Fast genome and metagenome distance estimation using MinHash

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

[1] National Biodefense Analysis and Countermeasures Center, Frederick, Maryland, USA
[2] Genome Informatics Section, Computational and Statistical Genomics Branch, National Human Genome Research Institute, National Institutes of Health, Bethesda, Maryland, USA
[*] Corresponding author: adam.phillippy@nih.gov

Abstract

Given a massive collection of sequences, it is infeasible to perform pairwise alignment for basic tasks like sequence clustering and search. To address this problem, we demonstrate that the MinHash technique, first applied to clustering web pages, can be applied to biological sequences with similar effect, and extend this idea to include biologically relevant distance and significance measures. Our new tool, Mash, uses MinHash locality-sensitive hashing to reduce large sequences to a representative sketch and rapidly estimate pairwise distances between genomes or metagenomes. Using Mash, we explored several use cases, including a 5,000-fold size reduction and clustering of all ~55,000 NCBI RefSeq genomes in 46 CPU hours. The resulting 93 MB sketch database includes all RefSeq genomes, effectively delineates known species boundaries, reconstructs approximate phylogenies, and can be searched in seconds using assembled genomes or raw sequencing runs from Illumina, Pacific Biosciences, and Oxford Nanopore. For metagenomics, Mash scales to thousands of samples and can replicate Human Microbiome Project and Global Ocean Survey results in a fraction of the time. Other potential applications include any problem where an approximate, global sequence distance is acceptable, e.g. to triage and cluster sequence data, assign species labels to unknown genomes, quickly identify mis-tracked samples, and search massive genomic databases. In addition, the Mash distance metric is based on simple set intersections, which are compatible with homomorphic encryption schemes. To facilitate integration with other software, Mash is implemented as a lightweight C++ toolkit and freely released under a BSD license at https://github.com/marbl/mash.

Introduction

When BLAST was first published in 1990[1], there were less than 50 million bases of nucleotide sequence in the public archives (http://www.ncbi.nlm.nih.gov/genbank/statistics); now a single sequencing instrument can produce over 1 trillion bases per run[2]. 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[3] to reduce large sequences (or sequence sets) to compressed sketch representations. Using only the sketches, which can be thousands of fold 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.

The MinHash technique is a form of locality-sensitive hashing[4] that has been widely used for the detection of near-duplicate Web pages and images[5, 6], but has seen limited use in genomics despite initial applications over ten years ago[7]. More recently, MinHash has been applied to the relevant problems of genome assembly[8], 16S rDNA gene clustering[9], and metagenomic sequence clustering[10]. 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 the Mash toolkit for flexible construction, manipulation, and comparison of MinHash sketches from genomic data. We build upon past applications of MinHash by deriving a new significance test and distance metric, the Mash distance, which estimates a simple evolutionary distance. Similar 'alignment-free' methods have a long history in bioinformatics[11, 12]. However, methods based on string matching must process the entire sequence with each comparison[13-16], while methods based on short word counts have lacked the ability to differentiate closely related sequences[17-20]. In contrast, the Mash distance can be quickly computed from the size-reduced sketches alone, producing a result that strongly correlates with Average Nucleotide Identity (ANI)[21]. Thus, Mash combines the high specificity of matching-based approaches with the dimensionality reduction of statistical approaches.

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 (Figure 1). The *dist* function compares two sketches and returns for the original sequences 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[16] (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, 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.

Results

Clustering all genomes in NCBI RefSeq

Mash enables scalable whole-genome clustering, which is an important application for the future of genomic data management. As genome databases increase in size, and whole-genome sequencing becomes routine, it will become infeasible to manually assign taxonomic labels for all genomes. Thus, generalized and automated methods will be useful for constructing and partitioning groups of related genomes, e.g. for the automated detection of outbreak clusters[22]. To illustrate the utility of Mash, we sketched and clustered all of NCBI RefSeq Release 70[23], totaling 54,118 organisms and 618 Gbp of genomic sequence. The resulting sketches total only 93 MB, yielding a compression factor of >5,000-fold versus the uncompressed FASTA (674 GB). Sketching all genomes and computing all ~1.5 billion pairwise distances required just 26.1 and 20.3 CPU hours, respectively (the sketch database is provided as Supplementary Data 1). 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 seconds per 5 MB genome (or 4 CPU minutes for a 3 GB genome).

The resulting Mash distances correlate well with ANI, with $D \approx 1-\text{ANI}$ over multiple sketch sizes and sizes of k (Figure 2). Due to the high cost of computing ANI, a subset of 500 Escherichia genomes was selected for comparison (Supplementary Table 1). For ANI in the range 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=1,000 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 Mash estimate becomes more variable and penalizes for genome size differences, whereas ANI is based solely on the core genome. Increasing the sketch size reduces the variance of the Mash estimates, especially for divergent genomes (Supplementary Figures 1 and 2), while 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 (Supplementary Figure 3). 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 (Methods); however, too large of a k-mer size could result in no shared k-mers found.

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 value—a historical, albeit debatable, definition of bacterial species[21]. 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 Supplementary Figures 4 and 5, 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 hours (11 minutes wall clock on 17 cores) with default parameters (s=1,000 and k=21) and used to build a neighbor-joining tree[24]. Figure 4 compares this tree to an alignment-based phylogenetic tree model downloaded from the UCSC genome browser[25]. 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[26] and mitochondrial trees[12]. On average the Mash branch lengths are slightly longer, with a Branch Score Distance[27] of 0.10 between the two trees, but additional distance corrections are possible for k-mer based models[16]. 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 large size differences, and should be used only in cases where such approximations are sufficient.

