MICROBIAL GENOMICS

RESEARCH ARTICLE

Noll *et al.*, *Microbial Genomics* 2023;9:001034

DOI 10.1099/mgen.0.001034

MICROBIOLOGY SOCIETY

OPEN DATA OPEN ACCESS

PanGraph: scalable bacterial pan-genome graph construction

Nicholas Noll[1]†, Marco Molari[2,3]†, Liam P. Shaw[4] and Richard A. Neher[2,3,*]

Abstract

The genomic diversity of microbes is commonly parameterized as SNPs relative to a reference genome of a well-characterized, but arbitrary, isolate. However, any reference genome contains only a fraction of the microbial *pangenome*, the *total* set of genes observed in a given species. Reference-based approaches are thus blind to the dynamics of the accessory genome, as well as variation within gene order and copy number. With the widespread usage of long-read sequencing, the number of high-quality, complete genome assemblies has increased dramatically. In addition to pangenomic approaches that focus on the variation in the sets of genes present in different genomes, complete assemblies allow investigations of the evolution of genome structure and gene order. This latter problem, however, is computationally demanding with few tools available that shed light on these dynamics. Here, we present *PanGraph*, a Julia-based library and command line interface for aligning whole genomes into a graph. Each genome is represented as a path along vertices, which in turn encapsulate homologous multiple sequence alignments. The resultant data structure succinctly summarizes population-level nucleotide and structural polymorphisms and can be exported into several common formats for either downstream analysis or immediate visualization.

DATA SUMMARY

No new data were generated as part of this study. The code and accession numbers of data used are available on github at https://github.com/neherlab/pangraph and archived on zenodo at https://zenodo.org/record/7740393.

INTRODUCTION

During evolution, microbial genomes change by both local mutations and large-scale alterations [1]. Local mutations only change a few nucleotides by substitution or small insertions and deletions. Conversely, large-scale alterations reorganize the sequence, and involve either the homologous recombination of large segments, gene loss or gain, inversions, or mobilization of genetic elements. The accumulation of such changes over time complicates comparative genomic analyses of present-day isolates. Homologous recombination is rapid enough that most genes in many bacterial core genomes have distinct phylogenies [2] and even closely related genomes differ dramatically in gene content [3–5].

Recent advances in long-read sequencing have enabled the low-cost assembly of complete genomes at the quality of reference databases [6]. The accumulation of so many complete genomes promises to rapidly improve our ability to quantify the evolutionary dynamics that drive microbial diversity in natural populations. Since reference-based approaches only partially capture microbial diversity [7], the concept of the *pangenome* has motivated the development of methods that account for substantial variation in gene content. However, though such approaches can accurately capture nucleotide polymorphisms within genes, they usually approximate structural polymorphisms as gene presence–absence relationships [8, 9] irrespective of gene order or orientation. This new era of pangenomics demands novel data structures to encapsulate the *complete* diversity of a given genomic sample set.

In recent years, efforts have focused on generalizing the *pangenome* framework of microbial diversity to graphical models [10]. At a high level, pangenome graphs generalize the reference sequence coordinate system conventionally used and encode genomes

Received 13 December 2022; Accepted 14 April 2023; Published 06 June 2023

Author affiliations: [1]Kavli Institute for Theoretical Physics, University of California, Santa Barbara, CA, USA; [2]Swiss Institute of Bioinformatics, Basel, Switzerland; [3]Biozentrum, University of Basel, Basel, Switzerland; [4]Department of Biology, University of Oxford, Oxford, UK.

***Correspondence:** Richard A. Neher, richard.neher@unibas.ch

Keywords: pangenome; graphs; microbial diversity.

Abbreviations: MGE, mobile genetic element; PRG, pangenome reference graph.

†These authors contributed equally to this work

Data statement: All supporting data, code and protocols have been provided within the article or through supplementary data files. Seven supplementary figures and one supplementary table are available with the online version of this article.

Impact Statement

In addition to evolution through the accumulation of small changes of the genome that arise as errors during replication of their genome, bacteria change and adapt by taking up genetic material and duplicating or deleting parts of their genome. Together with the action of mobile genetic elements, such horizontal exchange and structural modifications are the dominant mechanism of evolution. However, despite its importance, how genome structure and content evolves is poorly understood and few tools are available that help to address this problem. We have developed a tool, *PanGraph*, that can compare the genomes of hundreds of bacteria and summarize them as a graph. This graph represents the large-scale diversity that accumulated in the history of the collection of genomes and is the entry point for further analyses of their evolution.

as directed paths through the graph consisting of nodes that represent sequence. Method development for pangenome graph inference of humans and other eukaryotes is a very active and broad area of research [10]. In multicellular eukaryotes, pangenome graphs usually have an overall linear structure in which structural diversity can be encoded as short-range excursions. In microbial genomics, in contrast, large-scale rearrangements and inversions give rise to complex graph topologies. Here, we focus on pangenome graphs of closely related bacterial genomes.

While easy to conceptualize, the construction of pangenome graphs has proven computationally challenging. Coloured and compacted generalizations of the de Bruijn graph-based assemblers have been successively used to build graphs from large sequence sets [11, 12], with tools existing to build graphs containing thousands of isolates [13]. These graphs have been shown to be useful in problems such as the detection of the core genome [14] or the improvement of gene prediction, annotation and clustering [15]. However, in these methods the graph structure encodes, at the same time, nucleotide-level variation and large-scale structural rearrangements. For investigations of the evolutionary dynamics of these genomes, separating these two scales is often of interest.

An orthogonal approach has been to formulate the inference of the pangenome graph as a multiple genome alignment. Some early methods relied on homology searches to locate syntenic regions, termed *locally collinear blocks* [16, 17], and proved useful in various problems such as improving gene annotations [18]. However, these early methods usually scale poorly to large sets of genomes. Other methods rely on first grouping genes into orthologous clusters. These clusters are then treated as nodes in the graph, with edges linking clusters that are adjacent on at least one of the genomes. These graphs are usually characterized by a syntenic backbone of core gene clusters, with interspersed regions of structural diversity [19, 20]. The structural information contained in these graphs has been used to categorize core/accessory regions [21, 22], and to improve variant detection [23] or gene annotations [24, 25]. By using genes as nodes, these graphs are much smaller in size. However, relying on gene annotations means that these methods are often sensitive to any errors in this initial annotation.

