Revisiting pangenome openness with *k*-mers

Luca Parmigiani[1,2,3][0000−0002−2139−3259],
Roland Wittler[1,2][0000−0002−2249−9880], and Jens Stoye[1,2][0000−0002−4656−7155]

[1] Faculty of Technology and Center for Biotechnology (CeBiTec), Bielefeld University, Bielefeld, 33615, Germany
[2] Bielefeld Institute for Bioinformatics Infrastructure (BIBI), Bielefeld University, Bielefeld, 33615, Germany
[3] Graduate School "Digital Infrastructure for the Life Sciences" (DILS), Bielefeld University, Bielefeld, 33615, Germany
{luca.parmigiani, roland.wittler, jens.stoye}@uni-bielefeld.de

Abstract. Pangenomics is the study of related genomes collectively, usually from the same species or closely related taxa. Originally, pangenomes were defined for bacterial species. After the concept was extended to eukaryotic genomes, two definitions of pangenome evolved in parallel: the gene-based approach, which defines the pangenome as the union of all genes, and the sequence-based approach, which defines the pangenome as the set of all nonredundant genomic sequences.
Estimating the total size of the pangenome for a given species has been subject of study since the very first mention of pangenomes. Traditionally, this is performed predicting the ratio at which new genes are discovered, referred to as the *openness* of the species.
Here, we abstract each genome as a set of items, which is entirely agnostic of the two approaches (gene-based, sequence-based). Genes are a viable option for items, but also other possibilities are feasible, e.g., genome sequence substrings of fixed length k (k-mers).
In the present study, we investigate the use of k-mers to estimate the openness as an alternative to genes, and compare the results. An efficient implementation is also provided.

Keywords: Pangenomics · Comparative genome analysis.

1 Introduction

With the advent of high throughput sequencing technologies, the number of genomes that we have at our disposal has been steadily increasing. The practice of sequencing and genome analysis, for what regards most of the already sequenced species, usually involves comparing the nucleotide sequences to some reference genome. However, a single reference sequence can hardly account for all the variability present in nature and can not truly represent a whole species [5].

In 2005 the term *pangenome* was used by Tettelin et al. [16] to describe the set of all distinct genes present in a species: either present in all genomes, defined as *core genes*, or present in just some, called *dispensable genes*. One goal was to

predict how many additional genomes should be sequenced to fully describe a bacterial species. While the concept of pangenomes was later extended beyond bacteria, to plants and animals, one of the most outstanding discoveries at the time was that some species possess an *open* pangenome and others a *closed* pangenome.

For an open pangenome, the predicted number of genomes that need to be sequenced in order to get a full picture of the species is large or even undetermined. New distinct genes are found each time a newly sequenced genome is added. A closed pangenome, instead, needs much fewer genomes to portray the species.

In this paper, we define a genome as a set of abstract items. Possible choices of items include, but are not limited to, genes, ORFs or genome intervals of fixed size. The pangenome is defined as the union of these sets. The estimation of the pangenome *openness* requires the computation of the pangenome growth [17]. The pangenome growth is computed starting with one genome and sequentially performing the union with new genomes from the set until all genomes have been considered. It follows that the final total size of the pangenome is independent of the ordering in which we choose the genomes, but the pangenome growth is not. To make it independent from the order, we can compute the pangenome growth for every possible order of the genomes, and then calculate an average pangenome growth. This procedure can be simplified by computing the average over all possible combinations of m genomes, where m goes from one to the total number of genomes.

While it is possible to compute the average pangenome growth (hereafter referred to simply as *pangenome growth*) for a small number of genomes – e.g., with the eight isolates of *Streptococcus agalactiae* used in [16] – it is clear that the computation cannot scale with bigger datasets. Even if a more complex solution has been proposed to minimize the number of combinations to consider [20], the most common and practical solution is to randomly sample multiple permutations of the ordering of genomes. This practice is used in Roary [13] and Pantools [15]. Both tools accept annotated genomes as input and they can output the pangenome openness based on genes.

Comparing genomes using genes as items requires clustering genes into gene families. Different definitions of gene homology determine how similar two genes have to be to end up in the same family, affecting the total size of the pangenome. Hereafter we describe the clustering methods used in Roary and Pantools.

