The Journal of Open Source Software

Back to sequences: Find the origin of k-mers

Anthony Baire[1], **Pierre Marijon**[2], **Francesco Andreace**[3], **and Pierre Peterlongo**[1]

1 Univ. Rennes, Inria, CNRS, IRISA - UMR 6074, Rennes, F-35000 France **2** Laboratoire de Biologie Médicale Multisites SeqOIA, Paris, France **3** Department of Computational Biology, Institut Pasteur, Université Paris Cité, Paris, F-75015, France

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Open Journals

Reviewers:

- @openjournals

Submitted: 01 January 1970
Published: unpublished

Abstract

A vast majority of bioinformatics tools dedicated to the treatment of raw sequencing data heavily use the concept of k-mers, which are words of length k. This enables us to reduce the redundancy of data (and thus the memory pressure), to discard sequencing errors, and to dispose of objects of fixed size that can be easily manipulated and compared to each other. A drawback is that the link between each k-mer and the original set of sequences to which it belongs is lost. Given the volume of data considered in this context, recovering this association is costly. In this work, we present "back_to_sequences", a simple tool designed to index a set of k-mers of interest and to stream a set of sequences, extracting those containing at least one of the indexed k-mer. In addition, the occurrence positions of k-mers in the sequences can be provided. Our results show that back_to_sequences streams ≈ 200 short reads per millisecond, allowing to search k-mers in hundreds of millions of reads in a matter of a few minutes.

Statement of Need

In the 2010s, following the emergence of next-generation sequencing technology, read assembly strategies based on the overlap-layout-consensus paradigm (OLC) were unable to scale to tens of millions of reads or more, prompting the usage of the *de Bruijn* graph (dBG) data structure (Flicek & Birney, 2009; Schatz et al., 2010). The success of dBG was due to the fact that the main difficulties associated with the nature of the sequencing data (read redundancy, nonuniform coverage and nonuniform overlap between reads, sequencing errors, unknown sequencing strand) were complex to handle with OLC while being easy to handle or simply solved with the dBG approach (Li et al., 2012).

Recall that in the dBG assembly approach, 1. All k-mers (words oflength k) from a set of reads are counted; 2. Those with an abundance lower than a threshold are considered to contain sequencing errors and are discarded; 3. The remaining k-mers are organized in a dBG; 4. The paths of the dBG form the basis of the assembly, later improved thanks to scaffolding tools (Huson et al., 2002) such as the tools provided, for instance, by the Spades assembler (Bankevich et al., 2012).

The usefulness of k-mers did not end with their use in dBGs. A large and redundant set of sequences, such as a sequencing read set, can be summarized by its set of k-mers. Among multiple fundamental tasks, this has been the basis for metagenome comparisons (Benoit et al., 2016), for taxonomy characterization (Wood et al., 2019), for indexing purposes (Cracco & Tomescu, 2023; Lemane, Medvedev, et al., 2022), for genotyping (Grytten et al., 2022), for species identification (Sarmashghi et al., 2019), for transcript expression estimation (Zhang & Wang, 2014), or for variant discovery (Uricaru et al., 2015) to cite only a few examples.

The Journal of Open Source Software

One of the keys to the success of the use of k-mers is its low resource needs. Whatever the sequencing coverage, once filtered, the number of distinct k-mers is at most equal to the original genome size. This offers a minimal impact on random access memory (RAM) and/or disk needs. However, this comes at the cost of losing the link between each k-mer and the sequence(s) from which it originates. Storing these links explicitly would reintroduce the problem associated with the abundance of original reads, as the link between each original read and each of its k-mers would have to be stored. For instance, considering k-mers from a sequencing experiment of a human genome (≈ 3 billion nucleotides) with a coverage of 50x (each k-mer occurs on average in 50 distinct reads) would require more than 2TB of space considering 64 bits for storing each link and 64 bits for storing the associated read identifier. This is not acceptable.

There are many situations in which finding the origin of k-mers from a set in a set of reads is informative. For instance, this is the case in studies in which the output is composed of k-mers associated with biological knowledge such as biological variants (Uricaru et al., 2015), or k-mers specific to a phenotypic trait (Lemane, Chikhi, et al., 2022). This approach can also be used for quality control (Plaza Onate et al., 2015) or contamination removal (González et al., 2023) for instance.

