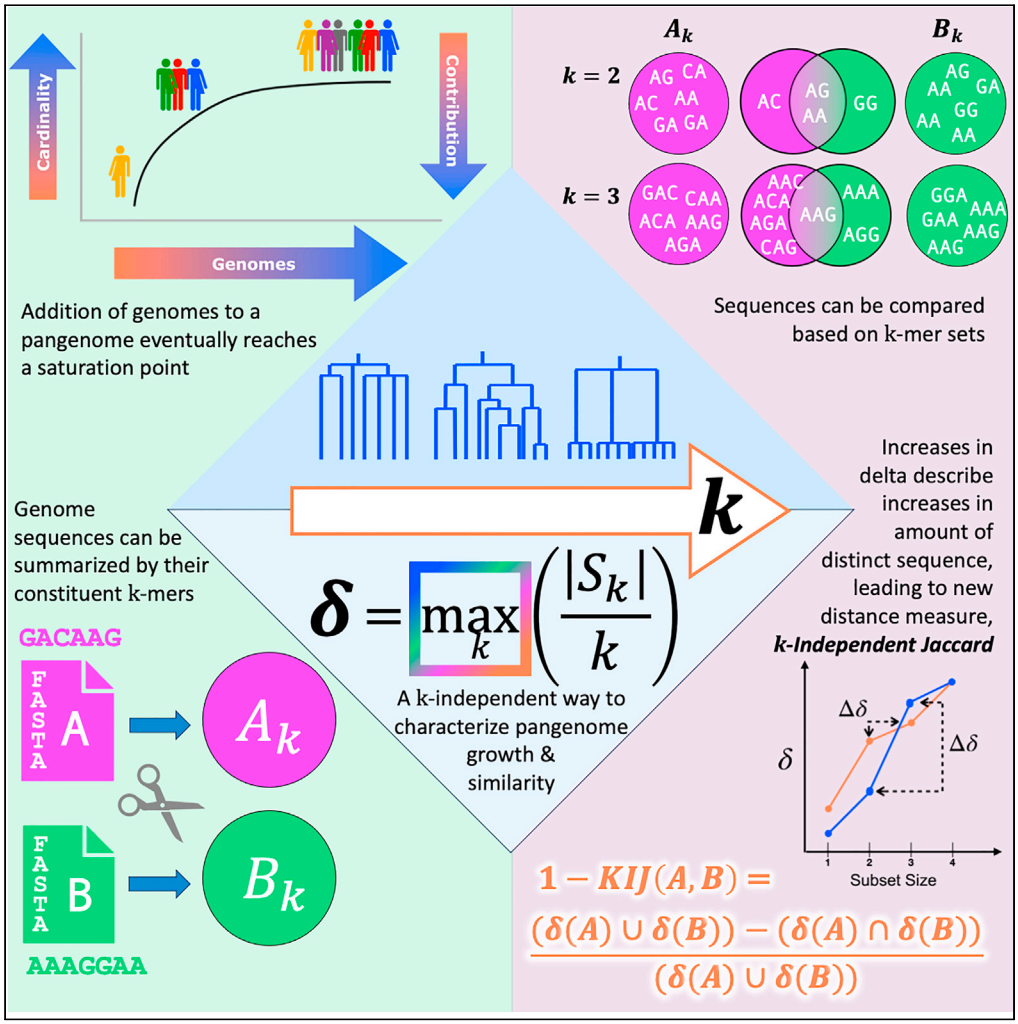
iScience

CellPress
OPEN ACCESS

Article

DandD: Efficient measurement of sequence growth and similarity



Jessica K. Bonnie, Omar Y. Ahmed, Ben Langmead

jbonnie1@jhu.edu (J.K.B.)
langmea@cs.jhu.edu (B.L.)

Highlights

DandD uses a measure from data compression to measure amount of distinct sequence

The measure can be used to characterize pangenome "openness"

The measure can also be used as a length-independent measure of sequence similarity

Bonnie et al., iScience 27, 109054
March 15, 2024
https://doi.org/10.1016/j.isci.2024.109054

iScience

CellPress
OPEN ACCESS

Article

DandD: Efficient measurement of sequence growth and similarity

Jessica K. Bonnie,[1,*] Omar Y. Ahmed,[1] and Ben Langmead[1,2,*]

SUMMARY

Genome assembly databases are growing rapidly. The redundancy of sequence content between a new assembly and previous ones is neither conceptually nor algorithmically easy to measure. We introduce pertinent methods and DandD, a tool addressing how much new sequence is gained when a sequence collection grows. DandD can describe how much structural variation is discovered in each new human genome assembly and when discoveries will level off in the future. DandD uses a measure called δ ("delta"), developed initially for data compression and chiefly dependent on *k*-mer counts. DandD rapidly estimates δ using genomic sketches. We propose δ as an alternative to *k-mer*-specific cardinalities when computing the Jaccard coefficient, thereby avoiding the pitfalls of a poor choice of *k*. We demonstrate the utility of DandD's functions for estimating δ, characterizing the rate of pangenome growth, and computing all-pairs similarities using *k*-independent Jaccard.

INTRODUCTION

Pangenomes are growing as more high-quality assemblies are produced. Once a sufficient number of assemblies have been added, a pangenome can reach a point of diminishing returns, where each new genome contributes little novel sequence to the collection.[1] Measuring the amount of new sequence per assembly, though, is neither conceptually nor algorithmically straightforward. By surveying studies that computed the average amount of novel sequence per individual in human genome assemblies, Sherman et al.[2] found that estimates varied from 0.2 Mbp to 14 Mbp. This wide range in values was attributable to reasonable investigator choices such as selection of alignment parameters or criteria for identifying contigs with novel sequence. In short, the parameter choices led to a wide range of answers, making the end results difficult to compare.

Alignment-free, i.e., *k*-mer based, approaches offer an alternative. Such methods have been used to determine if a pangenome has reached the point of being "closed," with open/closed status determined by fitting a Heaps'-Law model to an empirical *k*-mer growth function.[1] While avoiding many of the parameter-selection pitfalls of alignment-based methods, *k*-mer based approaches, as the name suggests, still require an initial choice of substring (*k*-mer) length, with subsequent measurements dependent on this selection.

