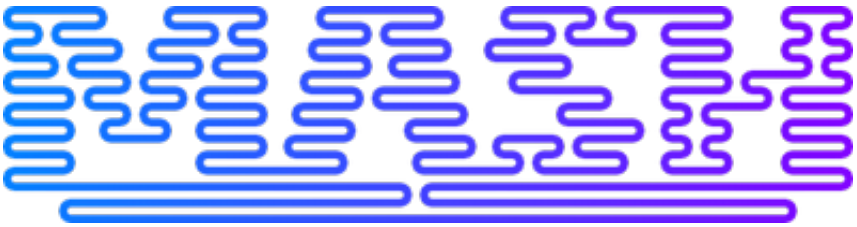
mash Documentation

Release 2.0

Brian Ondov, Todd Treangen, Adam Phillippy

Nov 13, 2019

Contents



Fast genome and metagenome distance estimation using MinHash

Mash: fast genome and metagenome distance estimation using MinHash. Ondov BD, Treangen TJ, Melsted P, Mallonee AB, Bergman NH, Koren S, Phillippy AM. Genome Biol. 2016 Jun 20;17(1):132. doi: 10.1186/s13059-016-0997-x.

Mash Screen: High-throughput sequence containment estimation for genome discovery. Ondov BD, Starrett GJ, Sappington A, Kostic A, Koren S, Buck CB, Phillippy AM. BioRxiv. 2019 Mar. doi: 10.1101/557314

CHAPTER 1

Supporting Data

1.1 Mash: fast genome and metagenome distance estimation using MinHash

RefSeqSketches.msh.gz: Mash sketch database (k=16, s=400) for RefSeq release 70 (48MB)

RefSeqSketchesDefaults.msh.gz: Mash sketch database (k=21, s=1000) for RefSeq release 70 (255MB)

Escherichia.tar.gz: Names and accessions for 500 selected Escherichia genomes, pairwise ANI, and pairwise Jaccard indexes for various k-mer and sketch sizes (24MB)

mash-1.0.tar.gz: Mash version 1.0 codebase (93KB)

SRR2671867.BaAmes.poretools.fastq.gz: Nanopore 1D + 2D sequences generated by poretools (157MB)

SRR2671868.Bc10987.poretools.fastq.gz: Nanopore 1D + 2D sequences generated by poretools (250MB)

1.2 Mash Screen: High-throughput sequence containment estimation for genome discovery

1.2.1 Custom scripts and intermediate data:

MashScreen_supp.tar.gz

1.2.2 Data files:

Mash Sketch databases for RefSeq release 88:

- RefSeq88n.msh.gz: Genomes (k=21, s=1000), 1.2Gb uncompressed
- RefSeq88p.msh.gz: Proteomes (k=9, s=1000), 1.1Gb uncompressed

art.fastq.gz: Simulated reads for Shakya experiment

Figure 5:

- fig5.html: Interactive version
- fig5.tsv: Source data

1.2.3 Screen of SRA metagenomes vs. RefSeq

- sra_meta_nucl_95idy.tsv.gz (2.3Gb uncompressed)
- sra_meta_nucl_80idy_3x.tsv.gz (6.7Gb uncompressed)
- sra_meta_prot_95idy.tsv.gz (2.1Gb uncompressed)
- sra_meta_prot_80idy_3x.tsv.gz (8.3Gb uncompressed)

These files have a line for each RefSeq genome listing all metagenomic SRA runs (as of August 2018) with Mash Containment Scores above the specified threshold. They are provided for two screen modes:

- `nucl`: Genomic RefSeq sequences
- `prot`: Proteomic RefSeq sequences (combined amino acid sequences per organism). **NOTE:** Protein tables above are not p-value filtered and thus large (> ~50Gb) runs may have spurious hits. They also do not contain plasmids. Updates coming soon!

. . . and at two thresholds:

- `95idy`: 95% Mash Containment Score, any coverage. Useful for finding runs containing a specific genome.
- `80idy_3x`: 80% Mash Containment Score, at least 3x median k-mer multiplicity. Useful for finding related, but novel, sequences.

