



**HAL**  
open science

## Efficient and Robust Search of Microbial Genomes via Phylogenetic Compression

Karel Břinda, Leandro Lima, Simone Pignotti, Natalia Quinones-Olvera,  
Kamil Salikhov, Rayan Chikhi, Gregory Kucherov, Zamin Iqbal, Michael  
Baym

► **To cite this version:**

Karel Břinda, Leandro Lima, Simone Pignotti, Natalia Quinones-Olvera, Kamil Salikhov, et al.. Efficient and Robust Search of Microbial Genomes via Phylogenetic Compression. 2024. hal-04287842v2

**HAL Id: hal-04287842**

**<https://hal.science/hal-04287842v2>**

Preprint submitted on 13 May 2024

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

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



Distributed under a Creative Commons Attribution - NonCommercial 4.0 International License

# Efficient and Robust Search of Microbial Genomes via Phylogenetic Compression

Karel Břinda <sup>1,2\*</sup>, Leandro Lima <sup>3</sup>, Simone Pignotti <sup>2,4</sup>, Natalia Quinones-Olvera <sup>2</sup>, Kamil Salikhov <sup>4</sup>, Rayan Chikhi <sup>5</sup>, Gregory Kucherov <sup>4</sup>, Zamin Iqbal <sup>3</sup>, Michael Baym <sup>2\*</sup>

1 Inria, Irista, Univ. Rennes, 35042 Rennes, France

2 Department of Biomedical Informatics, Harvard Medical School, MA 02115 Boston, USA

3 EMBL-EBI, CB10 1SD Hinxton, UK

4 LIGM, CNRS, Univ. Gustave Eiffel, 77454 Marne-la-Vallée, France

5 Institut Pasteur, Univ. Paris Cité, G5 Sequence Bioinformatics, 75015 Paris, France

\* Correspondence to [karel.brinda@inria.fr](mailto:karel.brinda@inria.fr) and [baym@hms.harvard.edu](mailto:baym@hms.harvard.edu)

## 1 **ABSTRACT**

2

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

## 16 INTRODUCTION

17

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

25

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

31

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

40

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

52 compression to the more tractable problem of local compression of phylogenetically grouped and  
53 ordered genomes.

54

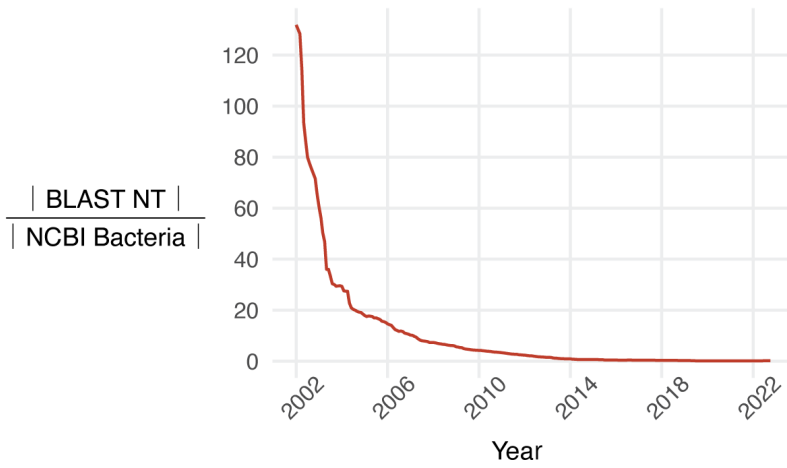
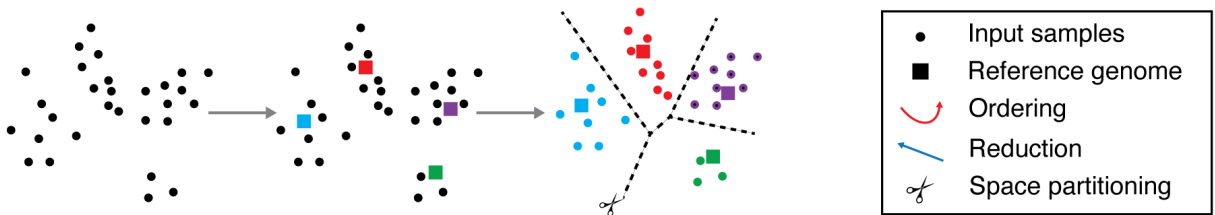
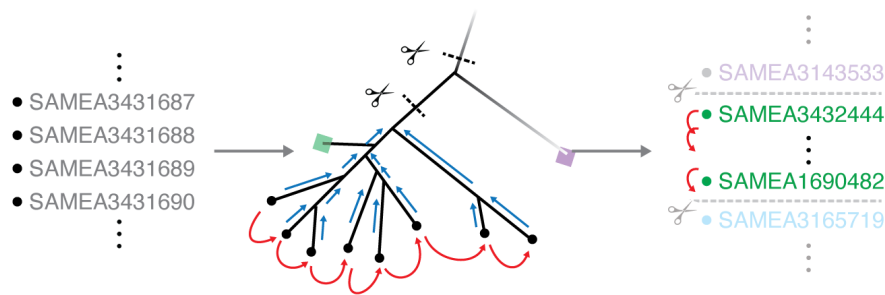
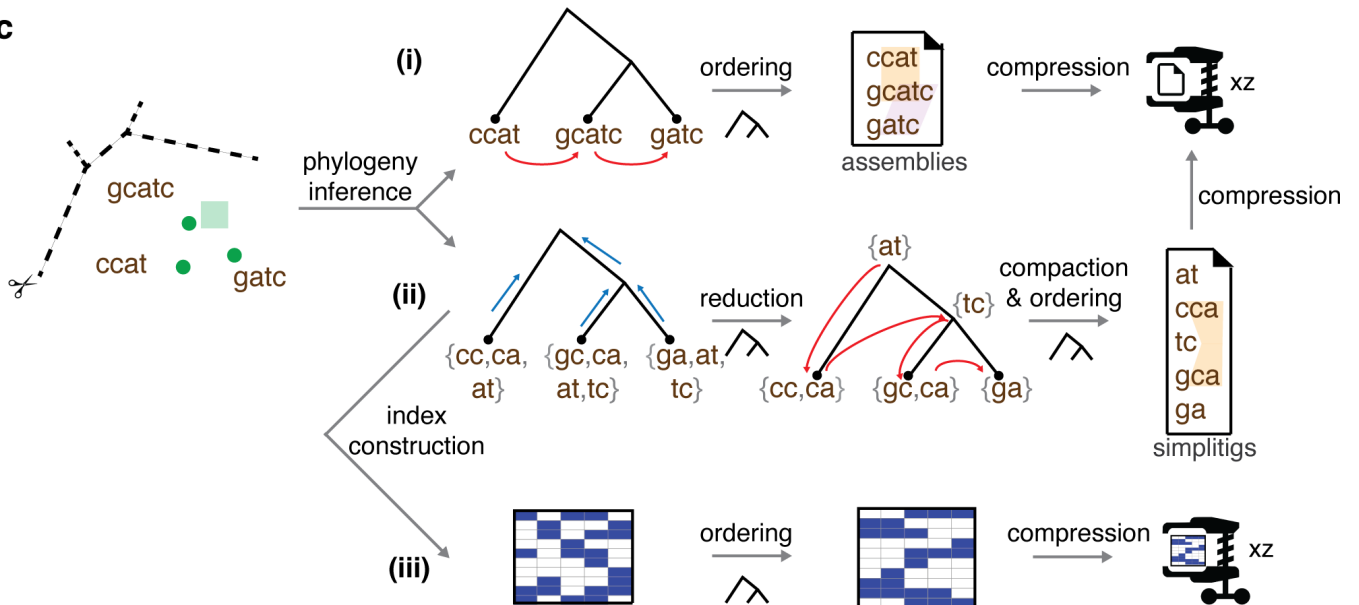
55 Phylogenetic relatedness is effective at estimating the similarity and compressibility of microbial  
56 genomes and their data representations. The closer two genomes are phylogenetically, the closer they  
57 are likely to be in terms of mathematical similarity measures, such as the edit distance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

65

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

72

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

**a****b (i)****(ii)****c**

84 **Fig. 1: Overview of phylogenetic compression and its applications to different data types.**

85 **a)** Exponential decrease of data searchability over the past two decades illustrated by the size of the  
86 BLAST NT database divided by the size of the NCBI Bacterial Assembly database. **b)** The first three  
87 stages of phylogenetic compression prior to the application of a low-level compressor/indexer. **(i)** A  
88 given collection is partitioned into size- and diversity-balanced batches of phylogenetically related  
89 genomes (e.g., using metagenomic classification of the original reads). **(ii)** The input data are reversibly  
90 reordered based on a compressive phylogeny, performed separately for each batch. **c)** Examples of  
91 specific protocols for phylogenetic compression of individual data types, performed separately for each  
92 batch. **(i)** Assemblies are sorted left-to-right according to the topology of the phylogeny, and then  
93 compressed using a low-level compressor such as XZ<sup>7,32</sup> or MBGC<sup>18</sup>. **(ii)** For de Bruijn graphs, *k*-mers  
94 are propagated bottom-up along the phylogeny, and the resulting *k*-mer sets are compacted into  
95 simplitigs<sup>33</sup>, which are then compressed using XZ. **(iii)** For BIGSI *k*-mer indexes, Bloom filters (in  
96 columns) are ordered left-to-right according to the phylogeny, and then compressed using XZ.

97

98

99 **RESULTS**

100

101 We developed a technique called phylogenetic compression for evolutionarily informed compression and  
102 search of microbial collections (**Fig. 1**, <https://brinda.eu/mof>). Phylogenetic compression combines  
103 four ingredients (**Fig. 1b**): 1) *clustering* of samples into phylogenetically related groups, followed by  
104 2) inference of a *compressive phylogeny* that acts as a template for 3) *data reordering*, prior to 4) the  
105 application of a calibrated *low-level compressor/indexer* (Methods). This general scheme can be  
106 instantiated to individual protocols for various data types as we show in **Fig. 1c**; for instance, a set of  
107 bacterial assemblies can be phylogenetically compressed by XZ (the Lempel-Ziv Markov-Chain  
108 Algorithm 7, implemented in XZ Utils<sup>32</sup>) by a left-to-right enumeration of the assemblies, with respect to  
109 the topology of their compressive phylogeny obtained via sketching<sup>34</sup>.

110

111 We implemented phylogenetic compression protocols for assemblies, for de Bruijn graphs, and for *k*-  
112 mer indexes in a tool called MiniPhy (Minimization via Phylogenetic compression,  
113 <https://github.com/karel-brinda/miniphy>). To cluster input genomes, MiniPhy builds upon the  
114 empirical observation that microbial genomes in public repositories tend to form clusters corresponding  
115 to individual species<sup>35</sup>, and species for individual genomes can be identified rapidly via metagenomic  
116 classification<sup>36</sup> (**Fig. 1b**, Methods). As some of the resulting clusters may be too large or too small, and  
117 thus unbalancing downstream parallelization, it further redistributes the clustered genomes into size-  
118 and diversity-balanced batches (Methods, **Supplementary Fig. 1**). This batching enables compression  
119 and search in a constant time (using one node per batch on a cluster) or linear time (using a single

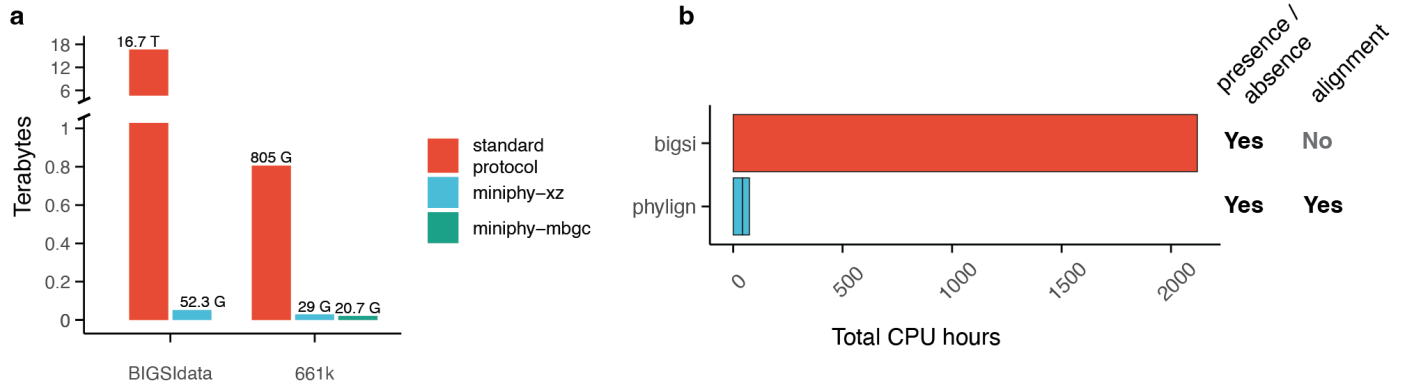
120 machine) (Methods). For every batch, a compressive phylogeny – either provided by the user or  
121 computed automatically using Mashtree<sup>34</sup> / Attotree (<https://github.com/karel-brinda/attotree>,  
122 Methods) – is then used for data reordering (Methods). Finally, the obtained reordered data are  
123 compressed per batch using XZ with particularly optimized parameters (Methods), and possibly further  
124 re-compressed or indexed using some general or specialized low-level tool, such as MBGC<sup>18</sup> or COBS<sup>37</sup>  
125 (Methods).

126

127 We evaluated phylogenetic compression using five microbial collections, selected as representatives of  
128 the compression-related tradeoffs between characteristics including data quality, genetic diversity,  
129 genome size, and collection size (GISP, NCTC3k, SC2, 661k, and BIGSIdata; Methods, **Supplementary**  
130 **Table 1**). We quantified the distribution of their underlying phylogenetic signal (Methods,  
131 **Supplementary Table 2, Supplementary Fig. 2**), used them to calibrate the individual steps of the  
132 phylogenetic compression workflow (Methods, **Supplementary Fig. 3–5**), and evaluated the resulting  
133 performance, tradeoffs, and extremal characteristics (Methods, **Supplementary Table 3,**  
134 **Supplementary Fig. 6**). As one extreme, we found that 591k SARS-CoV-2 genomes can be  
135 phylogenetically compressed using XZ to only 18.1 bytes/genome (Methods, **Supplementary Table 3,**  
136 **Supplementary Fig. 4, 6**), resulting in a file size of 10.7 Mb (13.2× more compressed than GZip). A  
137 summary detailing the sensitivity/stability of performance to various factors is provided in  
138 **Supplementary Note 1**.

139

140 We found that phylogenetic compression improved the compression of genome assembly collections  
141 that comprise hundreds of thousands of isolates of over 1,000 species by more than an order of  
142 magnitude compared to the state-of-the-art (**Fig. 2a, Supplementary Table 3**). Specialized high-  
143 efficiency compressors such as MBGC<sup>18</sup> are not directly applicable to highly diverse collections,  
144 therefore, the compression protocols deployed in practice for extremely large and diverse collections are  
145 still based on the standard GZip, such as the 661k collection, containing all bacteria pre-2019 from  
146 ENA<sup>16</sup> (n=661,405, 805 GB). Here, MiniPhy recompressed the collection to 29.0 GB (27.8×  
147 improvement; 43.8 KB/genome, 0.0898 bits/bp, 5.23 bits/distinct *k*-mer) using XZ as a low-level tool,  
148 and further to 20.7 GB (38.9× improvement; 31.3 KB/genome, 0.0642 bits/bp, 3.74 bits/distinct *k*-mer)  
149 when combined with MBGC<sup>18</sup> that also accounts for reverse complements (**Fig. 2a, Supplementary**  
150 **Table 3**, Methods). Additionally, we found that the lexicographically ordered ENA datasets, as being  
151 partially phylogenetically ordered, can serve as an approximation of phylogenetic compression, with  
152 compression performance only degraded by a factor of 4.17 compared to full phylogenetic compression  
153 (**Supplementary Table 3**, Methods). The resulting compressed files are provided for download from  
154 Zenodo (**Supplementary Table 4**).



156

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

163

164

165 We then studied de Bruijn graphs, a common genome representation directly applicable to raw-read  
 166 data<sup>17,38</sup>, and found that phylogenetic compression can improve state-of-the-art approaches by one-to-  
 167 two orders of magnitude (**Fig. 2a, Supplementary Table 3, Methods**). As standard and colored de  
 168 Bruijn graphs lack methods for joint compression at the scale of millions of genomes and thousands of  
 169 species, single graphs are often distributed individually<sup>39</sup>. For instance, the graphs of the BIGSIdata  
 170 collection<sup>17</sup>, comprising all viral and bacterial genomes from pre-2017 ENA (n=447,833), are provided  
 171 in an online repository in the McCortex binary format<sup>40</sup> and occupy in total >16.7 TB (Methods). Here,  
 172 we retrieved n=425,160 graphs from the Internet (94.5% of the original count) (Methods) and losslessly  
 173 recompressed them using the MiniPhy methodology, with a bottom-up propagation of the *k*-mer  
 174 content, to 52.3 GB (319× improvement; 123. KB/genome, 0.248 bits/unitig bp, 10.2 bits/distinct *k*-  
 175 mer) (**Fig. 2a, Supplementary Table 3, Methods**). Further, as recent advances in de Bruijn graph  
 176 indexing<sup>15</sup> may lead to more efficient storage protocols in the future, we also compared MiniPhy to  
 177 MetaGraph<sup>38</sup>, an optimized tool for indexing on high-performance servers with a large amount of  
 178 memory. Here, we found that MiniPhy still provided an improvement of a factor of 5.78 (Methods).

179

180 Phylogenetic compression can be applied to any genomic data structure based on a genome-similarity-  
 181 preserving representation (Methods, **Supplementary Note 2**). We demonstrate this using the  
 182 Bitsliced Genomic Signature Index (BIGSI)<sup>17</sup> (**Fig. 1c(iii)**), a *k*-mer indexing method using an array of



183 Bloom filters, which is widely used for large-scale genotyping and presence/absence queries of genomic  
184 elements<sup>16,17</sup>. Using the same data, batches, and orders as inferred previously, we phylogenetically  
185 compressed the BIGSI indexes of the 661k collection, computed using a modified version of COBS<sup>37</sup>  
186 (**Supplementary Table 5**, Methods). Phylogenetic compression provided an 8.51× overall  
187 improvement compared to the original index (from 937 GB to 110 GB), making it finally usable on  
188 ordinary computers. After we further omitted the 3.7% genomes that had not passed quality control in  
189 the original study<sup>16</sup> (the 661k-HQ collection, visualized in **Supplementary Fig. 7**), the resulting  
190 phylogenetic compression ratio improved to 12.3× (72.8 GB) (**Supplementary Table 5**).

191  
192 To better understand the impact of phylogenetic compression across the tree of life, we analyzed the  
193 661k MiniPhy batches of assemblies and COBS indexes, both before and after compression  
194 (**Supplementary Fig. 8**). We found that although the top ten species constituted 80% of the genomic  
195 content, they occupied less than half of the database space post-compression for both genome  
196 representations (**Supplementary Fig. 8**). Conversely, the ‘dustbin’ batches, which include genomes  
197 from sparsely sampled species, expanded to occupy a proportion that was 9.4× larger in the database  
198 post-compression, compared to their precompression proportion, again for both representations  
199 (**Supplementary Fig. 8**). This consistent effect of compression on both assemblies and COBS indexes  
200 suggests that phylogenetic compressibility adheres to the same principles, irrespective of the specific  
201 genome representation used, with divergent genomes being a major driver of the final size.

