Melbourne Bioinformatics
BIOINFORMATICS + DATA SERVICES + INFRASTRUCTURE, FOR LIFE SCIENCES TODAY

POSTERA CRESCAM LAUDE
THE UNIVERSITY OF
MELBOURNE

# COMP90014

## Algorithms for Bioinformatics

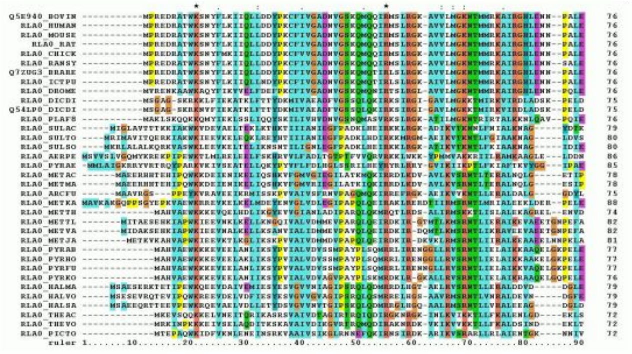
## Week 4A - Comparing Sequences

# Comparing Sequences

MinHash

Minimizers

Multiple Sequence Alignment



# Comparing Sequences

How we’ve previously been using kmers

- Kmers as distance metric (seqA vs seqB)
- Kmer indexes (alignment heuristic using seeds)

# Comparing Sequences

## How we've previously been using kmers

Kmers as distance metric (seqA vs seqB)

Kmer indexes (alignment heuristic using seeds)

## How we will improve both today

MinHash: Blazingly fast fingerprinting & comparison

Minimizers: Small memory footprint indexes

# Comparing Sequences

## How we’ve previously been using kmers

Kmers as distance metric (seqA vs seqB)

Kmer indexes (alignment heuristic using seeds)

## How we will improve both today

MinHash: Blazingly fast fingerprinting & comparison

Minimizers: Small memory footprint indexes

## Aligning Multiple Sequences

No kmers sadly

# MinHash

# MinHash

Fingerprinting

Sets & Jaccard Coefficient

Sketches and Estimation

Applications

Limitations

# MinHash

这是一种用于快速估计数据集相似度的算法。MinHash特别适用于大数据集，因为它能有效地压缩数据并比较它们的“指纹”来确定相似性。这里的“指纹”是指数据集的压缩表示。

## Fingerprinting

数据压缩：MinHash技术通过创建一个较小的数据表示形式，即“指纹”，来压缩数据。这种指纹足以表示原始数据的重要特性。
比较指纹而非完整数据：使用MinHash创建的指纹，可以比较两个数据集的指纹而不是它们的完整数据，这样可以大幅减少计算量和时间

Compresses data into smaller form.

Can compare ***fingerprints*** rather than full data.

Used as heuristic - are two things similar?

Eg identifying person via literal fingerprints - forensics

MinHash通常用作一种启发式方法来快速估计两个集合的相似度，尤其是在需要比较集合中元素的情况下。
它不是完美准确的，但是可以给出是否有足够相似性的良好指示

# MinHash

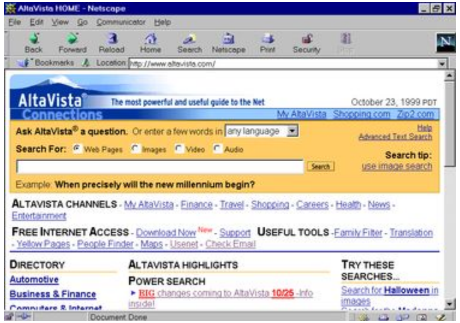
## Fingerprinting

Compresses data into smaller form.

Can compare ***fingerprints*** rather than full data.

Used as heuristic - are two things similar?

Eg identifying person via literal fingerprints - forensics



AltaVista HOME - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security
Bookmarks Location http://www.altavista.com/
Search
Document Done

AltaVista® Connections The most powerful and useful guide to the Net October 23, 1999 PDT

My AltaVista Shopping.com Zip2.com

Ask AltaVista® a question. Or enter a few words in any language

Help
Advanced Text Search

Search For: ● Web Pages ○ Images ○ Video ○ Audio

Search tip: use image search

Example: When precisely will the new millennium begin?

ALTAVISTA CHANNELS - My AltaVista - Finance - Travel - Shopping - Careers - Health - News - Entertainment

FREE INTERNET ACCESS - Download Now New - Support USEFUL TOOLS - Family Filter - Translation - Yellow Pages - People Finder - Maps - Usenet - Check Email

DIRECTORY
Automotive
Business & Finance
Computers & Internet

ALTAVISTA HIGHLIGHTS
POWER SEARCH
BIG changes coming to AltaVista 10/25 - Info inside!

TRY THESE SEARCHES...
Search for Halloween in images

## Found everywhere in big data

Webpages - are two pages similar?
(MinHash - from the 1997 AltaVista search engine)

Shazam - are two audio tracks similar?

Bioinformatics - are two sequences similar?

# MinHash

## Fingerprinting

In bioinformatics, fingerprinting usually involves ***kmers.***

Fingerprinting is generally only seen in one-to-many or many-to-many tasks. Why?

# MinHash

## Fingerprinting

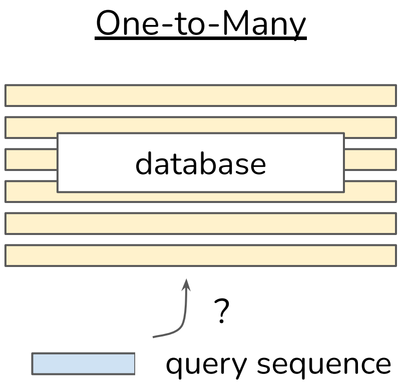
In bioinformatics, fingerprinting usually involves ***kmers.***

Fingerprinting is generally only seen in one-to-many or many-to-many tasks. Why?

## One-to-Many

For a given sequence, search massive database to find similar sequence(s)

-> BLAST, Kraken2, MetaPhlAn4



# MinHash

## Fingerprinting

In bioinformatics, fingerprinting usually involves ***kmers.***

Fingerprinting is generally only seen in one-to-many or many-to-many tasks. Why?

## One-to-Many

For a given sequence, search massive database to find similar sequence(s)

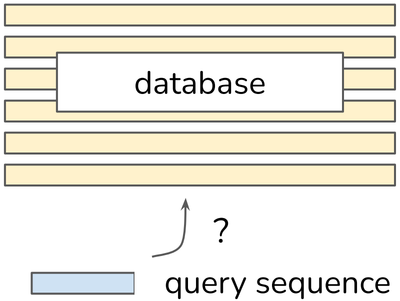
-> BLAST, Kraken2, MetaPhlAn4

## Many-to-Many

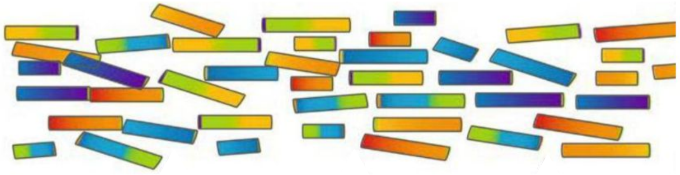
For millions of reads, which reads have overlapping sections?

-> Canu assembler

One-to-Many



Many-to-Many



Which fragments overlap?

# MinHash

## Sets & Jaccard Coefficient

In MinHash, we use sets to compare how similar two sequences are.

