Koverage: Read-coverage analysis for massive (meta)genomics datasets

Michael J. Roach [1,2¶], **Bradley J. Hart** [3], **Sarah J. Beecroft** [4], **Bhavya Papudeshi** [1], **Laura K. Inglis** [1], **Susanna R. Grigson** [1], **Vijini Mallawaarachchi** [1], **George Bouras** [5,6], **and Robert A. Edwards** [1]

1 Flinders Accelerator for Microbiome Exploration, Flinders University, Adelaide, SA, Australia **2** Adelaide Centre for Epigenetics and the South Australian Immunogenomics Cancer Institute, Faculty of Health and Medical Sciences, The University of Adelaide, Adelaide, SA, Australia **3** Health and Biomedical Innovation, Clinical and Health Sciences, University of South Australia, SA, Australia **4** Pawsey Supercomputing Research Centre, Kensington, WA, Australia **5** Adelaide Medical School, Faculty of Health and Medical Sciences, The University of Adelaide, Adelaide, SA, Australia **6** The Department of Surgery – Otolaryngology Head and Neck Surgery, Central Adelaide Local Health Network, Adelaide, SA, Australia ¶ Corresponding author

DOI: 10.21105/joss.06235

Software

- Review
- Repository
- Archive

Editor: Charlotte Soneson

Reviewers:

- @lparsons
- @telatin

Submitted: 30 November 2023
Published: 26 February 2024

Summary

Genomes of organisms are constructed by assembling sequence reads from whole genome sequencing. It is useful to determine sequence read-coverage of genome assemblies, for instance identifying duplication or deletion events, identifying related contigs for binning metagenomes (Mallawaarachchi et al., 2021; Mallawaarachchi & Lin, 2022), or analysing taxonomic compositions of metagenomes (Wu et al., 2023). Although calculating read-coverage is a routine task, it typically involves several complete read and write operations (I/O operations). This is not a problem for small datasets, but can be a significant bottleneck for very large datasets. Koverage reduces I/O burden as much as possible to enable maximum scalability. Koverage includes a kmer-based method that significantly reduces the computational complexity for very large reference genomes. Koverage uses Snakemake (Mölder et al., 2021), providing out-of-the-box support for HPC and cloud environments. It utilises the Snaketool (Roach, Pierce-Ward, et al., 2022) command line interface, and is installable with PIP or Conda for maximum ease of use. Source code and documentation are available at https://github.com/beardymcjohnface/Koverage.

Statement of need

With the current state of sequencing technologies, it is trivial to generate terabytes of sequencing data for hundreds or even thousands of samples. Databases such as the Sequence Read Archive and the European Nucleotide Archive, containing nearly 100 petabytes combined of sequencing data, are constantly being mined and reanalysed in bioinformatics analyses. Memory and I/O bottlenecks lead to under-utilisation of CPUs, and computational inefficiencies at such scales waste thousands of dollars in compute costs. I/O heavy processes in large parallel batches can result in significantly impaired performance. This is especially true for HPC clusters with shared file storage, or for cloud environments using cost-efficient bucket storage.

While there are existing tools for performing coverage calculations, they are not optimised for deployment at large scales, or when analysing large reference files. They require several complete I/O operations of the sequencing data in order to generate coverage statistics. Mapping to very large genomes requires large amounts of memory, or alternatively, aligning reads in chunks creating more I/O operations. Moving I/O operations into memory, for example via `tempfs` may alleviate I/O bottlenecks. However, this is highly system-dependent and will exacerbate memory bottlenecks.

Koverage addresses these I/O bottlenecks by eliminating the sorting, reading, and writing of intermediate alignments. Koverage includes a kmer-based implementation to eliminate memory bottlenecks from screening large genomes. Koverage can be utilised as is, but has also been incorporated into the metagenomics pipelines Hecatomb (Roach, Beecroft, et al., 2022), Phables (Mallawaarachchi et al., 2023), and Reneo (Mallawaarachchi, 2023).