202  
203 To demonstrate the practical utility of phylogenetic compression, we used it to implement BLAST-like  
204 search across all high-quality pre-2019 bacteria for standard desktop and laptop computers (Phylign,  
205 <http://github.com/karel-brinda/phyalign>, Methods). For a given a set of queries, Phylign first identifies  
206 for each query those genomes that match best globally across the whole 661k-HQ collection, by  
207 proceeding via progressive in-memory decompression and querying of individual phylogenetically  
208 compressed COBS<sup>37</sup> *k*-mer indexes (described above). Subsequently, Phylign iterates over the  
209 phylogenetically compressed genome assemblies (described above) and computes the corresponding full  
210 alignments using on-the-fly instances of Minimap 2<sup>41</sup> (Methods). The choice of tools was arbitrary, and  
211 other programs or core data structures could readily be used instead. The resulting requirements  
212 amount to only 102 GB disk (for the compressed COBS indexes and assemblies: 195 KB/genome, 0.329  
213 bits/bp, 23.0 bits/distinct *k*-mer) (**Supplementary Table 6**) and 12 GB RAM, and Phylign can thus be  
214 deployed on most modern laptop and desktop computers.

215  
216 We first evaluated Phylign with 661k-HQ using three different types of queries – resistance genes (the  
217 entire ARG-ANNOT database of resistance genes<sup>42</sup>, n=1,856), plasmids (EBI plasmid database,  
218 n=2,826), and a nanopore sequencing experiment (n=158,583 reads), with results available within 3.9,

219 11, and 4.3 hours, respectively, on an iMac desktop (**Supplementary Table 7**). Benchmarking against  
220 other tools was not possible, as we were unable to find any tool capable of aligning queries to 661k-HQ  
221 in a comparable setup. We therefore used the EBI plasmid dataset to compare Phylign to BIGSI with its  
222 original database of 448k genomes (which is essentially a subset of 661k-HQ with 1.43× less genomes)<sup>17</sup>.  
223 We found that Phylign was over an order of magnitude faster (**Fig. 2b, Supplementary Table 7**); the  
224 search required 74.1 CPU hours and improved performance by a factor of 28.6× compared to the same  
225 BIGSI benchmark with its smaller database (**Fig. 2b, Supplementary Table 7**), while providing the  
226 full alignments rather than presence/absence only (**Fig. 2b**). To our knowledge, this is the first time  
227 that alignment to a collection of a comparable size and diversity has been locally performed.

228

229

## 230 **DISCUSSION**

231

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

244

245 Elements of our approach and related techniques have been previously used in other contexts.  
246 Reversible reordering to improve compression forms the core of the Burrows-Wheeler Transform<sup>21</sup> and  
247 its associated indexes<sup>43-45</sup>, and it has also been used for read compression<sup>22-25</sup>. Tree hierarchies have  
248 been applied in metagenomics for both lossy<sup>26,27,46</sup> and lossless<sup>28</sup> reference data compression. Finally, a  
249 divide-and-conquer methodology has been employed to accelerate the inference of species trees<sup>47</sup>.  
250 However, this is the first time all these ideas have been combined together to improve the scalability of  
251 search in large genome databases.

252

253 As with all forms of compression, our ability to reduce data is fundamentally limited by the underlying  
254 entropy. For genome collections, this is not introduced solely by the underlying genetic signal, but it is

255 also tightly connected with the sequencing process and our capacity to reconstruct genomes from  
256 sequencing reads. The noise in the underlying  $k$ -mer histograms (**Supplementary Fig. 7**) suggests  
257 that any method for compression or search will have to address noise in the forms of contamination,  
258 missing regions, and technological artifacts, with legacy data posing a major challenge for both storage  
259 and analysis. Future methods may choose to incorporate stricter filtering, and as our experiments have  
260 demonstrated, this not only helps in reducing data volume but also in improving the quality of search  
261 outputs. These issues may be alleviated by innovative computational strategies, such as taxonomic filters  
262 <sup>48</sup> or sweep deconvolution <sup>49</sup>.

263

264 In light of technological development, the benefits of phylogenetic compression will grow over time.  
265 Currently, only a fraction of the world's microbial diversity has been sequenced. However, as sequencing  
266 becomes more comprehensive, the tree of life will not change, thus enhancing the relative advantage of  
267 phylogenetic compression. We foresee its use ranging from mobile devices to large-scale distributed  
268 cloud environments and anticipate promising applications in global epidemiological surveillance <sup>50</sup> and  
269 rapid diagnostics <sup>51</sup>. Overall, the phylogenetic compression of data structures has broad applications  
270 across computational biology and represents a fundamental design principle for future genomics  
271 infrastructure.

272

273

## 274 REFERENCES

275

- 276 1. Stephens, Z. D. *et al.* Big Data: Astronomical or Genomical? *PLoS Biol.* **13**, e1002195 (2015).
- 277 2. Altschul, S. F., Gish, W., Miller, W., Myers, E. W. & Lipman, D. J. Basic local alignment search tool.  
278 *J. Mol. Biol.* **215**, 403–410 (1990).
- 279 3. Navarro, G. & Mäkinen, V. Compressed full-text indexes. *ACM Computing Surveys* **39**, 2-es (2007).
- 280 4. Loh, P.-R., Baym, M. & Berger, B. Compressive genomics. *Nat. Biotechnol.* **30**, 627–630 (2012).
- 281 5. Yu, Y. W., Daniels, N. M., Danko, D. C. & Berger, B. Entropy-Scaling Search of Massive Biological  
282 Data. *Cell Systems* **1**, 130–140 (2015).
- 283 6. Giancarlo, R., Scaturro, D. & Utro, F. Textual data compression in computational biology: a  
284 synopsis. *Bioinformatics* **25**, 1575–1586 (2009).
- 285 7. Salomon, D. & Motta, G. Dictionary Methods. in *Handbook of Data Compression* 329–441  
286 (Springer London, London, 2010).
- 287 8. Daniels, N. M. *et al.* Compressive genomics for protein databases. *Bioinformatics* **29**, i283–90

- 288 (2013).
- 289 9. Deorowicz, S. & Grabowski, S. Data compression for sequencing data. *Algorithms Mol. Biol.* **8**, 25  
290 (2013).
- 291 10. Giancarlo, R., Rombo, S. E. & Utro, F. Compressive biological sequence analysis and archival in the  
292 era of high-throughput sequencing technologies. *Brief. Bioinform.* (2013) doi:10.1093/bib/bbto88.
- 293 11. Zhu, Z., Zhang, Y., Ji, Z., He, S. & Yang, X. High-throughput DNA sequence data compression.  
294 *Brief. Bioinform.* **16**, 1–15 (2015).
- 295 12. Hosseini, M., Pratas, D. & Pinho, A. J. A Survey on Data Compression Methods for Biological  
296 Sequences. *Information* **7**, 56 (2016).
- 297 13. Jayasankar, U., Thirumal, V. & Ponnurangam, D. A survey on data compression techniques: From  
298 the perspective of data quality, coding schemes, data type and applications. *Journal of King Saud*  
299 *University - Computer and Information Sciences* **33**, 119–140 (2021).
- 300 14. Navarro, G. Indexing Highly Repetitive String Collections, Part I: Repetitiveness Measures. *ACM*  
301 *Comput. Surv.* **54**, 1–31 (2021).
- 302 15. Marchet, C. *et al.* Data structures based on k-mers for querying large collections of sequencing data  
303 sets. *Genome Res.* **31**, 1–12 (2021).
- 304 16. Blackwell, G. A. *et al.* Exploring bacterial diversity via a curated and searchable snapshot of  
305 archived DNA sequences. *PLoS Biol.* **19**, e3001421 (2021).
- 306 17. Bradley, P., den Bakker, H. C., Rocha, E. P. C., McVean, G. & Iqbal, Z. Ultrafast search of all  
307 deposited bacterial and viral genomic data. *Nat. Biotechnol.* **37**, 152–159 (2019).
- 308 18. Grabowski, S. & Kowalski, T. M. MBGC: Multiple Bacteria Genome Compressor. *Gigascience* **11**,  
309 (2022).
- 310 19. Deorowicz, S., Danek, A. & Li, H. AGC: compact representation of assembled genomes with fast  
311 queries and updates. *Bioinformatics* **39**, (2023).
- 312 20. Zielezinski, A., Vinga, S., Almeida, J. & Karlowski, W. M. Alignment-free sequence comparison:  
313 benefits, applications, and tools. *Genome Biol.* **18**, 186 (2017).
- 314 21. Burrows, M. & Wheeler, D. J. *A Block-Sorting Lossless Data Compression Algorithm.* (1994).

- 315 22. Hach, F., Numanagic, I., Alkan, C. & Sahinalp, S. C. SCALCE: boosting sequence compression  
316 algorithms using locally consistent encoding. *Bioinformatics* **28**, 3051–3057 (2012).
- 317 23. Patro, R. & Kingsford, C. Data-dependent bucketing improves reference-free compression of  
318 sequencing reads. *Bioinformatics* **31**, 2770–2777 (2015).
- 319 24. Grabowski, S., Deorowicz, S. & Roguski, Ł. Disk-based compression of data from genome  
320 sequencing. *Bioinformatics* **31**, 1389–1395 (2015).
- 321 25. Chandak, S., Tatwawadi, K. & Weissman, T. Compression of genomic sequencing reads via hash-  
322 based reordering: algorithm and analysis. *Bioinformatics* **34**, 558–567 (2018).
- 323 26. Wood, D. E. & Salzberg, S. L. Kraken: ultrafast metagenomic sequence classification using exact  
324 alignments. *Genome Biol.* **15**, R46 (2014).
- 325 27. Kim, D., Song, L., Breitwieser, F. P. & Salzberg, S. L. Centrifuge: rapid and sensitive classification of  
326 metagenomic sequences. *Genome Res.* **26**, 1721–1729 (2016).
- 327 28. Břinda, K. Novel computational techniques for mapping and classification of Next-Generation  
328 Sequencing data. (Université Paris-Est, 2016).
- 329 29. Břinda, K., Salikhov, K., Pignotti, S. & Kucherov, G. *ProPhyle: An Accurate, Resource-Frugal and*  
330 *Deterministic DNA Sequence Classifier*. (Zenodo, 2017). doi:10.5281/zenodo.1045429.
- 331 30. Ge, H., Sun, L. & Yu, J. Fast batch searching for protein homology based on compression and  
332 clustering. *BMC Bioinformatics* **18**, (2017).
- 333 31. Reiter, T. Clustering the NCBI nr database to reduce database size and enable faster BLAST  
334 searches. Preprint at <https://doi.org/10.57844/ARCADIA-W8XT-PC81> (2023).
- 335 32. Collin, L., Tan, J. & Pavlov, I. XZ Utils. Preprint at <https://tukaani.org/xz/> (2009).
- 336 33. Břinda, K., Baym, M. & Kucherov, G. Simplitigs as an efficient and scalable representation of de  
337 Bruijn graphs. *Genome Biol.* **22**, 96 (2021).
- 338 34. Katz, L. *et al.* Mashtree: a rapid comparison of whole genome sequence files. *J. Open Source Softw.*  
339 **4**, 1762 (2019).
- 340 35. Jain, C., Rodriguez-R, L. M., Phillippy, A. M., Konstantinidis, K. T. & Aluru, S. High throughput ANI  
341 analysis of 90K prokaryotic genomes reveals clear species boundaries. *Nat. Commun.* **9**, 5114

- 342 (2018).
- 343 36. Breitwieser, F. P., Lu, J. & Salzberg, S. L. A review of methods and databases for metagenomic  
344 classification and assembly. *Brief. Bioinform.* **20**, 1125–1136 (2019).
- 345 37. Bingmann, T., Bradley, P., Gauger, F. & Iqbal, Z. COBS: A Compact Bit-Sliced Signature Index. in  
346 *String Processing and Information Retrieval* 285–303 (Springer International Publishing, 2019).
- 347 38. Karasikov, M. *et al.* MetaGraph: Indexing and Analysing Nucleotide Archives at Petabase-scale.  
348 *Cold Spring Harbor Laboratory* 2020.10.01.322164 (2020) doi:10.1101/2020.10.01.322164.
- 349 39. Rahman, A., Chikhi, R. & Medvedev, P. Disk compression of k-mer sets. *Algorithms Mol. Biol.* **16**,  
350 10 (2021).
- 351 40. Turner, I., Garimella, K. V., Iqbal, Z. & McVean, G. Integrating long-range connectivity information  
352 into de Bruijn graphs. *Bioinformatics* **34**, 2556–2565 (2018).
- 353 41. Li, H. Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics* 1–3 (2018).
- 354 42. Gupta, S. K. *et al.* ARG-ANNOT, a New Bioinformatic Tool To Discover Antibiotic Resistance Genes  
355 in Bacterial Genomes. *Antimicrob. Agents Chemother.* **58**, 212–220 (2014).
- 356 43. Ferragina, P. & Manzini, G. Opportunistic data structures with applications. in *Proceedings 41st*  
357 *Annual Symposium on Foundations of Computer Science* 390–398 (IEEE Comput. Soc, 2000).
- 358 44. Gagie, T., Navarro, G. & Prezza, N. Fully Functional Suffix Trees and Optimal Text Searching in  
359 BWT-Runs Bounded Space. *J. ACM* **67**, 1–54 (2020).
- 360 45. Zakeri, M., Brown, N. K., Ahmed, O. Y., Gagie, T. & Langmead, B. Movi: a fast and cache-efficient  
361 full-text pangenome index. *bioRxiv* (2023) doi:10.1101/2023.11.04.565615.
- 362 46. Ames, S. K. *et al.* Scalable metagenomic taxonomy classification using a reference genome database.  
363 *Bioinformatics* **29**, 2253–2260 (2013).
- 364 47. Molloy, E. K. & Warnow, T. Statistically consistent divide-and-conquer pipelines for phylogeny  
365 estimation using NJMerge. *Algorithms Mol. Biol.* **14**, 14 (2019).
- 366 48. Goig, G. A., Blanco, S., Garcia-Basteiro, A. L. & Comas, I. Contaminant DNA in bacterial sequencing  
367 experiments is a major source of false genetic variability. *BMC Biol.* **18**, 24 (2020).
- 368 49. Mäklin, T. *et al.* Bacterial genomic epidemiology with mixed samples. *Microb Genom* **7**, (2021).

- 369 50. Gardy, J. L. & Loman, N. J. Towards a genomics-informed, real-time, global pathogen surveillance  
370 system. *Nat. Rev. Genet.* (2017) doi:10.1038/nrg.2017.88.
- 371 51. Břinda, K. *et al.* Rapid inference of antibiotic resistance and susceptibility by genomic neighbour  
372 typing. *Nat Microbiol* **5**, 455–464 (2020).
- 373 52. Kitts, P. A. *et al.* Assembly: a resource for assembled genomes at NCBI. *Nucleic Acids Res.* **44**, D73-  
374 80 (2016).
- 375 53. Stamatakis, A. RAxML version 8: a tool for phylogenetic analysis and post-analysis of large  
376 phylogenies. *Bioinformatics* **30**, 1312–1313 (2014).
- 377 54. Price, M. N., Dehal, P. S. & Arkin, A. P. FastTree 2 – approximately maximum-likelihood trees for  
378 large alignments. *PLoS One* **5**, e9490 (2010).
- 379 55. Lu, J. *et al.* Metagenome analysis using the Kraken software suite. *Nat. Protoc.* **17**, 2815–2839  
380 (2022).
- 381 56. Wood, D. E., Lu, J. & Langmead, B. Improved metagenomic analysis with Kraken 2. *Genome Biol.*  
382 **20**, 257 (2019).
- 383 57. Lu, J., Breitwieser, F. P., Thielen, P. & Salzberg, S. L. *Bracken: Estimating Species Abundance in*  
384 *Metagenomics Data*. 1–14 <http://biorxiv.org/lookup/doi/10.1101/051813> (2016)  
385 doi:10.1101/051813.
- 386 58. Broder, A. Z. On the resemblance and containment of documents. in *Proceedings. Compression and*  
387 *Complexity of SEQUENCES 1997 (Cat. No.97TB100171)* 21–29 (IEEE Comput. Soc, 1997).
- 388 59. Saitou, N. & Nei, M. The neighbor-joining method: a new method for reconstructing phylogenetic  
389 trees. *Mol. Biol. Evol.* **4**, 406–425 (1987).
- 390 60. Gascuel, O. Neighbor-Joining Revealed. *Mol. Biol. Evol.* **23**, 1997–2000 (2006).
- 391 61. Howe, K., Bateman, A. & Durbin, R. QuickTree: building huge Neighbour-Joining trees of protein  
392 sequences. *Bioinformatics* **18**, 1546–1547 (2002).
- 393 62. Fan, H., Ives, A. R., Surget-Groba, Y. & Cannon, C. H. An assembly and alignment-free method of  
394 phylogeny reconstruction from next-generation sequencing data. *BMC Genomics* **16**, 522 (2015).
- 395 63. Huerta-Cepas, J., Serra, F. & Bork, P. ETE 3: Reconstruction, analysis, and visualization of

- 396 phylogenomic data. *Mol. Biol. Evol.* **33**, 1635–1638 (2016).
- 397 64. Köster, J. & Rahmann, S. Snakemake—a scalable bioinformatics workflow engine. *Bioinformatics*  
398 **28**, 2520–2522 (2012).
- 399 65. Rahman, A. & Medvedev, P. Representation of k-Mer sets using spectrum-preserving string sets. *J.*  
400 *Comput. Biol.* **28**, 381–394 (2021).
- 401 66. Brinda, K. Novel computational techniques for mapping and classification of Next-Generation  
402 Sequencing data. (Université Paris-Est, 2016).
- 403 67. Li, H. *Seqtk: Toolkit for Processing Sequences in FASTA/Q Formats.* (2016).
- 404 68. Marçais, G. & Kingsford, C. A fast, lock-free approach for efficient parallel counting of occurrences  
405 of k-mers. *Bioinformatics* **27**, 764–770 (2011).
- 406 69. Grüning, B. *et al.* Bioconda: sustainable and comprehensive software distribution for the life  
407 sciences. *Nat. Methods* **15**, 475–476 (2018).
- 408 70. Grad, Y. H. *et al.* Genomic Epidemiology of Gonococcal Resistance to Extended-Spectrum  
409 Cephalosporins, Macrolides, and Fluoroquinolones in the United States, 2000–2013. *J. Infect. Dis.*  
410 **214**, 1579–1587 (2016).
- 411 71. Zerbino, D. R. & Birney, E. Velvet: algorithms for de novo short read assembly using de Bruijn  
412 graphs. *Genome Res.* **18**, 821–829 (2008).
- 413 72. Croucher, N. J. *et al.* Rapid phylogenetic analysis of large samples of recombinant bacterial whole  
414 genome sequences using Gubbins. *Nucleic Acids Res.* **43**, e15 (2015).
- 415 73. Dicks, J. *et al.* NCTC3000: a century of bacterial strain collecting leads to a rich genomic data  
416 resource. *Microb. Genom.* **9**, (2023).
- 417 74. Elbe, S. & Buckland-Merrett, G. Data, disease and diplomacy: GISAID’s innovative contribution to  
418 global health. *Glob Chall* **1**, 33–46 (2017).
- 419 75. roblanf & Mansfield, R. *Roblanf/Sarscov2phylo: 13-11-20.* (Zenodo, 2020).  
420 doi:10.5281/ZENODO.3958883.
- 421 76. Iqbal, Z., Caccamo, M., Turner, I., Flicek, P. & McVean, G. De novo assembly and genotyping of  
422 variants using colored de Bruijn graphs. *Nat. Genet.* **44**, 226–232 (2012).



