Algorithms in Bioinformatics

MinHash Signatures

Recap

- HW due today ←
- Final coding project:

  ideally by Friday of finals week

  welcome to work with a partner

Measuring string similarity:

We've seen a bunch of methods so far:

- Hamming distance
- Edit distance
- Global alignment

Another idea:

Convert the string into a set of items, & see how similar the sets are.

For documents:
reduce to words or phrases

In biology:
k-mers

Why? Faster, rougher notion of similarity

So: convert "words" to IDs.
$\nwarrow$ "k-shingles"

If the documents are similar, then expect lots of identical IDs.

Note: No semantic meanings are attached!

So - a major weakness:

Two sets could be similar even if documents are not.

However, still remarkably useful:

- News aggregators
- Plagiarism detection
- More recently, fast similarity tests for biological data

Core problem: Set Similarity

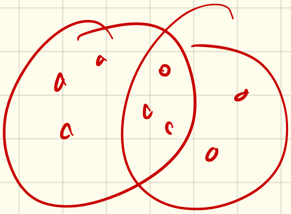
Given two sets, how similar are they?

Example: Netflix watch lists

- We've both seen 100 movies
- 50 are identical.

Jaccard similarity:

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



3/8

In our example:

$$\frac{50}{150} = \frac{1}{3}$$

You can do this in the obvious way:

For each pair, calculate Jaccard similarity

Return any above a threshold

How many? $\binom{n}{2} = O(n^2)$

<u>Problem</u>: Large datasets.

- Suppose similarity calculation takes only 1ms per pair.
- If ~1 million documents
  $\binom{n}{2} \approx 500$ billion

→ time $\approx$ 16 years

So intsead: MinHash signatures

- Fixed length (independent of set size)
- We will compare these signatures in order to get an approximation of Jaccard Similarity.

↳ in *expected* value

Algorithm:

- Pick a (nearly) collision-free hash function

if $a \neq b$, $h(a) \neq h(b)$

Why? Permutation, uniformly at random

Example: $h(x) = (ax+b) \% c$

a, b: both < max value of x (relatively prime)

c: prime # just larger than max x

Then:

Generate a family of c of these hash functions.

If they are "good", each will essentially permute $0 \ldots (2^{32}-1)$, c different ways

Then: the signature is computed by computing

- the minimum hash value produced by h_1
- min by h_2

  ⋮
- min by h_c

↳ c values per data set

Use the same c hash functions for every document in the data set.

$$\text{Similarity} = \frac{\#\text{ same components in signature}}{\text{total }\#\text{ components in signature}}$$

Simple example:

$A = \{32, \underline{3}, 22, 6, \underline{15}, \underline{11}\}$

$B = \{\underline{15}, 30, 7, \underline{11}, 28, \underline{3}, 17\}$

Jaccard similarity: $\frac{3}{10}$

Now: Minhash calculation (ideally) is just taking union of the two sets + randomly permuting it.

$A \cup B = \{32, \underline{3}, 22, 6, \underline{15}, \underline{11}, 30, 7, 28, 17\}$

Q: What is probability that something from $A \cap B$ is first in the list after permuting randomly?

$3/10$

Now, back to full signature:

Say we do 20 hash functions to get the signature.

How many minhash values should they have in common?

$$\#\text{ components of signature} \times \text{probability of a match}$$

$$E\left[\#\text{ in common}\right] = 20 \cdot 3/10 = 6$$

☆ (assuming no collisions)

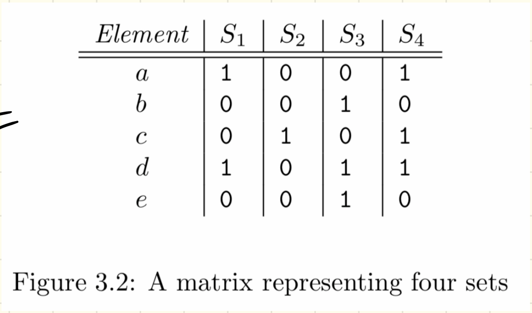
So expected value of minhash similarity

= ~~Minhash~~ Jaccard similarity

$$\frac{6}{20} = \frac{3}{10} \quad ✓$$

Another example view:

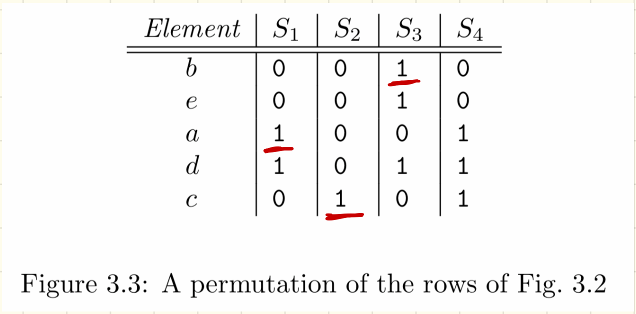
$M =$



Element	S_1	S_2	S_3	S_4
a	1	0	0	1
b	0	0	1	0
c	0	1	0	1
d	1	0	1	1
e	0	0	1	0

Figure 3.2: A matrix representing four sets

One "hash":



