MUSET: Set of utilities for the construction of abundance unitig matrices from sequencing data

Riccardo Vicedomini, Francesco Andreace, Yoann Dufresne, Rayan Chikhi, Camila Duitama González

▶ **To cite this version:**

Riccardo Vicedomini, Francesco Andreace, Yoann Dufresne, Rayan Chikhi, Camila Duitama González. MUSET: Set of utilities for the construction of abundance unitig matrices from sequencing data. 2024. hal-04831168

HAL Id: hal-04831168
https://hal.science/hal-04831168v1

Preprint submitted on 11 Dec 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Copyright

Bioinformatics, 2024, pp. 1–4

doi: DOI HERE
Advance Access Publication Date: Day Month Year
Paper

PAPER

MUSET: Set of utilities for the construction of abundance unitig matrices from sequencing data.

Riccardo Vicedomini[1,*] Francesco Andreace[2,3] Yoann Dufresne[2,3] Rayan Chikhi[2] and Camila Duitama González[2,*]

[1] GenScale, Univ. Rennes, Inria RBA, CNRS UMR 6074, F-35000, Rennes, France, [2] Institut Pasteur, Université Paris Cité, Sequence Bioinformatics unit, F-75015, Paris, France and [3]Sorbonne Université, Collège doctoral, F-75005, Paris, France
*Corresponding authors. E-mails: camila.duitama-gonzalez@pasteur.fr; riccardo.vicedomini@irisa.fr
FOR PUBLISHER ONLY Received on Date Month Year; revised on Date Month Year; accepted on Date Month Year

Abstract

Summary: MUSET is a novel set of utilities designed to efficiently construct abundance unitig matrices from sequencing data. Unitig matrices extend the concept of k-mer matrices, which are gaining popularity for sequence-phenotype association studies, by merging overlapping k-mers that unambiguously belong to the same sequence. MUSET addresses the limitations of current software by integrating k-mer counting and unitig extraction to generate unitig matrices containing abundance values, as opposed to only presence-absence in previous tools. These matrices preserve variations between samples while reducing disk space and reducing the number of rows in comparison to k-mer matrices. We evaluated MUSET's performance using datasets derived from a 618-GB collection of ancient oral sequencing samples, producing a filtered unitig matrix that records abundances in less than 10 hours and 20 GB memory. As a comprehensive pipeline for generating these matrices, MUSET will facilitate the extraction of biologically significant sequences, making it a valuable contribution to downstream sequencing data analyses such as genome-wide (or metagenome-wide) association studies.
Availability and implementation: MUSET is open source and publicly available under the MIT licence in GitHub at `https://github.com/CamilaDuitama/muset`.

Key words: unitigs, matrix, abundance, k-mers, de Bruijn graph

Introduction

Unitigs are biological sequences that compactly and exhaustively represent sequencing data or assembled genomes. They are constructed from k-mers, but unlike k-mers, they avoid the redundancy problem of multiple overlapping sequences covering the same genomic locus. They have proven useful for analyzing genomic diversity across several sequencing datasets [6, 11, 12]. A more formal definition characterizes unitigs as maximal non-branching paths in a de Bruijn graph (dBG). The de Bruijn graph is a fundamental data structure in bioinformatics and is widely used in many genomics applications [1, 5, 16]. It represents a set of distinct k-mers (substrings of size k) and their $k-1$ prefix-suffix overlaps as a graph [3]. The process of generating maximal unitigs from k-mers, known as *compaction*, consists of consolidating all maximal non-branching paths of k-mers into single strings [3, 17]. The resulting graph is called compacted dBG. In this work, the term "unitigs" will always refer to the deterministically unique set of maximal unitigs.

A unitig matrix is a data structure that represents sequence content across multiple experiments by recording a numerical value for each unitig across all samples. In a scenario involving a collection $\mathcal{S}$ of samples, a binary unitig matrix M has elements $M(i,j)$ that indicate the presence (1) or absence (0) of unitig i in sample j. In other words, rows are unitigs and columns are samples.

Related work

Numerous cutting-edge tools that rely on the construction of a de Bruijn graph for unitig computation have been developed over the years, including BCALM [4], Cuttlefish [13], Bifrost [10], and GGCAT [6]. By computing the unitigs of a union of multiple samples, one can construct a binary unitig matrix. However, current software overlooks the concept of abundance, which estimates the frequency with which a unitig appears in a sample.

