nature methods

Article

https://doi.org/10.1038/s41592-025-02625-2

Efficient and robust search of microbial genomes via phylogenetic compression

Received: 8 June 2023

Accepted: 12 February 2025

Published online: 9 April 2025

Check for updates

Karel Břinda[1,2] ✉, Leandro Lima[3], Simone Pignotti[2,4], Natalia Quinones-Olvera[2], Kamil Salikhov[4], Rayan Chikhi[5], Gregory Kucherov[4], Zamin Iqbal[3,6] & Michael Baym[2] ✉

Comprehensive collections approaching millions of sequenced genomes have become central information sources in the life sciences. However, the rapid growth of these collections has made it effectively impossible to search these data using tools such as the Basic Local Alignment Search Tool (BLAST) and its successors. Here, we present a technique called phylogenetic compression, which uses evolutionary history to guide compression and efficiently search large collections of microbial genomes using existing algorithms and data structures. We show that, when applied to modern diverse collections approaching millions of genomes, lossless phylogenetic compression improves the compression ratios of assemblies, de Bruijn graphs and *k*-mer indexes by one to two orders of magnitude. Additionally, we develop a pipeline for a BLAST-like search over these phylogeny-compressed reference data, and demonstrate it can align genes, plasmids or entire sequencing experiments against all sequenced bacteria until 2019 on ordinary desktop computers within a few hours. Phylogenetic compression has broad applications in computational biology and may provide a fundamental design principle for future genomics infrastructure.

Comprehensive collections of genomes have become an invaluable resource for research across the life sciences. However, their exponential growth, exceeding improvements in computation, makes their storage, distribution and analysis increasingly cumbersome[1]. As a consequence, traditional search approaches, such as BLAST[2] and its successors, are becoming less effective with the available reference data, which poses a major challenge for organizations such as the National Center for Biotechnology Information (NCBI) or European Bioinformatics Institute (EBI) in maintaining the searchability of their repositories.

The keys to achieving search scalability are compressive approaches that aim to store and analyze genomes directly in the compressed domain[3,4]. Genomic data have low fractal dimension and entropy[5], offering the possibility of efficient search algorithms[5]. However, despite the progress in compression-related areas of computer science[4–15], it remains a practical challenge to compute parsimonious compressed representations of the exponentially growing public genome collections.

Microbial collections are particularly difficult to compress due to the huge number of genomes and their exceptional levels of genetic diversity, which reflect the billions of years of evolution across the domain. Even though substantial efforts have been made to construct comprehensive collections of all sequenced microbial genomes, such as the 661k assembly collection[16] (661k pre-2019 bacteria) and the BIGSIdata de Bruijn graph collection[17] (448k de Bruijn graphs of all pre-2017 bacterial and viral raw sequence), the resulting data archives and indexes range from hundreds of gigabytes (661k) to tens of terabytes (BIGSIdata). This scale exceeds the bandwidth, storage and data processing capacities of most users, making local computation on these data functionally impossible.

[1]Inria, Irisa, Univ. Rennes, Rennes, France. [2]Department of Biomedical Informatics, Harvard Medical School, Boston, MA, USA. [3]EMBL-EBI, Hinxton, UK. [4]LIGM, CNRS, Univ. Gustave Eiffel, Marne-la-Vallée, France. [5]Institut Pasteur, Univ. Paris Cité, G5 Sequence Bioinformatics, Paris, France. [6]Milner Centre for Evolution, University of Bath, Bath, UK. ✉e-mail: karel.brinda@inria.fr; baym@hms.harvard.edu

We reasoned that the redundancies among microbial genomes are efficiently predictable, as they reflect underlying processes that created the collection: evolution and sampling. While genomes in nature can accumulate substantial diversity through vertical and horizontal mutational processes, this process is functionally sparse, and at the same time subjected to selective pressures and drift that limit their overall entropy. The amount of sequenced diversity is further limited by selective biases due to culture and research or clinical interests, resulting in sequencing efforts being predominantly focused on narrow subparts of the tree of life, associated with model organisms and human pathogens[16]. Importantly, such subtrees have been shown to be efficiently compressible when considered in isolation, as low-diversity groups of oversampled phylogenetically related genomes, such as isolates of the same species under epidemiological surveillance[18,19]. This suggests that the compression of comprehensive collections could be informed by their evolutionary history, reducing the complex problem of general genome compression to the more tractable problem of local compression of phylogenetically grouped and ordered genomes.

Phylogenetic relatedness is effective at estimating the similarity and compressibility of microbial genomes and their data representations. The closer two genomes are phylogenetically, the closer they are likely to be in terms of mathematical similarity measures, such as the edit distance or *k*-mer distances[20], and thus also more compressible. Importantly, this principle holds not only for genomes, but also for de Bruijn graphs and many *k*-mer indexes. We reasoned that phylogenetic trees could be embedded into computational schemes to group similar data together, as a preprocessing step for boosting local compressibility of data. The well-known Burrows–Wheeler transform[21] has a similar purpose in a different context and similar ideas have been used for read and alignment compression[22–25]. Other related ideas have previously been used for scaling up metagenomic classification using taxonomic trees[26–29] and search in protein databases[30,31].

At present, the public version of BLAST is frequently used to identify the species of a given sequence by comparing it to exemplars, but it is practically impossible to align against all sequenced bacteria. Despite the increasing number of bacterial assemblies available in the NCBI repositories, the searchable fraction of bacteria is exponentially decreasing over time (Fig. 1a and Supplementary Note 1). This limits our ability to study bacteria in the context of their known diversity, as the gene content of different strains can vary substantially, and important hits can be missed due to the database being unrepresentative.

Here, we present a solution to the problem of searching vast libraries of microbial genomes: 'phylogenetic compression', a technique for an evolutionary-guided compression of arbitrarily sized genome collections. We show that the underlying evolutionary structure of microorganisms can be efficiently approximated and used as a guide for existing compression and indexing tools. Phylogenetic compression can then be applied to collections of assemblies, de Bruijn graphs and *k*-mer indexes, and run in parallel for efficient processing. The resulting compression yields benefits ranging from a quicker download (reducing Internet bandwidth and storage costs), to efficient search on personal computers. We show this by implementing BLAST-like search on all sequenced pre-2019 bacterial isolates, which allow us to align genes, plasmids and sequencing reads on an ordinary laptop or desktop computer within a few hours, a task that was infeasible with previous techniques.

Results

We developed a technique called phylogenetic compression for evolutionarily informed compression and search of microbial collections (Fig. 1; https://brinda.eu/mof/). Phylogenetic compression combines four ingredients (Fig. 1b): (1) clustering of samples into 'phylogenetically related groups', followed by (2) inference of a 'compressive phylogeny' that acts as a template for (3) 'data reordering', before (4) the application of a calibrated 'low-level compressor/indexer' (Methods). This general scheme can be instantiated to individual protocols for various data types as we show in Fig. 1c; for instance, a set of bacterial assemblies can be phylogenetically compressed by XZ (the Lempel–Ziv–Markov chain algorithm[7], implemented in XZ Utils[32]) by a left-to-right enumeration of the assemblies, with respect to the topology of their compressive phylogeny obtained via sketching[33].

We implemented phylogenetic compression protocols for assemblies, for de Bruijn graphs, and for *k*-mer indexes in a tool called MiniPhy (Minimization via Phylogenetic compression; https://github.com/karel-brinda/miniphy/). To cluster input genomes, MiniPhy builds upon the empirical observation that microbial genomes in public repositories tend to form clusters corresponding to individual species[34], and species for individual genomes can be identified rapidly via metagenomic classification[35] by the Kraken suite[26] (Fig. 1b and Methods). As some of the resulting clusters may be too large or too small, and thus unbalancing downstream parallelization, it further redistributes the clustered genomes into size-balanced and diversity-balanced batches (Methods and Extended Data Fig. 1). This batching enables compression and search in a constant time (using one node per batch on a cluster) or linear time (using a single machine; Methods). For every batch, a compressive phylogeny—either provided by the user or computed automatically using Mashtree[33]/Attotree (https://github.com/karel-brinda/attotree/; Methods)—is then used for data reordering (Methods). Finally, the obtained reordered data are compressed per batch using XZ with particularly optimized parameters (Methods), and possibly further recompressed or indexed using some general or specialized low-level tool, such as MBGC[18] or COBS[36] (Methods).

Improved compression of large genome collections

We evaluated phylogenetic compression using five microbial collections, selected as representatives of the compression-related trade-offs between characteristics including data quality, genetic diversity, genome size and collection size (GISP, NCTC3k, SC2, 661k and BIGSI-data; Methods, Supplementary Note 2 and Supplementary Table 1). We quantified the distribution of their underlying phylogenetic signal (Methods, Supplementary Table 2 and Extended Data Fig. 2), used them to calibrate the individual steps of the phylogenetic compression workflow (Methods and Extended Data Figs. 3–5) and evaluated the resulting performance, trade-offs and extremal characteristics (Methods, Supplementary Table 3 and Extended Data Fig. 6). As one extreme, we found that 591,000 severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2) genomes can be phylogenetically compressed using XZ to only 18.1 bytes per genome (Methods, Supplementary Table 3 and Extended Data Figs. 4 and 6), resulting in a file size of 10.7 Mb (13.2 times more compressed than GZip). A summary detailing the sensitivity/stability of performance to various factors is provided in Supplementary Note 3.

We found that phylogenetic compression improved the compression of genome assembly collections that comprise hundreds of thousands of isolates of over 1,000 species by more than an order of magnitude compared to the state-of-the-art approach (Fig. 2a and Supplementary Table 3). Specialized high-efficiency compressors such as MBGC[18] are not directly applicable to highly diverse collections; therefore, the compression protocols deployed in practice for extremely large and diverse collections are still based on the standard GZip, such as the 661k collection, containing all bacteria pre-2019 from the European Nucleotide Archive (ENA)[16] (n = 661,405; 805 GB). Here, MiniPhy recompressed the collection to 29.0 GB (27.8 times the improvement; 43.8 kB per genome, 0.0898 bits per base pair, 5.23 bits per distinct *k*-mer) using XZ as a low-level tool, and further to 20.7 GB (38.9 times the improvement; 31.3 kB per genome, 0.0642 bits per base pair, 3.74 bits per distinct *k*-mer) when combined with MBGC[18] that also accounts for reverse complements (Fig. 2a, Supplementary Table 3 and Methods). Additionally, we found that the lexicographically ordered ENA datasets, as being partially phylogenetically ordered, can serve