The items in the sets are kmers
(extracted from the sequences)

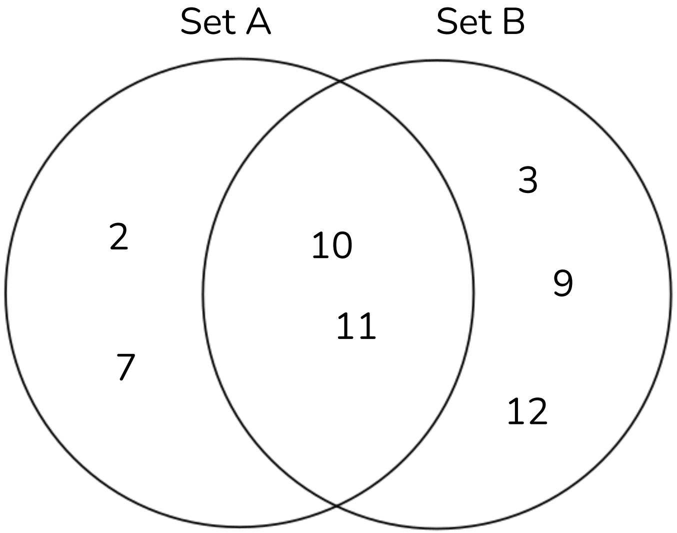
We use Jaccard Coefficient as our measure of similarity.

```
Shared, Unique SeqA, Unique SeqB

SeqA: AGTCGTAGC

3-mers: {AGT, GTC, TCG, CGT, GTA, TAG, AGC}

SeqB: AGTCGGTAG

3-mers: {AGT, GTC, TCG, CGG, GGT, GTA, TAG}
```

# MinHash

## Sets & Jaccard Coefficient

```
Set A:  {2, 7, 10, 11}
Set B:  {3, 9, 10, 11, 12}
```

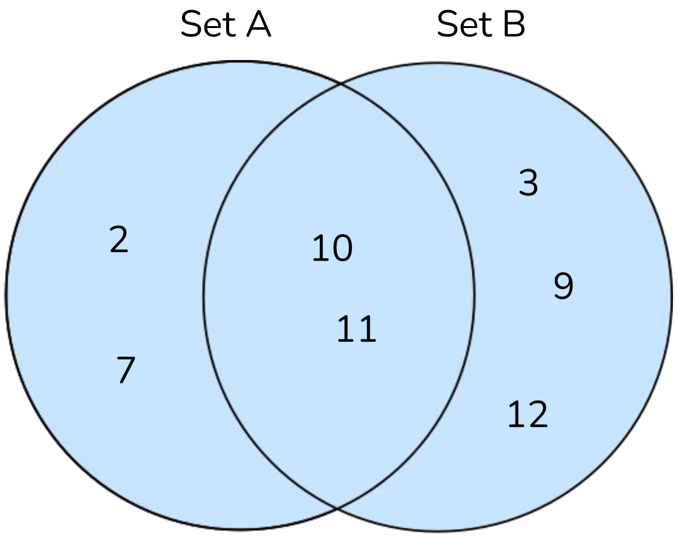


# MinHash

Sets & Jaccard Coefficient

```
Set A:  {2, 7, 10, 11}
Set B:  {3, 9, 10, 11, 12}
```

**Union** *(A ∪ B)*
Items in A or B
`{2, 3, 7, 9, 10, 11, 12}`



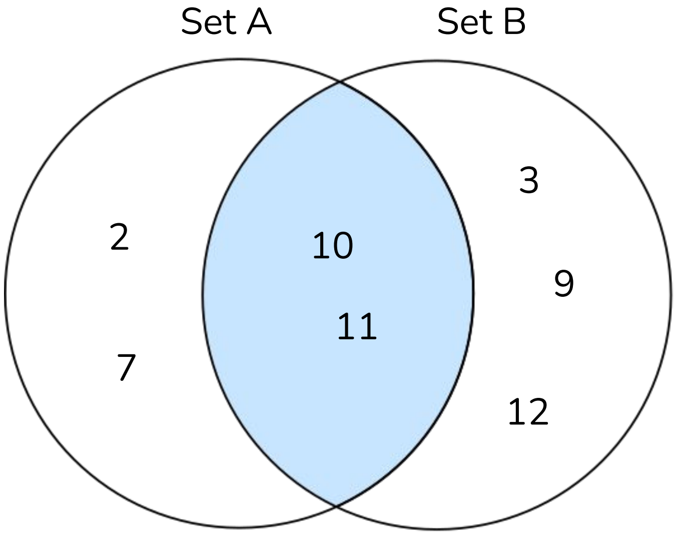
# MinHash

Sets & Jaccard Coefficient

```
Set A:  {2, 7, 10, 11}
Set B:  {3, 9, 10, 11, 12}
```

**Union** *(A ∪ B)*
Items in A or B
`{2, 3, 7, 9, 10, 11, 12}`

**Intersection** *(A ∩ B)*
Items in A and B
`{10, 11}`



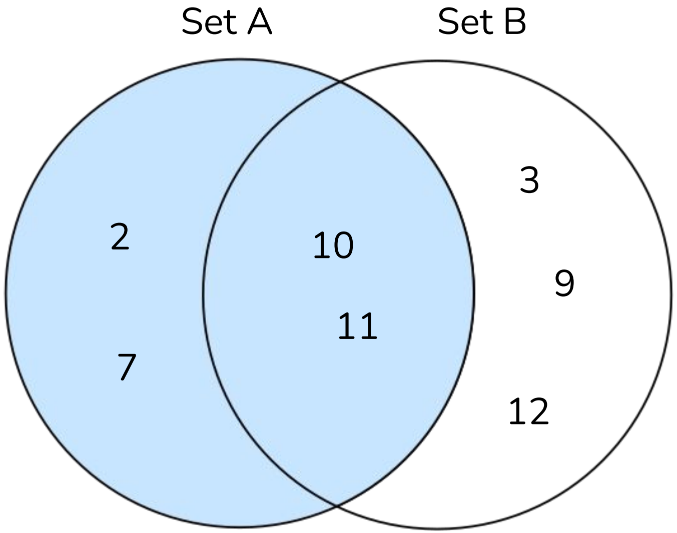
# MinHash

Sets & Jaccard Coefficient

```
Set A:  {2, 7, 10, 11}
Set B:  {3, 9, 10, 11, 12}
```

**Union** *(A ∪ B)*
Items in A or B
`{2, 3, 7, 9, 10, 11, 12}`

**Intersection** *(A ∩ B)*
Items in A and B
`{10, 11}`



**Exclusion** *(A \ B)*
Items in A and not B
`{2, 7, 10, 11}`

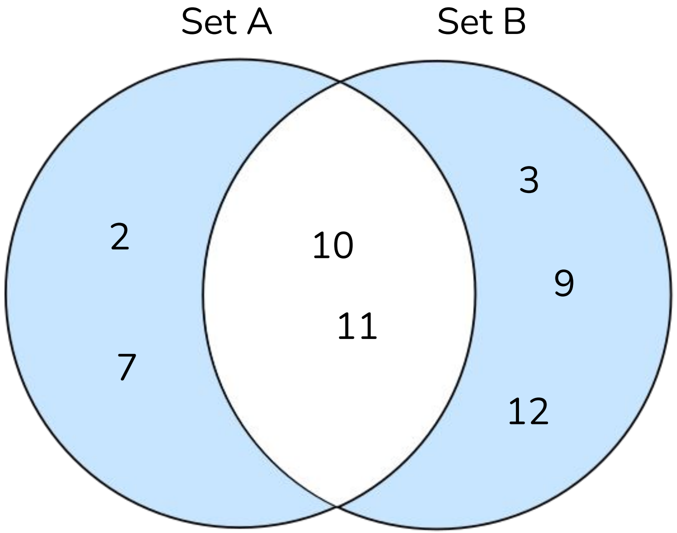
# MinHash

## Sets & Jaccard Coefficient

```
Set A:  {2, 7, 10, 11}
Set B:  {3, 9, 10, 11, 12}
```

**Union** *(A ∪ B)*
Items in A or B
`{2, 3, 7, 9, 10, 11, 12}`

**Intersection** *(A ∩ B)*
Items in A and B
`{10, 11}`



**Exclusion** *(A \ B)*
Items in A and not B
`{2, 7, 10, 11}`

**Symmetric Difference** *(A Δ B)*
Items unique to A or B
`{2, 3, 7, 9, 12}`

# MinHash

Sets & Jaccard Coefficient

```
Set A:  {2, 7, 10, 11}
Set B:  {3, 9, 10, 11, 12}
```