Rapid genome identification from assemblies or reads

With a pre-computed sketch database, Mash is able to rapidly identify unknown genomes from both genome assemblies and raw sequencing reads. To illustrate, we computed Mash distances for multiple *Escherichia coli* datasets compared against the RefSeq sketch database (Table 1). 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 in a few seconds using Lowest Common Ancestor (LCA) classification. For raw sequencing reads, the correct species was identified in all cases, including 1D MinION reads[28], which had an average sequencing error rate of ~40%. The reduced resolution obtained when using raw sequences is due to erroneous k-mers, which introduce noise into the sketch. To mitigate this problem, Mash uses a streaming Bloom filter to probabilistically ignore single-copy k-mers from raw reads sets, but some fraction of erroneous k-mers will persist and skew the Mash distance. Increasing the sketch and Bloom filter sizes will improve accuracy when dealing with raw sequencing data, as well as read trimming and/or correction.

To further test Mash's discriminatory power, we searched MinION reads collected from two closely related Bacillus species (ANI≈95%) against the full RefSeq sketch database. In both cases Mash was able to correctly identify the species, using 43,806 and 91,379 sequences collected from single MinION R7.3 runs of *Bacillus anthracis* Ames and *Bacillus 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 with simple LCA classification. These two searches both required just one minute of

CPU and 209 MB of RAM. Such low-overhead searches could be used for quickly triaging unknown samples or to rapidly select a reference genome for performing further, more detailed comparative analyses.

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[29]. By definition, Mash rapidly approximates this result by filtering unique k-mers with a Bloom filter and estimating the Jaccard index via MinHashing. COMMET also uses k-mers to approximate similarity, but attempts to identify a set of similar reads between two samples using Bloom filters[30, 31]. 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. Figure 5a replicates the analysis in Maillet *et al.*[30] using both Mash and COMMET to cluster Global Ocean Survey (GOS) data[32]. Mash is over 50-fold 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, a new GOS sample could be added to the Mash distance table in less than a minute, compared to an hour required for COMMET, making Mash ideal for real-time sample analysis.

For a large-scale test, samples from the Human Microbiome Project[33] (HMP) and Metagenomics of the Human Intestinal Tract[34] (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 hours to process all assemblies and 279.6 CPU hours to process all read sets. The two clusterings are remarkably similar, with all samples clearly grouped by body site (Figure 5b). However, because the Mash distance is based on k-mer sets, it is not sensitive to changes in relative abundance and may be more prone to batch effects, such as sequencing error rate. For example, Mash does not cluster MetaHIT samples by health status, as previously reported[34], and MetaHIT samples appear to preferentially cluster with one another. It is not clear if this reflects true sample differences (e.g. American vs. European stool) or batch effect. 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 (Supplementary Figure 6).

Discussion

Mash enables the comparison and clustering of whole genomes 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. The strong correlation between Mash distance and ANI promise 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[35], enabling privacy-preserving genomic tests[36].

Future applications of Mash could include read mapping and metagenomic sequence classification via windowed sketches or a containment score[3] to test for the presence of one sequence within another. 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[37]. Finally, both the *sketch* and *dist* functions 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 has 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[22].

METHODS

The Mash version 1.0 codebase is provided as Supplementary Data 2. Precompiled binary releases and source code updates are available from https://github.com/marbl/mash.

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 uses a "bottom sketch" strategy returning the s smallest hashes output by h over all k-mers in the genome (Figure 1). For a sketch size s and genome size n, this 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 ith hash of the genome will enter the sketch is s/i, so the expected runtime of the algorithm is $O(n + s \log s \log n)$ (Ref. [3]), which becomes nearly linear when $n >> s$.

As demonstrated by Figure 3, a sketch comprising 400 32-bit hash values is sufficient to roughly group microbial genomes by species. With these parameters, the sketch for a ~3 billion base-pair

human genome represents a million-fold (lossy) compression. However, the probability of a given k-mer K appearing in a random genome X of size n is:

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

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[38]:

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

which yields k=14 and k=19 for 5 Mbp and 3 Gbp genomes (q=0.01), respectively. We have found k=21 gives accurate estimates in most cases (including metagenomes), so this is set as the default. 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 with noisy data like single-molecule sequencing (Supplementary Figure 3).

Lastly, for sketching raw sequencing reads, Mash provides a streaming Bloom filter to remove erroneous k-mers. This approach assumes 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 once. To probabilistically exclude unique k-mers from the sketch, new hashes are only inserted into the sketch if they are 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.

Mash distance

A MinHash sketch of size s=1 is equivalent to the subsequent 'minimizer' concept of Roberts *et al.*[39], which has been used in genome assembly[40], k-mer counting[41], and metagenomics[42]. 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[3]. 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:

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

which is an unbiased estimate of the true Jaccard index, as illustrated in Figure 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[43]. Specific confidence intervals are given below and in Supplementary Figure 1. 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 Average Nucleotide Identity (ANI), the most common metric used to describe global sequence similarity. However, like the MUM index[13], 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[44]. 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.*[16], 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 $e^{-kd} = \frac{w}{t}$ for d gives $d = -\frac{1}{k}\ln\frac{w}{t}$. To account for two genomes of different sizes, Fan *et al.*[16] set t to the smaller of the two genome's k-mer counts, thereby measuring containment of the k-mer set. However, 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:

$$D = -\frac{1}{k}\ln\frac{2j}{1+j} \tag{4}$$

Equation 4 carries many assumptions and does not attempt to model more complex evolutionary processes, but closely approximates the divergence of real genomes (Figure 2). With appropriate choices of s and k, it can be used as a replacement for costly ANI computations. Supplementary

Figure 2 gives 99% confidence bounds on the Mash distance for various sketch sizes, and Supplementary Figure 3 illustrates the relationship between the Jaccard index, Mash distance, k-mer size, and genome size.