Here we present *PanGraph*, a Julia [26] library and command line interface, designed to efficiently align large sets of closely related microbial genomes into a pangenome graph on personal computers. The *PanGraph* algorithm groups homologous sequences through nucleotide alignments alone, without relying on gene annotation. The resulting graph both compresses the input sequence set and succinctly captures the population diversity at multiple scales: from nucleotide mutations and indels to structural polymorphisms driven by inversions, rearrangements and gene gain/loss. Importantly, these two scales are naturally separated in our representation. The underlying graph data structure can be exported into numerous formats for downstream analysis and visualization in software such as Bandage [27].

ALGORITHMS AND IMPLEMENTATION

PanGraph transforms an arbitrary set of genomes into a *graph* that simultaneously compresses the collection of sequences and exhaustively summarizes both the structural and the nucleotide-level polymorphisms. The graph is composed of *pancontigs*, which represent linear multiple-sequence alignments of homologous sequence found within one or more input genomes. *Pancontigs* are connected by an edge if they are adjacent in at least one input sequence; individual sequences are then recapitulated by contiguous *paths* through the graph. Pancontigs are directed, meaning their orientation in a path can be either 5′ to 3′ or vice versa. More details on the structure of our graph representation can be found in the Supporting Information (SI) sect. I available with the online version of this article.

To construct a pangraph, the algorithm needs to find homologous sequence within and among all input genomes. *PanGraph*'s overarching strategy is to approximate multiple-genome alignment by iterative pairwise alignment of graphs of subsets of sequences, in the spirit of progressive alignment tools [17, 28, 29]. Pairwise graph alignment is performed by an all-to-all alignment of the consensus sequence of all *pancontigs* between both graphs, and the order of pairwise alignments is determined by a guide tree.

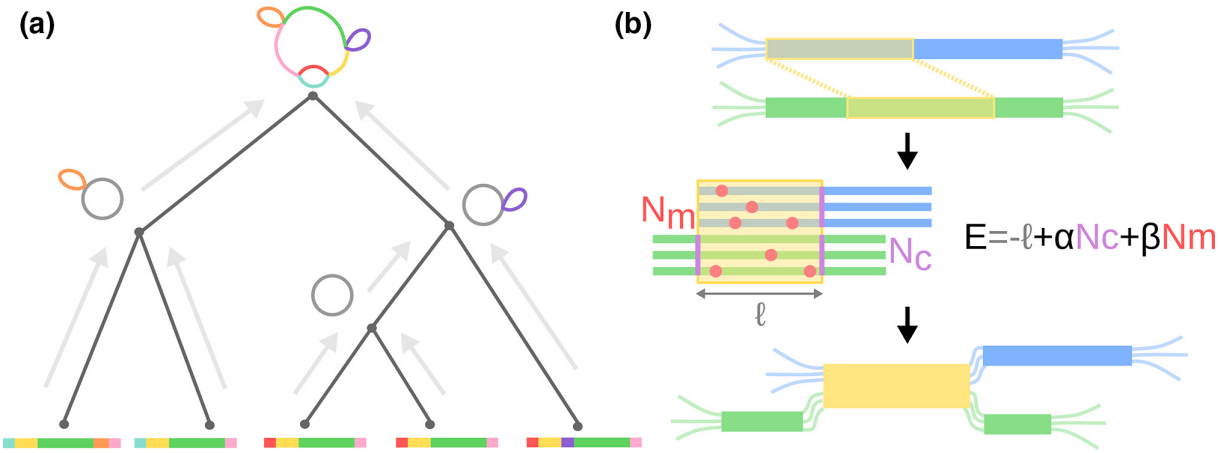

Fig. 1. Overview of the *PanGraph* algorithm. (a) The alignment graph is constructed progressively by aligning graphs pairwise up a guide tree constructed from neighbour-joining the minimizer overlap between strains. (b) During pairwise alignment, *pancontigs* (blue and green) are merged by identifying homologous intervals (shown in yellow). If the underlying alignments are viewed as compatible, i.e. the energy is less than 0, the pancontigs are merged.

a. Guide tree construction

The alignment guide tree is constructed subject to three design constraints: (i) similar sequences are aligned first, (ii) the similarity computation must have good scaling properties with the length and number of input sequences, and (iii) the resultant tree is balanced to maximize parallelism. To this end, we formulate the algorithm as a two-step process. The initial guide tree is constructed by neighbour-joining [30]; the pairwise distance between sequences is approximated by the Jaccard distance between sequence minimizers [31]. Computationally, each sequence can be sketched into its set of minimizers in linear time while the cardinality of all pairwise intersections can be computed by sorting the list of all minimizers to efficiently count overlaps. Hence, the pairwise distance matrix is estimated in a time that increases log-linearly in the length of sequences, and in our implementation can scale sub-quadratically with the number of isolates (see SI sect. II for details). The final guide tree is constructed as the balanced binary tree constrained to reproduce the topological ordering of leaves found initially. This balancing maximizes the number of independent pairwise alignments and thereby allows efficient parallelism. See Fig. 1(a) for a graphical depiction of the guide tree.

b. Iterative graph alignment

The full pangraph representing all genomes is constructed by aligning/merging graphs that represent subsets of the genomes in an iterative manner illustrated in Fig. 1. The iteration starts with one subgraph per input genome, each representing its respective genome as a single *pancontig*. Pairs of subgraphs are aligned in a post-order traversal of the guide tree. The identified homologous intervals of the pancontig are then merged, thereby creating shorter contigs that represent homologous sections of multiple input genomes (see Fig. 1b for a graphical depiction). These steps of pairwise graph alignment and merging of homologous intervals are repeated until the root of the guide tree is reached. *Pancontigs* encapsulate linear multiple-sequence alignments which are modelled internally by a star phylogeny, i.e. are assumed to be well described by a reference sequence augmented by independent SNPs and indels for each contained isolate.

Pairwise alignment

To align two graphs, the consensus sequences of all pancontigs in both graphs are searched for homologies and aligned. Full genome alignment between two closely related isolates is a well-studied problem with many sensitive and efficient tools available [32, 33]. We chose to use *minimap2* as the core pairwise genome aligner for its proven speed, sensitivity and easy-to-use exposed library API [32]. This alignment kernel is included within a custom Julia wrapper, available at github.com/nnoll/minimap2_jll.jl. However, we note that *PanGraph* is written to be modular, and additional alignment kernels can be added with ease. In particular we decided to include the option to use *mmseqs2* [34] as an alternative alignment kernel, because of its sensitivity on highly diverged sequences at the cost of higher computational time.

