The Journal of Open Source Software

KrakenParser: Efficient Post-processing of Kraken2-like Taxonomic Reports

Ilia V. Popov [1]

1 Faculty of Bioengineering and Veterinary Medicine, Don State Technical University, Russia

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Claudia Solis-Lemus

Reviewers:

- @johanneswerner

Submitted: 01 June 2025
Published: unpublished

Summary

KrakenParser is an open-source software tool (with a command-line interface and Python API) designed to streamline the post-analysis of metagenomic classification results produced by Kraken2 (Wood et al., 2019) and similar taxonomic profilers such as Bracken (Lu et al., 2017) and Metabuli (J. Kim & Steinegger, 2024). Kraken2 is a widely used taxonomic classifier that assigns metagenomic reads to taxa using exact k-mer matches, achieving high speed and accuracy. However, the raw output of Kraken2 (Wood et al., 2019) (and related tools) is a text report that can be cumbersome to interpret and aggregate across multiple samples. KrakenParser addresses this need by converting multiple Kraken-format reports into structured tables (CSV files) at various taxonomic ranks (from phylum down to species), performing filtering and normalization (including relative abundance calculations), and providing APIs to produce publication-ready plots. The tool automates the multi-step process of combining and cleaning Kraken results, allowing researchers to quickly obtain human-readable summaries of community composition. KrakenParser's focus is on efficiency, ease-of-use, and integration: it can run an entire conversion pipeline with a single command and also be imported as a Python library for custom workflows. In summary, KrakenParser significantly reduces the manual effort required to post-process metagenomic classification data, enabling scientists to go from raw classifier output to analysis-ready tables and figures in one step.

Statement of need

Analyzing the taxonomic profiles of metagenomic samples often involves running k-mer based classifiers (like Kraken2) that generate detailed reports of read counts and abundances across taxa. These reports, while information-rich, are not immediately convenient for comparative analysis: they list each taxon in a hierarchical format for a single sample, and researchers must manually parse and merge multiple files to compare communities across samples. Existing scripts such as the KrakenTools suite (Lu et al., 2022) (developed alongside Kraken) provide some post-processing functionality, but they require multiple steps and technical expertise to use. Similarly, interactive tools like Pavian focus on visualization and exploration of Kraken results rather than automated batch processing (Breitwieser & Salzberg, 2020). There is a clear need for a streamlined solution to transform raw Kraken-family outputs into tidy data matrices and summary statistics that can be readily used in downstream analysis or publication figures. KrakenParser fulfills this need by offering an all-in-one pipeline that reads in multiple Kraken2/Bracken/Metabuli reports and outputs clean CSV tables of taxonomic counts or relative abundances, optionally filtering out low-abundance taxa or non-target taxa (e.g. human reads) as specified by the user. This greatly simplifies metagenomic workflows, especially in comparative studies or clinical settings where dozens of samples must be processed consistently. By bridging the gap between raw classifier output and statistical analysis, KrakenParser empowers researchers who may not be bioinformatics experts to leverage high-throughput metagenomics with minimal data wrangling.

The Journal of Open Source Software

Metagenomic classification has seen rapid development, with numerous tools available for assigning sequencing reads to taxa. Kraken was introduced in 2014 as an ultrafast k-mer based classifier (Wood & Salzberg, 2014), and its successor Kraken2 (Wood et al., 2019) further reduced memory usage and improved speed . Other k-mer classifiers include Bracken (Lu et al., 2017), which refines Kraken's counts to improve abundance estimates, KrakenUniq which tracks unique k-mers per taxon to reduce false positives (Breitwieser et al., 2018), Centrifuge which uses an FM-index to allow classification with compressed databases (D. Kim et al., 2016), and CLARK which uses discriminative k-mers for fast classification (Ounit et al., 2015). More recently, tools like Kaiju perform classification in protein space for greater sensitivity (especially on viruses) (Menzel et al., 2016), and Metabuli combines DNA and translated amino acid matching to improve accuracy (J. Kim & Steinegger, 2024). Comprehensive evaluations have benchmarked these methods' accuracy and speed, and community challenges like CAMI have pushed development of improved classifiers (Sczyrba et al., 2017). Despite the variety of classifiers, a common challenge remains: the output format. Many tools output reports similar to Kraken's: tab-delimited text with hierarchical labels and counts. To interpret such outputs, researchers often rely on additional scripts or manual processing. KrakenTools (Lu et al., 2022) provides scripts to combine Kraken reports, convert to other formats (e.g., Krona for visualization). Pavian and other interactive platforms allow users to visualize results with Sankey diagrams and heatmaps (Breitwieser & Salzberg, 2020), but require use of a web interface or R environment. There are also lightweight utilities (e.g., spideog) to convert Kraken reports to CSV or clean them, and researchers adept in programming sometimes write custom parsing scripts. In summary, prior to KrakenParser, users had to piece together multiple tools to achieve tasks like merging reports from multiple samples, summing reads at specific taxonomic ranks, and computing relative abundances. KrakenParser builds on this state of the field by consolidating the post-processing steps into one tool. It serves as an ideological successor to KrakenTools (Lu et al., 2022), using some of the same internal conversion steps (like KrakenTools' report-to-MPA conversion) but adding improvements in automation, filtering, and output formatting. By producing standardized CSV tables (with samples as rows and taxa as columns) and by computing percentages automatically, KrakenParser greatly accelerates the transition from raw classification data to biological insights. This is particularly valuable given the increasing scale of metagenomic studies (where dozens or hundreds of samples are profiled) and the need for reproducible, efficient analysis pipelines.