We present a new parameter-free method and tool for measuring the amount of sequence in a pangenome based on ideas from string compression. We use a quantity "delta" (δ) that measures compressibility of a repetitive string.[3] Other quantities have been proposed for this purpose, including the number of runs in the Burrows-Wheeler Transform (r),[4] number of phrases in the Lempel-Ziv parse (z),[5] and the size of the string attractor (γ).[6] All these measures have distinct algorithms and interpretations, but δ is known to have advantageous bounds compared to the others. For instance, $\delta \leq \gamma$ for all strings.[3]

Computational difficulty among measures quantifying novel sequence varies widely. z and r require computing a Lempel-Ziv parse or Burrows-Wheeler Transform, respectively, across the entire input string. Computing γ is nondeterministic polynomial-time (NP) complete. Computing δ, however, requires little more than a single pass over the input to count *k*-mers for various values of *k* (Figure 1). Available tools like KMC[7] can do this efficiently.

Besides being a useful measure of repetitiveness, δ is also remarkably easy to estimate. This is true not only when estimating δ over a given sequence collection, but also when estimating over *unions* of sequences, as is needed to assess pangenome saturation. Estimating δ for sequences and their unions reduces to the problem of estimating set cardinality. Our main insight is that estimating cardinalities over large sequences and their unions is highly efficient using sketches such as MinHash[8,9] or HyperLogLog.[10,11]

Additionally, we propose a measure called the *k*-independent Jaccard (KIJ), as an alternative to the Jaccard coefficient. KIJ avoids the risks of preselecting *k*-mer length by using δ. Having a principled way to measure similarity without a pre-determined *k* is critically important since poor choices of *k* can lead to incorrect conclusions downstream, as we show in the context of phylogenetic reconstruction.

[1]Department of Computer Science, Johns Hopkins University, Baltimore, MD, USA
[2]Lead contact
*Correspondence: jbonnie1@jhu.edu (J.K.B.), langmea@cs.jhu.edu (B.L.)
https://doi.org/10.1016/j.isci.2024.109054

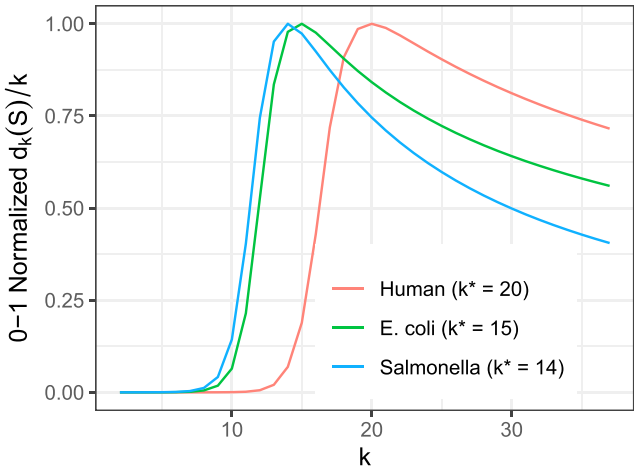


Figure 1. Different pangenomes find their maxima at different values of *k*

Vertical axis shows $\frac{d_k(S)}{k}$ standardized to between 0 and 1 per pangenome. Each pangenome is comprised of multiple distinct genome sequences: *H. sapiens* ($n = 12$), *E.coli* ($n = 10$), and *S. enterica* ($n = 10$).

We describe the algorithms and data structures implemented in the new DandD software tool, which can compute and estimate δ in a variety of scenarios relevant to genomics and pangenomics. For example, we demonstrate that (a) DandD can efficiently and accurately estimate δ using genomic sketches, (b) DandD can measure how much new sequence is in each new assembled human genome in a pangenome, including when the input consists of assemblies without common coordinates, and finally (c) DandD can be used to compute KIJ, yielding phylogenetic trees that closely match known truth.

DandD is open source software available at: https://github.com/jessicabonnie/dandd.

RESULTS

Behavior of δ, r, and z in practice

We began with an empirical study of the relationship between *z*, *r*, and δ using real genome sequence data. We collected 50 *Salmonella enterica* genomes from Refseq, putting them in an arbitrary order. We computed *z*, *r*, and δ for sets of these genomes, starting with the first and cumulatively adding one at a time. In the case of *z*, we used a combination of the newscanNT.x tool from Big-BWT[12] and the lz_77 tool from PFP_LZ77.[13] To compute *r*, we used the pfbwt-f64 tool.[12] For δ, we used DandD in its –exact mode, which in turn uses KMC3.[7] For *r* and *z*, both the genomes and their reverse complements were added to the set at the same time. For δ, we accomplished this through *k*-mer canonicalization. Results are shown in Figure 2. The three measures were normalized by first subtracting the minimum then dividing by the maximum, to obtain values ranging from 0 to 1. There is an obvious, strong relationship between the measures, with no two of the normalized measures differing by more than ± 0.0273 at any point. Figure S1 also shows the wall clock time required by these methods for computing *z*, *r*, and δ, with the KMC3-based method for computing δ being the fastest.

Efficient cardinality estimation with dashing

We sought to compare the computational performance and accuracy of DandD's two modes, the –exact mode ("exact") and the sketch-based mode ("approximate"). We ran DandD in both modes on three datasets: 10 *E. coli* genomes, 10 salmonella genomes, and 12 human genomes. We performed all the computations for a single value of *k*. These methods are being assessed for performance in computing (or estimating) the cardinatlity $d_k(S)$, not δ. Given that computing δ means repeatedly performing this process for several values of *k*, this provides sufficient basis for comparing the methods. For each dataset and method, we performed three computations. First, we used the selected method to build a *k*-mer database (for "exact") or sketch (for "approximate") over each input genome. We used the selected method to perform a series of cumulative unions, starting from a single database/sketch and combining them in successive steps until all genomes are included. Finally, we performed a single global union of all of the individual databases/sketches. In all cases, we used /usr/bin/time -v to measure the time and the maximum memory usage. In nearly all cases, the "approximate" method is faster than the "exact" method (Table 1). This is particularly true for the union steps for the human genome inputs, where the approximate method is as much as three orders of magnitude faster than the exact method. The "approximate" method's peak memory footprint is consistently smaller than the exact method's, sometimes by two orders of magnitude. Further, for every experiment described in Table 1, we used Dashing to estimate the cardinality from the sketch produced and compared this to the true cardinality as computed by KMC3. We computed the relative error of the Dashing estimate as $|D - K|/K$, where D is Dashing's estimate and K is KMC3's exact count. Overall, the mean relative error was 6.537×10^{-4}. The maximum observed relative error was 1.32×10^{-3}.