BCALM and Cuttlefish simply produce a set of unitigs, i.e. a *compacted* dBG, without recording the sample of origin. GGCAT and Bifrost, on the other hand, can build *colored compacted* dBGs, a variant of dBGs that additionally keeps track of the source of each k-mer [11]. Such graphs are implicitly binary unitig matrices. Notably, GGCAT has demonstrated superior performance to its counterparts [6].

Unitig matrices are data structures analogous to k-mer matrices, which are commonly used to represent

© The Author 2024. Published by Oxford University Press. All rights reserved. For permissions, please e-mail: journals.permissions@oup.com

sequence content across multiple experiments, e.g. for indexing environmental metagenomes [14], source attribution of pathogenic bacteria via supervised learning [2], transcriptomic analyses [20], contamination removal and contamination assessment of ancient metagenomic data [7, 9]. While k-mer matrices can be constructed efficiently (e.g. using kmtricks [15]), storing and processing large k-mer matrices remains challenging, especially when integrating them with existing libraries for machine learning, dimensionality reduction, or visualization. As an alternative, unitig matrices offer a more compact and manageable representation, preserving the variations between samples while optimizing disk-space usage and processing time. Individual k-mer counts can be naturally averaged along a unitig to produce a single abundance value per unitig, robustly approximating the counts of all k-mers from that unitig [18].

Using unitigs instead of k-mers can reduce multiple testing burden, a common challenge in the study of genomic variation across datasets [12]. Presence-absence unitig matrices have proven to be useful for statistical analyses in human genomic studies, where the frequency of sequences that characterize certain species can effectively distinguish between phenotypes more accurately than their mere presence or absence [8]. Unitig matrices also have the potential to be useful for RNA-Seq differential expression analyses in transcriptomics, where low-coverage unitigs can help identify sequencing errors. However, to our knowledge, there is no software that effectively addresses the gap that exists in the construction of abundance unitig matrices, as opposed to presence-absence unitig matrices. For this reason, we introduce MUSET, a pipeline designed for the practical construction of abundance unitig matrices. Along with it, we additionally provide `kmat_tools`, a comprehensive suite of tools (internally exploited by MUSET) for manipulating k-mer matrices, and `muset_pa`, a pipeline that uses GGCAT and `kmat_tools` to rapidly build a presence-absence unitig matrix with no k-mer filtering.

Tool description and evaluation

MUSET overview

MUSET leverages kmtricks [15] for efficient k-mer counting over large collections of genomic sequences provided as FASTA/FASTQ files. It then uses GGCAT [6] for unitig construction and SSHash [19] to assign k-mer counts to unitigs. Intermediate filtering steps are performed at both k-mer and unitig levels. The final output is a unitig matrix where rows correspond to unitigs, columns correspond to samples, and each element encodes the average abundance and fraction of the unitig's k-mers that are present in a sample. The pipeline is depicted in Fig. 1 and consists of the following main steps explained below: (i) k-mer matrix construction, (ii) k-mer matrix filtering, (iii) unitig construction, and (iv) unitig matrix construction.

k-mer matrix construction

kmtrick's pipeline is run to build a k-mer matrix from the input FASTA/Q files. The matrix is stored in partitions using a custom lz4-compressed binary format.

k-mer matrix filtering

kmtricks's matrix partitions are filtered concurrently and merged into a single k-mer matrix in text format. The filtering aims to retain k-mers that potentially reflect differences between samples. More precisely, by default, we keep k-mers that are present in at least 10% of the samples and absent in at least 10% of the samples. Users can set custom values to both of these thresholds by providing either a fraction or an absolute number.

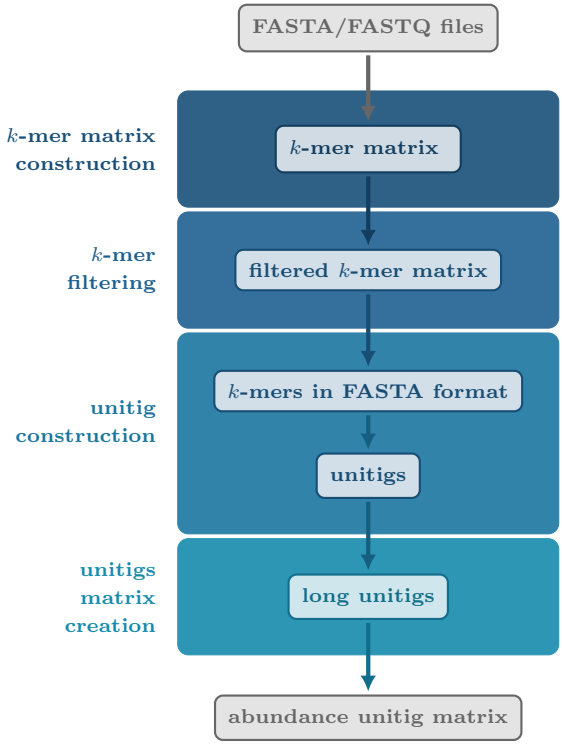

Fig. 1. MUSET pipeline. It consists of four main steps: k-mer matrix construction, k-mer matrix filtering, unitig creation, and unitig matrix creation.