Recovering the link between a k-mer and each original read in which it occurs can be performed by indexing the reads (Marchet et al., 2020) which is too costly for hundreds of millions reads. One may also apply *grep-like* evolved pattern matching approaches such as `pt` (Monochromegane, 2018). However, even though they have been highly optimized in recent decades, these approaches cannot efficiently detect thousands of k-mers in millions of reads. The approaches that use k-mers for genotyping such as kage (Grytten et al., 2022) may find the number of occurrences of k-mers but do not extract the sequences from which they originate.

In this context, we propose `back_to_sequences`, a tool specifically dedicated to extracting from $\mathcal{S}$, a set of sequences (e.g. reads), those that contain some of the k-mers from a set $\mathcal{K}$ given as input. The occurrence positions of k-mers in each sequence of the queried set $\mathcal{S}$ can also be output.

Possible Alternatives

To the best of our knowledge, there exists no tool specifically dedicated to this task, while indexing the set of k-mers $\mathcal{K}$.

Genotypers as kage (Grytten et al., 2022), using `kmer mapper` (Ivar Grytten, 2020), provide a way to count the number of occurrences of each k-mer from a set of reference k-mers in a read file. However, they do not offer a feature for extracting reads that contain any reference k-mer.

Finding one unique k-mer of interest in a set of sequences can be done using the classical `grep` or more recent pattern-matching tools such as "*The Platinum Searcher*" (Monochromegane, 2018) or "*The Silver Searcher*" (Greer, 2020).

As for testing, we queried one k-mer (with $k = 31$) in a dataset composed of 100 million sequences, each of length 100 nucleotides (see the documentation for details), using these three tools.

- `grep` required 44 seconds. Thus, by simple extrapolation, searching for one million k-mers on a single computer would require approximately 500 days, to be compared to less than a minute using `back_to_sequences`.
- `pt` (*The Platinum Searcher*) required 15 seconds, which can be extrapolated to approximately 175 days if searching for one million k-mers.
- `ag` (*The Silver Searcher*) did not finish after 400 seconds.

Summing up, we find these alternative tools are not appropriate for querying numerous patterns at the same time and do not scale to large problem instances.

Note also that these alternative tools are not specialized for genomic data in which one is interested in searching for a k-mer and potentially its reverse complement. Finally, these tools do not easily provide the number of occurrences or occurrence positions of each of the searched patterns when there are many.

Conclusion

We believe that back_to_sequences is a generic and handy tool that will be beneficial for building pipelines that require manipulating k-mers and recovering the sequences from which they originate and/or counting their number of occurrences in a set of genomic sequences. We also believe that back_to_sequences will have other straightforward applications, in such areas as quality control, contamination removal, or genotyping known pieces of sequences in raw sequencing datasets. Because of the efficiency of our approach, such applications could be executed in real time during the sequencing process.

Acknowledgements

We acknowledge the GenOuest core facility (https://www.genouest.org) for providing the computing infrastructure. The work was funded by ANR SeqDigger (ANR-19-CE45-0008), and by the Inria Challenge "OmicFinder" (https://project.inria.fr/omicfinder/).

References

Bankevich, A., Nurk, S., Antipov, D., Gurevich, A. A., Dvorkin, M., Kulikov, A. S., Lesin, V. M., Nikolenko, S. I., Pham, S., Prjibelski, A. D., & others. (2012). SPAdes: A new genome assembly algorithm and its applications to single-cell sequencing. *Journal of Computational Biology*, *19*(5), 455–477. https://doi.org/10.1089/cmb.2012.0021

Benoit, G., Peterlongo, P., Mariadassou, M., Drezen, E., Schbath, S., Lavenier, D., & Lemaitre, C. (2016). Multiple comparative metagenomics using multiset k-mer counting. *PeerJ Computer Science*, *2*, e94. https://doi.org/10.7717/peerj-cs.94

Cracco, A., & Tomescu, A. I. (2023). Extremely fast construction and querying of compacted and colored de bruijn graphs with GGCAT. *Genome Research*, gr–277615. https://doi.org/10.1101/gr.277615.122

Flicek, P., & Birney, E. (2009). Sense from sequence reads: Methods for alignment and assembly. *Nature Methods*, *6*(Suppl 11), S6–S12. https://doi.org/10.1038/nmeth.1376

González, C. D., Rangavittal, S., Vicedomini, R., Chikhi, R., & Richard, H. (2023). aKmerBroom: Ancient oral DNA decontamination using bloom filters on k-mer sets. *Iscience*, *26*(11). https://doi.org/10.1016/j.isci.2023.108057

Greer, G. (2020). *The Silver Searcher*. https://github.com/ggreer/the_silver_searcher.