- 423 77. Bradley, P. & Iqbal, Z. Supplementary Info for “Ultrafast search of all deposited bacterial and viral  
424 genomic data.” figshare <https://doi.org/10.6084/M9.FIGSHARE.5702776> (2017).
- 425 78. Blackwell, G. *et al.* Additional material for article “Exploring bacterial diversity via a curated and  
426 searchable snapshot of archived DNA sequences.” figshare  
427 <https://doi.org/10.6084/M9.FIGSHARE.16437939> (2021).
- 428 79. Grad, Y. Data for “Genomic Epidemiology of Gonococcal Resistance to Extended-Spectrum  
429 Cephalosporins, Macrolides, and Fluoroquinolones in the United States, 2000-2013.” Zenodo  
430 <https://doi.org/10.5281/ZENODO.2618836> (2019).
- 431 80. Tange, O. GNU Parallel: the command-line power tool. ;*login: The USENIX Magazine* **36**, 42–47  
432 (2011).
- 433 81. Brinda, K. NCTC 3000 complete assemblies. Zenodo <https://doi.org/10.5281/ZENODO.4838517>  
434 (2021).
- 435 82. Lanfear, R. *A Global Phylogeny of SARS-CoV-2 Sequences from GISAID.* (2020).  
436 doi:10.5281/zenodo.4089815.
- 437 83. Larsson, N. J. & Moffat, A. Off-line dictionary-based compression. *Proc. IEEE* **88**, 1722–1732  
438 (2000).
- 439 84. Wan, R. *Browsing and Searching Compressed Documents.* ( University of Melbourne, Melbourne,  
440 Australia, 2003).
- 441 85. Cock, P. J. A. *et al.* Biopython: freely available Python tools for computational molecular biology  
442 and bioinformatics. *Bioinformatics* **25**, 1422–1423 (2009).
- 443 86. Chikhi, R., Limasset, A. & Medvedev, P. Compacting de Bruijn graphs from sequencing data quickly  
444 and in low memory. *Bioinformatics* **32**, i201–i208 (2016).
- 445 87. Břinda, K., Sykulski, M. & Kucherov, G. Spaced seeds improve k-mer-based metagenomic  
446 classification. *Bioinformatics* **31**, 3584–3592 (2015).
- 447 88. Li, H. *et al.* The Sequence Alignment/Map format and SAMtools. *Bioinformatics* **25**, 2078–2079  
448 (2009).
- 449 89. Inouye, M. *et al.* SRST2: Rapid genomic surveillance for public health and hospital microbiology

- 450 labs. *Genome Med.* **6**, 90 (2014).
- 451 90. Nasko, D. J., Koren, S., Phillippy, A. M. & Treangen, T. J. RefSeq database growth influences the  
452 accuracy of k-mer-based lowest common ancestor species identification. *Genome Biol.* **19**, 165  
453 (2018).
- 454 91. Lees, J. A., Tonkin-Hill, G., Yang, Z. & Corander, J. Mandrake: visualizing microbial population  
455 structure by embedding millions of genomes into a low-dimensional representation. *Philos. Trans.  
456 R. Soc. Lond. B Biol. Sci.* **377**, 20210237 (2022).
- 457 92. Salson, M., Lecroq, T., Léonard, M. & Mouchard, L. A four-stage algorithm for updating a Burrows–  
458 Wheeler transform. *Theor. Comput. Sci.* **410**, 4350–4359 (2009).
- 459 93. Kempa, D. & Prezza, N. At the roots of dictionary compression: string attractors. in *Proceedings of  
460 the 50th Annual ACM SIGACT Symposium on Theory of Computing* (ACM, New York, NY, USA,  
461 2018). doi:10.1145/3188745.3188814.
- 462 94. Shallit, J. ALGEBRAIC COMBINATORICS ON WORDS (encyclopedia of mathematics and its  
463 applications 90) by M. Lothaire: 504 pp., £60.00, ISBN 0 521 81220 8 (Cambridge university press,  
464 2002). *Bull. Lond. Math. Soc.* **35**, 570–571 (2003).
- 465 95. Bonnie, J. K., Ahmed, O. & Langmead, B. DandD: efficient measurement of sequence growth and  
466 similarity. *iScience* 109054 (2024).
- 467 96. www-core (Web team). Public Health England reference collections - Wellcome Sanger Institute.  
468 <https://www.sanger.ac.uk/resources/downloads/bacteria/nctc/>.
- 469 97. Blackwell, G. A. *et al.* Exploring bacterial diversity via a curated and searchable snapshot of  
470 archived DNA sequences. (2021) doi:10.1101/2021.03.02.433662.
- 471 98. Mathog, D. R. Parallel BLAST on split databases. *Bioinformatics* **19**, 1865–1866 (2003).
- 472 99. McGinnis, S. & Madden, T. L. BLAST: at the core of a powerful and diverse set of sequence analysis  
473 tools. *Nucleic Acids Res.* **32**, W20–5 (2004).
- 474 100. Vijaya Satya, R. *et al.* In silico microarray probe design for diagnosis of multiple pathogens. *BMC  
475 Genomics* **9**, 496 (2008).
- 476 101. Vijaya Satya, R., Kumar, K., Zavaljevski, N. & Reifman, J. A high-throughput pipeline for the design

477 of real-time PCR signatures. *BMC Bioinformatics* **11**, 340 (2010).

478 102. Yates, J. F. *NCBI Nt (Nucleotide) Database FASTA File from 2017-10-26*. (2020).

479 doi:10.5281/zenodo.4382154.

480 103. Meyer, F. *et al.* Critical Assessment of Metagenome Interpretation: the second round of challenges.

481 *Nat. Methods* **19**, 429–440 (2022).

482 104. Oskolkov, N. Bowtie2 index for full NCBI NT database, January 2021.

483 [https://figshare.scilifelab.se/articles/online\\_resource/Bowtie2\\_index\\_for\\_full\\_NCBI\\_NT\\_database](https://figshare.scilifelab.se/articles/online_resource/Bowtie2_index_for_full_NCBI_NT_database_January_2021/21070063/1)

484 [se\\_January\\_2021/21070063/1](https://figshare.scilifelab.se/articles/online_resource/Bowtie2_index_for_full_NCBI_NT_database_January_2021/21070063/1) (2022).

## 485 **METHODS**

### 486 487 488 ***Analysis of the decrease in bacteria BLAST searchability***

489  
490 **Estimation of BLAST NT database size.** The size of the BLAST NT database for the time period  
491 between 2002-01-01 and 2022-11-01 was estimated using five types of online resources, resulting in  
492 n=27 values. First, file sizes were manually recorded from the official NCBI website  
493 <https://ftp.ncbi.nih.gov/blast/db/FASTA/> (n=11, between 2020-04-05 and 2022-11-01). Second,  
494 additional values were obtained from the snapshots of this website and its NCBI mirrors on  
495 <http://web.archive.org> (n=7, between 2012-10-11 and 2022-06-06). Third, archived versions of the NT  
496 database were found in diverse online repositories (n=3, between 2017-10-26 and 2021-01-15). Fourth,  
497 the NT database size was documented in a software documentation (n=1, 2013-12-03). Finally, the  
498 number of base pairs in the NT database was also documented in literature (n=5, between 2002-01-01  
499 and 2010-01-01) (**Supplementary Table 8**). Conversion between the sizes of the GZip-compressed NT  
500 database and the corresponding total sequence lengths was performed using the 2.04 GZip bits per bp  
501 constant, estimated using the NT database as of 2022-06-20.

502  
503 **Estimation of NCBI Assembly database size.** The number of bacteria in the NCBI Assembly  
504 database <sup>52</sup> (<https://www.ncbi.nlm.nih.gov/assembly/>) and their compressed size were estimated from  
505 the GenBank assembly summary file  
506 [https://ftp.ncbi.nlm.nih.gov/genomes/genbank/bacteria/assembly\\_summary.txt](https://ftp.ncbi.nlm.nih.gov/genomes/genbank/bacteria/assembly_summary.txt) (n=1,280,758 records,  
507 downloaded on 2022-11-02). The file was sorted according to the 'seq\_rel\_date' field and then used for  
508 calculating the number of published assemblies till a given date, aggregated per month. The total lengths  
509 of assemblies for the corresponding time points were estimated using the mean length of a bacterial  
510 genome assembly in the 661k collection (3.90 Mbp) and then converted to the estimated GZip size as  
511 previously. Although updates in the `assembly_summary.txt` file, such as the removal of old  
512 contaminated records, may influence the resulting statistics, a manual inspection during a several-  
513 months-long period showed only a minimal impact of these changes on the old statistics.

514  
515 **Comparison of BLAST NT and NCBI Assembly database sizes (Fig 1a).** To compare the sizes of  
516 two databases at the same time points, their respective functions were first interpolated in the  
517 logarithmic scale using piecewise linear functions from the data extracted above. The resulting  
518 interpolations were then used to calculate the estimated proportion of the sizes of NT and the bacteria in  
519 the NCBI Assembly database at regular intervals (monthly). Although minor inaccuracies might be  
520 present in the calculations (such as variations in the mean bacterial assembly or in the GZip-bits-per-bp

521 conversion across different versions of the databases), these differences do not impact the overall  
522 exponential decrease of data searchability.

523

524

### 525 ***Conceptual overview of phylogenetic compression***

526

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

529

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

539

540 For microbes, clustering can be accomplished rapidly by metagenomic classification <sup>36</sup> applied to the raw  
541 reads or other methods for species identification. Microbial genomes in public repositories form distinct  
542 clusters, usually (but not always) corresponding to individual species <sup>35</sup>, and metagenomic classification  
543 can assign individual genomes to these respective clusters, defined by the underlying reference database  
544 such as NCBI RefSeq <sup>36</sup>. This requires only a constant time per dataset and can be fully parallelized,  
545 resulting thus in a constant-time clustering if sufficiently many computational nodes are available.

546

547 The obtained clusters are then reorganized into batches. First, too small clusters are merged, creating a  
548 special pseudo-cluster called dustbin, whose purpose is to collect divergent, weakly compressible  
549 genomes from sparsely sampled regions of the tree of life. Subsequently, the clusters that are too large –  
550 such as those corresponding to oversampled human pathogens (e.g., *S. enterica* or *E. coli*) – as well as  
551 the dustbin are then divided into smaller batches, to provide guarantees on the maximum required  
552 downstream computational resources per one batch. An additional discussion of batching is provided in  
553 **Supplementary Note 3.**

554

555 **Step 2: Inference of a compressive phylogeny (Fig. 1b(ii)).** In this step, the computed batches  
556 are equipped with a so-called *compressive phylogeny*, which is a phylogeny approximating the true

557 underlying phylogenetic signal with sufficient resolution for compression purposes. If accurate inference  
558 methods such as RAxML<sup>53</sup> or FastTree 2<sup>54</sup> cannot be applied due to the associated bioinformatics  
559 complexity or high resource requirements, phylogenies can be rapidly estimated via lighter approaches  
560 such as the Mashtree algorithm<sup>34</sup> (reimplemented more efficiently in Attotree,  
561 <https://github.com/karel-brinda/attotree>) instead, with only a negligible impact on the resulting  
562 compression performance (**Supplementary Fig. 5, Supplementary Note 1**).

563

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

572

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

588

589

### 590 ***The MiniPhy framework for phylogenetic compression***

591

592 Here, we describe the specific design choices of our implementation of phylogenetic compression for

593 assemblies and de Bruijn graphs. More information and relevant links, including specific tools such as  
594 MiniPhy and Phylign and the resulting databases, can be found on the associated website  
595 (<https://brinda.eu/mof>).

596  
597 **Clustering/batching.** As genome collections encountered in practice can vary greatly in their  
598 properties as well as the available metadata, clustering is expected to be performed by the user. The  
599 recommended procedure is to identify species clusters using standard metagenomic approaches, such as  
600 those implemented in the Kraken software suite<sup>55</sup> (i.e., Kraken 2<sup>56</sup> and Bracken<sup>57</sup> applied on the  
601 original read sets), as the obtained abundance profiles can also be used for quality control to filter out  
602 those samples that are likely contaminated. The next step is to divide the obtained genome clusters into  
603 smaller batches, analogically to the examples in **Supplementary Figure 1** and as discussed in more  
604 details in **Supplementary Note 3** (and the corresponding implementation in the MiniPhy package,  
605 see below). The order in which genomes are taken within individual clusters can impact the final  
606 compression performance; based on our experience, lexicographic order with accessions or ordering  
607 according to the number of distinct  $k$ -mers per genome provide surprisingly good performance as both  
608 of these approaches tend to group phylogenetically close genomes closer to each other. The protocol can  
609 be customized further to suit the performance characteristics of algorithms downstream, such as by  
610 adjusting the batch size or the parameters controlling the creation of dustbin batches (**Supplementary**  
611 **Note 3**). If the total size of a collection is small enough, the clustering/batching step may be skipped  
612 entirely and the entire collection treated as a single batch.

613  
614 **Inference of a compressive phylogeny.** Users have the option to provide a custom tree generated  
615 by an accurate inference method such as RAxML<sup>53</sup>. However, in most practical scenarios, such trees are  
616 not available, and MiniPhy then employs Attotree (<https://github.com/karel-brinda/attotree>), an  
617 efficient reimplement of the Mashtree algorithm<sup>34</sup>, to generate a compressive phylogeny through  
618 sketching. Both Mashtree and Attotree first use Mash<sup>58</sup> to estimate the evolutionary distances between  
619 all pairs of genomes, which are then used to infer a compressive phylogeny employing the Neighbor-  
620 Joining algorithm<sup>59,60</sup> as implemented in QuickTree<sup>61</sup>. The distance computation in Mash is based on  
621 estimating the Jaccard indexes of the corresponding  $k$ -mer sets and then estimating the likely mutation  
622 rate under a simple evolutionary model<sup>62</sup>. Finally, MiniPhy post-processes the obtained tree using  
623 standard tree-transformation procedures implemented in the ETE3 library<sup>63</sup>, involving tree  
624 standardization, setting a midpoint outgroup, ladderization, and naming the internal nodes.

625  
626 **MiniPhy** (<https://github.com/karel-brinda/miniphy>). This is a central package for phylogenetic  
627 compression, including support for batching, and for calculating the associated statistics (see below).  
628 MiniPhy is implemented as a Snakemake<sup>64</sup> pipeline, offering three protocols for phylogenetic

629 compression:

- 630 1) Compression of assemblies based on left-to-right reordering.
- 631 2) Compression of de Bruijn graphs represented by simplitigs<sup>33,65</sup> based on left-to-right reordering.
- 632 3) Compression of de Bruijn graphs through bottom-up  $k$ -mer propagation using ProPhyle<sup>28,29</sup>.

633

634 In the third protocol,  $k$ -mer propagation is executed recursively in a bottom-up manner: at each internal  
635 node, the  $k$ -mer sets of the child nodes are loaded, their intersection computed, stored at the node, the  
636 intersection subtracted from the child nodes, and all three  $k$ -mer sets saved in the form of simplitigs<sup>33,65</sup>;  
637 ProphAsm<sup>33</sup> performs all these operations. This process results in a progressive reduction of the  $k$ -mer  
638 content within the phylogeny in a lossless manner. Further details on this technique can be found in  
639 ref<sup>66</sup>.

640

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

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

644 For assemblies, these text files are the original assembly FASTA files, converted by SeqTK<sup>67</sup> to the  
645 single-line format with all nucleotides in uppercase ('seqtk seq -U {input.fa}'). For simplitigs, the  
646 text files are EOL-delimited lists of simplitigs in the order as computed by ProphAsm, obtained from its  
647 output using the command 'seqtk seq {input.fa} | grep -v \>'. The resulting TAR file is then  
648 compressed using XZ ('xz -9 -T1', see the section about calibration), and the resulting .tar.xz file  
649 distributed to users or further recompressed or indexed by other low-level tools, while preserving the  
650 underlying order.

651

652 **MiniPhy statistics.** For each of the three implemented protocols, MiniPhy generates a comprehensive  
653 set of statistics to quantify the compressibility of the batch, including: 1) *set* (the size of the  $k$ -mer set  
654 computed from all nodes of the compressive phylogeny), 2) *multiset* (the size of the  $k$ -mer multiset  
655 computed as a union of  $k$ -mer sets from individual nodes), 3) *sum\_ns* (the total number of sequences),  
656 4) *sum\_cl* (the total sequence length), 5) *recs* (the number of records corresponding to individual  
657 nodes), and 6) *xz\_size* (the size of the TAR file after XZ compression). The sizes of  $k$ -mer sets and  
658 multisets are determined from  $k$ -mer histograms computed by JellyFish 2<sup>68</sup> (v2.2.10) using the  
659 commands:

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

662 followed by

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



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

666

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

$$674 \quad \text{removed\_redundancy} = \frac{|\text{multiset\_preprop}| - |\text{multiset\_postprop}|}{|\text{multiset\_preprop}| - |\text{set}|}$$

675 For collections comprising multiple batches, these variables refer to the global statistics, i.e., the sizes of  
676 set and multiset unions across all batches.

677

678 **MiniPhy-COBS.** MiniPhy-COBS (<https://github.com/leoisl/miniphy-cobs>) is a Snakemake <sup>64</sup> pipeline  
679 designed to create phylogenetically compressed ClaBS COBS indexes <sup>37</sup> (Classical Bit-sliced index) from  
680 assemblies already phylogenetically compressed by MiniPhy. ClaBS is a variant of COBS analogous to  
681 the original BIGSI data structure <sup>17</sup>, using Bloom filters of the same size; this property is important for  
682 ensuring that the order of Bloom filters is preserved and that the neighboring Bloom filters are mutually  
683 compressible (**Supplementary Note 2**). The workflow for each batch involves three main steps:

684

- 685 1) Renaming input assemblies to align their lexicographic and phylogenetic orders within each batch,
- 686 2) Constructing COBS ClaBS indexes with:

```
687     cobs classic-construct -T 8 {batch} {output}.cobs_classic
```

- 688 3) Compressing the obtained indexes using:

```
689     xz -9 -T1 -e --lzma2=preset=9,dict=1500MiB,nice=250
```

690

691 **Updated ProPhyle.** To simplify the integration with MiniPhy for bottom-up  $k$ -mer propagation, a new  
692 version of ProPhyle <sup>28,29</sup> was released (v0.3.3.1, <https://github.com/prophyle/prophyle>). The main  
693 improvement compared to previous versions includes the possibility to stop after  $k$ -mer propagation,  
694 without proceeding to the construction of an FM-index, as such an index is unnecessary for phylogenetic  
695 compression using MiniPhy. The new version of ProPhyle is provided in the form of a Github release  
696 (<https://github.com/iqbal-lab-org/cobs/releases>) and pre-built packages on Bioconda <sup>69</sup>.

