nature computational science

Brief Communication

https://doi.org/10.1038/s43588-024-00596-6

Indexing and real-time user-friendly queries in terabyte-sized complex genomic datasets with kmindex and ORA

Received: 12 July 2023

Accepted: 16 January 2024

Published online: 26 February 2024

Check for updates

Téo Lemane[1,2] ✉, Nolan Lezzoche[3], Julien Lecubin[4], Eric Pelletier[2,5], Magali Lescot[3,5], Rayan Chikhi[6] & Pierre Peterlongo[1] ✉

Public sequencing databases contain vast amounts of biological information, yet they are largely underutilized as it is challenging to efficiently search them for any sequence(s) of interest. We present kmindex, an approach that can index thousands of metagenomes and perform sequence searches in a fraction of a second. The index construction is an order of magnitude faster than previous methods, while search times are two orders of magnitude faster. With negligible false positive rates below 0.01%, kmindex outperforms the precision of existing approaches by four orders of magnitude. Here we demonstrate the scalability of kmindex by successfully indexing 1,393 marine seawater metagenome samples from the *Tara* Oceans project. Additionally, we introduce the publicly accessible web server Ocean Read Atlas, which enables real-time queries on the *Tara* Oceans dataset.

Public genomic datasets are growing at an exponential rate. They contain treasures of genomic information that enable groundbreaking discoveries in fundamental domains such as agronomy, ecology and health[1,2]. Unfortunately, despite their public availability in repositories such as the Sequence Read Archive[3], these resources, measured in petabytes, are rarely ever reused globally because they cannot be searched efficiently. Many methodological developments toward sequencing data search engines have been introduced in recent years[4,5]. Current methods for searching genomic sequencing data look for *k*-mers (words of fixed length *k*, with *k* usually in [20; 50]) shared between a query sequence and each sample present in a reference database. The central operation is thus to determine, for each *k*-mer, in which indexed sample(s) it occurs.

In this Brief Communication, we focus on the challenge of indexing and querying large and complex metagenomic datasets. By 'complex' we arbitrarily design any dataset totaling over a hundred billion distinct *k*-mers. This is frequently observed in environmental metagenomic projects. Given the data size, that is, thousands of samples totaling tens of terabytes of compressed data, and its complexity, that is, thousands of billions of distinct *k*-mers, the computational challenge is immense. Once *k*-mers are extracted from raw data and filtered, a data structure is built to associate each *k*-mer to the sample(s) in which it occurs.

Techniques for associating *k*-mers to samples can be divided into three categories: sketching approaches that heavily subsample *k*-mers, exact data structures storing all *k*-mers and approximate membership data structures such as Bloom filters (BFs). Sketching approaches such as sourmash[6] or Needle[7] typically suffer from high false negative rates when short sequences are queried and are thus out of the scope of this work. Methods based on exact representations (for example, MetaGraph[8], BiFrost[9] and ggcat[10]) suffer from low scalability, as highlighted by our results. We are thus left with methods based on BFs[11], such as COBS[12] and SBT[13], later improved by HowDeSBT[14] and more recently by MetaProFi[15], which is able to index billions of *k*-mers using only a few dozen of gigabytes of space.

[1]Univ. Rennes, Inria, CNRS, IRISA - UMR 6074, Rennes, France. [2]Génomique Métabolique, Genoscope, Institut de Biologie François Jacob, CEA, CNRS, Univ. Evry, Université Paris-Saclay, Evry, France. [3]Aix-Marseille Université, Université de Toulon, IRD, CNRS, Mediterranean Institute of Oceanography (MIO), UM 110, Marseille, France. [4]SIP, OSU PYTHEAS, Marseille, France. [5]Research Federation for the Study of Global Ocean Systems Ecology and Evolution, FR2022/Tara Oceans GO-SEE, CNRS, Paris, France. [6]Institut Pasteur, Université Paris Cité, G5 Sequence Bioinformatics, Paris, France. ✉e-mail: teo.lemane@genoscope.cns.fr; pierre.peterlongo@inria.fr

https://doi.org/10.1038/s43588-024-00596-6

Table 1 | Overview of index construction and read query performance of kmindex compared to MetaProFi and COBS, on 50 *Tara* Ocean samples

		Build index			Query time		FP rate (%)	
		RAM	Disk	Index size	Number of queried reads			
	Time	GB	GB	GB	1	10 million	Average	Maximum
MetaProFi	30 h 15 min	278	5,684	226	12.72 s	1 h 29 min	11.18	21.55
COBS	26 h 30 min	278	5,684	184	1.51 s	15 h 56 min	13.29	24.60
kmindex	**2 h 56 min**	**107**	**878**	**164**	**0.06 s**	**4 min 21 s**	**0.006**	**0.18**

These are the only tools that succeeded in building an index and perform queries. The 'RAM' and 'Disk' columns provide the peak usage during the building process. The COBS and MetaProFi RAM and disk peaks are identical as they correspond to the same *k*-mer counting and filtration step. Queries are composed of one read and 10 million reads uniformly sampled from the 50 *Tara* Oceans datasets. All executions were performed on a cold cache. Extended results are presented in Supplementary Section 1. Bold texts highlight the best values.