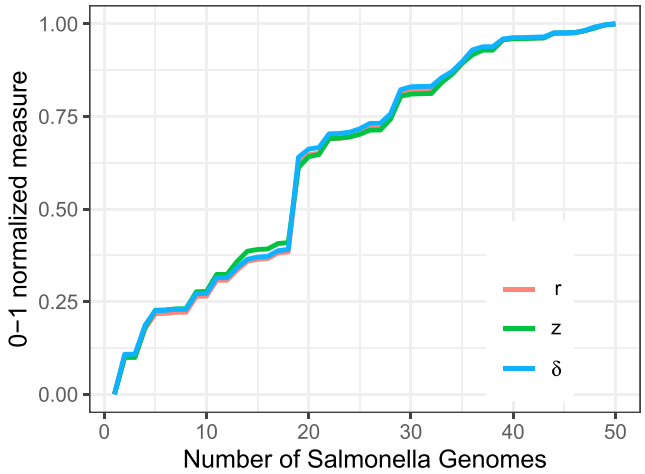


Figure 2. Normalized compression metrics δ, *r*, and *z* reflect the same pattern of growth with the inclusion of each additional assembly from an arbitary ordering of 50 Salmonella genomes

Normalization was achieved by scaling to the maximum value for each measure.

Using DandD to characterize sublinearity and openness

We ran the DandD progressive command on a set of 34 human haplotypes taken from the Human Genome Structural Variation Consortium (HGSVC2) project.[14,15] The haplotypes were chosen to all have an X chromosome, to avoid large increments in δ due only to the addition of the Y chromosome. HGSVC2 is organized into variant call format (VCF) files for each distinct variant type: single nucleotide variants (SNVs), small insertions and deletions (Indels), or structural variants (SVs). We repeated our experiment across different subsets of variant categories. For instance, to create the FASTA sequences used as input to our "SNV only" experiment, we used bcftools consensus[16] to create 34 haplotype-specific FASTA files, taking rows from the VCF file containing SNVs for 17 individuals annotated as female. For the "SNVs + indels" experiment, we did the same but taking rows from both the SNV and Indel VCF files. For the "SNVs + SVs + indels" experiment, we did the same but taking rows from all three VCFs. In all cases, the individual haplotype FASTAs were then provided to DandD progressive. Since it is impractical to attempt all orderings of 34 haplotypes, we used DandD's –norder 120 option to randomly generate a series of 120 possible permutations. We performed a corresponding experiment for 56 haplotypes from the Human Pangenome Reference Consortium (HPRC),[17] taken from 28 individuals annotated as female. In this case, the FASTA inputs to DandD were the phased assemblies provided by the HPRC. There is no accompanying VCF file describing variants or variant types, giving us no way to stratify by variant type as we did for HGSVC2. However, DandD is applicable regardless, since it simply accepts any FASTA inputs. We used DandD's –norder 90 option to randomly generate a series of 90 possible permutations, 5 of which are shown in Figure 3B. In all cases, we computed a Heaps'-Law fit to the mean δ values and recorded the α openness statistic[1] for each. Fit values of α are reported in the Figure 3 legends. As seen in Figure 3, all experiments showed sublinear growth in mean δ. For HGSVC2, we observed that "SNVs + SVs + Indels" had the highest δ overall, with "SNVs + Indels" having slightly lower values, and "SNVs only" having still lower values. This is expected, since the inclusion of each additional variant class should lead to new distinct sequence. The Heaps'-Law α was approximately the same for all three HGSVC2 variant subsets, ≈ 0.94 in all cases. Thus, all variant classes lead to the same conclusion about "openness" of the HGSVC2 pangenome, i.e., $\alpha \leq 1$ indicates it is open. The HPRC data also showed sublinear growth, and a range of δ's quite similar to those observed for HGSVC2. The Heaps'-Law $\alpha = 0.873$, again indicating an open pangenome. The fact that α is lower for HPRC may indicate that the *de novo* assemblies from long reads give access to a wider array of genetic variants, which in turn requires more genomes to saturate the pangenome. However, this is hard to disentangle from the effects of sequencing errors, which can be counted (spuriously) as novel sequence. Removing the effect of sequencing errors is an important problem, which we return to in the Discussion.

A *k*-independent approach to openness

We evaluated how DandD's k-independent approach can improve existing approaches for estimating pangenome openness. Past work characterized openness by compiling *k*-mer cardinality statistics across increasing subsets of genomes, then fitting a Heaps'-Law model to the *k*-mer cardinality growth function.[1] The fit value of the Heaps'-law α serves as an openness statistic, with $\alpha \leq 1$ indicating an open pangenome and $\alpha > 1$ indicating a closed pangenome. We hypothesized that DandD's *k*-independent approach could provide a robust basis for estimating openness without having to choose a particular *k* ahead of time. We used benchmarking datasets from the AFproject.[18] These were created by the AFproject to evaluate alignment-free clustering methods. In particular, we used the 25 fish mitochondrial genomes and the 29 *E. coli* genomes from that benchmark. After building initial *k*-mer databases via DandD's tree command in –exact mode, we used the progressive command with –ksweep –mink 2 –maxk 55 -n 20 to compute the number of distinct *k*-mers for *k* ranging from 2 to 55. These computations used KMC3 to count *k*-mers and used 20 random orderings to estimate the average increase in cardinality with each added genome. We then fit a Heaps'-Law function and obtained the fit value for the α parameter. We observed that for very small values of *k*, α was greater than 1, indicating

Table 1. Computational efficiency of KMC3 versus Dashing for preprocessing and unioning

	Exact (KMC3)		Approximate (Dashing)	
Task	Time (seconds)	Peak Mem (GB)	Time (seconds)	Peak Mem (GB)
E. coli (n = 10, k = 15)				
Preprocess 10 inputs (mean)	0.626	0.100	0.353	0.0138
9 cumulative unions (total)	4.39	0.381	2.28	6.58e-3
1 global union (total)	2.26	0.403	0.270	0.0129
Salmonella (n = 10, k = 14)				
Preprocess 10 inputs (mean)	0.643	0.100	0.340	0.0138
9 cumulative unions (total)	3.39	0.388	2.13	6.59e-3
1 global union (total)	1.85	0.389	0.28	0.0129
Human (n = 10, k = 20)				
Preprocess 10 inputs (mean)	40.8	11.7	44.2	0.127
9 cumulative unions (total)	1,750	8.14	2.16	6.59e-3
1 global union (total)	829	16.6	0.28	0.0129

