Open access • Posted Content • DOI:10.1101/029827

Fast genome and metagenome distance estimation using MinHash — Source link

Brian D. Ondov, Todd J. Treangen, Adam B. Mallonee, Nicholas H. Bergman ...+2 more authors

Institutions: National Institutes of Health

Published on: 26 Oct 2015 - bioRxiv (Cold Spring Harbor Labs Journals)

Topics: MinHash, Sequence clustering, RefSeq and Cluster analysis

Ondov et al. Genome Biology (2016) 17:132
DOI 10.1186/s13059-016-0997-x

Genome Biology

SOFTWARE Open Access

Mash: fast genome and metagenome distance estimation using MinHash

Brian D. Ondov[1], Todd J. Treangen[1], Páll Melsted[2], Adam B. Mallonee[1], Nicholas H. Bergman[1], Sergey Koren[3] and Adam M. Phillippy[3*]

Abstract

Mash extends the MinHash dimensionality-reduction technique to include a pairwise mutation distance and *P* value significance test, enabling the efficient clustering and search of massive sequence collections. Mash reduces large sequences and sequence sets to small, representative sketches, from which global mutation distances can be rapidly estimated. We demonstrate several use cases, including the clustering of all 54,118 NCBI RefSeq genomes in 33 CPU h; real-time database search using assembled or unassembled Illumina, Pacific Biosciences, and Oxford Nanopore data; and the scalable clustering of hundreds of metagenomic samples by composition. Mash is freely released under a BSD license (https://github.com/marbl/mash).

Keywords: Comparative genomics, Genomic distance, Alignment, Sequencing, Nanopore, Metagenomics

Background

When BLAST was first published in 1990 [1], there were less than 50 million bases of nucleotide sequence in the public archives [2]; now a single sequencing instrument can produce over 1 trillion bases per run [3]. New methods are needed that can manage and help organize this scale of data. To address this, we consider the general problem of computing an approximate distance between two sequences and describe Mash, a general-purpose toolkit that utilizes the MinHash technique [4] to reduce large sequences (or sequence sets) to compressed sketch representations. Using only the sketches, which can be thousands of times smaller, the similarity of the original sequences can be rapidly estimated with bounded error. Importantly, the error of this computation depends only on the size of the sketch and is independent of the genome size. Thus, sketches comprising just a few hundred values can be used to approximate the similarity of arbitrarily large datasets. This has important applications for large-scale genomic data management and emerging long-read, single-molecule sequencing technologies. Potential applications include any problem where an approximate, global distance is acceptable, e.g. to triage and cluster sequence data, assign species labels, build large guide trees, identify mis-tracked samples, and search genomic databases.

The MinHash technique is a form of locality-sensitive hashing [5] that has been widely used for the detection of near-duplicate Web pages and images [6, 7], but has seen limited use in genomics despite initial applications over ten years ago [8]. More recently, MinHash has been applied to the relevant problems of genome assembly [9], 16S rDNA gene clustering [10, 11], and metagenomic sequence clustering [12]. Because of the extremely low memory and CPU requirements of this probabilistic approach, MinHash is well suited for data-intensive problems in genomics. To facilitate this, we have developed Mash for the flexible construction, manipulation, and comparison of MinHash sketches from genomic data. We build upon past applications of MinHash by deriving a new significance test to differentiate chance matches when searching a database, and derive a new distance metric, the Mash distance, which estimates the mutation rate between two sequences directly from their MinHash sketches. Similar "alignment-free" methods have a long history in bioinformatics [13, 14]. However, prior methods based on word counts have relied on short words of only a few nucleotides, which lack the power to differentiate between closely related sequences and produce distance

* Correspondence: adam.phillippy@nih.gov
[3]Genome Informatics Section, Computational and Statistical Genomics Branch, National Human Genome Research Institute, National Institutes of Health, Bethesda, MD, USA
Full list of author information is available at the end of the article

measures that can be difficult to interpret [15–18]. Alternatively, methods based on string matching can produce very accurate estimates of mutation distance, but must process the entire sequence with each comparison, which is not feasible for all-pairs comparisons [19–22]. In contrast, the Mash distance can be quickly computed from the size-reduced sketches alone, yet produces a result that strongly correlates with alignment-based measures such as the Average Nucleotide Identity (ANI) [23]. Thus, Mash combines the high specificity of matching-based approaches with the dimensionality reduction of statistical approaches, enabling accurate all-pairs comparisons between many large genomes and metagenomes.

Mash provides two basic functions for sequence comparisons: *sketch* and *dist*. The *sketch* function converts a sequence or collection of sequences into a MinHash sketch (Fig. 1). The *dist* function compares two sketches and returns an estimate of the Jaccard index (i.e. the fraction of shared k-mers), a *P* value, and the Mash distance, which estimates the rate of sequence mutation under a simple evolutionary model [22] (see "Methods"). Since Mash relies only on comparing length *k* substrings, or k-mers, the inputs can be whole genomes, metagenomes, nucleotide sequences, amino acid sequences, or raw sequencing reads. Each input is simply treated as a collection of k-mers taken from some known alphabet, allowing many applications. Here we examine three specific use cases: (1) sketching and clustering the entire NCBI RefSeq genome database; (2) searching assembled and unassembled genomes against the sketched RefSeq database in real time; and (3) computing a distance between metagenomic samples using both assembled and unassembled read sets. Additional applications can be envisioned and are covered in the "Discussion".

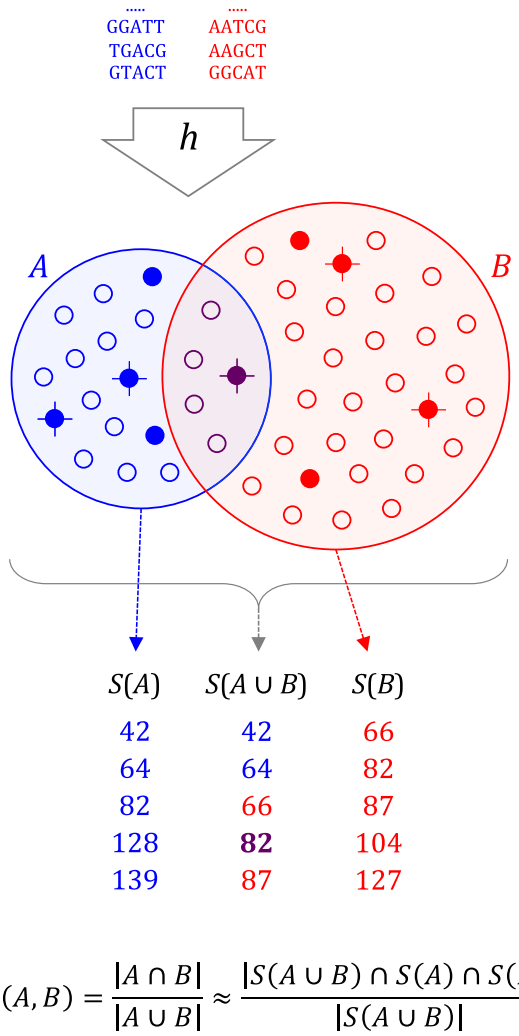

Fig. 1 Overview of the MinHash bottom sketch strategy for estimating the Jaccard index. First, the sequences of two datasets are decomposed into their constituent k-mers (*top*, *blue and red*) and each k-mer is passed through a hash function *h* to obtain a 32- or 64-bit hash, depending on the input k-mer size. The resulting hash sets, *A* and *B*, contain $|A|$ and $|B|$ distinct hashes each (*small circles*). The Jaccard index is simply the fraction of shared hashes (*purple*) out of all distinct hashes in *A* and *B*. This can be approximated by considering a much smaller random sample from the union of *A* and *B*. MinHash sketches $S(A)$ and $S(B)$ of size $s = 5$ are shown for *A* and *B*, comprising the five smallest hash values for each (*filled circles*). Merging $S(A)$ and $S(B)$ to recover the five smallest hash values overall for $A \cup B$ (*crossed circles*) yields $S(A \cup B)$. Because $S(A \cup B)$ is a random sample of $A \cup B$, the fraction of elements in $S(A \cup B)$ that are shared by both $S(A)$ and $S(B)$ is an unbiased estimate of $J(A,B)$

Results and discussion

Clustering all genomes in NCBI RefSeq

Mash enables scalable whole-genome clustering, which is an important application for the future of genomic data management, but currently infeasible with alignment-based approaches. As genome databases increase in size and whole-genome sequencing becomes routine, it will become impractical to manually assign taxonomic labels for all genomes. Thus, generalized and automated methods will be useful for constructing groups of related genomes, e.g. for the automated detection of outbreak clusters [24]. To illustrate the utility of Mash, we sketched and clustered all of NCBI RefSeq Release 70 [25], totaling 54,118 organisms and 618 Gbp of genomic sequence. The resulting sketches total only 93 MB (Additional file 1: Supplementary Note 1), yielding a compression factor of more than 7000-fold versus the uncompressed FASTA (674 GB). Further compression of the sketches is possible using standard compression tools. Sketching all genomes and computing all ~1.5 billion pairwise distances required just 26.1 and 6.9 CPU h, respectively. This process is easily parallelized, which can reduce the wall clock time to minutes with sufficient compute resources. Once constructed, additional genomes can be added incrementally to the full RefSeq database in just 0.9 CPU s per 5 MB

genome (or 4 CPU min for a 3 GB genome). Thus, we have demonstrated that it is possible to perform unsupervised clustering of all known genomes and to efficiently update this clustering as new genomes are added.