697

698

699 **Overview of the five test microbial collections**

700

701 **GISP.** The GISP collection comprises 1,102 draft assemblies of *N. gonorrhoeae* clinical isolates,  
702 collected in the US between 2000 and 2013 by the Centers for Disease Control and Prevention as part of  
703 the Gonococcal Isolate Surveillance Project (GISP) <sup>70</sup>. These isolates had been sequenced using Illumina  
704 HiSeq and assembled using Velvet <sup>71</sup>. The phylogenetic relationships among the isolates are known and  
705 had been determined using RAxML <sup>53</sup> after a correction for recombination by Gubbins <sup>72</sup>. The GISP  
706 collection provides an example of a high-quality collection of draft genomes of a single low-diversity  
707 bacterial species, generated using a standardized sequencing and assembly protocol.

708

709 **NCTC3k.** The NCTC3k collection comprises 1,065 draft and complete assemblies of isolates of various  
710 bacterial species, derived from strains in the National Collection of Type Cultures (NCTC) collection and  
711 analyzed by Public Health England, the Wellcome Sanger Institute, and Pacific Biosciences as part of the  
712 NCTC 3000 Project <sup>73</sup> (<https://www.culturecollections.org.uk/collections/nctc-3000-project.aspx>). The  
713 isolates were sequenced using the PacBio Single Molecule, Real-Time (SMRT) DNA Sequencing  
714 technology, and assembled using automated pipelines. The assembled genomes are publicly available  
715 from the <https://www.sanger.ac.uk/resources/downloads/bacteria/nctc/> website. The NCTC3k  
716 collection provides an example of a collection of high-quality, nearly complete genomes from diverse  
717 bacterial species.

718

719 **SC2.** The SC2 collection comprises 590,779 complete assemblies of SARS-CoV-2 isolates obtained from  
720 the GISAID database <sup>74</sup> as of 2021-05-18. These isolates were collected, sequenced, and assembled by  
721 various laboratories worldwide between 2020 and 2021 using various protocols. The phylogeny of the  
722 isolates is known and was computed by the sarscov2phylo software  
723 (<https://github.com/roblanf/sarscov2phylo/>, ref <sup>75</sup>). The SC2 collection provides an example of a large  
724 collection of genomes of varying quality obtained from epidemiological surveillance of a single viral  
725 species at a global scale.

726

727 **BIGSIdata.** The BIGSIdata collection comprises 425,160 cleaned de Bruijn graphs representing nearly  
728 all bacterial and viral isolates available in the European Nucleotide Archive (ENA) as of December  
729 2016 <sup>17</sup>. These isolates had originally been collected and sequenced by various laboratories worldwide,  
730 deposited as raw-read data or genome assemblies to repositories synchronized with the ENA (ENA,  
731 NCBI SRA, and DDBJ Sequence Read Archive), and later downloaded and transformed into cleaned de  
732 Bruijn graphs using McCortex <sup>40,76</sup> (k=31) by the European Bioinformatics Institute (EBI). The resulting  
733 graphs were provided on an HTTP/FTP website  
734 ([http://ftp.ebi.ac.uk/pub/software/bigsi/nat\\_biotech\\_2018](http://ftp.ebi.ac.uk/pub/software/bigsi/nat_biotech_2018)), along with metadata on Figshare <sup>77</sup>,  
735 although not all of the original 447,833 graphs could be retrieved in this study (see below). The

736 BIGSIdata collection provides an example of a large and diverse collection of bacterial and virus isolates,  
737 collected and sequenced across the globe using various sequencing technologies and all provided in a  
738 unified graph representation.

739  
740 **661k.** The 661k collection comprises 661,405 draft assemblies of all Illumina-sequenced bacterial  
741 isolates present in the ENA as of November 2018 <sup>16</sup>. These isolates had originally been collected and  
742 sequenced by various laboratories worldwide, and their raw-read data deposited to repositories  
743 synchronized with the ENA (ENA, NCBI SRA, and DDBJ Sequence Read Archive). The assemblies were  
744 generated using a single unified pipeline (<https://github.com/iqbal-lab-org/assemble-all-ena>) based on  
745 Shovill (<https://github.com/tseemann/shovill>) by EBI, and provided on an HTTP/FTP website  
746 (<https://ftp.ebi.ac.uk/pub/databases/ENA2018-bacteria-661k/>), along with metadata on FigShare <sup>78</sup>.  
747 The 661k collection provides an example of a large and diverse collection of assembled bacterial isolates,  
748 collected and sequenced across the globe using a single sequencing technology, i.e., the state-of-the-art  
749 of the short read-assembly era.

750  
751 Basic characteristics of the five test collections, including the original file size, number of samples,  
752 species count, and the number of distinct *k*-mers, are provided in **Supplementary Table 1**.

753  
754

### 755 ***Acquisition of the test collections***

756

757 **GISP.** The GISP collection was obtained from the [https://github.com/c2-d2/rase-db-ngonorrhoeae-](https://github.com/c2-d2/rase-db-ngonorrhoeae-gisp)  
758 [gisp](https://github.com/c2-d2/rase-db-ngonorrhoeae-gisp) repository (version 04a132c) as published in ref <sup>51</sup>. The assemblies (n=1,102) were obtained from  
759 the “isolates/contigs” subdirectory of Github repository (containing the original genomes including the  
760 plasmids), and the associated RAXML phylogenetic tree was downloaded from the “tree/” subdirectory  
761 of the same repository. The original data had originally been analyzed in ref <sup>70</sup> and provided for  
762 download on Zenodo <sup>79</sup>.

763

764 **NCTC3k.** The assemblies were obtained in the GFF format from

765 <ftp://ftp.sanger.ac.uk/pub/project/pathogens/NCTC3000> by

766 `wget -m -np -nH --cut-dirs 3 -retr-symlinks \`

767 `ftp://ftp.sanger.ac.uk/pub/project/pathogens/NCTC3000 .`

768 The obtained files were converted them to the FASTA format by any2fasta

769 (<https://github.com/tseemann/any2fasta>, v0.4.2) parallelized by GNU Parallel <sup>80</sup> and uploaded to

770 Zenodo (ref <sup>81</sup>, <http://doi.org/10.5281/zenodo.4838517>). The number of species in the collection was

771 determined based on the data provided in the main Sanger/Public Health England assembly table for

772 NCTC 3000 (<https://www.sanger.ac.uk/resources/downloads/bacteria/nctc/>, retrieved on 2022-09-  
773 14). The HTML table was manually exported to XLSX and used to construct a translation table from  
774 NCTC accession numbers to corresponding species. The accessions of the assemblies in our collection  
775 were then extracted from file names, translated to species, and the species counted. Overall, this resulted  
776 in n=1,065 assemblies of 259 species.

777

778 **SC2.** The SARS-Cov-2 data were downloaded from the GISAID website (<https://www.gisaid.org/>, as of  
779 2021-05-18) in the form of an assembly file ('sequences\_fasta\_2021\_05\_18.tar.xz', n=1,593,858)  
780 and a Sarscov2phylo phylogeny<sup>82</sup> ('gisaid-hcov-19-phylogeny-2021-05-11.zip', n=590,952). After  
781 converting both files to the same set of identifiers and removing isolates with missing data, we obtained  
782 n=590,779 genome assemblies organized in a phylogenetic tree.

783

784 **BIGSIdata.** The BIGSI collection data<sup>17</sup> were downloaded from the associated FTP  
785 ([http://ftp.ebi.ac.uk/pub/software/bigsi/nat\\_biotech\\_2018/](http://ftp.ebi.ac.uk/pub/software/bigsi/nat_biotech_2018/)), including cleaned de Bruijn graphs,  
786 taxonomic information inferred using Kraken<sup>26</sup>, and abundance reports computed using Bracken<sup>57</sup>. The  
787 download was done using RSync in groups corresponding to individual EBI prefixes (e.g., DRR000) by

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

791 The prefixes were organized into 15 groups of at most 100 prefixes each, and the groups were processed  
792 individually in succession on a research computing cluster, with a parallelization using Slurm and jobs  
793 deployed using Snakemake<sup>64</sup> (between 2020-08-01 and 2020-09-15). From the downloaded McCortex  
794 files, unitigs were extracted using McCortex:

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

796 Only those graphs with an uncorrupted McCortex file, Bracken information available, unitigs of total  
797 length  $\geq 2$  kbp with  $\leq 15$  M distinct  $k$ -mers, and with no file system error encountered were used in the  
798 subsequent processing. This resulted in n=425,161 de Bruijn graphs (out of the original n=463,331  
799 genomes from the FTP or n=447,833 genomes reported in ref<sup>17</sup>).

800

801 **661k.** The 661k collection was downloaded in March 2021 from the official FTP repository specified in  
802 ref<sup>16</sup>, using RSync by

```
803     rsync -avP rsync://ftp.ebi.ac.uk/pub/databases/ENA2018-bacteria-  
804     661k/Assemblies/{pref}
```

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

807

808

809 ***Calibration and evaluation of phylogenetic compression***

810

811 **Calibration of XZ as a low-level tool for phylogenetic compression (Supplementary Fig.**

812 **3)**. The compression performance of GZip, BZip2, and XZ was evaluated using the GISP collection,  
813 converted to the single-line FASTA format and with genomes sorted left-to-right according to the  
814 Mashtree phylogeny. For each compressor, the compression was performed with a range of presets and  
815 always with a single thread. To evaluate the compression performance with large resources available,  
816 two additional manually tuned modes with larger dictionaries, denoted by ‘M’ and ‘MM’, were added to  
817 the XZ benchmark, corresponding to the parameters

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

819 and

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

821 respectively.

822

823 To evaluate the impact of different line lengths on the compression, the source FASTA was reformatted  
824 for different lengths using SeqTK<sup>67</sup> and compressed using XZ by

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

826

827 **Comparison of scaling modes (Supplementary Fig. 4)**. The SC2 collection was provided in the

828 left-to-right order according to Sarscov2phylo phylogeny. The genomes were progressively uniformly  
829 subsampled, stored as EOL-separated lists of sequences (without sequence headers), and then  
830 compressed using individual compressors, namely: 1) XZ: ‘xz -9 -T1’, 2) BZip2: ‘bzip2 --best’,  
831 3) GZip: ‘gzip -9’, and 4) Re-Pair<sup>83,84</sup> (<https://github.com/rwanwork/Re-Pair>, version as of 2021-10-  
832 26):

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

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

838

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

845  
846 **Order comparison (Supplementary Fig. 5).** The SC2 collection was put into three different  
847 orderings: the original ordering (corresponding to the lexicographical ordering by sequence names), the  
848 left-to-right ordering of the phylogeny, and a randomized order. In all cases, a custom Python script  
849 using BioPython<sup>85</sup> was used to order the FASTA file and remove sequence names, and its output was  
850 compressed by the XZ compressor using 1 thread and the best preset ('xz -T1 -9'). The comparisons  
851 for GISP and NCTC3k was performed analogically, but with sequence headers preserved.

852  
853 **Summary of MiniPhy calibration.** XZ with the parameters 'xz -9 -T1' was chosen as the default  
854 compression procedure for MiniPhy, and Mashtree<sup>34</sup> as the default method for inferring compressive  
855 phylogenies. These choices were done based on the observations that the most popular method, GZip,  
856 always performed poorly for bacteria, although provided a moderate compression performance for  
857 viruses. On the other hand, XZ achieved steep compression curves for low-diversity collections, with  
858 compression ratio improving by one order per one order increase of the number of genomes, for both  
859 viruses and bacteria. NCTC3k as a high-diversity collection was weakly compressible even with the best  
860 approaches (<1 order of magnitude of compression after a 3 orders-of-magnitude increase of the  
861 number of genomes). One of the best available (but still highly experimental) grammar-based  
862 compressors, Re-Pair<sup>83,84</sup>, achieved a similar asymptotic behavior as XZ, indicative of the potential of  
863 grammar compressors for phylogenetic compression to provide random access, but its usability remains  
864 experimental. Phylogenetic reordering boosted compression substantially for both low- and high-  
865 diversity collections (reduction in size between 38% and 67% compared to random orders). Finally,  
866 compressive phylogenies computed using Mashtree<sup>34</sup> provided nearly equal compression performance  
867 as an accurate approach using RAxML<sup>53</sup>.

868  
869

### 870 ***Phylogenetic compression of the BIGSI data collection of de Bruijn graphs***

871

872 **Clustering and batching.** For every sample, the output of Kraken<sup>26</sup> and Bracken<sup>57</sup> were extracted  
873 from the downloaded data as provided in the online FTP repository  
874 ([http://ftp.ebi.ac.uk/pub/software/bigsi/nat\\_biotech\\_2018/ctx/](http://ftp.ebi.ac.uk/pub/software/bigsi/nat_biotech_2018/ctx/)) in the Bracken files  
875 ('{accession}.ctx\_bracken.report') as the previously identified most prevalent species  
876 (corresponding to the row with the highest value of the 'fraction\_total\_reads' column). Clustering  
877 and batching then proceeded as depicted in **Supplementary Fig. 1** and further commented in  
878 **Supplementary Note 3**, with genomes being sorted according to the number of *k*-mers before their  
879 partitioning into batches. Overall, the genomes of the 1,443 identified species (clusters) were partitioned  
880 into 568 regular batches and 6 dustbin batches, resulting in a total of 574 batches.

881

882 **Phylogenetic compression.** Phylogenetic compression was performed twice, with slightly different  
883 workflows.

884

885 First, phylogenetic compression proceeded manually, via a workflow whose modified version was later  
886 implemented in MiniPhy. For individual batches, compressive phylogenies were computed using  
887 MashTree<sup>34</sup> with the default parameters. The resulting trees and McCortex unitig files were then used as  
888 input for ProPhyle (v0.3.3.0) to propagate *k*-mers along the phylogenies, compute simplitigs<sup>33</sup>, and  
889 merge the output FASTA files into a single one by

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

891 The resulting FASTA files produced by ProPhyle (called 'index.fa') were converted into the single-line  
892 format using SeqTK<sup>67</sup> and compressed using XZ by

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

894 The resulting files occupied 74.4 GB and were deposited on <https://doi.org/10.5281/zenodo.4086456>  
895 and <https://doi.org/10.5281/zenodo.4087330>. Support for this version of the data set was incorporated  
896 into De-MiniPhy-BIGSIdata (see below), and the correctness of the end-to-end protocol and of the  
897 resulting files was validated by De-MiniPhy-BIGSIdata and subsequent *k*-mer counting using kc-c3  
898 (<https://github.com/lh3/kmer-cnt>, commit 'e257471'). The obtained *k*-mer counts were compared to  
899 those obtained from the original McCortex files (from the total length and count of unitigs); all *k*-mer  
900 counts were equal with the exception of 4 samples with 17–26 more reported *k*-mers after  
901 decompression.

902

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

907

908 **Decompression of BIGSIdata de Bruijn graphs.** To decompress de Bruijn graphs from the files  
909 obtained by *k*-mer propagation, all *k*-mers along all root-to-leaf paths need to be collected. We  
910 implemented this specifically for BIGSIdata in a Python package called De-MiniPhy-BIGSIdata  
911 (<https://github.com/karel-brinda/De-MiniPhy-BIGSIdata>). The program downloads individual data  
912 files from Zenodo from the accessions above (the first version of the dataset) and reconstructs the  
913 original *k*-mer sets using the following procedure. First, it decompresses the XZ file of a given batch,  
914 splits it according to files corresponding to individual nodes of the compressive phylogeny, recompresses  
915 individual nodes using GZip parallelized by GNU Parallel<sup>80</sup>, and for all leaves (genomes) it reconstructs  
916 the corresponding *k*-mer sets by merging all GZip files along the corresponding root-to-leaf paths using

917 the Unix cat command. From the obtained output FASTA files, de Bruijn graphs can be easily  
918 reconstructed by standard tools such as BCALM2 <sup>86</sup>.

919

920 **Comparison to the original compression protocol.** As the samples in our BIGSIdata collection  
921 do not fully correspond to the data that were used in the original publication of BIGSI <sup>17</sup>, we recalculated  
922 the size statistics of the published McCortex files of our graphs based on the FTP list-off files as provided  
923 within individual subdirectories of [http://ftp.ebi.ac.uk/pub/software/bigsi/nat\\_biotech\\_2018/](http://ftp.ebi.ac.uk/pub/software/bigsi/nat_biotech_2018/) (as of  
924 2021-08-27). These were downloaded per individual prefix directories recursively using wget by

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

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

931

932 **Comparison to Metagraph** <sup>38</sup>. The size of the phylogenetically compressed BIGSIdata collection was  
933 compared to the size of an analogous Metagraph index from the original paper <sup>38</sup>, based on the statistics  
934 in Table 1 and Supplementary Table 1 therein (the SRA-Microbe collection): n=446,506 indexed  
935 datasets, 39.5 G canonical *k*-mers (with the same *k*-mer size *k*=31), and the size of the annotated de  
936 Bruijn graph being 291 GB (graph 30 GB + annotations 261 GB). This index was constructed from the  
937 same datasets as in the original BIGSI paper <sup>17</sup> but using a slightly different computational methodology.  
938 Consequently, the index of Metagraph contained approximately 4% fewer distinct canonical *k*-mers  
939 (*k*=31) compared to BIGSIdata as used in this paper. To compare the two compression approaches  
940 (MiniPhy with bottom-up *k*-mer propagation and XZ as a low-level tool vs. Metagraph), both applied to  
941 the similar but different input data, we used the number of bits per distinct *k*-mer as the statistic for  
942 comparison, which was found to be 10.2 and 58.9, respectively. Therefore, the MiniPhy compression was  
943 more efficient by an estimated factor of 5.78. We note that phylogenetic compression could be directly  
944 embedded into Metagraph (by imposing the phylogenetic order of columns during index construction),  
945 which may help to further reduce its index size.

946

947

#### 948 ***Phylogenetic compression of the 661k assembly collection***

949

950 **Clustering and batching.** Species clusters were identified based on the most prevalent species in the  
951 sample as identified using Kraken 2 <sup>56</sup> and Bracken <sup>57</sup> from the original raw-read data; i.e., based on the  
952 'V2' column in the 'File1\_full\_krakenbracken.txt' file of the supplementary materials of ref <sup>16</sup>. The



953 creation of the dustbin pseudo-cluster and formation of individual batches proceeded by the steps  
954 documented in **Supplementary Fig. 1** and as later implemented directly within MiniPhy, with  
955 genomes pre-sorted lexicographically according to ENA accessions.

956

957 **Phylogenetic compression using MiniPhy.** The obtained batches were compressed using the  
958 MiniPhy pipeline as described above; i.e., compressive phylogenies were computed using Mashtree<sup>34</sup>  
959 and used for 1) left-to-right reordering of the assemblies, 2) left-to-right reordering of simplitigs of the  
960 corresponding de Bruijn graphs, and 3) bottom-up *k*-mer propagation and simplitig computation by  
961 ProPhyle; while in all cases storing the simplitigs and assemblies as text and FASTA file, respectively,  
962 followed by a compression by ‘xz -9 -T1’. The compressed assemblies were deposited on  
963 <https://doi.org/10.5281/zenodo.4602622>.