Mash *P*-value

In cases of distantly related genomes it can be difficult to judge the significance of a given Jaccard index or Mash distance. As illustrated by Equation 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 random genomes? This depends on the sketch size and the probability of a random k-mer appearing in the genome. From Equation 1, the expected Jaccard index r between two random genomes X and Y is given by:

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

Knowing the expected population size m of all distinct k-mers in X and Y:

$$m = |X \cup Y| = |X| + |Y| - r|X \cup Y| = \frac{|X| + |Y|}{(1 + r)}$$
$$m = \frac{|\Sigma|^k \big(P(K \in X) + P(K \in Y)\big)}{(1 + r)} \tag{6}$$

The probability p of observing x or more matches between the sketches of these two random genomes can be computed using the hypergeometric cumulative distribution function. For the sketch size s, expected Jaccard index r, and expected population size m:

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

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:

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

Mash uses Equation 8 to compute the P-value of observing a given Mash distance (or less) under the null hypothesis of both genomes being purely random with uniform character frequencies. Of course biological sequences are not random and 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). Note, this 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 (Supplementary Figure 1). 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 0.09 and 0.12.

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 hours on a heterogeneous cluster of AMD processors. 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 20.3 CPU hours to create pairwise distance tables for each chunk. 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 were selected to present a range of distances yet bound the runtime of the comparatively expensive ANI computation (Supplementary Table 1). ANI was computed using MUMmer's 'dnadiff' program and extracting the 1-to-1 'AvgIdentity' field from the resulting report files[45]. 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 minutes each and a maximum time of 11 minutes (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[46] *neighbor* to produce the phylogeny. Accessions for the 17 genomes used are given in Supplementary Table 2. The UCSC tree was downloaded from:

http://hgdownload.cse.ucsc.edu/goldenPath/hg38/multiz20way/

RefSeq search

Each dataset listed in Table 1 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. 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 smallest significant distance, with ties broken by *P*-value, was also reported.

Metagenomic clustering

The Global Ocean Survey (GOS) dataset[32] 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.*[32]. This is the dataset used for benchmarking in the Compareads paper[30], and that analysis was replicated using both Mash and COMMET[31], the successor to Compareads. COMMET 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 hours (2,069 CPU minutes) and 4 GB of RAM. The heatmaps were generated in R using the quartile coloring of COMMET[31] (Supplementary Note 1). Supplementary Figure 7 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 hours (37 CPU minutes) and 19.6 GB of RAM (or 8 MB without Bloom filtering). 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 second to complete.

All available HMP and MetaHIT samples were downloaded from:

http://downloads.hmpdacc.org/data/Illumina/ (HMP reads)

http://downloads.hmpdacc.org/data/HMASM/ (HMP assemblies)

ftp://ftp.sra.ebi.ac.uk/vol1/ERA000/ERA000116/fastq/ (MetaHIT reads)

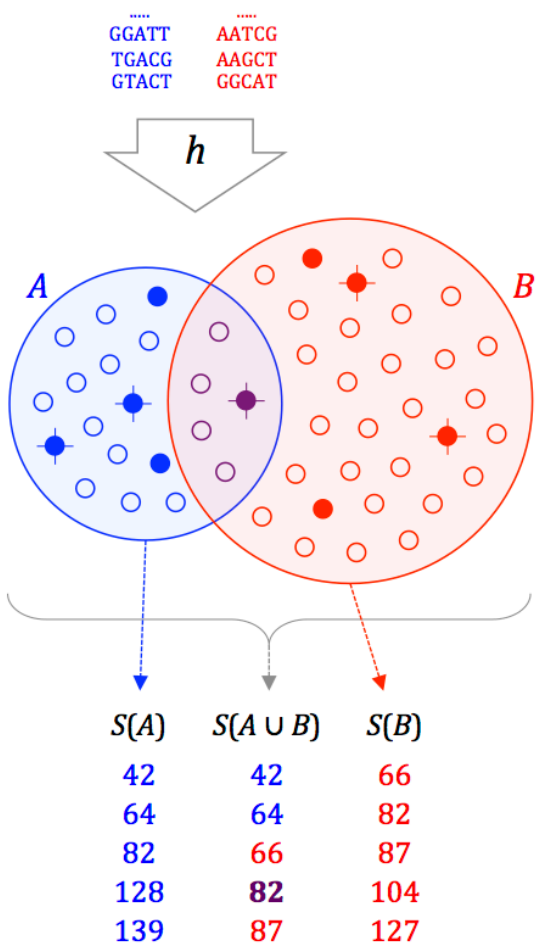
http://www.bork.embl.de/~arumugam/Qin_et_al_2010/ (MetaHIT assemblies)

totaling 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 hours (average 0.34, max 2.01), and the 755 assemblies required 3.7 CPU hours (average 0.005). Sketching the 124 MetIDBA sequencing runs required 20 CPU hours (average 0.16, max 0.62), and the 124 assemblies required 0.64 CPU hours (average 0.005). Mash distances were computed for all pairs of samples as before for GOS. This required 3.3 CPU minutes for both sequencing runs and assemblies. HMP samples that did not pass HMP QC requirements[33] were removed from Figure 5b, but Supplementary Figure 6 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[47] for FASTA parsing, Cap'n Proto for serialized output (https://capnproto.org), MurmurHash3 for k-mer hashing (https://code.google.com/p/smhasher), GNU Scientific Library[48] (GSL) for *P*-value computation, and the 'bloom' Bloom filter library (https://code.google.com/p/bloom). All Mash code is licensed with a 3-clause BSD license. If needed, Mash can also be built using the Boost library[49] 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.