Importantly, the resulting Mash distances correlate well with ANI (a common measure of genome similarity), with $D \approx 1 - ANI$ over multiple sketch and k-mer sizes (Fig. 2). Due to the high cost of computing ANI via whole-genome alignment, a subset of 500 Escherichia genomes was selected for comparison (Additional file 1: Supplementary Note 1). For ANI in the range of 90–100 %, the correlation with Mash distance is very strong across multiple sketch sizes and choices of k. For the default sketch size of $s = 1000$ and $k = 21$, Mash approximates 1–ANI with a root-mean-square error of 0.00274 on this dataset. This correlation begins to degrade for more divergent genomes because the variance of the Mash estimate grows with distance. Increasing sketch size improves the accuracy of Mash estimates, especially for divergent genomes (Table 1, Additional file 1: Figures S1 and S2). This results in a negligible increase in runtime for sketching, but the size of the resulting sketches and time required for distance comparisons increases linearly (Table 2). The choice of k is a tradeoff between sensitivity and specificity. Smaller values of k are more sensitive for divergent genomes, but lose specificity for large genomes due to chance k-mer collisions (Additional file 1: Figure S3). Such chance collisions will skew the Mash distance, but given a known genome size, undesirable k-mer collisions can be avoided by choosing a suitably large value of k (see "Methods"). However, too large of a k-mer will reduce sensitivity and so choosing the smallest k that avoids chance collisions is recommended.

Approximate species clusters can be generated from the all-pairs distance matrix by graph clustering methods or simple thresholding of the Mash distance to create connected components. To illustrate, we linked all RefSeq genomes with a pairwise Mash distance ≤0.05, which equates to an ANI of ≥95 %. This threshold roughly corresponds to a 70 % DNA-DNA reassociation

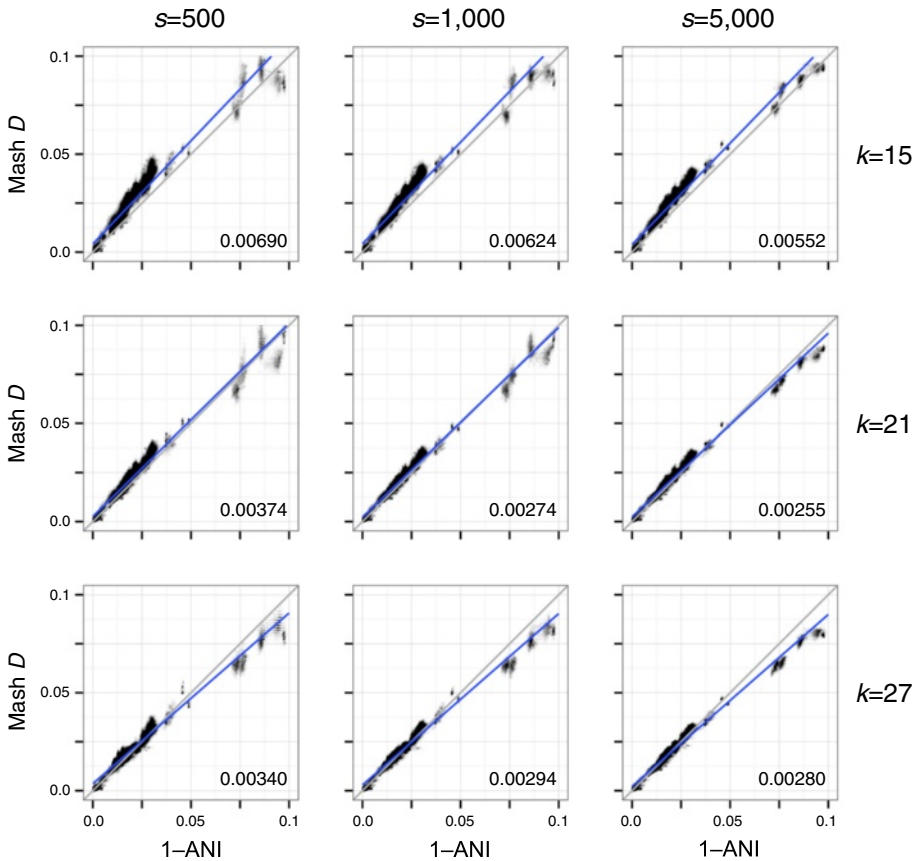

Fig. 2 *Scatterplots* illustrating the relationship between ANI and Mash distance for a collection of Escherichia genomes. Each *plot column* shows a different sketch size *s* and each plot row a different k-mer size *k*. *Gray lines* show the model relationship $D = 1–ANI$ and *numbers* in the *bottom right* of each plot give the root-mean-square error versus this perfect model. *Blue lines* show linear regression models. Increasing the sketch size improves the accuracy of the Mash distance, especially for more divergent sequences. However, there is a limit on how well the Mash distance can approximate ANI, especially for more divergent genomes (e.g. ANI considers only the core genome)

Table 1 Example Mash error bounds for a k-mer size of 21 and increasing sketch sizes

	Mash distance							
Sketch size	0.05	0.10	0.15	0.20	0.25	0.30	0.35	0.40
100	0.0271	0.0868	–	–	–	–	–	–
500	0.0098	0.0245	0.0473	–	–	–	–	–
1000	0.0068	0.0158	0.0323	0.0630	–	–	–	–
5000	0.0029	0.0065	0.0124	0.0235	0.0460	–	–	–
10,000	0.0020	0.0046	0.0086	0.0159	0.0300	0.0726	–	–
50,000	0.0009	0.0020	0.0037	0.0065	0.0116	0.0219	0.0396	0.0822
100,000	0.0006	0.0014	0.0026	0.0046	0.0081	0.0143	0.0250	0.0492
500,000	0.0003	0.0006	0.0011	0.0020	0.0035	0.0060	0.0105	0.0187
1,000,000	0.0002	0.0004	0.0008	0.0014	0.0024	0.0042	0.0072	0.0128

For a given sketch size and Mash distance, the Mash estimation error will be less than the given value with 0.99 probability, as calculated by the binomial inverse cumulative distribution function. Missing values indicate that the estimate is unbounded, i.e. there is a chance that no matching k-mers will be found and the Mash distance will be undefined. Plots of the upper and lower error bounds for $k = 16$ and $k = 21$ are given in Additional file 1: Figure S2

value—a historical, albeit debatable, definition of bacterial species [23]. Figure 3 shows the resulting graph of significant ($P \leq 10^{-10}$) pairwise distances with $D \leq 0.05$ for all microbial genomes. Simply considering the connected components of the resulting graph yields a partitioning that largely agrees with the current NCBI bacterial species taxonomy. Eukaryotic and plasmid components are shown in Additional file 1: Figures S4 and S5, but would require alternate parameters for species-specific clustering due to their varying characteristics.

Beyond simple clustering, the Mash distance is an approximation of the mutation rate that can also be used to rapidly approximate phylogenies using hierarchical clustering. For example, all pairwise Mash distances for 17 RefSeq primate genomes were computed in just 2.5 CPU h (11 min wall clock on 17 cores) with default parameters ($s = 1000$ and $k = 21$) and used to build a neighbor-joining tree [26]. Figure 4 compares this tree to an alignment-based phylogenetic tree model downloaded from the UCSC genome browser [27]. The Mash and UCSC primate trees are topologically consistent for everything except the Homo/Pan split, for which the Mash topology is more similar to past phylogenetic studies [28] and mitochondrial trees [14]. On average, the Mash branch lengths are slightly longer, with a Branch Score Distance [29] of 0.10 between the two trees, but additional distance corrections are possible for k-mer based models [22]. However, due to limitations of both the k-mer approach and simple distance model, we emphasize that Mash is not explicitly designed for phylogeny reconstruction, especially for genomes with high divergence or large size differences. For example, clustering the treeshrew, mouse, rat, guinea pig, and rabbit genomes alongside the primate genomes causes the tarsier to become misplaced (Additional file 1: Figure S6). Increasing the sketch size from 1000 to 5000 corrects this placement, but Mash has limited accuracy at these distances and should only be used in cases where such approximations are sufficient.