https://doi.org/10.1038/s41592-025-02625-2

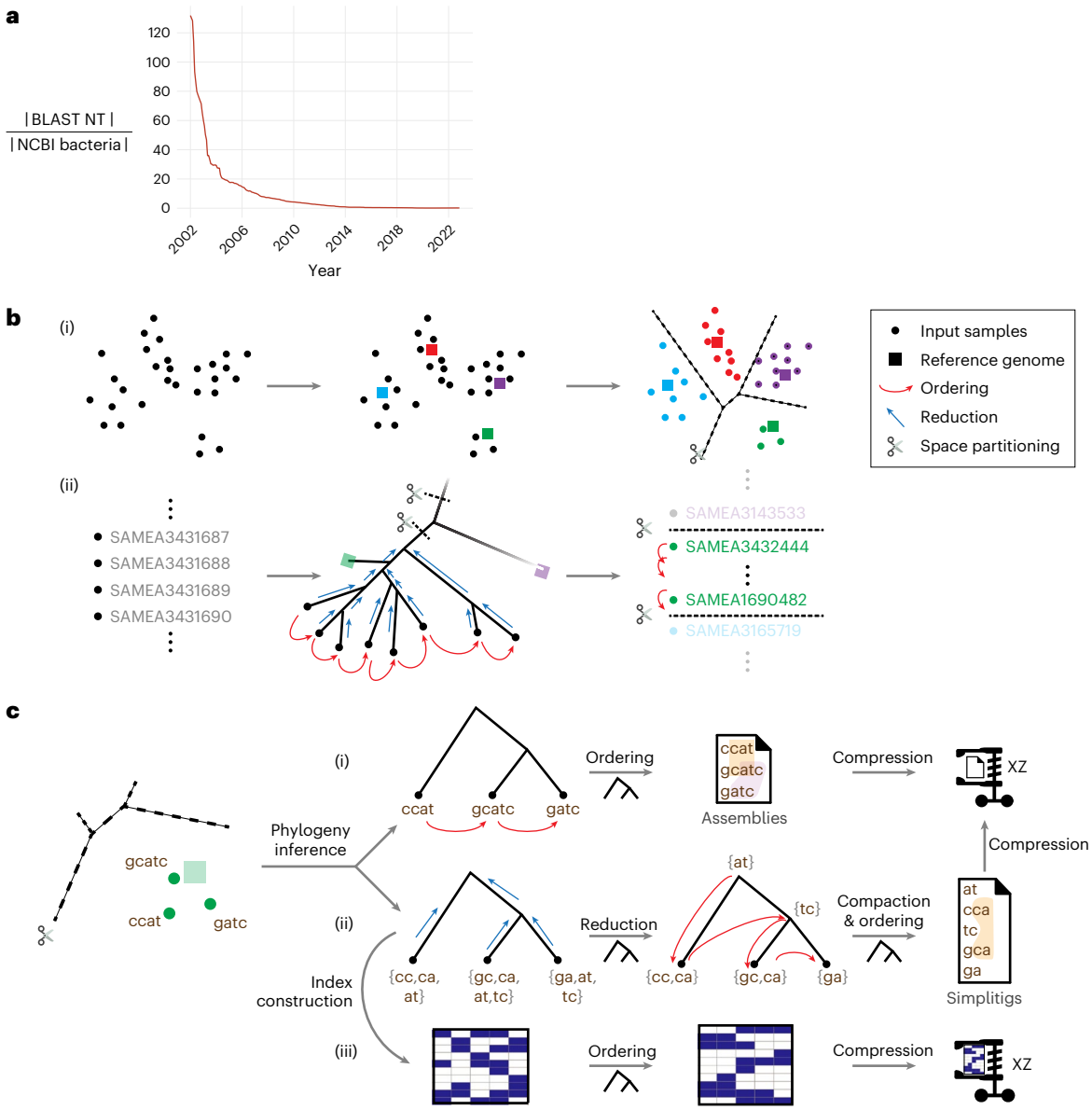


Fig. 1 | Overview of phylogenetic compression and its applications to different data types. a, Exponential decrease of data searchability over the past two decades illustrated by the size of the BLAST NT database divided by the size of the NCBI Bacterial Assembly database (Supplementary Note 1). **b**, The first three stages of phylogenetic compression before the application of a low-level compressor/indexer. (i) A given collection is partitioned into size-balanced and diversity-balanced batches of phylogenetically related genomes (for example, using metagenomic classification of the original reads). (ii) The input data are reversibly reordered based on a compressive phylogeny, performed separately for each batch. **c**, Examples of specific protocols for phylogenetic compression of individual data types, performed separately for each batch. (i) Assemblies are sorted from left to right according to the topology of the phylogeny, and then compressed using a low-level compressor such as XZ[7,32] or MBGC[18]. (ii) For de Bruijn graphs, *k*-mers are propagated in a bottom-up fashion along the phylogeny, and the resulting *k*-mer sets are compacted into simplitigs[28,53,54], which are then compressed using XZ. (iii) For BIGSI *k*-mer indexes, Bloom filters (in columns) are ordered from left to right according to the phylogeny, and then compressed using XZ.

as an approximation of phylogenetic compression, with compression performance only degraded by a factor of 4.17 compared to full phylogenetic compression (Supplementary Table 3 and Methods).

We then studied de Bruijn graphs, a common genome representation directly applicable to raw-read data[17,37], and found that phylogenetic compression can improve state-of-the-art approaches by one to two orders of magnitude (Fig. 2a, Supplementary Table 3 and Methods). As standard and colored de Bruijn graphs lack methods for joint compression at the scale of millions of genomes and thousands of species, single graphs are often distributed individually[38]. For instance, the graphs of the BIGSIdata collection[17], comprising all viral and bacterial genomes from pre-2017 ENA (n = 447,833), are provided in an online repository in the McCortex binary format[39] and occupy in total >16.7 TB (Methods). Here, we retrieved n = 425,160 graphs from the Internet (94.5% of the original count; Methods) and performed lossless recompression using the MiniPhy methodology, with a bottom-up propagation of the *k*-mer content, to 52.3 GB (319 times the improvement; 123 kB per genome, 0.248 bits per base pair (in unitigs), 10.2 bits per distinct *k*-mer; Fig. 2a, Supplementary Table 3 and Methods). Further, as recent advances in de Bruijn graph indexing[15] may lead to more efficient storage protocols in the future, we also compared MiniPhy to MetaGraph[37], an optimized tool for indexing on high-performance servers with a large amount of memory. Here, we found that MiniPhy still provided an improvement of a factor of 5.78 (Methods).

https://doi.org/10.1038/s41592-025-02625-2

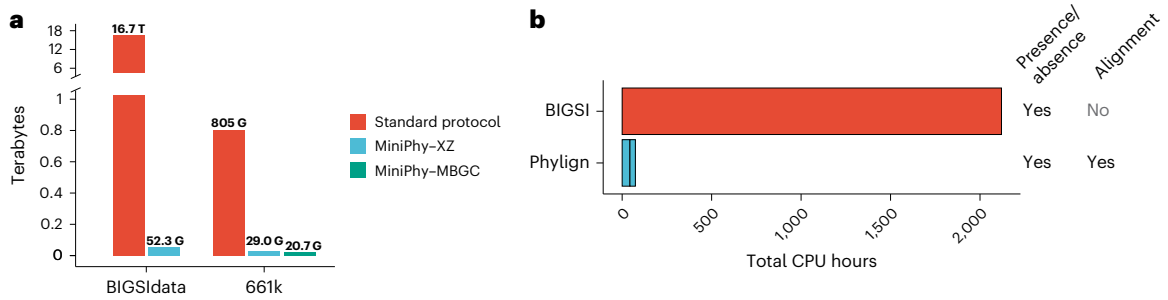


Fig. 2 | Results of phylogenetic compression. a, Compression by MiniPhy of the two comprehensive genome collections: BIGSI (425,160 de Bruijn graphs; the standard compression is based on McCortex binary files) and 661k (661,405 bacterial assemblies; the standard protocol is based on GZip). For BIGSIdata, MBGC is not included as it does not support simplitigs. **b**, Comparison of the Phylign versus BIGSI methods on search of all plasmids from the EBI database. For Phylign, the two segments correspond to the times of matching and alignment, respectively.

Phylogenetic compression can be applied to any genomic data structure based on a genome-similarity-preserving representation (Methods and Supplementary Note 4). We demonstrate this using the Bitsliced Genomic Signature Index (BIGSI)[17] (Fig. 1c(iii)), a *k*-mer indexing method using an array of Bloom filters, which is widely used for large-scale genotyping and presence/absence queries of genomic elements[16,17]. Using the same data, batches and orders as inferred previously, we phylogenetically compressed the BIGSI indexes of the 661k collection, computed using a modified version of COBS[36] (Supplementary Table 4 and Methods). Phylogenetic compression provided 8.51 times the overall improvement compared to the original index (from 937 GB to 110 GB), making it finally usable on ordinary computers. After we further omitted the 3.24% of genomes that had not passed quality control in the original study[16] (the 661k-HQ collection; visualized in Extended Data Fig. 7), the resulting phylogenetic compression ratio improved to 12.3× (72.8 GB; Supplementary Table 4).

To better understand the impact of phylogenetic compression across the tree of life, we analyzed the 661k MiniPhy batches of assemblies and COBS indexes, both before and after compression (Extended Data Fig. 8). We found that although the top ten species constituted nearly 80% of the genomic content, they occupied less than half of the database space after compression for both genome representations (Extended Data Fig. 8). Conversely, the 'dustbin' batches, which include genomes from sparsely sampled species, expanded to occupy a proportion that was 9.4 times larger in the database after compression, compared to their precompression proportion, again for both representations (Extended Data Fig. 8). This consistent effect of compression on both assemblies and COBS indexes suggests that phylogenetic compressibility adheres to the same principles, irrespective of the specific genome representation used, with divergent genomes being a major driver of the final size.

BLAST-like alignment to all bacteria on desktops in hours

To demonstrate the practical utility of phylogenetic compression, we used it to implement BLAST-like search across all high-quality pre-2019 bacteria for standard desktop and laptop computers (Phylign; https://github.com/karel-brinda/phylign/; Methods). For a given a set of queries, Phylign first identifies for each query those genomes that match best globally across the whole 661k-HQ collection, by proceeding via progressive in-memory decompression and querying of individual phylogenetically compressed COBS[36] *k*-mer indexes (described above). Subsequently, Phylign iterates over the phylogenetically compressed genome assemblies (described above) and computes the corresponding full alignments using on-the-fly instances of Minimap2 (Methods)[40]. The choice of tools was arbitrary, and other programs or core data structures could readily be used instead. The resulting requirements amount to only 102 GB from disk (for the compressed COBS indexes and assemblies: 159 kB per genome, 0.329 bits per base pair, 23.0 bits per distinct *k*-mer; Supplementary Table 5) and 12 GB RAM, and Phylign can thus be deployed on most modern laptop and desktop computers.

We first evaluated Phylign with 661k-HQ using three different types of queries—resistance genes (the entire ARG-ANNOT database of resistance genes[41], $n = 1{,}856$), plasmids (EBI plasmid database, $n = 2{,}826$), and a nanopore sequencing experiment ($n = 158{,}583$ reads), with results available within 3.9, 11 and 4.3 h, respectively, on an iMac desktop (Supplementary Table 6). Benchmarking against other tools was not possible, as we were unable to find any tool capable of aligning queries to 661k-HQ in a comparable setup. Therefore, we used the EBI plasmid dataset to compare Phylign to BIGSI with its original database of 447,833 genomes (which is essentially a subset of 661k-HQ with 1.43 times less genomes)[17]. We found that Phylign was over an order of magnitude faster (Fig. 2b and Supplementary Table 6); the search required 74.1 CPU hours and improved performance by a factor of 28.6× compared to the same BIGSI benchmark with its smaller database (Fig. 2b and Supplementary Table 6), while providing the full alignments rather than presence/absence only (Fig. 2b). To our knowledge, this is the first time that alignment to a collection of a comparable size and diversity has been locally performed.

Discussion

It is hard to overstate the impact on bioinformatics of BLAST[2], which has allowed biologists across the world to handily and rapidly compare their sequence of interest with essentially all known genomes—to the extent that the tool name has become a verb. The web version provided by NCBI/EBI is so standard that it is easy to overlook how representative or complete its database is. However, 24 years on, sequencing data have far outstripping BLAST's ability to keep up. Much work has gone into approximate solutions[15], but full alignment to the complete corpus of bacterial genomes has remained effectively impossible. We have addressed this problem and made substantial progress, via phylogenetic compression, a highly efficient general technique using evolutionary history of microorganisms to improve existing compressive data structures and search algorithms by orders of magnitude. More concretely, BLAST-like search of all microorganisms is now possible, not just for NCBI/EBI, but for anyone on a personal laptop. This has wide-ranging benefits, from an easy and rapid download of large and diverse genome collections, to reductions in bandwidth requirements, transmission/storage costs and computational time.

Elements of our approach and related techniques have been previously used in other contexts. Reversible reordering to improve compression forms the core of the Burrows–Wheeler transform[21] and its associated indexes[42–44], and it has also been used for read compression[22–25]. Tree hierarchies have been applied in metagenomics for both lossy[27,45,46] and lossless[28] reference data compression. Finally, a divide-and-conquer methodology has been used to accelerate the inference of species trees[47]. Our approach combines these ideas to

improve the scalability and portability of search and alignment in large genome databases.

As with all forms of compression, our ability to reduce data is fundamentally limited by the underlying entropy. For genome collections, this is not introduced solely by the underlying genetic signal, but it is also tightly connected with the sequencing process and our capacity to reconstruct genomes from sequencing reads. The noise in the underlying *k*-mer histograms (Extended Data Fig. 7) suggests that any method for compression or search will have to address noise in the forms of contamination, missing regions and technological artifacts, with legacy data posing a major challenge for both storage and analysis. Future methods may choose to incorporate stricter filtering, and as our experiments have demonstrated, this helps not only in reducing data volume but also in improving the quality of search outputs; these issues may be alleviated by innovative computational strategies, such as taxonomic filters[48] or sweep deconvolution[49]. Another limitation of our approach is its reliance on phylogenetic trees as a backbone structure for explaining data redundancies. While this applies in regimes where vertical descent predominates or where genetic descent is well approximated by tree structure, in cases where it does not (for example, within a eukaryotic species), the assumption of a phylogenetic tree may not yield substantial gains. In this case, future versions may better use alternate graph structures to trees, such as ancestral recombination graphs[50].

In light of technological development, the benefits of phylogenetic compression will grow over time. Currently, only a fraction of the world's microbial diversity has been sequenced. However, as sequencing becomes more comprehensive, the tree of life will not change, thus enhancing the relative advantage of phylogenetic compression. We foresee its use ranging from mobile devices to large-scale distributed cloud environments and anticipate promising applications in global epidemiological surveillance[51] and rapid diagnostics[52]. Overall, the phylogenetic compression of data structures has broad applications across computational biology and may represent a fundamental design principle for future genomics infrastructure.