Figures and Tables



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

Figure 1. Overview of the MinHash bottom sketch strategy for estimating the Jaccard index. First, two sequences 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 5 smallest hash values for each (filled circles). Merging *S*(*A*) and *S*(*B*) to recover the 5 smallest hash values overall for *A*∪*B* (crossed circles) yields *S*(*A*∪*B*). The fraction of elements in *S*(*A*∪*B*) that are shared by both *S*(*A*) and *S*(*B*) forms an unbiased estimate of *J*(*A*,*B*).

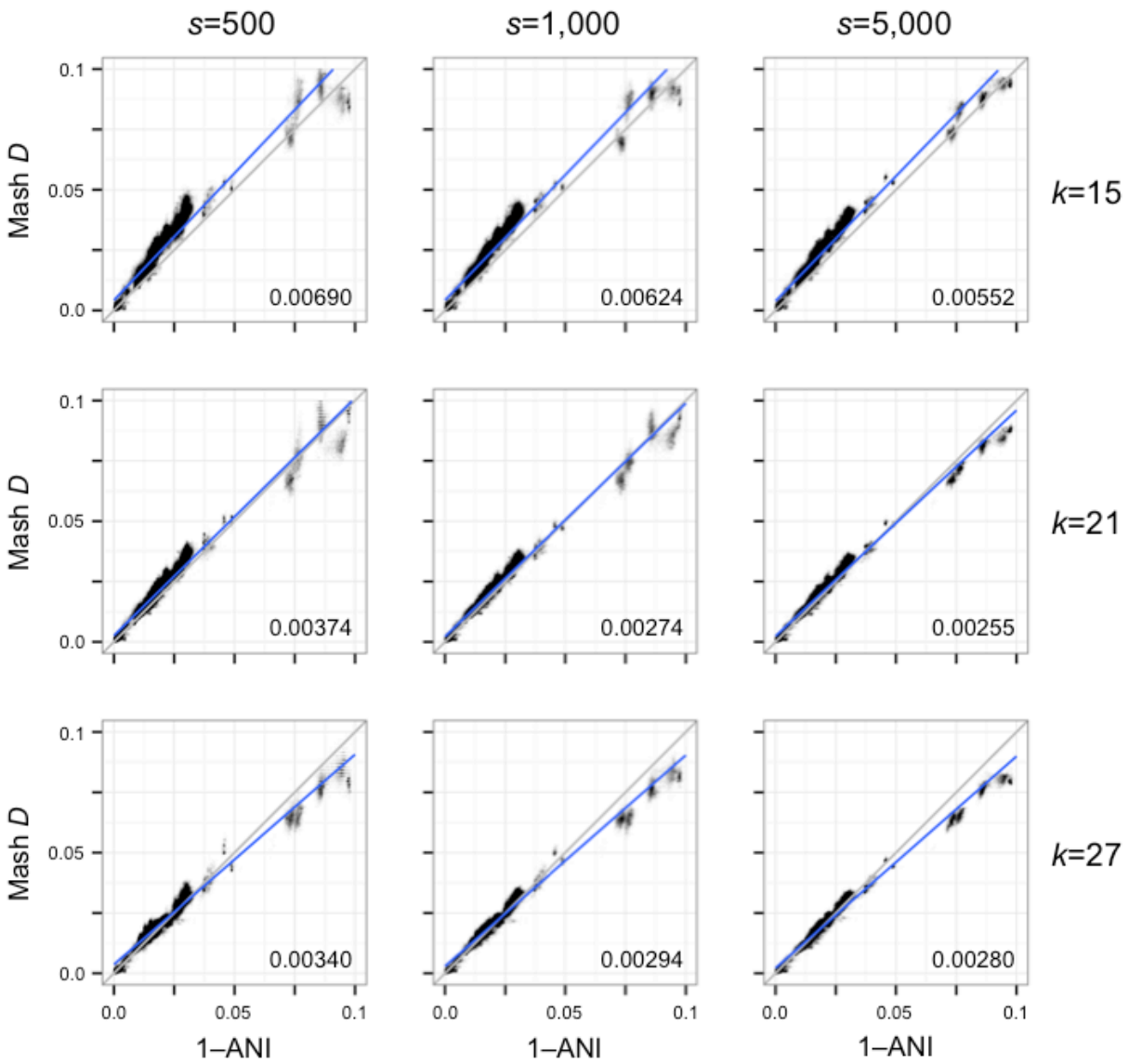


Figure 2. Scatterplots illustrating the relationship between Average Nucleotide Identity and Mash distance for a collection of Escherichia genomes. Each plot column shows a different sketch size *s*, and each plot row a different k-mer size *k*. Gray lines show the model relationship *D*=1–ANI, and numbers 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 lowers the variance 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).