964

965 **Calculations of the statistics.** All the statistics used in the plots and tables were calculated based on  
966 the numbers obtained from MiniPhy. Additionally, the total number of *k*-mers was calculated using  
967 JellyFish<sup>68</sup> (v2.2.10) by

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

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

976

977 **Recompression using MBGC.** Individual phylogenetically compressed batches from the previous  
978 step were converted to single FASTA files by ‘tar -xOvf {input.xz}’ and then compressed using  
979 MBGC<sup>18</sup> (v1.2.1) with 8 threads and the maximum compression level by

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

981

982 **Compression in the lexicographic order.** Data in ENA and other similar repositories have  
983 identifiers assigned in the order in which they are uploaded, individual uploads typically proceed by  
984 uploading entire projects, and these typically involve phylogenetically very close genomes. For instance,  
985 genomes from a study investigating a hospital outbreak often occupy a range of accessions. Therefore,  
986 lexicographically sorted genomes from ENA may be used as an approximation of phylogenetic  
987 compression. To compare the compressibility of the 661k collection in the ENA accession lexicographic  
988 order to the full phylogenetic compression, we streamed the genomes from the main collection file

989 provided on [http://ftp.ebi.ac.uk/pub/databases/ENA2018-bacteria-661k/661\\_assemblies.tar](http://ftp.ebi.ac.uk/pub/databases/ENA2018-bacteria-661k/661_assemblies.tar),  
990 decompressed them on-the-fly, converted them to the one-line FASTA format using SeqTK<sup>67</sup>, and  
991 compressed them using XZ with 32 threads by

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

993

994

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

996

997 **The 661k-HQ collection.** To reduce biases in *k*-mer matching, a high-quality variant of the 661k  
998 collection, called 661k-HQ, was constructed from the 661k collection by excluding genomes that had not  
999 passed quality control in the original study<sup>16</sup> (3.7% of the genomes). For simplicity, the batches and  
1000 genome orders in 661k-HQ were kept the same as in 661k.

1001

1002 **Phylogenetic compression of the 661k/661k-HQ COBS indexes.** COBS indexes for the 661k and  
1003 661k-HQ collection were constructed per batch using the MiniPhy-COBS pipeline (see the MiniPhy-  
1004 COBS section), which produces the ClaBS variant of the index with all Bloom filters of the same size  
1005 sorted left-to-right according to the phylogeny, and compresses them using XZ.

1006

1007 **Comparisons to the compact COBS indexes.** The compact variant of the COBS index (default in  
1008 COBS), based on adaptive adjustments of Bloom filter sizes through subindexes of different heights, was  
1009 used as a baseline in our comparisons. For the 661k collection, we used the original index as provided  
1010 ([http://ftp.ebi.ac.uk/pub/databases/ENA2018-bacteria-661k/661k.cobs\\_compact](http://ftp.ebi.ac.uk/pub/databases/ENA2018-bacteria-661k/661k.cobs_compact), retrieved on 2022-  
1011 09-08, 937 GB). For building a COBS index for 661k-HQ, we used the same construction protocol as in  
1012 ref<sup>16</sup>. Both indexes were then compressed on a highly performant server by XZ using 32 cores ('xz -9 -  
1013 T32').

1014

1015 All of the obtained data points are provided in **Supplementary Table 5**.

1016

1017

### 1018 ***Phylign pipeline for alignment against all pre-2019 bacteria from ENA***

1019

1020 **Overview.** The Phylign pipeline (<https://github.com/karel-brinda/phylign>) uses phylogenetically  
1021 compressed assemblies (661k) and COBS indexes (661k-HQ) as described above to align queries against  
1022 the entire 661k-HQ collection in a fashion similar to BLAST (**Supplementary Note 4**). The search  
1023 procedure consists of two phases: matching the queries against the *k*-mer indexes using COBS<sup>37</sup> to  
1024 identify the database's most similar genomes for each query, followed by an alignment of the queries to

1025 their best-matching genomes using Minimap 2 <sup>41</sup>. Phylign is developed as a Snakemake <sup>64</sup> pipeline, using  
1026 Bioconda <sup>69</sup> for an automatic software management and the standard Snakemake resource  
1027 management <sup>64</sup> to control the CPU cores assignments and limit RAM usage according to user-specified  
1028 parameters. Upon its first execution, Phylign downloads its phylogenetically compressed reference  
1029 database from the Internet (102 GB), consisting of 29.2 GB of assemblies and 72.8 GB of COBS indexes.

1030  
1031 **Matching.** The matching step involves *k*-mer matching of all user queries against the entire 661k-HQ  
1032 database using a modified version of COBS (v0.3, see below), based on the principle that the number of  
1033 *k*-mer matches between a genome and a query correlates with the alignment score <sup>87</sup>. Each  
1034 phylogenetically compressed COBS index is decompressed in memory and queried for the input user  
1035 sequences, reporting all matches between the queries and genomes in the current batch with a sufficient  
1036 (user-specified) proportion of matching *k*-mers. The computed matches are then aggregated across all  
1037 batches, and for each query, only a (user-specified) number of best matches, plus ties, are retained and  
1038 passed to the subsequent alignment step. Matching is parallelized by Snakemake, with the number of  
1039 threads for each COBS instance adjusted based on batch size.

1040  
1041 **Alignment.** For each batch independently and fully in parallel, Phylign then iterates over the  
1042 phylogenetically compressed genome assemblies, and if a given genome has at least one match passed  
1043 from the matching phase, it builds on-the-fly, in memory, a new Minimap 2 <sup>41</sup> (v2.24) instance for this  
1044 genome and aligns all relevant queries to this genome, while saving Minimap 2 outputs in a batch-  
1045 specific output file. Once all batches are processed, the resulting alignments are aggregated and provided  
1046 to the user in a modified SAM format <sup>88</sup>.

1047  
1048 **Performance characteristics.** The total matching time is primarily driven by the time complexity of  
1049 COBS, with decompression accounting for less than 2 CPU hours (**Supplementary Fig. 9**). In the  
1050 alignment step, decompression requires less than 1.5 CPU hours (**Supplementary Fig. 9**), and the  
1051 remainder of the time is primarily driven by the time to create a new Minimap2 instance (estimated 0.3  
1052 CPU seconds per instance in the current implementation). If the queries are long and Minimap 2 is used  
1053 with a sensitive preset, the actual Minimap 2 alignment time becomes the main time component (e.g., in  
1054 the plasmid experiment in **Supplementary Tab. 6**).

1055  
1056 **Updated COBS.** To integrate COBS into Phylign, new versions of COBS <sup>37</sup> were created (v0.2, v0.3,  
1057 <https://github.com/iqbal-lab-org/cobs>). The updates include support for macOS, streaming of indexes  
1058 into memory, and multiple bug fixes. The new versions of COBS are provided in the form of Github  
1059 releases (<https://github.com/iqbal-lab-org/cobs/releases>) and pre-built packages on Bioconda <sup>69</sup>.

1060

1061 **Benchmarking of the decompression time.** Decompression times were evaluated on the same  
1062 desktop computer as the alignment experiments, separately for the phylogenetically compressed  
1063 assemblies vs. COBS indexes and for in-memory decompression (`xzcat {file} > /dev/null`) vs. on-  
1064 disk decompression (`xzcat {file} > {tmpfile}`), resulting in four experiments. Within each  
1065 experiment, decompression was parallelized using GNU Parallel (`parallel -L1 -v -progress`), with  
1066 time measured using GNU time both for the whole experiment and for each batch in a given compressed  
1067 representation.

1068

1069

## 1070 *Evaluating Phylign*

1071

1072 **Overview of the benchmarking procedure.** The search using Phylign was evaluated on three  
1073 datasets, representative of different query scenarios: a database of antibiotic resistance genes, a database  
1074 of plasmids, and an Oxford nanopore sequencing experiment. In all cases, the search parameters –  
1075 including the number of hits of interest, the COBS  $k$ -mer threshold, and the Minimap preset – were  
1076 tailored to each specific query type. The experiments were conducted on an iMac with a Quad-Core Intel  
1077 CPU i7, 4.2 GHz with 4 physical (8 logical) cores and 42.9 GB (40 GiB) RAM.

1078

1079 **Time measurements.** The wall clock and CPU time were measured using GNU time and calculated as  
1080 real and `usr+sys`, respectively. The measurements were done for the matching and alignment steps  
1081 separately.

1082

1083 **Memory measurements.** We have not found any reliable way of measuring peak memory  
1084 consumption on macOS: both GNU time and the `psutil` Python library were significantly  
1085 underestimating the memory footprint of our Snakemake pipeline. Therefore, we performed additional  
1086 measurements on a Linux cluster using the SLURM job manager, using jobs allocated with a  
1087 configuration similar to the parameters of our iMac computer. For `max_ram_gb` set to 30 GB, we  
1088 observed a peak memory consumption of 26.2 GB, thus by 12.7% lower compared to the specified  
1089 maximum. Such a discrepancy is expected because the `max_ram_gb` parameter defines an upper bound  
1090 for the Snakemake resource management <sup>64</sup>, representing the worst-case scenario for parallel job  
1091 combinations.

1092

1093 **Resistance genes – ARGannot.** The resistance genes search was performed using the ARG-ANNOT  
1094 database <sup>42</sup> comprising 1,856 genes/alleles, as distributed within the SRST2 software toolkit <sup>89</sup>  
1095 ([https://github.com/katholt/srst2/blob/master/data/ARGannot\\_r3.fasta](https://github.com/katholt/srst2/blob/master/data/ARGannot_r3.fasta), retrieved on 2022-07-24).  
1096 The search parameters were set to require a minimum of 50% matching  $k$ -mers, with 1,000 best hits

1097 plus ties taken for every gene/allele query. Alignment was performed with the Minimap preset for short  
1098 reads ('sr').

1099

1100 **Plasmids – the EBI plasmid database.** The list of EBI plasmid was downloaded from the associated  
1101 EBI website (<https://www.ebi.ac.uk/genomes/plasmid.details.txt>, retrieved on 2022-04-03), and  
1102 individual plasmids were subsequently downloaded from the ENA using curl and GNU parallel<sup>80</sup>. The  
1103 search parameters were set to require at least 40% matching *k*-mers (the threshold previously used in  
1104 ref<sup>17</sup>), with 1,000 best hits plus ties taken for every plasmid. Alignment was performed with the  
1105 Minimap preset for long, highly divergent sequences ('asm20').

1106

1107 **Oxford Nanopore reads.** The ERR9030361 experiment, comprising 159k nanopore reads from an  
1108 isolate of *M. tuberculosis*, was downloaded from SRA NCBI. The search parameters were set to require  
1109 at least 40% matching *k*-mers, with 10 best hits plus ties taken for every read. Alignment was performed  
1110 with the Minimap preset for nanopore reads ('map-ont').

1111

1112 **Comparison to BIGSI.** As we were unable to reproduce the original plasmid search experiment<sup>17</sup> with  
1113 BIGSI on our iMac computer (due to the required database transfer of 1.43 TB over an unstable FTP  
1114 connection), we used the values provided in the original publication<sup>17</sup>. To ensure a fair comparison, we  
1115 focused on evaluating the total CPU time (sys+usr) and verified that our parallelization efficiency was  
1116 close to the maximal one (680% out of 800% possible achieved, based on the values in **Supplementary**  
1117 **Table 7**).

1118

#### 1119 **DATA AVAILABILITY STATEMENT**

1120 All data supporting the findings of this study and the developed software are available within the paper  
1121 and its Supplementary Materials (**Supplementary Table 4**).

1122

#### 1123 **ACKNOWLEDGEMENTS**

1124

1125 This work was partially supported by the NIGMS of the National Institutes of Health (R35GM133700),  
1126 the David and Lucile Packard Foundation, the Pew Charitable Trusts, and the Alfred P. Sloan  
1127 Foundation. R.C. was partially supported by the European Union's Horizon 2020 research and  
1128 innovation programme (grants agreements No. 872539, 956229, and 101047160) and ANR Transipedia,  
1129 SeqDigger, Inception, and PRAIRIE grants (ANR-18-CE45-0020, ANR-19-CE45-0008, PIA/ANR16-  
1130 CONV-0005, ANR-19-P3IA-0001). Portions of this research were conducted on the O2 high-  
1131 performance compute cluster, supported by the Research Computing Group at Harvard Medical School,  
1132 and on the GenOuest bioinformatics core facility (<https://www.genouest.org>).

## 1133 **Supplementary notes**

1134

### 1135 **Supplementary Note 1. Stability of phylogenetic compression**

1136

1137 The overall performance of phylogenetic compression stems from a combination of trade-offs between  
1138 the individual layers of a given phylogenetic compression protocol (such as for assemblies, de Bruijn  
1139 graphs, or  $k$ -mer indexes). These layers include the specific clustering and batching strategy,  
1140 compressive phylogeny inference, and the low-level compression/indexing technique.

1141

1142 **Clustering.** Clustering can be performed using various direct or indirect methods. All these methods  
1143 expected to identify similar clusters thanks to the pronounced species structure across public microbial  
1144 isolate dataset <sup>35</sup>. However, both classes of approaches have specific caveats that may downgrade the  
1145 resulting compression performance.

1146

1147 *Caveats of indirect approaches:* When clustering is based on species identification by Kraken or other  
1148 LCA-based classifiers, clustering might be impacted by the loss of resolution due to reference database  
1149 growth <sup>90</sup>. While this is unlikely to significantly affect phylogenetic compression performance with  
1150 collections akin to 661k (where phylogenetically related genomes would still be clustered together,  
1151 although under biological incorrect species names); a carefully analysis of the data structure will be  
1152 necessary for atypical collections, such as those comprised of metagenome-assembled genomes.

1153

1154 *Caveats of direct approaches:* Direct clustering methods, now feasible at the scale of millions  
1155 genomes <sup>91</sup>, are contamination-oblivious and may thus be sensitive to various contamination patters  
1156 (see, e.g., the discussion of *C. difficile* in ref <sup>91</sup>). Contamination is very common in public genomic  
1157 datasets, and if not properly controlled by metagenomic profiling or other quality control techniques, it  
1158 can impede both downstream compression and search.

1159

1160 **Batching.** For 661k and BIGSI data, batching has been implemented heuristically, with lexicographic  
1161 preordering based on accessions, to ensure that genomes sequenced around the same time would, within  
1162 the same species cluster, be batches together. An alternative pre-sorting strategy, based on the number  
1163 of  $k$ -mers in a given dataset, was tested for BIGSIdata (data not shown), and led to mostly comparable  
1164 results.

1165

1166 **Compressive phylogeny.** In most scenarios, compressive phylogeny is used for within-batch  
1167 reordering of either assemblies directly or of columns corresponding to individual genomes in case of  $k$ -

1168 mer indexes. When clustering and batching are done correctly and a robust low-level compressor used  
1169 (e.g., XZ), such reordering by itself provides a moderate improvement (30–55% reduction, see  
1170 **Supplementary Fig. 5**). Nevertheless, the impact is much stronger with less advanced compression  
1171 techniques; for instance, run-length encoding (RLE) applied to  $k$ -mer matrices improves by up to an  
1172 order of magnitude when the columns are reordered according to phylogenies (data not shown). When  
1173 considering different approaches to compute phylogenies, even sketching combined with neighbor  
1174 joining provides a sufficient resolution; Mashtree yields nearly as good compression results as full-scale  
1175 methods for phylogenetic inference, such as RAxML<sup>53</sup>. Differences in the resulting compression ratios  
1176 are relatively minor, with RAxML phylogenies showing a slight advantage in Lempel-Ziv-based  
1177 compression on assemblies over Mash trees (**Supplementary Table 3**), and conversely, Mash trees  
1178 exhibit slightly better performance in compressing de Bruijn graphs or  $k$ -mer sets (**Supplementary**  
1179 **Table 2, 3**).

1180

1181 **Low-level compressor or indexer.** The final performance of phylogenetic compression is  
1182 significantly influenced by the capabilities of the used low-level compressor or indexer. For dictionary  
1183 compressors, an essential parameter is the dictionary size or the window size (**Supplementary**  
1184 **Fig. 3a, Supplementary Fig. 4**), which disqualifies many popular compressors, including gzip and  
1185 bzip2. For general compressors applied to assemblies, a crucial factor is converting FASTA to the one-  
1186 line format (**Supplementary Fig. 3b**). There are also notable differences in compression speed:  
1187 compressing a single batch of assemblies using XZ might require up to several hours (albeit with rapid  
1188 decompression), while MBGC (v2.0) requires approximately ten minutes per batch.

1189

## 1190 **Supplementary Note 2. Genome-similarity-preserving representations in phylogenetic** 1191 **compression**

1192

1193 As a prerequisite for phylogenetic compression, it is fundamental to assume that the core genome  
1194 representations preserve similarity. Informally, this means that little changes in the input genome lead  
1195 to only little changes in its representation, ensuring that closely phylogenetically related genomes have  
1196 highly mutually compressible representations. Although the similarity-preserving property can be  
1197 rigorously defined in specific cases using mathematical formalism including specific input and output  
1198 distances and embeddings, we adopt a more conceptual perspective to maintain a broader view.

1199

1200 **Examples of genome-similarity-preserving representations:**

- 1201 • **Complete genomes assemblies.** Complete genome assemblies precisely reflect the sequence  
1202 of nucleotides in DNA molecules, with single mutation events resulting in single changes in the  
1203 assembly.
- 1204 • **Draft genome assemblies.** Similar to complete assemblies, but may not fully resolve  
1205 repetitive regions, leading to a fragmented assembly. In contrast to complete assemblies, a single  
1206 evolutionary event might induce a more substantial change in the representation. For example, a  
1207 mutation in a previously non-resolvable repetitive region could make it resolvable by turning an  
1208 exact repeat into an inexact one. Nevertheless, such events are rather infrequent, and for many  
1209 compression techniques (e.g., those based on Lempel-Ziv), the distance between the two  
1210 representations remains minimal.
- 1211 • **Burrows-Wheeler Transform of assemblies.** The Burrows-Wheeler transform <sup>21</sup> is  
1212 characterized by its locality, in the sense that a localized change in the input induces only a  
1213 localized change in the BWT-transformed string <sup>92</sup>.
- 1214 •  **$k$ -mer spectrum.** Changing, deleting, or inserting one nucleotide in the genome alters the  $k$ -  
1215 mer spectrum by the removal and addition of up to  $2k+2$   $k$ -mers.
- 1216 • **MinHash sketches.** The addition or removal of a  $k$ -mer to or from a spectrum may lead to the  
1217 replacement of one hash value by a smaller or larger one, respectively, and such a replacement  
1218 happens only with a very low probability. Therefore, sketches of similar genomes are either  
1219 identical or very similar.
- 1220 • **Minimizer de Bruijn graphs.** These combine properties of de Bruijn graphs and minimizers,  
1221 with the genome-similarity-preserving property following naturally from this combination.
- 1222 • **Bloom filters of fixed size.** The addition or removal of element to or from a set always alters  
1223 the Bloom filter by a maximum of  $m$  bits, where  $m$  is the number of hash functions; therefore, a  
1224 small change in the genome results only in a small change in the corresponding fixed-size Bloom  
1225 filter.

1226  
1227 **Examples of representations that are not similarity-preserving:**

- 1228 • **Bloom filters with adaptive sizes.** Adaptive size adjustment (such as implemented in  
1229 COBS's default strategy <sup>37</sup>, which uses smaller Bloom filters for smaller genomes), disrupts  
1230 similarity preservation. For instance, an event such as an acquisition of a plasmid by an *E. coli*  
1231 strain may cause the Bloom filter to expand, reflecting an increase in genome and  $k$ -mer set size,  
1232 altering also the underlying hash functions (or the associated modulo function). In consequence,  
1233 adaptive-size Bloom filters of even closely related genomes can be very dissimilar. As a result, we  
1234 did not use the COBS default strategy, but forced it to use the same size of Bloom filters for all  
1235 genomes in a given batch.