When indexing large and complex metagenomic datasets, existing tools face important limitations in either disk usage, memory usage, computation time (either during indexing and/or query), false positive (FP) rate or false negative rate. Overcoming all these limitations simultaneously makes the design of an efficient data indexing strategy particularly challenging. We present kmindex, a tool that performs indexing and queries using orders of magnitude fewer resources than previous approaches. Also, kmindex provides results with no false negative calls and with negligible FP rates, approximately four orders of magnitude smaller than those obtained by other tools. kmindex is primarily designed for indexing complex sequencing samples. Owing to engineering choices, it is currently not suited for indexing large collections of genomes (that is, hundreds of thousands of samples).

To showcase the features of kmindex on a dataset of high biological interest, we introduce a web server named 'Ocean Read Atlas' (ORA) available at ref. 16. ORA allows to search one or several sequences across all of *Tara* Oceans metagenomic raw sequencing data[17]. It enables the visualization of the results on a geographic map and their correlation with each of the 56 environmental variables collected during the circumnavigation campaign. The ORA server enables to perform instant searches on a large and complex dataset, providing new perspectives on the deep exploitation of *Tara* Oceans resources.

We evaluated the performances of kmindex together with eight state-of-the-art *k*-mer indexers: themisto[18], ggcat[10], HIBF[19], PAC[20], MetaProFi[15], MetaGraph[8], Bifrost[9] and COBS[12]. The dataset for this benchmark is composed of metagenomic seawater sequencing data from 50 *Tara* Oceans samples, of 1.4 TB of gzipped fastq files. It contains approximately 1,420 billion *k*-mers. Among them, approximately 394 billion are distinct, and 132 billion occur twice or more.

The benchmarking setup is described in 'Benchmark setup' section in Methods. The results of all the following claims are described in Supplementary Section 1.

kmindex has better index construction performance

Among the nine tested tools, only MetaProFi, COBS and kmindex completed the index creation phase and were able to perform queries correctly. As shown in Table 1, building an index with kmindex is an order of magnitude faster than MetaProFi and COBS, and uses 2.6× less memory and 6.5× less disk. The final index sizes are all within the same magnitude range, with the smallest one produced by kmindex. The kmindex construction took less than 3 h, a peak random-access memory (RAM) of 107 GB and a peak disk usage of 878 GB.

kmindex enables real-time queries

As shown in Table 1, at query time, kmindex outperforms MetaProFi and COBS in terms of both computation time and memory resources (Supplementary Section 1.2). kmindex is between 20 and 200 times faster than MetaProFi and COBS for querying one read or millions of reads. kmindex is capable of performing millions of queries in a matter of minutes while allowing real-time resolution for small queries. This opens the doors to analyzing complete read sets as queries, and the deployment of real-time query servers as presented in the next section. Of note, kmindex also offers a 'fast mode', presented in Supplementary Section 1.5, that uses more RAM to achieve even faster queries.

kmindex allows highly accurate queries

The kmindex, MetaProFi and COBS scalability is achieved thanks to the usage of BFs that generate FP calls at query time. FP rate analyses, summarized in Table 1, show that, for a similar index size, MetaProFi and COBS present sensible FP hits, on average 11.18% and 13.29%, respectively, over the 50 answers (one per indexed sample). In contrast, the kmindex FP rate is negligible (below 10^{-2}% on average).

kmindex provides a high level of usability

kmindex enables to add new samples to an index. A novel and independent index can be registered with a previous one. At query time, each registered index is queried independently. This offers the possibility to query only a subset of the registered indexes. This is well adapted when indexing samples with distinct characteristics. Alternatively, users can extend an existing index, and the parameters of the previous index (such as the ad-hoc hash function or the BF sizes) are automatically reused. This second choice is less flexible but provides better performances at query time (see results presented in Supplementary Section 1.5). Also, kmindex enables the filtration of erroneous *k*-mers, not only relying on their abundance in a dataset but also on their co-abundances in all indexed datasets. This enables to 'rescue' low-abundance *k*-mers that would have otherwise been removed. To the best of our knowledge, no other indexing tool can integrate this feature. This feature is inherited from the kmtricks[21] algorithm.

kmindex query results can be provided with various degrees of precision. For each indexed sample, users can access the average similarity of queried sequences or a similarity value per queried sequence. kmindex can provide the distribution of hits, enabling to highlight some regions of interest among the queried sequences. Finally, kmindex is well documented and simple to install. Queries can be performed via a command line Interface, via an application programming interface or a hypertext transfer protocol (HTTP) server.

Indexing 1,393 *Tara* Oceans samples in the ORA web server

With kmindex, we built and made available a public web interface able to perform queries on a dataset composed of 1,393 samples (distinct locations and distinct fraction sizes) of the *Tara* Oceans project[17] representing 36.7 TB of raw fastq.gz files. A user can query sequences, determining their similarity with the 1,393 indexed samples. A world map depicts the resulting biogeography, as well as the environmental parameters associated with the sequences.