Online content

Any methods, additional references, Nature Portfolio reporting summaries, source data, extended data, supplementary information, acknowledgements, peer review information; details of author contributions and competing interests; and statements of data and code availability are available at https://doi.org/10.1038/s41592-025-02625-2.

References

1. Stephens, Z. D. et al. Big data: astronomical or genomical? *PLoS Biol.* **13**, e1002195 (2015).
2. Altschul, S. F., Gish, W., Miller, W., Myers, E. W. & Lipman, D. J. Basic local alignment search tool. *J. Mol. Biol.* **215**, 403–410 (1990).
3. Navarro, G. & Mäkinen, V. Compressed full-text indexes. *ACM Comput. Surv.* **39**, 2 (2007).
4. Loh, P. -R., Baym, M. & Berger, B. Compressive genomics. *Nat. Biotechnol.* **30**, 627–630 (2012).
5. Yu, Y. W., Daniels, N. M., Danko, D. C. & Berger, B. Entropy-scaling search of massive biological data. *Cell Syst.* **1**, 130–140 (2015).
6. Giancarlo, R., Scaturro, D. & Utro, F. Textual data compression in computational biology: a synopsis. *Bioinformatics* **25**, 1575–1586 (2009).
7. Salomon, D. & Motta, G. in *Handbook of Data Compression*, 329–441 (Springer, 2010).
8. Daniels, N. M. et al. Compressive genomics for protein databases. *Bioinformatics* **29**, i283–i290 (2013).
9. Deorowicz, S. & Grabowski, S. Data compression for sequencing data. *Algorithms Mol. Biol.* **8**, 25 (2013).
10. Giancarlo, R., Rombo, S. E. & Utro, F. Compressive biological sequence analysis and archival in the era of high-throughput sequencing technologies. *Brief. Bioinform.* https://doi.org/10.1093/bib/bbt088 (2013).
11. Zhu, Z., Zhang, Y., Ji, Z., He, S. & Yang, X. High-throughput DNA sequence data compression. *Brief. Bioinform.* **16**, 1–15 (2015).
12. Hosseini, M., Pratas, D. & Pinho, A. J. A survey on data compression methods for biological sequences. *Information* **7**, 56 (2016).
13. Jayasankar, U., Thirumal, V. & Ponnurangam, D. A survey on data compression techniques: from the perspective of data quality, coding schemes, data type and applications. *J. King Saud University-Computer Information Sci.* **33**, 119–140 (2021).
14. Navarro, G. Indexing highly repetitive string collections, part I: repetitiveness measures. *ACM Comput. Surv.* **54**, 1–31 (2021).
15. Marchet, C. et al. Data structures based on k-mers for querying large collections of sequencing data sets. *Genome Res* **31**, 1–12 (2021).
16. Blackwell, G. A. et al. Exploring bacterial diversity via a curated and searchable snapshot of archived DNA sequences. *PLoS Biol.* **19**, e3001421 (2021).
17. Bradley, P., den Bakker, H. C., Rocha, E. P. C., McVean, G. & Iqbal, Z. Ultrafast search of all deposited bacterial and viral genomic data. *Nat. Biotechnol.* **37**, 152–159 (2019).
18. Grabowski, S. & Kowalski, T. M. MBGC: multiple bacteria genome compressor. *Gigascience* **11**, giab099 (2022).
19. Deorowicz, S., Danek, A. & Li, H. AGC: compact representation of assembled genomes with fast queries and updates. *Bioinformatics* **39**, btad097 (2023).
20. Zielezinski, A., Vinga, S., Almeida, J. & Karlowski, W. M. Alignment-free sequence comparison: benefits, applications, and tools. *Genome Biol.* **18**, 186 (2017).
21. Burrows, M. & Wheeler, D. J. *A* block-sorting lossless data compression algorithm. SRC Research Report 124, Digital Equipment Corporation, 1–24 (Digital Equipment Corporation Press, 1994).
22. Hach, F., Numanagic, I., Alkan, C. & Sahinalp, S. C. SCALCE: boosting sequence compression algorithms using locally consistent encoding. *Bioinformatics* **28**, 3051–3057 (2012).
23. Patro, R. & Kingsford, C. Data-dependent bucketing improves reference-free compression of sequencing reads. *Bioinformatics* **31**, 2770–2777 (2015).
24. Grabowski, S., Deorowicz, S. & Roguski, Ł. Disk-based compression of data from genome sequencing. *Bioinformatics* **31**, 1389–1395 (2015).
25. Chandak, S., Tatwawadi, K. & Weissman, T. Compression of genomic sequencing reads via hash-based reordering: algorithm and analysis. *Bioinformatics* **34**, 558–567 (2018).
26. Lu, J. et al. Metagenome analysis using the Kraken software suite. *Nat. Protoc.* **17**, 2815–2839 (2022).
27. Kim, D., Song, L., Breitwieser, F. P. & Salzberg, S. L. Centrifuge: rapid and sensitive classification of metagenomic sequences. *Genome Res.* **26**, 1721–1729 (2016).
28. Břinda, K. *Novel Computational Techniques for Mapping and Classification of Next-generation Sequencing Data*. PhD thesis, Univ. Paris-Est (2016).
29. Břinda, K., Salikhov, K., Pignotti, S. & Kucherov, G. ProPhyle: an accurate, resource-frugal and deterministic DNA sequence classifier. *Zenodo* https://doi.org/10.5281/zenodo.1045429 (2017).
30. Ge, H., Sun, L. & Yu, J. Fast batch searching for protein homology based on compression and clustering. *BMC Bioinform.* **18**, 508 (2017).
31. Reiter, T. Clustering the NCBI nr database to reduce database size and enable faster BLAST searches. *Arcadia Science* https://doi.org/10.57844/ARCADIA-W8XT-PC81 (2023).

32. Collin, L. & Pavlov, I. XZ Utils. Available from https://tukaani.org/xz/ (2009).
33. Katz, L. et al. Mashtree: a rapid comparison of whole genome sequence files. *J. Open Source Softw.* **4**, 1762 (2019).
34. Jain, C., Rodriguez-R, L. M., Phillippy, A. M., Konstantinidis, K. T. & Aluru, S. High throughput ANI analysis of 90K prokaryotic genomes reveals clear species boundaries. *Nat. Commun.* **9**, 5114 (2018).
35. Breitwieser, F. P., Lu, J. & Salzberg, S. L. A review of methods and databases for metagenomic classification and assembly. *Brief. Bioinform.* **20**, 1125–1136 (2019).
36. Bingmann, T., Bradley, P., Gauger, F. & Iqbal, Z. COBS: A Compact Bit-Sliced Signature Index. in *String Processing and Information Retrieval* 285–303 (Springer International Publishing, 2019).
37. 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).
38. Rahman, A., Chikhi, R. & Medvedev, P. Disk compression of *k*-mer sets. *Algorithms Mol. Biol.* **16**, 10 (2021).
39. Turner, I., Garimella, K. V., Iqbal, Z. & McVean, G. Integrating long-range connectivity information into de Bruijn graphs. *Bioinformatics* **34**, 2556–2565 (2018).
40. Li, H. Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics* https://doi.org/10.1093/bioinformatics/bty191 (2018).
41. Gupta, S. K. et al. ARG-ANNOT, a new bioinformatic tool to discover antibiotic resistance genes in bacterial genomes. *Antimicrob. Agents Chemother.* **58**, 212–220 (2014).
42. Ferragina, P. & Manzini, G. Opportunistic data structures with applications. In *Proc. 41st Annual Symposium on Foundations of Computer Science* 390–398 https://doi.org/10.1109/SFCS.2000.892127 (IEEE Computer Society, 2000).
43. Gagie, T., Navarro, G. & Prezza, N. Fully functional suffix trees and optimal text searching in BWT-runs bounded space. *J. ACM* **67**, 1–54 (2020).
44. Zakeri, M., Brown, N. K., Ahmed, O. Y., Gagie, T. & Langmead, B. Movi: a fast and cache-efficient full-text pangenome index. *iScience* https://doi.org/10.1016/j.isci.2024.111464 (2024).
45. Ames, S. K. et al. Scalable metagenomic taxonomy classification using a reference genome database. *Bioinformatics* **29**, 2253–2260 (2013).
46. Wood, D. E. & Salzberg, S. L. Kraken: ultrafast metagenomic sequence classification using exact alignments. *Genome Biol.* **15**, R46 (2014).
47. Molloy, E. K. & Warnow, T. Statistically consistent divide-and-conquer pipelines for phylogeny estimation using NJMerge. *Algorithms Mol. Biol.* **14**, 14 (2019).
48. Goig, G. A., Blanco, S., Garcia-Basteiro, A. L. & Comas, I. Contaminant DNA in bacterial sequencing experiments is a major source of false genetic variability. *BMC Biol.* **18**, 24 (2020).
49. Mäklin, T. et al. Bacterial genomic epidemiology with mixed samples. *Microb. Genom.* **7**, 000691 (2021).
50. Kelleher, J. et al. Inferring whole-genome histories in large population datasets. *Nat. Genet.* **51**, 1330–1338 (2019).
51. Gardy, J. L. & Loman, N. J. Towards a genomics-informed, real-time, global pathogen surveillance system. *Nat. Rev. Genet.* https://doi.org/10.1038/nrg.2017.88 (2017).
52. Břinda, K. et al. Rapid inference of antibiotic resistance and susceptibility by genomic neighbour typing. *Nat. Microbiol.* **5**, 455–464 (2020).
53. Břinda, K., Baym, M. & Kucherov, G. Simplitigs as an efficient and scalable representation of de Bruijn graphs. *Genome Biol.* **22**, 96 (2021).
54. Rahman, A. & Medevedev, P. Representation of *k*-mer sets using spectrum-preserving string sets. *J. Comput. Biol.* **28**, 381–394 (2021).

Methods

Conceptual overview of phylogenetic compression

To organize input genomes into phylogenetic trees and compress/index them in a scalable manner, phylogenetic compression combines four conceptual steps.

Step 1: Clustering/batching. The goal of this step (illustrated in Fig. 1b(i)) is to partition genomes into batches of phylogenetically related genomes, of a limited size and diversity, that can be easily compressed and searched together using highly reduced computational resources. During downstream compression, indexing and analyses, these individual batches are processed separately, and their maximum size and diversity can establish upper bounds on the maximum time and space necessary for processing a single batch. For instance, in the realm of *k*-mer aggregative methods (see an overview in ref. 15), this corresponds to a matrix decomposition of a large *k*-mer annotation matrix into a series of small matrices that have both dimensions small, and analogically in the realm of dictionary compression, to reducing the input strings and dictionary sizes.

For microorganisms, clustering can be accomplished rapidly by metagenomic classification[35] applied to the raw reads or other methods for species identification. Microbial genomes in public repositories form distinct clusters, usually (but not always) corresponding to individual species[34], and metagenomic classification can assign individual genomes to these respective clusters, defined by the underlying reference database such as NCBI RefSeq[35]. This requires only a constant time per dataset and can be fully parallelized, resulting thus in a constant-time clustering if sufficiently many computational nodes are available.

The obtained clusters are then reorganized into batches. First, too small clusters are merged, creating a special pseudo-cluster called dustbin, whose purpose is to collect divergent, weakly compressible genomes from sparsely sampled regions of the tree of life. Subsequently, the clusters that are too large—such as those corresponding to oversampled human pathogens (for example, *Salmonella enterica* or *Escherichia coli*)—as well as the dustbin are then divided into smaller batches, to provide guarantees on the maximum required downstream computational resources per one batch. An additional discussion of batching is provided in Supplementary Note 5.

Step 2: Inference of a compressive phylogeny. In this step (illustrated in Fig. 1b(ii)), the computed batches are equipped with a so-called 'compressive phylogeny', which is a phylogeny approximating the true underlying phylogenetic signal with sufficient resolution for compression purposes. If accurate inference methods such as RaxML[55] cannot be applied due to the associated bioinformatics complexity or high resource requirements, phylogenies can be rapidly estimated via lighter approaches such as the Mashtree algorithm[33] (re-implemented more efficiently in Attotree; https://github.com/karel-brinda/attotree/) instead, with only a negligible impact on the resulting compression performance (Extended Data Fig. 5 and Supplementary Note 3).

Step 3: Data reduction/reordering. The compressive phylogenies obtained in the previous step serve as a template for phylogenetic reordering of individual batches. The specific form of reordering can vary depending on the specific data representations, intended applications and method of subsequent compression or indexing. In principle, the reordering can occur in two directions: as a left-to-right genome reordering based on the topology of the compressive phylogeny, or as a bottom-up reduction of genomic content along the phylogeny (followed by left-to-right enumeration). Regardless of the specific form, this transformation is always reversible, thus sharing similarities with methods such as the Burrows–Wheeler transform[21]. Fig. 1b(ii) illustrates this via the colored arrows.