#### 1237 **Supplementary Note 4. Core principles of the MiniPhy batching approach**

1238

1239 The batching approach used in this paper, as summarized for the 661k and BIGSIdata collections in  
1240 **Supplementary Fig. 1**, is based on the following principles. At its core, phylogenetic compression  
1241 involves the phylogenetic reordering of input data. For large collections, this process entails partitioning  
1242 genomes into batches that adhere to specific constraints on certain characteristics, and then reordering  
1243 them phylogenetically based on compressive phylogenies.

1244

1245 To ensure the essential guarantees from the paper, and to maximize the batches' usability across  
1246 combinations of tools and in diverse application use cases, the batches are required to have following  
1247 properties:

- 1248 1) **An upper-bounded compressed size** – to guarantee easy internet transmission, even over  
1249 unreliable networks.
- 1250 2) **A lower-bounded compressed size** – to limit the negative effects of excessively unbalanced  
1251 batches in workflow managers such as Snakemake and Nextflow and in resource allocation  
1252 systems such as Slurm.
- 1253 3) **An upper-bounded decompressed size** – to minimize the maximum memory required per  
1254 batch in downstream data analysis and to facilitate the parallel processing of multiple batches in  
1255 memory-constraint environments.
- 1256 4) **An upper-bounded number of genomes per batch** – to establish a limit on the time  
1257 required per batch for phylogenetic inference and for downstream data analyses.
- 1258 5) **Optimization for maximal compression ratio within these constraints** – to minimize  
1259 the overall necessary data transmission over the Internet and within a computer (e.g., from disk  
1260 to RAM).

1261

1262 On a mathematical level, these constraints lead to interesting optimization problems that may be  
1263 formalized and solved by techniques such as integer linear programming or answer set programming, in  
1264 combinations with techniques for estimating data compressibility via measures such as the size of  
1265 minimal string attractors<sup>93</sup>, factor complexity<sup>94</sup>, or the  $\delta$  measure<sup>95</sup>.

1266

1267 However, for simplicity, our approach used in MiniPhy is empirical, informed by the following  
1268 observations about bacterial genomes and the structure of ENA:

- 1269 1) **Constrained genome size range.** For bacteria, their genome size can be assumed to fit within  
1270 a range of one order of magnitude, typically 1 Mbp to 10 Mbp (see the principles behind  
1271 BIGSI<sup>17</sup>).

- 1272 2) **Relatedness within bioprojects.** In public repositories such as ENA, sequencing data are  
1273 usually uploaded per individual projects, and ENA accession ranges often contain highly  
1274 phylogenetically related genomes.
- 1275 3) **Species clusters.** Individual microbial species form clusters in public repositories such as  
1276 ENA <sup>35</sup>.
- 1277 4) **Sampling biases.** Public repositories exhibit prevalent sampling biases, enabling a rough  
1278 classification of bacterial species into two categories: highly sampled and sparsely sampled (see,  
1279 e.g., Fig. 1 in ref <sup>16</sup>).

1280  
1281 Altogether, this understanding led to the following general heuristic for batching genomes in  
1282 comprehensive genome corpuses:

- 1283 1) Cluster genomes based on their species. Specifically, identify the species of each genome, and  
1284 then treat all genomes belonging to the same species as individual clusters.
- 1285 2) Within each cluster, arrange genomes in the lexicographic order of their accessions, to maximize  
1286 the chance that highly related genomes, sequenced at the same time, stay in the same batch in  
1287 the subsequent steps,
- 1288 3) Iterate over individual species clusters and compare their size with a predefined threshold  
1289 ('batch-min-size' in MiniPhy):
- 1290 a.  $\text{size} \geq \text{threshold}$ : Classify the species as highly sampled and proceed according to Step 5.
  - 1291 b.  $\text{size} < \text{threshold}$ : Classify the species as sparsely sampled and proceed according to Step 4
- 1292 4) Merge all sparsely sampled species clusters into a single pseudo-cluster called a dustbin,  
1293 proceeding in the order of lexicographically sorted species names (while preserving the order of  
1294 genomes within each cluster).
- 1295 5) Split the dustbin pseudo-cluster into batches of a predefined size ('dustbin-batch-max-size'  
1296 in MiniPhy).
- 1297 6) Split each highly sampled species cluster into batches of a predefined size ('batch-max-size' in  
1298 MiniPhy).

1299  
1300 The calibration of this heuristic was performed empirically, in the environment of the Harvard O2  
1301 cluster, with the paratemers adjusted based on observed performance. In particular, if the  
1302 Mashtree/Attotree inference <sup>34</sup> or XZ compression of any batch exceeded a predefined time limit, the  
1303 batch-max-size or dustbin-batch-max-size parameters were modified accordingly.

1304  
1305 The resulting heuristic, including the default parameters, is provided in the MiniPhy repository in the  
1306 'create\_batches.py' script. The heuristic is also summarized, including the specific parameters used  
1307 for 611k and the BIGSIdata, in **Supplementary Fig. 1**.

1308

1309 **Supplementary Note 4. Comparison of the Phylign and BLAST approaches**

1310

1311 As tools for alignment against very large genome databases, Phylign and BLAST share many similarities,  
1312 but at the same time, they differ in several key aspects. First, while Phylign is tailored specifically for  
1313 bacterial genomes, BLAST is typically used with databases that encompass more types of sequences,  
1314 including genes, transcripts, and genomes of non-bacterial organisms. Second, both tools produce  
1315 alignments and compute alignment scores; however, while BLAST, computing local alignments,  
1316 complements the score with an E-value to quantify the expected number of alignments of similar quality  
1317 occurring by chance, Phylign targets longer alignments (primarily semiglobal, but can be adjusted by  
1318 modifying Minimap parameters) and does not include E-values. Third, while both tools compute  
1319 alignments using heuristic approaches, BLAST uses a seed-and-extend procedure, applied at the level of  
1320 the entire database, whereas Phylign initially pre-filters target genomes using *k*-mer-based methods and  
1321 then applies Minimap's seed-chain-align procedure<sup>41</sup> at the level of individual reference genomes. In  
1322 summary, Phylign and BLAST are designed for partially overlapping use cases, but they use different  
1323 computational strategies.

1324

1325

## Supplementary Tables

1326

### Supplementary Table 1: Five test collections used for the calibration and evaluation of phylogenetic compression.

1327

Characteristics of the genome collection used for calibrating and evaluating phylogenetic compression throughout the paper. Within-genome

1328

*k*-mer duplicates refer to the proportion of *k*-mer occurrences (*k*=31, canonical *k*-mers) that disappear when transforming genome assemblies

1329

to their corresponding de Bruijn graphs; the fact that this proportion is always low for microbial genomes, even for complete assemblies,

1330

suggests that de Bruijn graphs are a faithful representation of microbial genomes and the *k*-mer content can be used for quantifying data

1331

redundancy.

Collection	Description			Size		Diversity			Characteristics			
	Original samples	Genome representation	Data source	Nb. of genomes	Total sequence length	Nb. of species	Nb. of distinct <i>k</i> -mers	Within-genome <i>k</i> -mer duplicates	Unified construction pipeline	Data quality	Data volume	Repetitiveness
GISP	<i>N. gonorrhoeae</i> isolates	Draft assemblies <sup>70,79</sup>	.tar.gz file (726 MB) <a href="https://doi.org/10.5281/zenodo.2618836">https://doi.org/10.5281/zenodo.2618836</a>	1,102	2.36 Gbp	1	4.18 M	2.02%	Yes	Very high	Low	Very high
NCTC3k	Bacterial isolates	Complete and draft assemblies <sup>96</sup>	.gff files (6.48 GB) <a href="ftp://ftp.sanger.ac.uk/pub/project/pathogens/NCTC3000">ftp://ftp.sanger.ac.uk/pub/project/pathogens/NCTC3000</a>  Converted to FASTA and uploaded to: <a href="https://doi.org/10.5281/zenodo.4838517">https://doi.org/10.5281/zenodo.4838517</a> .fa.gz files (1.25 GB)	1,065	4.35 Gbp	259	992 M	2.80%	Partially	High	Low	Medium
SC2	SARS-CoV-2 isolates	Complete assemblies <sup>74</sup>	.xz file (201 MB) <a href="http://gisaid.org">http://gisaid.org</a>	590,779	17.6 Gbp	1	1.85 M	0.000700%	No	Low	High	Very high
661k	Bacterial isolates	Draft assemblies <sup>97</sup>	.fa.gz files (805 GB) <a href="http://ftp.ebi.ac.uk/pub/databases/ENA2/018-bacteria-661k">http://ftp.ebi.ac.uk/pub/databases/ENA2/018-bacteria-661k</a>	661,405	2.58 Tbp	2,336 (est., ref <sup>97</sup> )	44.3 G	0.846%	Yes	Medium	Very high	High
BIGSIdata	Bacterial and viral isolates	de Bruijn graphs <sup>17</sup>	McCortex files (16.7 TB) <a href="http://ftp.ebi.ac.uk/pub/software/bigsi/nat_biotech_2018/ctx">http://ftp.ebi.ac.uk/pub/software/bigsi/nat_biotech_2018/ctx</a>	425,160	1.68 Tbp <sup>a</sup>	1,443 (est., ref <sup>17</sup> )	41.1 G	-	Yes	Low	Very high	High

1332

**Footnotes:**

1333

<sup>a</sup> Computed as the total length of unitigs of the individual de Bruijn graphs.

1334 **Supplementary Table 2: Proportion of redundancy explained by compressive phylogenies in the five test collections.**

1335 The amount of reduction of genomic  $k$ -mer content ( $k=31$ , canonical  $k$ -mers) via  $k$ -mer propagation along compressive phylogenies.  $k$ -mer  
 1336 multisets correspond to the unions of  $k$ -mer sets before and after  $k$ -mer propagation, reduction factor is the ratio of their sizes, and removed  
 1337 redundancy quantifies the proportion of removed  $k$ -mers among the removable ones (100% if each  $k$ -mers was entirely associated with a  
 1338 single subtree). In the case of BIGSIdata and 661k, a phylogeny was built for each batch independently.

1339

Collection	Compressive phylogeny	$k$ -mer multiset size		Reduction statistics	
		Before reduction	After reduction	Reduction factor	Removed redundancy
GISP	Mashtree	2.31 G	63.3 M	36.5	97.4%
	RAxML	2.31 G	72.6 M	31.8	97.0%
NCTC3k	Mashtree	4.23 G	1.79 G	2.36	75.3%
SC2 <sup>a</sup>	GISAID Sarscov2phylo	1.49 G	32.8 M	45.3	97.8%
BIGSIdata	Mashtree (1 tree/batch, 574 batches)	1.39 T	212. G	6.58	87.4%
661k	Mashtree (1 tree/batch, 305 batches)	2.55 T	233. G	11.0	92.5%

1340 **Footnotes:**

1341 <sup>a</sup> In order to use ProPhyle with SC2, the collection was subsampled to 50k genomes, which corresponds to 8.47% of the original genome count. The original  $k$ -mer multiset  
 1342 size was 17.5 G.

1343 **Supplementary Table 3: Results of phylogenetic compression.**

1344 Size of the resulting files, mean space per single genome, bits per single base pair in the data, bits per distinct canonical  $k$ -mer ( $k=31$ ),  
 1345 The three baselines include a FASTA-like baseline computed as 8 bits per single character (i.e., FASTA without sequence headers and EOLs), a  
 1346 GZip-like baseline (2 bits per bp), and the file size with original compression protocol.

1347

Collection	Phylogenetic compression protocol			Compression statistics				Improvement over baselines		
	Reordering scheme	Compressive phylogeny	Low-level compressor	Compressed size	Kilobytes per genome	Bits per bp	Bits per distinct $k$ -mer	FASTA-like baseline (8 bits per bp)	GZip-like baseline (2 bits per bp)	Original compression protocol
GISP	Left-to-right	Mashtree	XZ	5.67 MB	5.15	0.0192	10.9	416.×	104.×	-
	Left-to-right	RAxML	XZ	5.44 MB	4.94	0.0184	10.4	434.×	109.×	-
NCTC3k	Left-to-right	Mashtree	XZ	257 MB	242	0.473	2.07	16.9×	4.23×	-
SC2	Left-to-right	Sarscov2phylo	XZ	10.7 MB	0.0181	0.00486	46.2	1,647×	412× <sup>a</sup>	-
BIGSIdata	Batches & $k$ -mer propagation	Mashtree (1 tree/batch, 574 batches)	XZ	52.3 GB	123.	0.248	10.2	32.2×	8.06×	319×
661k	Batches & left-to-right	Mashtree (1 tree/batch, 305 batches)	XZ	29.0 GB	43.8	0.0898	5.23	89.1×	22.3×	27.8×
			MBGC	20.7 GB	31.3	0.0642	3.74	125×	31.2×	38.9×
	Lexicographically by ENA accessions <sup>b</sup>	-	XZ	121 GB	182.	0.374	21.8	21.4×	5.35×	6.67×
			MBGC <sup>c</sup>	-	-	-	-	-	-	-

1348 **Footnotes:**

1349 <sup>a</sup> Due to viral genomes being short, GZip can outperform the 2-bits-per-bp entropy bound that was previously determined from bacteria, and the real improvement of XZ over  
 1350 GZip for phylogenetic compression is 13.2× in this case (see also **Supplementary Fig. 4**). The table displays the “GZip-like” value for consistency with the rest of the table.

1351 <sup>b</sup> Dataset accessions in ENA are partially phylogenetically ordered since sequencing studies often involve phylogenetically related genomes that are uploaded in succession.

1352 <sup>c</sup> Computation was systematically failing due to Out-of-Memory events, even for jobs with 200 GB RAM of allocated memory.

1353 **Supplementary Table 4: Software and data provided for download.**

1354 The table lists the developed software for phylogenetic compression and provides links to all phylogenetically compressed versions of the test  
 1355 collections (with the exceptions of SC2 that could not be published due to the licensing restrictions of GISAID).

1356

Type	Name	Description	URL
Software	Phylign	Snakemake pipeline	<a href="https://github.com/karel-brinda/phylign">https://github.com/karel-brinda/phylign</a>
	MiniPhy	Snakemake pipeline	<a href="https://github.com/karel-brinda/miniphy">https://github.com/karel-brinda/miniphy</a>
	MiniPhy-COBS	Snakemake pipeline	<a href="https://github.com/leoisl/miniphy-cobs">https://github.com/leoisl/miniphy-cobs</a>
	De-MiniPhy-BIGSIdata	Client program to download and decompress de Bruijn graphs from the BIGSIdata collection	<a href="https://github.com/karel-brinda/de-miniphy-bigsidata">https://github.com/karel-brinda/de-miniphy-bigsidata</a>
	ProPhyle (modified, vo.3.3)	ProPhyle metagenomic classifier	<a href="https://github.com/prophyle/prophyle">https://github.com/prophyle/prophyle</a>
	COBS (modified, vo.3)	COBS <i>k</i> -mer indexer	<a href="https://github.com/iqbal-lab-org/cobs">https://github.com/iqbal-lab-org/cobs</a>
	Attotree	A fast reimplementaion of Mashtree functionality	<a href="https://github.com/karel-brinda/attotree">https://github.com/karel-brinda/attotree</a>
Phylogenetically compressed genome collections	NCTC3k	Assemblies (XZ)	<a href="https://doi.org/10.5281/zenodo.5533354">https://doi.org/10.5281/zenodo.5533354</a>
	BIGSIdata	De Bruijn graphs (simplitigs after <i>k</i> -mer propagation; XZ)	<a href="https://doi.org/10.5281/zenodo.5555253">https://doi.org/10.5281/zenodo.5555253</a>
	661k	Assemblies (XZ)	<a href="https://doi.org/10.5281/zenodo.4602622">https://doi.org/10.5281/zenodo.4602622</a>
		Assemblies (MBGC)	<a href="https://doi.org/10.5281/zenodo.6347064">https://doi.org/10.5281/zenodo.6347064</a>
		<i>k</i> -mer index (COBS; XZ)	<a href="https://doi.org/10.5281/zenodo.7313926">https://doi.org/10.5281/zenodo.7313926</a> <a href="https://doi.org/10.5281/zenodo.7313942">https://doi.org/10.5281/zenodo.7313942</a> <a href="https://doi.org/10.5281/zenodo.7315499">https://doi.org/10.5281/zenodo.7315499</a>
	661k-HQ	<i>k</i> -mer index (COBS; XZ)	<a href="https://doi.org/10.5281/zenodo.6849657">https://doi.org/10.5281/zenodo.6849657</a> <a href="https://doi.org/10.5281/zenodo.6845083">https://doi.org/10.5281/zenodo.6845083</a>

1357

1358 **Supplementary Table 5: Compressibility of different variants of BIGSI indexes for the 661k/661k-HQ collections.**

1359

Collection	Protocol				Compression statistics		
	Index variant	Mode of construction	Low-level compression	Comment	Size	Total improvement over baseline	Improvement over uncompressed
661k	COBS-compact <sup>a</sup>	Per entire collection	None <sup>b</sup>	Baseline (with adaptive Bloom filters)	937. GB	1.00×	1.00×
			XZ <sup>c</sup>	Direct compression	243. GB	3.86×	3.86×
	COBS-classic <sup>d</sup>	Per MiniPhy batch; columns sorted left-to-right	None	Reordered data, Bloom filter size fixed per batch	2.46 TB	0.380×	1.00×
			XZ	Phylogenetic compression	<b>110. GB</b>	<b>8.51×</b>	22.5×
661k-HQ	COBS-compact <sup>a</sup>	Per entire collection	None	Baseline (with adaptive Bloom filters)	893. GB	1.00×	1.00×
			XZ <sup>c</sup>	Direct compression	205. GB	4.35×	4.35×
	COBS-classic <sup>d</sup>	Per MiniPhy batch; columns sorted left-to-right	None	Reordered data, Bloom filter size fixed per batch	1.06 TB	0.842×	1.00×
			XZ	Phylogenetic compression	<b>72.8 GB</b>	<b>12.3×</b>	14.5×

1360

1361

1362

1363

1364

1365

1366

1367

1368

**Footnotes:**

<sup>a</sup> As the COBS-compact classifies datasets to be indexed into bins based on the number of *k*-mers, it is at the same time also grouping phylogenetically related genomes across the whole database into the same bins. The resulting file is thus moderately compressible using XZ, even though the resulting archive is not suitable for downstream applications because of its size and the associated overheads.

<sup>b</sup> Provided for download on <http://ftp.ebi.ac.uk/pub/databases/ENA2018-bacteria-661k/>.

<sup>c</sup> Streamed compression of COBS subindexes that are internally created by COBS based on the number of *k*-mers in individual datasets.

<sup>d</sup> Besides better compressibility by general compressors, COBS-classic brings an additional benefit of decreasing the associated false positive error rate for a majority of the datasets under indexing.



