RESEARCH

KPop: An assembly-free and scalable method for the comparative analysis of microbial genomes

Xavier Didelot[1] and Paolo Ribeca[2,3*]

*Correspondence:
paolo.ribeca@ukhsa.gov.uk
[3]UK Health Security Agency, London, UK
Full list of author information is available at the end of the article

Abstract

It has become increasingly difficult for traditional analysis techniques based on assembly and variant calling to cope with the recent explosion in the amount of available sequencing data, especially in the field of microbial genomics. Here we introduce KPop, a novel versatile method based on full k-mer spectra and dataset-specific transformations, that allows for the accurate comparison of hundreds of thousands of assembled or unassembled microbial genomes in a matter of hours. We validate the method on simulated datasets and show that it can correctly classify sequences into lineages and rapidly identify related sequences. We also demonstrate its usefulness on several real-life datasets, in the case of both viral and bacterial pathogens. The KPop open-source code is available on GitHub at https://github.com/PaoloRibeca/KPop.

Keywords: microbial genomics; comparative analysis; sequence relatedness; classification; k-mers

Background

The last 20 years have seen such a revolution in high-throughput sequencing technology that it is now possible to sequence whole microbial genomes quickly and at low cost, enabling their use for a wide range of applications in the epidemiology of infectious diseases [1, 2, 3, 4]. Consequently, many studies have recently been carried out comparing hundreds or thousands of microbial genomes within a population of interest, which can be put in context with the very large numbers of genomes available in publicly accessible databases such as GenBank [5], BIGSdb [6] and EnteroBase [7]. However, the bioinformatic process needed to compare microbial genomes remains complex and laborious.

The first bioinformatic step typically involves the analysis of the raw sequencing reads. It can be performed by first aligning the reads to a reference genome, for example using `bwa` [8], Bowtie [9], or GEM [10], followed by calling variants relative to the reference, for example using `samtools` [11] or FreeBayes [12]. Alternatively, analysis can be done without the use a reference genome, for example using Velvet [13] or SPAdes [14], in which case the process is known as *de novo* assembly [15]. Neither of these two approaches is fully satisfactory. A reference-based approach requires a predetermined reference genome, the choice of which becomes of paramount importance. Only genomic regions within this reference can be compared, which creates a bias in the analysis and forbids any pan-genome analysis [16]. Assembling the genomes *de novo* does not suffer from these difficulties, but requires an additional step to align the assembled genomes

against each other. A full alignment is typically impossible beyond a handful of genomes, for example using progressiveMauve [17] or Mugsy [18]. Therefore the alignment is often performed only for core regions or genes, for example using a cgMLST scheme [19], or by aligning each genome against a reference, for example MUMmer [20], which reintroduces the need and role of a reference and the loss of any information on the accessory genome. All these approaches are also sensitive to biological evolutionary processes such as genome rearrangements and recombination.

Here we propose a novel methodology for the comparison of large numbers of microbial genomes, that does not require assembly or alignment. Our approach is based on k-mers, which are nucleotide sequences of short length k found in the genomes. The use of k-mers in microbial genomics has been important before, especially for studies interested in quantifying the variability of the pan-genome [21, 22], including genome-wide association mapping studies that need to capture evolution in both core and accessory genome [23, 24].

Our strategy is conceptually different from most other comparable methods based on k-mers, which can be broadly divided into the following categories:

(1) Methods using specific, possibly very long k-mers to detect the presence or absence of specific sequences, as in Mykrobe [25].
(2) Methods using all k-mers with a low value of $k = 4$ or 5 and relying on a dimensional reduction in order to cluster and visualise sequences in an abstract small-dimensional space ("binning", see for instance [26, 27]).
(3) Methods using simplified signatures based on some dataset-independent choices (for instance, k-mer "minimizers" as in Mash [28]) to keep signatures ("sketches") small.
(4) Methods such as PopPUNK [29] that use distances, possibly derived from sketches, to summarise the relationships between sequences.

Instead, we use a full k-mer spectrum with a value of k typically ranging from 10 to 12. This is more robust than methods in class (1) above since it does not rely on long k-mers which can be disrupted by mutations, but is still powerful because we consider a full spectrum. With respect to methods of type (2), using longer k-mers requires a more sophisticated algorithmic implementation due to the much larger size of the resulting k-mer spectra, but it also allows us to estimate distances much more accurately based on more specific sequence matches. As for methods in (3), we also use a transformation on k-mer spectra to reduce their dimensionality, but one that is tailored to the content of the dataset under analysis; whereas the use of minimizers is an arbitrary choice that does not guarantee optimality. Compared to methods in (4), our method also allows us to estimate distances, but it additionally retains a representation of each sample as a vector embedded in a reduced-dimension space, which can be stored for later use and presented as input, for instance, to classification workflows.

Furthermore, most of the existing methods are based on the k-mers found in *de novo* assemblies of the genomes. Dealing with unassembled genomes presents a major additional computational challenge since raw read data can be two orders of magnitude larger than assembled data, so that only very few methods have been proposed to analyse k-mers directly from unassembled genomes. One exception

is Mash, which uses only a subset of k-mers, leading to high scalability even when used directly on unassembled raw reads. However, its usefulness is mostly for categorisation at the species level, rather than comparison of closely related genomes [28].

By contrast, the methodology we describe here allows the comparison of large numbers of genomes with high resolution, be they assembled or not, based on the full set of k-mers present in all the sequencing reads or in some specific subset of them. To demonstrate the accuracy of our approach, we apply it to several simulated datasets for which the true relationships between genomes are known, and show that it can correctly classify sequences into lineages and rapidly identify related genomes. We also demonstrate the usefulness of our method on several real-life datasets, in the case of both viral and bacterial pathogens.

Results

General overview

Let us consider that we want to analyse a *dataset*, which is a collection of *samples*. In turn, each sample will usually consist of one or more files generated with high-throughput sequencing techniques, and associated metadata. Alternatively, albeit less commonly, the sample might be one or more genomic sequences previously assembled from the sequencing data. Input sequences can therefore be either NGS FASTQ read files or assembled FASTA sequences.

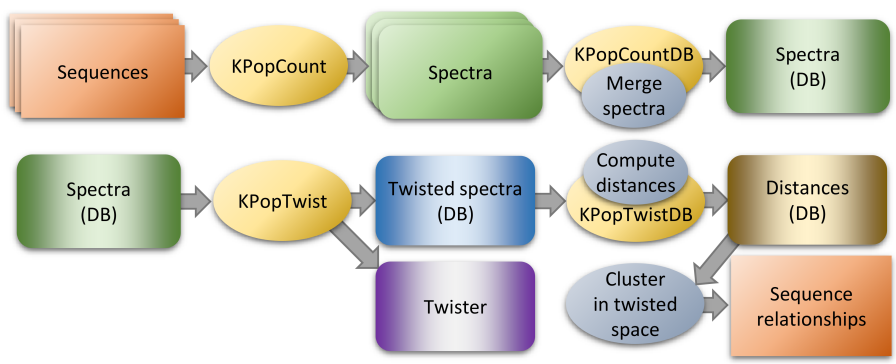

Figure 1 Basic workflow. Full k-mer spectra are deduced from input sequences and, after processing through `KPopCount` and `KPopCountDB`, stored into a binary database. These spectra can then be twisted to a reduced-dimensional representation thanks to `KPopTwist`; the resulting twisted vectors can be used to establish relations between sequences through their distances using `KPopTwistDB` or directly with other classification methods (not illustrated).

The overall workflow we use and describe in this paper is depicted in Figure 1 and can be summarised in three steps:

(1) First, we consider the unbiased frequencies of *all* k-mers for a given and sufficiently large value of k, after suitable denoising and rescaling have been optionally performed.

(2) Second, we determine a transformation that performs a dimensional reduction step *optimised for the dataset at hand*. We call this transformation the "twister"; and we call "twisted" any k-mer spectrum to which the transformation has been applied. Computing the twister involves full k-mer spectra and

is relatively compute- and resource-intensive. However, the optimisation of the twister can be usually performed on a representative subset of the sequences, which simplifies computation. The twister can then be stored for future use (for instance, to perform clustering and/or classification). Once the optimal twister has been determined, its application to novel samples to twist their k-mer spectra takes a very little amount of computing time, which is determined by the dimension of the twister and independent of the sample.

(3) Third, from the twisted sequences, classifiers can be built and relations between samples established. This can happen either via the computation of distances in twisted space or directly, for instance with methods such as decision trees [30] or random forests [31].