The approach chosen in Roary consists of four steps. First, coding sequences translated from each genome are clustered using CD-HIT [7]. In this step, CD-HIT is applied iteratively with decreasing similarity thresholds. At each iteration, every cluster of genes that contains as many genes as genomes is filtered out. The clustering process of CD-HIT goes as follow. First, sequences are decreasingly sorted based on their length. The first sequence is defined to be the representative of the first gene cluster. Each sequence is then compared to all the previous gene representatives. If it is similar enough to one of these genes, it is added to the cluster, otherwise a new gene cluster is created with this sequence as the

representative. Similarities between sequences are calculated using local alignment. All the proteins that are not filtered out after the last CD-HIT iteration are further processed in the second and third step. In these two steps, an all versus all comparison is performed using BLASTP, and the calculated scores are used to cluster the proteins using MCL [6]. The final step of Roary is to post process the clusters, for example distributing paralogs into separate groups.

Pantools also uses MCL to cluster proteins, but with a different similarity score. Similarities between proteins are computed using local alignment. To reduce the number of comparisons, an alignment is explicitly computed only when the number of shared hexamers (substrings of length six) of the two proteins can guarantee a minimum alignment score. If the similarity score of the local alignment, normalized to be independent of the length, is greater than a second threshold, an edge between the two proteins is added to a similarity graph. For each connected component in the resulting graph, a similarity matrix is computed. Each score is further scaled based on the species of origin of the considered proteins. MCL is then applied to each similarity matrix.

Short genome intervals of constant length k, also known as k-mers, are an established way to express the genomic sequence content. Examples of their use can be found in many different applications, like genome assembly [4], read mapping [19] and metagenomics [18].

In this work, we investigate the use of k-mers as items for determining pangenome openness and compare to the gene-based approach. One of the advantages of using k-mers is that they require only the genome sequence, avoiding several potentially expensive preprocessing steps needed by the gene-based approaches. For example, genome assemblies must be available or computed when using a gene-based approach, while k-mers can be extracted directly from sequencing reads. Moreover, in the absence of genome annotation, each sample must be annotated in silico. Lastly, the comparison between gene sequences requires an adequate definition and computation of homology, and, based on that, a definition and computation of gene families. In summary, the lack of preprocessing steps, coupled with the existence of fast algorithms for handling k-mers [3, 11, 12], make the estimation of the openness based on k-mers fast and straightforward.

The remainder of this paper is organized as follows. Section 2 defines the pangenome growth function and gives an efficient method for its computation. Section 3 contains the empirical results of the k-mer-based and two gene-based approaches (Roary and Pantools) on twelve bacterial pangenomes to compare their predicted open/closed classification. Section 4 concludes the text.

2 Methods

In this section, we abstract the notion of genome to a set of *items*. Items have to be of the form that genomes may share items, while items can also be unique to genomes. Genomes are described as a subset of the universal set of items $\mathcal{K}$ present in a species, or other taxonomic ranks. The set $\mathcal{K}$ contains the union of

all the items appearing in the pangenome plus all the unseen items that may appear in some genomes not included yet. The problem of estimating the size of $\mathcal{K}$ can be reduced to a well studied problem in information theory, where the increment in the total number of items is studied as a function of the number of *objects* (here, genomes) contemplated. This function is known to follow a power law, referred to as Heaps' law [9]. The exponent of the power law defines a species to be either open or closed [17]. To exemplify the concept, we use the same two datasets in the whole section, representing different sides of the openness spectrum: 88 genomes of *Streptococcus pneumoniae*, predicted to be open, and 56 genome of *Yersinia pestis*, predicted to be almost closed.

2.1 Definitions

Let $\mathcal{G} = \{G_1, \ldots, G_n\}$ be a set of genomes, $\mathcal{P}(\mathcal{G})$ the power set representing all possible subsets of $\mathcal{G}$, and $\mathcal{G}_m = \{\mathcal{G}' \in \mathcal{P}(\mathcal{G}) \mid |\mathcal{G}'| = m\}$ the set of subsets of $\mathcal{G}$ of cardinality m. Let M be a binary matrix possessing a column for each genome and a row for each item. If item x is present in genome i then $M_{x,i} = 1$, otherwise $M_{x,i} = 0$. M is usually referred to as *pan-matrix*.

