

---

# **mash Documentation**

*Release 1.0*

**Brian Ondov, Todd Treangen, Adam Phillippy**

**May 26, 2017**



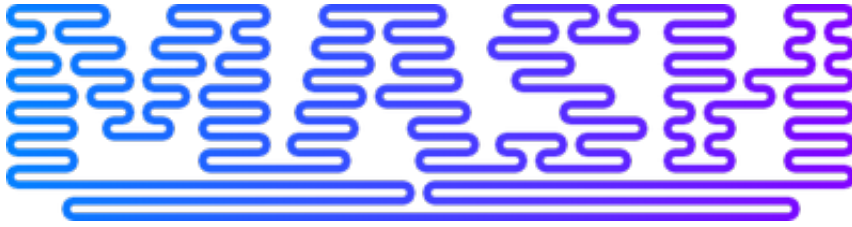
---

## Contents

---

<b>1</b>	<b>Supporting Data</b>	<b>3</b>
<b>2</b>	<b>Downloads</b>	<b>5</b>
<b>3</b>	<b>Documentation</b>	<b>7</b>





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: fast genome and metagenome distance estimation using Min-Hash**

[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)





## CHAPTER 2

---

### Downloads

---

- [Linux/OSX binaries and source](#)



---

## Tutorials

### 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
```

### Saving time by sketching first

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

### 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 genom1.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
```

### Querying read sets against an existing RefSeq sketch

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

[RefSeqSketches.msh.gz](#)

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 read_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 -k 16 -s 400 reads.fastq
```

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

```
mash dist RefSeqSketches.msh reads.fastq.msh > distances.tab
```

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

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

### Building a custom RefSeq database

To create the RefSeq Mash database, genomes were downloaded from NCBI (<ftp.ncbi.nlm.nih.gov/refseq/release/complete>, fasta sequence and GenBank annotations for genomic), and the [refseqCollate](#) utility was used to collate contigs/chromosomes into individual fasta files per genome. Groups of these files were sketched in parallel and then pasted together with `mash paste`. This process could be repeated for more current or custom databases.

### 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.

## 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:

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

...where  $g$  is the genome size and  $\Sigma$  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:

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

...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.

## 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*).

## 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-values. 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.

## 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.

## Distance Estimation

### 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<sup>1</sup>. 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).

### 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.<sup>2</sup>:

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

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  $n$ :

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

---

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

<sup>2</sup> 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).

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:

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

### 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:

$\Sigma$  := 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

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

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

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

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:

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