Note that, for reasons of robustness and continuity of service, the index is deployed on a networked and redundant filesystem with lower performances compared to the benchmark environment, although suitable for this type of service. Details about indexed read sets, and more information about the server architecture and setup, are provided in 'The ORA server' section in Methods.

The resulting web server named ORA, whose representation is provided in Supplementary Fig. 2, extends the 'Ocean Gene Atlas' server[22,23] that supports queries to assembled genes from *Tara* Oceans[17] and Malaspina[24]. We believe this server will be of great importance to the *Tara* Oceans consortium as a whole, and more broadly to anybody interested in marine genetic data.

In conclusion, we present kmindex, which could open up a new channel for leveraging genetic data, removing the obstacles that often isolate studies from each other. Currently, kmindex only indexes the presence of a *k*-mer in a given sample. Hence, there is room for improvement by allowing the indexing of the abundance of each *k*-mer in each sample. The ability to perform fast queries is achieved by not compressing the data structure. As a result, the indexes are approximately 10% the size of the compressed input data, which can be a major limitation to scale-up to petabyte-size datasets. An area for future research will involve developing compression schemes specifically tailored to this framework.

Methods

Conceptually, the presence of each indexed *k*-mer is stored in one BF per input read set. The BF construction relies on kmtricks[21], which allows to filter erroneous *k*-mers and to efficiently build a partitioned matrix of BFs. Each partition indexes a subset of *k*-mers corresponding to a specific set of minimizers. In practice, with kmindex, matrices are inverted to limit cache misses during the query process, that is, each row is a bit vector representing the presence/absence of a *k*-mer in each indexed sample. At query time, *k*-mers from queried sequences are grouped into batches, and, avoiding cache misses, BFs are queried to determine the presence or absence of each *k*-mer in each input dataset.

Index construction

The construction of BFs from raw sequencing data is delegated to kmtricks[21], which allows partitioned construction of one-hash BF matrices. For each input dataset, *k*-mers are counted and filtered on the basis of their abundance. Additionally, and contrary to other methods, during one of the index-building steps, for a given *k*-mer, its abundance in all datasets is known. This offers the possibility to conserve a *k*-mer having an abundance lower than the fixed threshold, which would be filtered out by other methods, but having some occurrences in the other datasets. This may reflect the presence in one of the samples of a low-abundant species for which we want to preserve the data. Once *k*-mers are filtered, submatrices are built. Each submatrix indexes a subset of *k*-mers matching a specific set of minimizers.

As represented in Fig. 1 (right), the resulting index built by kmindex consists of P distinct matrices (with P being the number of partitions, equal to 3 in the figure). To save indexing and query time, the index is 'inverted': given a *k*-mer, the N bits indicating its presence/absence in the N indexed datasets are consecutive in the index. This allows for fast queries across numerous datasets. Hence, in practice, in a matrix, each row is a bit vector representing the presence or absence of a hash value in each indexed sample. Note that the rows are not packed to save construction and query time. This results in the fact that each row is composed of $\left\lceil \frac{N}{8} \right\rceil \times 8$ bits. Doing so, $\min(0, 8 - N \bmod 8)$ bits are unused for each row, as represented by a double arrow in Fig. 1. This is up to 7 bits per row. These few lost bits may appear as a drawback, but this is negligible regarding the N value that is meant to be in the order of a few hundred or thousands, and, importantly, this enables us to efficiently append novel indexed samples to an existing index.

By default, the resulting index is not compressed. Although requiring more space, this ensures optimal access time (both for writing and reading), and it offers the possibility to dynamically append new datasets to an existing index.

Index query

The query process introduced in kmindex is also sketched in Fig. 1. Batch processing is used for queries. This allows maximum throughput while maintaining control over memory usage. The user can specify the batch size and the maximum number of parallel batches according to the system's capabilities.

The resolution of a batch proceeds as follows:

1. Bucketing. The index is organized by partition, each corresponding to a set of minimizers. The first step consists of splitting query sequences into *k*-mers, which are then hashed and inserted into the right partition according to their minimizers. Each *k*-mer partition of the batch can then be solved by querying the corresponding index partition.
2. Sorting. Each partition is sorted to enable its resolution in a single sequential pass on the corresponding index partition, reducing cache misses.
3. *k*-Mer level resolution. Querying a specific *k*-mer consists of fetching the row that corresponds to its hash value in the index to retrieve the bit vector corresponding to its presence or absence in each sample. For each query, the response vectors are aggregated by summation, resulting in an integer vector that represents the number of positive hits in each indexed sample. Obtaining the response vector for each *k*-mer is the current bottleneck because of input/output (I/O) operations. For this reason, instead of loading the index into memory, index partitions are read through memory-mapped files. This allows reading only the parts of the index that are relevant to the batch resolution, which is particularly beneficial in the case of small queries.
   The memory-mapped files can be managed in two different ways. Normal mode: each batch manages its own mappings of index partitions. The mapping of a partition is closed as soon as all *k*-mers belonging to the partition are resolved. The cached pages are then marked as available for eviction, resulting in lower memory usage (see results presented in Supplementary Section 1.4). Fast mode: all batches share the same mappings. This way, a larger number of pages are kept in the cache when conditions are favorable, that is, without memory pressure, avoiding possible new I/O operations when solving the remaining batches. The memory usage may therefore seem high due to important page caching, up to the size of the index in the context of large query sets. Note that both modes require the same minimum amount of memory; the other part of the memory usage corresponds only to page caching, which is automatically managed by the kernel. In other words, under memory pressure, both modes show the same memory usage.
4. Sequence-level resolution. Finally, a result file is generated in either 'json' or 'tsv' format depending on the user's choice. Query results are filtered on the basis of the threshold specified by the user. The user can also request the distribution of hits along the query sequences, represented as a binary vector.