Step 4: Compression or indexing using a calibrated low-level tool. Finally, the reordered data are compressed or indexed using a low-level tool. At this stage, thanks to both phylogeny-based clustering and phylogeny-based reordering, the data are highly locally compressible, which enables the use of a wide range of general and specialized genome compressors/indexes. Nevertheless, it is crucial to ensure that the properties of the underlying algorithms and their parameters are closely tailored to the specific characteristics of the input data and their intended applications. For instance, to compress genomes in FASTA format, compressors based on Lempel–Ziv require the window/dictionary sizes to be large enough to span multiple genomes (Extended Data Fig. 3a), and general compressors also critically depend on FASTA being in a one-line format (Extended Data Fig. 3b). As a general rule, general compressors must always be carefully tested and calibrated for specific genomic data types, potentially requiring format cleaning and parameter calibration, whereas specialized genome compressors and indexers are usually pre-calibrated in their default setting and provided with well-tested configuration presets. While in many practical scenarios, individual batches are compressed/indexed separately, some protocols may involve merging reordered batches to create a single comprehensive archive/index. This step applies to the results shown in Fig. 1c.

The MiniPhy framework for phylogenetic compression

Here, we describe the specific design choices of our implementation of phylogenetic compression for assemblies and de Bruijn graphs. More information and relevant links, including specific tools such as MiniPhy and Phylign and the resulting databases, can be found on https://brinda.eu/mof/.

Clustering/batching. As genome collections encountered in practice can vary greatly in their properties as well as the available metadata, clustering is expected to be performed by the user. The recommended procedure is to identify species clusters using standard metagenomic approaches, such as those implemented in the Kraken software suite[26], as the obtained abundance profiles can also be used for quality control to filter out those samples that are likely contaminated. The next step is to divide the obtained genome clusters into smaller batches, analogically to the examples in Extended Data Fig. 1 and as discussed in more details in Supplementary Note 5 (and the corresponding implementation in the MiniPhy package, see below). The order in which genomes are taken within individual clusters can impact the final compression performance; based on our experience, lexicographic order with accessions and ordering according to the number of distinct *k*-mers per genome provide surprisingly good performance as both of these approaches tend to group phylogenetically close genomes closer to each other. The protocol can be customized further to suit the performance characteristics of algorithms downstream, such as by adjusting the batch size or the parameters controlling the creation of dustbin batches (Supplementary Note 5). If the total size of a collection is small enough, the clustering/batching step may be skipped entirely and the entire collection treated as a single batch.

Inference of a compressive phylogeny. Users have the option to provide a custom tree generated by an accurate inference method such as RAxML[55]. However, in most practical scenarios, such trees are not available, and MiniPhy then uses Attotree (https://github.com/karel-brinda/attotree/), an efficient re-implementation of the Mashtree algorithm[33], to generate a compressive phylogeny through sketching. Both Mashtree and Attotree use Mash[56] to estimate, using the MinHash sketching technique[57] and a simple evolutionary model[58], the evolutionary distances between all pairs of genomes. This is followed by the inference of a compressive phylogeny using the Neighbor-Joining algorithm[59], as implemented in QuickTree[60]. Finally, MiniPhy post-processes the obtained tree using standard tree-transformation procedures

implemented in the ETE3 library[61], involving tree standardization, setting a midpoint outgroup, ladderization and naming the internal nodes.

MiniPhy. This is a central package for phylogenetic compression, including support for batching, and for calculating the associated statistics (see below). MiniPhy (https://github.com/karel-brinda/miniphy/) is implemented as a Snakemake[62] pipeline, offering three protocols for phylogenetic compression: (1) compression of assemblies based on left-to-right reordering; (2) compression of de Bruijn graphs represented by simplitigs[28,53,54] based on left-to-right reordering; and (3) compression of de Bruijn graphs through bottom-up *k*-mer propagation using ProPhyle[28,29].

In the third protocol, *k*-mer propagation is executed recursively in a bottom-up manner: at each internal node, the *k*-mer sets of the child nodes are loaded, their intersection computed, stored at the node, the intersection subtracted from the child nodes, and all three *k*-mer sets saved in the form of simplitigs[28,53,54]; ProphAsm[53] performs all these operations. This process results in a progressive reduction of the *k*-mer content within the phylogeny in a lossless manner[28].

The output of each of the three protocols is a TAR file containing text files in their phylogenetic order, created from the corresponding list of files using the following command:

```
tar cvf - -C $(dirname {input.list}) -T {input.list}
--dereference
```

For assemblies, these text files are the original assembly FASTA files, converted by SeqTK[63] to the single-line format with all nucleotides in uppercase (‘`seqtk seq -U {input.fa}`’). For simplitigs, the text files are end-of-line-delimited lists of simplitigs in the order as computed by ProphAsm, obtained from its output using the command ‘`seqtk seq {input.fa} | grep -v \>`’. The resulting TAR file is then compressed using XZ (‘`xz -9 -T1`’; see section ‘Calibration and evaluation of phylogenetic compression’), and the resulting .tar.xz file distributed to users or further recompressed or indexed by other low-level tools, while preserving the underlying order.

MiniPhy statistics. For each of the three implemented protocols, MiniPhy generates a comprehensive set of statistics to quantify the compressibility of the batch, including: (1) set (the size of the *k*-mer set computed from all nodes of the compressive phylogeny); (2) multiset (the size of the *k*-mer multiset computed as a union of *k*-mer sets from individual nodes); (3) sum_ns (the total number of sequences); (4) sum_cl (the total sequence length); (5) recs (the number of records corresponding to individual nodes); and (6) xz_size (the size of the TAR file after XZ compression). The sizes of *k*-mer sets and multisets are determined from *k*-mer histograms computed by JellyFish 2 (v2.2.10)[64] using the commands:

```
jellyfish count --threads {threads} --canonical
--mer-len 31 --size 20M
--output {jf_file} {input}
```

followed by

```
jellyfish histo --threads {threads} --high 1000000
{jf_file}
```

The computed statistics are used for calculating additional compression-related metrics, such as the number of bits per distinct *k*-mer or kilobytes per genome.

Phylogeny-explained redundancy. By comparing the sizes of *k*-mer sets and multisets before and after reduction by *k*-mer propagation along a compressive phylogeny, it is possible to quantify the proportion of the *k*-mer signal that is explained by the phylogeny. This yields the so-called ‘phylogeny-explained *k*-mer redundancy’, quantifying the proportion of redundant occurrences of canonical *k*-mers that can be eliminated through *k*-mer propagation, of those potentially eliminable if the phylogeny perfectly explained the distribution of all the *k*-mers (that is, every *k*-mer occurring only once after propagation and thus being associated with a single entire subtree):

$$\text{removed\_redundancy} = \frac{|\text{multiset\_preprop}| - |\text{multiset\_postprop}|}{|\text{multiset\_preprop}| - |\text{set}|}$$

For collections comprising multiple batches, these variables refer to the global statistics, that is, the sizes of set and multiset unions across all batches.

MiniPhy-COBS. MiniPhy-COBS (https://github.com/leoisl/miniphy-cobs/) is a Snakemake[62] pipeline designed to create phylogenetically compressed ClaBS COBS indexes[36] (classical bit-sliced index) from assemblies already phylogenetically compressed by MiniPhy. ClaBS is a variant of COBS analogous to the original BIGSIdata structure[17], using Bloom filters of the same size; this property is important for ensuring that the order of Bloom filters is preserved and that the neighboring Bloom filters are mutually compressible (Supplementary Note 4). The workflow for each batch involves three main steps:

1. Renaming input assemblies to align their lexicographic and phylogenetic orders within each batch
2. Constructing COBS ClaBS indexes with:
   `cobs classic-construct -T 8 {batch} {output}.cobs_classic`
3. Compressing the obtained indexes using:
   `xz -9 -T1 -e --lzma2=preset=9,dict=1500MiB,nice=250`

Updated ProPhyle. To simplify the integration with MiniPhy for bottom-up *k*-mer propagation, a new version of ProPhyle[28,29] was released (v0.3.3.1; https://github.com/prophyle/prophyle/). The main improvement compared to previous versions includes the possibility to stop after *k*-mer propagation, without proceeding to the construction of an FM-index, as such an index is unnecessary for phylogenetic compression using MiniPhy. The new version of ProPhyle is provided in the form of a GitHub release (https://github.com/prophyle/prophyle/releases/) and pre-built packages on Bioconda[65].

Acquisition of the test collections

An overview of the five test collections is provided in Supplementary Note 2, and their basic characteristics, including the original file size, number of samples, species count and the number of distinct *k*-mers, are provided in Supplementary Table 1.

GISP. The GISP collection was obtained from GitHub repository available at https://github.com/c2-d2/rase-db-ngonorrhoeae-gisp/ (version 04a132c)[52]. The assemblies (n = 1,102) were obtained from the ‘isolates/contigs’ subdirectory of the GitHub repository (containing the original genomes including the plasmids), and the associated RAxML phylogenetic tree was downloaded from the ‘tree/’ subdirectory of the same repository. The original data had originally been analyzed in ref. 66 and provided for download on Zenodo (https://doi.org/10.5281/zenodo.2618836; 2019).

NCTC3k. The assemblies were obtained in GFF format from ftp://ftp.sanger.ac.uk/pub/project/pathogens/NCTC3000 by

```
wget -m -np -nH --cut-dirs 3 -retr-symlinks
ftp://ftp.sanger.ac.uk/pub/project/pathogens/NCTC3000
```

The obtained files were converted to the FASTA format by any2fasta (https://github.com/tseemann/any2fasta/, v0.4.2) parallelized by GNU Parallel[67] and uploaded to Zenodo (https://doi.org/10.5281/zenodo.4838517). The number of species in the collection was determined based on the data provided in the main Sanger/Public Health England assembly table for NCTC 3000 (https://www.sanger.ac.uk/resources/downloads/bacteria/nctc/, retrieved on 14 September 2022). The HTML table was manually exported to XLSX and used to construct a translation table from NCTC accession numbers to corresponding species. The accessions of the assemblies in our collection were then extracted from file names and translated to species, and the species were counted. Overall, this resulted in $n = 1{,}065$ assemblies of 259 species.

SC2. The SARS-CoV-2 data were downloaded from the GISAID website (https://www.gisaid.org/, 18 May 2021) in the form of an assembly file ('`sequences_fasta_2021_05_18.tar.xz`', $n = 1{,}593{,}858$) and a Sarscov2phylo phylogeny (https://doi.org/10.5281/zenodo.4089815, '`gisaid-hcov-19-phylogeny-2021-05-11.zip`', $n = 590{,}952$). After converting both files to the same set of identifiers and removing isolates with missing data, we obtained $n = 590{,}779$ genome assemblies organized in a phylogenetic tree.

BIGSIdata. The BIGSI collection data[17] were downloaded from the associated FTP server (http://ftp.ebi.ac.uk/pub/software/bigsi/nat_biotech_2018/), including cleaned de Bruijn graphs, taxonomic information and abundance reports computed using Kraken and Bracken[26]. The download was done using RSync in groups corresponding to individual EBI prefixes (for example, DRR000) by

```
rsync -avP --min-size=1 --exclude '*stats*'
--exclude '*uncleaned*' --exclude '*bloom*'
--exclude '*log*'
rsync://ftp.ebi.ac.uk/pub/software/bigsi/
nat_biotech_2018/ctx/{prefix}
```

The prefixes were organized into 15 groups of at most 100 prefixes each, and the groups were processed individually in succession on a research computing cluster, with a parallelization using Slurm and jobs deployed using Snakemake[62] (between 1 August 2020 and 15 September 2020). From the downloaded McCortex files, unitigs were extracted using McCortex:

```
bzcat -f {input} | mccortex31 unitigs -m 3G -
```

Only those graphs with an uncorrupted McCortex file, Bracken information available, unitigs of total length ≥2 kbp with ≤15 million distinct *k*-mers and with no file system error encountered were used in the subsequent processing. This resulted in $n = 425{,}160$ de Bruijn graphs (of the original $n = 463{,}331$ genomes from the FTP or $n = 447{,}833$ genomes reported in ref. 17).

661k. The 661k collection was downloaded in March 2021 from the official FTP repository[16], using RSync by

```
rsync -avp rsync://ftp.ebi.ac.uk/pub/databases/
ENA2018-bacteria-661k/Assemblies/{pref}
```

The command was run for individual prefixes ranging from 000 to 661, which resulted in $n = 661{,}405$ .fa.gz files.

Calibration and evaluation of phylogenetic compression

Calibration of XZ as a low-level tool for phylogenetic compression. The compression performance of GZip, BZip2 and XZ was evaluated using the GISP collection, converted to the single-line FASTA format and with genomes sorted from left to right according to the Mashtree phylogeny (Extended Data Fig. 3). For each compressor, the compression was performed with a range of presets and always with a single thread. To evaluate the compression performance with large resources available, two additional manually tuned modes with larger dictionaries, denoted by 'M' and 'MM', were added to the XZ benchmark, corresponding to the parameters

```
--lzma2=preset=9,dict=512MiB
```

and

```
--lzma2=preset=9,dict=1500MiB,nice=250
```

respectively.