Element	S_1	S_2	S_3	S_4
b	0	0	$\underline{1}$	0
e	0	0	1	0
a	$\underline{1}$	0	0	1
d	1	0	1	1
c	0	$\underline{1}$	0	1

Figure 3.3: A permutation of the rows of Fig. 3.2

$$\left.\begin{aligned} h_1(S_1) &= a \\ h_1(S_2) &= c \\ h_1(S_3) &= b \\ h_1(S_4) &= a \end{aligned}\right\}$$

hash computation

(really, can't do permutation)

To get a signature:

- Pick n permutations:

  $h_1 \dots h_n$

- For each set (or column), generate

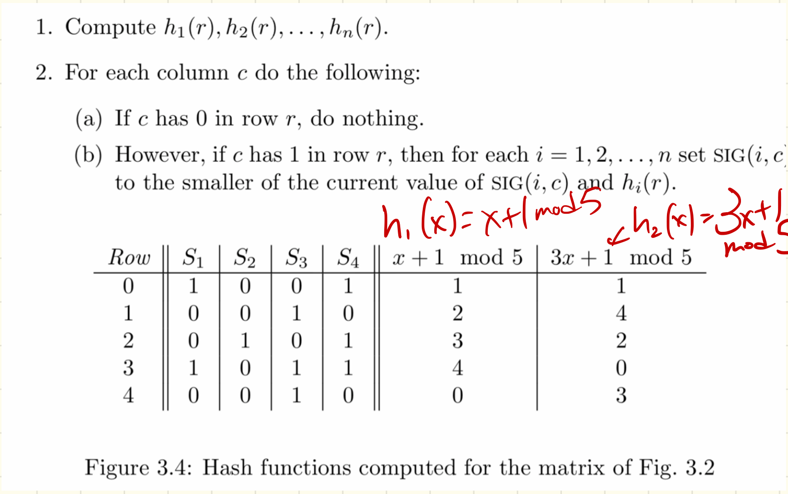
  $h_1(S), h_2(S), \dots, h_n(S)$

Get Signature matrix:

an $\underbrace{n}_{\text{\# hash functions}} \times |S|$ matrix

(usually much smaller than original $\underbrace{M}_{\text{\# elements} \times |S|}$)

These large matrices are impractical, which is why we use 'hash functions' instead.



1. Compute $h_1(r), h_2(r), \ldots, h_n(r)$.
2. For each column c do the following:
   (a) If c has 0 in row r, do nothing.
   (b) However, if c has 1 in row r, then for each $i = 1, 2, \ldots, n$ set $\text{SIG}(i, c$ to the smaller of the current value of $\text{SIG}(i, c)$ and $h_i(r)$.

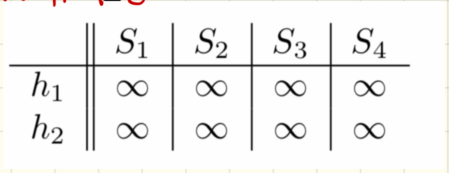
$h_1(x) = x+1 \bmod 5$ $\swarrow h_2(x) = 3x+1 \bmod 5$

Row	S_1	S_2	S_3	S_4	$x+1 \bmod 5$	$3x+1 \bmod 5$
0	1	0	0	1	1	1
1	0	0	1	0	2	4
2	0	1	0	1	3	2
3	1	0	1	1	4	0
4	0	0	1	0	0	3

Figure 3.4: Hash functions computed for the matrix of Fig. 3.2

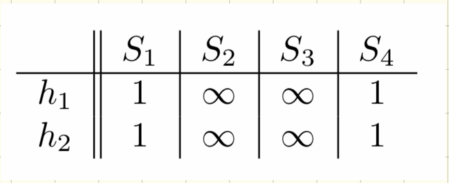
How it goes:

initialize $= \infty$ for all



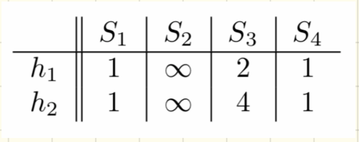
	S_1	S_2	S_3	S_4
h_1	∞	∞	∞	∞
h_2	∞	∞	∞	∞

Now, consider row 0.
$h_1(0)$ & $h_2(0)$ are both 1, so:



