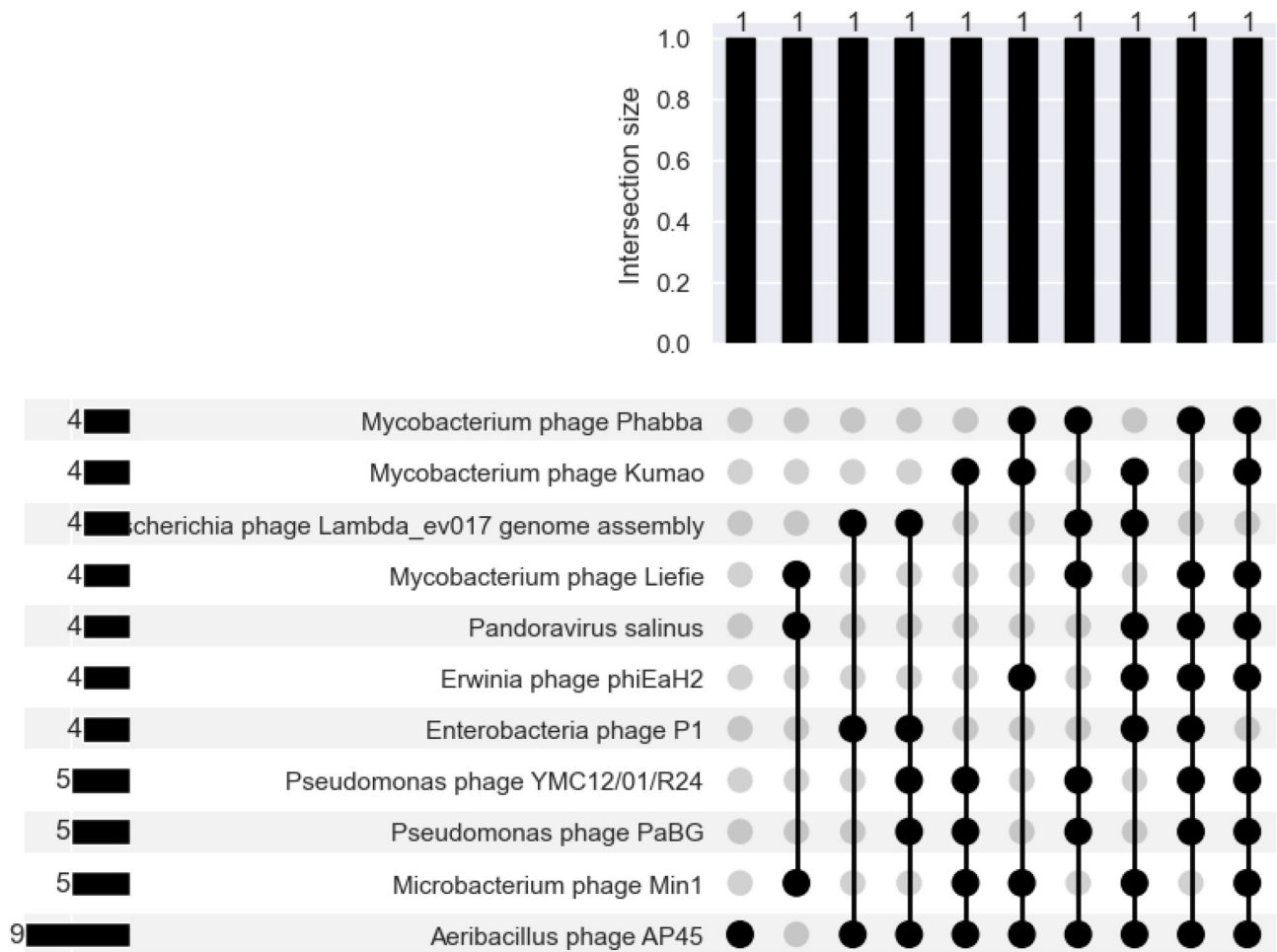


Figure 10. There is more overlap between the virus and/or phage profiles of different samples than there are bacteria.