Reducing the FP rate

The kmindex algorithm embeds the findere approach[25]. findere enables a drastic reduction of the FP rate when querying successive *k*-mers from a query while using an approximate membership query (AMQ) data structure (such as BFs) for indexing. For each indexed dataset, the central idea consists in indexing its *s*-mers instead of its *k*-mers in the AMQ, with $s \leq k$. At query time, a *k*-mer is considered as existing in the dataset if all its $k - s + 1$ constituent *s*-mers are reported as present by the AMQ. In the general case, using this approach, a *k*-mer is wrongly reported as present (a FP) when all its *s*-mers are themselves FPs. This has the effect of exponentially decreasing the FP rate with respect to the $k - s$ value. When querying *k*-mers from a sequence of length n, in the general case, $n - k + 1$ calls to the AMQ have to be made. Using findere, $n - s + 1$ calls must be made ($k - s$ more than without using findere). This has negligible and no measurable impact on query time.

https://doi.org/10.1038/s43588-024-00596-6

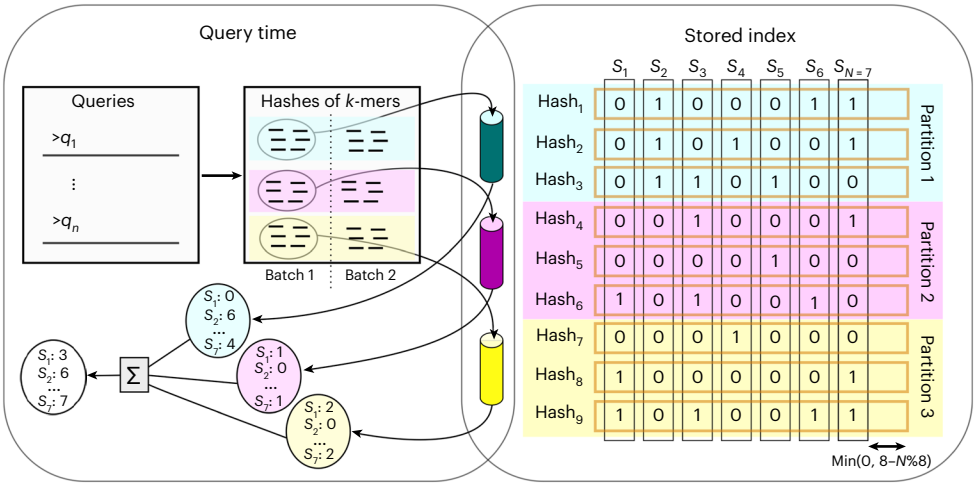

Fig. 1 | kmindex: an overview of the data structure and the query process. Right: data structure showing the stored index on disk. Vertical black rectangles represent the *N* BFs (one per input sample). The orange horizontal rectangles represent the actual data structure saved on disk, storing consecutively the 0/1 values of distinct BFs for the same hash value. The three colored horizontal rectangles represent three partitions, each saved in a distinct file. Left: query example. Hashes of *k*-mers are symbolized by small horizontal lines, divided into two batches and grouped by partitions. Each group is sorted. Each cylinder represents the streaming of a set of hashed *k*-mers, querying lines of BFs mapped into memory. For simplicity, the image shows only queries from hashed *k*-mers from one of the two represented batches. In practice, for each partition, all batches are queried. For each sample, the results from each partition and each batch are finally summed up (as symbolized by the 'Σ' symbol in this figure). Also for simplicity, this figure does not represent the use case in which the distribution of hits along the query sequences is reported as a binary vector.

Benchmark setup

In situations involving large indexes and queries, various factors such as I/O operations or caching can impact performance. To account for these effects, we performed the benchmarks from a user's perspective. As a result, all measurements are obtained using the command line tools with particular attention to caching effects. The reported values include input parsing, query execution and output writing. The results, presented in Supplementary Section 1.4, demonstrate the performance in a cold (the most likely and also the least favorable) or warm cache context. We recall that the dataset for this benchmark is composed of metagenomic seawater sequencing data from 50 *Tara* Oceans samples, of 1.4 TB of gzipped fastq files. It contains approximately 1,420 billion *k*-mers. Among them, approximately 394 billion are distinct, and 132 billion occur twice or more. These data are publicly available.

Executions were performed on the GenOuest platform on a node with 64 cores (128 threads) Xeon 2.2 GHz (L1, 48 KB; L2, 1.25 MB; L3, 48 MB shared) with 900 GB of memory. All computations are performed on an xfs filesystem allowing 1,052 MB s^{-1} sequential reads, 473 MB s^{-1} sequential writes and 908 MB s^{-1} random reads (throughput measurements are obtained using fio[26]). All tested tools were parameterized to use 32 threads.