In general, the `KPop` method presented here has several desirable features. It produces unbiased (database-independent) spectra, that can be re-used in the future no matter which sequences are known at a given point in time. If novel sequences that are very different from what was already present in the database become known, one just has to regenerate the twister at step (2) and re-compute reduced-dimensional signatures from the spectra, which is quick and straightforward. In fact, `KPop` can be applied to samples with an arbitrary k-mer composition – no classical genomic method consisting of read alignment or assembly is needed. When comparing strains of micro-organisms, our method can automatically use information about the full genome rather than just a set of core genes, which represents a substantial advantage – for instance, `KPop` is able to automatically take into account the presence of novel plasmids carrying antimicrobial resistance genes or other virulence factors even when the exact sequence for such genetic elements is not known or when the bacterial strains do not differ in their chromosomal sequences. On the other hand, by performing a suitable pre-processing of the data (see Methods), it is possible to focus on specific regions of the genome whenever needed. Finally, classifiers based on `KPop` are largely insensitive to recombination and other mechanisms that would be challenging for the models of sequence evolution underlying the construction of phylogenetic trees.

Many different variations on this general strategy can be implemented, especially if one provides the different computational blocks listed above as modular units that can be freely combined into more complex workflows. A detailed description of the protocols used in the rest of the paper, including commands to reproduce the examples, is available in the Methods section or online [32]. All the tests mentioned in the text were performed on an Intel Xeon CPU E5-2680 v4 clocked at 2.40GHz, of which we only used 16/56 logical hyperthreads and up to 256GB of RAM. While for this paper we were not interested in an exact measurement of the time spent in each example, we do provide approximate timings.

Classification of simulated tuberculosis genomes

We simulated a dataset of 1000 genomes of *Mycobacterium tuberculosis*. First a genealogy was generated with samples taken between year 2000 and 2022, at random from one of ten lineages that emerged in year 1900 and last shared a common ancestor around year 1500 (cf Figure 2A). The ten lineages were sampled with the same probability, so that the number of samples from each lineage was

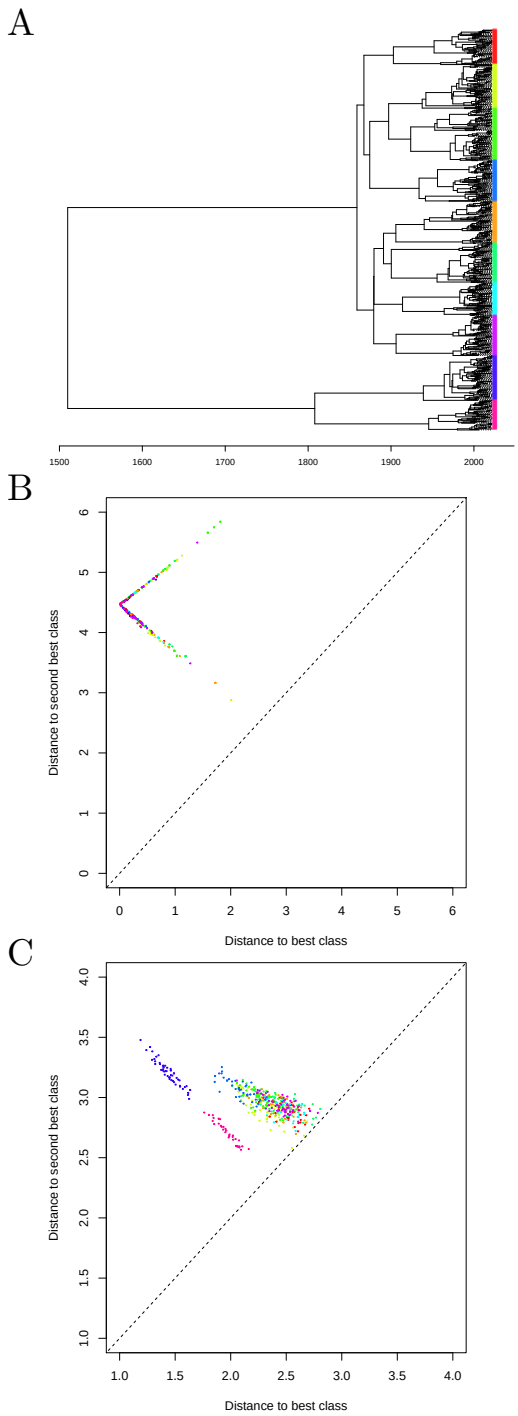

Figure 2 Analysis of simulated dataset of 1000 tuberculosis genomes. (A) Genealogy used for the simulation. The leaves of the tree are shown using unique colors for each of the ten lineages. (B) Results of the classifier using whole-genome sequences. Each test genome is shown as a dot, with the x-axis showing the distance to the closest class and the y-axis the distance to the second closest class. (C) Same as part (B) but based on raw read sequences rather than whole-genome sequences.

relatively constant, ranging between 75 and 130. Genomes of length 4.4 Mbp were then generated by mutation on the branches of this tree, with a substitution rate of 1.1×10^{-7} per year per site [33] being applied to the reference genome H37Rv [34]. Within each lineage, a randomly selected half of the genomes was used to train the classifier and the remaining half was used to test its accuracy.

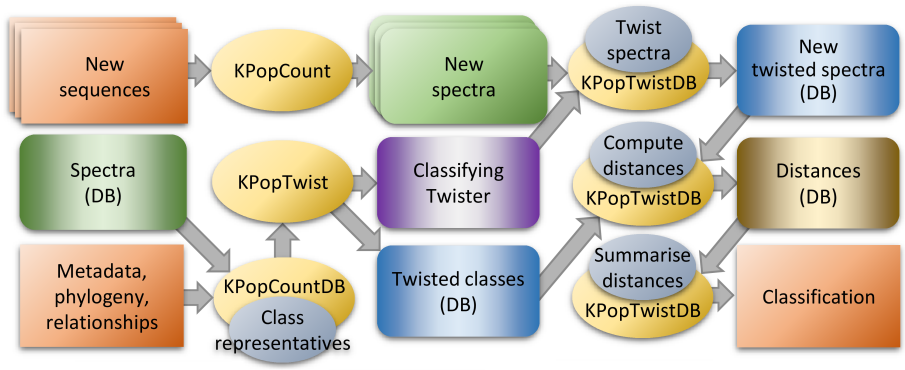

Figure 3 Classification workflow. Lower left corner: One starts with a set of spectra that have been computed according to the upper part of the basic workflow (Figure 1). We also assume that a suitable categorisation of such spectra into classes can be inferred from available metadata. One then computes representative spectra for each class, by generating with `KPopCountDB` a linear combination of the normalised spectra for all the sequences belonging to the class. The representative spectra are then twisted with `KPopTwist`. That returns a classifying twister (mauve box) and the twisted spectra for the representatives of each class (blue box). **Upper row:** Full k-mer spectra are computed with `KPopCount` for new sequences that we would like to classify; through `KPopTwistDB` they are twisted according to the transformation determined at the previous step, and stored into a database of twisted spectra. **Lower right**: Distances between the twisted spectra for the new sequences and the twisted spectra for class representatives are computed and summarised with `KPopTwistDB`. The name of the closest class representative is considered to be the correct classification. More class representatives, up to a set number of closest neighbours, can be considered to provide estimates of the likelihood of the classification and/or to implement k-NN classification schemes. **Not illustrated:** Alternatively, classifiers that do not imply computation of distances (for instance, using decision trees [30] or random forests [31]) can also be implemented if one starts directly from the twisted spectra for the *training* sequences and their classification as deduced from associated metadata.

We first built the classifier (obtained following the procedure illustrated in Figure 3) using the whole genome sequence data as input. Training the classifier took 3 minutes and applying it to the test genomes took 4 minutes. All genomes were correctly classified. Figure 2B shows that in every case the distance to the closest (correct) class was much smaller than the distance to the second closest class, meaning that the classification was categorical.

We then attempted to classify the same dataset based on raw read data rather than the whole genome sequences. In order to do so, simulated next-generation sequencing (NGS) data was generated for each genome using ART [35], emulating Illumina HiSeq 2500 paired-end reads of length 150bp with an average coverage of 20 fold. Training the classifier took 32 minutes and applying it to the test genomes took 17 minutes. Once again all the genomes were correctly classified. However, Figure 2C shows that for a few genomes the distance to the closest class was sometimes only slightly smaller than the distance to the second closest class. This was especially true for genomes belonging to a lineage that had a closely related lineage in the original phylogeny shown in Figure 2A.