Merging of homologous sequence

If the above alignment step detected homologous stretches between the consensus sequences of two pancontigs, these pancontigs, or parts of them, can be merged. It is not uncommon that one pancontig has homology with multiple other pancontigs and the iterative algorithm has to make a choice on which potential mergers are performed and in which order. We rank each alignment between two pancontigs according to the pseudo-energy

$$E = -\ell + \alpha N_c + \beta N_m \quad (1)$$

where ℓ, N_c and N_m denote the alignment length, number of additional *pancontigs* created by the merger and number of polymorphisms in the alignment of the consensus sequences, respectively (see Fig. 1b for a graphical definition for each term). E thus compares the compactification of the graph by the length of the match with the increase in complexity of encoding the graph in terms of the number of additional blocks N_c and the diversity within the block N_m. The smaller E, the more favourable the match and only mergers whose alignment has negative pseudo-energy are performed.

The parameter β controls the maximal sequence divergence between the two merger candidates. The negative pseudo-energy requirement means that no merger will be performed if the Hamming distance exceeds $1/\beta$. Importantly, this divergence threshold is applied to the comparison of the consensus sequences of the merger candidates. The pseudo-energy ranking also effectively results in mergers being performed in reverse evolutionary order, with less diverged sequences with long-range synteny being merged first.

The parameter α controls the fragmentation of the graph and imposes a minimal length on the pancontigs resulting from the merger. This parameter has no effect if merging does not increase the number of contigs, but mergers that introduce one, two, three or four cuts are only performed for sufficiently long homologous stretches as parameterized by α. In addition to the fragmentation parameter α, there is a further parameter $L_{\min}$ that explicitly controls the minimal length of pancontigs. Structural variation at a scale smaller than this threshold (i.e. short indels) is stored in pancontig alignments, rather than in the graph structure, and no mergers are performed if they have length shorter than this threshold. For instance, when a merger would generate flanking contigs shorter than this threshold, the merging is performed but the alignment is extended to include these regions. The default value for this parameter is 100 bp.

At the graph level, the merger of two *pancontigs* defines a new *pancontig*, connected on both sides by edges to the neighbouring *pancontigs* of both inputs, and thus locally collapses the two graphs under consideration. At the nucleotide level, the pairwise alignment of two *pancontigs* maps the consensus of one onto the other; merging two *pancontigs* requires the application of the map onto the nucleotide-level diversity of the underlying multiple-sequence alignment. Once both sets of sequences are placed onto a common coordinate system, the resultant consensus sequence, and thus polymorphisms, are recomputed. This procedure can be viewed as an approximate multiple sequence alignment algorithm that aligns homologous parts without further investigating insertions in the two sub-alignments. This shortens considerably the time needed, but might introduce minor inconsistencies and artefacts in the alignments. After building the graph, these can be removed by performing a standard multiple alignment of sequences within each pancontig (see 'polish' command in the documentation and SI sect. IIB).

The above procedure is repeated until no alignments with negative energy remain. More details on the merging procedure and the effect of parameters can be found in SI sect. II.

c. Parallelism

PanGraph guide trees are balanced binary trees that break the task into many independent sub-problems and thereby enable scalable parallelism, as shown in Fig. 1(a) for a cartoon example. Each internal node of the guide tree represents a job that performs a single pairwise graph alignment between its two children. The process will block until both of its children processes have completed and subsequently pass the result of their pairwise graph alignment up to the parent. All jobs run concurrently from the start of the algorithm; the Julia scheduler resolves the order of dependencies naturally [26]. As such, the number of parallel computations is automatically scaled to the number of available threads allocated by the user at the onset of the alignment.

d. Graph export and availability

The constructed pangenome graph can be exported in a variety of file formats for downstream analysis and visualization. In addition to a custom JSON schema, *PanGraph* can export the alignment as a GFA file, where each *pancontig* is represented as a segment and each genome as a path. This allows for visualization in software such as Bandage [27]. Lastly, we provide functionality to export as a conventional presence/absence pangenome – albeit one with *pancontigs* taking the place of putative gene clusters – that can be visualized directly by the PanX toolkit [9]. *PanGraph* is published under an MIT licence with source code, extensive documentation, examples and instructions for installation available at github.com/neherlab/pangraph. All data and scripts used to validate *PanGraph* are available within the same repository.

VALIDATION AND PERFORMANCE

a. Validation on synthetic data

As a first validation step we use generated synthetic data to quantify the performance characteristics of *PanGraph* as a function of input size, and its accuracy as a function of sequence diversity.

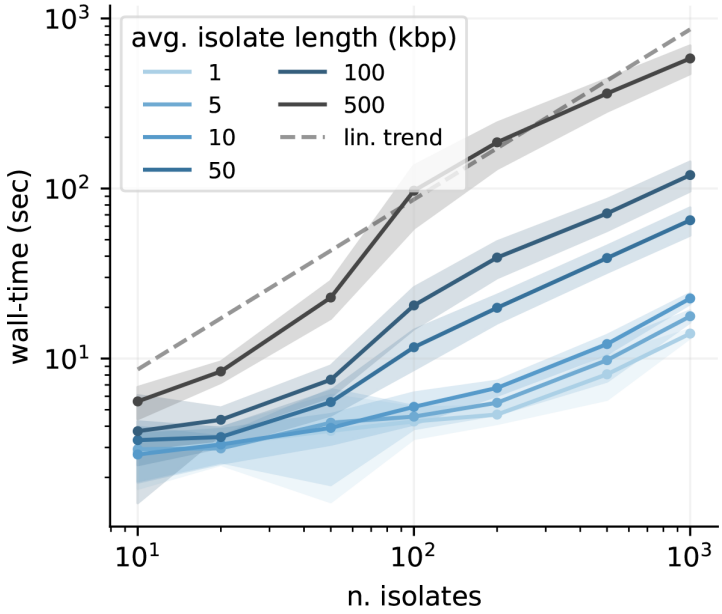

Fig. 2. Algorithm performance. *PanGraph* scales linearly with the number of input genomes. This is a direct result of the guide tree simplification. The solid line and ribbons display the mean and standard deviation over 50 runs. All runs were performed utilizing eight cores, and with the default *minimap2* alignment kernel and *asm20* option.

Generation of synthetic data

We simulated populations of size N and sequence length L utilizing a Wright–Fisher model [35] evolved for T=50 generations. In addition to nucleotide mutations that occur at rate μ per generation, we modelled inversions, deletions (respective rates 0.01 and 0.05 per generation), and horizontal sequence transfer that occurs with tunable rate h per generation per genome (see SI sect. III and Table S1 for details on the simulation and standard parameter values). The ancestral state for each sequence is tracked through each evolutionary event so that the true mosaic relatedness structure can easily be converted to a graph. The simulation framework is distributed within the *PanGraph* command line tools for external use.