See 'Code availability' section for information about used commands, data accession identifiers, input random sequence and output files.

The ORA server

Dataset. The ORA[27] index is composed of 1,393 samples (distinct locations and distinct fraction sizes) of the *Tara* Oceans project. These samples are divided into six distinct groups, determined by the size fraction of the sequenced species. These fractions correspond to the physical filter sizes used during the sampling campaign. Based on this clustering we built six distinct indexes (all with the same parameters). At query time, as all the six indexes are registered in a unique meta-index, the whole set of samples is queried. A description of the dataset is available in Extended Data Table 1. The size of the final uncompressed index is approximately 13% of the size of the raw fastq.gz files, which is 36.7 TB.

Sequencing data availability. Shotgun metagenomic sequences of all the samples from the *Tara* Oceans Expedition (2009–2013) are available at the European Nucleotide Archive[28] under global accession number PRJEB402 (PRJEB1787 and PRJEB9740 for bacteria and archaea, PRJEB1788 for giant viruses, PRJEB4352 and PRJEB9691 for protists, and PRJEB4419 and PRJEB9742 for DNA viruses).

Environmental data. The environmental data are from the *Tara* Oceans Expedition (2009–2013) and are available on PANGAEA[29]. The environmental database contains currently:

- BIODIV[30]
- CARB[31]
- HPLC[32]
- MESOSCALE[33]
- NUT[34]
- SENSORS[35]
- SEQUENCING[36]
- WATERCOLUMN[37]

ORA server workflow. Supplementary Fig. 1 shows an overview of the ORA server.

The ORA service is composed of three parts:

1. The environmental database containing the parameters measured and estimated during the sampling of the *Tara* Oceans Expedition;
2. The kmindex server to query the kmindex index made with sequencing reads from *Tara* Oceans Expedition via HTTP requests;
3. The ORA server to query the index and make the link between *k*-mers (contained in the query sequence) and the environmental parameters of samples sharing these *k*-mers. ORA provides results on a webpage including maps, plots and table files.

A representation of these components is available in Supplementary Fig. 2.

kmindex server. The index is stored on a Ceph storage cluster (SSD pool). The CRUSH algorithm enables the Ceph storage cluster to scale, rebalance and recover data dynamically. Nodes of this storage cluster are on a redundant network using Link Aggregation Control Protocol trunking configuration.

The kmindex HTTP server runs in a Qemu/KVM virtual machine (VM) with 16 cores and 32 GB RAM, supporting 16 concurrent queries. VM is stored on a Proxmox virtualization cluster with HA capabilities. Depending on our available resources, storage and VM, capacities may be expanded if needed. This infrastructure is mandatory to ensure service continuity.

Supplementary Fig. 2 represents the overall ORA workflow.

About ORA usages and limitations. The service currently supports unique query of a FASTA file limited to 10 kilobase pairs via the web interface. Each user is limited to 200 jobs per 24 h. The results can either be delivered directly or sent by email with a link valid for 2 weeks. In the future and depending on our computational capacity, we expect to offer an application programming interface with more features such as the integration of the query abundance and metadata associated with target sequences.

A user guide manual is available at ref. 38, where the 'Interfaces' section[39] provides a detailed explanation about the submission and results interfaces. Users can contact us by sending an email to ocean-readatlas@mio.osupytheas.fr.

Reporting summary

Further information on research design is available in the Nature Portfolio Reporting Summary linked to this article.

Data availability

A list of publicly available data used in this work is presented in the https://github.com/pierrepeterlongo/kmindex_benchmarks repository[40].

Code availability

kmindex is an open-source software available at https://github.com/tlemane/kmindex (ref. 41). The documentation is available at https://tlemane.github.io/kmindex/. The exhaustive list of tool versions and commands used are presented in a companion website[40], which also reports the FP computation protocols and a detailed description of the dataset considered for this benchmark. The ORA server code is available through a GitLab repository[27].