Classification of *Mycobacterium* sequencing read data

We collected a high-throughput sequencing dataset containing 1318 *Mycobacterium* genomes, all of which are available on the Sequence Read Archive [36] (Table S1). Each genome belongs to one of the 9 species *M. abscessus*, *M. avium*, *M. bovis*, *M. canettii*, *M. caprae*, *M. orygis*, *M. smegmatis*, *M. tuberculosis* and *M. ulcerans*. Furthermore, *M. bovis* was subdivided into whether the genomes were derived or not from the Bacille Calmette-Guérin (BCG) and *M. tuberculosis* was also subdivided into the 7 lineages L1, L2, L3, L4, L5, L6 and L7. There were therefore a total of 16 classes in this dataset, and the class of each genome was taken either from BIGSdb [6], TB-Profiler [37], or from previous publications [38]. The BCG genomes were manually selected and downloaded from SRA.

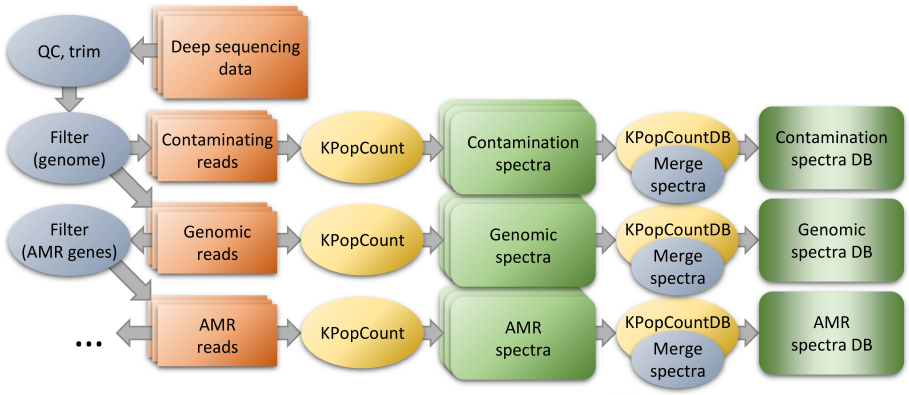

Figure 4 Data preprocessing workflow. When analysing NGS datasets with KPop, one can optionally pre-process sequencing reads in order to eliminate biases and/or have the method focus on specific parts of the genome. For instance, one might align reads to a (pan-)genome and separate them into reads that align (likely to originate from the organism being studied) and reads that do not (likely to come from contaminations). Furthermore, reads that do map to the pan-genome might be separated into groups specific to different genomic features; for instance, one might align them to a set of MLST genes or AMR genes. Full k-mer spectra would then be separately obtained from each group of reads (contaminations, pan-genomic, MLST genes, AMR genes) and given as input to downstream/classification methods. The choice of the group of reads from which spectra are computed determines the set of sequences seen by the method, and hence the scope of the classification.

A data preprocessing workflow was applied to all the genomes (see Figure 4 and Methods) to filter out contaminations, i.e. reads not originating from the *Mycobacterium* genome being sequenced. Briefly, all the reads for each sample were adaptor- and quality-trimmed with TrimGalore [39] and subsequently split into chunks of 25nt. The chunks were mapped with the GEM mapper [10] version 3 onto a "pangenome" obtained by downloading from GenBank [5] all the complete genomes for the 9 species considered. Each paired-end read was kept when both ends had a length $\geq$ 50nt after trimming, and at least 3 chunks had been mapped to the *Mycobacterium* pangenome. The surviving reads were subsequently merged with `flash` [40] and their spectrum computed with `KPopCount`. Samples for which less than 5×10^7 k-mers were left after this procedure (corresponding to a coverage of roughly $10\times$) were discarded. A similar workflow was applied to samples for which single-end reads were available.

Table 1 Results of the *Mycobacterium* classification using distance to class.

Lineage	Number	First correct	Second correct	Neither
M. abscessus	49	49	0	0
M. avium	34	34	0	0
M. bovis	50	44	4	2
M. bovis BCG	47	47	0	0
M. canettii	5	4	0	1
M. caprae	50	44	5	1
M. orygis	8	8	0	0
M. smegmatis	21	21	0	0
M. tuberculosis L1	55	54	1	0
M. tuberculosis L2	50	47	2	1
M. tuberculosis L3	52	51	1	0
M. tuberculosis L4	51	49	1	1
M. tuberculosis L5	51	51	0	0
M. tuberculosis L6	50	50	0	0
M. tuberculosis L7	28	28	0	0
M. ulcerans	49	49	0	0
Total	650	630	14	6

Table 2 Results of the *Mycobacterium* classification using k-nearest neighbours.

Lineage	Number	Correct, certain	Correct, uncertain	Incorrect, uncertain	Incorrect, certain
M. abscessus	49	49	0	0	0
M. avium	34	32	2	0	0
M. bovis	50	50	0	0	0
M. bovis BCG	47	46	0	1	0
M. canettii	5	4	0	0	1
M. caprae	50	49	1	0	0
M. orygis	8	8	0	0	0
M. smegmatis	21	21	0	0	0
M. tuberculosis L1	55	51	4	0	0
M. tuberculosis L2	50	47	0	2	1
M. tuberculosis L3	52	51	1	0	0
M. tuberculosis L4	51	48	0	2	1
M. tuberculosis L5	51	51	0	0	0
M. tuberculosis L6	50	50	0	0	0
M. tuberculosis L7	28	28	0	0	0
M. ulcerans	49	49	0	0	0
Total	650	634	8	5	3

Table 3 Results of the *Mycobacterium* classification using random forest.

Lineage	Number	Correct	Incorrect
M. abscessus	49	49	0
M. avium	34	34	0
M. bovis	50	49	1
M. bovis BCG	47	47	0
M. canettii	5	3	2
M. caprae	50	50	0
M. orygis	8	7	1
M. smegmatis	21	21	0
M. tuberculosis L1	55	55	0
M. tuberculosis L2	50	49	1
M. tuberculosis L3	52	51	1
M. tuberculosis L4	51	49	2
M. tuberculosis L5	51	51	0
M. tuberculosis L6	50	50	0
M. tuberculosis L7	28	28	0
M. ulcerans	49	49	0
Total	650	642	8

A total of 668 genomes, roughly half in each class, was used to train the classifier, and the remaining 650 genomes were used to test its accuracy. Training and testing took approximately one hour each.

Three different classifications methods were used:

1 A method based on finding the closest class in twisted space, as described in the previous section and online [32].
2 A k-nearest neighbour method (with $k = 5$) based on finding the most represented class among the k training sequences which are closest to the test sequence in twisted space.
3 A random forest [31] method based on the coordinates and metadata of the training sequences in twisted space.

In all cases the twister used is the one obtained by running `KPopTwist` on class representatives, which are defined as the averaged normalised linear combinations of the k-mer spectra of all the sequences belonging to each class (see Methods and [32]).

The results of the classification of the test set with method 1 are summarised in Table 1. For 630 of the genomes (97%) the correct class was identified as the closest one, with a further 14 genomes (2%) for which the correct class was found to be the second closest one. This left only six genomes ($< 1\%$) for which the correct class was neither the first nor the second closest one.

The results of the 5-NN classification (method 2) are shown in Table 2. For 634 of the genomes (98%) the correct class was found with higher probability, and for a further eight genomes (1%) the correct class was found with lower probability. This left only respectively five and three genomes for which an incorrect class was identified, with low and high probability, respectively.

Finally, Table 3 shows the results of method 3, the random forest classifier [31], in which only eight genomes (1%) were misclassified.

The three genomes that were misclassified with high probability in the k-nearest neighbours analysis (Table 2) were among the six genomes for which the correct class was neither the first or second closest one (Table 1), and among the eight genomes misclassified according to the random forest method (Table 3). According to the original metadata for our dataset, these three genomes belonged to *M. canettii*, *M. tuberculosis* L2 and *M. tuberculosis* L4, respectively. However, there might have been some mistakes in these classifications, and we noted that some of the designations in BIGSdb [6] did not always agree with the results of TB-Profiler [37]. Such mistakes would make our results look worse than they are if they are present in the test set, but, more importantly, they could affect classification result even for test genomes that are correctly labeled if they are present in the training set. Overall, it is likely that a better curated dataset would produce an even more accurate classifier.

Interestingly, the method seems to work in a way which is largely independent of the sequencing technology and the coverage provided by each NGS experiment — the number of reads available for the different samples spans almost three orders of magnitude.