The files are tab separated, with each line beginning with a RefSeq assembly accession, followed by SRA accessions, for example:

```
GCF_000001215.4       SRR3401361      SRR3540373
GCF_000001405.36      SRR5127794      ERR1539652      SRR413753       ERR206081
GCF_000001405.38      SRR5127794      ERR1539652      ERR1711677      SRR413753
↪ERR206081
```

We also provide simple scripts for searching these files: search.tar

1.2.4 Public data sources

The BLAST `nr` database was downloaded from `ftp://ftp.ncbi.nlm.nih.gov/blast/db/nr.*`.

HMP data were downloaded from `ftp://public-ftp.ihmpdcc.org/`, reads from the `Ilumina/` directory and coding sequences from the `HMGI/` directory. Within these folders, sample SRS015937 resides in `tongue_dorsum/` and SRS020263 in `right_retroauricular_crease/`.

SRA runs downloaded with the SRA Toolkit.

RefSeq genomes downloaded from the `genomes/refseq/` directory of `ftp.ncbi.nlm.nih.gov`.

1.2.5 Public data products

Quebec Polyomavirus is submitted to GenBank as BK010702.

CHAPTER 2

Downloads

- Linux/OSX binaries and source
- RefSeq sketch database

CHAPTER 3

Documentation

3.1 Tutorials

3.1.1 Simple distance estimation

Download example *E. coli* genomes:

genome1.fna
genome2.fna

Run:

```
mash dist genome1.fna genome2.fna
```

The results are tab delimited lists of Reference-ID, Query-ID, Mash-distance, P-value, and Matching-hashes:

```
genome1.fna   genome2.fna     0.0222766       0       456/1000
```

3.1.2 Saving time by sketching first

```
mash sketch genome1.fna
mash sketch genome2.fna
mash dist genome1.fna.msh genome2.fna.msh
```

3.1.3 Pairwise comparisons with compound sketch files

Download additional example *E. coli* genome:

genome3.fna

Sketch the first two genomes to create a combined archive, use `mash info` to verify its contents, and estimate pairwise distances:

```
mash sketch -o reference genome1.fna genome2.fna
mash info reference.msh
mash dist reference.msh genome3.fna
```

This will estimate the distance from each query (which there is one of) to each reference (which there are two of in the sketch file):

```
genome1.fna   genome3.fna     0         0         1000/1000
genome2.fna   genome3.fna     0.0222766         0         456/1000
```

3.1.4 Querying read sets against an existing RefSeq sketch

Download the pre-sketched RefSeq archive (reads not provided here; 10x-100x coverage of a single genome with any sequencing technology should work):

refseq.genomes.k21.s1000.msh

Plasmids also available:

refseq.genomes+plasmids.k21.s1000.msh refseq.plasmids.k21.s1000.msh

Concatenate paired ends (this could also be piped to `mash` to save space by specifying - for standard input, zipped or unzipped):

```
cat reads_1.fastq reads_2.fastq > reads.fastq
```

Sketch the reads, using `-m 2` to improve results by ignoring single-copy k-mers, which are more likely to be erroneous:

```
mash sketch -m 2 reads.fastq
```

Run `mash dist` with the RefSeq archive as the reference and the read sketch as the query:

```
mash dist refseq.genomes.k21.s1000.msh reads.fastq.msh > distances.tab
```

Sort the results to see the top hits and their p-values:

```
sort -gk3 distances.tab | head
```

3.1.5 Screening a read set for containment of RefSeq genomes

(new in Mash v2.0)

If a read set potentially has multiple genomes, it can be "screened" against the database to estimate how well each genome is contained in the read set. We can use the SRA Toolkit to download ERR024951:

```
fastq-dump ERR024951
```

. . . and screen it against Refseq Genomes (link above), sorting the results:

```
mash screen refseq.genomes.k21s1000.msh ERR024951.fastq > screen.tab
sort -gr screen.tab | head
```