References

1. Edgar, R. C. et al. Petabase-scale sequence alignment catalyses viral discovery. *Nature* **602**, 142–147 (2022).
2. Paoli, L. et al. Biosynthetic potential of the global ocean microbiome. *Nature* **607**, 111–118 (2022).
3. Katz, K. et al. The Sequence Read Archive: a decade more of explosive growth. *Nucleic Acids Res.* **50**, D387–D390 (2022).
4. Chikhi, R., Holub, J. & Medvedev, P. Data structures to represent a set of *k*-long DNA sequences. *ACM Comput. Surv.* **54**, 1–22 (2021).
5. Marchet, C. et al. Data structures based on *k*-mers for querying large collections of sequencing data sets. *Genome Res.* **31**, 1–12 (2021).
6. Pierce, N. T., Irber, L., Reiter, T., Brooks, P. & Brown, C. T. Large-scale sequence comparisons with sourmash. *F1000Research* **8**, 1006 (2019).
7. Darvish, M., Seiler, E., Mehringer, S., Rahn, René & Reinert, K. Needle: a fast and space-efficient prefilter for estimating the quantification of very large collections of expression experiments. *Bioinformatics* **38**, 4100–4108 (2022).
8. Karasikov, M. et al. Metagraph: indexing and analysing nucleotide archives at petabase-scale. Preprint at *bioRxiv* https://doi.org/10.1101/2020.10.01.322164 (2020).
9. Holley, G. & Melsted, P. áll Bifrost: highly parallel construction and indexing of colored and compacted de Bruijn graphs. *Genome Biol.* **21**, 249 (2020).
10. Cracco, A. & Tomescu, A. I. Extremely fast construction and querying of compacted and colored de Bruijn graphs with ggcat. *Genome Res.* **33**, 1198–1207 (2023).
11. Bloom, B. H. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM* **13**, 422–426 (1970).
12. Bingmann, T., Bradley, P., Gauger, F. & Iqbal, Z. COBS: a Compact Bit-Sliced Signature index. String Processing and Information Retrieval, SPIRE 2019. In *Lecture Notes in Computer Science*, Vol. 11811 (Springer, Cham, 2019).
13. Solomon, B. & Kingsford, C. Improved search of large transcriptomic sequencing databases using split sequence Bloom trees. *J. Comput. Biol.* **25**, 755–765 (2018).
14. Harris, R. S. & Medvedev, P. Improved representation of sequence Bloom trees. *Bioinformatics* **36**, 721–727 (2020).
15. Srikakulam, S. K., Keller, S., Dabbaghie, F., Bals, R. & Kalinina, O. V. Metaprofi: an ultrafast chunked Bloom filter for storing and querying protein and nucleotide sequence data for accurate identification of functionally relevant genetic variants. *Bioinformatics* **39**, btad101 (2023).
16. The Ocean Read Atlas. *OSU Institut Pytheas* https://ocean-read-atlas.mio.osupytheas.fr/ (2023).
17. Sunagawa, S. et al. *Tara* Oceans: towards global ocean ecosystems biology. *Nat. Rev. Microbiol.* **18**, 428–445 (2020).
18. Alanko, J. N., Vuohtoniemi, J., Mäklin, T. & Puglisi, S. J. Themisto: a scalable colored *k*-mer index for sensitive pseudoalignment against hundreds of thousands of bacterial genomes. *Bioinformatics* **39**, i260–i269 (2023).
19. Mehringer, S. et al. Hierarchical interleaved Bloom filter: enabling ultrafast, approximate sequence queries. *Genome Biol.* **24**, 131 (2023).
20. Marchet, C. & Limasset, A. Scalable sequence database search using partitioned aggregated Bloom comb trees. *Bioinformatics* **39**, i252–i259 (2023).
21. Lemane, T., Medvedev, P., Chikhi, R. & Peterlongo, P. kmtricks: efficient and flexible construction of Bloom filters for large sequencing data collections. *Bioinform. Adv.* **2**, vbac029 (2022).
22. Villar, E. et al. The Ocean Gene Atlas: exploring the biogeography of plankton genes online. *Nucleic Acids Res.* **46**, W289–W295 (2018).
23. Vernette, C. et al. The Ocean Gene Atlas v2. 0: online exploration of the biogeography and phylogeny of plankton genes. *Nucleic Acids Res.* **50**, W516–W526 (2022).
24. Acinas, S. G. et al. Deep ocean metagenomes provide insight into the metabolic architecture of bathypelagic microbial communities. *Commun. Biol.* **4**, 604 (2021).
25. Robidou, L. & Peterlongo, P. findere: fast and precise approximate membership query. In *International Symposium on String Processing and Information Retrieval* 151–163 (Springer, 2021).
26. fio. *GitHub* https://github.com/axboe/fio (2023).
27. DOI of the provided ORA server GitLab code. *Zenodo* https://doi.org/10.5281/zenodo.10462412 (2024).
28. European Nucleotide Archive. *European Bioinformatics Institute* https://www.ebi.ac.uk/ena/ (2023).
29. Tara Oceans Consortium, Coordinators; Tara Oceans Expedition, Participants. Registry of all samples from the Tara Oceans Expedition (2009–2013). *PANGAEA* https://doi.org/10.1594/PANGAEA.875582 (2017).
30. Guidi, L., Gattuso, J.-P. & Pesant, S. Tara Oceans Consortium, Coordinators; Tara Oceans Expedition, Participants. Environmental context of all samples from the Tara Oceans Expedition (2009–2013), about carbonate chemistry in the targeted environmental feature. *PANGAEA* https://doi.org/10.1594/PANGAEA.875567 (2017).
31. Tara Oceans Consortium, Coordinators; Tara Oceans Expedition, Participants. Biodiversity context of all samples from the Tara Oceans Expedition (2009–2013). *PANGAEA* https://doi.org/10.1594/PANGAEA.853809 (2015).