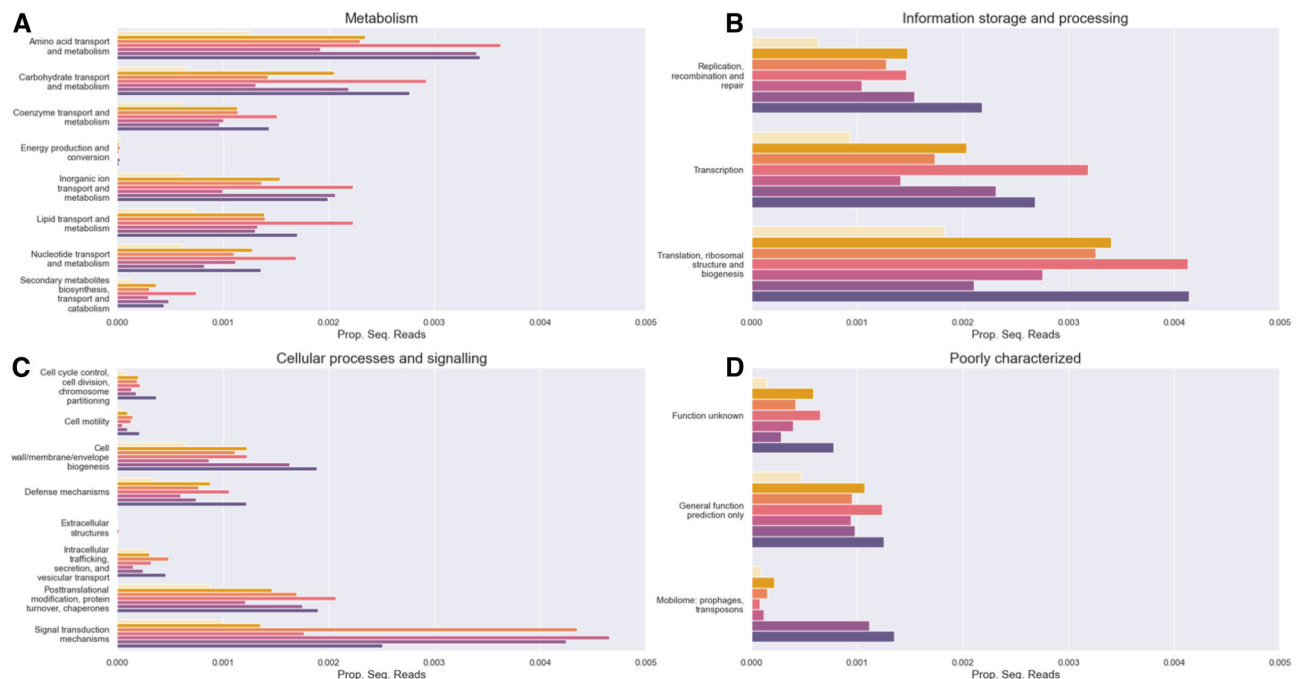


Figure 11. (A) Metabolism, (B) information and storage, (C) cellular process and signaling, and (D) poorly characterized COGs annotated in each sample. The x-axis refers to the proportion of sequencing reads aligned to genes from a COG.

Table 9. Time required to process all 10 samples with 64 CPUs available to the Snakemake command

Module	Time (dd:hh:mm:ss)
Short-read preprocessing	00:00:34:03
Short-read assembly	00:08:14:05
MAG binning	00:01:45:12
MAG quality-checking	00:01:56:50
Short-read taxonomy	05:22:35:42
Virus-phage inference	03:17:31:18
Gene cataloguing	03:15:08:22

functional inference, with the support for both short and long reads. To the best of our knowledge, CAMP is the most modular and functionally comprehensive metagenomic analysis system available to date, allowing researchers the maximum flexibility to design customized workflows catering to their specific research needs. In addition to user experience, CAMP has been designed with developer accessibility in mind, with a standard module structure setup for easy development of new modules using the templates provided.