We see the expected organism, *Salmonella enterica*, but also an apparent contaminant, *Klebsiella pneumoniae*. The fields are [identity, shared-hashes, median-multiplicity, p-value, query-ID, query-comment]:

```
0.99957       991/1000        26      0       GCF_000841985.1_ViralProj14228_genomic.
↪fna.gz   NC_004313.1 Salmonella phage ST64B, complete genome
0.99957       991/1000        24      0       GCF_002054545.1_ASM205454v1_genomic.fna.
↪gz      [57 seqs] NZ_MYON01000010.1 Salmonella enterica strain BCW_4905 NODE_10_
↪length_152932_cov_1.77994, whole genome shotgun sequence [...]
0.999522      990/1000        102     0       GCF_900086185.1_12082_4_85_genomic.fna.
↪gz       [51 seqs] NZ_FLIP01000001.1 Klebsiella pneumoniae strain k1037, whole␣
↪genome shotgun sequence [...]
0.999329      986/1000        24      0       GCF_002055205.1_ASM205520v1_genomic.fna.
↪gz      [72 seqs] NZ_MYOO01000010.1 Salmonella enterica strain BCW_4904 NODE_10_
↪length_177558_cov_3.07217, whole genome shotgun sequence [...]
0.999329      986/1000        24      0       GCF_002054075.1_ASM205407v1_genomic.fna.
↪gz      [88 seqs] NZ_MYNK01000010.1 Salmonella enterica strain BCW_4936 NODE_10_
↪length_177385_cov_3.78874, whole genome shotgun sequence [...]
0.999329      986/1000        24      0       GCF_000474475.1_CFSAN001184_01.0_
↪genomic.fna.gz [45 seqs] NZ_AUQM01000001.1 Salmonella enterica subsp. enterica␣
↪serovar Typhimurium str. CDC_2009K1158 isolate 2009K-1158 SEET1158_1, whole genome␣
↪shotgun sequence [...]
0.999329      986/1000        24      0       GCF_000474355.1_CFSAN001186_01.0_
↪genomic.fna.gz [46 seqs] NZ_AUQN01000001.1 Salmonella enterica subsp. enterica␣
↪serovar Typhimurium str. CDC_2009K1283 isolate 2009K1283 (Typo) SEET1283_1, whole␣
↪genome shotgun sequence [...]
0.999329      986/1000        24      0       GCF_000213635.1_ASM21363v1_genomic.fna.
↪gz       [2 seqs] NC_016863.1 Salmonella enterica subsp. enterica serovar␣
↪Typhimurium str. UK-1, complete genome [...]
0.999281      985/1000        24      0       GCF_001271965.1_Salmonella_enterica_CVM_
↪N43825_v1.0_genomic.fna.gz      [67 seqs] NZ_LIMN01000001.1 Salmonella enterica␣
↪subsp. enterica serovar Typhimurium strain CVM N43825 N43825_contig_1, whole genome␣
↪shotgun sequence [...]
0.999281      985/1000        24      0       GCF_000974215.1_SALF-297-3.id2_v1.0_
↪genomic.fna.gz      [90 seqs] NZ_LAPO01000001.1 Salmonella enterica subsp. enterica␣
↪serovar Typhimurium strain SALF-297-3 NODE_1, whole genome shotgun sequence [...]
```

Note, however, that multiple strains of *Salmonella enterica* have good identity. This is because they are each contained well when considered independently. For this reason `mash screen` is not a true classifier. However, we can remove much of the redundancy for interpreting the results using the winner-take-all strategy (`-w`). And while we're at it, let's throw some more cores at the task to speed it up (`-p 4`):

```
mash screen -w -p 4 refseq.genomes.k21s1000.msh ERR024951.fastq > screen.tab
sort -gr screen.tab | head
```

The output is now much cleaner, with just the two whole genomes, plus phages (a lot of other hits to viruses and assembly contigs would appear further down):

```
0.99957       991/1000        24      0       GCF_002054545.1_ASM205454v1_genomic.fna.
↪gz      [57 seqs] NZ_MYON01000010.1 Salmonella enterica strain BCW_4905 NODE_10_
↪length_152932_cov_1.77994, whole genome shotgun sequence [...]
0.99899       979/1000        26      0       GCF_000841985.1_ViralProj14228_genomic.
↪fna.gz   NC_004313.1 Salmonella phage ST64B, complete genome
0.998844      976/1000        101     0       GCF_900086185.1_12082_4_85_genomic.fna.
↪gz       [51 seqs] NZ_FLIP01000001.1 Klebsiella pneumoniae strain k1037, whole␣
↪genome shotgun sequence [...]
```

(continues on next page)