Real-time genome identification from assemblies or reads

With a pre-computed sketch database, Mash is able to rapidly identify isolated genomes from both assemblies and raw sequencing reads. To illustrate, we computed Mash distances for multiple *Escherichia coli* datasets compared against the RefSeq sketch database (Table 3). This test included the K12 MG1655 reference genome as well as assembled and unassembled sequencing runs from the ABI 3730, Roche 454, Ion PGM, Illumina MiSeq, PacBio RSII, and Oxford Nanopore MinION instruments. For assembled genomes, the correct strain was identified as the best hit in a few seconds. For each

Table 2 Mash runtime and output size for all-pairs RefSeq computation using various sketch and k-mer sizes

	k = 16				k = 21			
Sketch size	Sketch (CPU h)	Dist (CPU h)	Size (Mb)	gzip (Mb)	Sketch (CPU h)	Dist (CPU h)	Size (Mb)	gzip (Mb)
500	26.4	8.4	120.1	89.7	31.3	9.0	229.8	201.8
1000	27.7	15.9	224.9	179.7	31.3	17.4	439.2	399.6
5000	26.4	74.5	1022.5	873.8	31.6	83.6	2034.5	1924.6
10,000	26.8	146.9	1961.8	1691.1	31.7	164.0	3913.0	3696.2

Sketch: CPU h required for the Mash *sketch* operation for all 54,118 RefSeq genomes. *Dist:* CPU h required for the Mash *dist* table operation for all pairs of sketches. *Size:* combined size of the resulting sketches in megabytes. *gzip:* combined size of the resulting sketches after gzip compression

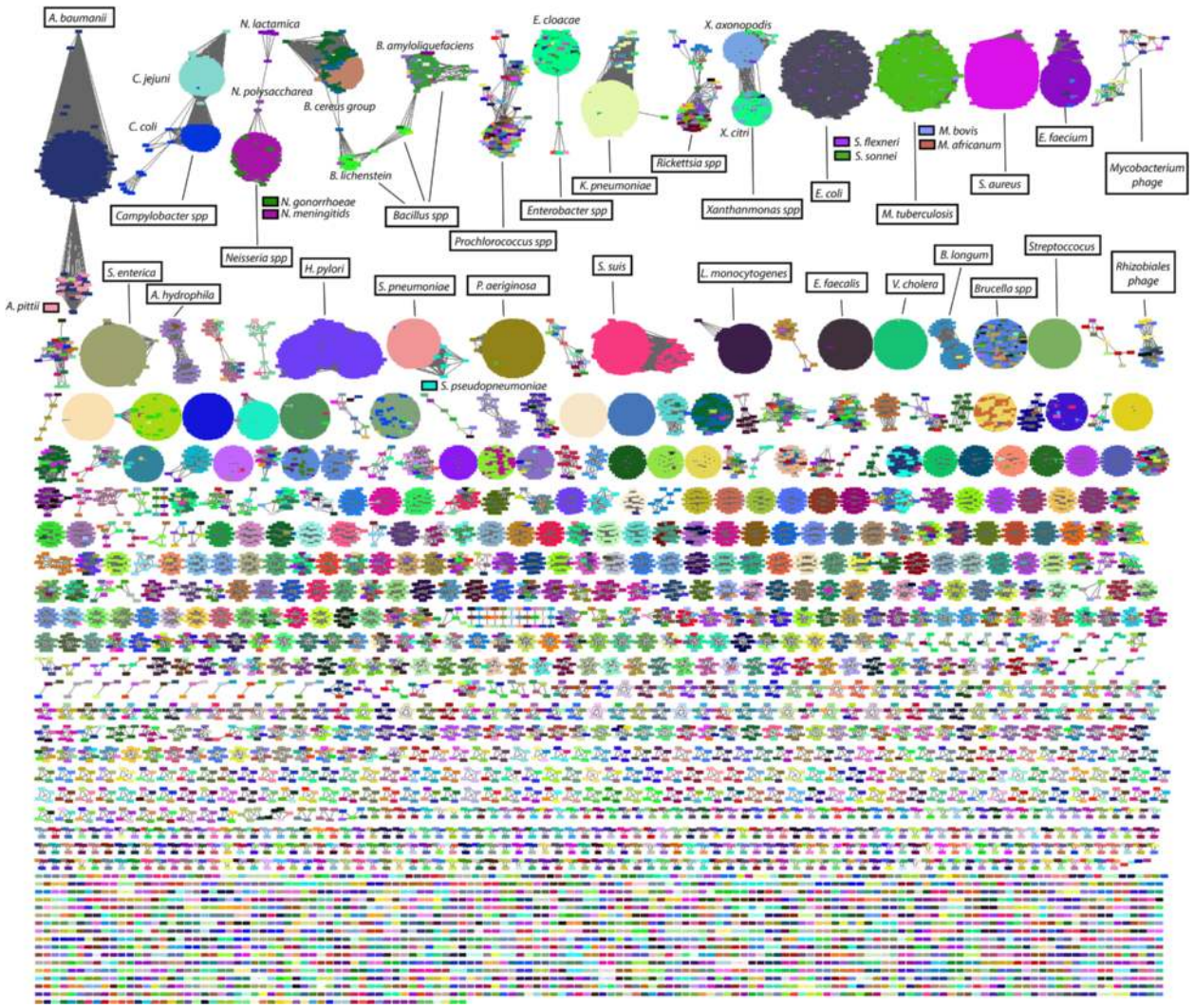

Fig. 3 Comparison and de novo clustering of all RefSeq genomes using Mash. Each *graph node* represents a genome. Two genomes are connected by an edge if their Mash distance $D \leq 0.05$ and P value $\leq 10^{-10}$. *Graph layout* was performed using Cytoscape [61] organic layout algorithm [62]. Individual nodes are colored by species and the *top two rows* of clusters have been annotated with the majority species label they contain. Only components containing microbial genomes are shown here (including viruses). Additional file 1: Figures S4 and S5 show eukaryotes, orphan plasmids, and organelles

unassembled genome, a single sketch was constructed from the collection of k-mers in the reads and compared to the sketch database. In these cases, the best hit was to the correct species, including for *E. coli* 1D MinION reads [30], which had an average sequencing error rate of ~40 %. However, the best-hit strain was often incorrect due to noise in the raw reads. To account for this uncertainly, we applied lowest common ancestor (LCA) classification (see "Methods"), which was correct in all cases, albeit with reduced resolution. To further mitigate the problem of erroneous k-mers, Mash can filter low-abundance k-mers from raw sequencing data to improve accuracy. Increasing the sketch size can also improve sensitivity, as would error correction using dedicated methods [31]. However, there are tradeoffs to consider when filtering or correcting low-coverage datasets (e.g. less than 5X coverage [22]).

To test Mash's discriminatory power, we searched Oxford Nanopore MinION reads collected from *Bacillus anthracis* and *Bacillus cereus* against the full RefSeq sketch database. In both cases Mash was able to correctly differentiate these closely related species (ANI ≈ 95 %) using 43,806 and 91,379 sequences collected from single MinION R7.3 runs of *B. anthracis* Ames and *B. cereus* ATCC 10987, respectively (combined 1D and 2D reads). In the case of the higher quality *B. cereus* reads, processed with a more recent ONT workflow (1.10.1 vs. 1.6.3), the correct strain was identified as the best hit. These two searches both required just 1 min of CPU and 209 MB of RAM. Such low-

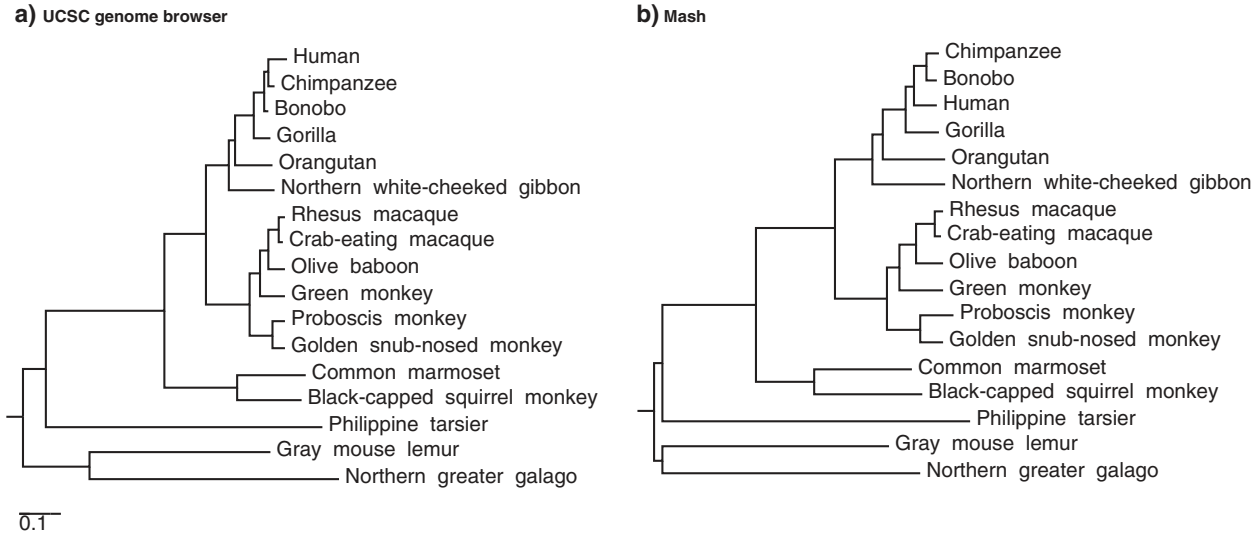

Fig. 4 Primate trees from the UCSC genome browser and Mash. **a** A primate phylogenetic tree model from the UCSC genome browser, with branch lengths derived from fourfold degenerate sites extracted from reference gene multiple alignments. **b** A comparable Mash-based tree generated from whole genomes using a sketch size of $s = 1000$ and k-mer size of $k = 21$. Additional file 1: Figure S6 includes this Mash tree with five additional mammals of increasing divergence

overhead searches could be used for quickly triaging unknown samples or to rapidly select a reference genome for performing further, more detailed comparative analyses. For example, Mash uses an online algorithm for sketch construction and can therefore compare a sequencing run against a sketch database in real time. When tested on the Ebola virus MinION dataset, the *Zaire ebolavirus* reference genome was matched with a Mash P value of 10^{-10} after processing the first 227,445 bases of sequencing data, which were collected by the MinION after just 770 s of sequencing. However, analyzing such streaming data presents a multiple testing problem and determining appropriate stopping conditions is left for future work (e.g. by monitoring the stability of a sketch as additional data are processed).