**Jaccard Similarity Coefficient**

(Jaccard Index)

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

# MinHash

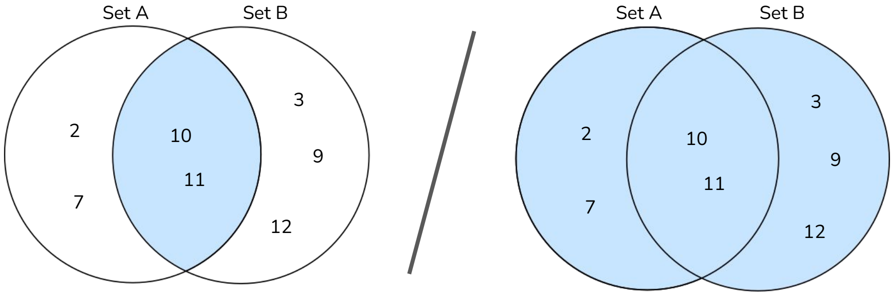
## Sets & Jaccard Coefficient

```
Set A:  {2, 7, 10, 11}
Set B:  {3, 9, 10, 11, 12}
```

**Jaccard Similarity Coefficient**

(Jaccard Index)

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



Jaccard: What proportion of items are shared?

# MinHash

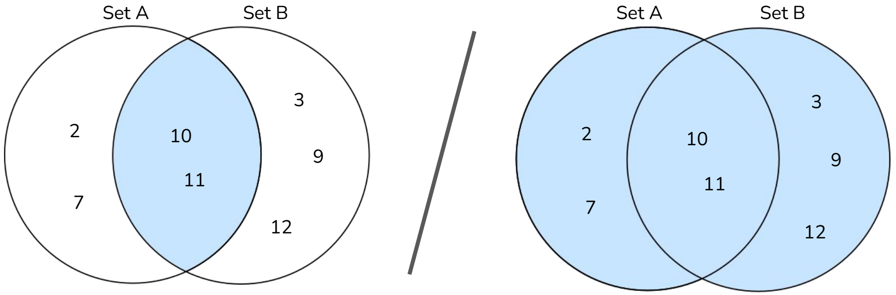
## Sets & Jaccard Coefficient

```
Set A:  {2, 7, 10, 11}
Set B:  {3, 9, 10, 11, 12}
```

**Jaccard Similarity Coefficient**

(Jaccard Index)

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



Jaccard: What proportion of items are shared?

```
J(A, B) = 2 / 7 = 0.29
```

# MinHash

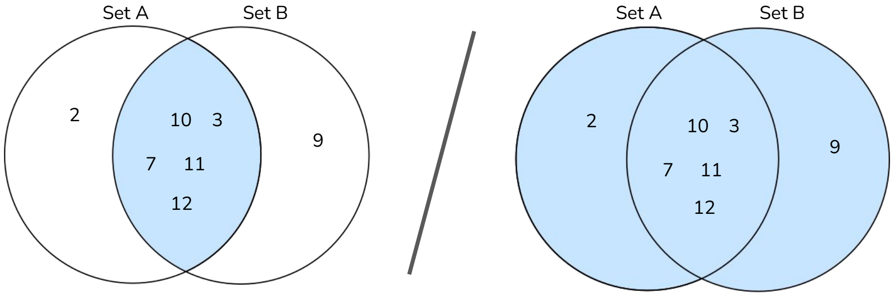
Sets & Jaccard Coefficient

```
Set A:  {2, 7, 10, 11}
Set B:  {3, 9, 10, 11, 12}
```

## Jaccard Similarity Coefficient

(Jaccard Index)

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



Jaccard: What proportion of items are shared?

```
J(A, B) = 5 / 7 = 0.71
```

# MinHash

## Sets & Jaccard Coefficient

For MinHash, we use sets to compare how similar two sequences are.

The items in the sets are kmers
(extracted from the sequences)

We use Jaccard Coefficient as our measure of similarity.

```
Shared, Unique SeqA, Unique SeqB

SeqA: AGTCGTAGC

3-mers: {AGT, GTC, TCG, CGT, GTA, TAG, AGC}

SeqB: AGTCGGTAG

3-mers: {AGT, GTC, TCG, CGG, GGT, GTA, TAG}
```

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = 5 \; / \; 9 \quad = 0.56$$

# MinHash

## Sets & Jaccard Coefficient

For MinHash, we use sets to compare how similar two sequences are.

The items in the sets are kmers (extracted from the sequences)

We use Jaccard Coefficient as our measure of similarity.

… Cool, but we've done this before??

```
Shared, Unique SeqA, Unique SeqB

SeqA: AGTCGTAGC

3-mers: {AGT, GTC, TCG, CGT, GTA, TAG, AGC}

SeqB: AGTCGGTAG

3-mers: {AGT, GTC, TCG, CGG, GGT, GTA, TAG}
```

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = 5 / 9 = 0.56$$

# MinHash

## Sketches and Estimation

If we were comparing two sequences, we could just use Jaccard.

The issue is when we have ***many*** sequences.

Imagine:

- N=10,000 kmers on avg. per seq.
- M=100,000 sequences
- *O(N x M)* = 1 billion operations

# MinHash

## Sketches and Estimation

If we were comparing two sequences, we could just use Jaccard.

The issue is when we have ***many*** sequences.

Imagine:

- N=10,000 kmers on avg. per seq.
- M=100,000 sequences
- *O(N x M)* = 1 billion operations

## The plan

1. Take a random sample of kmers as fingerprint from each seq
2. Compare fingerprints rather than full data?
3. Use this as a heuristic to pre-screen potentially similar sequences

From indexing weeks, we know that hashing a kmer produces a random integer

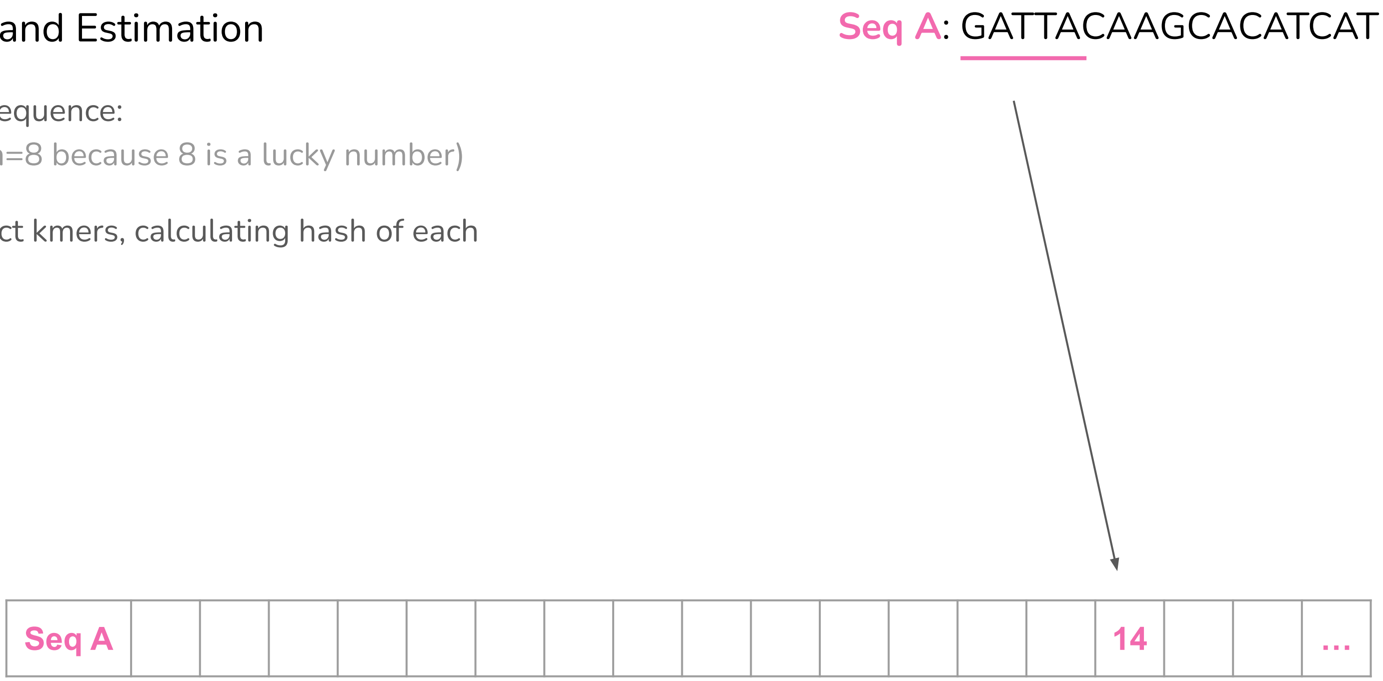
Can use a hash function to randomly select kmers