Implementation

KrakenParser is implemented in Python (available via PyPI as krakenparser) with several auxiliary scripts. It leverages the original KrakenTools (Lu et al., 2022) scripts for initial data reshaping and then applies its own pure-Python processing for downstream formatting. The software follows a pipeline of six main steps, which can be executed automatically in sequence (`--complete` mode) or run individually as needed:

1. Convert reports to MPA format: Each Kraken2/Bracken/Metabuli report (text file with taxon lines) is converted to an "MPA" table format using KrakenTools' `kreport2mpa.py` script. In MPA format, each row corresponds to a read and columns correspond to taxonomic ranks, allowing easy combination of multiple samples.
2. Combine MPA files: All per-sample MPA files are merged into a single master table (samples × taxa) using KrakenTools' combine_mpa.py. This yields a matrix of raw read counts, with entries where a taxon is absent in a sample filled with zero.
3. Deconstruct taxonomic levels: The combined data is split out by rank. KrakenParser extracts separate text files for phylum, class, order, family, genus, and species counts. During this step, it can optionally isolate certain domains; for example, using `--deconstruct_viruses` will produce a file of only viral species counts, ignoring other domains. Also, the default `--deconstruct` excludes reads classified as human to focus on microbial content.
4. Process extracted data: Each rank-specific text file is cleaned and formatted.

KrakenParser removes classification prefixes (like "s__" for species, "g__" for genus) and replaces underscores with spaces for readability. This step ensures taxon names are human-friendly (e.g. "s__Escherichia_coli" becomes "Escherichia coli").

5. Convert to CSV: The cleaned text tables are converted to CSV files (comma-separated values). In this transpose operation, taxa become columns and sample identifiers become rows, yielding a standard matrix format. This structured CSV is easy to import into statistical software, spreadsheets, or R/Python data frames for further analysis.
6. Calculate relative abundances: For each count table, KrakenParser can create a corresponding relative abundance table (`--relabund` option) by computing percentages of total reads per sample, using the formula: $\text{Relative Abundance} = \left(\frac{\text{Number of individuals of taxa}}{\text{Total number of individuals of all taxa}}\right) \times 100$. Users can specify a threshold to group low-abundance taxa into an "Other" category. This results in a normalized profile for each sample, often more interpretable in comparative studies than raw counts.

Each of these steps is exposed as a sub-command in the CLI, so advanced users can integrate KrakenParser into custom workflows. By default, running `KrakenParser --complete -i <reports_dir>/kreports` executes all steps sequentially, writing outputs to a structured directory tree (with subfolders for each step). The outputs include one CSV file per rank (e.g. counts_phylum.csv, counts_species.csv) containing absolute read counts, and similarly named files under a `csv_relabund/` directory for percentages if requested. KrakenParser is optimized for speed and memory efficiency given the nature of the task: it processes text files line by line and uses `pandas` data frames for merging and calculations, which easily handle dozens of samples and tens of thousands of taxa on a standard workstation. The reliance on `KrakenTools` for the initial conversion ensures that the parsing logic benefits from the robustness of well-tested scripts, while the unified interface adds convenience. The tool also includes built-in help for each subcommand (`-h`), guiding users on required inputs and options. KrakenParser's design reflects practical needs observed in the metagenomics community - it was tested during the 2025 "Bioinformatics Bootcamp" hackathon organized by ITMO University, where teams analyzing metagenomic datasets were able to obtain meaningful results in a short time thanks to KrakenParser's streamlined processing pipeline. By combining established methods with new automation, KrakenParser provides an efficient, reproducible, and user-friendly means to handle the otherwise tedious steps of post-classification data processing.