Classification of simulated SARS-CoV-2 whole genome sequences

We simulated a dataset of 10,000 whole genome sequences of SARS-CoV-2. First, a genealogy was simulated with sampling happening throughout year 2021, from 100 lineages that emerged in the second half of year 2020 and shared a last common ancestor in early 2020 (cf Figure 5A). Since the lineages emerged at different

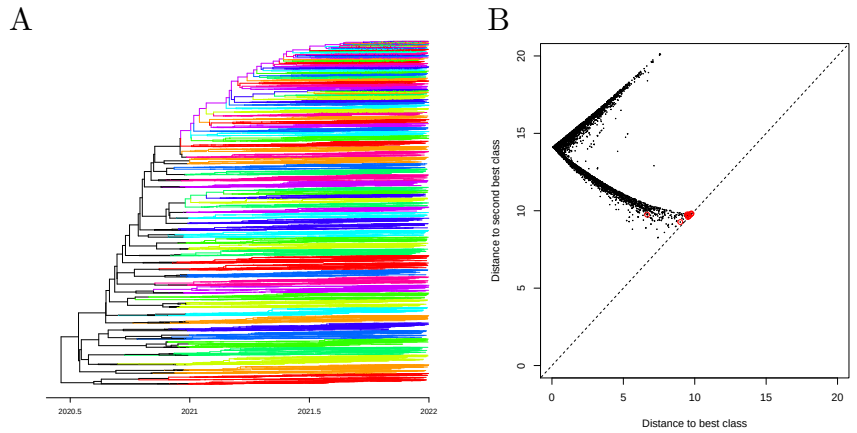

Figure 5 Analysis of simulated dataset of 10,000 SARS-CoV-2 genomes. (A) Genealogy used for the simulation. The branches of the tree are colored non-uniquely according to lineages. (B) Results of the classifier. Each test genome is shown as a dot, with the x-axis showing the distance to the closest class and the y-axis the distance to the second closest class. Genomes for which the closest class was not the correct class are highlighted with a red circle.

times, some were sampled more than others, with the number of samples for each lineage ranging between 2 and 309. Genomes of length 29,903 bp were generated by mutation on the branches of this tree, using as starting point the reference genome Wuhan-Hu-1 [41] and applying a substitution rate of 10^{-3} per year per site [42].

Within each lineage, a randomly selected half of the genomes was used to train the classifier and the remaining half was used to test the classifier. Training the classifier took about 8 minutes and applying it to the test set took about 4 minutes. Figure 5B shows for each of the test genomes the distance to the closest and second closest classes. For the vast majority (99.8%) of the genomes, the closest class corresponded to the correct class, so that the classifier identified the correct class. There were only 11 genomes for which this was not the case, highlighted with red circles in Figure 5B. Many of these had similar distances between the closest and second closest classes, with the latter being the correct class. The only genome for which that was not case was from a small class with just three representatives, two of which were used for training and one for testing, with the test genome happening to be more dissimilar to the two training genomes than to other genomes of other classes.

Classification of SARS-CoV-2 sequences

We also reanalysed the Pangolin dataset of COVID-19 lineage designations [43, 44]. This global dataset contains 1,285,005 full genome sequences of SARS-CoV-2 classified into 1636 lineages. The number of sequences per lineage is very unbalanced, ranging from 3 (for lineage C.6) to 68,299 (for lineage AY.122). We used half of the sequences in each lineage for training of the classifier and the other half for testing. Training and testing took about two hours each. 91.6% of the test sequences had their smallest distance to the correct class (Figure 6), with a further 4.4% having their second smallest distance to the correct class.

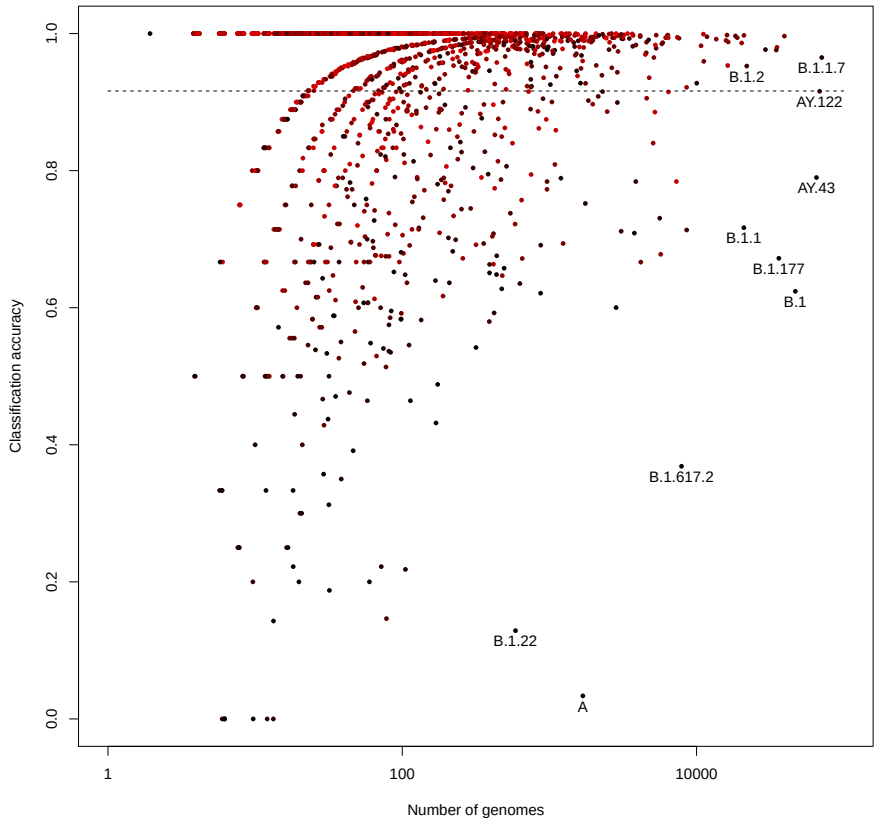

Figure 6 Analysis of Pangolin SARS-CoV-2 dataset. Each point represents a lineage, the x-axis shows the number of genomes and y-axis the proportion of correctly classified genomes. The colors represent the level of confidence in the classification, ranging from black to red proportionally with the difference in distance between the first and second closest classes.

Almost all cases of misclassifications were due to sublineage designations — in fact, they might be due to the fact that the sequences present in the dataset [44] for founding lineages, albeit being accurate at the time of their first designation, were not always updated to a more accurate classification as novel sub-lineage definitions became available. For example, one of the lineages with the lowest proportion of correct classification was B.1. Of the 22,558 B.1 sequences present in the test set, only 14,074 had smallest distance to B.1, but a further 8,370 had smallest distance to a B.1.x lineage. If we consider such classifications as correct, we find that overall 95.5% of the test sequences had their smallest distance to the correct class, with a further 2.8% having their second smallest distance to the correct class. Furthermore, if we consider only the sequences for which the smallest distance is at least 10% smaller than the second smallest distance, we find that 80% of the test sequences meet this criterion and 99% of these sequences have the smallest distance to the correct lineage. In other words, as expected, the accuracy of the classification is higher for lineages for which the classifier is more confident (Figure 6).

Identification of closely related SARS-CoV-2 sequences

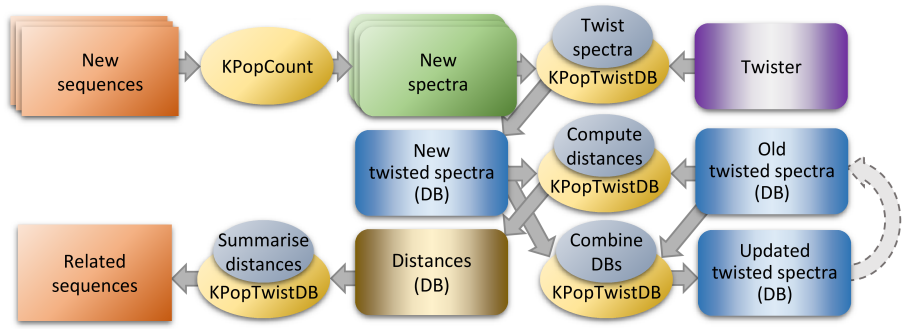

Figure 7 Relatedness engine workflow. The workflow is very similar to the classification one depicted in Figure 3, and can be considered a derivative of it — given a classification workflow, one can always automatically obtain a workflow determining relatedness according to the criteria used to classify sequences. After twisting new sequences according to the classifying twister found according to the procedure of Figure 3, one uses `KPopTwistDB` to compute the distances between them and a set of twisted spectra previously stored in a database. Such distances can be summarised with `KPopTwistDB` in order to identify a set number of closest sequences in twisted space. One would then use `KPopTwistDB` a third time in order to add the new twisted spectra just obtained to the database of old twisted spectra.