# MinHash

## Sketches and Estimation

For each sequence:
(Using min=8 because 8 is a lucky number)

1. Extract kmers, calculating hash of each



Seq A: GATTACAAGCACATCAT

| | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Seq A | | | | | | | | | | | | | | | 14 | | | … |

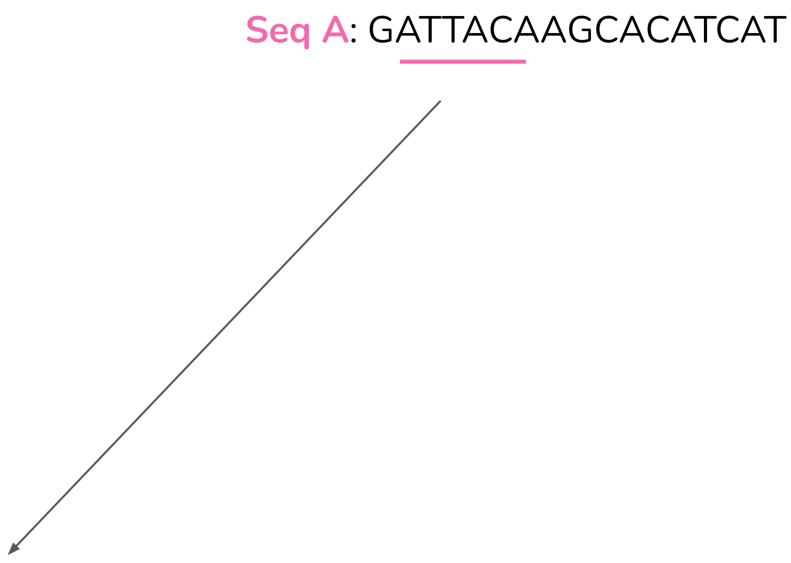
# MinHash

## Sketches and Estimation

For each sequence:

(Using min=8 because 8 is a lucky number)

1. Extract kmers, calculating hash of each



Seq A: GATTACAAGCACATCAT

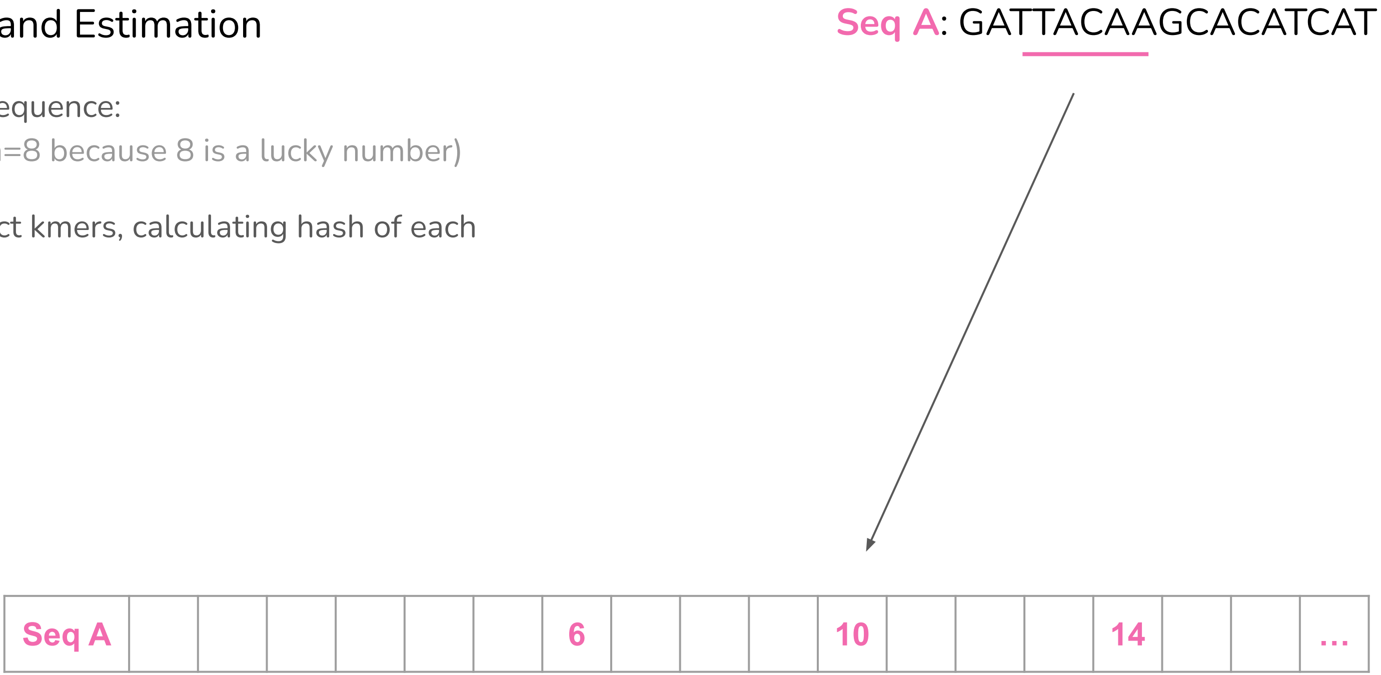
| | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Seq A | | | | | | | 6 | | | | | | | | 14 | | | ... |

# MinHash

## Sketches and Estimation

For each sequence:
(Using min=8 because 8 is a lucky number)

1. Extract kmers, calculating hash of each



Seq A: GATTACAAGCACATCAT

| | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Seq A | | | | | | | 6 | | | | 10 | | | | 14 | | | … |

# MinHash

## Sketches and Estimation

For each sequence:
(Using min=8 because 8 is a lucky number)

1. Extract kmers, calculating hash of each

**Seq A**: GATTACAAGCACATCAT

| | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Seq A | | | | | | | 6 | | | | 10 | | | | 14 | | | … |

# MinHash

## Sketches and Estimation

For each sequence:
(Using min=8 because 8 is a lucky number)