In order to study the openness of a species, we want to examine the size of the union of genomes when considering more and more of the species' genomes. Moreover, to be independent of the order in which genomes are added, we are interested in the average total size $f_{\text{tot}}(m)$ of the union for each possible species subset of size m, where $1 \leq m \leq n$,

$$f_{\text{tot}}(m) = \frac{1}{\binom{n}{m}} \sum_{\mathcal{G}' \in \mathcal{G}_m} \Big| \bigcup_{G \in \mathcal{G}'} G \Big|. \tag{1}$$

For convenience, we also define the average number $f_{\text{new}}(m)$ of new genes that are added when adding the m-th genome as:

$$f_{\text{new}}(m) = \begin{cases} 0 & \text{if } m = 0 \\ f_{\text{tot}}(m) - f_{\text{tot}}(m-1) & \text{otherwise} \end{cases} \tag{2}$$

As stated above, function f_{tot} is known to follow a power law distribution, referred to as Heaps' law [17], defined as $K_1 m^{\gamma}$, where K_1 is a positive constant and $0 \leq \gamma \leq 1$. Since f_{new} is defined as the discrete derivative of f_{tot}, it follows the derivative of the power law: $\frac{d}{dm} K_1 m^{\gamma} = \gamma K_1 m^{\gamma-1} = K_2 m^{-\alpha}$, where $\alpha = 1 - \gamma$ and $K_2 = \gamma K_1$. The estimation of the pangenome's openness is commonly performed by fitting $K_2 m^{-\alpha}$ on f_{new} [17]. When $\alpha \leq 1$, the pangenome is predicted to be open, estimating f_{new} to keep decreasing with time, but never reaching zero. When $\alpha > 1$ instead, the pangenome is considered closed. Figures 1 and 2 show the fitting for some real genome data using k-mers as items: *Streptococcus pneumoniae* in blue, predicted to be open, and *Yersinia pestis* in red, predicted to be almost closed. Note that k-mers were counted in their canonical form, i.e., a k-mer and its reverse complement are considered equivalent. For comparison, Supplementary Figures S1 and S2 show the fitting for the same two species using Roary and Pantools, respectively.

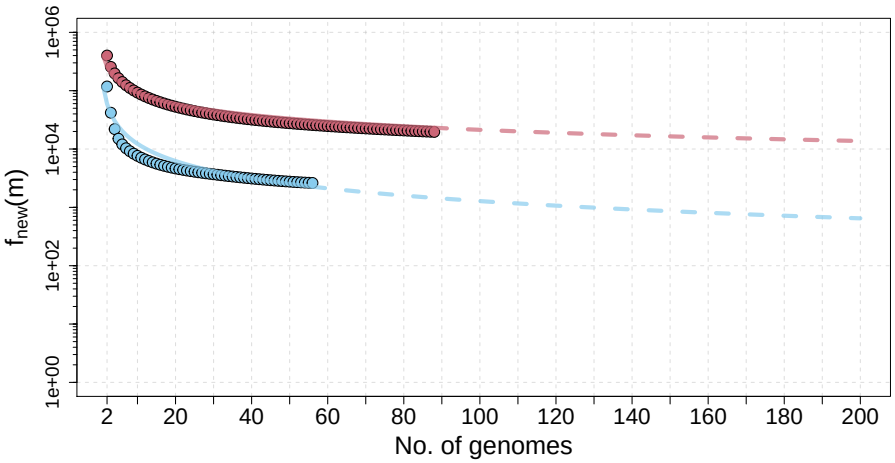

Fig. 1. The dots represent the average number of new canonical k-mers $f_{\text{new}}(m)$ for 88 genomes of *Streptococcus pneumoniae* (red) and of 56 genomes of *Yersinia pestis* (blue). The lines show the fitted Heaps' law functions $K_2 m^{-\alpha}$ with $\alpha = 0.64$ for *Streptococcus pneumoniae* and $\alpha = 0.98$ for *Yersinia pestis*.