For KMC3, preprocessing consists of building *k*-mer count databases. For Dashing, it consists of building genomic sketches. We analyzed three collections of genome assemblies: E. coli (n = 10, k = 15), salmonella (n = 10, k = 14), and human (n = 10, k = 20). For simplicity, we chose a single *k* for each which was appropriate to the species. We measured the time and memory (resident set size) required to preprocess the 10 inputs on average, reported in rows labeled "Preprocess 10 inputs." We chose a random ordering of the genomes and measured the resources required to perform a series of unions, each adding one additional genome to the union ("9 cumulative unions"). We also measured resources required to union all preprocessed datasets at once ("1 global union").

a closed pangenome (Figure 4). This was expected since a very small value of k leads to rapid saturation of the space of possible k-mers as genomes are added. For increasing values of k, α decreased and eventually stabilized to a value less than 1, giving a more robust and convincing indicator of an open pangenome. When we considered the value of α obtained using δ (triangles in Figure 4), it was also less than 1, consistent with the fact that α stabilizes to a value less than 1 when computed with most informative values of k. In short, computing α using DandD does not require selecting a value of k but yields an α consistent with the stable part of the function given by a k-specific method.

Evaluating *k*-Independent Jaccard

To evaluate the utility of the KIJ measure, we again used benchmarking datasets from the AFproject,[18] particularly the 25 fish mitochondrial genomes and the 29 *E. coli* genomes. For KIJ, we computed all pairwise distances between the sequences in the dataset. Since KIJ measures similarity, we ultimately report the distance, $1 - \text{KIJ}$. From this point, we proceeded with the steps of the AFproject protocol, which constructs a tree from the pairwise-distances, then compares that truth to a curated tree. The ultimate result is a normalized version of the Robinson-Foulds distance (nRF), which measures the degree of structural difference between two trees having the same set of sequences at the leaves. A low nRF indicates that the distances provided reflected the true phylogenetic relationships between the sequences. Having done this for $1 - \text{KIJ}$ distances, we repeated the process for distances based on k-specific Jaccard coefficients for a range of ks. We computed J_k and reported a matrix of pairwise $1 - J_k$ distances for $k = 2$ to 59. nRFs obtained for each of these are shown in Figure 5. We observed that the accuracy of the tree predicted by the $1 - J_k$ distance depends on the choice of k. A too-small value of k leads to non-specific distances that cannot distinguish the phylogenetic relationships, leading to high nRF toward the left-hand side of the plots in Figure 5. A too-large value of k can deplete the number of common k-mers between related sequences in a way that obscures their relationship, as seen toward the right-hand side of the "25 fish mitochondria" plot in Figure 5, where nRF climbs after k grows past 25. The $1 - \text{KIJ}$ distance, on the other hand, strikes a balance between these extremes. In both cases, the $1 - \text{KIJ}$ distance achieves minimal nRF compared to all of the $1 - J_k$ distances.

DISCUSSION

As sequencing technology improves, new genome assemblies will arrive more quickly. It will be increasingly important to identify when a collection of genomes has reached a point of saturation, i.e., when it represents a taxonomic grouping in a complete fashion without excess accumulation of rare variation. Since pangenomes can be seen as collections of repetitive strings, theory concerning compressibility provides tools well suited to this problem. Building on the δ measure, DandD provides an efficient and interpretable way to measure the growth of pangenomes and compare large sequence collections. δ has theoretical advantages but is also remarkably easy to compute. Genomic sketches make δ particularly easy to estimate over pangenomes and their unions. Further, δ provides a parameter-free way of quantifying the amount of distinct sequence in a pangenome, sidestepping any dependence on parameters. The methods underlying DandD treat pangenomes as sets, with KIJ providing a quantification similar to the Jaccard coefficient between sets. However, another way to represent and sketch pangenomes would be as multisets, where each item (i.e., k-mer) has an associated count; e.g., the number of times it occurs in the

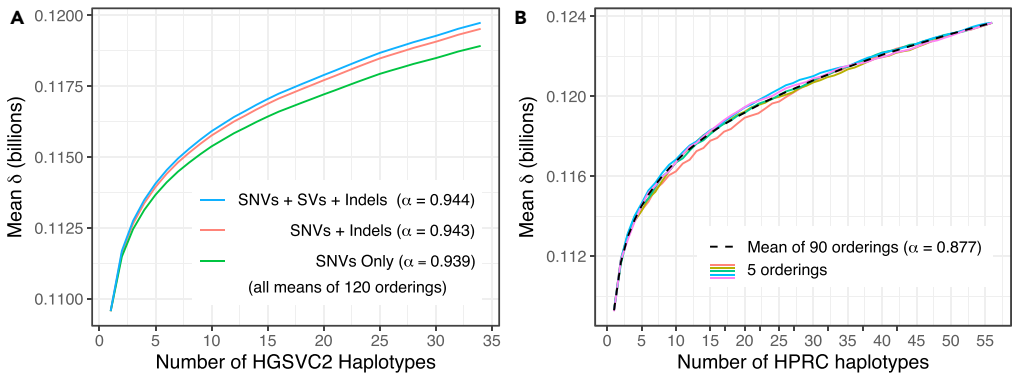


Figure 3. Empirical growth of δ for human haplotypes from the HGSVC2 project (A) and the HPRC (B)

For (A), lines show mean value of δ for each genome count across all 120 orderings. Colors denote how the variants were subsetted before constructing the FASTA files given to DandD. E.g. the blue line corresponds to genomes edited to include SNVs, small indels and structural variants, whereas the green line corresponds to genomes edited to include only SNVs. For (B), the dotted line shows the mean value of δ at each genome count across all 90 orderings. The colored lines show the particular values of δ from a random subset of 5 orderings.