1. Extract kmers, calculating hash of each

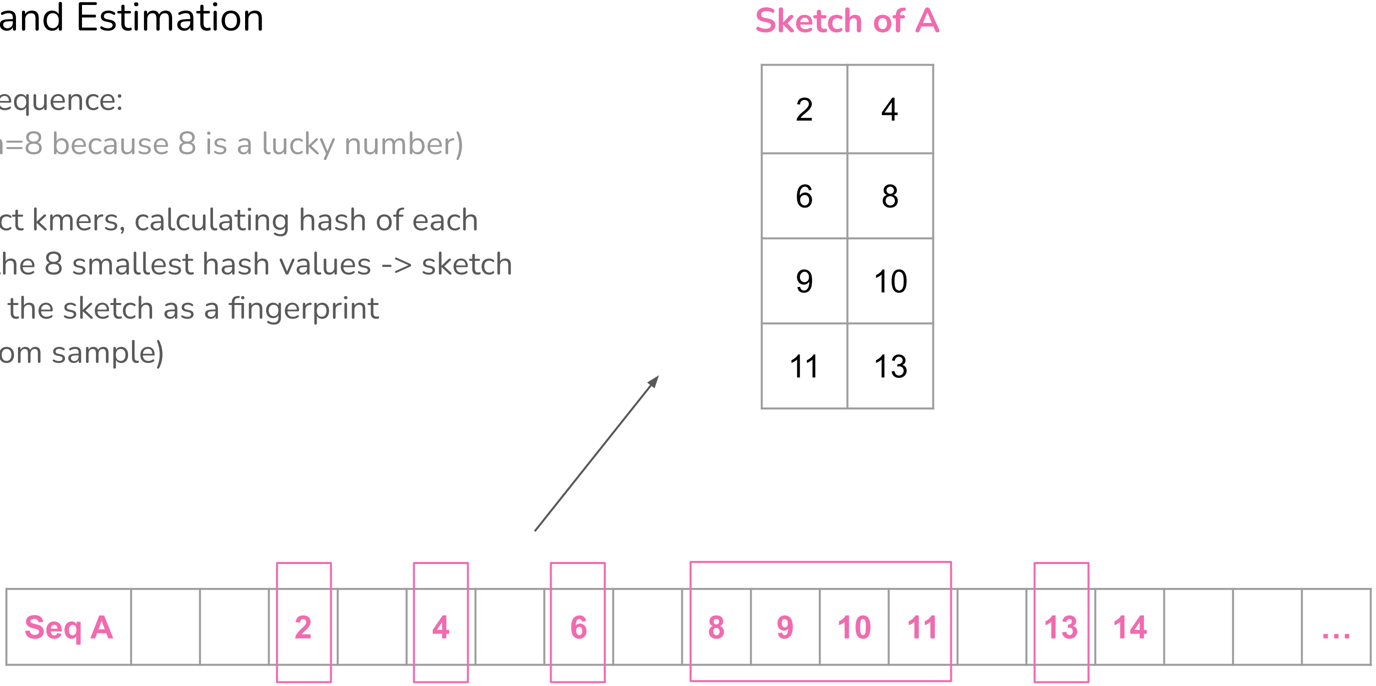
| | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Seq A | | | 2 | | 4 | | 6 | | 8 | 9 | 10 | 11 | | 13 | 14 | | | … |

# MinHash

## Sketches and Estimation

For each sequence:
(Using min=8 because 8 is a lucky number)

1. Extract kmers, calculating hash of each
2. Pick the 8 smallest hash values -> sketch
3. Store the sketch as a fingerprint (random sample)



Sketch of A

| | |
|---|---|
| 2 | 4 |
| 6 | 8 |
| 9 | 10 |
| 11 | 13 |

| | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Seq A | | | 2 | | 4 | | 6 | | 8 | 9 | 10 | 11 | | 13 | 14 | | | … |

# MinHash

## Sketches and Estimation

For each sequence:
(Using min=8 because 8 is a lucky number)

1. Extract kmers, calculating hash of each
2. Pick the 8 smallest hash values -> sketch
3. Store the sketch as a fingerprint
   (random sample)

Sketch of A

| | |
|---|---|
| 2 | 4 |
| 6 | 8 |
| 9 | 10 |
| 11 | 13 |

| | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Seq A | | | 2 | | 4 | | 6 | | 8 | 9 | 10 | 11 | | 13 | 14 | | | … |

# MinHash

## Sketches and Estimation

For each sequence:

(Using min=8 because 8 is a lucky number)

1. Extract kmers, calculating hash of each
2. Pick the 8 smallest hash values -> sketch
3. Store the sketch as a fingerprint
   (random sample)

Sketch of A

| | |
|---|---|
| 2 | 4 |
| 6 | 8 |
| 9 | 10 |
| 11 | 13 |

Sketch of B

| | |
|---|---|
| 2 | 5 |
| 6 | 9 |
| 10 | 11 |
| 13 | 15 |

| | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Seq A** | | | **2** | | **4** | | **6** | | **8** | **9** | **10** | **11** | | **13** | **14** | | | **…** |
| **Seq B** | | | **2** | | | **5** | **6** | | | **9** | **10** | **11** | | **13** | | **15** | **16** | **…** |

# MinHash

## Sketches and Estimation

For each sequence:
(Using min=8 because 8 is a lucky number)

1. Extract kmers, calculating hash of each
2. Pick the 8 smallest hash values -> sketch
3. Store the sketch as a fingerprint
   (random sample)

Can then compare sketches, rather than full data.
Jaccard index of sketches ≈ Jaccard index of full data.

### Sketch of A

| | |
|---|---|
| 2 | 4 |
| 6 | 8 |
| 9 | 10 |
| 11 | 13 |

### Sketch of B

| | |
|---|---|
| 2 | 5 |
| 6 | 9 |
| 10 | 11 |
| 13 | 15 |

| | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Seq A** | | | 2 | | 4 | | 6 | | 8 | 9 | 10 | 11 | | 13 | 14 | | | … |
| **Seq B** | | | 2 | | | 5 | 6 | | | 9 | 10 | 11 | | 13 | | 15 | 16 | … |

# MinHash

## Sketches and Estimation

For each sequence:
(Using min=8 because 8 is a lucky number)

1. Extract kmers, calculating hash of each
2. Pick the 8 smallest hash values -> sketch
3. Store the sketch as a fingerprint (random sample)

Can then compare sketches, rather than full data.
Jaccard index of sketches ≈ Jaccard index of full data.

Sketch of A

| | |
|---|---|
| 2 | 4 |
| 6 | 8 |
| 9 | 10 |
| 11 | 13 |

U

Sketch of B

| | |
|---|---|
| 2 | 5 |
| 6 | 9 |
| 10 | 11 |
| 13 | 15 |

=

Union Sketch
*(A U B)*

| | |
|---|---|
| 2 | 4 |
| 5 | 6 |
| 8 | 9 |
| 10 | 11 |

| | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Seq A** | | | **2** | | **4** | | **6** | | **8** | **9** | **10** | **11** | | **13** | **14** | | | **…** |
| **Seq B** | | | **2** | | | **5** | **6** | | | **9** | **10** | **11** | | **13** | | **15** | **16** | **…** |

# MinHash

## Sketches and Estimation

For each sequence:
(Using min=8 because 8 is a lucky number)

1. Extract kmers, calculating hash of each
2. Pick the 8 smallest hash values -> sketch
3. Store the sketch as a fingerprint
   (random sample)

Can then compare sketches, rather than full data.
Jaccard index of sketches ≈ Jaccard index of full data.

Sketch of A

| | |
|---|---|
| 2 | 4 |
| 6 | 8 |
| 9 | 10 |
| 11 | 13 |

∪

Sketch of B

| | |
|---|---|
| 2 | 5 |
| 6 | 9 |
| 10 | 11 |
| 13 | 15 |

=

Union Sketch
(A U B)

| | |
|---|---|
| 2 | 4 |
| 5 | 6 |
| 8 | 9 |
| 10 | 11 |

$J(A, B) \approx 5 / 8 = 0.625$

| | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Seq A** | | | 2 | | 4 | | 6 | | 8 | 9 | 10 | 11 | | 13 | 14 | | | … |
| **Seq B** | | | 2 | | | 5 | 6 | | | 9 | 10 | 11 | | 13 | | 15 | 16 | … |

# MinHash

## Limitations?

如果我们想知道一个特定部分是否具有高的MinHash Jaccard指数，MinHash本身并没有提供这种定位信息。

No position information

(In most cases not needed!)

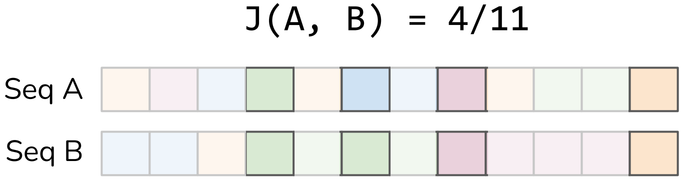
Finding read overlaps: would be nice to know if a section has high MinHash Jaccard Index.