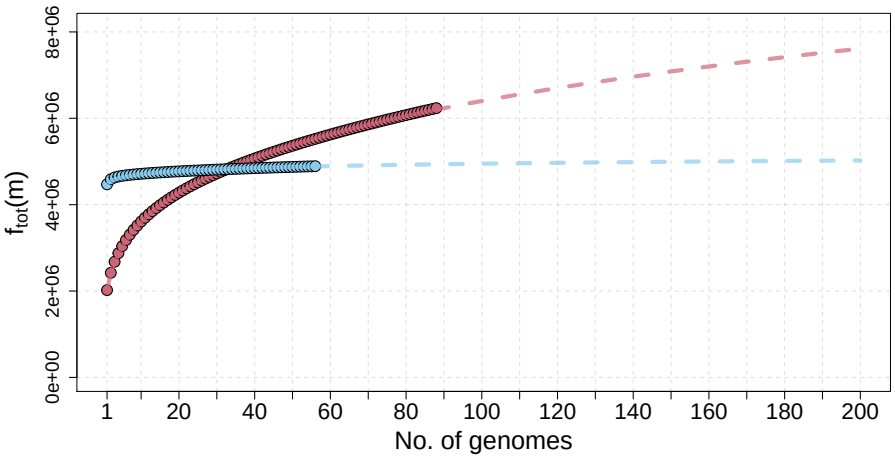

Fig. 2. The dots represent the average number of new canonical k-mers $f_{\text{tot}}(m)$ for 88 genomes of *Streptococcus pneumoniae* (red) and of for 56 genomes of *Yersinia pestis* (blue).

2.2 Obtaining f_{tot} efficiently

The computation of $f_{\text{tot}}(m)$ requires taking the average of $\binom{n}{m}$ values when performed as formulated in Definition 1, making it prohibitive to compute even for relatively small n. The most common solution is to *approximate* $f_{\text{tot}}(m)$ selecting a subset of all the possible permutations of the order of the genomes, as suggested in [17]. Although this works well for gene-based approaches it does not scale for k-mers. The reason is that the number of k-mers that can appear in a species can be two to four orders of magnitude bigger than the number of genes (Supplementary Table S1). This, in turn, slows down drastically the computation of the union of genomes.

Here we describe a more efficient, *exact* way to calculate $f_{\text{tot}}(m)$, without computing $\binom{n}{m}$ subsets for each average. This method has been known for several decades in the field of ecology [10] but it seems to have gone unnoticed in the fields of pangenomics and linguistics [2]. Let $h(i)$ be the number of items that occur in exactly i of the n input genomes, $1 \leq i \leq n$. We consider the contribution of each item of multiplicity i based on the number of subsets $\mathcal{G}' \in \mathcal{G}_m$ in which the item will be counted. This can be done by subtracting from the number of subsets in $\mathcal{G}_m$, $\binom{n}{m}$, the number of subsets that *do not* contain an item of multiplicity i, $\binom{n-i}{m}$. We can then rewrite Equation 1 as:

$$
\begin{aligned}
f_{\text{tot}}(m) &= \frac{1}{\binom{n}{m}} \sum_{\mathcal{G}' \in \mathcal{G}_m} \Big| \bigcup_{G \in \mathcal{G}'} G \Big| = \frac{1}{\binom{n}{m}} \sum_{i=1}^{n} h(i) \Big(\binom{n}{m} - \binom{n-i}{m} \Big) \\
&= \sum_{i=1}^{n} h(i) \Big(1 - \binom{n-i}{m} / \binom{n}{m} \Big) = \sum_{i=1}^{n} h(i) \Big(1 - \frac{(n-i)!(n-m)!}{(n-i-m)!n!} \Big) \\
&= \sum_{i=1}^{n} h(i) \Big(1 - \frac{(n-i)^{\underline{m}}}{n^{\underline{m}}} \Big), \qquad (3)
\end{aligned}
$$

where $n^{\underline{m}}$ is the falling factorial defined as $n^{\underline{m}} = n(n-1)(n-2)\cdots(n-m+1)$. Observe that $n^{\underline{m}} = 0$ if $m > n$, since there is a zero in the product.