Implementation

Koverage is written in Snakemake (Mölder et al., 2021) and Python, and uses the Snaketool (Roach, Pierce-Ward, et al., 2022) command line interface (CLI). Snaketool takes the user input command line arguments to build a runtime config file, generate the Snakemake command, and run the pipeline. Any unrecognised arguments are assumed to be Snakemake arguments and are added to the Snakemake command. For cluster- or cloud-based execution, users are encouraged to generate a Snakemake profile, and Koverage is compatible with Snakemake's Cookiecutter (Greenfeld, 2013) template profiles. The only required inputs are the reference FASTA-format file (`--ref`), and the sample reads (`--reads`).

Sample parsing

Koverage will parse reads (`--reads`) using MetaSnek `fastq_finder` (Roach, 2023). Users supply either a directory of sequencing reads, or a tab-separated values (TSV) file of sample names and corresponding read filepaths. For a directory, sample names and read file pairs will be inferred from the file names. For a TSV file, sample names and filepaths are read from the file. More information and examples are available at https://gist.github.com/beardymcjohnface/bb161ba04ae1042299f48a4849e917c8

Mapping-based coverage

This is the default method for calculating coverage statistics. Reads are mapped sample-by-sample to the reference genome using Minimap2 (Li, 2018). The alignments are piped to a script that collects the counts per contig and total counts per sample. Koverage uses mapping coordinates to collect read counts for *bins* or *windows* along each contig. This allows for a fast approximation of the coverage of each contig by at least one read (hitrate), and of the evenness of coverage (variance) for each contig. The final counts, mean, median, hitrate, and variance are written to a Python pickle. A second script calculates the Reads Per Million (RPM), Reads Per Kilobase Million (RPKM), Reads Per Kilobase (RPK), and Transcripts Per Million (TPM) like so:

RPM $= \frac{10^6 \times N}{T}$

RPKM $= \frac{10^6 \times N}{T \times L}$

RPK $= \frac{N}{L}$

TPM $= \frac{10^6 \times RPK}{R}$

Where:

- N = number of reads mapped to the contig
- T = Total number of mapped reads for that sample
- L = length of contig in kilobases

- R = sum of all RPK values for that sample

To generate fast estimations for mean, median, hitrate, and variance, Koverage first collects the counts of the start coordinates of mapped reads within *bins* (or *windows*) across each contig (Figure 1). The user can customise bin width (default 100 bp); mean and median counts are comparable to read-depth when the binwidth is equal to the library's read length. Variance is calculated directly as the standard variance of the bin counts. The hitrate is calculated as the fraction of bins greater than zero.

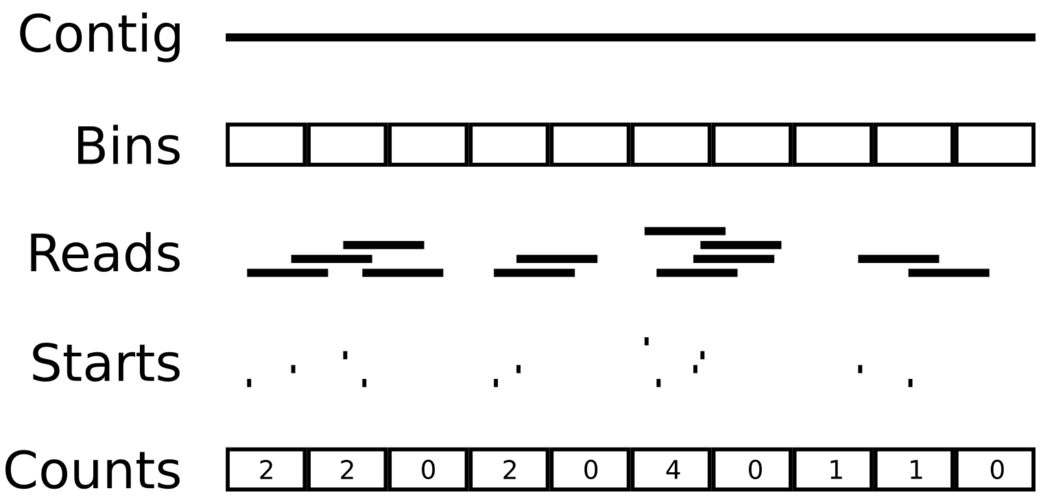

Figure 1: Windowed-coverage counts. Counts of start coordinates of mapped reads are collected for each bin across a contig. The counts array is used to calculate estimates for coverage hitrate and variance.