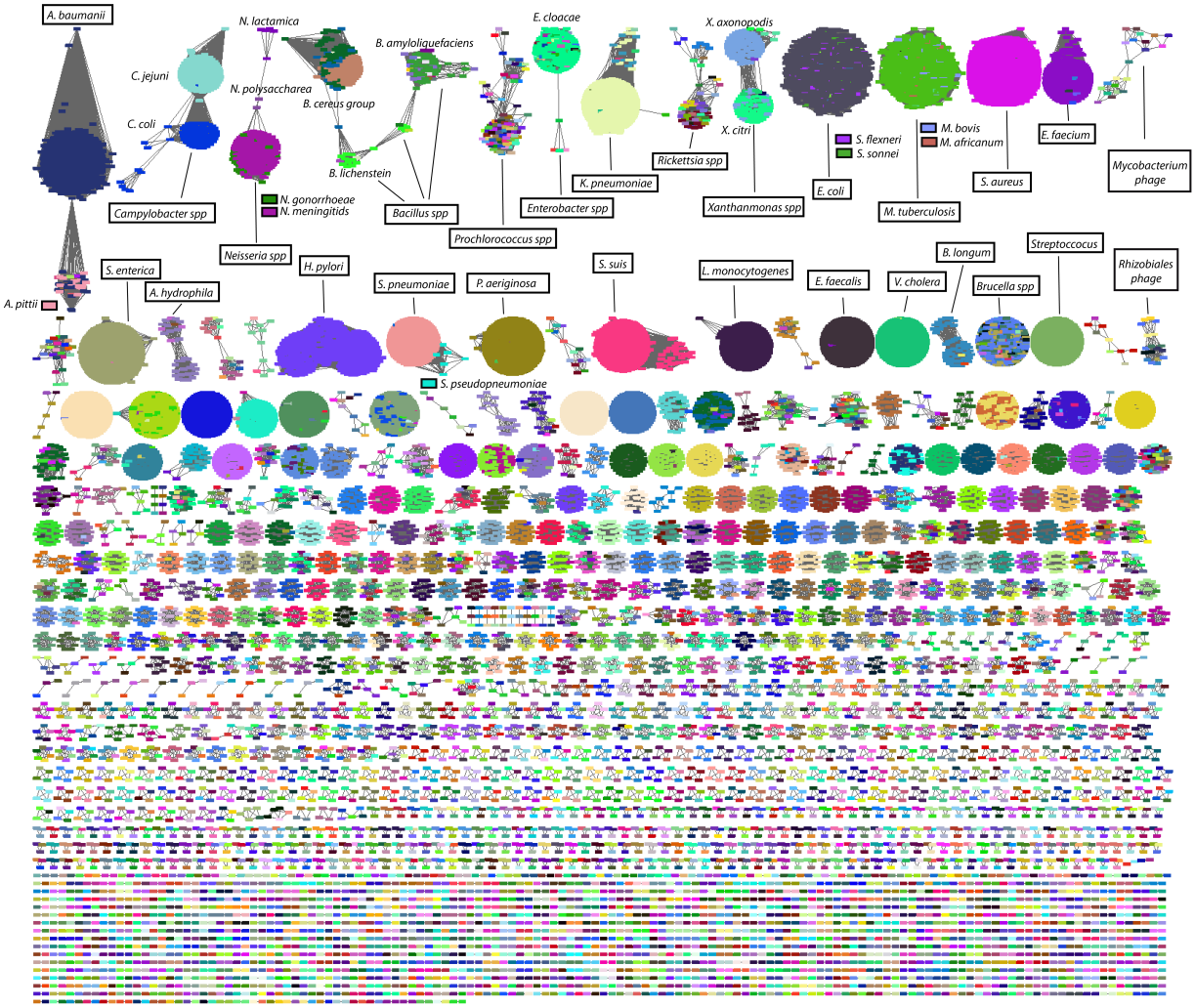


Figure 3. Comparison and *de novo* clustering of all RefSeq genomes using Mash. Each graph node represents a genome. Two genomes are connected by an edge if their Mash distance $D \leq 0.05$ and P-value$\leq 10^{-10}$. Graph layout was performed using Cytoscape[50] organic layout algorithm[51]. Individual 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). Supplementary Figures 4 and 5 show eukaryotes, orphan plasmids, and organelles.