Many of the workflows described in the previous sections are based on the computation of distances between twisted k-mer spectra, once a twister suitable for classification has been created. This observation naturally suggests the possibility of using our framework to store a large number of twisted sequences into a database and, once the twisted spectrum for a new sample becomes available, identify the sequences in the database that are closest to the new one. As explained in the Methods, this capability is natively provided by `KPopTwistDB`; for instance, it lies at the heart of the k-NN based classifier for *M. tuberculosis* presented above. After identification of nearest neighbours, the new twisted spectra can be added to the database (see Figure 7 for a complete description of the workflow).

Using the same COVID-19 dataset and twister described in the previous section, we tested this idea by finding the 300 closest neighbours of a number of test sequences in a binary `KPop` dataset containing the roughly 600,000 sequences used to train our classifier. The database was 8.4 GB in size — we did not test the possibility of splitting the sequences into smaller files, which might help to achieve a better parallelism. The database took about 2 minutes to load into memory (which, like for the index in in BLAST or other alignment programs, only happens once at the beginning of the workflow, independent of the number of sequences in the query). After that, determining the 300 closest neighbours took about 20 seconds for each sequence. The results were consistent with expectations, with most of the closest sequences identified belonging to the same class as that of the query.

Discussion

We have introduced `KPop`, a novel methodology for the comparative analysis of large datasets of microbial genomes. `KPop` does not require a sequenced sample to be assembled or aligned to a reference genome, and uses the full spectra of k-mers found in a set of sequences or raw reads. By applying an appropriate transformation

(the "twister") to this highly dimensional k-mer space, it becomes possible to compare very large numbers of genomes in a twisted space of reduced dimension. For example, one can classify genomes into groups or find close relatives at scale and with high precision. We simulated datasets emulating the genomic evolution of both *M. tuberculosis* and SARS-CoV-2, accounting for their completely different biologies and epidemiologies, and found in both cases that `KPop` could infer with very high accuracy the lineages to which genomes belonged. We also applied `KPop` to real, very large state-of-the-art datasets for both organisms, and found again that the classification results were in good agreement with labels from previously applied methods. In fact, in this case it seems likely that at least some of the discrepancies were caused by inaccuracies in the classification we started from.

Although many previous methods used k-mers to analyse microbial genomes, we are not aware of any that is directly comparable to `KPop`. For instance, `mash` [28] uses MinHash signatures [45] with cardinality many orders of magnitudes smaller than the full spectra of k-mers. As for the general strategy, this makes the MinHash approach even more scalable than the one we propose here, but at a significant cost to accuracy, for example when classifying genomes into lineages within species [28]. In contrast, `KPop` uses full spectra of (shorter) k-mers; as the results we present here demonstrate, this choice makes it natural for us to directly operate on short unassembled reads and gives excellent accuracy, while at the same time retaining enough scalability for most applications in genomic microbiology. As for classification, however, our workflows hinge upon `KPop`'s capability to explicitly embed k-mer spectra in a reduced-dimensionality space; given that `mash` does not provide such a feature, performing a direct comparison with its results would not be possible.

Compared to other methods that are specifically designed to perform a single task such as establishing relations [28] or searching for specific genetic elements [25], our strategy retains superior flexibility. Thanks to its implementation as a set of modular programs that can easily be combined into a number of different workflows, one can explore a number of scenarios. In particular, more complex workflows than the ones presented here can be easily performed and automated in our framework, without sacrificing scalability — all computer-intensive steps are naturally parallelised whenever possible, so that very large datasets can be analysed within a short wall-clock time frame given enough CPUs and memory.

The fact that `KPop` can efficiently compute relations and distances between genomes makes it a natural starting point for the future development of microbial phylogenetic methods [46]. In the vein of MashTree [47], a straightforward approach to doing so would be to compute the matrix of pairwise distances between all pairs of genomes, and apply a clustering algorithm such as UPGMA [48] or Neighbour-Joining [49]. However, the space of twisted k-mer spectra provides an opportunity to develop more specific methods which are likely to improve the phylogenetic reconstruction, and we intend to explore these ideas in future work.

Methods

Implementation and availability

For the abstract methods described above to be turned into practice, a number of practical steps need to be performed. We have implemented them as a complete

suite of modular programs. Such programs are mostly written in OCaml [50] with a few sections in R [51], and can be freely downloaded as a `conda` package or source code from [32]. They can also be easily compiled and installed from scratch and do not depend on special hardware. The programs performing heavy-duty computation (mainly `KPopCountDB`, `KPopTwist`, and `KPopTwistDB`) have built-in parallelisation support to reduce the overall wall-clock execution time.

`KPop` programs

The workflows used in this paper rely upon four main programs in the `KPop` suite:

1. `KPopCount`. It allows to extract k-mer spectra from FASTA and single- or paired-end FASTQ files. Spectra are output in textual format
2. `KPopCountDB`. It allows to collect k-mer spectra into binary databases, and export the resulting tables as either binary or text files. A number of transformations can be performed on the content of the spectral database
3. `KPopTwist`. Given a database of k-mer spectra, it allows to generate a coordinate transformation that implements an unsupervised dimensional reduction optimised for that database. Such transformation (the "twister" in `KPop` jargon) turns ("twists") a k-mer spectrum into a typically much smaller numerical vector; the transformation can be stored in binary format and applied to other spectra in the future
4. `KPopTwistDB`. It is the Swiss knife of twisted spectra. Among other things, it can be used to twist k-mer spectra if a suitable twister is provided; to collect twisted spectra into databases that can be output as either binary or text files; to compute and summarise distances between twisted spectra.

More details about the methods used in each program follow.

`KPopCount`

`KPopCount` accepts as input a number of single- or paired-end files FASTA/FASTQ files, and counts the k-mers of a specified size present in them. At the moment regular k-mers are used, although support for other schemes (notably gapped k-mers [52]) will be explored in the future. `KPopCount` accepts both DNA and protein sequence. To cope with the lack of directionality of most sequencing protocols, the reverse complement of the input sequences is automatically generated and taken into account when counts are computed for DNA. DNA spectra are subsequently de-duplicated by computing the reverse complement of each k-mer and only keeping the lexicographically smaller of the two — the counts for palindromic k-mers are halved.

It should be noted that the choice of the correct value for k depends on the problem at hand, and cannot be straightforwardly guessed. In particular, it is recommended that saturation analysis is conducted while the analysis workflow is set up — i.e., k should be chosen as the smallest values for which the results become stable. That is because too big a k will likely lead to overfitting, while k-mer spectra generated using a value of k which is too small for the degrees of freedom of the system being studied will not have sufficient resolution to produce good results.

`KPopCountDB`

`KPopCountDB` can be used to perform operations on collections of k-mer spectra.

Table structure

Spectra are stored as columns of a "database"; conceptually, a database can be seen as the binary representation of a matrix, although additional data structures are present in practice to make the matrix extendable and searchable. New columns in the format produced by `KPopCount`, or new metadata in tabular format, can be added to an existing matrix.

In more detail, the general conceptual layout of a database is as follows:

	S_1	$\cdots$	S_{n_S}	L_1	$\cdots$	L_{n_L}
M_1	ms_{11}	$\cdots$	ms_{1n_S}	ml_{11}	$\cdots$	ml_{1n_L}
$\vdots$	$\vdots$	$\cdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$
M_{n_M}	$\mathrm{ms}_{n_M 1}$	$\cdots$	$\mathrm{ms}_{n_M n_S}$	$\mathrm{ml}_{n_M 1}$	$\cdots$	$\mathrm{ml}_{n_M n_L}$
K_1	ks_{11}	$\cdots$	ks_{1n_S}	kl_{11}	$\cdots$	kl_{1n_L}
$\vdots$	$\vdots$	$\cdots$	$\vdots$	$\vdots$	$\cdots$	$\vdots$
K_{n_K}	$ks_{n_K 1}$	$\cdots$	$ks_{n_K n_S}$	$kl_{n_K 1}$	$\cdots$	$kl_{n_K n_L}$

In this example, $ks_{11} \ldots ks_{n_K 1}$ would be the k-mer spectrum corresponding to sample 1 — it would have n_K k-mers.