KrakenParser also offers a suite of Python-based visualization tools to facilitate the interpretation of taxonomic profiles:

- Stacked Bar Plots: Utilizing `matplotlib` (Hunter, 2007) and `pandas` (team, 2020), KrakenParser can generate stacked bar plots that display the relative abundances of taxa across multiple samples. These plots provide a clear comparison of taxonomic compositions between samples.
- Streamgraphs: For a more dynamic representation, KrakenParser can create streamgraphs using `matplotlib`'s (Hunter, 2007) stackplot function with a symmetric baseline. This visualization emphasizes changes in taxa abundances over a series of samples, highlighting temporal or sequential patterns.
- Combined Visualizations: To offer both detailed and overarching views, KrakenParser supports combined plots that integrate stacked bar plots and streamgraphs. This dual representation aids in comprehensive data analysis.
- Clustermaps: Employing `seaborn` (Waskom, 2021), KrakenParser can produce clustermaps that perform hierarchical clustering on taxa and samples. These heatmaps reveal patterns and groupings in the data, facilitating the identification of similar taxonomic profiles.

These visualization tools are accessible through the KrakenParser Python API, allowing users to customize and integrate them into their analysis workflows seamlessly.

The Journal of Open Source Software

Acknowledgements

The development of `KrakenParser` was supported by the Russian Science Foundation (Project no. 25-24-00351).

References

Breitwieser, F. P., Baker, D. N., & Salzberg, S. L. (2018). KrakenUniq: Confident and fast metagenomics classification using unique k-mer counts. *Genome Biology*, *19*, 198. https://doi.org/10.1186/s13059-018-1568-0

Breitwieser, F. P., & Salzberg, S. L. (2020). Pavian: Interactive analysis of metagenomics data for microbiome studies and pathogen identification. *Bioinformatics*, *36*(4), 1303–1304. https://doi.org/10.1093/bioinformatics/btz715

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, *9*(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

Kim, D., Song, L., Breitwieser, F. P., & Salzberg, S. L. (2016). Centrifuge: Rapid and sensitive classification of metagenomic sequences. *Genome Research*, *26*(12), 1721–1729. https://doi.org/10.1101/gr.210641.116

Kim, J., & Steinegger, M. (2024). Metabuli: Sensitive and specific metagenomic classification via joint analysis of amino acid and DNA. *Nature Methods*, *21*(6), 971–973. https://doi.org/10.1038/s41592-024-02273-y

Lu, J., Breitwieser, F. P., Thielen, P., & Salzberg, S. L. (2017). Bracken: Estimating species abundance in metagenomics data. *PeerJ Computer Science*, *3*, e104. https://doi.org/10.7717/peerj-cs.104

Lu, J., Rincon, N., Wood, D. E., Breitwieser, F. P., Pockrandt, C., Langmead, B., Salzberg, S. L., & Steinegger, M. (2022). Metagenome analysis using the kraken software suite. *Nature Protocols*, *17*, 2815–2839. https://doi.org/10.1038/s41596-022-00738-y

Menzel, P., Ng, K. L., & Krogh, A. (2016). Fast and sensitive taxonomic classification for metagenomics with kaiju. *Nature Communications*, *7*, 11257. https://doi.org/10.1038/ncomms11257

Ounit, R., Wanamaker, S., Close, T. J., & Lonardi, S. (2015). CLARK: Fast and accurate classification of metagenomic and genomic sequences using discriminative k-mers. *BMC Genomics*, *16*, 236. https://doi.org/10.1186/s12864-015-1419-2

Sczyrba, A., Hofmann, P., Belmann, P., Koslicki, D., Janssen, S., Dröge, J., Gregor, I., Majda, S., Fiedler, J., Dahms, E., & others. (2017). Critical assessment of metagenome interpretation: A benchmark of metagenomics software. *Nature Methods*, *14*(11), 1063–1071. https://doi.org/10.1038/nmeth.4458

team, T. pandas development. (2020). *Pandas-dev/pandas: pandas* (latest). Zenodo. https://doi.org/10.5281/zenodo.3509134

Waskom, M. L. (2021). Seaborn: Statistical data visualization. *Journal of Open Source Software*, *6*(60), 3021. https://doi.org/10.21105/joss.03021

Wood, D. E., Lu, J., & Langmead, B. (2019). Improved metagenomic analysis with kraken 2. *Genome Biology*, *20*, 257. https://doi.org/10.1186/s13059-019-1891-0

Wood, D. E., & Salzberg, S. L. (2014). Kraken: Ultrafast metagenomic sequence classification using exact alignments. *Genome Biology*, *15*(3), R46. https://doi.org/10.1186/gb-2014-15-3-r46