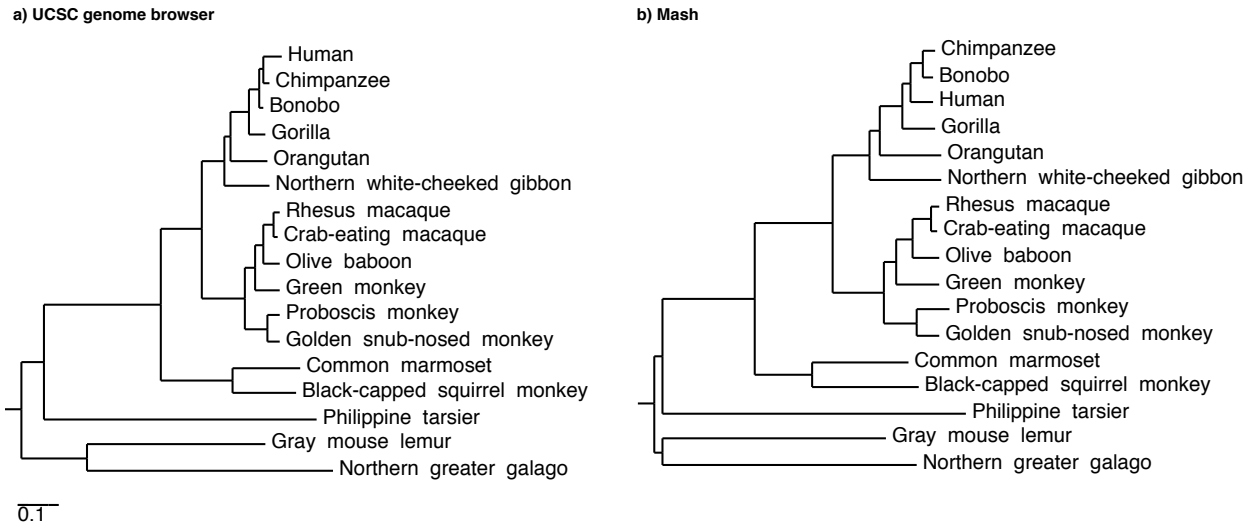


Figure 4. Primate trees from the UCSC genome browser and Mash. (a) A primate phylogenetic tree model from the UCSC genome browser, with branch lengths derived from 4-fold degenerate sites extracted from reference gene multiple alignments. **(b)** A comparable Mash-based tree generated from whole genomes using a sketch size of *s*=1,000 and k-mer size of *k*=21.

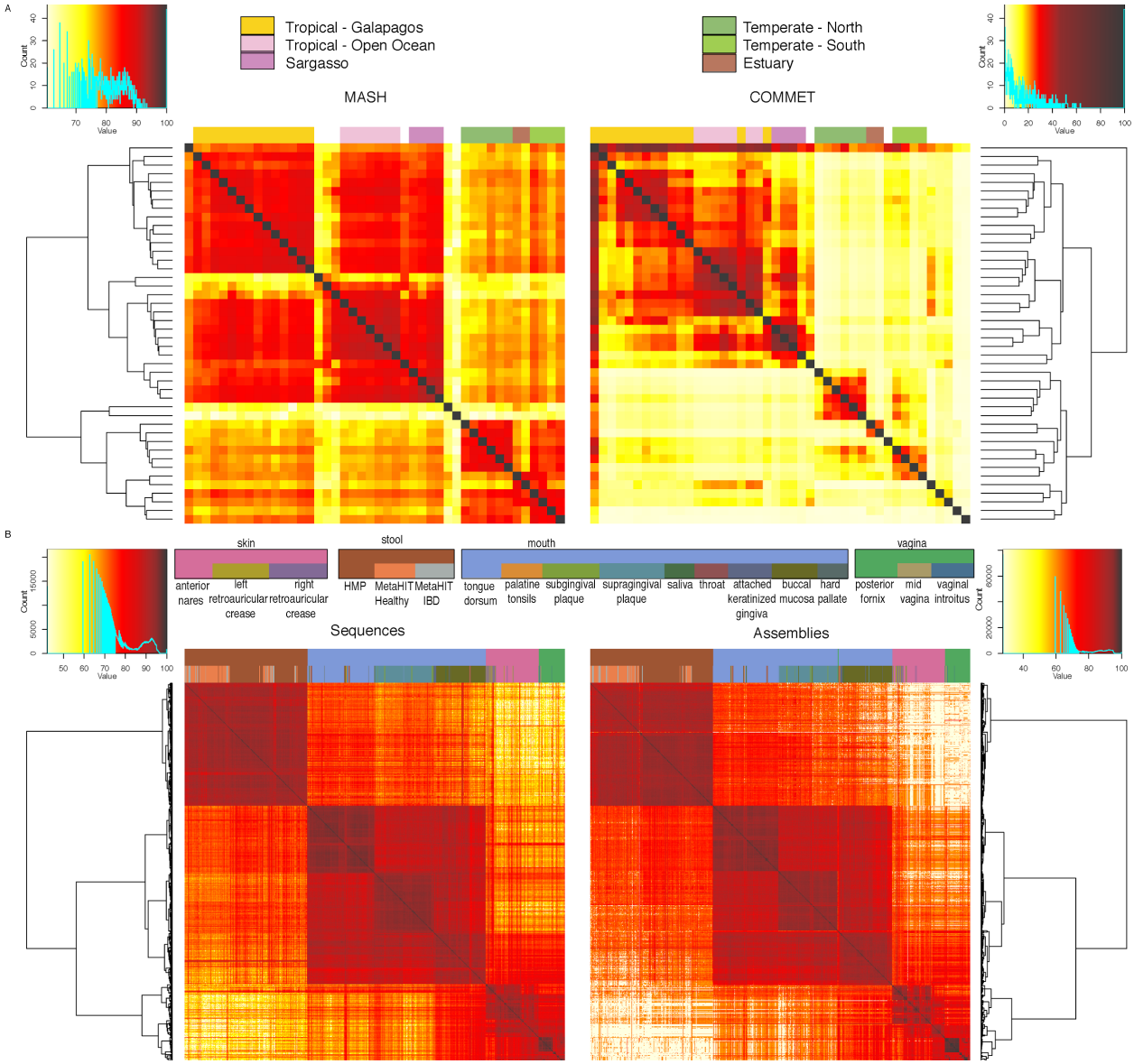


Figure 5. Metagenomic clustering of ocean and human metagenomes using Mash. (a) Comparison of Global Ocean Survey (GOS) clustering using Mash (top left) and COMMET (top right) using raw Sanger sequencing data. Heat maps illustrate the pairwise similarity between samples, scaled between 0 (white) and 100 (red) for comparison to COMMET. Sample groups are identified and colored using the same key as in Rusch *et al.*[32]. 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.

