The Journal of Open Source Software

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

Michael J. Roach [1,2¶], **Brad Hart** [3], **Sarah Beecroft** [4], **Bhavya Papudeshi** [1], **Laura Inglis** [1], **Susanna 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.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Charlotte Soneson

Reviewers:

- @lparsons
- @telatin

Submitted: 30 November 2023
Published: unpublished

Summary

Genomes of organisms are constructed by assembling short fragments (called sequencing reads) that are the resulting data outputs of whole genome sequencing (WGS). It is useful to determine the read-coverage of sequencing reads in the resulting genome assembly for many reasons, such as identifying duplication or deletion events, identifying related contigs for binning in metagenome assemblies (Mallawaarachchi et al., 2021; Mallawaarachchi & Lin, 2022), or analysing taxonomic compositions of metagenomic samples (Wu et al., 2023). Although calculating the read-coverage of sequencing reads to a reference genome is a routine task, it typically involves several read and write operations (I/O operations) of the sequencing data. Although this is not a problem for small datasets, it can be a significant bottleneck when analysing a large number of samples, or when screening very large reference sequence files. Koverage is designed to reduce the I/O burden as much as possible to enable maximum scalability for large sample sizes. Koverage also includes a kmer-based coverage method that significantly reduces the computational complexity of screening large reference genomes such as the human genome. Koverage is a Snakemake (Mölder et al., 2021) based pipeline, providing out-of-the-box support for HPC and cloud environments. It utilises the Snaketool (N. T. A. S. Roach Michael J AND Pierce-Ward, 2022) command line interface and is available to install via PIP or Conda for maximum ease-of-use. The source code and documentation is 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 (SRA) and the European Nucleotide Archive (ENA), containing nearly 100 petabytes combined of sequencing data, are constantly being mined and reanalysed in bioinformatics analyses. Computational inefficiencies at such scales waste thousands of dollars in compute costs and contribute to excess CO2 emissions. Memory and I/O bottlenecks can lead to under-utilisation of CPUs and exacerbate these inefficiencies. In severe cases, I/O heavy processes in large parallel batches can result in significantly impaired performance. This is especially true for

HPC clusters with a shared scratch space of spinning disk hard drives, or for cloud-based analyses using cost-efficient network file systems or 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. This typically requires several complete I/O operations of the sequencing data in order to generate the coverage statistics. Furthermore, mapping to very large reference sequence files can require large amounts of memory, or alternatively, aligning reads in chunks and merging these chunked alignments at the end, resulting in even more I/O operations. Some proposed solutions involve moving I/O operations into memory, for example via `tempfs`. However, whether leveraging memory instead of I/O is a feasible option is highly system-dependent and will exacerbate any existing memory bottlenecks.

Koverage addresses the I/O bottleneck of large datasets by eliminating the sorting, reading, and writing of intermediate alignment files. Koverage also includes a kmer-based implementation to eliminate memory bottlenecks that may arise from screening large reference files. Koverage can be utilised as is, but has also been incorporated into the metagenomics pipelines Hecatomb (M. J. Roach 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 (N. T. A. S. Roach Michael J AND Pierce-Ward, 2022) command line interface (CLI). The Snaketool CLI will take the user input command line arguments and Koverage's default configuration to build a runtime config file, build the Snakemake command and run the pipeline. Any unrecognised command line 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 for their chosen deployment. Koverage has been designed to be 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` (M. J. Roach, 2023). Users supply either a directory containing their sequencing reads, or a tab-separated values (TSV) file listing their sample names and corresponding sequencing read filepaths. If users supply a directory to `--reads`, sample names and read file pairs will be inferred from the file names. If users supply a TSV file, sample names and filepaths will simply be 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 minimap2 alignments are parsed in real-time by a script that collects the counts per contig and total counts per sample. Koverage also uses the read mapping coordinates to collect read counts for `_bins_` or `_windows_` along the 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. Following mapping, the final counts, mean, median, hitrate, and variance are written to a TSV file. 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

As mentioned, Koverage uses a fast estimation for mean, median, hitrate, and variance. It estimates these values by first collecting the counts of the start coordinates of mapped reads within *bins* (or *windows*) across each contig (Figure 1). The user can customise the 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.