The coverage from all samples are collated, and a summary for each contig coverage by all samples is calculated. A summary HTML report is generated which includes interactive graphs and tables for both the per sample coverage, and the combined coverage from all samples. We utilized Datapane (Datapane Team, 2023) to embed a combined bar and line chart from Plotly (Plotly Technologies Inc, 2023) and an interactive table displaying the results.

Kmer-based coverage

Mapping to very large reference genomes can place considerable strain on computer resources. Koverage offers a kmer-based approach to estimating coverage. First, the reference genome is processed and kmers are sampled evenly across each contig. The user can customise kmer size, sampling interval, and minimum and maximum number of kmers to sample per contig. Jellyfish (Marçais & Kingsford, 2011) databases are created for each sample. The sampled reference kmers are queried against the sample kmer database. The kmer counts, and a kmer count array is created for each contig. The sum, mean, and median are calculated directly from the count array. Hitrate is calculated as the number of kmer counts > 0 divided by the total number of kmers queried. Variance is highly sensitive to large outliers, and kmer counts are especially prone to large outliers. Therefore, variance is calculated as the standard variance of the lowest 95 % of kmer counts.

CoverM wrapper

Koverage includes a wrapper for the popular CoverM (Woodcroft & Newell, 2017) tool. CoverM can parse aligned and sorted reads in BAM format. It can also align reads with Minimap2, saving the sorted alignments in a temporary filesystem (tempfs), and then process the aligned and sorted reads from tempfs. When a large enough tempfs is available, this method of running

CoverM is extremely fast. However, if the tempfs is insufficient for storing the alignments, they are instead written to and read from regular disk storage which can be a significant I/O bottleneck. This wrapper in Koverage will generate alignments with Minimap2, sort and save them in BAM format with SAMtools (Danecek et al., 2021), and run CoverM on the resulting BAM file. CoverM is currently not available for macOS and as such, this wrapper will only run on Linux systems.

Benchmarks

We tested Koverage's methods on the Pawsey Supercomputing Research Centre's Setonix HPC (commissioned in 2023) (Pawsey Supercomputing Research Centre, 2023) using a small coral metagenome dataset (Lima et al., 2023) consisting of 34 samples, a 360 Mbp metagenome assembly, and 9.1 GB of sequencing reads. This represents a typical metagenomics application in optimal conditions. Table 1 shows that CoverM is slightly faster than the default mapping-based method in spite of the extra read and write operations.

Table 1: Coral metagenome benchmarks with high performance I/O

Method	Runtime (HH:MM:SS)	CPU Walltime (HH:MM:SS)	Mean load (%)	Peak memory (Gb)
Map	00:40:34	01:49:38	270	4.6
Kmer	02:20:58	00:51:40	37	4.2
CoverM	00:31:49	01:12:17	227	7.4

We repeated the above benchmarking with Koverage directly reading and writing to Pawsey's S3 network bucket storage mounted using s3fs-fuse. Unlike the local scratch partition, this represents a scenario with a significant I/O bottleneck. Table 2 shows that while all methods are slower, Koverage's mapping and kmer methods perform much faster than the CoverM wrapper. The poor performance of the CoverM wrapper is entirely the result of generating the alignment BAM files, accounting for 85% of the overall runtime, rather than CoverM itself.

Table 2: Coral metagenome benchmarks with bottlenecked I/O

Method	Runtime (HH:MM:SS)	CPU Walltime (HH:MM:SS)	Mean load (%)	Peak memory (Gb)
Map	03:34:15	01:49:01	50	4.6
Kmer	03:18:33	01:13:53	14	4.6
CoverM	11:13:39	01:32:10	22	7.3

Acknowledgments

This work was supported by resources provided by the Pawsey Supercomputing Research Centre with funding from the Australian Government and the Government of Western Australia. The support provided by Flinders University for HPC research resources is acknowledged. This work was supported by an award from NIH NIDDK RC2DK116713 and an award from the Australian Research Council DP220102915.

References

Danecek, P., Bonfield, J. K., Liddle, J., Marshall, J., Ohan, V., Pollard, M. O., Whitwham, A., Keane, T., McCarthy, S. A., Davies, R. M., & Li, H. (2021). Twelve years of SAMtools and BCFtools. *GigaScience*, *10*(2). https://doi.org/10.1093/gigascience/giab008

Datapane Team. (2023). *Datapane (0.16.5) [Software]*. https://www.datapane.com.