While other pipelines may be sufficient for small-scale projects, CAMP is irreplaceable in many potential large-scale exploratory future projects. For example, the modular sandbox design makes CAMP an ideal candidate for module-wise tool benchmarking. Considering the amount of metagenomic analysis tools available and the variability of results produced by different tools, the amount of literature on metagenomic tool benchmarking is surprisingly low. This can be attributed to a combination of factors such as the lack of gold-standard datasets and the computational load. Despite being a current challenge, tool benchmarking is indispensable and unavoidable in the future as better quality datasets continue to emerge as a result of advancement in biotechnology and our understanding. With its modularity and HPC-compatibility, CAMP is prepared to address such needs whenever they arise in the future. Its modular design with standardized input and output formats allow convenient distribution of workloads and reproducibility of results in large-scale collaborative efforts. In addition to the modules mentioned in this paper, there are many more CAMP modules currently under active development, including long-read taxonomic profiling and decontamination. As our next milestone, we will further expand the utilities of CAMP to include multisample co-assembly and co-binning, variant and strain analysis, and metatranscriptomics. To further enhance reproducibility and interoperability for users with root access, we also plan to offer a containerized implementation of CAMP using Singularity.

We note that there is not a single tool or toolkit that is sufficient to meet every single need. With greater flexibility of CAMP comes greater responsibility on the side of the researchers. With less hard-coded defaults in parameters, CAMP places more weight on the expertise and discretion of the researchers in experiment design and parameter selection. The soft pauses and semiguided visualizations at the end of each module also provide users the opportunity to spot any potential inappropriate protocols and make modifications promptly.

We envision CAMP not to be another short-lived pipeline, but a sustainable ecosystem of high-quality toolkit for all core aspects of metagenomic analysis that will be consistently expanded, maintained, and fortified to lay a solid foundation for future metagenomic research.

Acknowledgements

We thank Haruo Suzuki and Yinka Osuolale for their copy-editing suggestions. R.B.T. and P.L. gratefully acknowledge Poland's high-performance Infrastructure PLGrid (HPC Centers: ACK Cyfronet AGH, PCSS, CI TASK, WCSS) for providing computing facilities.

Author contributions: L.M. and B.T. jointly conceived of the CAMP modular analysis concept, with design contributions from D.D. L.M., B.T., I.H., and C.M. jointly managed project development. L.M. developed the modular template and short-read assembly module, with contributions from B.T., C.R., and C.F. L.M., B.T., and C.F. developed the short-read quality-control module with contributions from C.R. L.M. developed the MAG binning and quality-control modules. B.T., L.M., and C.F. developed the short-read taxonomy module with contributions from S.Z. and J.H. B.T. and L.M. developed the gene cataloguing module. R.B.T. developed the viral inference module with contributions from J.S.A. and L.M. M.T. and B. Turhan developed the Nanopore long-read quality control module with contributions from L.M. and D.M. C.R. developed the decontamination module with contributions from M.K., A.N., Z.I., S.K., and N.N. D.H., V.K., A.F., and A.K. developed the mOTUs module. W.W. implemented parts of all modules. L.M. did base-case testing for all of the modules. A.H.S., J.C.S., and A.G.L. contributed to the debugging process. L.M. wrote the manuscript with suggestions from all of the other authors. W.W., A.G.L., and E.E. contributed to the editing process. B.T. created the ReadTheDocs.

Supplementary data

[Supplementary data](#) is available at NAR Genomics & Bioinformatics online.

Conflict of interest

None declared.

Funding

This work was supported by an National Institute of General Medical Sciences Maximizing Investigators' Research Award (MIRA, grant number R35 GM138152-01 to I.H.). I.H. was also supported by the Irma T. Hirsch Monique Weill-Caulier Research Award. C.E.M. was supported by the Katherine and John Bleckman Foundation, the National Aeronautics and Space Administration (80NSSC24K0728), and the National Institutes of Health (R01AI151059, U54AG089334). L.M., B. Turhan, M.T., J.S.A., D.M., and C.F. were also supported by the Tri-Institutional Training Program in Computational Biology and Medicine (CBM) funded by the National Institutes of Health grant 1T32GM083937. R.B.T. was funded by the Narodowe Centrum Nauki Sonata BIS (grant no. 2020/38/E/NZ2/00598). S.M. was supported by a grant of the Ministry of Research, Innovation and Digitization, under the Romania's National Recovery and Resilience Plan – Funded by EU – NextGenerationEU program, project "Metagenomics and Bioinformatics tools for Wastewater-based Genomic Surveillance of viral Pathogens for early prediction of public health risks - (MetBio-WGSP)" number 760286/27.03.2024, code 167/31.07.2023, within Pillar III, Component C9, Investment 8.