Performances as a function of dataset size

The algorithmic complexity was measured empirically by constructing pangenome graphs from data generated by simulating populations of varying size and sequence length. The mean and standard deviation of the run-time obtained from 50 iterations are shown in Fig. 2. Importantly, *PanGraph*'s computational complexity grows *linearly* with the number of input genomes once the number of genomes exceeds a certain threshold and parallelism can be exploited. We note that *PanGraph* scales approximately log-linear with average sequence length, as expected from the underlying algorithmic complexity of *minimap2* [32]. Benchmarks of CPU and memory usage are reported in SI sect. IVA.

Accuracy as a function of sequence divergence

To be useful and informative beyond data compression of microbial genomes, the contigs and break-points of the reconstructed graphs should correspond, at least approximately, to evolutionary events in the history of the sample. We quantified PanGraph's ability to accurately reconstruct the true graph by computing the displacement of the inferred breakpoints relative to their known locus stored from the evolutionary simulation (cf. SI Section IVB for details). We evaluate this displacement by generating datasets with different rates of mutation μ and horizontal sequence transfer h. For each (h, μ) pair we perform 25 different simulations, and build pangenome graphs using three different options for the alignment kernel: *minimap2* [32] with *asm10* and *asm20* options and *mmseqs2* [34]. Critically, we found that the accuracy was *independent* of the rate of horizontal transfer h and thus underlying graph complexity. The predominant determinant of accuracy was the sample diversity, controlled by the mutation rate μ in our simulations (cf. SI Fig. 4). For low-diversity datasets, breakpoints are inferred with accuracy of few basepairs, while for highly diverged isolates most breakpoints are displaced by several hundreds of basepairs (cf. SI Fig. 5). The choice of alignment kernel influences the threshold diversity at which accuracy is lost, suggesting that inability to detect homology between diverged sequences is the reason.

For each alignment kernel and value of average sequence divergence in the population we evaluated the fraction of breakpoints that have displacement greater than the default minimal pancontig size for *PanGraph* of 100 bp (cf. Fig. 3). On the sequences

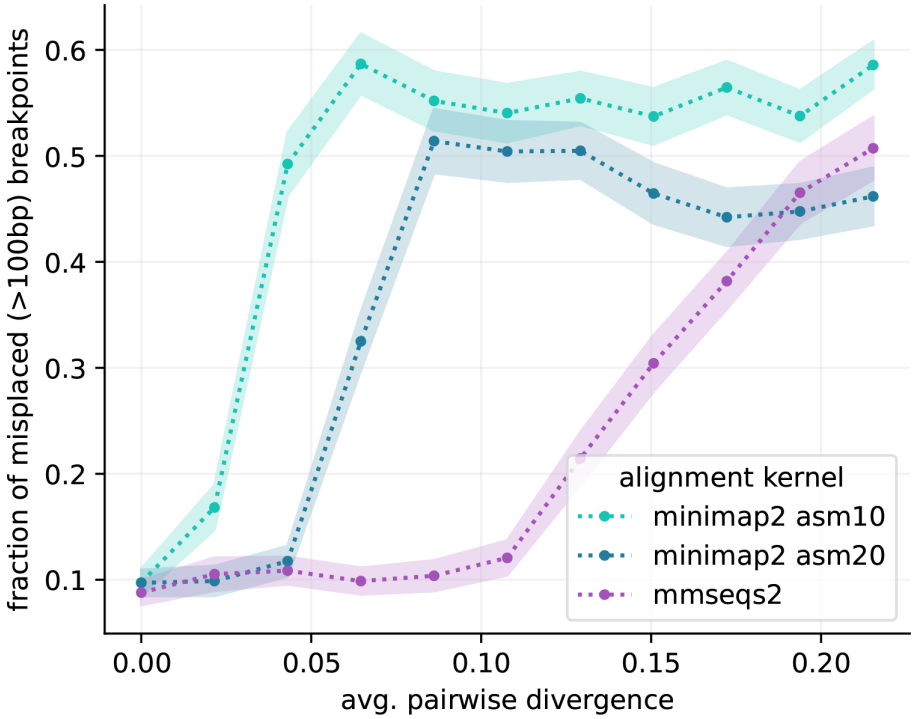

Fig. 3. Accuracy against synthetic data. We generated artificial data with varying degree of sequence divergence, and compared the real underlying pangenome graph with the one reconstructed by *PanGraph*, for three different alignment kernels: *minimap2* with *asm10* or *asm20* option, and *mmseqs2*. In each comparison we evaluated the misplacement of breakpoints that we can pair on the two graphs within 1 kb. The plot displays the fraction of breakpoints that have misplacement greater than the standard *PanGraph* precision threshold of L_{min}=100 bp, as a function of average pairwise sequence divergence. Line and shaded area represent mean and standard deviation over 25 repetitions. *mmseqs2* maintains accuracy at higher divergence, at the cost of higher computational time.

generated by our simulations, *minimap2* with option *asm20* shows a loss of accuracy at an average pairwise sequence divergence of around 5%, while *mmseqs2* is accurate up to around 12%, at the cost of higher computational time. These thresholds are lower than the nominal sensitivity of the aligners, which are 10 and 20% for *minimap2* with settings *asm10* and *asm20*, respectively, and around 30% for *mmseqs2*. This discrepancy is due to the fact that for accurate graph constructions, the largest divergence, rather than the average divergence, is the relevant quantity.

b. Validation on real data

We additionally validated *PanGraph* on genomes from natural populations sampled from RefSeq [36], focusing on the properties of the resulting pangenome graphs as a function of dataset size and diversity.

Dataset characterization

We downloaded from RefSeq [36] completely assembled chromosomes from five different bacterial species: *Escherichia coli*, *Klebsiella pneumoniae*, *Helicobacter pylori*, *Mycobacterium tuberculosis* and *Prochlorococcus marinus*. These data had been previously analysed using PanX [9]. Using the PanX analysis we estimated the size of the pangenome and core genome, and the average pairwise divergence on core genes (cf. Table 1 and SI Section VA).

Among these five data sets, the *E. coli*, *K. pneumoniae* and *M. tuberculosis* genomes have core genome diversity below the thresholds of the *minimap2* aligner, while *H. pylori* is diverse enough that we do not expect *minimap2* to find all relevant matches, while *mmseqs2* should process this set without problems. *P. marinus*, on the other hand, is very diverse with an average core genome diversity of 26% and is beyond what we expect *PanGraph* to handle.