Unitig construction

The k-mers of the filtered matrix are outputted in FASTA format (command `kmat_tools fasta`) to build a set of unitigs with GGCAT. Unitigs shorter than a certain threshold ($2k-1$ base pairs by default) are then discarded using the command `kmat_tools fafmt`.

Unitig matrix construction

This step involves the execution of the command `kmat_tools unitig` to create an abundance unitig matrix. First, an SSHash-based dictionary is built from the (filtered) unitigs in order to assign each k-mer to the unitig it belongs to. The filtered k-mer matrix is then processed to extract abundance values and k-mer presence, while simultaneously aggregating k-mer data at the unitig level. The aggregation is done for each unitig, by computing the fraction of the unitig's k-mers belonging to a given sample and their average abundance.

More precisely, the fraction of k-mers in a unitig u that are present in a sample S is defined as:

$$f(u,S) = \frac{\sum_{i=1}^{N} x_i}{N} \tag{1}$$

where N is the number of k-mers in u, and x_i is a binary variable that is 1 when the i-th k-mer is present in sample S and 0 otherwise.

Table 1. Running time, peak memory and disk usage of MUSET on the small and large datasets. The two datasets are derived from the same collection of FASTQ files of ancient DNA reads. The small dataset corresponds to a subset of the k-mer matrix of the large dataset. Wall-clock time and memory usage were computed using 20 threads.

MUSET step	Small dataset (11 GB)			Large dataset (618 GB)		
	Wall-clock time	Peak memory	Disk usage	Wall-clock time	Peak memory	Disk usage
k-mer matrix construction	3h 6m 46s	2.4 GB	11 GB	7h 47m 59s	19 GB	1437 GB
k-mer filtering	0h 2m 04s	<0.1 GB	0.1 GB	1h 37m 25s	<0.1 GB	80 GB
Unitig creation (GGCAT)	0h 0m 28s	0.7 GB	<0.1 GB	0h 2m 28s	1.2 GB	0.9 GB
Unitig matrix creation	0h 0m 01s	<0.1 GB	<0.1 GB	0h 15m 20s	2.3 GB	2.7 GB
Total	3h 8m 10s	2.4 GB	11.2 GB	9h 43m 12s	19 GB	1525 GB

Table 2. Comparison of running time, peak memory, and disk usage between MUSET (filtered unitig matrix) and GGCAT (unfiltered unitigs) on the large dataset. The large dataset consists of a 618-GB collection of 360 samples (compressed FASTQ files) from an ancient metagenomic dataset. Tools were run using 20 threads.

Method	Wall-clock time	Peak memory	Disk usage
MUSET	**9h 43m 12s**	**19 GB**	**1.5 TB**
GGCAT	24h 20m 40s	167 GB	641 GB

The average abundance of a unitig u with respect to a sample S is defined as:

$$A(u, S) = \frac{\sum_{i=1}^{N} c_i}{N} \tag{2}$$

where N is the number of k-mers in u, and c_i is the abundance of the i-th k-mer of u in sample S.

Evaluation

We evaluated the performance of MUSET (Table 1) on two datasets composed of a collection of 360 ancient oral samples and their potential contaminants [7]. This collection is made of compressed FASTQ files which add up to 618 GB. The first dataset, referred to as the small dataset, consists of a subset of rows of the full k-mer matrix obtained from the aforementioned collection. The small-dataset matrix contains ≈ 14.3 million k-mers and has a size of 11 GB. The second dataset, denoted as the large dataset, encompasses the entire matrix. The large-dataset matrix contains ≈ 64.6 billion k-mers and has a size of 1.4 TB. The instructions to reproduce our results are available at the following link: `https://github.com/CamilaDuitama/muset/tree/main/reproducibility`.

To the best of our knowledge, no other tool can directly create an abundance unitig matrix from sequencing data. The most similar state-of-the-art tool to produce analogous presence-absence matrices is GGCAT (which we happen to use inside MUSET). For this reason, we compared MUSET to a stand-alone execution of GGCAT by running both tools on the large test dataset.