(continued from previous page)

```
0.923964      190/1000        40      0       GCF_000900935.1_ViralProj181984_genomic.
↪fna.gz  NC_019545.1 Salmonella phage SPN3UB, complete genome
0.900615      111/1000        100     0       GCF_001876675.1_ASM187667v1_genomic.fna.
↪gz      [137 seqs] NZ_MOXK01000132.1 Klebsiella pneumoniae strain AWD5 Contig_(1-
↪18003), whole genome shotgun sequence [...]
0.887722      82/1000 31      3.16322e-233    GCF_001470135.1_ViralProj306294_genomic.
↪fna.gz  NC_028699.1 Salmonella phage SEN34, complete genome
0.873204      58/1000 22      1.8212e-156     GCF_000913735.1_ViralProj227000_genomic.
↪fna.gz  NC_022749.1 Shigella phage SfIV, complete genome
0.868675      52/1000 57      6.26251e-138    GCF_001744215.1_ViralProj344312_genomic.
↪fna.gz  NC_031129.1 Salmonella phage SJ46, complete genome
0.862715      45/1000 1       1.05185e-116    GCF_001882095.1_ViralProj353688_genomic.
↪fna.gz  NC_031940.1 Salmonella phage 118970_sal3, complete genome
0.856856      39/1000 21      6.70643e-99     GCF_000841165.1_ViralProj14230_genomic.
↪fna.gz   NC_004348.1 Enterobacteria phage ST64T, complete genome
```

3.2 Sketches

For sequences to be compared with `mash`, they must first be *sketched*, which creates vastly reduced representations of them. This will happen automatically if `mash dist` is given raw sequences. However, if multiple comparisons will be performed, it is more efficient to create sketches with `mash sketch` first and provide them to `mash dist` in place of the raw sequences. Sketching parameters can be provided to either tool via command line options.

3.2.1 Reduced representations with MinHash tables

Sketches are used by the *MinHash* algorithm to allow fast distance estimations with low storage and memory requirements. To make a sketch, each k-mer in a sequence is *hashed*, which creates a pseudo-random identifier. By sorting these identifiers (*hashes*), a small subset from the top of the sorted list can represent the entire sequence (these are *min-hashes*). The more similar another sequence is, the more min-hashes it is likely to share.

k-mer size

As in any k-mer based method, larger k-mers will provide more specificity, while smaller k-mers will provide more sensitivity. Larger genomes will also require larger k-mers to avoid k-mers that are shared by chance. K-mer size is specified with `-k`, and sketch files must have the same k-mer size to be compared with `mash dist`. When `mash sketch` is run, it automatically assesses the specified k-mer size against the sizes of input genomes by estimating the probability of a random match as:

$$p = \frac{1}{\frac{\left(\Sigma\right)^k}{g} + 1}$$

...where g is the genome size and Σ is the alphabet (ACGT by default). If this probability exceeds a threshold (specified by `-w`; 0.01 by default) for any input genomes, a warning will be given with the minimum k-mer size needed to get within the threshold.

For large collections of sketches, memory and storage may also be a consideration when choosing a k-mer size. Mash will use 32-bit hashes, rather than 64-bit, if they can encompass the full k-mer space for the alphabet in use. This will (roughly) halve the size of the size of the sketch file on disk and the memory it uses when loaded for `mash dist`. The criterion for using a 32-bit hash is:

$$\left(\Sigma\right)^k \leq 2^{32}$$

... which becomes $k \leq 16$ for nucleotides (the default) and $k \leq 7$ for amino acids.

sketch size

Sketch size corresponds to the number of (non-redundant) min-hashes that are kept. Larger sketches will better represent the sequence, but at the cost of larger sketch files and longer comparison times. The error bound of a distance estimation for a given sketch size s is formulated as:

$$\sqrt{\frac{1}{s}}$$

Sketch size is specified with `-s`. Sketches of different sizes can be compared with `mash dist`, although the comparison will be restricted to the smaller of the two sizes.

3.2.2 Strand and alphabet

By default, `mash` uses a nucleotide alphabet (ACGT), is case-insensitive, and will ignore strandedness by using canonical k-mers, as done in Jellyfish. This works by using the reverse complement of a k-mer if it comes before the original k-mer alphabetically. Strandedness can be preserved with `-n` (non-canonical) and case can be preserved with `-Z`. Note that the default nucleotide alphabet does not include lowercase and thus will filter out k-mers with lowercase nucleotides if `-Z` is specified. The amino acid alphabet can be specified with `-a`, which also changes the default k-mer size to reflect the denser information. A completely custom alphabet can also be specified with `-z`. Note that alphabet size affects p-value calculation and hash size (see Assessing significance with p-values and *k-mer size*).