To evaluate the impact of different line lengths on the compression, the source FASTA was reformatted for different lengths using SeqTK[63] and compressed using XZ by

```
seqtk seq -l {line_length} | xz -9 -T1
```

Comparison of scaling modes. The SC2 collection was provided in the left-to-right order according to Sarscov2phylo phylogeny (Extended Data Fig. 4). The genomes were progressively uniformly subsampled, stored as end-of-line-separated lists of sequences (without sequence headers), and then compressed using individual compressors, namely (1) XZ: '`xz -9 -T1`', (2) BZip2: '`bzip2 --best`', (3) GZip: '`gzip -9`' and (4) Re-Pair[68,69] (https://github.com/rwanwork/Re-Pair/; version as of 26 October 2021):

```
repair -v -I {inp_seqs}; tar cf {inp_seqs}.tar
{inp_seqs}.prel {inp_seqs}.seq
```

As Re-Pair did not provide sufficient scalability for the entire SC2 dataset and the implementation suffered from various bugs, the Re-Pair sub-experiment was limited only to $n \leq 70{,}000$, the integrity of the output files always verified via their decompression and line counting, and all archives lacking integrity were discarded from the subsequent analysis.

The scalability comparisons for the NCTC3k and GISP collections were performed analogically, but using MiniPhy (commit '`41976c7`') and with sequence headers preserved. The order of all assemblies was first randomized by '`sort -R`' and the individual sub-samplings for compression then generated as prefixes of this randomized list. The size comparisons were made based on the .tar.xz output file of the pipeline, as well as additional files obtained via their recompression by GZip and BZip2 with the same parameters as above.

Order comparison. The SC2 collection was put into three different orderings: the original ordering (corresponding to the lexicographical ordering by sequence names), the left-to-right ordering of the phylogeny and a randomized order (Extended Data Fig. 5). In all cases, a custom Python script using BioPython[70] was used to order the FASTA file and remove sequence names, and its output was compressed by the XZ compressor using one thread and the best preset ('`xz -T1 -9`'). The comparisons for GISP and NCTC3k were performed analogically, but with sequence headers preserved.

Summary of MiniPhy calibration. XZ with the parameters '`xz -9 -T1`' was chosen as the default compression procedure for MiniPhy, and Mashtree[33] or its reimplementation Attotree (https://github.com/karel-brinda/attotree/) as the default method for inferring compressive phylogenies. These choices were done based on

the observations that the most popular method, GZip, always performed poorly for bacteria, although provided a moderate compression performance for viruses. On the other hand, XZ achieved steep compression curves for low-diversity collections, with compression ratio improving by one order per one order increase of the number of genomes, for both viruses and bacteria. NCTC3k as a high-diversity collection was weakly compressible even with the best approaches (less than one order of magnitude of compression after a three orders-of-magnitude increase in the number of genomes). One of the best available (but still highly experimental) grammar-based compressors, Re-Pair[68,69], achieved a similar asymptotic behavior as XZ, indicative of the potential of grammar compressors for phylogenetic compression to provide random access, but its usability remains experimental. Phylogenetic reordering boosted compression substantially for both low-diversity and high-diversity collections (reduction in size between 38% and 67% compared to random orders). Finally, compressive phylogenies computed using Mashtree[33] provided nearly equal compression performance as an accurate approach using RaxML[55].

Phylogenetic compression of the BIGSIdata collection of de Bruijn graphs

Clustering and batching. For every sample, the outputs of Kraken and Bracken[26] were extracted from the downloaded data as provided in the online FTP repository (https://ftp.ebi.ac.uk/pub/software/bigsi/nat_biotech_2018/ctx/) in the Bracken files ('`{accession}.ctx_braken.report`') as the previously identified most prevalent species (corresponding to the row with the highest value of the '`fraction_total_reads`' column). Clustering and batching then proceeded as depicted in Extended Data Fig. 1 and further commented in Supplementary Note 5, with genomes being sorted according to the number of *k*-mers before their partitioning into batches. Overall, the genomes of the 1,443 identified species (clusters) were partitioned into 568 regular batches and 6 dustbin batches, resulting in a total of 574 batches.

Phylogenetic compression. Phylogenetic compression was performed twice, with slightly different workflows.

First, phylogenetic compression proceeded manually, via a workflow whose modified version was later implemented in MiniPhy. For individual batches, compressive phylogenies were computed using Mashtree[33] with the default parameters. The resulting trees and McCortex unitig files were then used as input for ProPhyle (v0.3.3.0) to propagate *k*-mers along the phylogenies, compute simplitigs[28,53,54] and merge the output FASTA files into a single one by

```
prophyle index -k 31 -A -g {dir_genomes} {tree}
{batch_name}
```

The resulting FASTA files produced by ProPhyle (called '`index.fa`') were converted into the single-line format using SeqTK[63] and compressed using XZ by

```
seqtk seq {prophyle_index_fa} | xz -9 -T8
```

The resulting files occupied 74.4 GB and were deposited on Zenodo (https://doi.org/10.5281/zenodo.4086456 and https://doi.org/10.5281/zenodo.4087330). The correctness of the whole approach was validated using a dedicated Python package for decompression (see below); the *k*-mer counts in the decompressed data (obtained by kc-c3, https://github.com/lh3/kmer-cnt/, commit '`e257471`') were compared to those obtained from the original McCortex files (from the total length and count of unitigs). All *k*-mer counts were equal, with the exception of four samples with from 17 to 26 more reported *k*-mers after decompression.

Second, an analogical version of the propagated simplitig files, but without sequence headers and with compression using a single thread only, was later created using the MiniPhy pipeline and resulted in files occupying in total 52.3 GB, which were subsequently deposited on Zenodo (https://doi.org/10.5281/zenodo.5555253).

Decompression of BIGSIdata de Bruijn graphs. To decompress de Bruijn graphs from the files obtained by *k*-mer propagation, all *k*-mers along all root-to-leaf paths need to be collected. We implemented this specifically for BIGSIdata in a dedicated Python package provided in GitHub (https://github.com/karel-brinda/phylogenetic-compression-supplement/). The program downloads individual data files from Zenodo from the accessions above (the first version of the dataset) and reconstructs the original *k*-mer sets using the following procedure. First, it decompresses the XZ file of a given batch, splits it according to files corresponding to individual nodes of the compressive phylogeny, recompresses individual nodes using GZip parallelized by GNU Parallel[67], and for all leaves (genomes) it reconstructs the corresponding *k*-mer sets by merging all GZip files along the corresponding root-to-leaf paths using the Unix cat command. From the obtained output FASTA files, de Bruijn graphs can be easily reconstructed by standard tools such as BCALM2 (ref. 71).

Comparison to the original compression protocol. As the samples in our BIGSIdata collection do not fully correspond to the data that were used in the original publication of BIGSI[17], we recalculated the size statistics of the published McCortex files of our graphs based on the FTP list-off files as provided within individual subdirectories of http://ftp.ebi.ac.uk/pub/software/bigsi/nat_biotech_2018/ (as of 27 August 2021). These were downloaded per individual prefix directories recursively using wget by

```
wget -nv -e robots=off -np -r -A .html
http://ftp.ebi.ac.uk/pub/software/bigsi/
nat_biotech_2018/ctx/{prefix}/
```

The corresponding parallelized Snakemake pipeline was run on a desktop computer. This resulted in a table containing 484,463 files, of which 162,645 were compressed using BZip2. The individual file records were compared with the list of accessions of files that were previously retrieved and sorted in our BIGSIdata collection, and the volume of the source graphs on FTP calculated to be 16.7 TB.

Comparison to Metagraph. The size of the phylogenetically compressed BIGSIdata collection was compared to the size of an analogous Metagraph index from the original paper[37], based on the statistics in Table 1 and Supplementary Table 1 therein (the Sequence Read Archive-Microbe collection): $n = 446{,}506$ indexed datasets, 39.5 G canonical *k*-mers (with the same *k*-mer size, $k = 31$) and the size of the annotated de Bruijn graph being 291 GB (graph 30 GB + annotations 261 GB). This index was constructed from the same datasets as those in the original BIGSI paper[17] but using a slightly different computational methodology. Consequently, the index of Metagraph contained approximately 4% fewer distinct canonical *k*-mers ($k = 31$) compared to BIGSIdata as used in this paper. To compare the two compression approaches (MiniPhy with bottom-up *k*-mer propagation and XZ as a low-level tool versus Metagraph), both applied to the similar but different input data, we used the number of bits per distinct *k*-mer as the statistic for comparison, which was found to be 10.2 and 58.9, respectively. Therefore, the MiniPhy compression was more efficient by an estimated factor of 5.78. We note that phylogenetic compression could be directly embedded into Metagraph (by imposing the

phylogenetic order of columns during index construction), which may help to further reduce its index size.

Phylogenetic compression of the 661k assembly collection

Clustering and batching. Species clusters were identified based on the most prevalent species in the sample as identified using Kraken 2 and Bracken[26] from the original raw-read data; that is, based on the 'V2' column in the 'File1_full_krakenbracken.txt' file of the supplementary materials of ref. 16. The creation of the dustbin pseudo-cluster and formation of individual batches proceeded by the steps documented in Extended Data Fig. 1 and as later implemented directly within MiniPhy, with genomes pre-sorted lexicographically according to ENA accessions.

Phylogenetic compression using MiniPhy. The obtained batches were compressed using the MiniPhy pipeline as described above; that is, compressive phylogenies were computed using Mashtree[33] and used for (1) left-to-right reordering of the assemblies, (2) left-to-right reordering of simplitigs[28,53,54] of the corresponding de Bruijn graphs, and (3) bottom-up *k*-mer propagation and simplitig computation by ProPhyle; while in all cases storing the simplitigs and assemblies as text and FASTA files, respectively, followed by a compression by 'xz -9 -T1'. The compressed assemblies were deposited on Zenodo (https://doi.org/10.5281/zenodo.4602622).

Calculations of the statistics. All the statistics used in the plots and tables were calculated based on the numbers obtained from MiniPhy. Additionally, the total number of *k*-mers was calculated using JellyFish[64] (v.2.2.10) by

```
jellyfish count --mer-len 31 --size 200G --threads 32
--output kmer_counting.jf --out-counter-len=1
--canonical
```

which resulted in 44,349,827,744 distinct *k*-mers (28,706,296,898 unique *k*-mers) for the 661k collection and in 35,524,194,027 distinct *k*-mers (22,904,412,202 unique *k*-mers) for the 661k-HQ collection (as described below). The files uploaded to Zenodo (https://doi.org/10.5281/zenodo.4602622) are higher by approximately 0.2 GB (approximately 0.7% of the total size) compared to the values in Supplementary Table 3 as the Zenodo submission was done with an older version of compressive phylogenies without their post-processing.

Recompression using MBGC. Individual phylogenetically compressed batches from the previous step were converted to single FASTA files by 'tar -xOvf {input.xz}' and then compressed using MBGC[18] (v.1.2.1) with eight threads and the maximum compression level by

```
mbgc -i {input.fa} -c 3 -t 8 {output.mbgc}
```

Compression in the lexicographic order. Data in the ENA and other similar repositories have identifiers assigned in the order in which they are uploaded; individual uploads typically proceed by uploading entire projects, and these typically involve phylogenetically very close genomes. For instance, genomes from a study investigating a hospital outbreak often occupy a range of accessions. Therefore, lexicographically sorted genomes from the ENA may be used as an approximation of phylogenetic compression. To compare the compressibility of the 661k collection in the ENA accession lexicographic order to the full phylogenetic compression, we streamed the genomes from the main collection file provided on http://ftp.ebi.ac.uk/pub/databases/ENA2018-bacteria-661k/661_assemblies.tar, decompressed them on the fly, converted them to the one-line FASTA format using SeqTK[63] and compressed them using XZ with 32 threads by

```
pv 661_assemblies.tar | tar -xOf - | gunzip -c |
seqtk seq | xz -9 -T32
```

Phylogenetic compression of the 661k/661k-HQ *k*-mer indexes

The 661k-HQ collection. To reduce biases in *k*-mer matching, a high-quality variant of the 661k collection, called 661k-HQ, was constructed from the 661k collection by excluding genomes that had not passed quality control in the original study[16] (3.24% of the genomes). For simplicity, the batches and genome orders in 661k-HQ were kept the same as in 661k.

Phylogenetic compression of the 661k/661k-HQ COBS indexes. COBS indexes for the 661k and 661k-HQ collection were constructed per batch using the MiniPhy-COBS pipeline (see 'MiniPhy-COBS' above), which produces the ClaBS variant of the index with all Bloom filters of the same size sorted in a left-to-right order according to the phylogeny, and compresses them using XZ.