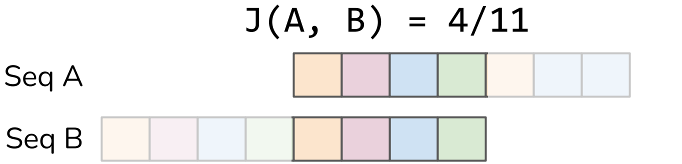
Redundant when comparing 2 sequences

Exact matching methods weak for distant sequences

- Can you think of a way to remedy?
- BLAST word neighborhoods

说明尽管MinHash提供了一个快速的相似性估计，但它并没有显示这些相似部分在序列中的位置。





当比较两个序列时，MinHash可能会遇到冗余，因为它可能在两个非常不同的序列上生成相似的指纹，尤其是当这些序列具有重复区域时。

对于差异较大的序列，完全依赖精确匹配的方法（例如，完全一致的字符串匹配）可能不会很有效，因为它们无法捕捉到相似性，只能确定是否有完全的匹配。

# MinHash

## Summary and Applications

MinHash is a way to fingerprint a sequence

- Fingerprint called a ‘sketch’
- Essentially a random sample
- Used as a heuristic to quickly compare seqs.
- No position information*

Jaccard Coefficient

- Measures ratio of shared kmers / all kmers
  (eg: how similar are two sequences?)
- Can estimate Jaccard using MinHash sketches
- Benefit: sketches more space / time efficient

## Seen in:

- Genome / metagenome distance (Mash)
- Genome Assembly (Canu)
- Sequence matching to database (sourmash)

*minimizier (explored next) hash sketches can be used instead of MinHash. Include position info, but may have issues with bias.

# Minimizers

# Minimizers

Problem with stride

Using windows and minimizers

Limitations

Applications

# Minimizers

如人类基因组（大约3Gb，即3亿个碱基对）时，我们不希望提取序列中的每一个kmer

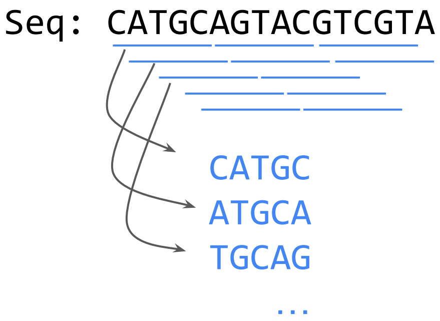
## Problem With Stride

For long sequences, don't want to extract every kmer.

Extracting every possible kmer:

- Storage size is roughly ***k*** x length(seq)
- Human genome ≈ 3 Gb, kmer size = 15
- Roughly 45 Gb index for 3 Gb sequence

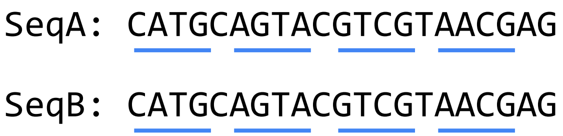
Ok, let's extract every 10th kmer!



# Minimizers

## Problem With Stride

Extracting every kmer using a stride ***w*** works...



```
SeqA:  CATGCAGTACGTCGTAACGAG

SeqB:  CATGCAGTACGTCGTAACGAG
```

# Minimizers

## Problem With Stride

Extracting every kmer using a stride ***w*** works...

```
SeqA:  CATGCAGTACGTCGTAACGAG

SeqB:  CATGCAGTACGTCGTAACGAG
```

k=4, w=5

| Pos | SeqA | SeqB |
|---|---|---|
| 0 | CATG | CATG |
| 5 | AGTA | AGTA |
| 10 | GTCG | GTCG |
| 15 | AACG | AACG |

# Minimizers

## Problem With Stride

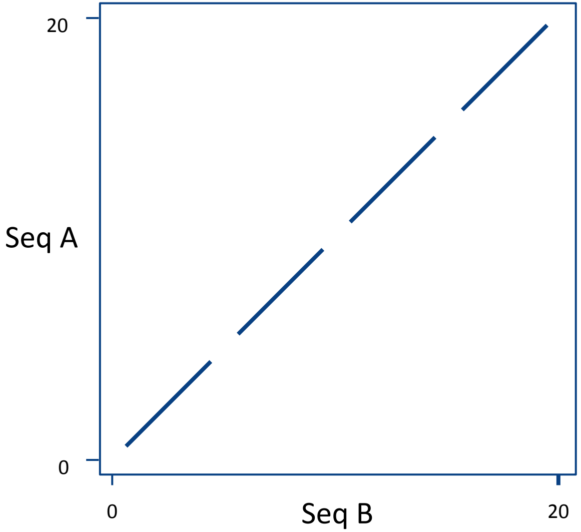
Extracting every kmer using a stride ***w*** works...

SeqA: CATGCAGTACGTCGTAACGAG

SeqB: CATGCAGTACGTCGTAACGAG

k=4, w=5

| Pos | SeqA | SeqB |
|---|---|---|
| 0 | CATG | CATG |
| 5 | AGTA | AGTA |
| 10 | GTCG | GTCG |
| 15 | AACG | AACG |



# Minimizers

## Problem With Stride

存在indel的话就会导致后续的匹配不上

Extracting every kmer using a stride *w* works...

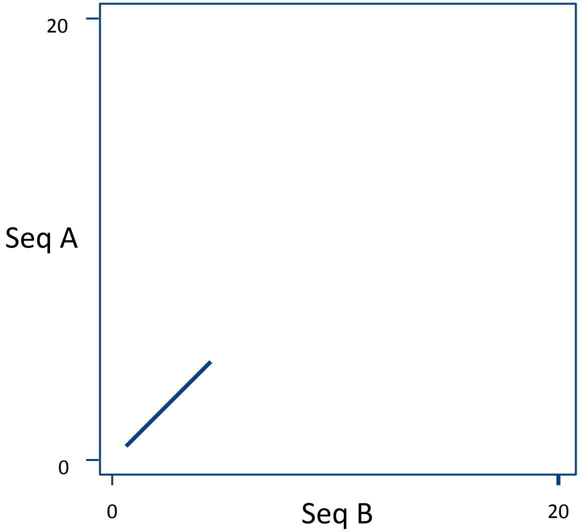
Until there are indels.

SeqA: CATGCAGTACGTCGTAACGAG

SeqB: CATGCAAGTACGTCGTAACGAG

k=4, w=5

| Pos | SeqA | SeqB |
|---|---|---|
| 0 | CATG | CATG |
| 5 | AGTA | AAGT |
| 10 | GTCG | CGTC |
| 15 | AACG | TAAC |



# Minimizers

## Using windows and minimizers

Rather than using a stride ***w***, use a window ***w***

Within the window, have an order (function) which picks a single kmer in that window

In this manner, we have:

window ***w***; kmer size ***k***; order ***o***

Window:

Minimizer: Select the minimum kmer from a window according to an order

| Window | Selected kmer (minimizer) |
|---|---|
| GATCGTACAGTCAGTC | **ACAGT** |
| ATCGTACAGTCAGTCA | ACAGT |
| TCGTACAGTCAGTCAA | ACAGT |
| CGTACAGTCAGTCAAT | ACAGT |
| GTACAGTCAGTCAATG | ACAGT |
| TACAGTCAGTCAATGC | ACAGT |
| ACAGTCAGTCAATGCA | ACAGT |
| CAGTCAGTCAATGCAT | **AATGC** |
| AGTCAGTCAATGCATT | AATGC |

在每个窗口内，应用order函数选择一个kmer作为该窗口的代表，称为minimizer。

选择的minimizer是该窗口内“最小”的kmer，其中“最小”是根据order函数定义的。这意味着在所有可能的kmers中，这个kmer在排序顺序中排在最前面。