Data availability

The raw sequencing data can be found under the SRA accession ID [PRJNA732392](https://www.ncbi.nlm.nih.gov/sra/PRJNA732392), as well as the GeoSeq repository.

References

- Almeida A, Mitchell AL, Boland M *et al.* A new genomic blueprint of the human gut microbiota. *Nature* 2019;568:499–504. <https://doi.org/10.1038/s41586-019-0965-1>
- Danko D, Bezdan D, Afshin EE *et al.* A global metagenomic map of urban microbiomes and antimicrobial resistance. *Cell* 2021;184:3376–93. <https://doi.org/10.1016/j.cell.2021.05.002>
- Kadosh E, Snir-Alkalay I, Venkatachalam A *et al.* The gut microbiome switches mutant p53 from tumour-suppressive to oncogenic. *Nature* 2020;586:133–8. <https://doi.org/10.1038/s41586-020-2541-0>
- Brito IL, Gurry T, Zhao S *et al.* Transmission of human-associated microbiota along family and social networks. *Nat Microbiol* 2019;4:964–71. <https://doi.org/10.1101/540252>
- Sierra MA, Ryon KA, Tierney BT *et al.* Microbiome and metagenomic analysis of Lake Hillier Australia reveals pigment-rich polyextremophiles and wide-ranging metabolic adaptations. *Environ Microbiol* 2022;17:60. <https://doi.org/10.1186/s40793-022-00455-9>
- Bahram M, Netherway T, Frioux C *et al.* Metagenomic assessment of the global diversity and distribution of bacteria and fungi. *Environ Microbiol* 2021;23:316–26.
- Hug LA, Baker BJ, Anantharaman K *et al.* A new view of the tree of life. *Nat Microbiol* 2016;1:16048.
- Gevers D, Knight R, Petrosino JF *et al.* The Human Microbiome Project: a community resource for the healthy human microbiome. *PLoS Biol* 2012;10:e1001377.
- Huttenhower C, Gevers D, Knight R *et al.* Structure, function and diversity of the healthy human microbiome. *Nature* 2012;486:207–14. <https://doi.org/10.1038/nature11234>
- Djaffardjy M, Marchment G, Sebe C *et al.* Developing and reusing bioinformatics data analysis pipelines using scientific workflow systems. *Comput Struct Biotechnol J* 2023;21:2075–85. <https://doi.org/10.1016/j.csbj.2023.03.003>
- 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>
- Mangul S, Martin LS, Eskin E *et al.* Improving the usability and archival stability of bioinformatics software. *Genome Biol* 2019;20:47. <https://doi.org/10.1186/s13059-019-1649-8>
- Kern F, Fehlmann T, Keller A. On the lifetime of bioinformatics web services. *Nucleic Acids Res* 2020;48:12523–33. <https://doi.org/10.1093/nar/gkaa1125>
- Merkel D. Docker: lightweight linux containers for consistent development and deployment. *Linux J* 2014;2014:2.
- Kurtzer GM. Singularity 2.5.2—linux application and environment containers for science. 2018. <https://zenodo.org/record/1308868> (27 October 2025, date last accessed)
- Kieser S, Brown J, Zdobnov EM *et al.* Atlas: A snakemake workflow for assembly, annotation, and genomic binning of Metagenome Sequence Data. *BMC Bioinformatics* 2019;21:257. <https://doi.org/10.1101/737528>
- Saheb Kashaf S, Almeida A, Segre JA *et al.* Recovering prokaryotic genomes from host-associated, short-read shotgun metagenomic sequencing data. *Nature Protocols* 2021;16:2520–41. <https://doi.org/10.1038/s41596-021-00508-2>
- Van Damme R, Hölzer M, Viehweger A *et al.* Metagenomics workflow for hybrid assembly, differential coverage binning, metatranscriptomics and pathway analysis (Muffin). *PLoS Comput Biol* 2021;17:e1008716. <https://doi.org/10.1371/journal.pcbi.1008716>