1369 **Supplementary Table 6: Disk space requirements of Phylign with the 661k-HQ collection.**

1370 The requirements correspond to the version of the database as used by Phylign in **Supplementary Table 6**.

1371

Component	Size requirements [GB]	Kilobytes per genome <sup>a</sup>	Bits per bp <sup>a</sup>
Assemblies <sup>b</sup>	29.2	45.6	0.0942
COBS	72.8	114.	0.235
<b>Total</b>	<b>102.</b>	<b>159.</b>	<b>0.329</b>

1372 **Footnotes:**

1373 <sup>a</sup> The statistics are computed with respect to the characteristics of the 661-HQ collection.

1374 <sup>b</sup> An older version of compressed assemblies is used in Phylign for consistency across experiments. This part of the index is, however, further compressible: first, the files were  
1375 generated by an older version of MiniPhy, without tree rebalancing, therefore its size is higher compared to the latest version in **Supplementary Table 3**; second, the  
1376 archives contain even low-quality genomes, which could be omitted here.

1377 **Supplementary Table 7: Results of BLAST-like search across the 661k-HQ collection on a desktop computer using Phylign.**

1378 Timing and alignment results for resistance genes, EBI plasmids, and a nanopore sequencing experiment using Phylign, performed on an iMac  
 1379 with eight 4.2 GHz cores and 42.9 GB RAM (Methods). Search parameters were adjusted for the corresponding type of search based on the  
 1380 typical values in literature (Methods). All measurements were done with in-memory decompression ('index\_load\_mode' set to 'mem-stream')  
 1381 and maximal memory consumption set to 30 GB (the 'max\_ram\_gb' parameter). The resulting peak memory consumption was estimated to  
 1382 be 26.2 GB (Methods).

Experiment				Computational time (real   cpu time)			Alignment statistics				
Query dataset	Search parameters	No. of queries	Cumul. length	Matching	Alignment	Total	No. of aligned queries	No. of aligned segments	No. of distinct (genome, query) pairs	No. of target genomes	Nb. of target batches
<b>ARGannot resistance genes</b>	cobs_kmer_thres: 0.5 nb_best_hits: 1000 minimap_preset: sr	1,856	1.65 Mbp	0.417h   2.01h	3.45h   24.9h	3.87h   26.9h	1,713	1,801,997	1,734,405	272,198	286
<b>EBI plasmid database</b>	cobs_kmer_thres: 0.4 nb_best_hits: 1000 minimap_preset: asm20	2,826	224 Mbp	6.61h   43.6h	4.26h   30.5h	10.9h   74.1h	1,871	8,980,429	838,830	205,231	296
<b>Nanopore sequencing experiment (ERR9030361)</b>	cobs_kmer_thres: 0.4 nb_best_hits: 10 minimap_preset: map-ont	158,583	191 Mbp	3.07h   18.1h	1.22h   7.97h	4.29h   26.1h	146,691	4,548,919	3,841,621	47,162	85

1383

1384 **Supplementary Table 8: Reconstructed history of the BLAST NT database.**

1385 The information about size of the BLAST nucleotide database (nt.gz) was retrieved from literature, its associated webpages, and other public  
 1386 repositories.

Date	GZip size [GiB]	Length [Gbp]	Source <sup>a</sup>
2002-01-01	-	7.372	<a href="https://doi.org/10.1093/bioinformatics/btg250">https://doi.org/10.1093/bioinformatics/btg250</a> (ref <sup>98</sup> )
2003-02-01	-	8.33	<a href="https://doi.org/10.1093/nar/gkh435">https://doi.org/10.1093/nar/gkh435</a> (ref <sup>99</sup> )
2004-02-01	-	10	<a href="https://doi.org/10.1093/nar/gkh435">https://doi.org/10.1093/nar/gkh435</a> (ref <sup>99</sup> )
2007-07-01	-	21	<a href="https://doi.org/10.1186/1471-2164-9-496">https://doi.org/10.1186/1471-2164-9-496</a> (ref <sup>100</sup> )
2010-01-01	-	30	<a href="https://doi.org/10.1186/1471-2105-11-340">https://doi.org/10.1186/1471-2105-11-340</a> (ref <sup>101</sup> )
2012-10-11	10.7	-	<a href="https://web.archive.org/web/20121011234515/http://ftp.ncbi.nih.gov/blast/db/FASTA">https://web.archive.org/web/20121011234515/http://ftp.ncbi.nih.gov/blast/db/FASTA</a>
2013-12-03	13.8	-	<a href="https://web.archive.org/web/20201005113118/https://github.com/PathoScope/PathoScope/wiki/Building-Library">https://web.archive.org/web/20201005113118/https://github.com/PathoScope/PathoScope/wiki/Building-Library</a>
2017-10-26	39.4	-	<a href="https://doi.org/10.5281/zenodo.4382154">https://doi.org/10.5281/zenodo.4382154</a> (ref <sup>102</sup> )
2019-01-03	47	-	<a href="https://openstack.cebitec.uni-bielefeld.de:8080/swift/v1/CAMI_2_DATABASES/ncbi_blast/nt.gz">https://openstack.cebitec.uni-bielefeld.de:8080/swift/v1/CAMI_2_DATABASES/ncbi_blast/nt.gz</a> (a part of ref <sup>103</sup> )
2020-04-05	67	-	<a href="https://ftp.ncbi.nih.gov/blast/db/FASTA/">https://ftp.ncbi.nih.gov/blast/db/FASTA/</a>
2020-07-05	72	-	<a href="https://ftp.ncbi.nih.gov/blast/db/FASTA/">https://ftp.ncbi.nih.gov/blast/db/FASTA/</a>
2020-08-04	77	-	<a href="https://ftp.ncbi.nih.gov/blast/db/FASTA/">https://ftp.ncbi.nih.gov/blast/db/FASTA/</a>
2020-10-11	81	-	<a href="https://ftp.ncbi.nih.gov/blast/db/FASTA/">https://ftp.ncbi.nih.gov/blast/db/FASTA/</a>
2021-01-15	90	-	<a href="https://doi.org/10.17044/scilifelab.21070063.v1">https://doi.org/10.17044/scilifelab.21070063.v1</a> (ref <sup>104</sup> )
2021-03-21	103	-	<a href="https://web.archive.org/web/20210322230129/https://ftp-trace.ncbi.nih.gov/blast/db/FASTA/">https://web.archive.org/web/20210322230129/https://ftp-trace.ncbi.nih.gov/blast/db/FASTA/</a>
2021-03-28	104	-	<a href="https://web.archive.org/web/20210402195739/https://ftp.ncbi.nih.gov/blast/db/FASTA/">https://web.archive.org/web/20210402195739/https://ftp.ncbi.nih.gov/blast/db/FASTA/</a>
2021-10-18	138	-	<a href="https://web.archive.org/web/20211020093701/ftp://ftp.ncbi.nih.gov/blast/db/FASTA/">https://web.archive.org/web/20211020093701/ftp://ftp.ncbi.nih.gov/blast/db/FASTA/</a>
2021-11-01	139	-	<a href="https://ftp.ncbi.nih.gov/blast/db/FASTA/">https://ftp.ncbi.nih.gov/blast/db/FASTA/</a>
2021-12-13	146	-	<a href="https://ftp.ncbi.nih.gov/blast/db/FASTA/">https://ftp.ncbi.nih.gov/blast/db/FASTA/</a>
2022-01-18	152	-	<a href="https://ftp.ncbi.nih.gov/blast/db/FASTA/">https://ftp.ncbi.nih.gov/blast/db/FASTA/</a>
2022-02-28	161	-	<a href="https://web.archive.org/web/20220307133636/https://ftp-trace.ncbi.nih.gov/blast/db/FASTA/">https://web.archive.org/web/20220307133636/https://ftp-trace.ncbi.nih.gov/blast/db/FASTA/</a>
2022-03-21	166	-	<a href="https://web.archive.org/web/20220326071216/https://ftp-trace.ncbi.nih.gov/blast/db/FASTA/">https://web.archive.org/web/20220326071216/https://ftp-trace.ncbi.nih.gov/blast/db/FASTA/</a>
2022-06-06	180	-	<a href="https://web.archive.org/web/20220609211512/https://ftp.ncbi.nih.gov/blast/db/FASTA/">https://web.archive.org/web/20220609211512/https://ftp.ncbi.nih.gov/blast/db/FASTA/</a>
2022-06-20	187	783.58	<a href="https://ftp.ncbi.nih.gov/blast/db/FASTA/">https://ftp.ncbi.nih.gov/blast/db/FASTA/</a>
2022-08-06	205	-	<a href="https://ftp.ncbi.nih.gov/blast/db/FASTA/">https://ftp.ncbi.nih.gov/blast/db/FASTA/</a>
2022-10-17	214	-	<a href="https://ftp.ncbi.nih.gov/blast/db/FASTA/">https://ftp.ncbi.nih.gov/blast/db/FASTA/</a>
2022-11-01	216	-	<a href="https://ftp.ncbi.nih.gov/blast/db/FASTA/">https://ftp.ncbi.nih.gov/blast/db/FASTA/</a>

1387 **Footnotes:**  
 1388 <sup>a</sup> All webpages except <https://ftp.ncbi.nih.gov/blast/db/FASTA/> were retrieved on 2022-11-02.

1389

## 1390 **Supplementary Figures**

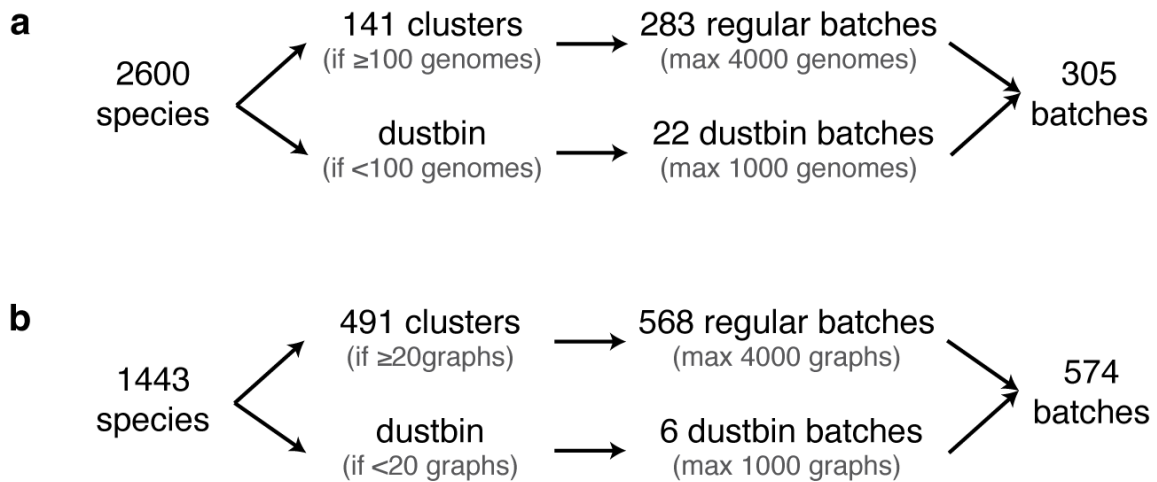
1391

1392

### 1393 **Supplementary Fig. 1: Batching strategies for the BIGSIdata and 661k collections.**

1394 As a clustering strategy, genomes are grouped by individual species, and clusters that are too small are  
1395 placed into a common pseudo-cluster called a dustbin. The obtained clusters and the dustbin are then  
1396 divided into size- and diversity-balanced batches. The plot depicts the batching strategies used for the  
1397 (a) 661k and (b) BIGSIdata collections. For further discussion of the batching, see **Supplementary**  
1398 **Note 3.**

1399



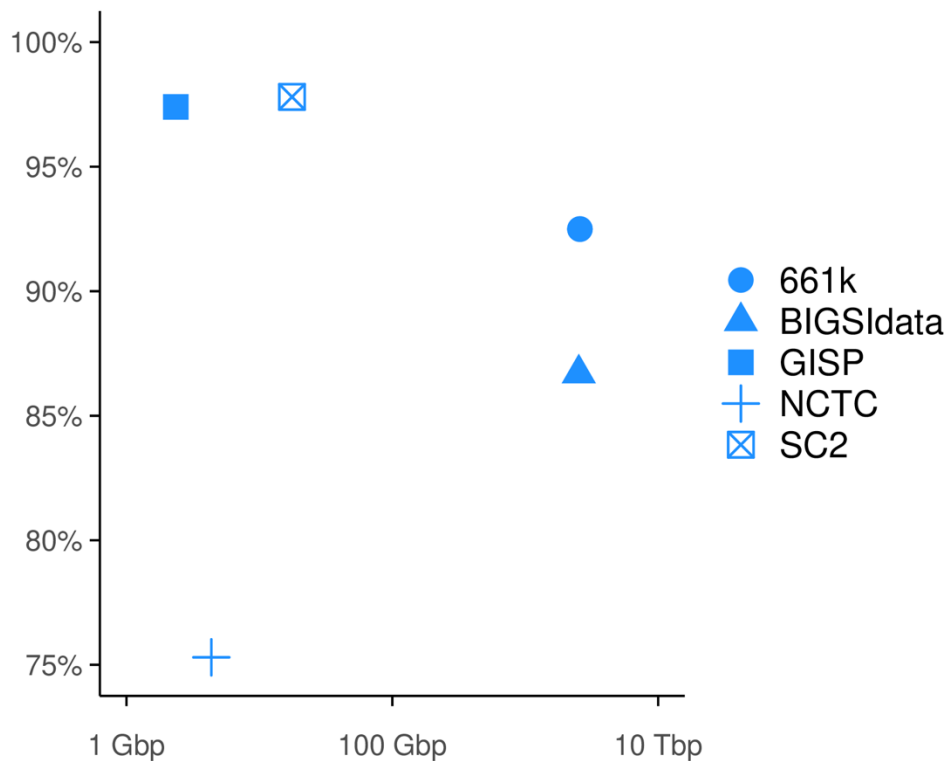
1400

1401

1402 **Supplementary Fig. 2: Quantification of phylogeny-explained data redundancy in the five**  
1403 **test collections.**

1404 The plot depicts the percentage of data redundancy that can be explained by the compressive  
1405 phylogenies in each of the five test collections. The explained redundancy is measured by bottom-up *k*-  
1406 mer propagation along the phylogenies performed by ProPhyle<sup>28,29</sup> and calculated as the proportion of  
1407 *k*-mer duplicities removed by the propagation (see Methods for the formula). A *k*-mer distribution that  
1408 is perfectly explained by the associated compressive phylogeny (i.e., all *k*-mers associated with complete  
1409 subtrees) would result in 100% phylogeny-explained redundancy. The plot shows that for single-species  
1410 batches (modeled by the GISP and SC2 collections), the majority of the signal can be explained by their  
1411 compressive phylogenies, indicative of their extremely high phylogenetic compressibility. In contrast,  
1412 high-diversity batches (modeled by the NCTC3k collection) have more irregularly distributed *k*-mer  
1413 content due to horizontal gene transfer combined with sparse sampling, indicative of their lower  
1414 compressibility (see **Supplementary Fig. 4**). Large and diverse collections, such as 661k and  
1415 BIGSIdata, exhibit thus a medium level of phylogenetically explained redundancies, with the level  
1416 depending on the amount of noise (higher for BIGSIdata and lower for 661k, as also visible in  
1417 **Supplementary Fig. 7**).

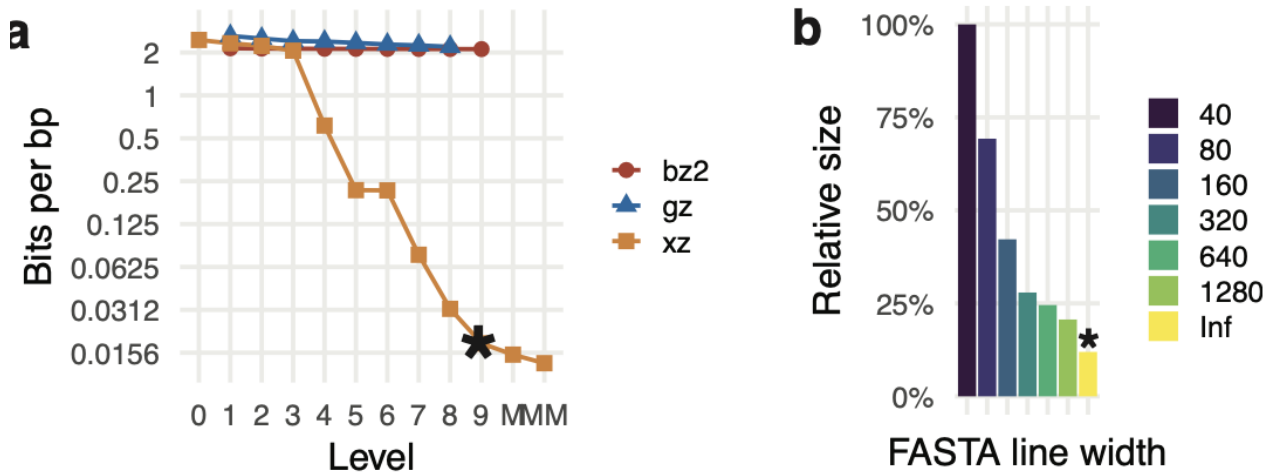
1418  
1419



1420

1421 **Supplementary Fig. 3: Calibration of XZ as a low-level tool for phylogenetic compression.**

1422 The comparison was performed using the assemblies from the GISP collection, with genomes sorted left-  
 1423 to-right according to the Mashree phylogeny. In both subplots, asterisk denotes the mode selected for  
 1424 phylogenetic compression in MiniPhy. **a)** The plot shows the compression performance XZ, GZip, and  
 1425 BZip2 in bits per base pair as a function compression presets (-1, -2, etc.) with single-line FASTA. Given  
 1426 the specific sizes of dictionaries and windows used in the individual algorithms and their individual  
 1427 presets, only XZ with a level  $\geq 4$  was capable of compressing bacterial genomes beyond the statistical  
 1428 entropy baseline (i.e., approximately 2 bits per bp). M and MM denote additional, manually tuned  
 1429 compression modes of XZ with an increased dictionary size (Methods), which slightly improved  
 1430 compression performance but at the same time substantially increased memory and CPU time and were  
 1431 thus not used in MiniPhy. **b)** The plot shows the impact of the FASTA line length on compression  
 1432 performance. With single-line FASTA (denoted by Inf), compression is improved to 12% of the 40 bps  
 1433 per line version. The plot highlights the importance of pre-formatting FASTA data before using general  
 1434 compressors such as XZ.  
 1435



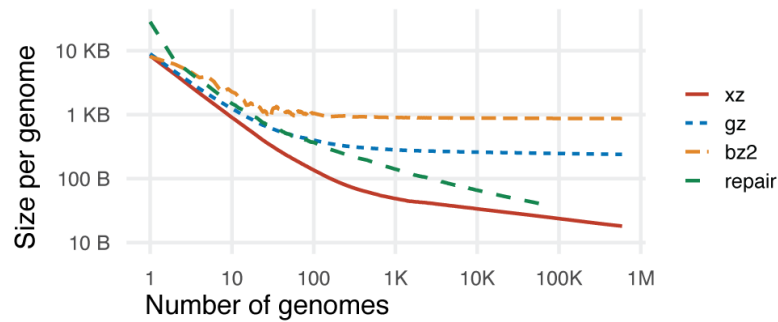
1436  
 1437

1438 **Supplementary Fig. 4: Comparison of three contrasting compression scaling modes of**  
1439 **microbial collections.**

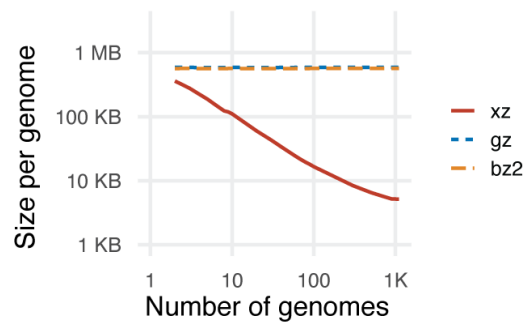
1440 The plots compare on the scaling behavior of the XZ, GZip, BZip2, and RePair compressors on the SC2  
1441 (a), GISP (b), and NCTC3k (c) collections, depicting the space per single genome as a function of the  
1442 number of jointly compressed genomes progressively increased on logarithmic scales. The results  
1443 highlight several key findings. First, XZ consistently outperforms the other compressors. Second, for  
1444 viral genomes all compressors are able to overcome the 2-bits-per-bp baseline thanks to their short  
1445 genome length, but only XZ is able to compress beyond this limit for bacterial genomes (consistent with  
1446 **Supplementary Fig. 3a**). Third, RePair compression can be nearly as effective as XZ for viruses, but  
1447 its non-scalability limits its applicability to large datasets. Fourth, the compressibility of divergent  
1448 bacteria is substantially limited even with the best compressors, with only a 4× improvement in per-  
1449 genome compression for NCTC3k (while the highly compressible SC2 and GISP collections show 171×  
1450 and 105× improvement for the same number of genomes).