通过在每个窗口中选取minimizer，可以减少总体的kmer数量，因为一些kmers会被重复选为多个窗口的minimizer。

# Minimizers

## Using windows and minimizers

Rather than using a stride ***w***, use a window ***w***

Within the window, have an order (function) which picks a single kmer in that window

In this manner, we have:

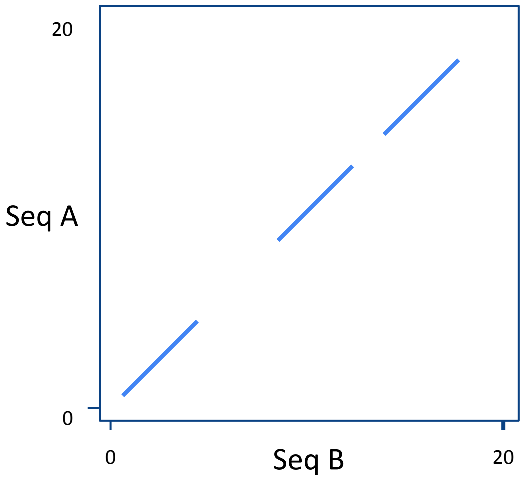
window ***w***; kmer size ***k***; order ***o***

Window:

Minimizer: Select the minimum kmer from a window according to an order

SeqA: CATGCAG TCAACGT TAACGAG

SeqB: CATGCAG TCAACGT TAACGAG



# Minimizers

## Using windows and minimizers

Rather than using a stride ***w***, use a window ***w***

Within the window, have an order (function) which picks a single kmer in that window

In this manner, we have:

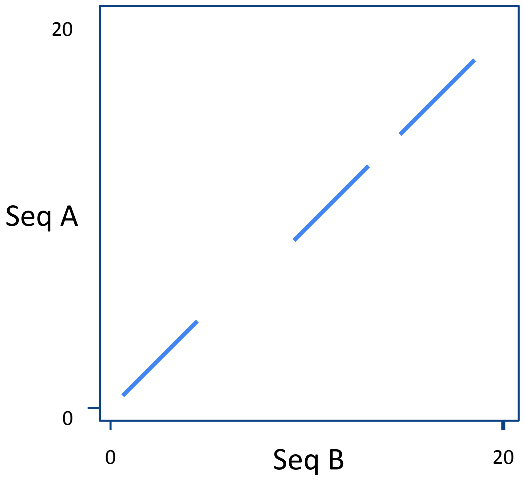
window ***w***; kmer size ***k***; order ***o***

Window:

Minimizer: Select the minimum kmer from a window according to an order

SeqA: CATGCAGTCAACGTTAACGAG

SeqB: CATGCAGTACAACGTTAACGAG



[NOT ASSESSED]

# Shazam

1.
Each fingerprint hash is calculated using audio samples near a corresponding point in time (distant events do not affect the hash)

2.
Fingerprint hashes derived from corresponding matching content are reproducible independent of position within an audio file

3.
Hashes generated from the original clean database track should be reproducible from a degraded copy of the audio

4.
Fingerprint tokens should have sufficiently high entropy in order to minimize the probability of false token matches at non-corresponding locations between the unknown sample and tracks within the database

[NOT ASSESSED]

# Shazam

## 1

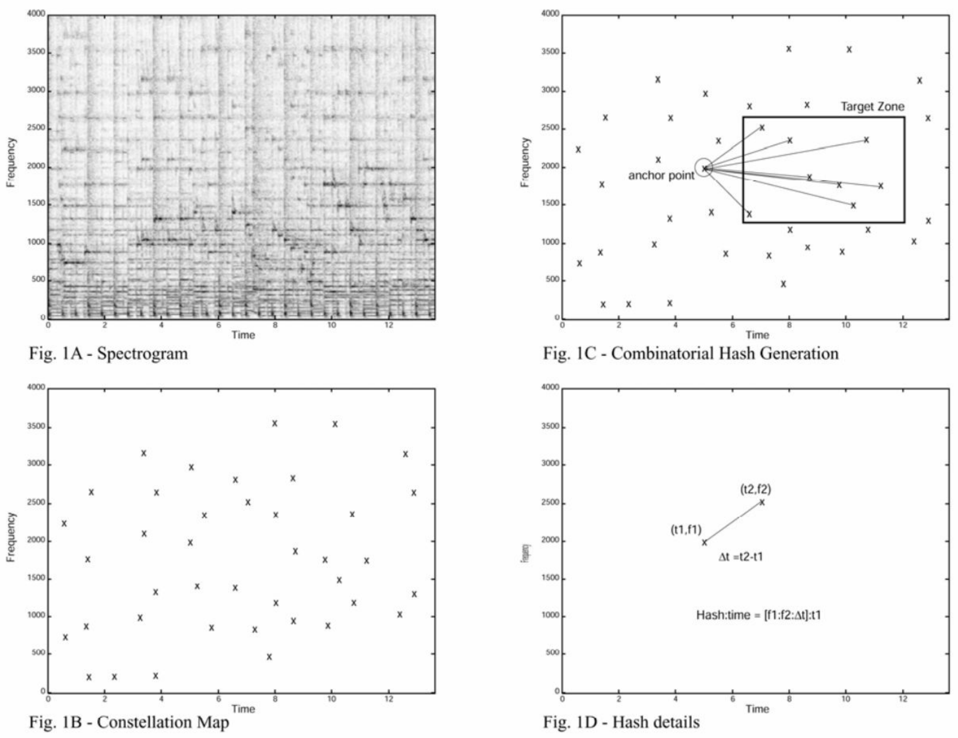


Fig. 1A - Spectrogram

Fig. 1C - Combinatorial Hash Generation

Fig. 1B - Constellation Map

Fig. 1D - Hash details

## 2

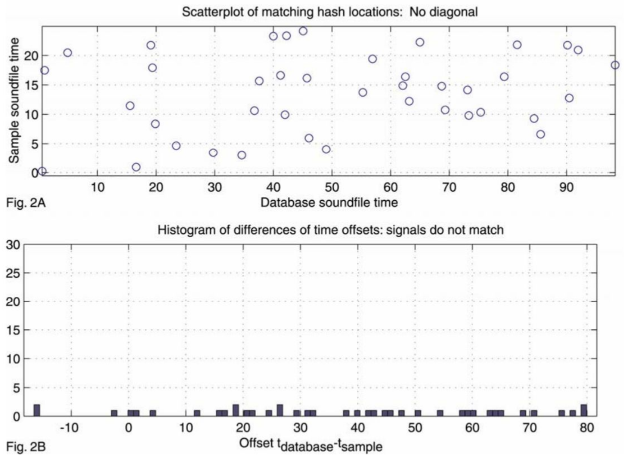


Fig. 2A

Fig. 2B

## 3

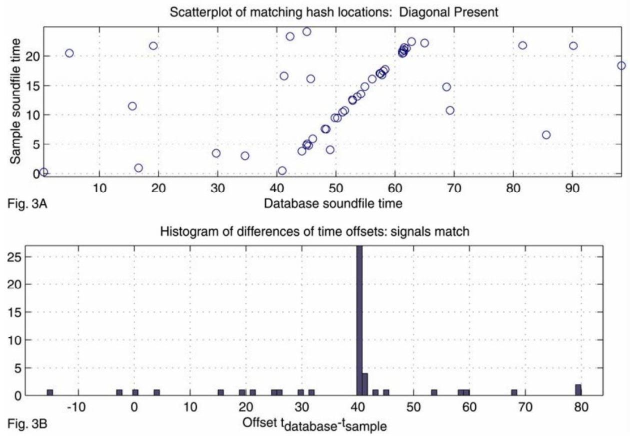


Fig. 3A

Fig. 3B

# Minimizers

## Limitations

***W*** and ***K*** are tradeoff between efficiency & accuracy

For ***W:***

W = 1: All kmers sampled, most accurate

W = 10: 1 in 10 kmers sampled, less accurate, more efficient

For ***K:***

Lower K: more minimizers will match, but the minimizers are less informative

Higher K: less minimizers will match, but the minimizers are more informative

# Minimizers

## Summary and Applications

Smaller representation

Preserves position information

## Seen in:

Characterisation tools (Kraken2)

Genome-genome aligners

Long read aligners (Minimap2)

Other tools (like shazam!)