3.2.3 Sketching read sets

When sketching reads instead of complete genomes or assemblies, `-r` should be specified, which will estimate genome size from k-mer content rather than total sequence length, allowing more accurate p-vlaues. Genome size can also be specified directly with `-g`. Additionally, Since MinHash is a k-mer based method, removing unique or low-copy k-mers usually improves results for read sets, since these k-mers are likely to represent sequencing error. The minimum copies of each k-mer required can be specified with `-m` (e.g. `-m 2` to filter unique). However, this could lead to high memory usage if genome size is high and coverage is low, such as in metagenomic read sets. In these cases a Bloom filter can be used (`-b`) to filter out most unique k-mers with constant memory. If coverage is high (e.g. >100x), it can be helpful to limit it to save time and to avoid repeat errors appearing as legitimate k-mers. This can be done with `-c`, which stops sketching reads once the estimated average coverage (based on k-mer multiplicity) reaches the target.

3.2.4 Working with sketch files

The sketch or sketches stored in a sketch file, and their parameters, can be inspected with `mash info`. If sketch files have matching k-mer sizes, their sketches can be combined into a single file with `mash paste`. This allows simple pairwise comparisons with `mash dist`, and allows sketching of multiple files to be parallelized.

3.3 Distance Estimation

3.3.1 MinHash Jaccard estimation

Given k-mer sets A and B, the MinHash algorithm provides an estimation of the Jaccard index:

$$j(A_s, B_s) = \frac{|A_s \cap B_s|}{s}$$

where A_s and B_s are subsets such that $|A_s \cup B_s|$ is equal to the sketch size, s, allowing for a known error bound as suggested by Broder[1]. This is done by using a merge-sort algorithm to find common values between the two sorted sketches and terminating when the total number of hashes seen reaches the sketch size (or all hashes in both sketches have been seen).

3.3.2 Mash distance formulation

For mutating a sequence with t total k-mers and a conserved k-mer count w, an approximate mutation rate d can be estimated using a Poisson model of mutations occurring in k-mers, as suggested by Fan et al.[2]:

$$d = \frac{-1}{k} \ln \frac{w}{t}$$

In order to use a Jaccard estimate j between two k-mer sets of arbitrary sizes, the Jaccard estimate can be framed in terms of the conserved k-mer count w and the average set size n:

$$j \approx \frac{w}{2n - w}$$

To substitute n for the total k-mer count t in the mutation estimation, this approximation can be reformulated as:

$$\frac{w}{n} \approx \frac{2j}{1 + j}$$

Substituting $\frac{w}{n}$ for $\frac{w}{t}$ thus yields the Mash distance:

$$D(k, j) = \begin{cases} 1 & , \ j = 0 \\ \frac{-1}{k} \ln \frac{2j}{1 + j} & , \ 0 < j \leq 1 \end{cases}$$

3.3.3 Assessing significance with p-values

Since MinHash distances are probabilistic estimates, it is important to consider the probability of seeing a given distance by chance. `mash dist` thus provides p-values with distance estimations. Lower p-values correspond to more confident distance estimations, and will often be rounded down to 0 due to floating point limits. If p-values are high (above, say, 0.01), the k-mer size is probably too small for the size of the genomes being compared.

When estimating the distance of genome 1 and genome 2 from sketches with the properties:

Σ := alphabet

k := k-mer size

l_1 := length of genome 1

l_2 := length of genome 2

s := sketch size

x := number of shared k-mers between sketches of size s of genome 1 and genome 2

[1] Broder, A.Z. On the resemblance and containment of documents. Compression and Complexity of Sequences 1997 - Proceedings, 21-29 (1998).

[2] 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).

... the chance of a k-mer appearing in random sequences of lengths l_1 and l_2 are estimated as:

$$r_1 = 1 - (1 - \frac{1}{|\Sigma|^k})^{l_1} \approx \frac{l_1}{l_1 + |\Sigma|^k}$$
$$r_2 = 1 - (1 - \frac{1}{|\Sigma|^k})^{l_2} \approx \frac{l_2}{l_2 + |\Sigma|^k}$$

The expected Jaccard index of the sketches of the random sequences is then:

$$j_r = \frac{r_1 r_2}{r_1 + r_2 - r_1 r_2}$$

... and the probability of observing at least x shared k-mers can be estimated with the tail of a cumulative binomial distribution:

$$p = 1 - \sum_{i=0}^{x-1} \binom{s}{i} j_r^i (1 - j_r)^{s-i}$$