Comparisons to the compact COBS indexes. The compact variant of the COBS index (default in COBS), based on adaptive adjustments of Bloom filter sizes through subindexes of different heights, was used as a baseline in our comparisons. For the 661k collection, we used the original index as provided (http://ftp.ebi.ac.uk/pub/databases/ENA2018-bacteria-661k/661k.cobs_compact; retrieved on 8 September, 937 GB). For building a COBS index for 661k-HQ, we used the same construction protocol as in ref. 16. Both indexes were then compressed on a highly performant server by XZ using 32 cores ('xz -9 -T32').

All of the obtained data points are provided in Supplementary Table 4.

Phylign pipeline for alignment against all pre-2019 bacteria from the ENA

Overview. The Phylign pipeline (https://github.com/karel-brinda/phylign/) uses phylogenetically compressed assemblies (661k) and COBS indexes (661k-HQ) as described above to align queries against the entire 661k-HQ collection in a fashion similar to BLAST (Supplementary Note 6). The search procedure consists of two phases: matching the queries against the *k*-mer indexes using COBS[36] to identify the database's most similar genomes for each query, followed by an alignment of the queries to their best-matching genomes using Minimap2 (ref. 40). Phylign is developed as a Snakemake[62] pipeline, using Bioconda[65] for an automatic software management and the standard Snakemake resource management[62] to control the CPU core assignments and limit RAM usage according to user-specified parameters. Upon its first execution, Phylign downloads its phylogenetically compressed reference database from the Internet (102 GB), consisting of 29.2 GB of assemblies and 72.8 GB of COBS indexes.

Matching. The matching step involves *k*-mer matching of all user queries against the entire 661k-HQ database using a modified version of COBS (v0.3, see below), based on the principle that the number of *k*-mer matches between a genome and a query correlates with the alignment score[72]. Each phylogenetically compressed COBS index is decompressed in memory and queried for the input user sequences, reporting all matches between the queries and genomes in the current batch with a sufficient (user-specified) proportion of matching *k*-mers. The computed matches are then aggregated across all batches and, for each query, only a (user-specified) number of best matches, plus ties, are retained and passed to the subsequent alignment step. Matching is parallelized by Snakemake[62],

with the number of threads for each COBS instance adjusted based on batch size.

Alignment. For each batch independently and fully in parallel, Phylign then iterates over the phylogenetically compressed genome assemblies, and if a given genome has at least one match passed from the matching phase, it builds on the fly, in memory, a new Minimap2 (v2.24)[40] instance for this genome and aligns all relevant queries to this genome, while saving Minimap2 outputs in a batch-specific output file. Once all batches are processed, the resulting alignments are aggregated and provided to the user in a modified SAM format[73].

Performance characteristics. The total matching time is primarily driven by the time complexity of COBS, with decompression accounting for less than 2 CPU hours (Extended Data Fig. 9). In the alignment step, decompression requires less than 1.5 CPU hours (Extended Data Fig. 9), and the remainder of the time is primarily driven by the time to create a new Minimap2 instance (estimated 0.3 CPU seconds per instance in the current implementation). If the queries are long and Minimap2 is used with a sensitive preset, the actual Minimap2 alignment time becomes the main time component (for example, in the plasmid experiment in Supplementary Table 6).

Updated COBS. To integrate COBS into Phylign, new versions of COBS[36] were created (v0.2, v0.3; https://github.com/iqbal-lab-org/cobs/). The updates include support for macOS, streaming of indexes into memory and multiple bug fixes. The new versions of COBS are provided in the form of GitHub releases (https://github.com/iqbal-lab-org/cobs/releases/) and pre-built packages on Bioconda[65].

Benchmarking of the decompression time. Decompression times were evaluated on the same desktop computer as the alignment experiments, separately for the phylogenetically compressed assemblies versus COBS indexes and for in-memory decompression (`xzcat {file} > /dev/null`) versus on-disk decompression (`xzcat {file} > {tmpfile}`), resulting in four experiments. Within each experiment, decompression was parallelized using GNU Parallel (`parallel -L1 -v --progress`), with time measured using the GNU time command both for the whole experiment and for each batch in a given compressed representation.

Evaluating Phylign

Overview of the benchmarking procedure. The search using Phylign was evaluated on three datasets, representative of different query scenarios: a database of antibiotic resistance genes, a database of plasmids and an Oxford Nanopore sequencing experiment. In all cases, the search parameters—including the number of hits of interest, the COBS *k*-mer threshold and the Minimap preset—were tailored to each specific query type. The experiments were conducted on an iMac with a Quad-Core Intel CPU i7, 4.2 GHz with four physical (eight logical) cores and 42.9 GB (40 GiB) RAM.

Time measurements. The wall clock and CPU time were measured using GNU time and calculated as real and usr + sys, respectively. The measurements were done for the matching and alignment steps separately.

Memory measurements. We have not found any reliable way of measuring peak memory consumption on macOS: both GNU time and the psutil Python library were substantially underestimating the memory footprint of our Snakemake pipeline. Therefore, we performed additional measurements on a Linux cluster using the SLURM job manager, using jobs allocated with a configuration similar to the parameters of our iMac computer. For `max_ram_gb` set to 30 GB, we observed a peak memory consumption of 26.2 GB, thus by 12.7% lower compared to the specified maximum. Such a discrepancy is expected because the `max_ram_gb` parameter defines an upper bound for the Snakemake resource management[62], representing the worst-case scenario for parallel job combinations.

Resistance genes—ARG-ANNOT. The resistance genes search was performed using the ARG-ANNOT database[41] comprising 1,856 genes/alleles, as provided on GitHub (https://github.com/katholt/srst2/blob/master/data/ARGannot_r3.fasta/; retrieved on 24 July 2022). The search parameters were set to require a minimum of 50% matching *k*-mers, with 1,000 best hits plus ties taken for every gene/allele query. Alignment was performed with the Minimap preset for short reads (`sr`).

Plasmids—the EBI plasmid database. The list of EBI plasmids was downloaded from the associated EBI website (https://www.ebi.ac.uk/genomes/plasmid.details.txt, retrieved on 3 April 2022), and individual plasmids were subsequently downloaded from the ENA using curl and GNU Parallel[67]. The search parameters were set to require at least 40% matching *k*-mers (the threshold previously used in ref. 17), with 1,000 best hits plus ties taken for every plasmid. Alignment was performed with the Minimap preset for long, highly divergent sequences (`asm20`).

Oxford Nanopore reads. The ERR9030361 experiment, comprising 158,583 nanopore reads from an isolate of *Mycobacterium tuberculosis*, was downloaded from the Sequence Read Archive NCBI database. The search parameters were set to require at least 40% matching *k*-mers, with ten best hits plus ties taken for every read. Alignment was performed with the Minimap preset for nanopore reads (`map-ont`).

Comparison to BIGSI. As we were unable to reproduce the original plasmid search experiment[17] with BIGSI on our iMac computer (due to the required database transfer of 1.43 TB over an unstable FTP connection), we used the values provided in the original publication[17]. To ensure a fair comparison, we focused on evaluating the total CPU time (sys + usr) and verified that our parallelization efficiency was close to the maximal one (680% of 800% possible achieved, based on the values in Supplementary Table 6).

Reporting summary

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

Data availability

The Zenodo depositions for the five phylogenetically compressed test collections are provided in the following table.

Dataset	Compressed form	Zenodo accession/URL
GISP	Assemblies (XZ)	https://doi.org/10.5281/zenodo.10070404
SC2	Assemblies (XZ)	Available upon request (GISAID license).
NCTC3k	Assemblies (XZ)	https://doi.org/10.5281/zenodo.5533354
BIGSIdata	De Bruijn graphs (simplitigs after *k*-mer propagation; XZ)	https://doi.org/10.5281/zenodo.5555253
661k	Assemblies (XZ)	https://doi.org/10.5281/zenodo.4602622
	Assemblies (MBGC)	https://doi.org/10.5281/zenodo.6347064
	k-mer index (COBS; XZ)	https://doi.org/10.5281/zenodo.7313926 https://doi.org/10.5281/zenodo.7313942 https://doi.org/10.5281/zenodo.7315499
661k-HQ	*k*-mer index (COBS; XZ)	https://doi.org/10.5281/zenodo.6845083 https://doi.org/10.5281/zenodo.6849657

Code availability

The GitHub repositories and Zenodo depositions for the developed/modified software are provided in the following table.

Software	Description	GitHub repository	Zenodo accession
Phylign (v0.2.0)	Snakemake pipeline	https://github.com/karel-brinda/phylign/	https://doi.org/10.5281/zenodo.10828249
MiniPhy (v0.4.0)	Snakemake pipeline	https://github.com/karel-brinda/miniphy/	https://doi.org/10.5281/zenodo.10798914
MiniPhy-COBS (v.0.0.1)	Snakemake pipeline	https://github.com/leoisl/miniphy-cobs/	https://doi.org/10.5281/zenodo.14212997
ProPhyle (modified, v0.3.3)	ProPhyle metagenomic classifier	https://github.com/prophyle/prophyle/	https://doi.org/10.5281/zenodo.11004671
COBS (modified, v0.3)	COBS *k*-mer indexer	https://github.com/iqbal-lab-org/cobs/	https://doi.org/10.5281/zenodo.14212977
Attotree (v0.1.6)	An efficient re-implementation of the Mashtree algorithm	https://github.com/karel-brinda/attotree/	https://doi.org/10.5281/zenodo.10945896

References

55. Stamatakis, A. RAxML version 8: a tool for phylogenetic analysis and post-analysis of large phylogenies. *Bioinformatics* **30**, 1312–1313 (2014).
56. Ondov, B. D. et al. Mash: fast genome and metagenome distance estimation using minhash. *Genome Biol.* **17**, 132 (2016).
57. Broder, A. Z. On the resemblance and containment of documents. In *Proc. International Conference on Compression and Complexity of sequences* 21–29 https://doi.org/10.1109/sequen.1997.666900 (IEEE, 1997).
58. Fan, H., Ives, A. R., Surget-Groba, Y. & Cannon, C. H. An assembly and alignment-free method of phylogeny reconstruction from next-generation sequencing data. *BMC Genomics* **16**, 522 (2015).
59. Saitou, N. & Nei, M. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Mol. Biol. Evol.* **4**, 406–425 (1987).
60. Howe, K., Bateman, A. & Durbin, R. QuickTree: building huge Neighbour-Joining trees of protein sequences. *Bioinformatics* **18**, 1546–1547 (2002).
61. Huerta-Cepas, J., Serra, F. & Bork, P. ETE 3: reconstruction, analysis, and visualization of phylogenomic data. *Mol. Biol. Evol.* **33**, 1635–1638 (2016).
62. Köster, J. & Rahmann, S. Snakemake-a scalable bioinformatics workflow engine. *Bioinformatics* **28**, 2520–2522 (2012).
63. Li, H. Seqtk: toolkit for processing sequences in FASTA/Q formats. *GitHub* https://github.com/lh3/seqtk (2016).
64. Marçais, G. & Kingsford, C. A fast, lock-free approach for efficient parallel counting of occurrences of *k*-mers. *Bioinformatics* **27**, 764–770 (2011).
65. Grüning, B. et al. Bioconda: sustainable and comprehensive software distribution for the life sciences. *Nat. Methods* **15**, 475–476 (2018).
66. Grad, Y. H. et al. Genomic epidemiology of gonococcal resistance to extended-spectrum cephalosporins, macrolides, and fluoroquinolones in the United States, 2000–2013. *J. Infect. Dis.* **214**, 1579–1587 (2016).
67. Tange, O. GNU Parallel: the command-line power tool. *The USENIX Magazine* **36**, 42–47 (2011).
68. Larsson, N. J. & Moffat, A. Off-line dictionary-based compression. *Proc. IEEE* **88**, 1722–1732 (2000).
69. Wan, R. *Browsing and Searching Compressed Documents*. PhD thesis, Univ. Melbourne (2003).
70. Cock, P. J. A. et al. Biopython: freely available Python tools for computational molecular biology and bioinformatics. *Bioinformatics* **25**, 1422–1423 (2009).
71. Chikhi, R., Limasset, A. & Medvedev, P. Compacting de Bruijn graphs from sequencing data quickly and in low memory. *Bioinformatics* **32**, i201–i208 (2016).
72. Břinda, K., Sykulski, M. & Kucherov, G. Spaced seeds improve *k*-mer-based metagenomic classification. *Bioinformatics* **31**, 3584–3592 (2015).
73. Li, H. et al. The Sequence Alignment/Map format and SAMtools. *Bioinformatics* **25**, 2078–2079 (2009).