Clustering massive metagenomic datasets

Mash can also replicate the function of k-mer based metagenomic comparison tools, but in a fraction of the time previously required. The metagenomic comparison tool DSM, for example, computes an exact Jaccard index using all k-mers that occur more than twice per sample [32]. By definition, Mash rapidly approximates this result by filtering unique k-mers and estimating the Jaccard index via MinHash. COMMET also uses k-mers to approximate similarity, but attempts to identify a set of similar reads between two samples using Bloom filters [33, 34]. The similarity of two samples is then defined as the fraction of similar reads that the two datasets share, which is essentially a read-level Jaccard index. Thus, both DSM and COMMET report Jaccard-like similarity measures, which drop rapidly with increasing divergence, whereas the Mash distance is linear in terms of the mutation rate, but becomes less accurate with increasing divergence. Figure 5a replicates the analysis in Maillet et al. [33] using both Mash and COMMET to cluster Global Ocean Survey (GOS) data [35]. On this dataset, Mash is over tenfold faster than COMMET and correctly identifies clusters from the original GOS study. This illustrates the incremental scalability of Mash where the primary overhead is sketching, which occurs only once per each sample. After sketching, computing pairwise distances is near instantaneous. Thus, Mash avoids the quadratic barrier usually associated with all-pairs comparisons and scales well to many samples. For example, COMMET would require 1 h to add a new GOS sample to this analysis, compared to less than 1 min for Mash.

For a large-scale test, samples from the Human Microbiome Project [36] (HMP) and Metagenomics of the Human Intestinal Tract [37] (MetaHIT) were combined to create a ~10 TB 888-sample dataset. Importantly, the size of a Mash sketch is independent of the input size, requiring only 70 MB to store the combined sketches ($s = 10{,}000$, $k = 21$) for these datasets. Both assembled and unassembled samples were analyzed, requiring 4.4 CPU h to process all assemblies and 279.6 CPU h to process all read sets. We estimated that COMMET would require at least 140,000 CPU h to process all read sets (500 times slower than Mash), so it was not run on the full dataset. The Mash assembly-based and read-based clusters are remarkably similar, with all samples clearly

Table 3 Sequencing runs and assemblies searched against the Mash RefSeq database

Organism	Tech	Type	NCBI accession	Size (Mbp)	Time (CPU s)	LCA	Best hit
E. coli K12 MG1655	MiSeq	Assembly	(SPAdes)	4.6	2.45	Entero.	*E. coli* K12 MG1655
E. coli K12 MG1655	PacBio	Assembly	GCA_000801205	4.6	2.66	Entero.	*E. coli* K12 MG1655
E. coli DH1	ABI 3730	Reads	(Trace Archive)	60	17.08	Entero.	*E. coli* DH1
E. coli K12 MG1655	454	Reads	SRR797242	233	57.12	Entero.	*E. coli* K12 MG1655
E. coli K12 MG1655	Ion PGM	Reads	SRR515925	407	72.01	*E. coli*	*E. coli* K12 1655
E. coli K12 MG1655	MiSeq	Reads	SRR1770413	387	72.01	Entero.	*E. coli* KLY
E. coli K12 MT203	HiSeq	Reads	SRR490124	2155	369.86	*E. coli*	*E. coli* GCF_000833635
E. coli K12 MG1655	PacBio	Reads	SRR1284073	397	77.96	*E. coli*	*E. coli* XH140A GCF_000226585
E. coli K12 MG1655	MinION	1D	ERR764952..55	248	55.52	Entero.	*E. coli* O113 H21
E. coli K12 MG1655	MinION	2D	ERR764952..55	134	27.82	*E. coli*	*E. coli* GCF_000953515
B. anthracis Ames	MinION	1D + 2D	SRR2671867	210	44.66	*B. anthracis*	*B. anthracis* str. Carbosap
B. cereus ATCC 10987	MinION	1D + 2D	SRR2671868	266	76.85	*B. cereus* ATCC 10987	*B. cereus* ATCC 10987
Zaire ebolavirus	MinION	1D + 2D	ERR1050070	8.7	2.06	*Zaire ebolavirus*	*Zaire ebolavirus* Mayinga

In all cases, Mash search required 21 MB of RAM for genome assemblies and 209 MB of RAM for sequencing runs (due to the additional Bloom filter overhead). *Organism:* source strain. *Tech:* Sequencing technology ABI 3730, 454 GS FLX, Illumina MiSeq, Illumina HiSeq, Ion PGM, PacBio RSII, Oxford Nanopore MinION. *Type:* Assembly, reads, 1D and 2D nanopore reads. *NCBI accession:* NCBI accession of the dataset or reads. The SPAdes [63] assembly was derived from the MiSeq reads. *Size:* total dataset size in Mbp. *LCA:* lowest common ancestor classification based on the NCBI taxonomy and the resulting hits within a significance tolerance of the best. In several cases, the LCA is at the family level (Enterobacteriaceae) due to significant Mash hits to both *E. coli* and *S. sonnei* species. This is a known species naming conflict within the NCBI taxonomy, with some genomes sharing ANI >98 % between these species. *Best hit:* reports the smallest significant distance reported

grouped by body site (Fig. 5b). Additionally, Mash identified outlier samples that were independently excluded by the HMP's quality control process. When included in the clustering, these samples were the only ones that failed to cluster by body site (Additional file 1: Figure S7). However, because the Mash distance is based on simple k-mer sets, it may be more prone to batch effects from sequencing or sample preparation methods. For example, Mash does not cluster MetaHIT samples by health status, as previously reported [37], and MetaHIT samples appear to preferentially cluster with one another.

Conclusions

Mash enables the comparison and clustering of whole genomes and metagenomes on a massive scale. Potential applications include the rapid triage and clustering of sequence data, for example, to quickly select the most appropriate reference genome for read mapping or to identify mis-tracked or low quality samples that fail to cluster as expected. Strong correlation between the Mash distance and sequence mutation rate enables approximate phylogeny construction, which could be used to rapidly determine outbreak clusters for thousands of genomes in real time. Additionally, because the Mash distance is based upon simple set intersections, it can be computed using homomorphic encryption schemes [38], enabling privacy-preserving genomic tests [39].

Future applications of Mash could include read mapping and metagenomic sequence classification via windowed sketches or a containment score to test for the presence of one sequence within another [4]. However, both of these approaches would require additional sketch overhead to achieve acceptable sensitivity. Improvements in database construction are also expected. For example, rather than storing a single sketch per sequence (or window), similar sketches could be merged to further reduce space and improve search times. Obvious strategies include choosing a representative sketch per cluster or hierarchically merging sketches via a Bloom tree [40]. Finally, both the *sketch* and *dist* functions