Grytten, I., Dagestad Rand, K., & Sandve, G. K. (2022). KAGE: Fast alignment-free graph-based genotyping of SNPs and short indels. *Genome Biology*, *23*(1), 209. https://doi.org/10.1186/s13059-022-02771-2

Huson, D. H., Reinert, K., & Myers, E. W. (2002). The greedy path-merging algorithm for contig scaffolding. *Journal of the ACM (JACM)*, *49*(5), 603–615. https://doi.org/10.1145/585265.585267

Ivar Grytten, K. D. R. (2020). *Kmer Mapper*. https://github.com/ivargr/kmer_mapper.

Lemane, T., Chikhi, R., & Peterlongo, P. (2022). Kmdiff, large-scale and user-friendly differential k-mer analyses. *Bioinformatics*, *38*(24), 5443–5445. https://doi.org/10.1093/bioinformatics/btac689

Lemane, T., Medvedev, P., Chikhi, R., & Peterlongo, P. (2022). Kmtricks: Efficient and flexible construction of bloom filters for large sequencing data collections. *Bioinformatics Advances*, *2*(1), vbac029. https://doi.org/10.1093/bioadv/vbac029

Li, Z., Chen, Y., Mu, D., Yuan, J., Shi, Y., Zhang, H., Gan, J., Li, N., Hu, X., Liu, B., & others. (2012). Comparison of the two major classes of assembly algorithms: Overlap–layout–consensus and de-bruijn-graph. *Briefings in Functional Genomics*, *11*(1), 25–37. https://doi.org/10.1093/bfgp/elr035

Marchet, C., Lecompte, L., Limasset, A., Bittner, L., & Peterlongo, P. (2020). A resource-frugal probabilistic dictionary and applications in bioinformatics. *Discrete Applied Mathematics*, *274*, 92–102. https://doi.org/10.1016/j.dam.2018.03.035

Monochromegane. (2018). *The Platinum Searcher*. https://github.com/monochromegane/the_platinum_searcher.

Plaza Onate, F., Batto, J.-M., Juste, C., Fadlallah, J., Fougeroux, C., Gouas, D., Pons, N., Kennedy, S., Levenez, F., Dore, J., & others. (2015). Quality control of microbiota metagenomics by k-mer analysis. *BMC Genomics*, *16*, 1–10. https://doi.org/10.1186/s12864-015-1406-7

Sarmashghi, S., Bohmann, K., P. Gilbert, M. T., Bafna, V., & Mirarab, S. (2019). Skmer: Assembly-free and alignment-free sample identification using genome skims. *Genome Biology*, *20*, 1–20. https://doi.org/10.1186/s13059-019-1632-4

Schatz, M. C., Delcher, A. L., & Salzberg, S. L. (2010). Assembly of large genomes using second-generation sequencing. *Genome Research*, *20*(9), 1165–1173. https://doi.org/10.1101/gr.101360.109

Uricaru, R., Rizk, G., Lacroix, V., Quillery, E., Plantard, O., Chikhi, R., Lemaitre, C., & Peterlongo, P. (2015). Reference-free detection of isolated SNPs. *Nucleic Acids Research*, *43*(2), e11–e11. https://doi.org/10.1093/nar/gku1187

Wood, D. E., Lu, J., & Langmead, B. (2019). Improved metagenomic analysis with kraken 2. *Genome Biology*, *20*, 1–13. https://doi.org/10.1186/s13059-019-1891-0

Zhang, Z., & Wang, W. (2014). RNA-skim: A rapid method for RNA-seq quantification at transcript level. *Bioinformatics*, *30*(12), i283–i292. https://doi.org/10.1093/bioinformatics/btu288