32. Guidi, L. et al. Tara Oceans Consortium, Coordinators; Tara Oceans Expedition, Participants. Environmental context of all samples from the Tara Oceans Expedition (2009–2013), about pigment concentrations (HPLC) in the targeted environmental feature. *PANGAEA* https://doi.org/10.1594/PANGAEA.875569 (2017).
33. Ardyna, M. et al. Tara Oceans Consortium, Coordinators; Tara Oceans Expedition, Participants. Environmental context of all samples from the Tara Oceans Expedition (2009–2013), about mesoscale features at the sampling location. *PANGAEA* https://doi.org/10.1594/PANGAEA.875577 (2017).
34. Guidi, L. et al. Tara Oceans Consortium, Coordinators; Tara Oceans Expedition, Participants. Environmental context of all samples from the Tara Oceans Expedition (2009–2013), about nutrients in the targeted environmental feature. *PANGAEA* https://doi.org/10.1594/PANGAEA.875575 (2017).
35. Guidi, L., Picheral, M. & Pesant, S. Tara Oceans Consortium, Coordinators; Tara Oceans Expedition, Participants. Environmental context of all samples from the Tara Oceans Expedition (2009–2013), about sensor data in the targeted environmental feature. *PANGAEA* https://doi.org/10.1594/PANGAEA.875576 (2017).
36. Alberti, A. & Pesant, S. Tara Oceans Consortium, Coordinators; Tara Oceans Expedition, Participants. Methodology used in the lab for molecular analyses and links to the Sequence Read Archive of selected samples from the Tara Oceans Expedition (2009–2013). *PANGAEA* https://doi.org/10.1594/PANGAEA.875581 (2017).
37. Speich, S. et al. Tara Oceans Consortium, Coordinators; Tara Oceans Expedition, Participants. Environmental context of all samples from the Tara Oceans Expedition (2009–2013), about the water column features at the sampling location. *PANGAEA* https://doi.org/10.1594/PANGAEA.875579 (2017).
38. Overview. *Ocean Read Atlas* https://ora.mio.osupytheas.fr/manual/pages/ (2023).
39. Interfaces. *Ocean Read Atlas* https://ora.mio.osupytheas.fr/manual/pages/interfaces.html (2023).
40. pierrepeterlongo/kmindex_benchmarks: initial release. *Zenodo* https://doi.org/10.5281/zenodo.10462379 (2024).
41. DOI of the kmindex GitHub repository. *Zenodo* https://doi.org/10.5281/zenodo.10462427 (2024).

Acknowledgements

We acknowledge the GenOuest core facility (https://www.genouest.org) and the TGCC (https://www-hpc.cea.fr/index-en.html) for providing the computing infrastructure, as well as France Génomique for funding of the TGCC computing resources used to process data used in this article. The authors thank J.-M. Aury for his help regarding the usage of the *Tara* Oceans datasets. *Tara* Oceans (which includes both the *Tara* Oceans and *Tara* Oceans Polar Circle expeditions) would not exist without the leadership of the *Tara* Ocean Foundation and the continuous support of *Tara* Oceans consortium members. The authors also thank K. Andre and M. Harun for their help regarding the usage of MetaGraph, A. Cracco and A. Tomescu for their help using ggcat, and C. Marchet and A. Limasset for their support using PAC. The web server is hosted by the OSU Pythéas cluster with the help of C. Blanpain and SIP members. A. Malgoyre from SIP is thanked for the development of the OSU Pythéas GitLab. The work was funded by ANR SeqDigger (ANR-19-CE45-0008) and the IPL Inria Neuromarkers, and received some support from the French government under the France 2030 investment plan, as part of the Initiative d'Excellence d'Aix-Marseille Université - A*MIDEX - Institute of Ocean Sciences (AMX-19-IET-016). This work is part of the ALPACA project that has received funding from the European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement nos. 956229 and 872539 (PANGAIA). R.C. was supported by ANR Full-RNA, Inception and PRAIRIE grants (ANR-22-CE45-0007, PIA/ANR16-CONV-0005 and ANR-19-P3IA-0001). The funders had no role in study design, data collection and analysis, decision to publish or preparation of the manuscript.

Author contributions

T.L., E.P., R.C. and P.P. have conceptualized the project. T.L., R.C. and P.P. developed the methodology. T.L. implemented the software. T.L. and P.P. conceived and conducted the experiments. M.L. and E.P. provided the data. T.L., R.C. and P.P. wrote the manuscript. N.L., J.L. and M.L. implemented and deployed the ORA server. R.C. and P.P. supervised the work. M.L., R.C. and P.P. obtained the funding. All authors reviewed the manuscript.

Competing interests

The authors declare no competing interests.

Additional information

Extended data is available for this paper at https://doi.org/10.1038/s43588-024-00596-6.

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s43588-024-00596-6.

Correspondence and requests for materials should be addressed to Téo Lemane or Pierre Peterlongo.

Peer review information *Nature Computational Science* thanks Natapol Pornputtapong, Guohua Wang and the other, anonymous, reviewer(s) for their contribution to the peer review of this work. Peer reviewer reports are available. Primary Handling Editor: Ananya Rastogi, in collaboration with the *Nature Computational Science* team.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.