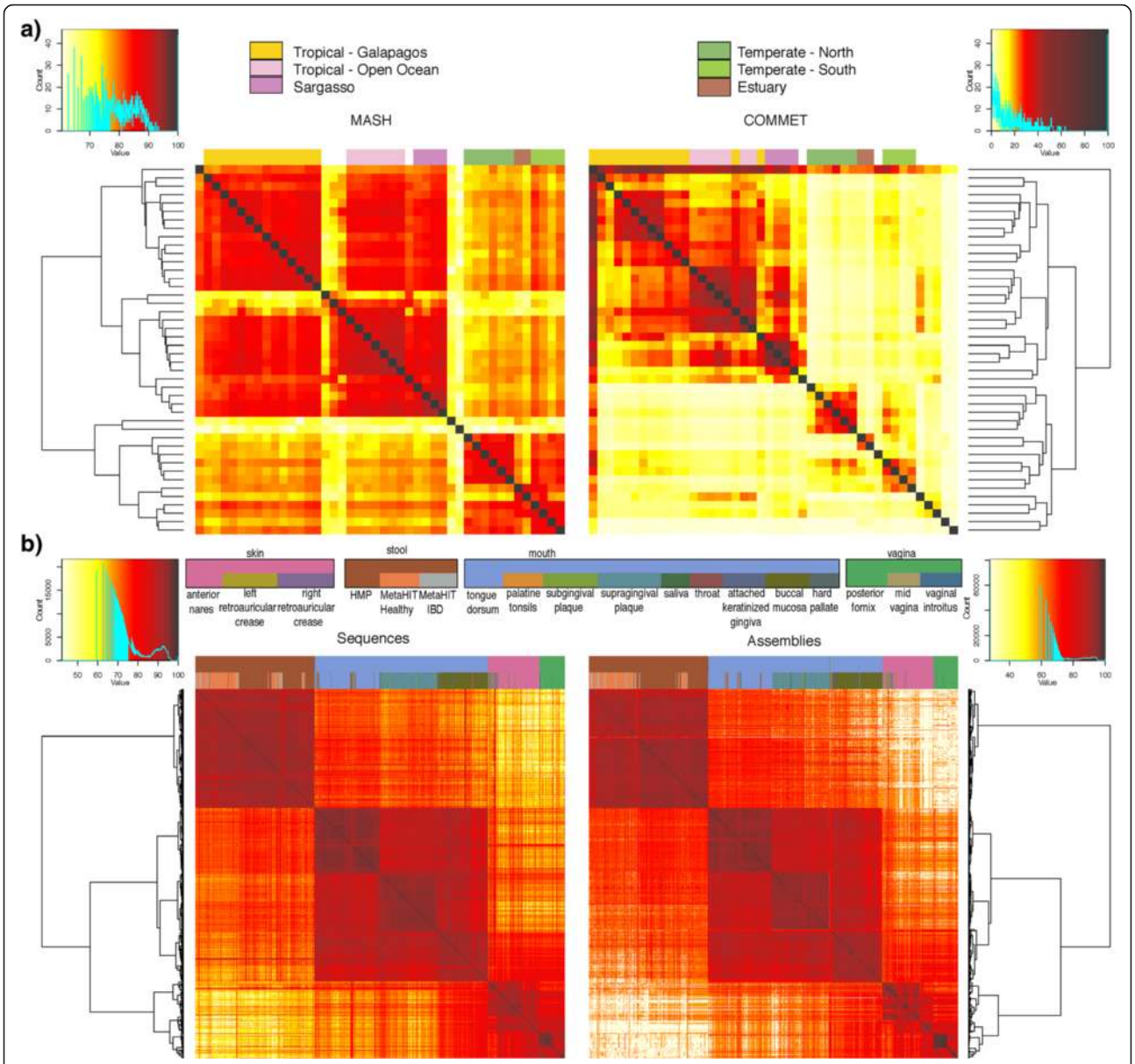

Fig. 5 Metagenomic clustering of ocean and human metagenomes using Mash. **a** Comparison of Global Ocean Survey (GOS) clustering using Mash (*top left*) and COMMET (*top right*) using raw Sanger sequencing data. *Heat maps* illustrate the pairwise similarity between samples, scaled between 0 (*white*) and 100 (*red*) for comparison to COMMET. Sample groups are identified and colored using the same key as in Rusch et al. [35]. The Mash clustering identifies two large clusters of temperate and tropical water samples as well as subgroupings consistent with the original GOS study. **b** Human metagenomic samples combined from the HMP and MetaHIT projects clustered by Mash from 888 sequencing runs (*bottom left*) and 879 assemblies (*bottom right*). For both sequencing reads and assemblies, Mash successfully clusters samples by body site and appropriately clusters MetaHIT and HMP stool samples together, even though these samples are from different projects with different protocols

are designed as online algorithms, enabling, for example, *dist* to continually update a sketch from a streaming input. The program could then be modified to terminate when enough data have been collected to make a species identification at a predefined significance threshold. This functionality is designed to support the analysis of real-time data streams, as is expected from nanopore-based sequencing sensors [24].

Methods

Mash sketch

To construct a MinHash sketch, Mash first determines the set of constituent k-mers by sliding a window of length k across the sequence. Mash supports arbitrary alphabets (e.g. nucleotide or amino acid) and both assembled and unassembled sequences. Without loss of generality, here we will assume a nucleotide

alphabet $\Sigma = \{A,C,G,T\}$. Depending on the alphabet size and choice of k, each k-mer is hashed to either a 32-bit or 64-bit value via a hash function, h. For nucleotide sequence, Mash uses canonical k-mers by default to allow strand-neutral comparisons. In this case, only the lexicographically smaller of the forward and reverse complement representations of a k-mer is hashed. For a given sketch size s, Mash returns the s smallest hashes output by h over all k-mers in the sequence (Fig. 1). Typically referred to as a "bottom-k sketch" for a sketch of size k, we refer to these simply as "bottom sketches" to avoid confusion with the k-mer size k. For a sketch size s and genome size n, a bottom sketch can be efficiently computed in $O(n \log s)$ time by maintaining a sorted list of size s and updating the current sketch only when a new hash is smaller than the current sketch maximum. Further, the probability that the i-th hash of the genome will enter the sketch is s/i, so the expected runtime of the algorithm is $O(n + s \log s \log n)$ [4], which becomes nearly linear when $n >> s$.

As demonstrated by Fig. 3, a sketch comprising 400 32-bit hash values is sufficient to roughly group microbial genomes by species. With these parameters, the resulting sketch size equals 1.6 kB for each genome. For large genomes, this represents an enormous lossy compression (e.g. compared to the 750 MB needed to store a 3 Gbp genome using 2-bit encoding). However, the probability of a given k-mer K appearing in a random genome X of size n is:

$$P(K \in X) = 1 - \left(1 - |\Sigma|^{-k}\right)^{n} \quad (1)$$

Thus, for $k = 16$ the probability of observing a given k-mer in a 3 Gbp genome is 0.50 and 25 % of k-mers are expected to be shared between two random 3 Gbp genomes by chance alone. This will skew any k-mer based distance and make distantly related genomes appear more similar than reality. To avoid this phenomenon, it is sufficient to choose a value of k that minimizes the probability of observing a random k-mer. Given a known genome size n and the desired probability q of observing a random k-mer (e.g. 0.01), this can be computed as [41]:

$$k' = \left\lceil \log_{|\Sigma|}(n(1-q)/q) \right\rceil \quad (2)$$

which yields $k = 14$ and $k = 19$ for 5 Mbp and 3 Gbp genomes ($q = 0.01$), respectively. We have found the parameters $k = 21$ and $s = 1000$ give accurate estimates in most cases (including metagenomes), so this is set as the default and still requires just 8 kB per sketch. However, for constructing the RefSeq database, $k = 16$ was chosen so that each hash could fit in 32 bits, minimizing the database size at the expense of reduced specificity for larger genomes. The small k also improves sensitivity, which helps when comparing noisy data like single-molecule sequencing (Additional file 1: Figures S2 and S3).

Lastly, for sketching raw sequencing reads, Mash provides both a two-stage MinHash and Bloom filter strategy to remove erroneous k-mers. These approaches assume that redundancy in the data (e.g. depth of coverage >5) will result in true k-mers appearing multiple times in the input, while false k-mers will appear only a few times. Given a coverage threshold c, Mash can optionally ignore such low-abundance k-mers with counts less than c. By default, the coverage threshold is set to one and all k-mers are considered for the sketch. Increasing this threshold enables the two-stage MinHash filter strategy, which is based on tracking both the k-mer hashes in the current sketch and a secondary set of candidate hashes. At any time, the current sketch contains the s smallest hashes of all k-mers that have been observed at least c times and the candidate set contains hashes that are smaller than the largest value in the sketch (sketch max), but have been observed less than c times. When processing new k-mers, those with a hash greater than the sketch max are immediately discarded, as usual. However, if a new hash is smaller than the current sketch max, it is checked against the candidate set. If absent, it is added to this set. If present with a count less than $c - 1$, its counter is incremented. If present with a count of $c - 1$ or greater, it is removed from the candidate set and added to the sketch. At this point, the sketch max has changed and the candidate set can be pruned to contain only values less than the new sketch maximum. The result of this online method is equivalent to running the MinHash algorithm on only those k-mers that occur c or more times in the input. However, in the worst case, if all k-mers in the input occur less than the coverage threshold c, no hashes would escape the candidate set and memory use would increase with each new k-mer processed.

Alternatively, a Bloom filter can be used to probabilistically exclude single-copy k-mers using a fixed amount of memory. In this approach, a Bloom filter is maintained instead of a candidate list and new hashes are inserted into the sketch only if they are less than sketch max and found in the Bloom filter. If a new hash would have otherwise been inserted in the sketch but was not found in the Bloom filter, it is inserted into the Bloom filter so that subsequent appearances of the hash will pass. This effectively excludes many single-copy k-mers from the sketch, but does not guarantee that all will be filtered. With this approach, filtering k-mers with a copy number greater than one would also be possible using a counting Bloom filter, but this has not been implemented since the exact method typically outperforms the Bloom method in practice, both in terms of accuracy and memory usage.

