

¹ Back to sequences: Find the origin of k -mers

² **Anthony Baire¹, Pierre Marijon  ², Francesco Andreace  ³, and Pierre Peterlongo **

⁴ **1** Univ. Rennes, Inria, CNRS, IRISA - UMR 6074, Rennes, F-35000 France **2** Laboratoire de Biologie
⁵ Médicale Multisites SeqOIA, Paris, France **3** Department of Computational Biology, Institut Pasteur,
⁶ Université Paris Cité, Paris, F-75015, France

DOI: [10.xxxxxx/draft](https://doi.org/10.xxxxxx/draft)

Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

Editor: [Open Journals](#) 

Reviewers:

- [@openjournals](#)

Submitted: 01 January 1970

Published: unpublished

License

Authors of papers retain copyright
and release the work under a
Creative Commons Attribution 4.0
International License ([CC BY 4.0](#))

⁷ Abstract

⁸ A vast majority of bioinformatics tools dedicated to the treatment of raw sequencing data
⁹ heavily use the concept of k -mers, which are words of length k . This enables us to reduce
¹⁰ the redundancy of data (and thus the memory pressure), to discard sequencing errors, and to
¹¹ dispose of objects of fixed size that can be easily manipulated and compared to each other. A
¹² drawback is that the link between each k -mer and the original set of sequences to which it
¹³ belongs is lost. Given the volume of data considered in this context, recovering this association
¹⁴ is costly. In this work, we present “back_to_sequences”, a simple tool designed to index a set
¹⁵ of k -mers of interest and to stream a set of sequences, extracting those containing at least
¹⁶ one of the indexed k -mer. In addition, the occurrence positions of k -mers in the sequences
¹⁷ can be provided. Our results show that back_to_sequences streams ≈ 200 short reads per
¹⁸ millisecond, allowing to search k -mers in hundreds of millions of reads in a matter of a few
¹⁹ minutes.

²⁰ Statement of Need

²¹ In the 2010s, following the emergence of next-generation sequencing technology, read assembly
²² strategies based on the overlap-layout-consensus paradigm (OLC) were unable to scale to
²³ tens of millions of reads or more, prompting the usage of the *de Bruijn* graph (dBG) data
²⁴ structure (Flicek & Birney, 2009; Schatz et al., 2010). The success of dBG was due to the fact
²⁵ that the main difficulties associated with the nature of the sequencing data (read redundancy,
²⁶ nonuniform coverage and nonuniform overlap between reads, sequencing errors, unknown
²⁷ sequencing strand) were complex to handle with OLC while being easy to handle or simply
²⁸ solved with the dBG approach (Li et al., 2012).

²⁹ Recall that in the dBG assembly approach, 1. All k -mers (words of length k) from a set of
³⁰ reads are counted; 2. Those with an abundance lower than a threshold are considered to
³¹ contain sequencing errors and are discarded; 3. The remaining k -mers are organized in a
³² dBG; 4. The paths of the dBG form the basis of the assembly, later improved thanks to
³³ scaffolding tools (Huson et al., 2002) such as the tools provided, for instance, by the Spades
³⁴ assembler (Bankevich et al., 2012).

³⁵ The usefulness of k -mers did not end with their use in dBGs. A large and redundant set of
³⁶ sequences, such as a sequencing read set, can be summarized by its set of k -mers. Among
³⁷ multiple fundamental tasks, this has been the basis for metagenome comparisons (Benoit et
³⁸ al., 2016), for taxonomy characterization (Wood et al., 2019), for indexing purposes (Cracco
³⁹ & Tomescu, 2023; Lemane, Medvedev, et al., 2022), for genotyping (Grytten et al., 2022), for
⁴⁰ species identification (Sarmashghi et al., 2019), for transcript expression estimation (Zhang &
⁴¹ Wang, 2014), or for variant discovery (Uricaru et al., 2015) to cite only a few examples.

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

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

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

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

71 Possible Alternatives

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

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

78 Finding one unique k -mer of interest in a set of sequences can be done using the classical grep
79 or more recent pattern-matching tools such as “*The Platinum Searcher*” (Monochromegane,
80 2018) or “*The Silver Searcher*” (Greer, 2020).