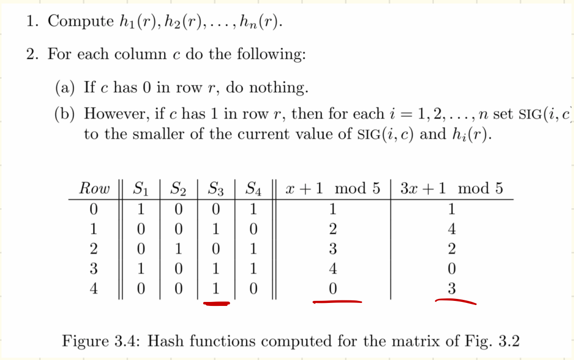
	S_1	S_2	S_3	S_4
h_1	1	∞	∞	1
h_2	1	∞	∞	1

Now row 1: has S_2, & $h_1(1) = 2$, $h_2(1) = 4$:



	S_1	S_2	S_3	S_4
h_1	1	∞	2	1
h_2	1	∞	4	1

Continuing:

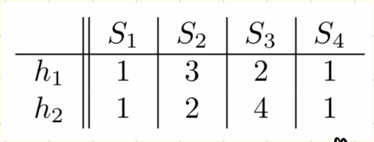


1. Compute $h_1(r), h_2(r), \ldots, h_n(r)$.
2. For each column c do the following:
   (a) If c has 0 in row r, do nothing.
   (b) However, if c has 1 in row r, then for each $i = 1, 2, \ldots, n$ set $\text{SIG}(i, c$ to the smaller of the current value of $\text{SIG}(i, c)$ and $h_i(r)$.

Row	S_1	S_2	S_3	S_4	$x + 1 \mod 5$	$3x + 1 \mod 5$
0	1	0	0	1	1	1
1	0	0	1	0	2	4
2	0	1	0	1	3	2
3	1	0	1	1	4	0
4	0	0	$\underline{1}$	0	$\underline{0}$	$\underline{3}$

Figure 3.4: Hash functions computed for the matrix of Fig. 3.2

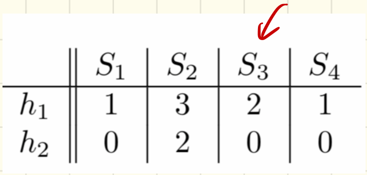
Row 2: S_2 & S_4, & $h_1(2) = 3$
$h_2(2) = 2$



	S_1	S_2	S_3	S_4
h_1	1	3	2	1
h_2	1	2	4	1

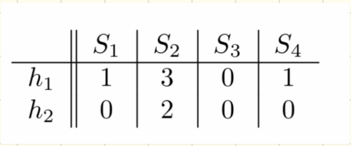
Note: same

Row 3: has S_1, S_3, & S_4
and $h_1(3) = 4$, $h_2(3) = 0$



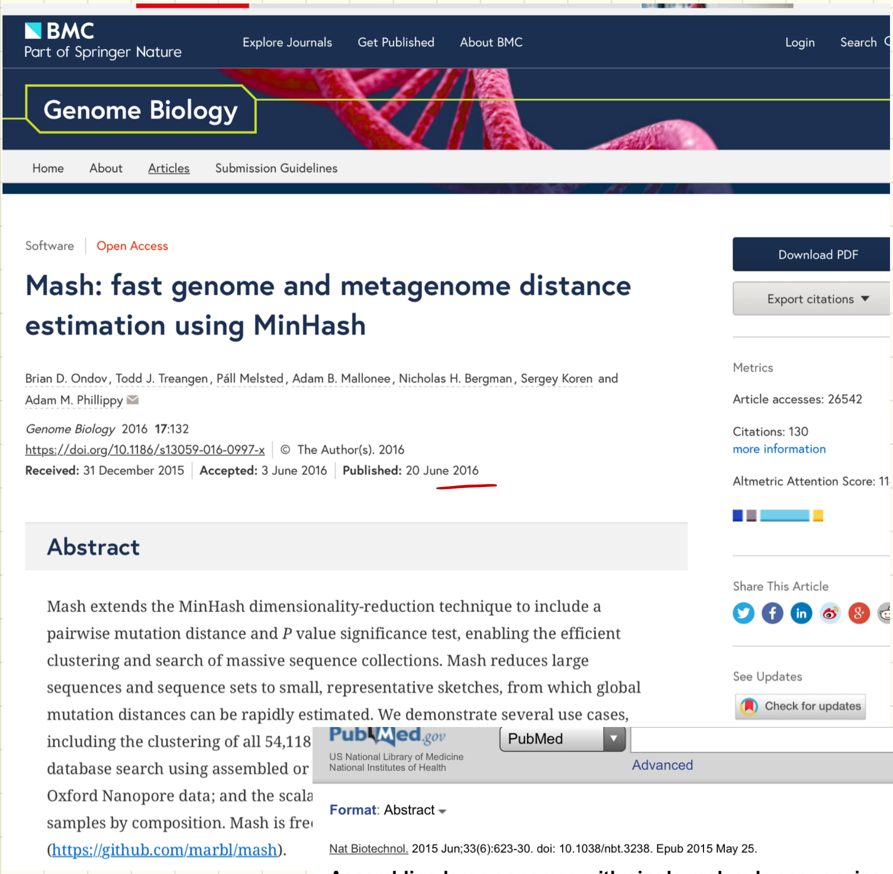
	S_1	S_2	S_3	S_4
h_1	1	3	2	1
h_2	0	2	0	0