Table 1. Sequencing runs and assemblies searched against the Mash RefSeq database.

Organism	Technology	Type	NCBI ID	Size (Mbp)	Time (s)	LCA	Best Hit
E. coli K12 MG1655	Illumina MiSeq	Assembly SPAdes	N/A	4.6	2.45	Entero.	*E. coli* K12 MG1655
E. coli K12 MG1655	PacBio RSII	Assembly CA/PBcR	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 GS FLX	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	Illumina MiSeq	Reads	SRR1770413	387	72.01	Entero.	*E. coli* KLY
E. coli K12 MT203	Illumina HiSeq	Reads	SRR490124	2,155	369.86	*E. coli*	*E. coli* GCF_000833635
E. coli K12 MG1655	PacBio RSII	Reads	SRR1284073	397	77.96	*E. coli*	*E. coli* XH140A GCF_000226585
E. coli K12 MG1655	Oxford Nanopore MinION	Reads 1D	ERX708228 ERX708229 ERX708230 ERX708231	248	55.52	Entero.	*E. coli* O113 H21
E. coli K12 MG1655	Oxford Nanopore MinION	Reads 2D	ERX708228 ERX708229 ERX708230 ERX708231	134	27.82	*E. coli*	*E. coli* GCF_000953515
B. anthracis Ames	Oxford Nanopore MinION	Reads 1D+2D	SRS1117538	210	44.66	*B. anthracis*	*B. anthracis* str. Carbosap
B. cereus ATCC 10987	Oxford Nanopore MinION	Reads 1D+2D	SRS1117539	266	76.85	*B. cereus* ATCC 10987	*B. cereus* ATCC 10987

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). *Size:* dataset size in Mbp. *LCA:* gives the 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 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.

ACKNOWLEDGEMENTS

The authors thank Konstantin Berlin for helpful discussions; Brian Walenz and Torsten Seemann for reviewing the draft; and Philip Ashton, Aleksey Jironkin, and Nicholas Loman for providing early feedback on the software. 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.

AUTHOR CONTRIBUTIONS

AMP conceived the project, designed the methods, and wrote the paper with input from BDO, TJT, and SK. 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.

COMPETING FINANCIAL INTERESTS

The authors have no financial interests to declare.

REFERENCES

1. Altschul, S.F., Gish, W., Miller, W., Myers, E.W. & Lipman, D.J. Basic local alignment search tool. *Journal of molecular biology* **215**, 403-410 (1990).
2. Stephens, Z.D. et al. Big Data: Astronomical or Genomical? *PLoS biology* **13**, e1002195 (2015).
3. Broder, A.Z. On the resemblance and containment of documents. *Compression and Complexity of Sequences 1997 - Proceedings*, 21-29 (1998).
4. Indyk, P. & Motwani, R. in Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing (ACM, Dallas, Texas, USA; 1998).
5. Broder, A.Z. Identifying and filtering near-duplicate documents. *Lect Notes Comput Sc* **1848**, 1-10 (2000).
6. Chum, O., Philbin, J. & Zisserman, A. in BMVC, Vol. 810 (2008).
7. Narayanan, M. & Karp, R.M. Gapped local similarity search with provable guarantees. *Algorithms in Bioinformatics, Proceedings* **3240**, 74-86 (2004).
8. Berlin, K. et al. Assembling large genomes with single-molecule sequencing and locality-sensitive hashing. *Nature biotechnology* **33**, 623-630 (2015).

9. Drew, J. & Hahsler, M. in Proceedings of the 5th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics (ACM, Newport Beach, CA, USA; 2014).
10. Rasheed, Z. & Rangwala, H. in 2013 IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum (IEEE, 2013).
11. Vinga, S. & Almeida, J. Alignment-free sequence comparison-a review. *Bioinformatics* **19**, 513-523 (2003).
12. Haubold, B. Alignment-free phylogenetics and population genetics. *Brief Bioinform* **15**, 407-418 (2014).
13. Deloger, M., El Karoui, M. & Petit, M.A. A genomic distance based on MUM indicates discontinuity between most bacterial species and genera. *Journal of bacteriology* **191**, 91-99 (2009).
14. Yi, H. & Jin, L. Co-phylog: an assembly-free phylogenomic approach for closely related organisms. *Nucleic acids research* **41**, e75 (2013).
15. Haubold, B., Klotzl, F. & Pfaffelhuber, P. andi: fast and accurate estimation of evolutionary distances between closely related genomes. *Bioinformatics* **31**, 1169-1175 (2015).
16. Fan, H., Ives, A.R., Surget-Groba, Y. & Cannon, C.H. An assembly and alignment-free method of phylogeny reconstruction from next-generation sequencing data. *BMC genomics* **16**, 522 (2015).
17. Blaisdell, B.E. A measure of the similarity of sets of sequences not requiring sequence alignment. *Proceedings of the National Academy of Sciences of the United States of America* **83**, 5155-5159 (1986).
18. Torney, D.C., Burks, C., Davison, D. & Sirotkin, K.M. in 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/edited by George I. Bell, Thomas G. Marr (Redwood City, Calif.: Addison-Wesley Pub. Co., 1990., 1990).
19. Lippert, R.A., Huang, H. & Waterman, M.S. Distributional regimes for the number of k-word matches between two random sequences. *Proceedings of the National Academy of Sciences of the United States of America* **99**, 13980-13989 (2002).
20. Yang, K. & Zhang, L. Performance comparison between k-tuple distance and four model-based distances in phylogenetic tree reconstruction. *Nucleic acids research* **36**, e33 (2008).
21. Konstantinidis, K.T. & Tiedje, J.M. Genomic insights that advance the species definition for prokaryotes. *Proceedings of the National Academy of Sciences of the United States of America* **102**, 2567-2572 (2005).
22. Schatz, M.C. & Phillippy, A.M. The rise of a digital immune system. *GigaScience* **1**, 4 (2012).
23. Pruitt, K.D., Tatusova, T., Brown, G.R. & Maglott, D.R. NCBI Reference Sequences (RefSeq): current status, new features and genome annotation policy. *Nucleic acids research* **40**, D130-135 (2012).
24. Saitou, N. & Nei, M. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Mol Biol Evol* **4**, 406-425 (1987).
25. Miller, W. et al. 28-way vertebrate alignment and conservation track in the UCSC Genome Browser. *Genome research* **17**, 1797-1808 (2007).