In Table 2, MUSET demonstrates superior performance over GGCAT in processing the large metagenomic dataset, creating a filtered abundance unitig matrix more than twice faster (9.7 h vs. 24.3 h), using 88 % less memory (19 GB vs. 167 GB), yet at the expense of a higher disk usage (1.5 TB vs. 641 GB). Notably, MUSET is able to reduce the size of the k-mer matrix by 2 (small dataset) to 3 (large dataset) orders of magnitude when converting it to a unitig matrix. MUSET not only achieves shorter running times and lower peak memory usage than GGCAT due to its upstream k-mer filter, but it is also more convenient and informative as it directly produces an abundance unitig matrix. GGCAT instead produces a colored compacted dBG in text format with colors in binary format, and the user would have to generate a presence-absence unitig matrix on their own: this is now directly achievable using the `muset_pa` tool we provide.

Discussion and conclusions

Abundance unitig matrices provide a compact, manageable, and more informative representation of sequencing data across multiple samples, offering a valuable resource for downstream analyses in large-scale (meta/pan)genomic studies. Besides, the representation of sequencing data as abundance unitig matrices offers a versatile option with potential applications in various contexts: from sequencing error filtering and indexing, to reference-free comparison of sequencing samples. It also provides a useful input for supervised and unsupervised learning models built on large-scale sequencing datasets.

The most time-consuming and memory-intensive step in MUSET is the k-mer matrix construction, which can be optionally skipped if users have already created their own k-mer matrix in text format. This is followed by the k-mer filtering step (see Table 1). While k-mer filtering can theoretically be bypassed by setting appropriate parameters, doing so will increase disk space and peak memory usage in subsequent pipeline steps. We recommend to apply this k-mer filtering to discard "uninformative" k-mers and reduce computational burden. However, stringent filtering may interfere with the biological interpretation of unitigs, by possibly yielding chimeric sequences, which is worth investigating and mitigating in future work.

To conclude, MUSET addresses the technical gap associated with generating abundance unitig matrices to represent extensive collections of sequencing datasets. The tool utilizes kmtricks for efficient indexing and k-mer counting, GGCAT for constructing unitigs, and SSHash to efficiently map k-mers to unitigs. Ultimately, MUSET reports the abundance of each unitig in the collection of samples by estimating the fraction of observed k-mers relative to the total number of k-mers that constitute each unitig, in addition to calculating the average abundance per sample (see Equation 2). We anticipate that MUSET will be a valuable replacement to k-mer matrices of sequencing data. By providing a tool that offers a smaller representation with equivalent information content, MUSET has the potential to accelerate and facilitate the provision of biologically significant and reference-free insights into sequencing data.

Competing interests

The authors declare no competing interests.

Author contributions statement

R.C., R.V. and C.D. conceived the experiments, C.D. and R.V conducted the experiments, R.V., C.D., F.A. and R.C. wrote the manuscript. R.V., C.D., F.A. and Y.D. developed and tested the software. All authors read and approved the manuscript.

Acknowledgments

We would like to thank Hugues Richard for his contribution in reading and suggesting improvements. R.C was supported by ANR (ANR-22-CE45-0007, ANR-19-CE45-0008, PIA/ANR16-CONV-0005, ANR-19-P3IA-0001, ANR-21-CE46-0012-03) and Horizon Europe grants No. 872539, 956229, 101047160 and 101088572.

Software availability

MUSET is open source, implemented in bash and C++, and provided with `kmat_tools`, a collection of tools for processing *k*-mer matrices. The instructions on how to install and run it are available at `https://github.com/CamilaDuitama/muset`.