Finally:



	S_1	S_2	S_3	S_4
h_1	1	3	0	1
h_2	0	2	0	0

similarity of S_1 + S_4 is 100%
S_1 + S_3: 50%



Software | Open Access

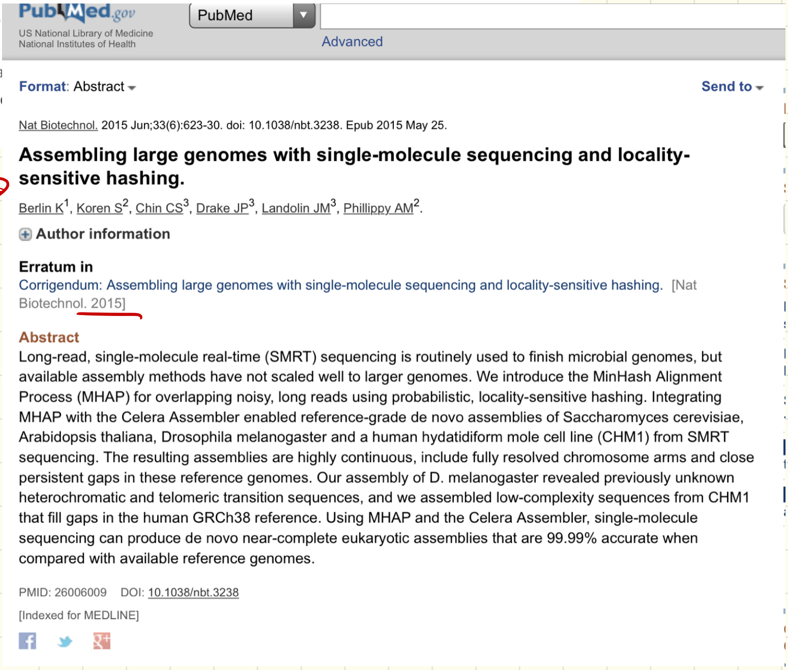
Mash: fast genome and metagenome distance estimation using MinHash

Brian D. Ondov, Todd J. Treangen, Páll Melsted, Adam B. Mallonee, Nicholas H. Bergman, Sergey Koren and Adam M. Phillippy

Genome Biology 2016 **17**:132
https://doi.org/10.1186/s13059-016-0997-x | © The Author(s). 2016
Received: 31 December 2015 | **Accepted:** 3 June 2016 | **Published:** 20 June 2016

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
database search using assembled or
Oxford Nanopore data; and the scala
samples by composition. Mash is fre
(https://github.com/marbl/mash).


PubMed.gov
US National Library of Medicine
National Institutes of Health
PubMed
Advanced
Format: Abstract
Send to

Nat Biotechnol. 2015 Jun;33(6):623-30. doi: 10.1038/nbt.3238. Epub 2015 May 25.

Assembling large genomes with single-molecule sequencing and locality-sensitive hashing.

Berlin K[1], Koren S[2], Chin CS[3], Drake JP[3], Landolin JM[3], Phillippy AM[2].

Author information

Erratum in
Corrigendum: Assembling large genomes with single-molecule sequencing and locality-sensitive hashing. [Nat Biotechnol. 2015]

Abstract

Long-read, single-molecule real-time (SMRT) sequencing is routinely used to finish microbial genomes, but available assembly methods have not scaled well to larger genomes. We introduce the MinHash Alignment Process (MHAP) for overlapping noisy, long reads using probabilistic, locality-sensitive hashing. Integrating MHAP with the Celera Assembler enabled reference-grade de novo assemblies of Saccharomyces cerevisiae, Arabidopsis thaliana, Drosophila melanogaster and a human hydatidiform mole cell line (CHM1) from SMRT sequencing. The resulting assemblies are highly continuous, include fully resolved chromosome arms and close persistent gaps in these reference genomes. Our assembly of D. melanogaster revealed previously unknown heterochromatic and telomeric transition sequences, and we assembled low-complexity sequences from CHM1 that fill gaps in the human GRCh38 reference. Using MHAP and the Celera Assembler, single-molecule sequencing can produce de novo near-complete eukaryotic assemblies that are 99.99% accurate when compared with available reference genomes.

PMID: 26006009 DOI: 10.1038/nbt.3238
[Indexed for MEDLINE]