- Krakau S, Straub D, Gourel H *et al.* NF-core/MAG: a best-practice pipeline for metagenome hybrid assembly and binning. *NAR Genom Bioinform* 2021; 4:1–6. <https://doi.org/10.1101/2021.08.29.458094>
- (NBIS) National Bioinformatics Infrastructure Sweden. NBIS-meta: Snakemake workflow for metagenomic assembly, binning, and analysis. 2022. <https://github.com/NBISweden/nbis-meta> (27 October 2025, date last accessed).
- Chen LX, Anantharaman K, Shaiber A *et al.* Accurate and complete genomes from metagenomes. *Genome Res* 2019;30:315–33. <https://doi.org/10.1101/808410>
- Moss EL, Maghini DG, Bhatt AS. Complete, closed bacterial genomes from microbiomes using nanopore sequencing. *Nat Biotechnol* 2020;38:701–7. <https://doi.org/10.1038/s41587-020-0422-6>
- Sczyrba A, Hofmann P, Belmann P *et al.* Critical assessment of metagenome interpretation—a benchmark of Metagenomics software. *Nat Methods* 2017;14:1063–71. <https://doi.org/10.1038/nmeth.4458>
- Orjuela J, Comte A, Ravel S *et al.* Culebront: a streamlined long reads multi-assembler pipeline for Prokaryotic and eukaryotic genomes. *Peer Commun J* 2022;2:e46. <https://doi.org/10.24072/pcjournal.153>
- Cookiecutter. Cookiecutter/Cookiecutter: a cross-platform command-line utility that creates projects from cookiecutters (project templates). 2022. <https://github.com/cookiecutter/cookiecutter> (27 October 2025, date last accessed).
- Di Tommaso P, Chatzou M, Floden EW *et al.* Nextflow enables reproducible computational workflows. *Nat Biotechnol* 2017;35:316–9. <https://doi.org/10.1038/nbt.3820>
- Mölder F, Jablonski KP, Letcher B *et al.* Sustainable data analysis with snakemake. *F1000Research* 2021;10:33.
- Ewels PA, Peltzer A, Fillinger S *et al.* The nf-core framework for community-curated bioinformatics pipelines. *Nat Biotechnol* 2020;38:276–8. <https://doi.org/10.1038/s41587-020-0439-x>
- Conda contributors. conda: a system-level, binary package and environment manager running on all major operating systems and platforms. <https://github.com/conda/conda> (27 October 2025, date last accessed).
- Chen S, Zhou Y, Chen Y *et al.* Fastp: An ultra-fast all-in-one FASTQ preprocessor. *Bioinformatics* 2018;34:i884–90. <https://doi.org/10.1093/bioinformatics/bty560>
- Langmead B, Salzberg SL. Fast gapped-read alignment with Bowtie 2. *Nat Methods* 2012;9:357–9. <https://doi.org/10.1038/nmeth.1923>
- Li H, Handsaker B, Wysoker A *et al.* The Sequence Alignment/Map format and SAMtools. *Bioinformatics* 2009;25:2078–9. <https://doi.org/10.1093/bioinformatics/btp352>
- Nikolenko SI, Korobeynikov AI, Alekseyev MA. Bayeshammer: Bayesian clustering for error correction in single-cell sequencing. *BMC Genomics* 2013;14 (Suppl 1):S7. <https://doi.org/10.1186/1471-2164-14-s1-s7>
- Bushnell B. BBMap: a fast, accurate, splice-aware aligner. *Lawrence Berkeley National Laboratory* 2014. <https://escholarship.org/uc/item/1h3515gn> (27 October 2025, date last accessed).
- Andrews S. A quality control tool for high throughput sequence data. 2010. <http://www.bioinformatics.babraham.ac.uk/projects/fastqc/> (27 October 2025, date last accessed).
- Ewels P, Magnusson M, Lundin S *et al.* MultiQC: summarize analysis results for multiple tools and samples in a single report. *Bioinformatics* 2016;32:3047–8. <https://doi.org/10.1093/bioinformatics/btw354>
- Wick RR. Porechop. 2018. <https://github.com/rrwick/Porechop> (27 October 2025, date last accessed).