References

1. Francesco Andreace, Pierre Lechat, Yoann Dufresne, and Rayan Chikhi. Comparing methods for constructing and representing human pangenome graphs. *Genome Biology*, 24(1):274, Nov 2023.
2. Pierluigi Castelli, Andrea De Ruvo, Andrea Bucciacchio, Nicola D'Alterio, Cesare Cammà, Adriano Di Pasquale, and Nicolas Radomski. Harmonization of supervised machine learning practices for efficient source attribution of Listeria monocytogenes based on genomic data. *BMC genomics*, 24(1):560, 2023.
3. Rayan Chikhi, Jan Holub, and Paul Medvedev. Data structures to represent a set of k-long DNA sequences. *ACM Computing Surveys (CSUR)*, 54(1):1–22, 2021.
4. Rayan Chikhi, Antoine Limasset, and Paul Medvedev. Compacting de Bruijn graphs from sequencing data quickly and in low memory. *Bioinformatics*, 32(12):i201–i208, 2016.
5. Phillip E. C. Compeau, Pavel A. Pevzner, and Glenn Tesler. How to apply de Bruijn graphs to genome assembly. *Nature Biotechnology*, 29(11):987–991, Nov 2011.
6. Andrea Cracco and Alexandru I Tomescu. Extremely fast construction and querying of compacted and colored de Bruijn graphs with GGCAT. *Genome Research*, 33(7):1198–1207, 2023.
7. Camila Duitama González, Riccardo Vicedomini, Téo Lemane, Nicolas Rascovan, Hugues Richard, and Rayan Chikhi. decOM: similarity-based microbial source tracking of ancient oral samples using k-mer-based methods. *Microbiome*, 11(1):243, 2023.
8. Arthur Frouin, Fabien Laporte, Lukas Hafner, Mylene Maury, Zachary R. McCaw, Hanna Julienne, Léo Henches, Rayan Chikhi, Marc Lecuit, and Hugues Aschard. ChoruMM: a versatile multi-components mixed model for bacterial-GWAS. *bioRxiv*, March 2023.
9. Camila Duitama González, Samarth Rangavittal, Riccardo Vicedomini, Rayan Chikhi, and Hugues Richard. aKmerBroom: Ancient oral DNA decontamination using Bloom filters on k-mer sets. *Iscience*, 26(11), 2023.
10. Guillaume Holley and Páll Melsted. Bifrost: highly parallel construction and indexing of colored and compacted de Bruijn graphs. *Genome biology*, 21:1–20, 2020.
11. Zamin Iqbal, Mario Caccamo, Isaac Turner, Paul Flicek, and Gil McVean. De novo assembly and genotyping of variants using colored de Bruijn graphs. *Nature genetics*, 44(2):226–232, 2012.
12. Magali Jaillard, Leandro Lima, Maud Tournoud, Pierre Mahé, Alex Van Belkum, Vincent Lacroix, and Laurent Jacob. A fast and agnostic method for bacterial genome-wide association studies: Bridging the gap between k-mers and genetic events. *PLoS genetics*, 14(11):e1007758, 2018.
13. Jamshed Khan, Marek Kokot, Sebastian Deorowicz, and Rob Patro. Scalable, ultra-fast, and low-memory construction of compacted de Bruijn graphs with Cuttlefish 2. *Genome biology*, 23(1):190, 2022.
14. Téo Lemane, Nolan Lezzoche, Julien Lecubin, Eric Pelletier, Magali Lescot, Rayan Chikhi, and Pierre Peterlongo. Indexing and real-time user-friendly queries in terabyte-sized complex genomic datasets with kmindex and ORA. *Nature Computational Science*, 4(2):104–109, 2024.
15. Téo Lemane, Paul Medvedev, Rayan Chikhi, and Pierre Peterlongo. Kmtricks: efficient and flexible construction of Bloom filters for large sequencing data collections. *Bioinformatics Advances*, 2(1):vbac029, 2022.
16. Hélène Lopez-Maestre, Lilia Brinza, Camille Marchet, Janice Kielbassa, Sylvère Bastien, Mathilde Boutigny, David Monnin, Adil El Filali, Claudia Marcia Carareto, Cristina Vieira, Franck Picard, Natacha Kremer, Fabrice Vavre, Marie-France Sagot, and Vincent Lacroix. SNP calling from RNA-seq data without a reference genome: identification, quantification, differential analysis and impact on the protein sequence. *Nucleic Acids Research*, 44(19):e148–e148, 07 2016.
17. Camille Marchet. Sneak peek at the-tig sequences: useful sequences built from nucleic acid data. 2022.
18. Camille Marchet, Zamin Iqbal, Daniel Gautheret, Mikaël Salson, and Rayan Chikhi. REINDEER: efficient indexing of k-mer presence and abundance in sequencing datasets. *Bioinformatics*, 36(Supplement_1):i177–i185, July 2020.
19. Giulio Ermanno Pibiri. Sparse and skew hashing of k-mers. *Bioinformatics*, 38(Supplement_1):i185–i194, 2022.
20. Raíssa Silva, Cédric Riedel, Benoit Guibert, Florence Ruffle, Anthony Boureux, and Thérèse Commes. A k-mer based transcriptomics analysis for NPM1-mutated AML. *medRxiv*, pages 2023–01, 2023.