Greenfeld, A. R. (2013). *Cookiecutter: A cross-platform command-line utility that creates projects from cookiecutters (project templates), e.g. Python package projects, C projects.* https://github.com/cookiecutter/cookiecutter/.

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

Lima, L. F., Alker, A. T., Papudeshi, B., Morris, M. M., Edwards, R. A., de Putron, S. J., & Dinsdale, E. A. (2023). Coral and Seawater Metagenomes Reveal Key Microbial Functions to Coral Health and Ecosystem Functioning Shaped at Reef Scale. *Microbial Ecology*. https://doi.org/10.1007/s00248-022-02094-6

Mallawaarachchi, V. (2023). *reneo: Unraveling Viral Genomes from Metagenomes*. https://github.com/Vini2/reneo.

Mallawaarachchi, V., & Lin, Y. (2022). Accurate Binning of Metagenomic Contigs Using Composition, Coverage, and Assembly Graphs. *Journal of Computational Biology*, *29*(12), 1357–1376. https://doi.org/10.1089/cmb.2022.0262

Mallawaarachchi, V., Roach, M. J., Decewicz, P., Papudeshi, B., Giles, S. K., Grigson, S. R., Bouras, G., Hesse, R. D., Inglis, L. K., Hutton, A. L. K., Dinsdale, E. A., & Edwards, R. A. (2023). Phables: from fragmented assemblies to high-quality bacteriophage genomes. *Bioinformatics*, *39*(10), btad586. https://doi.org/10.1093/bioinformatics/btad586

Mallawaarachchi, V., Wickramarachchi, A. S., & Lin, Y. (2021). Improving metagenomic binning results with overlapped bins using assembly graphs. *Algorithms for Molecular Biology*, *16*(1), 3. https://doi.org/10.1186/s13015-021-00185-6

Marçais, G., & Kingsford, C. (2011). A fast, lock-free approach for efficient parallel counting of occurrences of k-mers. *Bioinformatics*, *27*(6), 764–770. https://doi.org/10.1093/bioinformatics/btr011

Mölder, F., Jablonski, K., Letcher, B., Hall, M., Tomkins-Tinch, C., Sochat, V., Forster, J., Lee, S., Twardziok, S., Kanitz, A., Wilm, A., Holtgrewe, M., Rahmann, S., Nahnsen, S., & Köster, J. (2021). Sustainable data analysis with Snakemake. *F1000Research*, *10*(33). https://doi.org/10.12688/f1000research.29032.1

Pawsey Supercomputing Research Centre. (2023). *Setonix (HPC)*. Pawsey Supercomputing Research Centre. https://support.pawsey.org.au/documentation/display/US/Setonix+Guides

Plotly Technologies Inc. (2023). *Plotly (5.15.0) [Software]*. https://plot.ly.

Roach, M. J. (2023). *MetaSnek: Misc functions for metagenomic pipelines*. https://github.com/beardymcjohnface/metasnek.

Roach, M. J., Beecroft, S. J., Mihindukulasuriya, K. A., Wang, L., Paredes, A., Henry-Cocks, K., Lima, L. F. O., Dinsdale, E. A., Edwards, R. A., & Handley, S. A. (2022). Hecatomb: An End-to-End Research Platform for Viral Metagenomics. *bioRxiv*. https://doi.org/10.1101/2022.05.15.492003

Roach, M. J., Pierce-Ward, N. T., Suchecki, R., Mallawaarachchi, V., Papudeshi, B., Handley, S. A., Brown, C. T., Watson-Haigh, N. S., & Edwards, R. A. (2022). Ten simple rules and a template for creating workflows-as-applications. *PLOS Computational Biology*, *18*(12), 1–9. https://doi.org/10.1371/journal.pcbi.1010705

Woodcroft, B., & Newell, R. (2017). *WWOOD/coverm: Read coverage calculator for metagenomics*. https://github.com/wwood/CoverM.

Wu, E., Mallawaarachchi, V., Zhao, J., Yang, Y., Liu, H., Wang, X., Shen, C., Lin, Y., & Qiao, L. (2023). Contigs directed gene annotation (ConDiGA) for accurate protein sequence database construction in metaproteomics. *bioRxiv*. https://doi.org/10.1101/2023.04.19.537311