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

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

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

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

96 Conclusion

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

104 Acknowledgements

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

108 References

- 109 Bankevich, A., Nurk, S., Antipov, D., Gurevich, A. A., Dvorkin, M., Kulikov, A. S., Lesin, V.
110 M., Nikolenko, S. I., Pham, S., Prjibelski, A. D., & others. (2012). SPAdes: A new genome
111 assembly algorithm and its applications to single-cell sequencing. *Journal of Computational
112 Biology*, 19(5), 455–477. <https://doi.org/10.1089/cmb.2012.0021>
- 113 Benoit, G., Peterlongo, P., Mariadassou, M., Drezen, E., Schbath, S., Lavenier, D., & Lemaitre,
114 C. (2016). Multiple comparative metagenomics using multiset k -mer counting. *PeerJ Computer
115 Science*, 2, e94. <https://doi.org/10.7717/peerj-cs.94>
- 116 Cracco, A., & Tomescu, A. I. (2023). Extremely fast construction and querying of compacted
117 and colored de bruijn graphs with GGCAT. *Genome Research*, gr-277615. [https://doi.org/
118 10.1101/gr.277615.122](https://doi.org/10.1101/gr.277615.122)
- 119 Flicek, P., & Birney, E. (2009). Sense from sequence reads: Methods for alignment and
120 assembly. *Nature Methods*, 6(Suppl 11), S6–S12. <https://doi.org/10.1038/nmeth.1376>
- 121 González, C. D., Rangavittal, S., Vicedomini, R., Chikhi, R., & Richard, H. (2023). aKmer-
122 Broom: Ancient oral DNA decontamination using bloom filters on k -mer sets. *Iscience*,
123 26(11). <https://doi.org/10.1016/j.isci.2023.108057>
- 124 Greer, G. (2020). *The Silver Searcher*. https://github.com/ggreer/the_silver_searcher.
- 125 Grytten, I., Dagestad Rand, K., & Sandve, G. K. (2022). KAGE: Fast alignment-free graph-
126 based genotyping of SNPs and short indels. *Genome Biology*, 23(1), 209. [https://doi.org/
127 10.1186/s13059-022-02771-2](https://doi.org/10.1186/s13059-022-02771-2)
- 128 Huson, D. H., Reinert, K., & Myers, E. W. (2002). The greedy path-merging algorithm for
129 contig scaffolding. *Journal of the ACM (JACM)*, 49(5), 603–615. [https://doi.org/10.
130 1145/585265.585267](https://doi.org/10.1145/585265.585267)
- 131 Ivar Grytten, K. D. R. (2020). *Kmer Mapper*. https://github.com/ivargr/kmer_mapper.

- 132 Lemane, T., Chikhi, R., & Peterlongo, P. (2022). Kmdiff, large-scale and user-friendly
133 differential k-mer analyses. *Bioinformatics*, 38(24), 5443–5445. <https://doi.org/10.1093/bioinformatics/btac689>
- 135 Lemane, T., Medvedev, P., Chikhi, R., & Peterlongo, P. (2022). Kmtricks: Efficient and
136 flexible construction of bloom filters for large sequencing data collections. *Bioinformatics
137 Advances*, 2(1), vbac029. <https://doi.org/10.1093/bioadv/vbac029>
- 138 Li, Z., Chen, Y., Mu, D., Yuan, J., Shi, Y., Zhang, H., Gan, J., Li, N., Hu, X., Liu,
139 B., & others. (2012). Comparison of the two major classes of assembly algorithms:
140 Overlap–layout–consensus and de-bruijn-graph. *Briefings in Functional Genomics*, 11(1),
141 25–37. <https://doi.org/10.1093/bfgp/elr035>
- 142 Marchet, C., Lecompte, L., Limasset, A., Bittner, L., & Peterlongo, P. (2020). A resource-frugal
143 probabilistic dictionary and applications in bioinformatics. *Discrete Applied Mathematics*,
144 274, 92–102. <https://doi.org/10.1016/j.dam.2018.03.035>
- 145 Monochromegane. (2018). *The Platinum Searcher*. https://github.com/monochromegane/the_platinum_searcher.
- 147 Plaza Onate, F., Batto, J.-M., Juste, C., Fadlallah, J., Fougeroux, C., Gouas, D., Pons, N.,
148 Kennedy, S., Levenez, F., Dore, J., & others. (2015). Quality control of microbiota
149 metagenomics by k-mer analysis. *BMC Genomics*, 16, 1–10. <https://doi.org/10.1186/s12864-015-1406-7>
- 151 Sarmashghi, S., Bohmann, K., P. Gilbert, M. T., Bafna, V., & Mirarab, S. (2019). Skmer:
152 Assembly-free and alignment-free sample identification using genome skims. *Genome
153 Biology*, 20, 1–20. <https://doi.org/10.1186/s13059-019-1632-4>
- 154 Schatz, M. C., Delcher, A. L., & Salzberg, S. L. (2010). Assembly of large genomes using
155 second-generation sequencing. *Genome Research*, 20(9), 1165–1173. <https://doi.org/10.1101/gr.101360.109>
- 157 Uricaru, R., Rizk, G., Lacroix, V., Quillary, E., Plantard, O., Chikhi, R., Lemaitre, C., &
158 Peterlongo, P. (2015). Reference-free detection of isolated SNPs. *Nucleic Acids Research*,
159 43(2), e11–e11. <https://doi.org/10.1093/nar/gku1187>
- 160 Wood, D. E., Lu, J., & Langmead, B. (2019). Improved metagenomic analysis with kraken 2.
161 *Genome Biology*, 20, 1–13. <https://doi.org/10.1186/s13059-019-1891-0>
- 162 Zhang, Z., & Wang, W. (2014). RNA-skim: A rapid method for RNA-seq quantification at tran-
163 script level. *Bioinformatics*, 30(12), i283–i292. <https://doi.org/10.1093/bioinformatics/btu288>