pangenome. Genomic sketches like SetSketch can accommodate counts, and Dashing 2[19] can compute probability-weighted version of the Jaccard coefficient. In the future, it will be important to evaluate whether consideration of counts can be naturally combined with these compressibility measures. Past work suggests other ways to select a scenario-specific "best" k, for use with assembly or other algorithms. KmerGenie[20] uses k-mer count distributions to select the k that yields the most distinct high-count k-mers. While DandD's method represents a simple and mathematically grounded way to choose k, some scenarios might require more information than DandD uses, such as the assembly scenario targeted by KmerGenie. Besides δ, other measures have been proposed that take the form of having cardinality in the numerator and some normalization factor or "penalty" in the denominator that is a function of k. An example is sequence space coverage (SSC) and its normalized version (NSSC) used by Bussi et al.[21] In that case, the denominator includes 4^k, with modifications to account for guanine-cytosine (GC) bias and other factors. Importantly, the numerator is still the cardinality, and the denominator is still a simple function of k and n (total input sequence length). The methods proposed here, including the sketching methods, are equally applicable in that context, and will be applicable for any measure that includes cardinality as a key term. It should be possible to convert the $1 - \mathrm{KIJ}$ distance measure (discussed in "Evaluating KIJ" in Results) into a Mash distance,[9] though with the additional complication that KIJ is a function of three separate δ measures, $\delta(A)$, $\delta(B)$, and $\delta(A \cup B)$. These may use different underlying choices for k^*, creating ambiguity in how k should be specified in the Mash distance formula. In the future, it will be important to study how to handle multiple distinct k's in the Mash distance formula, and to evaluate how Mash distances derived from KIJ perform relative to those derived from J_k.

Limitations of the study

The progressive union function of DandD computes unions over several random orderings of the FASTA inputs. This is in contrast to other methods for which closed-form expressions are known.[1] The most accurate computations of δ will use exact k-mer counts. While DandD can employ KMC3 to obtain precise counts, building and querying a KMC database is substantially less efficient than sketching with Dashing. The

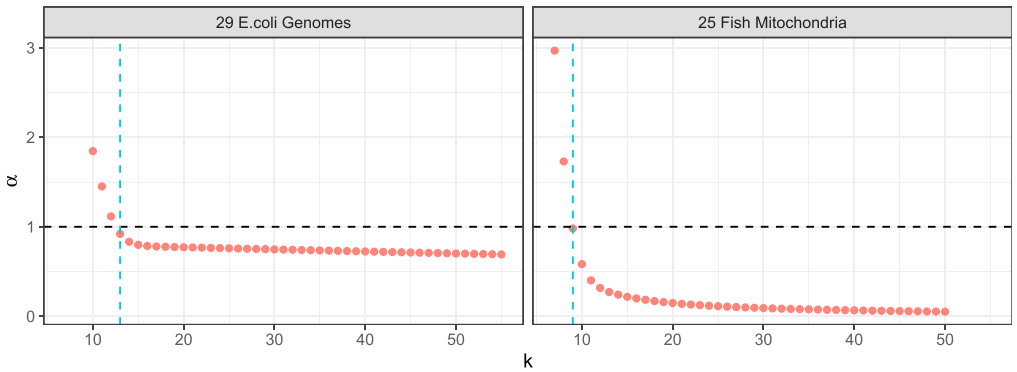


Figure 4. Fit values for the Heaps'-law α as a function of k (red dots) and the value obtained by using the k selected by δ (dotted blue line)

The transition from "open" ($\alpha > 1$) to "closed" ($\alpha > 1$) values occurs at different threshold values of k for fish mitochondria ($k = 9$) and *E. coli* ($k = 13$). The values of k^* which produce δ are slightly larger than the thresholds for both *E. coli* ($k^* = 14$) and fish mitochondria ($k^* = 10$).

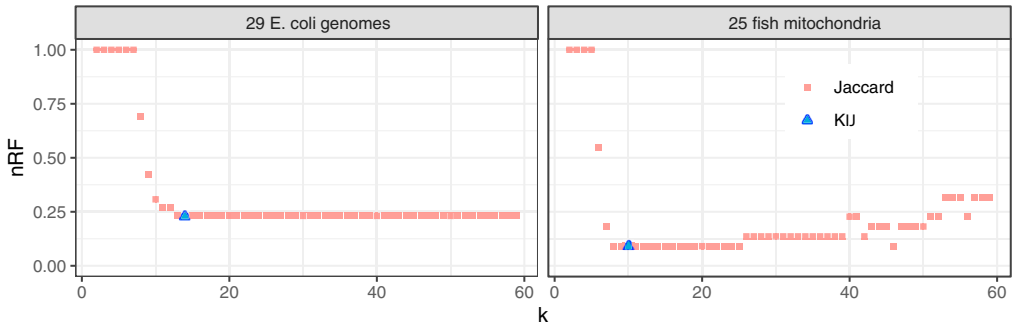


Figure 5. Clustering based on KIJ distances achieves low normalized Robinson-Foulds (nRF) distance with respect to the true phylogeny compared to clusterings based on the typical Jaccard coefficient

notable increase time and memory usage reduces the convenience of DandD's exact mode. The δ measure emerged from compression theory and lacks any intrinsic connection to population genetic measures like Nei's genetic distance. Though δ has a relationship to Mash distance, the need to specify a single value of *k* in the Mash distance formula is not compatible with δ's ability to choose different values of *k* for the marginal and joint components of the formula. Finally, use of δ may involve an implicit assumption that all *k*-mers are equally likely. This contrasts with biological reality, where some *k*-mers may be more likely due to the overall GC content of the genomes being compared. Other methods such as the normalized sequence space coverage (NSSC)[21] measure may be more useful in scenarios where such biological variables have a strong influence.

STAR★METHODS

Detailed methods are provided in the online version of this paper and include the following:

- KEY RESOURCES TABLE
- RESOURCE AVAILABILITY
  - Lead contact
  - Materials availability
  - Data and code availability
- METHOD DETAILS
  - The delta compressibility measure
  - Estimating cardinality
  - Estimating delta
  - Characterizing sublinear growth
  - *k*-independent Jaccard
  - Caching and lazy evaluation of sketches
  - Exact mode

SUPPLEMENTAL INFORMATION

Supplemental information can be found online at https://doi.org/10.1016/j.isci.2024.109054.