Mash distance

A MinHash sketch of size $s = 1$ is equivalent to the subsequent "minimizer" concept of Roberts et al. [42], which has been used in genome assembly [43], k-mer counting [44], and metagenomics [45]. Importantly, the more general MinHash concept permits an approximation of the Jaccard index $J(A,B) = \frac{|A \cap B|}{|A \cup B|}$ between two k-mer sets A and B. Mash follows Broder's original formulation and merge-sorts two bottom sketches $S(A)$ and $S(B)$ to estimate the Jaccard index [4]. The merge is terminated after s unique hashes have been processed (or both sketches exhausted), and the Jaccard estimate is computed as $j = \frac{x}{s'}$ for x shared hashes found after processing s' hashes. Because the sketches are stored in sorted order, this requires only $O(s)$ time and effectively computes:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \approx \frac{|S(A \cup B) \cap S(A) \cap S(B)|}{|S(A \cup B)|} \quad (3)$$

which is an unbiased estimate of the true Jaccard index, as illustrated in Fig. 1. Conveniently, the error bound of the Jaccard estimate $\varepsilon = O\left(\frac{1}{\sqrt{s}}\right)$ relies only on the sketch size and is independent of genome size [46]. Specific confidence bounds are given below and in Additional file 1: Figure S1. Note, however, that the relative error can grow quite large for very small Jaccard values (i.e. divergent genomes). In these cases, a larger sketch size or smaller k is needed to compensate. For flexibility, Mash can also compare sketches of different size, but such comparisons are constrained by the smaller of the two sketches $s < u$ and only the s smallest values are considered.

The Jaccard index is a useful measure of global sequence similarity because it correlates with ANI, a common measure of global sequence similarity. However, like the MUM index [19], J is sensitive to genome size and simultaneously captures both point mutations and gene content differences. For distance-based applications, the Jaccard index can be converted to the Jaccard distance $J_\delta(A,B) = 1 - J(A,B)$, which is related to the q-gram distance but without occurrence counts [47]. This can be a useful metric for clustering, but is non-linear in terms of the sequence mutation rate. In contrast, the Mash distance D seeks to directly estimate a mutation rate under a simple Poisson process of random site mutation. As noted by Fan et al. [22], given the probability d of a single substitution, the expected number of mutations in a k-mer is $\lambda = kd$. Thus, under a Poisson model (assuming unique k-mers and random, independent mutation), the probability that no mutation will occur in a given k-mer is e^{-kd}, with an expected value equal to the fraction of conserved k-mers w to the total number of k-mers t in the genome, $\frac{w}{t}$. Solving $e^{-kd} = \frac{w}{t}$ gives $d = -\frac{1}{k}\ln\frac{w}{t}$. To account for two genomes of different sizes, Fan et al. [22] set t to the smaller of the two genome's k-mer counts, thereby measuring containment of the k-mer set. In contrast, Mash sets t to the average genome size n, thereby penalizing for genome size differences and measuring resemblance (e.g. to avoid a distance of zero between a phage and a genome containing that phage). Finally, because the Jaccard estimate j can be framed in terms of the average genome size $j = \frac{w}{2n-w}$, the fraction of shared k-mers can be framed in terms of the Jaccard index $\frac{w}{n} = \frac{2j}{1+j}$, yielding the Mash distance:

$$D = -\frac{1}{k}\ln\frac{2j}{1+j} \quad (4)$$

Equation 4 carries many assumptions and does not attempt to model more complex evolutionary processes, but closely approximates the divergence of real genomes (Fig. 2). With appropriate choices of s and k, it can be used as a replacement for costly ANI computations. Table 1 and Additional file 1: Figure S2 give error bounds on the Mash distance for various sketch sizes and Additional file 1: Figure S3 illustrates the relationship between the Jaccard index, Mash distance, k-mer size, and genome size.

Mash *P* value

In the case of distantly related genomes it can be difficult to judge the significance of a given Jaccard index or Mash distance. As illustrated by Eq. 1, for small k and large n there can be a high probability of a random k-mer appearing by chance. How many k-mers then are expected to match between the sketches of two unrelated genomes? This depends on the sketch size and the probability of a random k-mer appearing in the genome, where the expected Jaccard index r between two random genomes X and Y is given by:

$$r = \frac{P(K \in X)P(K \in Y)}{P(K \in X) + P(K \in Y) - P(K \in X)P(K \in Y)} \quad (5)$$

From Eq. 1, the probability of a random k-mer depends both on the size of k, which is known, and total number of k-mers in the genome, which can be estimated from the sketch [48]. For the sketch size s, maximum hash value in the sketch v, and hash bits b, the number of distinct k-mers in the genome is estimated as $n = 2^b s/v$. For the population size m of all distinct k-mers in X and Y and the number of shared k-mers w, where:

$$m = |X \cup Y| = |X| + |Y| - w \quad (6)$$

the probability p of observing x or more matches between the sketches of these two genomes can be

computed using the hypergeometric cumulative distribution function. For the sketch size s, shared size w, and population size m:

$$p(x;s;w;m) = 1 - \sum_{i=0}^{x-1} \frac{\binom{w}{i}\binom{m-w}{s-i}}{\binom{m}{s}} \quad (7)$$

However, because m is typically very large and the sketch size is relatively much smaller, it is more practical to approximate the hypergeometric distribution with the binomial distribution where the expected value of $r = \frac{w}{m}$ can be computed using Eq. 5:

$$p(x;s;r) = 1 - \sum_{i=0}^{x-1} \binom{s}{i} r^i (1-r)^{s-i} \quad (8)$$

Mash uses Eq. 8 to compute the P value of observing a given Mash distance (or less) under the null hypothesis that both genomes are random collections of k-mers. This equation does not account for compositional characteristics like GC bias, but it is useful in practice for ruling out clearly insignificant results (especially for small values of k and j). Interestingly, past work suggests that a random model of k-mer occurrence is not entirely unreasonable [41]. Note, this P value only describes the significance of a single comparison and multiple testing must be considered when searching against a large database.

RefSeq clustering

By default, Mash uses 32-bit hashes for k-mers where $|\Sigma|^k \leq 2^{32}$ and 64-bit hashes for $|\Sigma|^k \leq 2^{64}$. Thus, to minimize the resulting size of the all-RefSeq sketches, $k = 16$ was chosen along with a sketch size $s = 400$. While not ideal for large genomes (due to the small k) or highly divergent genomes (due to the small sketch), these parameters are well suited for determining species-level relationships between the microbial genomes that currently constitute the majority of RefSeq. For similar genomes (e.g. ANI >95 %), sketches of a few hundred hashes are sufficient for basic clustering. As ANI drops further, the Jaccard index rapidly becomes very small and larger sketches are required for accurate estimates. Confidence bounds for the Jaccard estimate can be computed using the inverse cumulative distribution function for the hypergeometric or binomial distributions (Additional file 1: Figure S1). For example, with a sketch size of 400, two genomes with a true Jaccard index of 0.1 ($x = 40$) are very likely to have a Jaccard estimate between 0.075 and 0.125 (P >0.9, binomial density for $30 \leq x \leq 50$). For $k = 16$, this corresponds to a Mash distance between 0.12 and 0.09.

RefSeq Complete release 70 was downloaded from NCBI FTP (ftp://ftp.ncbi.nlm.nih.gov). Using FASTA and Genbank records, replicons and contigs were grouped by organism using a combination of two-letter accession prefix, taxonomy ID, BioProject, BioSample, assembly ID, plasmid ID, and organism name fields to ensure distinct genomes were not combined. In rare cases this strategy resulted in over-separation due to database mislabeling. Plasmids and organelles were grouped with their corresponding nuclear genomes when available; otherwise they were kept as separate entries. Sequences assigned to each resulting "organism" group were combined into multi-FASTA files and chunked for easy parallelization. Each chunk was sketched with:

```
mash sketch -s 400 -k 16 -f -o chunk *.fasta
```

This required 26.1 CPU h on a heterogeneous cluster of AMD processors. (Note: option -f is not required in Mash v1.1.) The resulting, chunked sketch files were combined with the Mash *paste* function to create a single "refseq.msh" file containing all sketches. Each chunked sketch file was then compared against the combined sketch file, again in parallel, using:

```
mash dist -t refseq.msh chunk.msh
```

This required 6.9 CPU h to create pairwise distance tables for all chunks. The resulting chunk tables were concatenated and formatted to create a PHYLIP formatted distance table.

For the ANI comparison, a subset of 500 Escherichia genomes was selected to present a range of distances yet bound the runtime of the comparatively expensive ANI computation. ANI was computed using the MUMmer v3.23 "dnadiff" program and extracting the 1-to-1 "AvgIdentity" field from the resulting report files [49]. The corresponding Mash distances were taken from the all-vs-all distance table as described above.

For the primate phylogeny, the FASTA files were sketched separately, in parallel, taking an average time of 8.9 min each and a maximum time of 11 min (Intel Xeon E5-4620 2.2 GHz processor and solid-state drive). The sketches were combined with Mash *paste* and the combined sketch given to *dist*. These operations took insignificant amounts of time, and table output from *dist* was given to PHYLIP v3.695 [50] *neighbor* to produce the phylogeny. Accessions for all genomes used are given in Additional file 1: Table S1. The UCSC tree was downloaded from [51].

RefSeq search

Each dataset listed in Table 3 was compared against the full RefSeq Mash database using the following command for assemblies:

```
mash dist refseq.msh seq.fasta
```

and the following command for raw reads:

```
mash dist -u refseq.msh seq.fasta
```

which enabled the Bloom filter to remove erroneous, single-copy k-mers. (Note: option -u was replaced by -b in Mash v1.1.) Hits were sorted by distance and all hits within one order of magnitude of the most significant hit ($P \leq 10^{-10}$) were used to compute the lowest common ancestor using an NCBI taxonomy tree. The RefSeq genome with the smallest significant distance, with ties broken by P value, was also reported.