A `KPop` k-mer database can also be augmented with additional information. In particular, linear combinations of existing columns/data points can be generated and stored as additional columns of the matrix — one such combination being the vector $kl_{11} \ldots kl_{n_K 1}$ in the example above. This feature allows, for instance, to generate and store an average representative spectrum for a set/class of sequences as the linear combination of the spectra of all the sequences belonging to that class. Such linear combinations can then be used as active or passive points in subsequent analyses. Additionally, metadata (typically as text strings, labelled $M_1 \ldots M_{n_M 1}$ in the figure) can be stored as additional rows for each sample, or added at a subsequent time.

The table can also be output as text. All expensive operations, including output, are natively parallelised in order to reduce the overall wall-clock time spent in them.

Tranformations

According to common experience in the field of k-mer statistics, the k-mer distribution follows a curve whereby there is a large number of low-frequency k-mers, typically due to sequencing noise. `KPopCountDB` supports clean-up of the spectra by implementing a number of column-based transformations of the $(ks)(kl)$ lower part of the database.

Different from what happens in detection-oriented methods, our approach does not primarily attempt to reduce the complexity of signatures – while using longer k-mers might allow to find sequences that are distinctive of, and specific to, some particular genome, in our case the information is captured in the overall vector of k-mer frequencies. However, it is desirable to eliminate the noise given by low-coverage k-mers, which is bound to manifest itself in any case due to the finite number of reads used to probe the sample. Depending on the genome, another problem might be given by high-frequency k-mers due to very repetitive regions, which might

excessively capture the attention of linear methods for dimensional reduction used downstream.

In order to tackle the latter, we use a strategy based on power-law or logarithmic mapping of the original k-mer frequencies. As downstream methods used by `KPopTwist` critically rely upon input data being positive, we define transformations as if we were re-mapping the original counts to new integers. Aside from possible overflow problems depending on the actual exponent of the power law, this strategy keeps the transformed vector compatible with the overall framework.

To accomplish that, if m is the maximum value of the vector of counts $\boldsymbol{v}$ and we are applying a logarithmic transformation, we renormalise the vector by a factor k defined as

$$k = \frac{1}{\log\left(\dfrac{m}{m-1}\right)} \approx m$$

which ensures that

$$k\left(\log m - \log(m-1)\right) \geqslant 1,$$

i.e. that m and $m-1$ are represented by different integer numbers. Hence, and imposing that renormalised counts are 0 for frequencies equal to or less than some cutoff c, we redefine $\boldsymbol{v}$ as

$$\boldsymbol{v}' = \max\left(\left\lfloor m \log \frac{\boldsymbol{v}+1}{c+1} \right\rfloor, 0\right). \tag{1}$$

It should be noted that apart from a column-dependent proportionality factor, this transformation coincides with the centered log-ratio (*clr*) transformation that is widely used for compositional data and "binning" programs, e.g. [26] — however, different from *clr*, Equation (1) is always positive and does not suffer from the problem of zero counts. Following the same criterion, given a $0 \leqslant p < 1$ we can define a power-law transformation

$$\boldsymbol{v}' = \max\left(\left\lfloor \frac{m^{1-p}}{p}(\boldsymbol{v}^p - c^p) \right\rfloor, 0\right).$$

having the same property that $v'(m) - v'(m-1) \geqslant 1$.

The transformations introduced so far tend to decrease the distance between samples. It is also possible to transform the data in order to *increase* the distance between samples and specificity — to accomplish that, one has to increase the relative importance of the most frequent k-mers. An obvious way of doing this is by performing a power-law transformation with $p \geqslant 1$. In this case, however, the representability condition becomes important for small rather than large numbers and we have to make sure that $v'(c+1) = 1$, resulting in the transformation

$$\boldsymbol{v}' = \max\left(\left\lfloor \frac{\boldsymbol{v}^p - c^p}{(c+1)^p - c^p} \right\rfloor, 0\right). \tag{2}$$

Depending on the dataset, the last transformation may incur overflow problems when values of p are too large. However, in our implementation we always subsequently convert to floating point and normalise the pseudocounts produced with Equation (2), which alleviates the problem.

Further cleanup would be possible at this stage, for instance by eliminating uninformative k-mers whose fractional frequencies are essentially constant across all samples. However, in general row-based transformation such as this one produce results that depend on the order whereby columns are added to the database, and hence we do support a few of them (mainly the removal of k-mer rows such that all the frequencies are all below a given threshold) but only when the table is exported as text.

`KPopTwist`

`KPopTwist` performs unsupervised dimensional reduction of a set of k-mer spectra and determines dataset-specific ways of transforming them (called "twists" in `KPop` terminology) that can be stored for use on samples obtained at a later time.

`KPopTwist` uses Correspondence Analysis (CA, [53, 54]) as the underlying procedure to identify, define and build a set of dimensions being relevant to the system being considered. CA is a long-standing statistical procedure, and it has been employed to characterise a number of systems in diverse fields ranging from econometrics to biology. Interestingly, when prototyping our method on a wide range of possible use cases we were able to determine that CA seems to consistently outperform other methods for dimensional reduction — PCA (principal component analysis) in particular, but also a number of more recent non-linear techniques. One reason might be due to k-mer spectra being compositional and CA a technique that is well-suited to compositional analysis [55]; another might be the fact that CA has known strong connections with network/clustering analysis [56]. One good property of CA is that, as PCA does, it ranks the resulting dimensions in order of decreasing quantitative importance; in addition, CA is an explainable procedure, in that one can trace back the definition of each relevant dimension to a definition in terms of a specific set of k-mers appearing in the original spectra. Taken together, these two factors would allow the identification of the k-mers that characterise each cluster or class highlighted by the procedure.

Finally, `KPopTwist` also implements a random downsampling of the set of k-mers appearing in the spectra. That is useful to validate the robustness of dimensional reduction with a procedure reminiscent of bootstrapping.

`KPopTwistDB`

`KPopTwistDB` performs a number of operations on twisted vectors. By using it one can:

- Twist k-mer spectra. That requires as input a sequence of spectra as produced by `KPopCount` and a twister transformation as produced by `KPopTwist`, and returns an object of type "twisted" that contains a set of vectors in twisted space
- Compute pairwise distances between two sets of twisted k-mers, and summarise them

- Accumulate twisted vectors or distances into larger databases. This is useful, for instance, to parallelise the creation and processing of datasets containing a large number of samples
- Convert objects of any kind (twisters, twisted, distance) from binary to tabular text form, and vice versa. The tabular representation can then be read into `R` or other programming languages for further downstream analysis.

Distance in twisted space and metric
Given two twisted vectors $\boldsymbol{v} \equiv v_1 \dots v_d$ and $\boldsymbol{w} \equiv w_1 \dots w_d$, d being the number of dimensions of the twisted space, `KPopTwistDB` implements a generalised Minkowski distance D_p between them defined as

$$D_p(\boldsymbol{v}, \boldsymbol{w}) = \left(\sum_{i=1}^{d} |v_i - w_i|^p \right)^{\frac{1}{p}} .$$

CA embeds k-mer spectra into lower-dimensionality spaces. Similar to what happens with PCA, such dimensions are sorted by decreasing "inertia" (a quantity analogous to variance), that we will note as $\boldsymbol{I} \equiv I_1 \dots I_d$. When computing distances between twisted vectors, it then makes sense to define a procedure that takes into account the relevance of each dimension in terms of explained inertia — i.e., we might wish the dimensions to contribute to the overall distance in a way which is somehow proportional to the amount of inertia they carry, in particular when the effective number of degrees of freedom of the system turns out to be smaller than the number of CA dimensions. In order to do so, we introduce the concept of *metric*, i.e. a vector $\boldsymbol{\mu}$ that is a (positive and separable) function of $\boldsymbol{I}$; we would then compute the distance between two twisted vectors $\boldsymbol{v}$ and $\boldsymbol{w}$ as

$$D_p^{\mu}(\boldsymbol{v}, \boldsymbol{w}) = \left(\sum_{i=1}^{d} |\mu(I_i)\ (v_i - w_i)|^p \right)^{\frac{1}{p}} .$$

Many choices would be possible for the function μ; in particular, we focus on three of them, which have been implemented in `KPopTwistDB`:

- *Flat*. This is $\mu(I) = 1$, i.e., no inertia-dependent scaling is performed.
- *Power*. This is $\mu(I) = I^c, c \in \mathbb{R}$, i.e., the inertias are scaled according to some power c. This allows to implement schemes whereby the contribution to distance of each dimension is weighted by the amount of inertia carried by the dimension.
- *Sigmoid*. This is a sigmoid-like function in the range $[0, 1]$ with a number of parameters, the rationale for this choice being that in some situations we would like a limited number of dimensions, up for instance to a given fraction of explained inertia, to be considered when we compute distances. In particular, the user can set the horizontal placement of the 0.5 point as a function of the cumulative value of inertia $C_i := \sum_{j=1}^{i} I_j$, and the tightnesses of the curve at both sides of the 0.5 point can be independently selected. More details and the precise definition of the curve can be found on GitHub [32].