ACKNOWLEDGMENTS

We thank Aaron Quinlan, Dominik Kempa, and Yun William Yu for helpful discussions. O.Y.A. and B.L. were supported by NIH /NHGRI grant R01HG011392 to B.L. J.K.B. and B.L. were supported by NIH /NIGMS grant R35GM139602 to B.L. and NIH /NHGRI grant R01HG012252. O.Y.A. was also supported by NIH /NIGMS training grant T32GM119998. This work was carried out at the Advanced Research Computing at Hopkins (ARCH) core facility (rockfish.jhu.edu), which is supported by the National Science Foundation (NSF) grant number OAC 1920103.

AUTHOR CONTRIBUTIONS

J.K.B., O.Y.A., and B.L. conceived the method, designed the experiments, and ran the experiments. J.K.B. and B.L. wrote the paper. J.K.B. wrote the DandD software tool. J.K.B., O.Y.A., and B.L. edited and approved the final manuscript.

DECLARATION OF INTERESTS

The authors declare no competing interests.

Received: December 29, 2023
Revised: January 11, 2024
Accepted: January 23, 2024
Published: February 1, 2024

REFERENCES

1. Parmigiani, L., Wittler, R., and Stoye, J. (2022). Revisiting pangenome openness with k-mers. Preprint at bioRxiv. https://doi.org/10.1101/2022.11.15.516472.
2. Sherman, R.M., and Salzberg, S.L. (2020). Pan-genomics in the human genome era. Nat. Rev. Genet. *21*, 243–254.
3. Kociumaka, T., Navarro, G., and Prezza, N. (2023). Towards a definitive compressibility measure for repetitive sequences. IEEE Trans. Inf. Theor. *69*, 2074–2092.
4. Burrows, M., and Wheeler, D. (1994). A block-sorting lossless data compression algorithm. In Digital SRC Research Report, Citeseer.
5. Ziv, J., and Lempel, A. (1977). A universal algorithm for sequential data compression. IEEE Trans. Inf. Theor. *23*, 337–343.
6. Kempa, D., and Prezza, N. (2018). At the roots of dictionary compression: string attractors. In Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, pp. 827–840.
7. Kokot, M., Dlugosz, M., and Deorowicz, S. (2017). KMC 3: counting and manipulating k-mer statistics. Bioinformatics *33*, 2759–2761.
8. Broder, A.Z. (1997). On the resemblance and containment of documents. In Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No. 97TB100171) (IEEE), pp. 21–29.
9. Ondov, B.D., Treangen, T.J., Melsted, P., Mallonee, A.B., Bergman, N.H., Koren, S., and Phillippy, A.M. (2016). Mash: fast genome and metagenome distance estimation using MinHash. Genome Biol. *17*, 132.
10. Meunier, F., Gandouet, O., Fusy, É., and Flajolet, P. (2007). Hyperloglog: The Analysis of a Near-Optimal Cardinality Estimation Algorithm (Discrete Mathematics & Theoretical Computer Science).
11. Baker, D.N., and Langmead, B. (2019). Dashing: fast and accurate genomic distances with HyperLogLog. Genome Biol. *20*, 265.
12. Boucher, C., Gagie, T., Kuhnle, A., Langmead, B., Manzini, G., and Mun, T. (2019). Prefix-free parsing for building big BWTs. Algorithm Mol. Biol. *14*, 13.
13. Hong, A., Rossi, M., and Boucher, C. (2023). Lz77 via prefix-free parsing. In 2023 Proceedings of the Symposium on Algorithm Engineering and Experiments (ALENEX) (SIAM), pp. 123–134.
14. Chaisson, M.J.P., Sanders, A.D., Zhao, X., Malhotra, A., Porubsky, D., Rausch, T., Gardner, E.J., Rodriguez, O.L., Guo, L., Collins, R.L., et al. (2019). Multi-platform discovery of haplotype-resolved structural variation in human genomes. Nat. Commun. *10*, 1784.
15. Ebert, P., Audano, P.A., Zhu, Q., Rodriguez-Martin, B., Porubsky, D., Bonder, M.J., Sulovari, A., Ebler, J., Zhou, W., Serra Mari, R., et al. (2021). Haplotype-resolved diverse human genomes and integrated analysis of structural variation. Science *372*, eabf7117.
16. Li, H., Handsaker, B., Wysoker, A., Fennell, T., Ruan, J., Homer, N., Marth, G., Abecasis, G., and Durbin, R.; 1000 Genome Project Data Processing Subgroup (2009). The sequence alignment/map format and samtools. Bioinformatics *25*, 2078–2079.
17. Wang, T., Antonacci-Fulton, L., Howe, K., Lawson, H.A., Lucas, J.K., Phillippy, A.M., Popejoy, A.B., Asri, M., Carson, C., Chaisson, M.J.P., et al. (2022). The Human Pangenome Project: a global resource to map genomic diversity. Nature *604*, 437–446.
18. Zielezinski, A., Girgis, H.Z., Bernard, G., Leimeister, C.A., Tang, K., Dencker, T., Lau, A.K., Röhling, S., Choi, J.J., Waterman, M.S., et al. (2019). Benchmarking of alignment-free sequence comparison methods. Genome Biol. *20*, 144.
19. Baker, D.N., and Langmead, B. (2022). Dashing 2: genomic sketching with multiplicities and locality-sensitive hashing. Preprint at bioRxiv. https://doi.org/10.1101/2022.10.16.512384.
20. Chikhi, R., and Medvedev, P. (2014). Informed and automated k-mer size selection for genome assembly. Bioinformatics *30*, 31–37.
21. Bussi, Y., Kapon, R., and Reich, Z. (2021). Large-scale k-mer-based analysis of the informational properties of genomes, comparative genomics and taxonomy. PLoS One *16*, e0258693.
22. Ertl, O. (2017). New cardinality estimation algorithms for hyperloglog sketches, CoRR abs/1702. Preprint at arXiv. http://arxiv.org/abs/1702.01284.arXiv:1702.01284.
23. Page, A.J., Cummins, C.A., Hunt, M., Wong, V.K., Reuter, S., Holden, M.T.G., Fookes, M., Falush, D., Keane, J.A., and Parkhill, J. (2015). Roary: rapid large-scale prokaryote pan genome analysis. Bioinformatics *31*, 3691–3693.
24. Sheikhizadeh, S., Schranz, M.E., Akdel, M., de Ridder, D., and Smit, S. (2016). PanTools: representation, storage and exploration of pan-genomic data. Bioinformatics *32*, i487–i493.