Metagenomic clustering

The Global Ocean Survey (GOS) dataset [35] was downloaded from the iMicrobe FTP site (ftp://ftp.imicrobe.us/projects/26). The full dataset was split into 44 samples corresponding to Table 1 in Rusch et al. [35]. This is the dataset used for benchmarking in the Compareads paper [33] and that analysis was replicated using both Mash and COMMET [34], the successor to Compareads. COMMET v24/07/2014 was run with default parameters (t = 2, m = all, k = 33) as:

```
python Commet.py read_sets.txt
```

where "read_sets.txt" points to the gzipped FASTQ files. This required 34 CPU h (2069 CPU min) and 4 GB of RAM. As suggested by COMMET's author, samples were also truncated to contain the same number of reads to improve runtime (50,980 reads per sample, Nicolas Maillet, personal communication). On this reduced dataset COMMET required 10 CPU h (598 CPU min). The heatmaps were generated in R using the quartile coloring of COMMET [34] (Additional file 1: Supplementary Note 2). Additional file 1: Figure S8 shows the original heatmap generated by COMMET on this dataset. Mash was run as:

```
mash sketch -u -g 3500 -k 21 -s 10000 -o gos *.fa
```

This required 0.6 CPU h (37 CPU min) and 19.6 GB of RAM with Bloom filtering or 8 MB without. (Note: options -u and -g were replaced by -b in Mash v1.1.) The resulting combined sketch file totaled just 3.4 MB in size, compared to the 20 GB FASTA input. Mash distances were computed for all pairs of samples as:

```
mash dist -t gos.msh gos.msh
```

which required less than 1 CPU s to complete.

All available HMP and MetaHIT samples were downloaded: HMP reads [52], HMP assemblies [53], MetaHIT reads (ENA accession ERA000116), and MetaHIT assemblies [54]. This totaled 764 sequencing runs (9.3 TB) and 755 assemblies (60 GB) for HMP and 124 sequencing runs (1.1 TB) and 124 assemblies (10 GB) for MetaHIT. Mash was run in parallel with the same parameters used for the GOS datasets and the resulting sketches merged with Mash *paste*. Sketching the 764 HMP sequencing runs required 259.5 CPU h (average 0.34, max 2.01) and the 755 assemblies required 3.7 CPU h (average 0.005). Sketching the 124 MetaHIT sequencing runs required 20 CPU h (average 0.16, max 0.62), and the 124 assemblies required 0.64 CPU h (average 0.005). COMMET was tested on three read sets (SAMN00038294, SAMN00146305, and SAMN00037421), which were smaller than the average HMP sample size and required an average of 655 CPU s per pairwise comparison. Thus, it was estimated to compare all 888^2 pairs of HMP and MetaHIT samples would require at least 143,471 CPU h. Mash distances were computed for all pairs of samples as before for GOS. This required 3.3 CPU min for both sequencing runs and assemblies. HMP samples that did not pass HMP QC requirements [36] were removed from Fig. 5b, but Additional file 1: Figure S7 shows all HMP assemblies clustered, with several samples that did not pass HMP quality controls included. These samples are the only ones that fail to group by body site. Thus, Mash can also act as an alternate QC method to identify mis-tracked or low-quality samples.

Mash engineering

Mash builds upon the following open-source software packages: kseq [55] for FASTA parsing, Cap'n Proto for serialized output [56], MurmurHash3 for k-mer hashing [57], GNU Scientific Library [58] (GSL) for P value computation, and the Open Bloom Filter Library [59]. All Mash code is licensed with a 3-clause BSD license. If needed, Mash can also be built using the Boost library [60] to avoid the GSL (GPLv3) license requirements. Due to Cap'n Proto requirements, a C++11 compatible compiler is required to build from source, but precompiled binaries are distributed for convenience.

Additional file

Additional file 1: Figure S1. Absolute and relative error bounds for Mash Jaccard estimates given various sketch sizes. **Figure S2.** Error bounds for Mash distance estimate using $k = 16$ and $k = 21$ and various sketch sizes. **Figure S3.** Effect of k-mer and genome size on the Mash distance. **Figure S4.** Eukaryotic components of the RefSeq clustering, colored by taxonomic order. **Figure S5.** Plasmid and organelle components of the RefSeq clustering, colored by taxonomic species. **Figure S6.** Mash tree from Fig. 4 supplemented with five additional mammals. **Figure S7.** Mash clustering of all HMP and MetaHit sample assemblies. **Figure S8.** Raw COMMET output for the GOS dataset. Supplementary Note 1. Supporting data. Supplementary Note 2. Metagenomic heatmap R code. (PDF 8062 kb)

Acknowledgements

The authors thank Konstantin Berlin, Ben Langmead, Michael Schatz, and Nicolas Maillet for their helpful suggestions; Brian Walenz and Torsten Seemann for reviewing the draft; Jiarong Guo, Sherine Awad, C. Titus Brown, and an anonymous referee for their constructive reviews; and Philip Ashton, Aleksey Jironkin, and Nicholas Loman for providing early feedback on the software.

Funding

This research was supported in part by the Intramural Research Program of the National Human Genome Research Institute, National Institutes of Health, and under Contract No. HSHQDC-07-C-00020 awarded by the Department of Homeland Security (DHS) Science and Technology Directorate (S&T) for the

management and operation of the National Biodefense Analysis and Countermeasures Center (NBACC), a Federally Funded Research and Development Center. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the DHS or S&T. In no event shall the DHS, NBACC, S&T, or Battelle National Biodefense Institute (BNBI) have any responsibility or liability for any use, misuse, inability to use, or reliance upon the information contained herein. DHS does not endorse any products or commercial services mentioned in this publication.

Availability of data and materials

Additional file 1 is available with the online version of this paper. This file includes all supplementary figures, tables, and notes referenced in the manuscript. The Oxford Nanopore MinION runs for *B. anthracis* and *B. cereus* are available from the NCBI SRA repository under accessions SRR2671867 and SRR2671868, respectively. All experiments described here were run using Mash v1.0. Mash source code and precompiled binary releases are freely available from https://github.com/marbl/mash under a three-clause BSD license. Mash documentation and additional supporting data are available from http://mash.readthedocs.org. Mash is written in C++ and has been tested on Linux and Mac OS X.

Authors' contributions

AMP conceived the project, designed the methods, and wrote the paper with input from BDO, TJT, SK, and PM. BDO wrote the software and assisted with analyses. TJT led the RefSeq and tree analyses. SK led the search and metagenomic analyses. ABM and NHB performed sequencing experiments. All authors read and approved the final manuscript.

Competing interests

The authors declare that they have no competing interests.

Consent for publication

Not applicable.

Ethics approval and consent to participate

Not applicable.

Author details

[1]National Biodefense Analysis and Countermeasures Center, Frederick, MD, USA. [2]Faculty of Industrial Engineering, Mechanical Engineering and Computer Science, University of Iceland, Reykjavik, Iceland. [3]Genome Informatics Section, Computational and Statistical Genomics Branch, National Human Genome Research Institute, National Institutes of Health, Bethesda, MD, USA.