Acknowledgements

This work was supported by the NIGMS of the National Institutes of Health (R35GM133700 to M.B.), the David and Lucile Packard Foundation (to M.B.), the Pew Charitable Trusts (to M.B.), the Alfred P. Sloan Foundation (to M.B.), the European Union's Horizon 2020 research and innovation programme (grant agreement nos. 872539, 956229 and 101047160 to R.C.) and the ANR Transipedia, SeqDigger, Inception and PRAIRIE grants (ANR-18-CE45-0020, ANR-19-CE45-0008, PIA/ANR16-CONV-0005 and ANR-19-P3IA-0001, respectively; to R.C.). Portions of this research were conducted on the O2 high-performance compute cluster, supported by the Research Computing Group at Harvard Medical School, and on the GenOuest bioinformatics core facility (https://www.genouest.org/).

Author contributions

K.B., Z.I. and M.B. designed and conceptualized the method and algorithms and wrote the paper. K.B. wrote the initial draft of the manuscript. K.B. and L.L. wrote the software. K.B. performed the analyses for the study. N.Q.-O., R.C. and G.K. contributed to the conception and design of the work. S.P. and K.S. contributed to the software development. All authors reviewed and approved the final version of the manuscript.

Competing interests

S.P. is currently employed by Eligo Bioscience. The remaining authors declare no competing interests.

Additional information

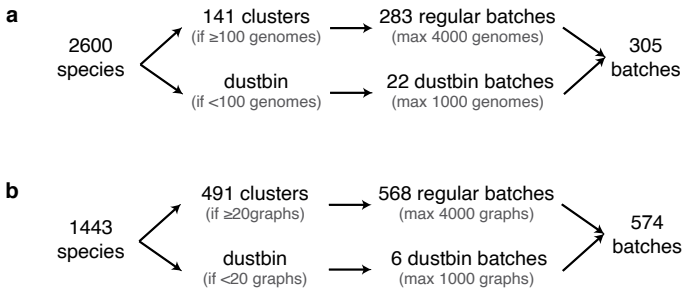
Extended data is available for this paper at https://doi.org/10.1038/s41592-025-02625-2.

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s41592-025-02625-2.

Correspondence and requests for materials should be addressed to Karel Břinda or Michael Baym.

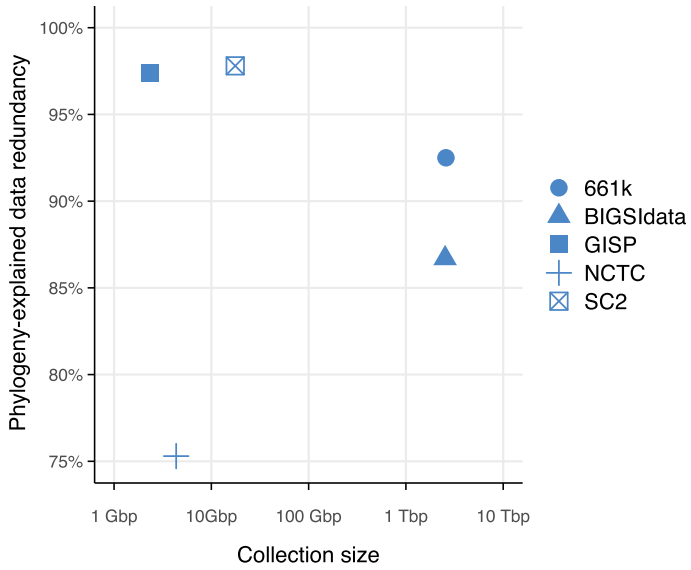
Peer review information *Nature Methods* thanks David Koslicki, Rob Patro and Harihara Subrahmaniam Muralidharan for their contribution to the peer review of this work. Primary Handling Editor: Lin Tang, in collaboration with the *Nature Methods* team. Peer reviewer reports are available.

https://doi.org/10.1038/s41592-025-02625-2



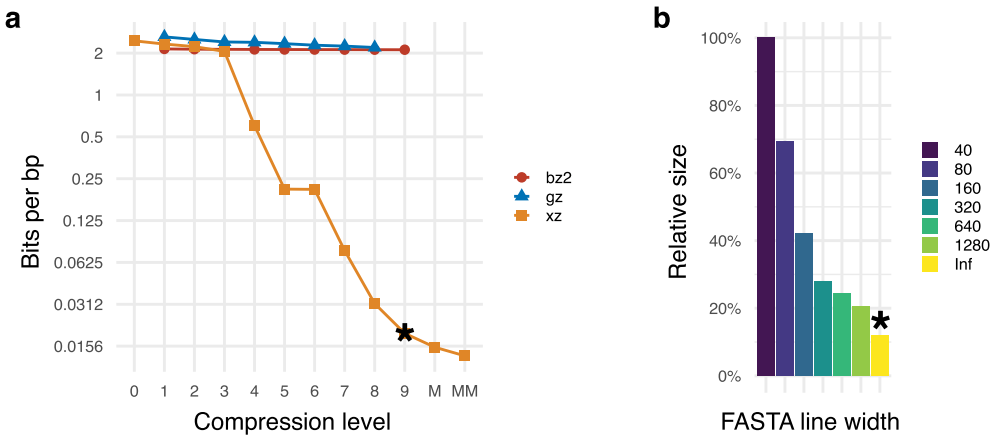
Extended Data Fig. 1 | Batching strategies for the 661k (a) and BIGSIdata (b) collections. Genomes are clustered by species, and clusters that are too small are placed into a common pseudo-cluster called a dustbin. The resulting clusters and the dustbin are then divided into size- and diversity-balanced batches. For more information on batching, see Methods and Supplementary Note 5.

https://doi.org/10.1038/s41592-025-02625-2



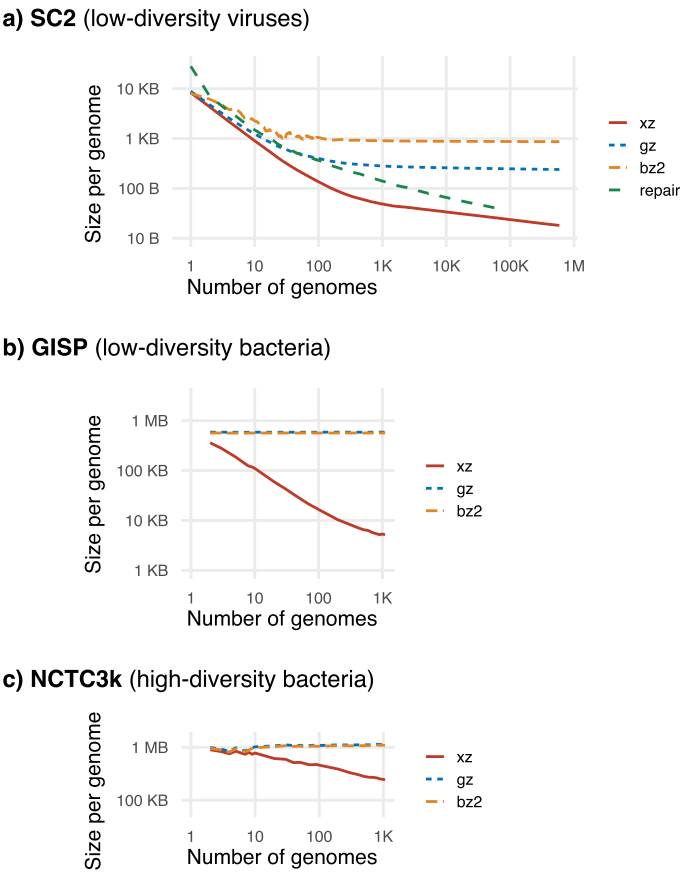
Extended Data Fig. 2 | Quantification of phylogeny-explained data redundancy in the five test collections. The plot depicts the percentage of data redundancy that can be explained by the compressive phylogenies in each of the five test collections. Explained redundancy is measured by bottom-up *k*-mer propagation along the phylogenies performed by ProPhyle and calculated as the proportion of duplicate *k*-mers removed by the propagation ($k = 31$, canonical; see Methods for the formula). A *k*-mer distribution perfectly explained by the associated compressive phylogeny (that is, all *k*-mers associated with complete subtrees) would result in 100% phylogeny-explained redundancy. The plot shows that for single-species batches (modeled by the GISP and SC2 collections), the majority of the signal can be explained by their compressive phylogenies, indicative of their extremely high phylogenetic compressibility (cf. Extended Data Fig. 4a, b). In contrast, high-diversity batches (modeled by the NCTC3k collection) have more irregularly distributed *k*-mer content due to horizontal gene transfer combined with sparse sampling, indicative of their lower compressibility (cf. Extended Data Fig. 4c). Large and diverse collections, such as 661k and BIGSIdata, thus exhibit a medium level of phylogenetically explained redundancies, with the level depending on the amount of noise (higher for BIGSIdata and lower for 661k, as also visible in Extended Data Fig. 7).

https://doi.org/10.1038/s41592-025-02625-2



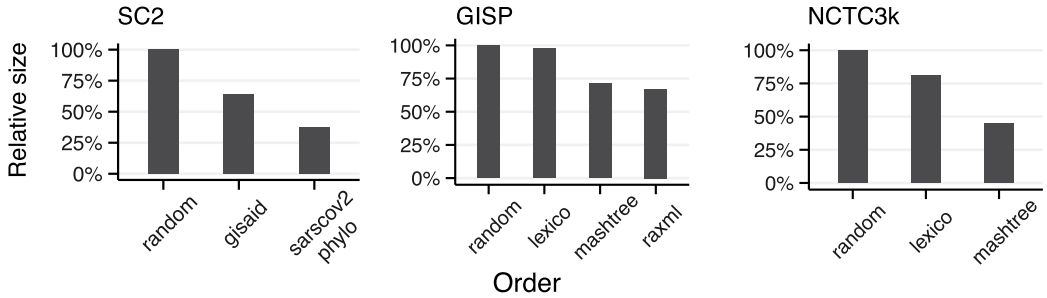
Extended Data Fig. 3 | Calibration of XZ as a low-level tool for phylogenetic compression of assemblies. The comparison was performed using the assemblies from the GISP collection, with genomes sorted left-to-right according to the Mashtree phylogeny. In both plots, an asterisk denotes the mode selected for phylogenetic compression in MiniPhy. **a**) The plot shows the compression performance of XZ, GZip, and BZip2 in bits per bp as a function of compression presets (-1, -2, etc.) with single-line FASTA. Given the specific sizes of dictionaries and windows used in the individual algorithms and their presets, only XZ with a level ≥ 4 was capable of compressing bacterial genomes beyond the statistical entropy baseline (that is, approximately 2 bits per bp). M and MM denote additional, manually tuned compression modes of XZ with increased dictionary sizes (Methods), which slightly improved compression performance but substantially increased memory and CPU time and were thus not used in MiniPhy. **b**) The plot shows the impact of FASTA line length on compression performance. With single-line FASTA (denoted by Inf), the compressed size is reduced to 12% compared to the 40-bp-per-line version. The plot highlights the importance of pre-formatting FASTA data before using general compressors such as XZ.

https://doi.org/10.1038/s41592-025-02625-2



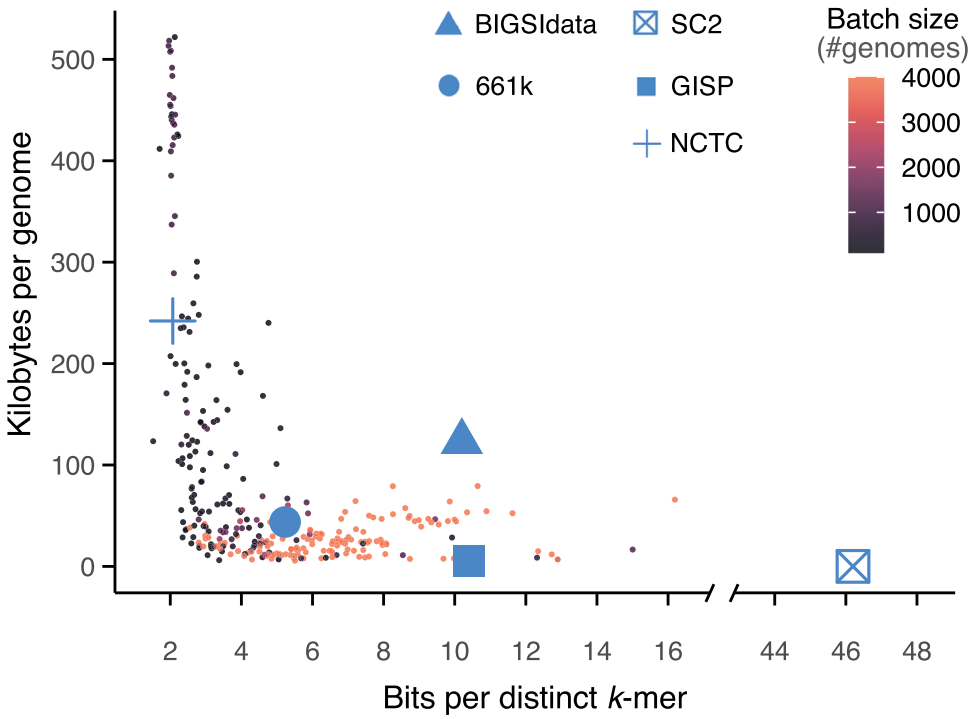
Extended Data Fig. 4 | Comparison of three contrasting compression scaling modes of microbial collections. The plots compare the scaling behavior of the XZ, GZip, BZip2, and Re-Pair compressors on the SC2 (**a**), GISP (**b**), and NCTC3k (**c**) collections, depicting the space per genome as a function of the number of jointly compressed genomes, progressively increased on logarithmic scales. The results highlight several key findings. First, XZ consistently outperforms the other compressors. Second, for viral genomes all four compressors are able to overcome the 2-bits-per-bp baseline thanks to their short genome length, but only XZ is able to compress beyond this limit for bacterial genomes (consistent with Extended Data Fig. 3a; the Re-Pair implementation used could not compress bacterial genomes due to their size). Third, Re-Pair compression can be nearly as effective as XZ for viruses, but its applicability to large datasets is limited by its scalability. Fourth, the compressibility of divergent bacteria is substantially limited even with the best compressors, with only a 4× improvement in per-genome compression for NCTC3k (while the highly compressible SC2 and GISP collections show 171× and 105× improvements for the same number of genomes).