2.3 Estimating the pangenome openness

Our procedure for the estimation of the pangenome openness is divided into three steps. First, the values $h(i)$ must be computed or provided. As a matter of computing, they can be extracted directly from the pan-matrix by counting the number of rows that sum up to i. More efficient solutions can be considered based on the item of choice as shown in Section 2.4. In the second step, f_{tot} is calculated according to Equation 3. We can rewrite Equation 3 as $f_{\text{tot}}(m) = \sum_{i=1}^{n} h(i) - \frac{1}{n^{\underline{m}}} \sum_{i=1}^{n-m} h(i)(n-i)^{\underline{m}}$ and compute $(n-i)^{\underline{j+1}} = (n-i-j)(n-i)^{\underline{j}}$, where $(n-i)^{\underline{1}} = (n-i)$. In practice, we can store the values $(n-i)^{\underline{j}}$ reducing the time complexity of the calculation of f_{tot} from $\Theta(n^3)$ to $\Theta(n^2)$, with the same space complexity, $\Theta(n)$. Lastly, f_{new} is computed from f_{tot} using Equation 2,

and the function $K_2 m^{-\alpha}$ is fitted on f_{new}, estimating the parameter α for the pangenome openness.

2.4 Implementation using k-mers as items

We present an implementation of the estimation of the pangenome openness that uses k-mers as items. The code is available as a Gitlab project[4].

For the computation of $h(i)$ we modified the k-mer counting tool yak [3]. Since storing k-mers in the pan-matrix is not efficient in terms of memory, yak uses multiple hash tables for different suffixes of k-mers, reducing access time and enabling parallelization. To prevent counting the same k-mer twice in the same genome, we associate a variable to each k-mer representing the last genome it was found in. Only, if this is different from the genome that is currently read, the counter is incremented. The histograms for *S. pneumoniae* and *Y. pestis* are shown in Figures 3(a) and 3(b), respectively.

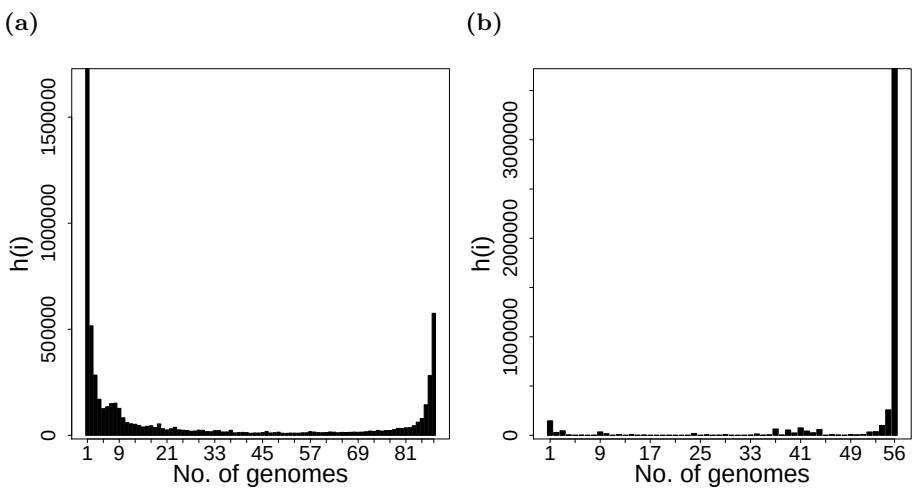

Fig. 3. Histogram $h(i)$ of the number of distinct canonical k-mers in 88 genomes of *Streptococcus pneumoniae* **(a)** and in 56 genomes of *Yersinia pestis* **(b)**.

In the second step, we compute f_{tot} as described in Section 2.3. Due to the large numbers that the summation needs to handle, the implementation of Equation 3 uses the Gnu Multiple Precision Arithmetic Library (GMP) [8]. Repeatedly summing over integer-divisions would propagate multiple approximation errors. For this reason, we can rearrange Equation 3 such that the last operation performed is the division. Since the only operation that would result in a non-integer value is the division by $\binom{n}{m}$, the output is equal to $\lfloor f_{\text{tot}}(m) \rfloor$.

[4] https://gitlab.ub.uni-bielefeld.de/gi/pangrowth

Finally, we compute f_{new} from f_{tot} (Equation 2), and f_{new} can then be used to fit the parameters of Heaps' law. Figure 1 reports a graph showing f_{new} computed on the two datasets with their fitting.

3 Results

We analysed twelve bacterial pangenomes with the two gene-based approaches, Roary [13] and Pantools [15], and with our k-mer-based approach. Genomes of each species were downloaded from NCBI RefSeq with the filter "Assembly level: Complete genome". Annotations have been carried out by Prokka [14] version 1.14.6 with standard parameters. The whole pipeline was wrapped in a Snakemake workflow for reproducibility and parallelization.