Benchmark on real data

Using *PanGraph*, we built multiple pangenome graphs for each species in the dataset (cf. SI sections VB and VC). These differ by the alignment kernel used (*minimap2* with *asm10* or *asm20* option, and *mmseqs2*) or by the value of the pseudo-energy parameters α and β from eq. (1). Different alignment kernels are expected to reach different accuracy on datasets with different diversities

Table 1. Dataset properties

Using the PanX analysis results we characterized the size and diversity of our dataset. Columns represent: number of isolates (*N*.), average chromosome length (L_{gen}), total pangenome length (L_{pang}), total core genome length (L_{core}, corresponding to genes present in every isolate), total soft-core genome length ($L_{95\%}$, corresponding to genes shared by at least 95% of the isolates), and average pairwise divergence of core genes (d_{core}). Note that L_{core} might be artificially low due to missing annotations in some of the input genomes and $L_{95\%}$ is a more robust measure of the core-genome size.

Species	*N*	L_{gen} (Mb)	L_{pang} (Mb)	L_{core} (Mb)	$L_{95\%}$ (Mb)	d_{core}
E. coli	307	5.0	17.7	0.7	2.0	1.6%
K. pneumoniae	109	5.3	13.0	2.3	3.7	0.8%
H. pylori	85	1.6	1.9	0.6	1.0	4.2%
M. tuberculosis	51	4.4	4.1	2.4	3.4	0.03%
P. marinus	10	1.8	3.3	0.7	0.7	26.9%

(cf. Fig. 3). Moreover the use of null energy parameters (α=β=0) is expected to remove the threshold divergence of 10% associated with the standard values of parameters (α=100, β=10) at the cost of more fragmented pangenome graphs.

PanGraph can build pangenome graphs comprising hundreds of isolates in a few hours (5 h for 307 *E. coli* isolates with *minimap2* kernel; cf. Fig. 4a). Use of *mmseqs2* consistently requires longer computational time, but provides higher sensitivity when merging highly diverged sequences.

Figs 4(b, c) and S6 quantify the size of the core genome, i.e. the sum of all pancontigs present once in every genome, identified by *PanGraph* and the compression of the dataset, i.e. the ratio of the sum of the lengths of all pancontigs and the sum of the lengths of all input genomes, for different datasets and the different parameters. As expected, the very homogenous *M. tuberculosis* dataset has a large core genome and a compression ratio that is almost equal to the inverse number of input genomes, independent of the aligner or parameters used. *E. coli* and *K. pneumoniae* compress similarly well, but their core genome is a much smaller fraction of the pangenome. Using the more sensitive aligner *mmseqs2* and relaxing the merger parameters leads to some additional compression, presumably due to merging of diverged paralogues.

The *H. pylori* dataset sits at the limit of what can be accurately merged with the *minimap2* kernel. In this case the use of *mmseqs2* and null energy parameters increases core pangenome fraction, decreases the total pangenome size and does not compromise the fragmentation of the graph (cf. Figs 4b, c and S6). The divergence of the *P. marinus* dataset sits instead beyond the capabilities of *PanGraph*. In this case no alignment kernel can reach satisfactory sequence compression, and only *mmseqs2* combined with null energy parameters is able to retrieve a few core pancontigs.

For species other than *P. marinus*, the pangraphs contain thousands to tens of thousands of pancontigs, and 50% of the pangenome sequence is contained in pancontigs spanning several thousand base pairs (see SI Fig. S6). For these species the use of null energy parameters slightly increases merging and sequence compression, at the cost of slightly more fragmented graphs

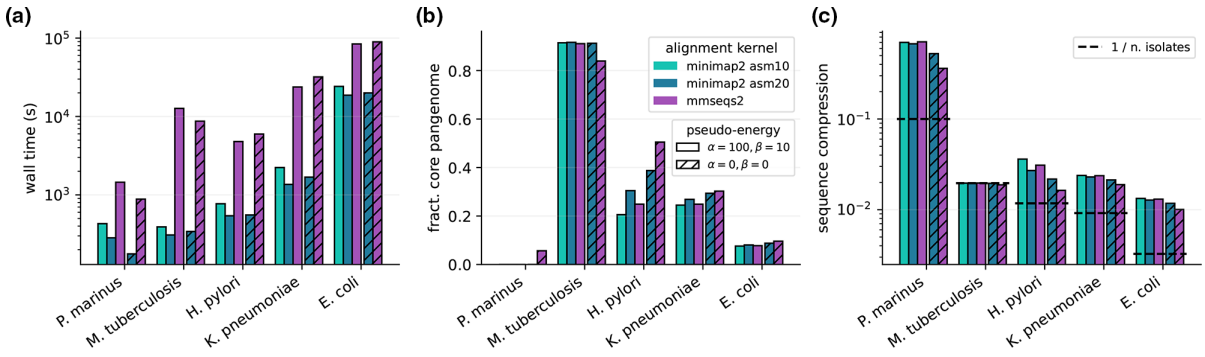

Fig. 4. Benchmark on real data. We built pangenome graphs from fully assembled chromosomes from five different bacterial species. For each species we built graphs with three different alignment kernel options (*minimap2* with *asm10* or *asm20* options and *mmseqs2*) and two different settings for the pseudo-energy parameters α and β (standard or null values). (a) *PanGraph* wall-time when run in parallel on eight cores. (b) Fraction of core pangenome in the pangenome graph. (c) Sequence compression, defined as the ratio between the pangenome graph size and the cumulative size of all the sequences contained in the graph. Since maximal compression depends on the number of isolates *n* in the pangenome graph, we mark for reference the value of 1/*n* for each dataset.