KPop workflows

Based on the modular set of programs just introduced, one can implement arbitrarily complex workflows; some of them are illustrated in Figures 1, 3, 4, and 7. For each of these workflows, some steps can be omitted depending on the nature of the dataset being analysed — for instance, one would not typically perform the read selection of Figure 4 for environmental datasets, datasets obtained from bacterial isolates, or datasets obtained from targeted sequencing. A more detailed list of commands and options can be found online in the KPop GitHub repository [32].

Competing interests
The authors declare that they have no competing interests.

Author's contributions
XD conceived and implemented the validation strategy based on simulated data. PR conceived and implemented the method. Both authors curated data, ran experiments, wrote the manuscript and approved it.

Acknowledgements
This study was partly funded by the National Institute for Health Research (NIHR) Health Protection Research Unit in Genomics and Enabling Data. The authors acknowledge the Research/Scientific Computing teams at The James Hutton Institute and NIAB for providing computational resources and technical support for the UK's Crop Diversity Bioinformatics HPC (BBSRC grant BB/S019669/1), use of which has contributed to the results reported within this paper. The invaluable assistance constantly provided by Dr Iain Milne is especially noted and appreciated.

Author details
[1]School of Life Sciences and Department of Statistics, University of Warwick, Coventry, UK. [2]Biomathematics and Statistics Scotland, The James Hutton Institute, Edinburgh, UK. [3]UK Health Security Agency, London, UK.

References
1. Didelot, X., Bowden, R., Wilson, D.J., Peto, T.E.A., Crook, D.W.: Transforming clinical microbiology with bacterial genome sequencing. Nat. Rev. Genet. **13**, 601–612 (2012). doi:10.1038/nrg3226
2. Loman, N.J., Pallen, M.J.: Twenty years of bacterial genome sequencing. Nat. Rev. Microbiol. **13**, 787–94 (2015). doi:10.1038/nrmicro3565
3. Goodwin, S., McPherson, J.D., McCombie, W.R.: Coming of age: ten years of next-generation sequencing technologies. Nat. Rev. Genet. **17**(6), 333–351 (2016). doi:10.1038/nrg.2016.49
4. Gardy, J.L., Loman, N.J.: Towards a genomics-informed, real-time, global pathogen surveillance system. Nat. Rev. Genet. **19**, 9–20 (2018). doi:10.1038/nrg.2017.88
5. Benson, D.A., Cavanaugh, M., Clark, K., Karsch-Mizrachi, I., Lipman, D.J., Ostell, J., Sayers, E.W.: GenBank. Nucleic Acids Res. **45**(D1), 37–42 (2017). doi:10.1093/nar/gkw1070
6. Jolley, K.A., Bray, J.E., Maiden, M.C.J.: Open-access bacterial population genomics: BIGSdb software, the PubMLST.org website and their applications [version 1; referees: 2 approved]. Wellcome Open Res. **3**(0), 1–20 (2018). doi:10.12688/wellcomeopenres.14826.1
7. Zhou, Z., Alikhan, N.F., Mohamed, K., Fan, Y., Achtman, M.: The EnteroBase user's guide, with case studies on Salmonella transmissions, Yersinia pestis phylogeny, and Escherichia core genomic diversity. Genome Res. **30**(1), 138–152 (2020). doi:10.1101/gr.251678.119
8. Li, H., Durbin, R.: Fast and accurate short read alignment with Burrows?Wheeler transform. Bioinformatics **25**(14), 1754–1760 (2009)
9. Langmead, B., Salzberg, S.L.: Fast gapped-read alignment with Bowtie 2. Nat. Methods **9**(4), 357–359 (2012). doi:10.1038/nmeth.1923
10. Marco-Sola, S., Sammeth, M., Guigó, R., Ribeca, P.: The GEM mapper: fast, accurate and versatile alignment by filtration. Nature methods **9**(12), 1185–1188 (2012)
11. Li, H., Handsaker, B., Wysoker, A., Fennell, T., Ruan, J., Homer, N., Marth, G., Abecasis, G., Durbin, R.: The Sequence Alignment/Map (SAM) Format and SAMtools. Bioinformatics **25**(16), 2078–2079 (2009). doi:10.1093/bioinformatics/btp352
12. Garrison, E., Marth, G.: Haplotype-based variant detection from short-read sequencing. arXiv preprint arXiv:1207.3907 (2012)
13. Zerbino, D.R., Birney, E.: Velvet: Algorithms for de novo short read assembly using de Bruijn graphs. Genome Res. **18**, 821–829 (2008). doi:10.1101/gr.074492.107
14. Bankevich, A., Nurk, S., Antipov, D., Gurevich, A.a., Dvorkin, M., Kulikov, A.S., Lesin, V.M., Nikolenko, S.I., Pham, S., Prjibelski, A.D., Pyshkin, A.V., Sirotkin, A.V., Vyahhi, N., Tesler, G., Alekseyev, M.a., Pevzner, P.a.: SPAdes: A New Genome Assembly Algorithm and Its Applications to Single-Cell Sequencing. J. Comput. Biol. **19**(5), 455–477 (2012). doi:10.1089/cmb.2012.0021
15. Nagarajan, N., Pop, M.: Sequence assembly demystified. Nat. Rev. Genet. **14**(3), 157–67 (2013). doi:10.1038/nrg3367
16. Vernikos, G., Medini, D., Riley, D.R., Tettelin, H.: Ten years of pan-genome analyses. Curr. Opin. Microbiol. **23**, 148–154 (2015). doi:10.1016/j.mib.2014.11.016
17. Darling, A.E., Mau, B., Perna, N.T.: progressiveMauve: Multiple Genome Alignment with Gene Gain, Loss and Rearrangement. PLoS One **5**(6), 11147 (2010). doi:10.1371/journal.pone.0011147
18. Angiuoli, S.V., Salzberg, S.L.: Mugsy: fast multiple alignment of closely related whole genomes. Bioinformatics **27**(3), 334–42 (2011)
19. Maiden, M.C.J., Jansen van Rensburg, M.J., Bray, J.E., Earle, S.G., Ford, S.a., Jolley, K.a., McCarthy, N.D.: MLST revisited: the gene-by-gene approach to bacterial genomics. Nat. Rev. Microbiol. **11**(10), 728–36 (2013). doi:10.1038/nrmicro3093
20. Kurtz, S., Phillippy, A., Delcher, A.L., Smoot, M., Shumway, M., Antonescu, C., Salzberg, S.L.: Versatile and open software for comparing large genomes. Genome Biol. **5**(2), 12 (2004). doi:10.1186/gb-2004-5-2-r12
21. Didelot, X., Darling, A., Falush, D.: Inferring genomic flux in bacteria. Genome Res. **19**(2), 306–317 (2008). doi:10.1101/gr.082263.108
22. Sims, G.E., Kim, S.-H.: Whole-genome phylogeny of Escherichia coli/Shigella group by feature frequency profiles (FFPs). Proc Natl Acad Sci USA **108**(20), 8329–34 (2011). doi:10.1073/pnas.1105168108
23. Sheppard, S.K., Didelot, X., Meric, G., Torralbo, A., Jolley, K.A., Kelly, D.J., Bentley, S.D., Maiden, M.C.J., Parkhill, J., Falush, D.: Genome-wide association study identifies vitamin B5 biosynthesis as a host specificity factor in Campylobacter. Proc Natl Acad Sci USA **110**(29), 11923–7 (2013). doi:10.5061/dryad.28n35.