© The Author(s), under exclusive licence to Springer Nature America, Inc. 2024

https://doi.org/10.1038/s43588-024-00596-6

Extended Data Table 1 | Description of the indexed dataset organized by size fraction. The "Fraction size" column indicates the size range of the target sequenced species

Fraction size	Number of samples	Average number of distinct k-mers per sample
Integrated $(0.8-2000\mu m)$	193	14.5e9
Meso $(180-2000\mu m)$	208	13.2e9
Micro $(20-280\mu m)$	195	15e9
Nano $(3-20\mu m)$	213	16.1e9
Pico $(0.2-5\mu m)$	425	11.1e9
Virus $(<0.2\mu m)$	159	2.3e8

nature portfolio

Corresponding author(s): Téo Lemane
Pierre Peterlongo

Last updated by author(s): Jan 7, 2024

Reporting Summary

Nature Portfolio wishes to improve the reproducibility of the work that we publish. This form provides structure for consistency and transparency in reporting. For further information on Nature Portfolio policies, see our Editorial Policies and the Editorial Policy Checklist.

Statistics

For all statistical analyses, confirm that the following items are present in the figure legend, table legend, main text, or Methods section.

n/a	Confirmed
☒	☐ The exact sample size (*n*) for each experimental group/condition, given as a discrete number and unit of measurement
☒	☐ A statement on whether measurements were taken from distinct samples or whether the same sample was measured repeatedly
☒	☐ The statistical test(s) used AND whether they are one- or two-sided *Only common tests should be described solely by name; describe more complex techniques in the Methods section.*
☒	☐ A description of all covariates tested
☒	☐ A description of any assumptions or corrections, such as tests of normality and adjustment for multiple comparisons
☒	☐ A full description of the statistical parameters including central tendency (e.g. means) or other basic estimates (e.g. regression coefficient) AND variation (e.g. standard deviation) or associated estimates of uncertainty (e.g. confidence intervals)
☒	☐ For null hypothesis testing, the test statistic (e.g. *F*, *t*, *r*) with confidence intervals, effect sizes, degrees of freedom and *P* value noted *Give P values as exact values whenever suitable.*
☒	☐ For Bayesian analysis, information on the choice of priors and Markov chain Monte Carlo settings
☒	☐ For hierarchical and complex designs, identification of the appropriate level for tests and full reporting of outcomes
☒	☐ Estimates of effect sizes (e.g. Cohen's *d*, Pearson's *r*), indicating how they were calculated

Our web collection on statistics for biologists contains articles on many of the points above.

Software and code

Policy information about availability of computer code

Data collection	All data were collected using sample identifiers and the SRA Toolkit (https://github.com/ncbi/sra-tools). Details are available at https://github.com/pierrepeterlongo/kmindex_benchmarks
Data analysis	kmindex is an open-source software available at https://github.com/tlemane/kmindex (https://doi.org/10.5281/zenodo.10462427). The documentation is available at https://tlemane.github.io/kmindex. The ORA server code is available at https://gitlab.osupytheas.fr/ocean_atlas/ora (https://doi.org/10.5281/zenodo.10462412). All other tools used for benchmarks are listed at https://github.com/pierrepeterlongo/kmindex_benchmarks (https://doi.org/10.5281/zenodo.10462379).

For manuscripts utilizing custom algorithms or software that are central to the research but not yet described in published literature, software must be made available to editors and reviewers. We strongly encourage code deposition in a community repository (e.g. GitHub). See the Nature Portfolio guidelines for submitting code & software for further information.

Data

Policy information about availability of data

All manuscripts must include a data availability statement. This statement should provide the following information, where applicable:

- Accession codes, unique identifiers, or web links for publicly available datasets
- A description of any restrictions on data availability
- For clinical datasets or third party data, please ensure that the statement adheres to our policy

A list of publicly available data used in this work is proposed in the https://github.com/pierrepeterlongo/kmindex_benchmarks repository. (https://doi.org/10.5281/zenodo.10462379)

Human research participants

Policy information about studies involving human research participants and Sex and Gender in Research.

Reporting on sex and gender	N/A
Population characteristics	N/A
Recruitment	N/A
Ethics oversight	N/A

Note that full information on the approval of the study protocol must also be provided in the manuscript.

Field-specific reporting

Please select the one below that is the best fit for your research. If you are not sure, read the appropriate sections before making your selection.

☒ Life sciences ☐ Behavioural & social sciences ☐ Ecological, evolutionary & environmental sciences

For a reference copy of the document with all sections, see nature.com/documents/nr-reporting-summary-flat.pdf

Life sciences study design

All studies must disclose on these points even when the disclosure is negative.

Sample size	N/A
Data exclusions	N/A
Replication	N/A
Randomization	N/A
Blinding	N/A

Reporting for specific materials, systems and methods

We require information from authors about some types of materials, experimental systems and methods used in many studies. Here, indicate whether each material, system or method listed is relevant to your study. If you are not sure if a list item applies to your research, read the appropriate section before selecting a response.

Materials & experimental systems

n/a	Involved in the study
☒	☐ Antibodies
☒	☐ Eukaryotic cell lines
☒	☐ Palaeontology and archaeology
☒	☐ Animals and other organisms
☒	☐ Clinical data
☒	☐ Dual use research of concern

Methods

n/a	Involved in the study
☒	☐ ChIP-seq
☒	☐ Flow cytometry
☒	☐ MRI-based neuroimaging