38. Coster WD, Rademakers R. Sequence analysis NanoPack2: population-scale evaluation of long-read sequencing data. *Bioinformatics* 2023;39:btad311.
39. Li H. Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics* 2018;34:3094–100. <https://doi.org/10.1093/bioinformatics/bty191>
40. Nurk S, Meleshko D, Korobeynikov A *et al.* MetaSPAdes: a new versatile metagenomic assembler. *Genome Res* 2017;27:824–34. <https://doi.org/10.1101/gr.213959.116>
41. Li D, Liu CM, Luo R *et al.* MEGAHIT: an ultra-fast single-node solution for large and complex metagenomics assembly via succinct de Bruijn graph. *Bioinformatics* 2015;31:1674–6. <https://doi.org/10.1093/bioinformatics/btv033>
42. Gurevich A, Saveliev V, Vyahhi N *et al.* QUAST: quality assessment tool for genome assemblies. *Bioinformatics* 2013;29:1072–5. <https://doi.org/10.1093/bioinformatics/btt086>
43. Antipov D, Korobeynikov A, McLean JS *et al.* HybridSPAdes: an algorithm for hybrid assembly of short and long reads. *Bioinformatics* 2016;32:1009–15. <https://doi.org/10.1093/bioinformatics/btv688>
44. Kolmogorov M, Bickhart DM, Behsaz B *et al.* metaFlye: scalable long-read metagenome assembly using repeat graphs. *Nat Methods* 2020;17:1103–10. <https://doi.org/10.1038/s41592-020-00971-x>
45. Oxford Nanopore Technologies Ltd. Medaka. 2019. <https://github.com/oxfordnanoporetech/medaka> (27 October 2025, date last accessed).
46. Bouras G, Judd LM, Edwards RA *et al.* How low can you go? Short-read polishing of Oxford Nanopore bacterial genome assemblies. *Microbial Genom* 2024;10:1–9. <https://doi.org/10.1099/mgen.0.001254>
47. Kang D, Li F, Kirton ES *et al.* MetaBAT 2: An adaptive binning algorithm for robust and efficient genome reconstruction from Metagenome Assemblies. *PeerJ* 2019;7:e7359. <https://doi.org/10.7287/peerj.preprints.27522v1>
48. Alneberg J, Bjarnason BS, De Bruijn I *et al.* Binning metagenomic contigs by coverage and composition. *Nat Methods* 2014;11:1144–6. <https://doi.org/10.1038/nmeth.3103>
49. Pan S, Zhu C, Zhao XM *et al.* A deep siamese neural network improves metagenome-assembled genomes in microbiome datasets across different environments. *Nat Commun* 2022;13:1–12. <https://doi.org/10.1038/s41467-022-29843-y>
50. Nissen JN, Johansen J, Allesøe RL *et al.* Improved metagenome binning and assembly using deep variational autoencoders. *Nat Biotechnol* 2021;39:555–60. <https://doi.org/10.1038/s41587-020-00777-4>
51. Wu YW, Simmons BA, Singer SW. MaxBin 2.0: an automated binning algorithm to recover genomes from multiple metagenomic datasets. *Bioinformatics* 2016;32:605–07. <https://doi.org/10.1093/bioinformatics/btv638>
52. Wang Z, Huang P, You R *et al.* MetaBin: a high-performance and stand-alone ensemble binning method to recover individual genomes from complex microbial communities. *Genome Biol* 2023;24:1. <https://doi.org/10.1186/s13059-022-02832-6>
53. Sieber CM, Probst AJ, Sharrar A *et al.* Recovery of genomes from metagenomes via a dereplication, aggregation, and scoring strategy. *Nat Microbiol* 2017;3:836–43. <https://doi.org/10.1101/107789>
54. Chklovski A, Parks DH, Woodcroft BJ *et al.* CheckM2: a rapid, scalable and accurate tool for assessing microbial genome quality using machine learning. *Nat Methods* 2023;20:1203–12. <https://doi.org/10.1038/s41592-023-01940-w>
55. Parks DH, Imelfort M, Skennerton CT *et al.* CheckM: assessing the quality of microbial genomes recovered from isolates, single cells, and metagenomes. *Genome Res* 2015;25:1043–55. <https://doi.org/10.1101/gr.186072.114>
56. Orakov A, Fullam A, Coelho LP *et al.* GUNC: detection of chimerism and contamination in prokaryotic genomes. *Genome Biol* 2021;22:178. <https://doi.org/10.1186/s13059-021-02393-0>