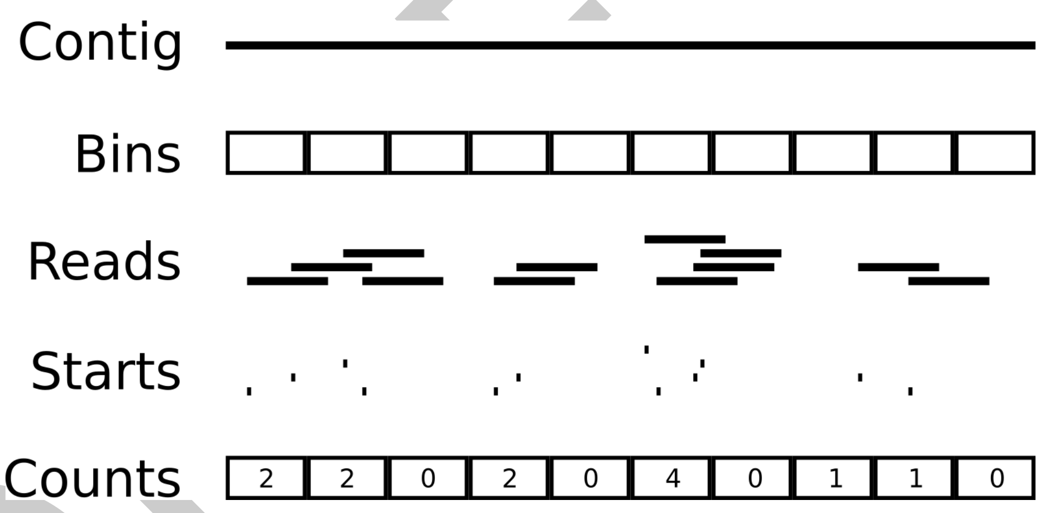

Figure 1: Windowed-coverage counts. Counts of start coordinates of mapped reads are collected for each bin across a contig. The counts array is used to calculate estimates for coverage hitrate and variance.

Lastly, the coverage from all samples are collated, and a summary of the coverage for each contig by all samples is calculated. A summary HTML report is then generated which includes interactive graphs and tables for both the per sample coverge, and the combined coverage from all samples. In the HTML report, we utilized Datapane (Datapane Team, 2023) to embed both a combined bar and line chart from Plotly (Plotly Technologies Inc, 2023) and an interactive table displaying the results. This visualization represents the reads that have been mapped to each contig within the given reference sequence. The visualization is organized into two distinct tabs: one showcasing the individual read files with their associated mapping, and the other illustrating the combined read files with their respective mapping.

Kmer-based coverage

Mapping to very large reference genomes can place considerable strain on computer resources. As an alternative, Koverage offers a kmer-based approach to estimating coverage across contigs. First, the reference genome is processed and kmers are sampled evenly across each contig.

The user can customise the kmer size, sampling interval, and minimum and maximum number of kmers to sample for each contig. Jellyfish (Marçais & Kingsford, 2011) databases are then created for each sample. Koverage will initiate an interactive Jellyfish session for each sample's kmer database. The kmers that were sampled from each reference contig are queried against the sample kmer database and 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, and the hitrate is calculated as the number of kmer counts > 0 divided by the total number of kmers queried. As variance is highly sensitive to large outliers, and kmer counts are especially prone to large outliers for repetitive sequences, the 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 use Minimap2 to generate alignments, sort them and save them in BAM format with SamTools (Danecek et al., 2021), and then run CoverM on the resulting BAM file. While this is not the fastest method for running CoverM, it is convenient for users wishing to retain the sorted alignments in BAM format, and for automated running over many samples with a combined output summary 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 the CoverM wrapper 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 Setonix's high performance 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, which accounted 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., Putron, S. J. de, & 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, N. T. A. S., Michael J AND Pierce-Ward. (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