https://doi.org/10.1038/s41592-025-02625-2



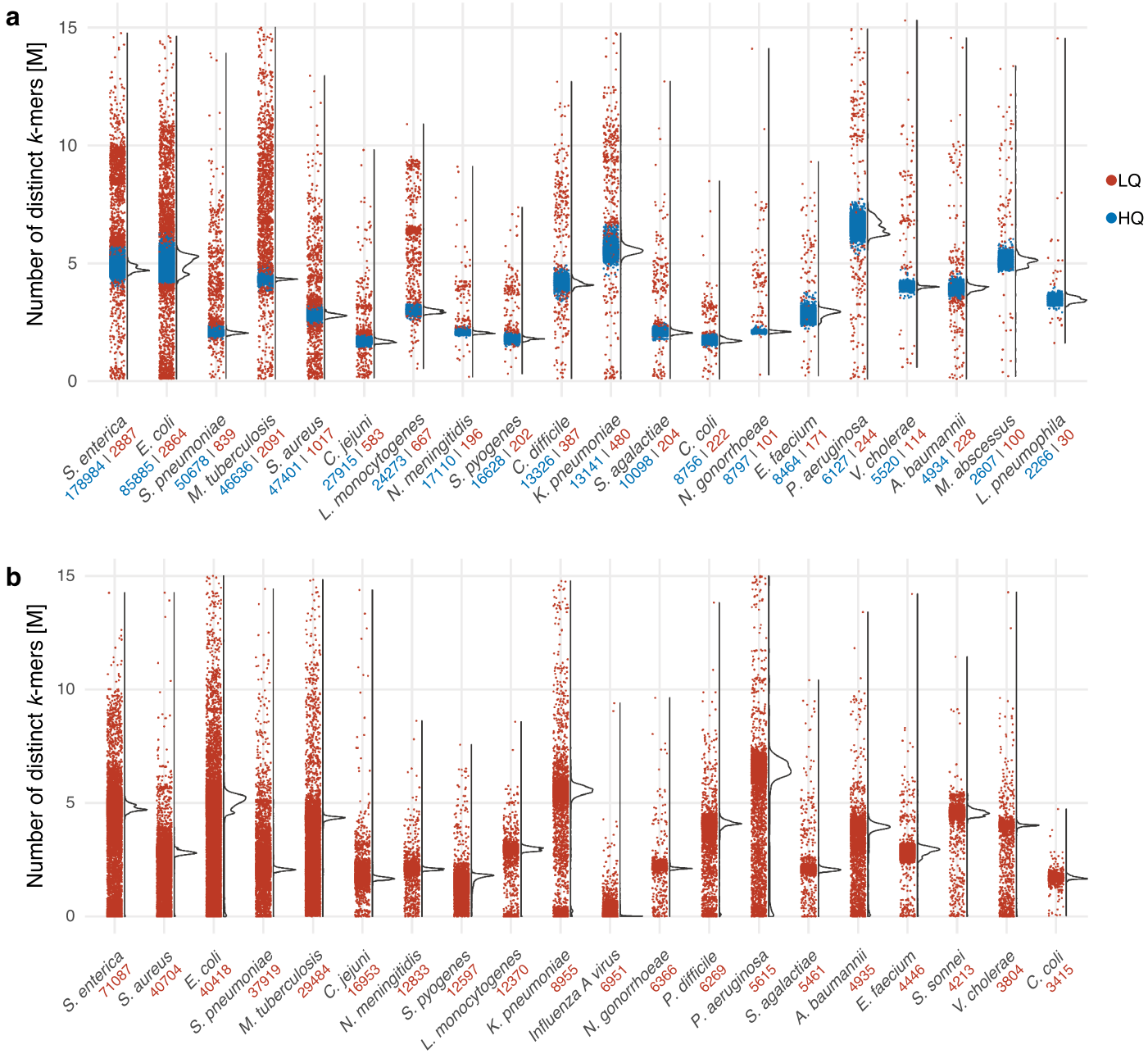
Extended Data Fig. 5 | Impact of within-batch genome order on the compressibility of microbial collections. While a substantial part of the benefits of phylogenetic compression comes from organizing genomes into batches of phylogenetically related genomes, proper genome reordering within individual batches is also crucial for maximizing data compressibility. The plots demonstrate that the impact of within-batch reordering grows with the amount of diversity included (GISP vs. NCTC3k) and with the number of genomes (GISP vs. SC2). Accurate phylogenies inferred using RAxML provided a small compression benefit for assemblies over trees computed using Mashtree (GISP).

https://doi.org/10.1038/s41592-025-02625-2



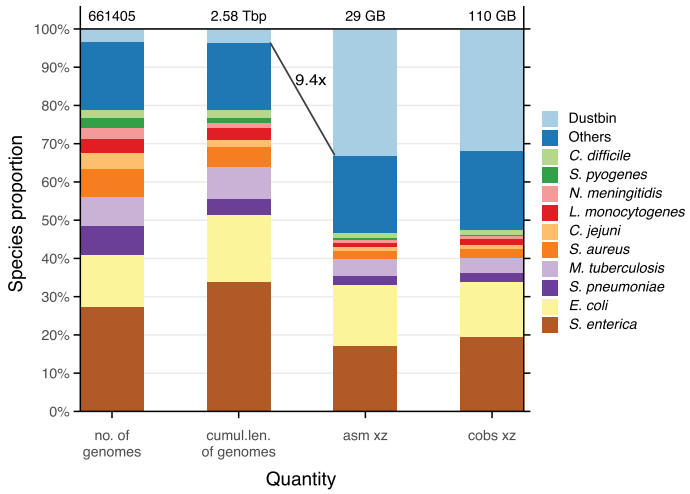
Extended Data Fig. 6 | Compression trade-offs for the five test collections and for individual batches of the 661k collection. The plot illustrates the trade-off between the per-genome size after compression and the number of bits per distinct *k*-mer ($k = 31$, canonical). The larger points represent individual genome collections and correspond to values from Supplementary Table 3. The smaller points represent individual batches within the 661k collection, with colors indicating the number of genomes in each batch. Overall, the plot reveals the influence of genomic diversity on the resulting compression characteristics. The trade-off follows an L-shaped pattern, where compressing genome groups with high diversity leads to smaller space per *k*-mer but larger space per genome, and vice versa for genome groups with low diversity.

https://doi.org/10.1038/s41592-025-02625-2



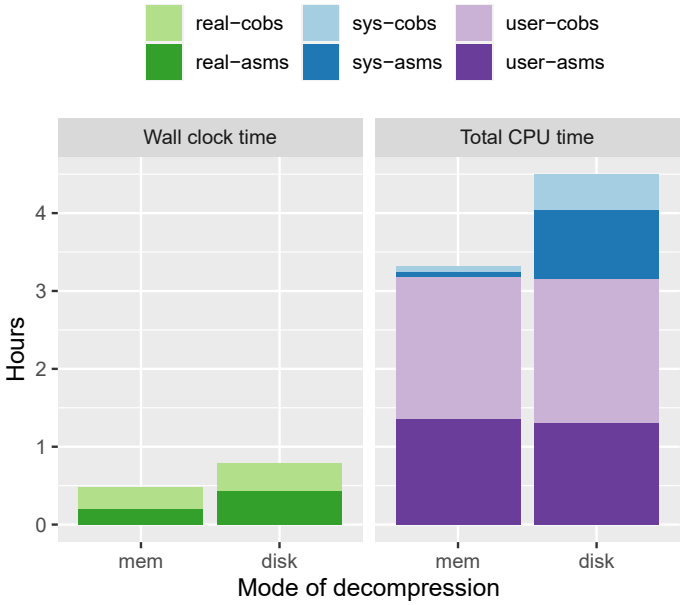
Extended Data Fig. 7 | Distribution of the number of distinct *k*-mers in the top 20 species in (a) the 661k and (b) BIGSIdata collections. For the 661k collection, colors represent the quality of the assemblies (LQ: low-quality, HQ: high-quality), as determined as part of the quality control in the original publication. For BIGSIdata, no quality control information is available. The numbers below the species name indicate the number of samples within each category. The plots were created for canonical 31-mers.

https://doi.org/10.1038/s41592-025-02625-2



Extended Data Fig. 8 | Proportions of top 10 species (their corresponding batches) in the 661k collection before and after phylogenetic compression. The plot depicts the proportions of the top 10 species, the Dustbin pseudo-cluster, and the remaining species grouped as Others, while comparing the following four quantitative characteristics: the number of genomes, their cumulative length, the size of the phylogenetically compressed assemblies, and the size of the phylogenetically compressed COBS indexes (for $k = 31$). Transitioning from the number of genomes to their cumulative length has only a minor impact on the proportions (corresponding to different mean genome lengths of individual species). However, the divergent genomes occupy a substantially higher proportion of the collection after compression. Moreover, despite genome assemblies and k-mer COBS indexes being fundamentally different genome representations (horizontal vs. vertical, respectively), the observed post-compression proportions in them were nearly identical.

https://doi.org/10.1038/s41592-025-02625-2



Extended Data Fig. 9 | Time required for decompressing the Phylign 661k-HQ database. The wall clock and total CPU time required to decompress the Phylign 661k-HQ database, both from disk and in memory, were measured on an iMac desktop computer with 4 physical (8 logical) cores. The in-memory decompression process, which is implemented in Phylign, was completed under 30 min. This duration represents only a fraction of the typical time required for search experiments (see Supplementary Table 6).

nature portfolio

Corresponding author(s): Karel Brinda

Last updated by author(s): Nov 25, 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 downloaded from public repositories as described in Methods using standard Unix tools such as wget and curl.
Data analysis	All the software developed within this study is available from Github and is documented in the Methods section and in the Code Availability Statement. The software used in this paper includes: Attotree (v0.1.2, v0.1.6) COBS (v0.2.1, v0.3) ETE3 (v3.1.3) GNU Make (v4.3) Jellyfish (v2.2.10) Mash (v1.1) Mashtree (v1.4.6) MBGC (v1.2.1) McCortex (v1.0) Miniphy (v0.4.0) MiniPhy-COBS (v0.0.1) Minimap2 (v2.24) Pandas (v2.1.1) Phylign (v0.2.0) ProphAsm (v0.1.1)

ProPhyle (v0.3.2.0, v0.3.3.0)
Re-Pair (https://github.com/rwanwork/Re-Pair, commit 093260)
Quicktree (v2.5)
Seqtk (v1.3, v1.4)
Xopen (v1.7.0, v0.7.3)

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

All data produced have accessions provided in the Data Availability section.

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	The paper presents new methods for compression and search of large genome collections, and as such sample sizes do not affect the statistical validity or conclusions of the study. The genome collection sizes ranged from n=1065 (the NCTC3k collection) to n=661405 (the 661k collection).
Data exclusions	No data were excluded from the analyses. For BLAST-like alignment using Phylign, the index contains only those assemblies that had passed quality control in the original study (96.3% of assemblies).
Replication	All the code and data were made publicly available. The functionality and performance of Phylign and MiniPhy was independently verified on multiple other desktop and laptop computers. The method was also independently verified by other studies, including the paper about the AllTheBacteria genome collection (https://doi.org/10.1101/2024.03.08.584059), where phylogenetic compression using MiniPhy was used as the central compression method.
Randomization	No experimental groups were used.
Blinding	Not applicable, as this is a software method paper.

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