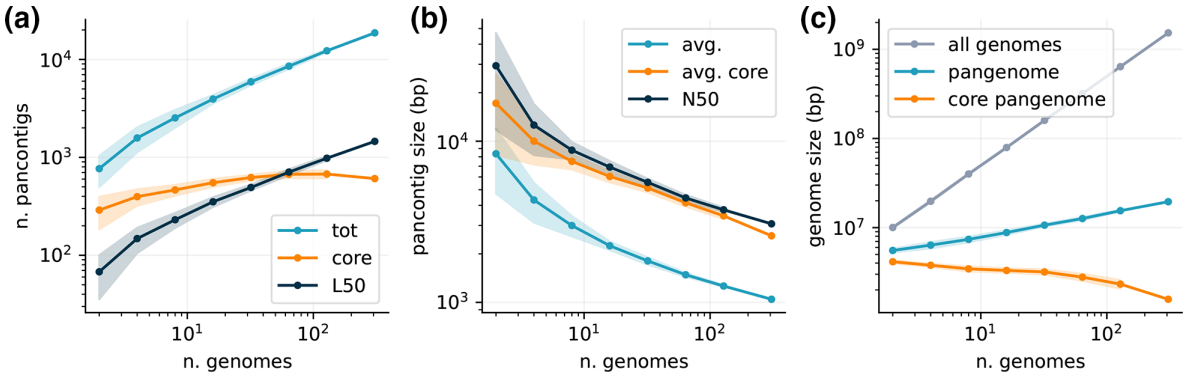

Fig. 5. Pangenome graph properties vs. dataset size. We built pangenome graphs with an increasing number of isolates from the *E. coli* dataset and measured the scaling of different properties of the graphs. Graphs were built using the *minimap2* alignment kernel with *asm20* option. Lines and shaded areas represent the mean and standard deviation over 10 different repetitions on random subsets of isolates, except for the final point indicating the full graph (307 isolates). (a) Number of pancontigs in the graph. We count the total number of pancontigs (blue), the number of core pancontigs (orange) and the minimum number of pancontigs that contain more than 50% of the pangenome (L50, black). (b) Average size of pancontigs (blue), of only core pancontigs (orange), and size of the smallest pancontig in the minimal set that spans 50% of the pangenome (N50, black). (c) Cumulative size of all genomes in the pangenome graph (grey), total pangenome size (blue) and size of the core pangenome (orange).

(cf. SI Section VC and Fig. S6). Overall this benchmark showcases the capabilities and limits of *PanGraph*, and demonstrates the value of adapting the pseudo-energy parameter and choice of alignment kernel to the expected diversity of the dataset considered.

Scaling with dataset size

We then explored how the properties of the pangraph scale with the dataset size. To do so, we built pangenome graphs from randomized subsets of isolates of increasing size from the *E. coli* dataset. The results are reported in Fig. 5.

The number and size of pancontigs scale sub-linearly with the number of isolates, with more than 50% of the pangenome being included in the 10% longest pancontigs. Core pancontigs have higher-than-average length. This is expected given that core genes tend to stay syntenic over large evolutionary distances. Adding more strains does not generate excessive fragmentation of these pancontigs, and their size remains of several thousand base pairs even as hundreds of strains are added. The size of the core genome does not decrease significantly with the addition of new strains, while the total pangenome size remains orders of magnitude smaller than the total size of all genomes included in the graph.

GRAPH MARGINALIZATION

The interpretation of a pangenome graph containing hundreds of strains can be challenging. To this end, it is often informative to inspect simpler sub-graphs comprising only a subset of isolates. However, building many such sub-graphs is computationally intensive. To facilitate this task *PanGraph* provides the *marginalize* command, which can be used to project a large pangenome graph on a small subset of strains, removing all the other paths and merging transitive pancontigs. This operation is computationally much cheaper than building a new graph for the subset of strains considered.

In addition to being a useful operation, marginalization allows us to quantify the consistency and robustness of pangraph construction on real data, where the ground truth is unknown. To verify that marginalized graphs are compatible with newly built graphs, we built a pangenome graph for 50 randomly selected strains from the *K. pneumoniae* dataset. We then picked 50 random pairs of strains from the same set and built two graphs for each pair: the graph obtained by marginalizing the complete graph on the pair (i.e. marginalized graph in Fig. 6a top), and the graph built directly from the pair (i.e. pairwise graph in Fig. 6a top). In order to compare the two graphs, we compute the partition they generated on the genome of the isolates they include. Each genome is partitioned in pancontigs that are either shared on the pair or private to one isolate (cf. Fig. 6a bottom). We can classify the segments in the intersection of the two partitions in different categories, depending on whether the two partitions agree on whether the segment is shared or private. Moreover, segments on which the two partitions agree are further subdivided depending on whether they are private or shared on both partitions.

The compatibility between the two graphs requires segments on which the two partitions disagree to be few and short. Indeed, we verified that over the pairs we picked, these segments cover a very small fraction of the genome (< 1%) and have average size compatible with the default 100 bp precision threshold of *PanGraph* set by the value of L_{min} (cf. Fig. 6b, c). Conversely, segments on which the partitions agree have average size of several thousand base pairs. The fact that the two partitions almost completely agree cannot be explained simply by the fact that most of the genome is shared, since segments that are shared on both graphs cover

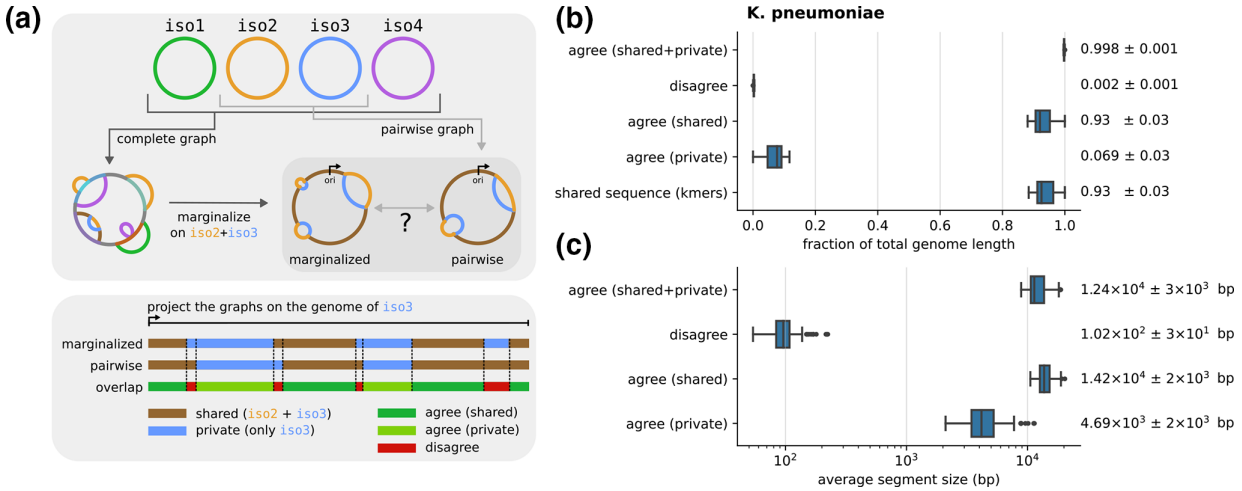

Fig. 6. Test of graph marginalization. (a) We built a pangenome graph from 50 randomly chosen strains from the *K. pneumoniae* dataset. We then randomly picked 50 pairs of strains. For each pair we compared the pangenome graph obtained by marginalizing the complete graph on the pair of strains, and the one obtained by building a new graph for the pair (top). The comparison is done by considering that each graph partitions a genome in shared and private segments. By combining the partitions generated by the marginalized and pairwise graphs we categorize segments in three categories, depending on whether the two partitions agree or not, and if they agree depending on whether segments are shared or private. All graphs were built using the *minimap2* alignment kernel with *asm20* option and default value for the energy parameters. (b) Distribution of the average fraction of the genome covered by segments of each category, over the 50 pairs considered (two entries per pair). The last line represents the distribution of shared sequence, approximated using the fraction of shared k-mers corrected using sequence divergence as described in the main text. Next to each distribution we report its mean and standard deviation. (c) Distribution of average segment lengths for each category over the 50 pairs considered (two entries per pair). Mean and standard deviation are reported.