3.1 Choice of parameters

While Pantools automatically outputs the value α fitted using the R function *optim*, Roary returns the number of new genes for each m and for each permutation. Since different fitting functions can notably change the value of α, we employed the same fitting as Pantools with *optim* for the output of Roary and the k-mer-based approach.

Pantools was run with default parameters, while the number of permutations carried out by Roary was extended from ten to one thousand, since it is the default value of Pantools. In our method, the only parameter is k for which we used the heuristic proposed in [1]: Given the length of the genome, the value of k is chosen to be the smallest that ensures the probability of a random k-mer to occur in a genome is below 0.001. On the considered datasets, this heuristic suggests $k = 17$ for almost all species, since the scale of the size of bacterial genomes is similar, with the exception of *Buchnera aphidicola*, where $k = 15$ is proposed due to its smaller genome size. Even though inversions and other rearrangements that affect the direction of substrings within the genome can be accounted as variability between genomes, we are more interested in new items (genes or k-mers). For this reason we choose to look at canonical k-mers.

3.2 Overall comparison

Figure 4(a) compares the results of Roary and our k-mer approach for all twelve datasets, showing very similar values of α. The k-mer approach has a slight tendency to predict species to be more open, i.e., yields lower values of α. This is expected for a series of reasons. First, the size of the k-mer space is bigger compared to the gene space, with one single point mutation potentially resulting in k new k-mers. Second, genomes possessing homologous genes but ordered differently will generate new k-mers for each different adjacency. Moreover, silent point mutations do not affect homologous genes but they affect k-mers, i.e., possibly increasing the total number of k-mers but not the total size of the gene-based pangenome.

A comparison of k-mers and Pantools in Figure 4(b) shows that the two methods do not report similar openness. Nevertheless, Figure 4(c) shows that also the two gene-based tools, Pantools and Roary, disagree on the reported openness, indicating that Pantools is the outlier among the three tools considered here.

To estimate how sensible the openness is to the choice of k, we computed α for different values of k (Supplementary Figure S3). Almost every species considered shows an open pangenome when k is larger than or equal to 10 or 11, except *Yersinia pestis* which starts to be open for $k = 15$, with an α very close to one. For $k \geq 17$ the values of α do not change considerably, as shown in Supplementary Figure S4, which compares the values of α for $k = 17$ and $k = 31$.

Overall, results show similar trends in the estimated openness over multiple species between the k-mer-based and gene-based methods. Combined with the absence of outliers, they support the use of k-mers as a reasonable alternative to genes in the estimation of the openness.

4 Conclusion

The use of k-mers to process pangenomes is common practice. In fact, k-mers are almost ubiquitous when processing and analysing genetic sequences. The main focus of this manuscript is the comparison of genes and k-mers when computing the pangenome openness. We showed that the values of α resulting from the k-mer-based approach in bacterial species is similar to the results obtained with a gene-based approach.

One of the motivations for the use of k-mers is the possibility to assess the openness also for non-bacterial species. The concept of openness can be naturally extended to non-bacterial genomes as a measure of the richness of items present in a species. While bacterial genomes are notoriously composed mostly of coding regions, this is not true for eukaryotic genomes, making k-mers more suitable.

To test our method with a non-bacterial dataset, we applied our tool to 100 human genomes (containing only autosomes). We used our server machine equipped with 28 Intel Xeon Processors (2.6 GHz) and 64 GB of RAM. The construction of the histogram took around 1 hour and 15 minutes, while the time for the construction of the pangenome growth from the histogram was around 10 milliseconds. For this dataset we observed $\alpha = 0.789$.

Our experiments show that different gene homology definitions can vary the prediction of the openness. Our k-mer approach does not require gene homology but it still needs a proper choice of k. However, except for small values of k where almost all k-mers appear, the values of α are not very sensible to the choice of k.