Received: 31 December 2015 Accepted: 3 June 2016
Published online: 20 June 2016

References

1. Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ. Basic local alignment search tool. J Mol Biol. 1990;215:403–10.
2. GenBank and WGS Statistics. http://www.ncbi.nlm.nih.gov/genbank/statistics. Accessed 31 May 2016.
3. Stephens ZD, Lee SY, Faghri F, Campbell RH, Zhai C, Efron MJ, et al. Big data: astronomical or genomical? PLoS Biol. 2015;13:e1002195.
4. Broder AZ. On the resemblance and containment of documents. Compression and Complexity of Sequences 1997 - Proceedings 1998:21–29.
5. Indyk P, Motwani R. Approximate nearest neighbors: towards removing the curse of dimensionality. In: Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing. Dallas, TX: ACM; 1998.
6. Broder AZ. Identifying and filtering near-duplicate documents. In: COM '00 Proceedings of the 11th Annual Symposium on Combinatorial Pattern Matching. London: Springer; 2000. p. 1–10.
7. Chum O, Philbin J, Zisserman A. Near Duplicate Image Detection: min-Hash and tf-idf Weighting. In: Proceedings of the British Machine Vision Conference 2008. Durham, UK: British Machine Vision Association and Society for Pattern Recognition; 2008.
8. Narayanan M, Karp RM. Gapped local similarity search with provable guarantees. Algorithms in Bioinformatics, Proceedings. 2004;3240:74–86.
9. Berlin K, Koren S, Chin CS, Drake JP, Landolin JM, Phillippy AM. Assembling large genomes with single-molecule sequencing and locality-sensitive hashing. Nat Biotechnol. 2015;33:623–30.
10. Yang X, Zola J, Aluru S. Parallel metagenomic sequence clustering via sketching and maximal quasi-clique enumeration on map-reduce clouds. In: Parallel & Distributed Processing Symposium (IPDPS), 2011 IEEE International. IEEE. 2011. p. 1223–33.
11. Drew J, Hahsler M. Strand: fast sequence comparison using mapreduce and locality sensitive hashing. In: Proceedings of the 5th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics. Newport Beach, CA: ACM; 2014.
12. Rasheed Z, Rangwala H. A Map-Reduce Framework for Clustering Metagenomes. In: 2013 IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum: IEEE. 2013.
13. Vinga S, Almeida J. Alignment-free sequence comparison-a review. Bioinformatics. 2003;19:513–23.
14. Haubold B. Alignment-free phylogenetics and population genetics. Brief Bioinform. 2014;15:407–18.
15. Blaisdell BE. A measure of the similarity of sets of sequences not requiring sequence alignment. Proc Natl Acad Sci U S A. 1986;83:5155–9.
16. Torney DC, Burks C, Davison D, Sirotkin KM. Computation of d2: a measure of sequence dissimilarity. In: Bell GI, Marr TG, editors. Computers and DNA: the proceedings of the Interface between Computation Science and Nucleic Acid Sequencing Workshop, held December 12 to 16, 1988 in Santa Fe, New Mexico. Redwood City: Addison-Wesley Pub. Co; 1990.
17. Lippert RA, Huang H, Waterman MS. Distributional regimes for the number of k-word matches between two random sequences. Proc Natl Acad Sci U S A. 2002;99:13980–9.
18. Yang K, Zhang L. Performance comparison between k-tuple distance and four model-based distances in phylogenetic tree reconstruction. Nucleic Acids Res. 2008;36:e33.
19. Deloger M, El Karoui M, Petit MA. A genomic distance based on MUM indicates discontinuity between most bacterial species and genera. J Bacteriol. 2009;191:91–9.
20. Yi H, Jin L. Co-phylog: an assembly-free phylogenomic approach for closely related organisms. Nucleic Acids Res. 2013;41:e75.
21. Haubold B, Klotzl F, Pfaffelhuber P. andi: fast and accurate estimation of evolutionary distances between closely related genomes. Bioinformatics. 2015;31:1169–75.
22. Fan H, Ives AR, Surget-Groba Y, Cannon CH. An assembly and alignment-free method of phylogeny reconstruction from next-generation sequencing data. BMC Genomics. 2015;16:522.
23. Konstantinidis KT, Tiedje JM. Genomic insights that advance the species definition for prokaryotes. Proc Natl Acad Sci U S A. 2005;102:2567–72.
24. Schatz MC, Phillippy AM. The rise of a digital immune system. Gigascience. 2012;1:4.
25. Pruitt KD, Tatusova T, Brown GR, Maglott DR. NCBI Reference Sequences (RefSeq): current status, new features and genome annotation policy. Nucleic Acids Res. 2012;40:D130–5.
26. Saitou N, Nei M. The neighbor-joining method: a new method for reconstructing phylogenetic trees. Mol Biol Evol. 1987;4:406–25.
27. Miller W, Rosenbloom K, Hardison RC, Hou M, Taylor J, Raney B, et al. 28-way vertebrate alignment and conservation track in the UCSC Genome Browser. Genome Res. 2007;17:1797–808.
28. Perelman P, Johnson WE, Roos C, Seuanez HN, Horvath JE, Moreira MA, et al. A molecular phylogeny of living primates. PLoS Genet. 2011;7:e1001342.
29. Kuhner MK, Felsenstein J. A simulation comparison of phylogeny algorithms under equal and unequal evolutionary rates. Mol Biol Evol. 1994;11:459–68.
30. Loman NJ, Quick J, Simpson JT. A complete bacterial genome assembled de novo using only nanopore sequencing data. Nat Methods. 2015;12:733–5.
31. Song L, Florea L, Langmead B. Lighter: fast and memory-efficient sequencing error correction without counting. Genome Biol. 2014;15:509.
32. Seth S, Valimaki N, Kaski S, Honkela A. Exploration and retrieval of whole-metagenome sequencing samples. Bioinformatics. 2014;30:2471–9.
33. Maillet N, Lemaitre C, Chikhi R, Lavenier D, Peterlongo P. Compareads: comparing huge metagenomic experiments. BMC Bioinformatics. 2012;13 Suppl 19:S10.
34. Maillet N, Collet G, Vannier T, Lavenier D, Peterlongo P. COMMET: comparing and combining multiple metagenomic datasets. In: 2014 IEEE International Conference on Bioinformatics and Biomedicine (BIBM): IEEE. 2014.

35. Rusch DB, Halpern AL, Sutton G, Heidelberg KB, Williamson S, Yooseph S, et al. The Sorcerer II Global Ocean Sampling expedition: northwest Atlantic through eastern tropical Pacific. PLoS Biol. 2007;5:e77.
36. Human Microbiome Project C. Structure, function and diversity of the healthy human microbiome. Nature. 2012;486:207–14.
37. Qin J, Li R, Raes J, Arumugam M, Burgdorf KS, Manichanh C, et al. A human gut microbial gene catalogue established by metagenomic sequencing. Nature. 2010;464:59–65.
38. Freedman MJ, Nissim K, Pinkas B. Efficient private matching and set intersection. Advances in Cryptology - Eurocrypt 2004, Proceedings. 2004;3027:1–19.
39. De Cristofaro E, Faber S, Gasti P, Tsudik G. Genodroid: are privacy-preserving genomic tests ready for prime time? In: Proceedings of the 2012 ACM workshop on Privacy in the electronic society. Raleigh, NC: ACM; 2012.
40. Solomon B, Kingsford C. Large-scale search of transcriptomic read sets with sequence bloom trees. bioRxiv. 2015. doi:10.1101/017087.
41. Fofanov Y, Luo Y, Katili C, Wang J, Belosludtsev Y, Powdrill T, et al. How independent are the appearances of n-mers in different genomes? Bioinformatics. 2004;20:2421–8.
42. Roberts M, Hayes W, Hunt BR, Mount SM, Yorke JA. Reducing storage requirements for biological sequence comparison. Bioinformatics. 2004;20:3363–9.
43. Roberts M, Hunt BR, Yorke JA, Bolanos RA, Delcher AL. A preprocessor for shotgun assembly of large genomes. J Comput Biol. 2004;11:734–52.
44. Deorowicz S, Kokot M, Grabowski S, Debudaj-Grabysz A. KMC 2: fast and resource-frugal k-mer counting. Bioinformatics. 2015;31:1569–76.
45. Wood DE, Salzberg SL. Kraken: ultrafast metagenomic sequence classification using exact alignments. Genome Biol. 2014;15:R46.
46. Patrascu M, Thorup M. The power of simple tabulation hashing. J ACM. 2012;59:14.
47. Ukkonen E. Approximate string-matching with Q-grams and maximal matches. Theor Comput Sci. 1992;92:191–211.
48. Bar-Yossef Z, Jayram TS, Kumar R, Sivakumar D, Trevisan L. Counting distinct elements in a data stream. In: Proceedings of the 6th International Workshop on Randomization and Approximation Techniques. Springer-Verlag; 2002. p. 1–10.
49. Phillippy AM, Schatz MC, Pop M. Genome assembly forensics: finding the elusive mis-assembly. Genome Biol. 2008;9:R55.
50. Felsenstein J. PHYLIP - Phylogeny Inference Package (Version 3.2). Cladistics. 1989;5:164–6.
51. UCSC multiz20way. http://hgdownload.cse.ucsc.edu/goldenPath/hg38/multiz20way/. Accessed 31 May 2016.
52. HMP Illumina WGS Reads. http://hmpdacc.org/HMIWGS/all/. Accessed 31 May 2016.
53. HMP Illumina WGS Assemblies. http://hmpdacc.org/HMASM/all/. Accessed 31 May 2016.
54. MetaHIT assemblies. http://www.bork.embl.de/~arumugam/Qin_et_al_2010/. Accessed 31 May 2016.
55. Li H, Durbin R. Fast and accurate short read alignment with Burrows-Wheeler transform. Bioinformatics. 2009;25:1754–60.
56. Cap'n Proto. https://capnproto.org. Accessed 31 May 2016.
57. MurmurHash3. https://code.google.com/p/smhasher. Accessed 31 May 2016.
58. Gough B. GNU scientific library reference manual. Godalming: Network Theory Ltd.; 2009.
59. Open Bloom Filter Library. https://code.google.com/p/bloom. Accessed 31 May 2016.
60. Siek JG, Lee L-Q, Lumsdaine A. The Boost Graph Library: User Guide and Reference Manual. New York, NY: Pearson Education; 2001.
61. Shannon P, Markiel A, Ozier O, Baliga NS, Wang JT, Ramage D, et al. Cytoscape: a software environment for integrated models of biomolecular interaction networks. Genome Res. 2003;13:2498–504.
62. Kamada T, Kawai S. An algorithm for drawing general undirected graphs. Inform Process Lett. 1989;31:7–15.
63. Bankevich A, Nurk S, Antipov D, Gurevich AA, Dvorkin M, Kulikov AS, et al. SPAdes: a new genome assembly algorithm and its applications to single-cell sequencing. J Comput Biol. 2012;19:455–77.