on average only 85% of the sequence. We confirmed these results using the fraction of shared k-mers, using an approach inspired by PopPUNK [37]. Namely, we approximate the fraction of homologous sequence between any two pairs using the fraction of shared k-mers (k=21) divided by $(1-d)^k$, where d is the average pairwise divergence on core genes for the pair considered. This divisor corrects for k-mers on homologous sequence that are not shared due to mutations. The resulting distribution of shared sequence is in very good agreement with the fraction of segments that are shared on both graphs (cf. Fig. 6b), suggesting that homologous sequences are correctly merged on the complete, marginalized, and pairwise graphs.

We performed the same test on species from the other datasets, obtaining similar results on all species whose divergence is compatible with *PanGraph* capabilities (cf. SI Section VI and Fig. S7).

DISCUSSION

While single nucleotide differences in the core genome are straightforward to analyse with existing tools, these analyses miss the great majority of genetic diversity. The ability to rapidly align large sets of complete genomes of a bacterial species is crucial for the investigation of the processes governing microbial genome evolution and structure. *PanGraph*, the tool presented here, is able to capture the structural and nucleotide diversity in both the core and accessory genome in a scalable way. The resulting data structure captures large-scale structural variation in the connectivity of the graph, while nucleotide-level variation is included in pancontig alignments.

In our analysis we demonstrated the capabilities and limits of this tool, both on synthetic and real sequences. The efficient implementation of *PanGraph* allows it to create pangenome graphs containing hundreds of isolates in a few hours on a eight-core machine. The size of pancontigs and the fraction of core sequence have good scaling properties with the number of isolates in the graph, indicating that *PanGraph* is able to successfully capture pangenome properties. By construction, *PanGraph* operates in a gene-agnostic way, being only based on sequence homology, and is thus robust to gene annotation errors.

One of the main limitations is the diversity of the input sequences. With the default *minimap2* alignment kernel, *PanGraph* is able to correctly merge genomes with up to 5% average divergence. Sequences with higher divergence (up to 10–15%) can be merged using the *mmseqs2* alignment kernel, and tuning the α and β energy parameters (cf. eq. 1), trading off speed for sensitivity. For more diverged data sets, homology detection tools that use protein sequences would probably be necessary.

We also provide the ability to quickly marginalize big graphs on a subset of strains, obtaining simpler graphs that can more easily be explored. Graphs can also be exported in different formats for further analysis and visualization.

The applications of *PanGraph* are not limited to whole bacterial chromosomes. Plasmids and other mobile genetic elements (MGEs) play a fundamental role in microbial evolution [38, 39], including in the spread of antibiotic resistance [40]. MGEs often display a wide variety of sequence and structural diversity. Being able to capture and represent this diversity is key to understanding their evolution, particularly on epidemiologically relevant timescales where very few SNPs accumulate but large-scale structural changes are frequent [41, 42]. In this sense, the pangenome graph is a naturally powerful representation, and *PanGraph* can be used to explore this diversity. Some example applications on plasmids are available in *PanGraph*'s documentation.

Even if *PanGraph* was not developed with this aim in mind, its outputs can be used as input for other tools that perform variant calling and genotyping using Pangenome Reference Graphs (PRGs), such as *pandora* [23] and *gramtools* [43]. To this end the multiple sequence alignment of each block can be produced using the export command (see *PanGraph*'s documentation), and can be transformed in a set of PRGs using *make_prg* (see https://github.com/iqbal-lab-org/make_prg). However, we did not directly test or benchmark these approaches in the current paper.

We hope that *PanGraph* will prove a valuable tool, with the potential to spur new insights into microbial diversity and the processes by which bacteria adapt and change.

Funding information
This study was funded by the University of Basel (M.M. and R.A.N.) and the Gordon and Betty Moore Foundation Grant #2919.02 (N.N.). L.P.S. is a Sir Henry Wellcome Postdoctoral Fellow funded by Wellcome (Grant 220422/Z/20/Z).

Acknowledgement
We are grateful to Boris Shraiman for stimulating discussions.

Conflicts of interest
The authors declare that they are not aware of relevant conflicts of interests.