1451

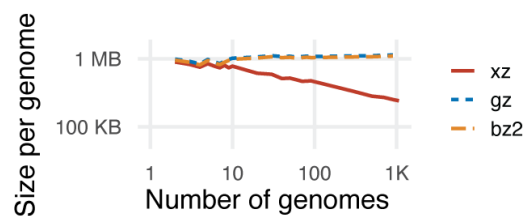
a) SC2 (low-diversity viruses)



b) GISP (low-diversity bacteria)



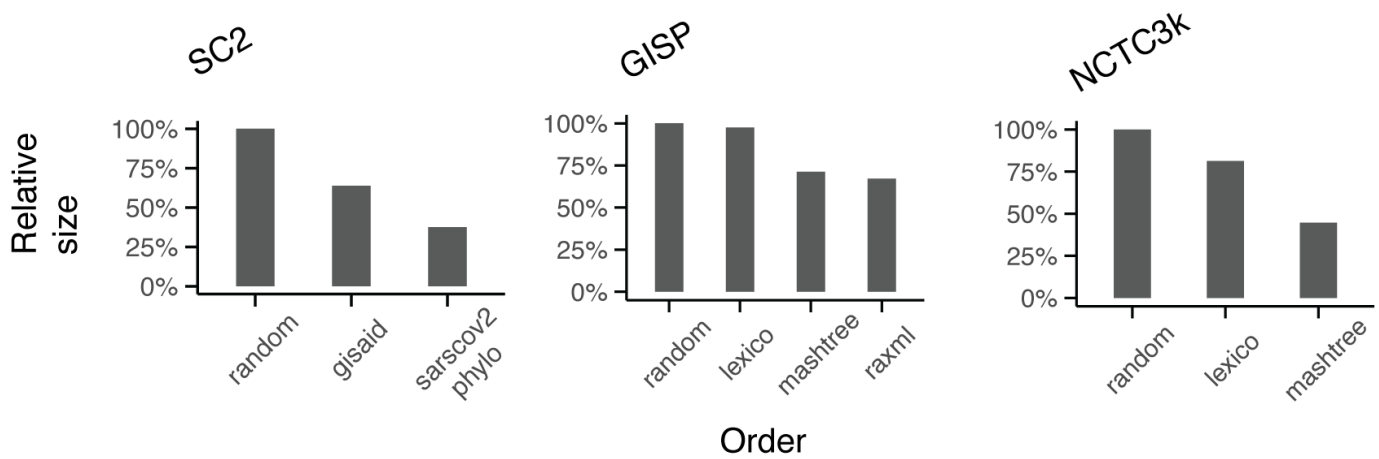
c) NCTC3k (high-diversity bacteria)



1452

1453 **Supplementary Fig. 5: Impact of within-batch genome order on the compressibility of**  
1454 **microbial collections**

1455 While a substantial part of the benefits of phylogenetic compression comes from the organization of  
1456 genomes into batches of phylogenetically related genomes, proper genome reordering within individual  
1457 batches is also crucial for maximizing data compressibility. The plots demonstrate that the impact of  
1458 within-batch reordering grows with the amount of diversity included (GISP vs. NCTC3k) and with the  
1459 number of genomes (GISP vs. SC 2). Accurate phylogenies inferred using RAXML provided only little  
1460 benefits over trees computed using Mashtree (GISP).  
1461

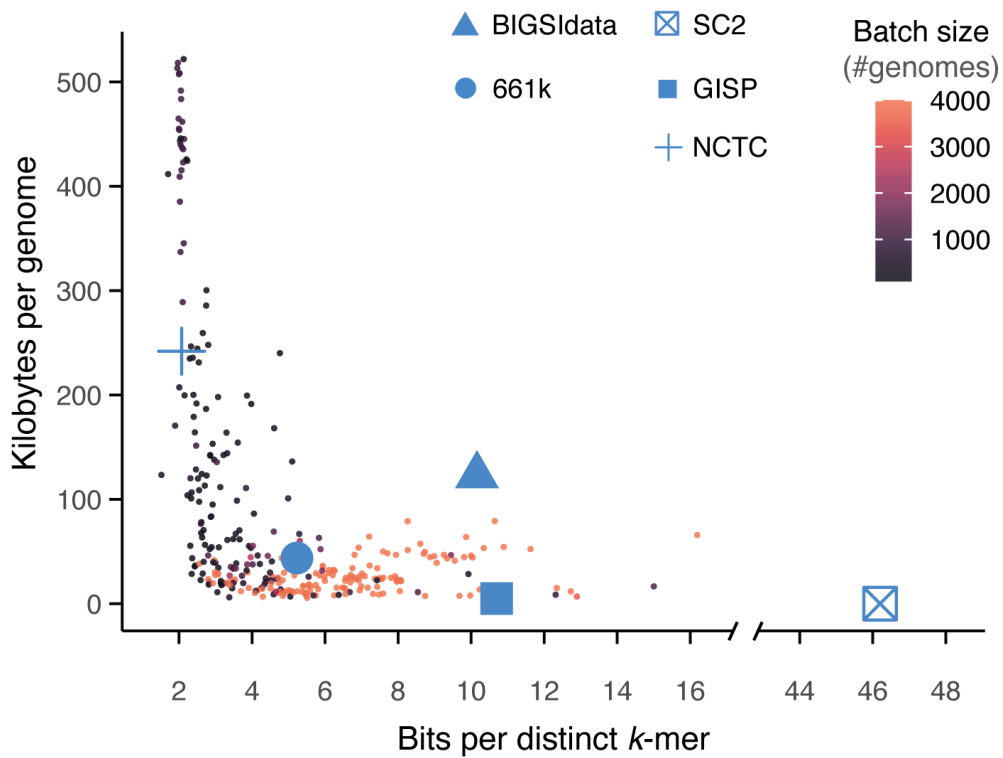




1462 **Supplementary Fig. 6: Compression tradeoffs for the five test collections and for**  
1463 **individual batches of the 661k collection.**

1464 The plot illustrates the tradeoff between the per-genome size after compression and the number of bits  
1465 per distinct  $k$ -mer. The larger points represent individual genome collections and correspond to values  
1466 from **Supplementary Table 3**. The smaller points represent individual batches of the 661k collection,  
1467 with color indicating the number of genomes in each batch. Overall, the plot reveals the influence of  
1468 genomic diversity on the resulting compression characteristics. The tradeoff follows an L-shaped  
1469 pattern, where compression of genome groups with a high diversity leads to smaller space per  $k$ -mer but  
1470 larger space per genome, and conversely for genome groups with a low diversity.

1471

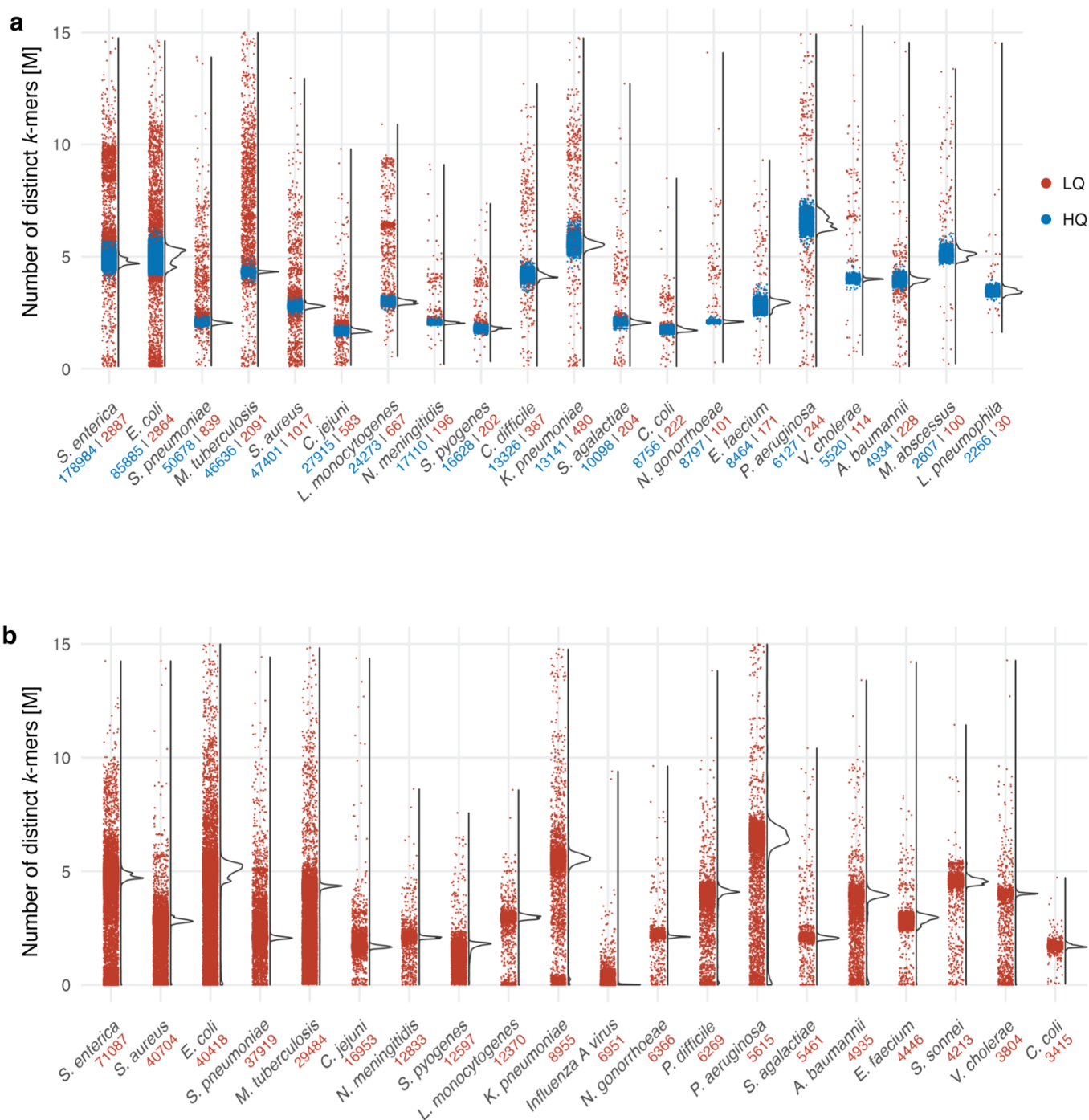


1472

1473

1474 **Supplementary Fig. 7: Distribution of the number of distinct  $k$ -mers in the top 20 species**  
 1475 **in (a) the 661k and (b) BIGSIdata collections.**

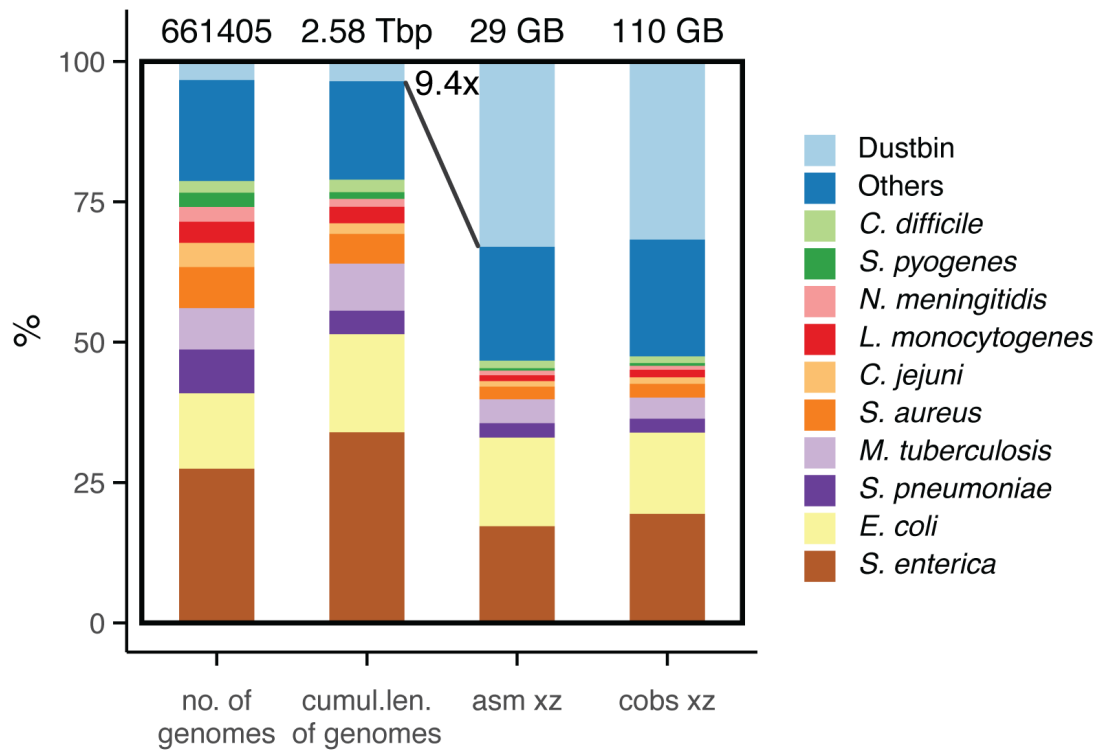
1476 For the 661k collection, colors represent the quality of the assemblies (LQ: low-quality, HQ: high-  
 1477 quality), as determined as part of the quality control in ref<sup>97</sup>. For BIGSIdata, no quality control  
 1478 information is available. The numbers below the species name indicate the number of samples within  
 1479 each category.  
 1480



1482 **Supplementary Fig. 8: Proportions of top 10 species in the 661k collection before and**  
 1483 **after compression.**

1484 The proportions of individual species (their corresponding batches) of the phylogenetically compressed  
 1485 661k collection. The plot depicts the proportions of the top 10 species, the dustbin pseudo-cluster  
 1486 comprising divergent genomes, and the remaining species grouped in Others, while comparing the  
 1487 following four quantitative characteristics: the number of genomes, their cumulative length, the size of  
 1488 the phylogenetically compressed assemblies, and the size of the phylogenetically compressed COBS  
 1489 indexes. While transitioning from the number of genomes to their cumulative length has only a little  
 1490 impact on the proportions (only corresponding to different mean genome lengths of individual species),  
 1491 the divergent genomes occupy a substantially higher proportion of the collection after compression.  
 1492 Moreover, despite genome assemblies and *k*-mer COBS indexes are fundamentally different genome  
 1493 representations (horizontal vs. vertical, respectively), the observed post-compression proportions in  
 1494 them were nearly identical, indicative of that their compression is governed by the same rules.

1495

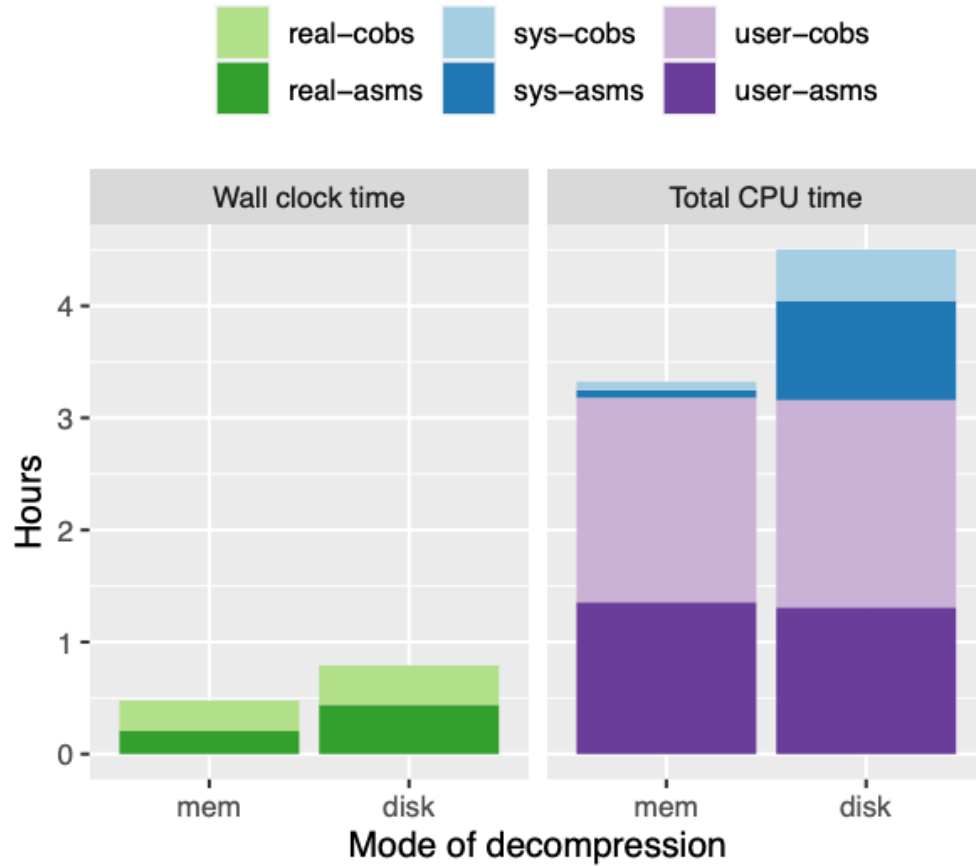


1496

1497 **Supplementary Fig. 9: Time required for decompressing the Phylign database.**

1498 The wall clock and total CPU time required to decompress the Phylign 661k-HQ database, both on a disk  
1499 and in memory, measured on an iMac desktop computer with 4 physical (8 logical) cores. The  
1500 decompression process in memory, which reflects the type of decompression used by Phylign, was  
1501 completed under 30 mins, which is only a fraction of the typical duration of search experiments (see  
1502 **Supplementary Tab. 6**).

1503



1504

1505 **SUPPLEMENTARY FILES**

1506  
1507 Additional supplementary files are provided in a dedicated online repository on  
1508 <http://github.com/karel-brinda/phylogenetic-compression-supplement>.  
1509