Another common pangenome analysis is the identification of the pangenomic core. Equation 3 can be promptly modified to estimate the size of the core. Similarly to the computation of f_{tot}, the core estimation requires taking the average number of shared items. The only difference with f_{tot} is the use of the intersection of each genome instead of the union. We can also relax the definition

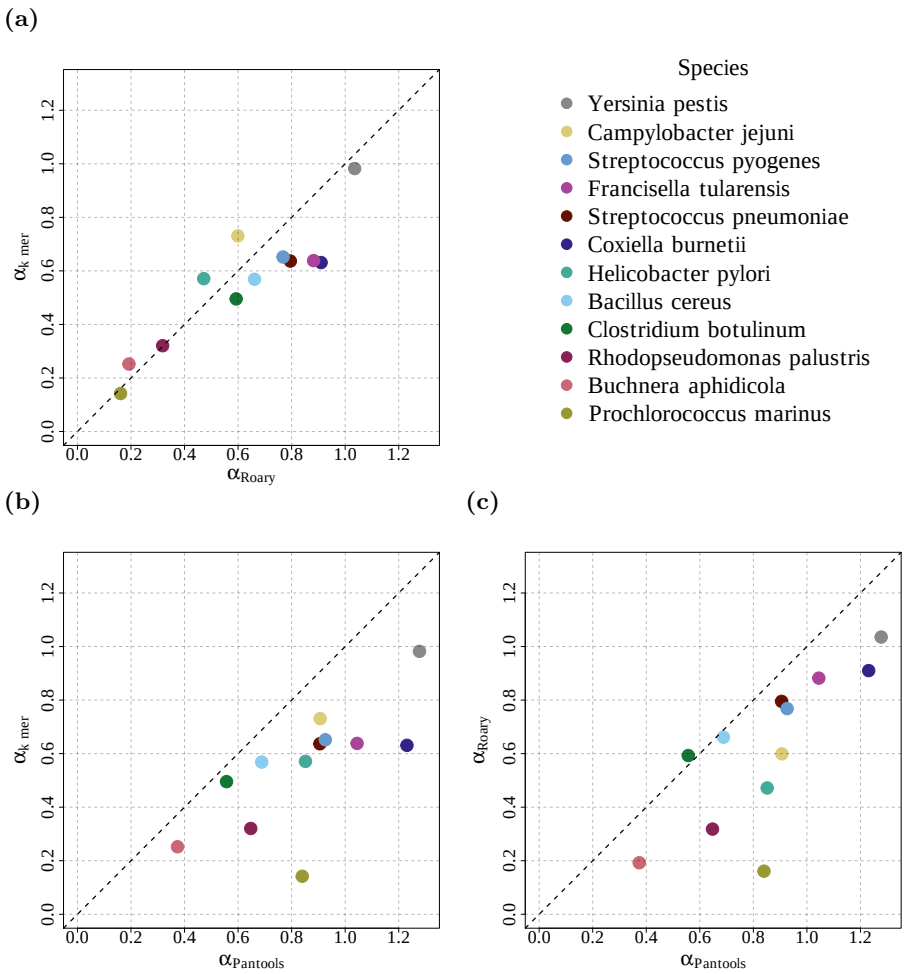

Fig. 4. Comparison of the fitted α values in each species. **(a)** k-mer vs. Roary. **(b)** k-mer vs. Pantools. **(c)** Roary vs. Pantools.

of core items to count the items present in a subset of the set of genomes, e.g., items present in 90% of the genomes.

Acknowledgments

This project received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 956229. It was also supported by the BMBF-funded de.NBI Cloud within the German Network for Bioinformatics Infrastructure (de.NBI) (031A532B, 031A533A, 031A533B, 031A534A, 031A535A, 031A537A, 031A537B, 031A537C, 031A537D, 031A538A).