24. Méric, G., Mageiros, L., Pensar, J., Laabei, M., Yahara, K., Pascoe, B., Kittiwan, N., Tadee, P., Post, V., Lamble, S., Bowden, R., Bray, J.E., Morgenstern, M., Jolley, K.A., Maiden, M.C.J., Feil, E.J., Didelot, X., Miragaia, M., de Lencastre, H., Moriarty, T.F., Rohde, H., Massey, R., Mack, D., Corander, J., Sheppard, S.K.: Disease-associated genotypes of the commensal skin bacterium Staphylococcus epidermidis. Nat. Commun. **9**(1), 5034 (2018). doi:10.1038/s41467-018-07368-7
25. Bradley, P., Gordon, N.C., Walker, T.M., Dunn, L., Heys, S., Huang, B., Earle, S., Pankhurst, L.J., Anson, L., de Cesare, M., Piazza, P., Votintseva, A.A., Golubchik, T., Wilson, D.J., Wyllie, D.H., Diel, R., Niemann, S., Feuerriegel, S., Kohl, T.A., Ismail, N., Omar, S.V., Smith, E.G., Buck, D., McVean, G., Walker, A.S., Peto, T., Crook, D.W., Iqbal, Z.: Rapid antibiotic resistance predictions from genome sequence data for S. aureus and M. tuberculosis. Nat Comm **6**, 018564 (2015). doi:10.1101/018564. arXiv:1011.1669v3
26. Laczny, C.C., Pinel, N., Vlassis, N., Wilmes, P.: Alignment-free visualization of metagenomic data by nonlinear dimension reduction. Scientific reports **4**(1), 1–12 (2014)
27. Pan, S., Zhu, C., Zhao, X.-M., Coelho, L.P.: A deep siamese neural network improves metagenome-assembled genomes in microbiome datasets across different environments. Nature Communications **13**(1), 1–12 (2022)
28. Ondov, B.D., Treangen, T.J., Melsted, P., Mallonee, A.B., Bergman, N.H., Koren, S., Phillippy, A.M.: Mash: Fast genome and metagenome distance estimation using MinHash. Genome Biol. **17**(1), 1–14 (2016). doi:10.1186/s13059-016-0997-x
29. Lees, J.A., Harris, S.R., Tonkin-Hill, G., Gladstone, R.A., Lo, S.W., Weiser, J.N., Corander, J., Bentley, S.D., Croucher, N.J.: Fast and flexible bacterial genomic epidemiology with PopPUNK. Genome Res. **29**(2), 304–316 (2019). doi:10.1101/gr.241455.118
30. Ripley, B.: Classification and regression trees [R package tree version 1.0-42]. Comprehensive R Archive Network (CRAN) (2022). https://cran.r-project.org/web/packages/tree/index.html
31. Liaw, A., Wiener, M.: Classification and Regression by randomForest. R News **2**(3), 18–22 (2002)
32. KPop source code and distributions. https://github.com/PaoloRibeca/KPop. Accessed: 2022-05-30
33. Menardo, F., Duchêne, S., Brites, D., Gagneux, S.: The molecular clock of Mycobacterium tuberculosis. PLoS Pathog. **15**(9), 1–24 (2019). doi:10.1371/journal.ppat.1008067
34. Cole, S., Brosch, R., Parkhill, J., Garnier, T., Churcher, C., Harris, D., Gordon, S.V., Eiglmeier, K., Gas, S., Barry, C.E.r., Others: Deciphering the biology of Mycobacterium tuberculosis from the complete genome sequence. Nature **396**, 537–544 (1998)
35. Huang, W., Li, L., Myers, J.R., Marth, G.T.: ART: A next-generation sequencing read simulator. Bioinformatics **28**(4), 593–594 (2012). doi:10.1093/bioinformatics/btr708
36. The NCBI Short Read Archive. https://www.ncbi.nlm.nih.gov/sra. Accessed: 2022-05-30
37. Phelan, J.E., O'Sullivan, D.M., Machado, D., Ramos, J., Oppong, Y.E.A., Campino, S., O'Grady, J., McNerney, R., Hibberd, M.L., Viveiros, M., Huggett, J.F., Clark, T.G.: Integrating informatics tools and portable sequencing technology for rapid detection of resistance to anti-tuberculous drugs. Genome Med. **11**(1), 1–7 (2019). doi:10.1186/s13073-019-0650-x
38. Brites, D., Loiseau, C., Menardo, F., Borrell, S., Boniotti, M.B., Warren, R., Dippenaar, A., Parsons, S.D.C., Beisel, C., Behr, M.A., Fyfe, J.A., Coscolla, M., Gagneux, S.: A new phylogenetic framework for the animal-adapted mycobacterium tuberculosis complex. Front. Microbiol. **9**(NOV), 1–14 (2018). doi:10.3389/fmicb.2018.02820
39. Trim Galore. https://www.bioinformatics.babraham.ac.uk/projects/trim_galore. Accessed: 2022-05-30
40. Magoč, T., Salzberg, S.L.: FLASH: fast length adjustment of short reads to improve genome assemblies. Bioinformatics **27**(21), 2957–2963 (2011)
41. Wu, F., Zhao, S., Yu, B., Chen, Y.-M., Wang, W., Song, Z.-G., Hu, Y., Tao, Z.-W., Tian, J.-H., Pei, Y.-Y., Yuan, M.-L., Zhang, Y.-L., Dai, F.-H., Liu, Y., Wang, Q.-M., Zheng, J.-J., Xu, L., Holmes, E.C., Zhang, Y.-Z.: A new coronavirus associated with human respiratory disease in China. Nature **579**(7798), 265–269 (2020). doi:10.1038/s41586-020-2008-3
42. van Dorp, L., Acman, M., Richard, D., Shaw, L.P., Ford, C.E., Ormond, L., Owen, C.J., Pang, J., Tan, C.C.S., Boshier, F.A.T., Ortiz, A.T., Balloux, F.: Emergence of genomic diversity and recurrent mutations in SARS-CoV-2. Infect. Genet. Evol. **83**(April), 104351 (2020). doi:10.1016/j.meegid.2020.104351
43. O'Toole, Á., Scher, E., Underwood, A., Jackson, B., Hill, V., McCrone, J.T., Colquhoun, R., Ruis, C., Abu-Dahab, K., Taylor, B., Yeats, C., du Plessis, L., Maloney, D., Medd, N., Attwood, S.W., Aanensen, D.M., Holmes, E.C., Pybus, O.G., Rambaut, A.: Assignment of epidemiological lineages in an emerging pandemic using the pangolin tool. Virus Evol. **7**(2), 1–9 (2021). doi:10.1093/ve/veab064
44. Pangolin Lineage Dataset. https://raw.githubusercontent.com/cov-lineages/pango-designation/master/lineages.csv. Accessed: 2022-05-30
45. Broder, A.Z.: On the resemblance and containment of documents. In: Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No. 97TB100171), pp. 21–29 (1997). IEEE
46. Kapli, P., Yang, Z., Telford, M.J.: Phylogenetic tree building in the genomic age. Nat. Rev. Genet. **21**(July) (2020). doi:10.1038/s41576-020-0233-0
47. Katz, L.S., Griswold, T., Morrison, S.S., Caravas, J.A., Zhang, S., Bakker, H.C.d., Deng, X., Carleton, H.A.: Mashtree: a rapid comparison of whole genome sequence files. Journal of Open Source Software **4**(44), 1762 (2019). doi:10.21105/joss.01762. Accessed 2022-06-07
48. Sokal, R.R.: A statistical method for evaluating systematic relationships. Univ. Kansas, Sci. Bull. **38**, 1409–1438 (1958)
49. Saitou, N., Nei, M.: The neighbor-joining method: a new method for reconstructing phylogenetic trees. Mol. Biol. Evol. **4**(4), 406–25 (1987)
50. The OCaml programming language. https://ocaml.org. Accessed: 2022-05-30
51. R Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria (2021). R Foundation for Statistical Computing. https://www.R-project.org/
52. Ghandi, M., Lee, D., Mohammad-Noori, M., Beer, M.A.: Enhanced regulatory sequence prediction using

gapped k-mer features. PLoS computational biology **10**(7), 1003711 (2014)
53. Benzécri, J.-P.: Correspondence Analysis Handbook. CRC Press LLC, ??? (1992)
54. Nenadic, O., Greenacre, M.: Correspondence analysis in r, with two- and three-dimensional graphics: The ca package. Journal of Statistical Software **20**(3), 1–13 (2007)
55. Quinn, T.P., Erb, I., Gloor, G., Notredame, C., Richardson, M.F., Crowley, T.M.: A field guide for the compositional analysis of any-omics data. GigaScience **8**(9), 107 (2019)
56. van Dam, A., Dekker, M., Morales-Castilla, I., Rodríguez, M., Wichmann, D., Baudena, M.: Correspondence analysis, spectral clustering and graph embedding: applications to ecology and economic complexity. Sci. Rep. **11**(1), 8926 (2021). doi:10.1038/s41598-021-87971-9