57. Chaumeil PA, Mussig AJ, Hugenholtz P *et al.* GTDB-TK: a toolkit to classify genomes with the Genome Taxonomy Database. *Bioinformatics* 2019;36:1925–7. <https://doi.org/10.1093/bioinformatics/btz848>
58. Marçais G, Delcher AL, Phillippy AM *et al.* MUMmer4: a fast and versatile genome alignment system. *PLoS Comput Biol* 2018;14:e1005944. <https://doi.org/10.1371/journal.pcbi.1005944>
59. Bowers RM, Kyrpides NC, Stepanauskas R *et al.* Minimum information about a single amplified genome (MISAG) and a metagenome-assembled genome (MIMAG) of bacteria and Archaea. *Nat Biotechnol* 2017;35:725–31. <https://doi.org/10.1038/nbt.3893>
60. Seemann T. Prokka: rapid prokaryotic genome annotation. *Bioinformatics* 2014;30:2068–9. <https://doi.org/10.1093/bioinformatics/btu153>
61. Olm MR, Brown CT, Brooks B *et al.* dRep: a tool for fast and accurate genomic comparisons that enables improved genome recovery from metagenomes through de-replication. *ISME J* 2017;11:2864–8.
62. Williams TJ, Allen MA, Ray AE *et al.* Novel endolithic bacteria of phylum Chloroflexota reveal a myriad of potential survival strategies in the Antarctic desert. *Appl Environ Microbiol* 2024;90:e02264–23. <https://doi.org/10.1128/aem.02264-23>
63. Gurbich TA, Almeida A, Beracochea M *et al.* MGnify Genomes: a resource for biome-specific microbial genome catalogues. *J Mol Biol* 2023;435:168016. <https://doi.org/10.1016/j.jmb.2023.168016>
64. Parks DH, Rinke C, Chuvochina M *et al.* Recovery of nearly 8,000 metagenome-assembled genomes substantially expands the tree of life. *Nat Microbiol* 2017;2:1533–42. <https://doi.org/10.1038/s41564-017-0012-7>
65. Wood DE, Lu J, Langmead B. Improved metagenomic analysis with Kraken 2. *Genome Biol* 2019;20:257. <https://doi.org/10.1101/762302>
66. Lu J, Breitwieser FP, Thielen P *et al.* Bracken: estimating species abundance in metagenomics data. *PeerJ Comput Sci* 2017;3:e104. <https://doi.org/10.7717/peerj-cs.104>
67. Blanco-Miguez A, Beghini F, Cumbo F *et al.* Extending and improving metagenomic taxonomic profiling with uncharacterized species with MetaPhlAn 4. *Nat Biotechnol* 2022;41:1633–44. <https://doi.org/10.1101/2022.08.22.504593>
68. Gabeal/Utree: K-Mer searching with trees. 2022. <https://github.com/Gabeal/UTree> (27 October 2025, date last accessed).
69. Antipov D, Raiko M, Lapidus A *et al.* Metaviral SPAdes: assembly of viruses from metagenomic data. *Bioinformatics* 2020;36:4126–9. <https://doi.org/10.1093/bioinformatics/btaa490>
70. Kieft K, Zhou Z, Anantharaman K. VIBRANT: automated recovery, annotation and curation of microbial viruses, and evaluation of viral community function from genomic sequences. *Microbiome* 2020;8:90. <https://doi.org/10.1186/s40168-020-00867-0>
71. Guo J, Bolduc B, Zayed AA *et al.* VirSorter2: a multi-classifier, expert-guided approach to detect diverse DNA and RNA viruses. *Microbiome* 2021;9:37. <https://doi.org/10.1186/s40168-020-00990-y>
72. Ren J, Song K, Deng C *et al.* Identifying viruses from metagenomic data using deep learning. *Quant Biol* 2020;8:64–77. <https://doi.org/10.1007/s40484-019-0187-4>
73. Camargo AP, Roux S, Schulz F *et al.* Identification of mobile genetic elements with geNomad. *Nat Biotechnol* 2024;42:1303–12. <https://doi.org/10.1038/s41587-023-01953-y>
74. Moraru C. VirClust—a tool for hierarchical clustering, core protein detection and annotation of (Prokaryotic) viruses. *Viruses* 2023;15:1007. <https://doi.org/10.3390/v15041007>
75. Nayfach S, Camargo AP, Schulz F *et al.* CheckV assesses the quality and completeness of metagenome-assembled viral genomes. *Nat Biotechnol* 2021;39:578–85. <https://doi.org/10.1038/s41587-020-00774-7>