STAR★METHODS

KEY RESOURCES TABLE

REAGENT or RESOURCE	SOURCE	IDENTIFIER
Deposited data		
28 *Homo sapiens* phased genome assemblies	Human Pangenome Reference Consortium; https://doi.org/10.5281/zenodo.5826274	https://github.com/jessicabonnie/dandd_experiments/blob/main/accessions/figure3_hprc.txt
26 *Homo sapiens* full genomes – integrated variant call sets	Human Structural Variation Consortium phase 2; https://www.internationalgenome.org/data-portal/data-collection/hgsvc2	https://github.com/jessicabonnie/dandd_experiments/blob/main/accessions/figure3_hgsvc2.txt
29 *Escherichia coli* assembled genomes	AFproject; https://afproject.org/app/benchmark/genome/std/assembled/ecoli/dataset/	https://github.com/jessicabonnie/dandd_experiments/blob/main/accessions/figure4_ecoli.txt
25 fish mitochondria (mtDNA) assembled genomes	AFproject; https://afproject.org/app/benchmark/genome/std/assembled/fish_mito/dataset/	https://github.com/jessicabonnie/dandd_experiments/blob/main/accessions/figure4_fish-mito.txt
12 *Homo Sapiens* reference genomes, GRCh38	NCBI GenBank	https://github.com/jessicabonnie/dandd_experiments/blob/main/accessions/figure1_human.txt
10 *Escherichia coli* genome assemblies	NCBI RefSeq	https://github.com/jessicabonnie/dandd_experiments/blob/main/accessions/table1_ecoli.txt
60 *Salmonella enterica* genome assemblies	NCBI RefSeq	https://github.com/jessicabonnie/dandd_experiments/blob/main/accessions/table1_salmonella.txt and https://github.com/jessicabonnie/dandd_experiments/blob/main/accessions/figure2_salmonella.txt
Software and algorithms		
DandD	This Study	Zenodo DOI; https://doi.org/10.5281/zenodo.10138641 GitHub: https://github.com/jessicabonnie/dandd Experiments GitHub: https://github.com/jessicabonnie/dandd_experiments
Dashing	Baker & Langmead	DOI: https://doi.org/10.1101/501726 ; GitHub: https://github.com/dnbaker/dashing
KMC3	REFRESH Bioinformatics Group	GitHub: https://github.com/refresh-bio/KMC/tree/v3.2.2

RESOURCE AVAILABILITY

Lead contact

Further information and requests for resources and reagents should be directed to and will be fulfilled by the lead contact, Ben Langmead (langmea@cs.jhu.edu).

Materials availability

This study did not generate new unique reagents.

Data and code availability

- This paper analyzes existing, publicly available data. These accession numbers for the datasets are listed in the key resources table.
- All original code has been deposited at Zenodo and is publicly available as of the date of publication. DOIs are listed in the key resources table.
- Any additional information required to reanalyze the data reported in this paper is available from the lead contact upon request.

METHOD DETAILS

The delta compressibility measure

Various measures have been proposed for how to quantify the amount of distinct sequence in a pangenome. Some of these measures identified as byproducts of particular compression strategies. For instance, the measure z is derived by computing a Lempel-Ziv parse of the pangenome;[5] z is equal to the number of phrases in that parse. The measure r is obtained by computing the Burrows-Wheeler Transform

(BWT) of the pangenome;[4] r is equal to the number of maximal same-letter runs in the BWT-transformed string. Other proposals generalize the notion of compressibility to be independent of any compression strategy, such as the string-attractor γ.[6] Delta (δ) is another measure of compressibility, defined over a pangenome S:

$$\delta(S) = \max_k \frac{d_k(S)}{k} \quad \text{(Equation 1)}$$

where $d_k(S)$ is the number of distinct length-k substrings among all strings in S. We use k^* to denote the value of k that achieves the maximum. When S consists of a single string s, the expression $d_k(S)/k$ can be considered to undergo three phases of growth with respect to k. For values of k that are so short that virtually all possible k-mer arrangements of the alphabet appear in s, $d_k(S)/k$ grows exponentially. For values of k approaching $|s|$. , $d_k(S)/k$ decreases linearly as k-mers outgrow s, eventually reaching 1 when $k = |s|$. For intermediate values of k, these trends are in tension; increasing k both increases the space of possible k-mers, but eventually stops gaining many new distinct k-mers. Choosing the k that maximizes $d_k(S)/k$ identifies this point of diminishing returns. While it is useful to think in terms of these phases, it is important to note that the growth of $d_k(S)/k$ is not strictly bitonic, i.e. it does not necessarily grow monotonically, reach a peak, then fall monotonically. Rather it can experience local fluctuations leading to non-global maxima. δ is insensitive to full-string reversals and monotone with respect to appending or prepending symbols to S. The k in the denominator of Equation 1 links δ to other measures. For instance, with k in the denominator for δ, it is easy to show that $\delta \leq \gamma$, where γ is the string attractor size.[3] As a function of the substring composition of S, δ is comparatively easy to compute relative to other measures. The computation of z or r is concerned with the entirety of S, whereas δ can be computed incrementally by considering substrings of the members of S one-by-one. The advantages of this incremental approach are two-fold: (a) δ can be computed simply and in linear time by scanning S and counting k-mers for an appropriate range of values of k, and (b) DNA k-mers can be "canonicalized" at the outset, allowing DNA strings and their reverse complements to be treated as equivalent for the purpose of computing δ, as is common. During a single scan of S, each k-mer can be tallied either as itself or as its reverse complement (whichever is lexicographically smaller). For z and r, allowing for equal treatment of forward and reverse complement strands would require a more drastic approach, e.g. first concatenating S with its reverse complement, then running the corresponding algorithm.

Estimating cardinality

mer counting is resource intensive, potentially requiring a large memory footprint for pangenome inputs. Instead, we propose a method for estimating δ by estimating the numerator $d_k(S)$ (i.e. the cardinality) from Equation 1. The Dashing[11] tool, while chiefly used to estimate similarities between sequencing datasets, can also be used to estimate cardinalities like this via its dashing card function, which uses Ertl's maximum likelihood estimator.[11,22] Genomic sketches are composable, meaning that sketches built over datasets A and B can be easily combined to form the sketch for $A \cup B$. In the case of HyperLogLog sketches, this is accomplished by simply taking the elementwise maximum of the register values for the sketches of A and B. The resulting sketch is identical to the one that would have resulted from sketching $A \cup B$. Since registers usually number in the thousands-to-millions range, this is a fast operation, using only sequential memory accesses.