References

1. S. S. Anari, D. de Ridder, M. E. Schranz, and S. Smit. Efficient inference of homologs in large eukaryotic pan-proteomes. *BMC Bioinformatics*, 19(1), 2018.
2. A. Chacoma and D. H. Zanette. Heaps' law and Heaps functions in tagged texts: evidences of their linguistic relevance. *Royal Society Open Science*, 7(3):200008, 2020.
3. H. Cheng, G. T. Concepcion, X. Feng, H. Zhang, and H. Li. Haplotype-resolved de novo assembly using phased assembly graphs with hifiasm. *Nature Methods*, 18(2):170–175, 2021.
4. P. E. C. Compeau, P. A. Pevzner, and G. Tesler. How to apply de Bruijn graphs to genome assembly. *Nature Biotechnology*, 29(11):987–991, 2011.
5. T. C. P.-G. Consortium. Computational pan-genomics: status, promises and challenges. *Briefings in Bioinformatics*, 19(1):118–135, 10 2016.
6. A. J. Enright. An efficient algorithm for large-scale detection of protein families. *Nucleic Acids Research*, 30(7):1575–1584, 2002.
7. L. Fu, B. Niu, Z. Zhu, S. Wu, and W. Li. CD-HIT: accelerated for clustering the next-generation sequencing data. *Bioinformatics*, 28(23):3150–3152, 2012.
8. T. Granlund and the GMP development team. *GNU MP: The GNU Multiple Precision Arithmetic Library*, 5.0.5 edition, 2012. http://gmplib.org/.
9. H. S. Heaps. *Information Retrieval: Computational and Theoretical Aspects*. Academic Press, Inc., USA, 1978.
10. K. L. Heck, G. van Belle, and D. Simberloff. Explicit calculation of the rarefaction diversity measurement and the determination of sufficient sample size. *Ecology*, 56(6):1459–1461, 1975.
11. M. Kokot, M. Długosz, and S. Deorowicz. KMC 3: counting and manipulating k-mer statistics. *Bioinformatics*, 33(17):2759–2761, 05 2017.
12. G. Marçais and C. Kingsford. A fast, lock-free approach for efficient parallel counting of occurrences of k-mers. *Bioinformatics*, 27(6):764–770, Jan. 2011.
13. A. J. Page, C. A. Cummins, M. Hunt, V. K. Wong, S. Reuter, M. T. Holden, M. Fookes, D. Falush, J. A. Keane, and J. Parkhill. Roary: rapid large-scale prokaryote pan genome analysis. *Bioinformatics*, 31(22):3691–3693, 2015.
14. T. Seemann. Prokka: rapid prokaryotic genome annotation. *Bioinformatics*, 30(14):2068–2069, 2014.

15. S. Sheikhizadeh, M. E. Schranz, M. Akdel, D. de Ridder, and S. Smit. PanTools: representation, storage and exploration of pan-genomic data. *Bioinformatics*, 32(17):i487–i493, 2016.
16. H. Tettelin, V. Masignani, M. J. Cieslewicz, C. Donati, D. Medini, N. L. Ward, S. V. Angiuoli, J. Crabtree, A. L. Jones, A. S. Durkin, R. T. DeBoy, T. M. Davidsen, M. Mora, M. Scarselli, I. M. y Ros, J. D. Peterson, C. R. Hauser, J. P. Sundaram, W. C. Nelson, R. Madupu, L. M. Brinkac, R. J. Dodson, M. J. Rosovitz, S. A. Sullivan, S. C. Daugherty, D. H. Haft, J. Selengut, M. L. Gwinn, L. Zhou, N. Zafar, H. Khouri, D. Radune, G. Dimitrov, K. Watkins, K. J. B. O'Connor, S. Smith, T. R. Utterback, O. White, C. E. Rubens, G. Grandi, L. C. Madoff, D. L. Kasper, J. L. Telford, M. R. Wessels, R. Rappuoli, and C. M. Fraser. Genome analysis of multiple pathogenic isolates of *Streptococcus agalactiae*: Implications for the microbial "pan-genome". *Proceedings of the National Academy of Sciences of the United States of America*, 102(39):13950–13955, 2005.
17. H. Tettelin, D. Riley, C. Cattuto, and D. Medini. Comparative genomics: the bacterial pan-genome. *Current Opinion in Microbiology*, 11(5):472–477, 2008.
18. D. E. Wood and S. L. Salzberg. Kraken: ultrafast metagenomic sequence classification using exact alignments. *Genome Biology*, 15(3):R46, 2014.
19. H. Xin, D. Lee, F. Hormozdiari, S. Yedkar, O. Mutlu, and C. Alkan. Accelerating read mapping with fasthash. *BMC Genomics*, 14(1):S13, Jan 2013.
20. Y. Zhao, X. Jia, J. Yang, Y. Ling, Z. Zhang, J. Yu, J. Wu, and J. Xiao. PanGP: A tool for quickly analyzing bacterial pan-genome profile. *Bioinformatics*, 30(9):1297–1299, 2014.