References

1. Arnold BJ, Huang I-T, Hanage WP. Horizontal gene transfer and adaptive evolution in bacteria. *Nat Rev Microbiol* 2022;20:206–218.
2. Sakoparnig T, Field C, van Nimwegen E. Whole genome phylogenies reflect the distributions of recombination rates for many bacterial species. *Elife* 2021;10:e65366.
3. Touchon M, Perrin A, de Sousa JAM, Vangchhia B, Burn S, *et al*. Phylogenetic background and habitat drive the genetic diversification of *Escherichia coli*. *PLoS Genet* 2020;16:e1008866.
4. Touchon M, Hoede C, Tenaillon O, Barbe V, Baeriswyl S, *et al*. Organised genome dynamics in the *Escherichia coli* species results in highly diverse adaptive paths. *PLoS Genet* 2009;5:e1000344.
5. Doolittle WF, Zhaxybayeva O. On the origin of prokaryotic species. *Genome Res* 2009;19:744–756.
6. Whibley A, Kelley JL, Narum SR. The changing face of genome assemblies: guidance on achieving high-quality reference genomes. *Mol Ecol Resour* 2021;21:641–652.
7. Tettelin H, Riley D, Cattuto C, Medini D. Comparative genomics: the bacterial pan-genome. *Curr Opin Microbiol* 2008;11:472–477.
8. Page AJ, Cummins CA, Hunt M, Wong VK, Reuter S, *et al*. Roary: rapid large-scale prokaryote pan genome analysis. *Bioinformatics* 2015;31:3691–3693.
9. Ding W, Baumdicker F, Neher RA. panX: pan-genome analysis and exploration. *Nucleic Acids Res* 2018;46:e5–5.
10. Eizenga JM, Novak AM, Sibbesen JA, Heumos S, Ghaffaari A, *et al*. Pangenome graphs. *Annu Rev Genomics Hum Genet* 2020;21:139–162.
11. Iqbal Z, Caccamo M, Turner I, Flicek P, McVean G. De novo assembly and genotyping of variants using colored de Bruijn graphs. *Nat Genet* 2012;44:226–232.
12. Muggli MD, Bowe A, Noyes NR, Morley PS, Belk KE, *et al*. Succinct colored de Bruijn graphs. *Bioinformatics* 2017;33:3181–3187.
13. Holley G, Melsted P. Bifrost: highly parallel construction and indexing of colored and compacted de Bruijn graphs. *Genome Biol* 2020;21:249.
14. Schulz T, Wittler R, Stoye J. Sequence-based pangenomic core detection. *iScience* 2022;25:104413.
15. Horsfield ST, Croucher NJ, Lees JA. Accurate and fast graph-based pangenome annotation and clustering with ggCaller. bioRxiv;2023:01–35. DOI: 10.1101/2023.01.24.524926.
16. Angiuoli SV, Salzberg SL. Mugsy: fast multiple alignment of closely related whole genomes. *Bioinformatics* 2011;27:334–342.
17. Darling AE, Mau B, Perna NT, Stajich JE. progressiveMauve: multiple genome alignment with gene gain, loss and rearrangement. *PLoS One* 2010;5:e11147.
18. Angiuoli SV, Dunning Hotopp JC, Salzberg SL, Tettelin H. Improving pan-genome annotation using whole genome multiple alignment. *BMC Bioinformatics* 2011;12:1–11.
19. Chan AP, Sutton G, DePew J, Krishnakumar R, Choi Y, *et al*. A novel method of consensus pan-chromosome assembly and large-scale comparative analysis reveal the highly flexible pan-genome of *Acinetobacter baumannii*. *Genome Biol* 2015;16:143.
20. Oliveira PH, Touchon M, Cury J, Rocha EPC. The chromosomal organization of horizontal gene transfer in bacteria. *Nat Commun* 2017;8:841.
21. Sutton G, Fogel GB, Abramson B, Brinkac L, Michael T, *et al*. A pan-genome method to determine core regions of the *Bacillus* subtilis and *Escherichia coli* genomes. *F1000Res* 2021;10:286.
22. Gautreau G, Bazin A, Gachet M, Planel R, Burlot L, *et al*. PPanGGOLiN: depicting microbial diversity via a partitioned pangenome graph. *PLoS Comput Biol* 2020;16:e1007732.
23. Colquhoun RM, Hall MB, Lima L, Roberts LW, Malone KM, *et al*. Pandora: nucleotide-resolution bacterial pan-genomics with reference graphs. *Genome Biol* 2021;22:267.
24. Tonkin-Hill G, MacAlasdair N, Ruis C, Weimann A, Horesh G, *et al*. Producing polished prokaryotic pangenomes with the Panaroo pipeline. *Genome Biol* 2020;21:180.
25. Zhou Z, Charlesworth J, Achtman M. Accurate reconstruction of bacterial pan- and core genomes with PEPPAN. *Genome Res* 2020;30:1667–1679.
26. Bezanson J, Edelman A, Karpinski S, Shah VB. Julia: a fresh approach to numerical computing. *SIAM Rev* 2017;59:65–98.
27. Wick RR, Schultz MB, Zobel J, Holt KE. Bandage: interactive visualization of de novo genome assemblies. *Bioinformatics* 2015;31:3350–3352.
28. Feng DF, Doolittle RF. Progressive sequence alignment as a prerequisite to correct phylogenetic trees. *J Mol Evol* 1987;25:351–360.

29. Armstrong J, Hickey G, Diekhans M, Fiddes IT, Novak AM, *et al*. Progressive Cactus is a multiple-genome aligner for the thousand-genome era. *Nature* 2020;587:246–251.
30. Saitou N, Nei M. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Mol Biol Evol* 1987;4:406–425.
31. Roberts M, Hayes W, Hunt BR, Mount SM, Yorke JA. Reducing storage requirements for biological sequence comparison. *Bioinformatics* 2004;20:3363–3369.
32. Li H, Birol I. Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics* 2018;34:3094–3100.
33. Marçais G, Delcher AL, Phillippy AM, Coston R, Salzberg SL, *et al*. MUMmer4: a fast and versatile genome alignment system. *PLoS Comput Biol* 2018;14:e1005944.
34. Steinegger M, Söding J. MMseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. *Nat Biotechnol* 2017;35:1026–1028.
35. Hudson RR. Generating samples under a Wright-Fisher neutral model of genetic variation. *Bioinformatics* 2002;18:337–338.
36. O'Leary NA, Wright MW, Brister JR, Ciufo S, Haddad D, *et al*. Reference sequence (RefSeq) database at NCBI: current status, taxonomic expansion, and functional annotation. *Nucleic Acids Res* 2016;44:D733–D745.
37. Lees JA, Harris SR, Tonkin-Hill G, Gladstone RA, Lo SW, *et al*. Fast and flexible bacterial genomic epidemiology with PopPUNK. *Genome Res* 2019;29:304–316.
38. Haudiquet M, de Sousa JM, Touchon M, Rocha EPC. Selfish, promiscuous and sometimes useful: how mobile genetic elements drive horizontal gene transfer in microbial populations. *Phil Trans R Soc B* 2022;377:1861.
39. Frost LS, Leplae R, Summers AO, Toussaint A. Mobile genetic elements: the agents of open source evolution. *Nat Rev Microbiol* 2005;3:722–732.
40. van der Zee A, Kraak WB, Burggraaf A, Goessens WHF, Pirovano W, *et al*. Spread of carbapenem resistance by transposition and conjugation among *Pseudomonas aeruginosa*. *Front Microbiol* 2018; 9:2057.
41. Sheppard AE, Stoesser N, Wilson DJ, Sebra R, Kasarskis A, *et al*. Nested Russian doll-like genetic mobility drives rapid dissemination of the carbapenem resistance gene blaKPC. *Antimicrob Agents Chemother* 2016;60:3767–3778.
42. Noll N, Urich E, Wüthrich D, Hinic V, Egli A, *et al*. Resolving structural diversity of Carbapenemase-producing gram-negative bacteria using single molecule sequencing. *Microbiology* 2018.
43. Letcher B, Hunt M, Iqbal Z. Gramtools enables multiscale variation analysis with genome graphs. *Genome Biol* 2021;22:1–27.