76. Schwengers O, Jelonek L, Dieckmann MA *et al.* Bakta: rapid and standardized annotation of bacterial genomes via alignment-free sequence identification. *Microbial Genom* 2021;7:000685. <https://doi.org/10.1099/MGEN.0.000685>
77. Hauser M, Steinegger M, Söding J. MMseqs software suite for fast and deep clustering and searching of large protein sequence sets. *Bioinformatics* 2016;32:1323–30. <https://doi.org/10.1093/bioinformatics/btw006>
78. Buchfink B, Xie C, Huson DH. Fast and sensitive protein alignment using DIAMOND. *Nat Methods* 2014;12:59–60. <https://doi.org/10.1038/nmeth.3176>
79. Davis NM, Proctor DM, Holmes SP *et al.* Simple statistical identification and removal of contaminant sequences in marker-gene and metagenomics data. *Microbiome* 2018;6:226. <https://doi.org/10.1186/s40168-018-0605-2>
80. Martí JM. Recentrifuge: robust comparative analysis and contamination removal for metagenomics. *PLoS Comput Biol* 2019;15:e1006967. <https://doi.org/10.1371/journal.pcbi.1006967>
81. Ruscheweyh HJ, Milanese A, Paoli L *et al.* mOTUs: profiling taxonomic composition, transcriptional activity and strain populations of microbial communities. *Curr Protoc* 2021;1:e218.
82. Hansen KD, Brenner SE, Dudoit S. Biases in Illumina transcriptome sequencing caused by random hexamer priming. *Nucleic Acids Res* 2010;38:e131. <https://doi.org/10.1093/nar/gkq224>
83. Hofmeyr S, Egan R, Georganas E *et al.* Terabase-scale metagenome coassembly with metahipmer. *Sci Rep* 2020;10:10689. <https://doi.org/10.1038/s41598-020-67416-5>
84. Vicedomini R, Quince C, Darling AE *et al.* Strawberry: automated strain separation in low-complexity metagenomes using long reads. *Nat Commun* 2021;12:4485. <https://doi.org/10.1038/s41467-021-24515-9>
85. diCenzo GC, Finan TM. The divided bacterial genome: structure, function, and evolution. *Microbiol Mol Biol Rev* 2017;81:e00019–17. <https://doi.org/10.1128/MMBR.00019-17>
86. von Meijenfödt FAB, Arkhipova K, Cambuy DD *et al.* Robust taxonomic classification of uncharted microbial sequences and bins with CAT and BAT. *Genome Biol* 2019;20:217.
87. Hill TC, Walsh KA, Harris JA *et al.* Using ecological diversity measures with bacterial communities. *FEMS Microbiol Ecol* 2003;43:1–11.
88. Prost V, Gazut S, Bröls T. A zero inflated log-normal model for inference of sparse microbial association networks. *PLoS Comput Biol* 2021;17:e1009089.
89. Reddy AB, Gopinathan KP. Characterization of genomic DNA of mycobacteriophage I3. *Curr Microbiol* 1987;14:241–5.
90. Ross A, Ward S, Hyman P. More is better: selecting for broad host range bacteriophages. *Front Microbiol* 2016;7:1352.