26. Perelman, P. et al. A molecular phylogeny of living primates. *PLoS Genet* **7**, e1001342 (2011).
27. Kuhner, M.K. & Felsenstein, J. A simulation comparison of phylogeny algorithms under equal and unequal evolutionary rates. *Mol Biol Evol* **11**, 459-468 (1994).
28. Loman, N.J., Quick, J. & Simpson, J.T. A complete bacterial genome assembled de novo using only nanopore sequencing data. *Nat Methods* **12**, 733-735 (2015).
29. Seth, S., Valimaki, N., Kaski, S. & Honkela, A. Exploration and retrieval of whole-metagenome sequencing samples. *Bioinformatics* **30**, 2471-2479 (2014).
30. Maillet, N., Lemaitre, C., Chikhi, R., Lavenier, D. & Peterlongo, P. Compareads: comparing huge metagenomic experiments. *BMC bioinformatics* **13 Suppl 19**, S10 (2012).
31. Maillet, N., Collet, G., Vannier, T., Lavenier, D. & Peterlongo, P. in 2014 IEEE International Conference on Bioinformatics and Biomedicine (BIBM) (IEEE, 2014).
32. Rusch, D.B. et al. The Sorcerer II Global Ocean Sampling expedition: northwest Atlantic through eastern tropical Pacific. *PLoS biology* **5**, e77 (2007).
33. Human Microbiome Project, C. Structure, function and diversity of the healthy human microbiome. *Nature* **486**, 207-214 (2012).
34. Qin, J. et al. A human gut microbial gene catalogue established by metagenomic sequencing. *Nature* **464**, 59-65 (2010).
35. Freedman, M.J., Nissim, K. & Pinkas, B. Efficient private matching and set intersection. *Advances in Cryptology - Eurocrypt 2004, Proceedings* **3027**, 1-19 (2004).
36. De Cristofaro, E., Faber, S., Gasti, P. & Tsudik, G. in Proceedings of the 2012 ACM workshop on Privacy in the electronic society (ACM, Raleigh, North Carolina, USA; 2012).
37. Solomon, B. & Kingsford, C. Large-Scale Search of Transcriptomic Read Sets with Sequence Bloom Trees. *bioRxiv* (2015).
38. Fofanov, Y. et al. How independent are the appearances of n-mers in different genomes? *Bioinformatics* **20**, 2421-2428 (2004).
39. Roberts, M., Hayes, W., Hunt, B.R., Mount, S.M. & Yorke, J.A. Reducing storage requirements for biological sequence comparison. *Bioinformatics* **20**, 3363-3369 (2004).
40. Roberts, M., Hunt, B.R., Yorke, J.A., Bolanos, R.A. & Delcher, A.L. A preprocessor for shotgun assembly of large genomes. *J Comput Biol* **11**, 734-752 (2004).
41. Deorowicz, S., Kokot, M., Grabowski, S. & Debudaj-Grabysz, A. KMC 2: fast and resource-frugal k-mer counting. *Bioinformatics* **31**, 1569-1576 (2015).
42. Wood, D.E. & Salzberg, S.L. Kraken: ultrafast metagenomic sequence classification using exact alignments. *Genome Biol* **15** (2014).
43. Patrascu, M. & Thorup, M. The Power of Simple Tabulation Hashing. *J Acm* **59** (2012).
44. Ukkonen, E. Approximate String-Matching with Q-Grams and Maximal Matches. *Theor Comput Sci* **92**, 191-211 (1992).
45. Phillippy, A.M., Schatz, M.C. & Pop, M. Genome assembly forensics: finding the elusive mis-assembly. *Genome Biol* **9**, R55 (2008).
46. Felsenstein, J. PHYLIP - Phylogeny Inference Package (Version 3.2). *Cladistics* **5**, 164-166 (1989).
47. Li, H. & Durbin, R. Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics* **25**, 1754-1760 (2009).
48. Gough, B. GNU scientific library reference manual. (Network Theory Ltd., 2009).

49. Siek, J.G., Lee, L.-Q. & Lumsdaine, A. Boost Graph Library: User Guide and Reference Manual, The. (Pearson Education, 2001).
50. Shannon, P. et al. Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome research* **13**, 2498-2504 (2003).
51. Kamada, T. & Kawai, S. An Algorithm for Drawing General Undirected Graphs. *Inform Process Lett* **31**, 7-15 (1989).