Estimating delta

To estimate δ, we seek the substring length k giving maximal $d_k(S)/k$, which we denote k^*. Determining k^* requires a scan, similar to a root-finding procedure. DandD accomplishes this with a simple sweep starting from a user specified initial value of k. The sweep tries successively larger and smaller values of k, searching for three consecutive values of k such that:

$$\frac{d_{k'-1}(S)}{k'-1} < \frac{d_{k'}(S)}{k'} > \frac{d_{k'+1}(S)}{k'+1} \quad \text{(Equation 2)}$$

By default, DandD begins its search at $k = 14$. The final value of k^* is dataset-dependent, as illustrated in Figure 1, which shows a 0-1 normalized version of δ for human, E. coli and salmonella pangenomes. In some of DandD's modes (such as progressive and kij), it computes δ with respect to the union of two or more inputs for which it has previously computed δ. In such cases, DandD initializes the search for the union k^* by taking the maximum of the previously-computed k^*s of the inputs.

Characterizing sublinear growth

DandD, via the DandD progressive command, can characterize the rate of growth of a pangenome by measuring δ with respect to cumulative subsets of its constituent genomes. For small pangenomes (up to 6–7 genomes), it can be practical to examine all possible orderings (permutations) of the genomes. However, for larger real-world pangenomes, it is sufficient to use a random subset of all possible orderings. This method provides a way to estimate the average δ for subsets of a given size. That is, by taking the mean of all the values for δ obtained after adding the i^{th} genome in each ordering, we have an estimate for δ(all size-i subsets). Past methods for characterizing pangenome growth also made use of random orderings of large collections. These methods seek to determine whether the pangenome is "open" (still accumulating new sequence), or "closed" (substantially complete).[1,23,24] DandD provides a new way of performing this analysis over genome sequences in a parameter-free way, not requiring foreknowledge of where genes are located or how to choose an appropriate value of k. The DandD progressive command allows the user to provide a set of genomes and a desired number of orderings to try. DandD will then (1) preprocess all of the inputs individually, (2) generate the random orderings, and then (3) iterate through each ordering, "progressively" building larger unions by accumulating one more genome at each step. The DandD progressive command outputs a file describing, for each step of each ordering,

which genome was added in that step and the value of δ for the new union. Besides giving useful plots (seen in "Using DandD to characterize sublinearity and openness" in Results), this output can be used to fit a Heap's Law model to the change in delta ($\Delta\delta$) at each step. After fitting, the fit value of the Heaps'-Law α parameter can be used to characterize whether the pangenome is open ($\alpha \leq 1$) or closed ($\alpha > 1$).

k-independent Jaccard

The Jaccard coefficient is a widely used metric for comparing large datasets:

$$J_k(A,B) = \frac{d_k(A \cap B)}{d_k(A \cup B)} \quad \text{(Equation 3)}$$

For consistency we use $d_k(A)$ (rather than $|A|_k$) to denote the cardinality of the set of *k*-mers in a collection of strings *A*. Methods like MinHash estimate this quantity directly. Methods based on the HyperLogLog sketch, like Dashing, obtain separate cardinalities $d_k(A)$, $d_k(B)$ and $d_k(A \cup B)$ and compute J_k using an expression equivalent (by the inclusion-exclusion principle) to the one above:

$$J_k(A,B) = \frac{d_k(A)+d_k(B) - d_k(A \cup B)}{d_k(A \cup B)} \quad \text{(Equation 4)}$$

The above expressions have *k*-mer length *k* as a parameter. To obtain a *k*-independent notion of Jaccard coefficient, we replace d_k with δ:

$$\mathrm{KIJ}(A,B) = \frac{\delta(A)+\delta(B) - \delta(A \cup B)}{\delta(A \cup B)} \quad \text{(Equation 5)}$$

Following this formula, the task of computing or estimating *k*-Independent Jaccard (KIJ) reduces to the task of obtaining $\delta(A)$, $\delta(B)$ and $\delta(A \cup B)$ (or estimates thereof). DandD provides a command (dandd kij) to compute all-pairwise KIJs given two or more input FASTA files. Since downstream tools may expect to receive distances rather than similarities, DandD can output all-pairwise $1 - \mathrm{KIJ}$s instead. If users wish to produce their own scans of likely *k*s for the original jaccard metric they can add the –jaccard command along with –mink and –maxk to specify a range.

Caching and lazy evaluation of sketches

The most time and memory intensive step of solving for δ is the creation of the component sketches from the input FASTA files. Computationally, it is far simpler to produce the union of two or more sketches or estimate cardinality from existing sketches. Given these uneven resource requirements, DandD is designed to reuse sketches within tasks and across experiments within the same pangenome. DandD reduces its footprint and prevents the production of duplicate sketches by caching sketches on the file system and tracking the sketches it has already built. To maintain the association between a union sketch and its component FASTAs, the sketch file is named using a checksum over the constituent FASTAs as computed by the cryptographic hash function BLAKE. This serves a dual purpose of insuring that each combination of input FASTAs is sketched only once, regardless of order, and providing a mechanism to confirm agreement between the currently available FASTAs and any information pertaining to them that may be stored within DandD's tree structure. In addition to a naming convention, DandD also creates a directory structure to store intermediate sketches and databases for easy reuse and access. When the user specifies the same sketch directory across many invocations of DandD, they will maximize the benefit of reusing sketch files.

Exact mode

DandD includes an "exact" option (–exact) which enables computation of δ directly by way of *k*-mer counting. This option can be used in combination with any of DandD's modes (tree, progressive, kij, etc). Instead of using Dashing, "exact" mode uses KMC3 for counting (via the kmc command) and unioning (via kmc_tools complex).[7] Just as Dashing must build a sketch prior to estimating cardinalities, KMC3 builds a "database" of *k*-mer counts for each combination. Note that while unioning two Dashing sketches requires only an elementwise maximum over the sketches, unioning two KMC databases requires a merge sort over all the *k*-mers and counts. This can be quite expensive, as detailed in Table 1. DandD's file naming scheme together with the metadata it saves allows users to locate KMC databases and Dashing sketches corresponding to particular inputs and their unions. This facilitates further experimentation; i.e. a user can use KMC3's tools to explore which *k*-mers contributed to a particularly large increase in δ.