# Multiple Sequence Alignment

# Multiple Sequence Alignment

Evolutionary Conservation

Aims of MSA

Definition

Scoring MSAs

Computing MSAs

# Multiple Sequence Alignment

## Evolutionary Conservation

DNA -> AA -> Fold

## DNA -> AA:  Codon wobble

Due to wobble, 64 codons -> 20 proteins.

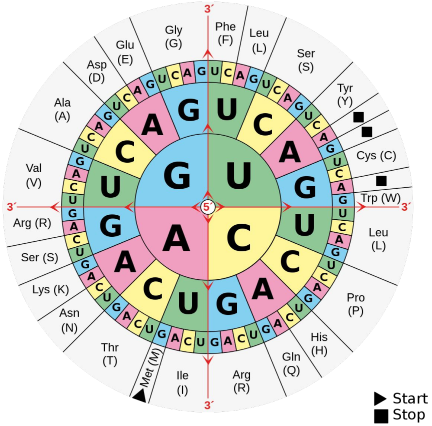
Some DNA can change (last base in codon) while AA remains the same.

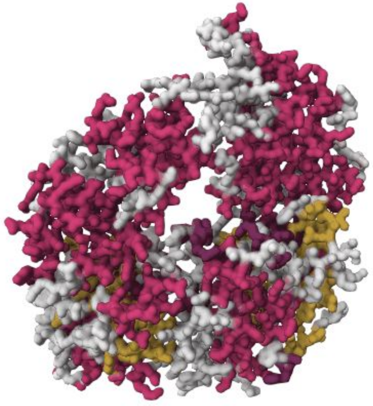
## AA -> Fold:   Structural preservation

Due to properties, some AA can change while protein fold remains roughly the same.

Proteins - fold + key sites dictate abilities / functionality.

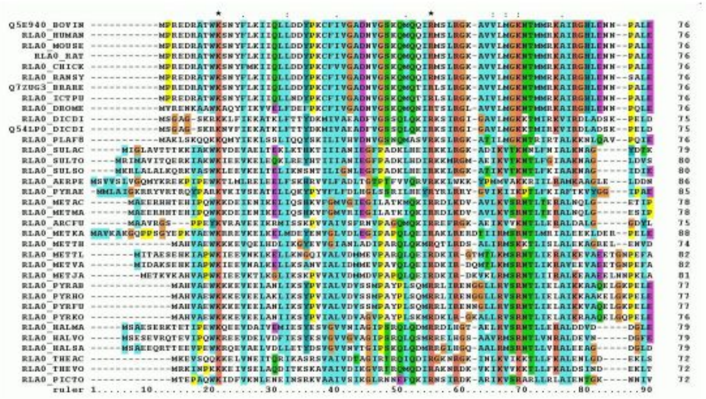
Preserving fold is important, not necessarily exact amino acids

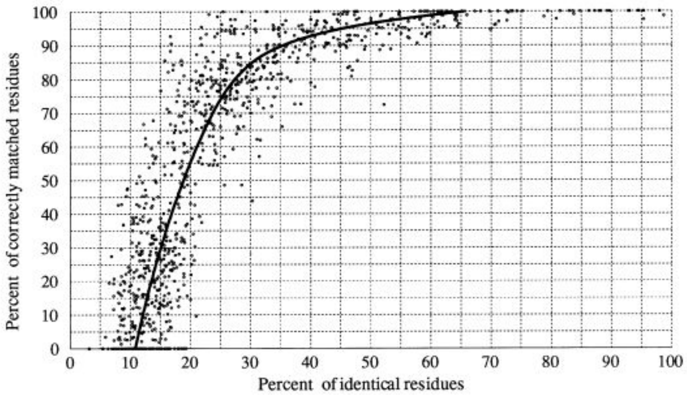




Molecular surface of helicase

# Goal: identify similarities between sequences





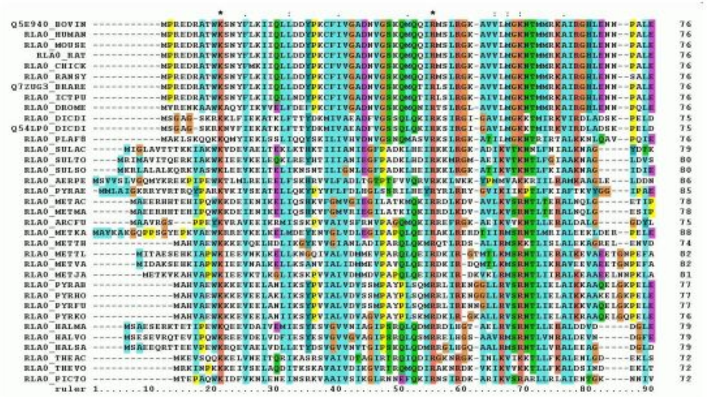
- pairwise alignments identify similarities for related sequences
- might not work for distant sequences

Multiple sequence alignment (MSA):

- Find conserved patterns (motifs) in a protein family

Miguel Andrade via Wikimedia Commons; Vogt *et al.*, 1995. 10.1006/jmbi.1995.0340

# Applications



Miguel Andrade via Wikimedia Commons

Trees: evolutionary relationships between genes and proteins

Alignment: conserved or functional domains

Structure: predict protein function (homology modelling)

# Definition

```
S1 = ACG--GAGA
S2 = -CGTTGACA
S3 = AC-T-GA-A
S4 = CCGTTCAC-
           *
```

* homologous position

**Formally:**

- given $k$ sequences $S = \{S_1, S_2, ..., S_k\}$
- a multiple alignment of $S$ is a set of $k$ equal-length sequences:
  $\{S'_1, S'_2, ..., S'_k\}$
  where $S'_i$ is obtained by inserting gaps into $S_i$

- The multiple sequence alignment problem aims to find a multiple alignment which optimizes a certain score

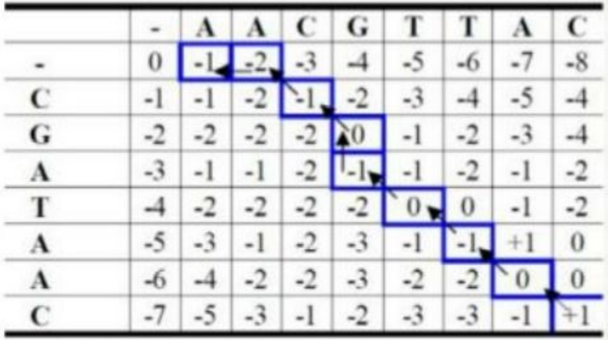
# Evaluating multiple sequence alignments

```
S1  =  ACG--GAGA
S2  =  -CGTTGACA
S3  =  AC-T-GA-A
S4  =  CCGTTCAC-
```

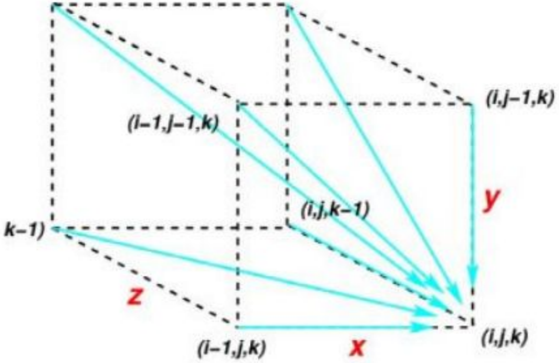
### Sum-of-pairs (SP-score)

- extension of the scoring used in pairwise alignments
- For a given column $(a_1, \ldots, a_k)$,
  SP-score $= \sum_{1 \leq i < j \leq k} \delta(a_i, a_j)$
- Assumes statistical independence between columns

# Optimal alignment



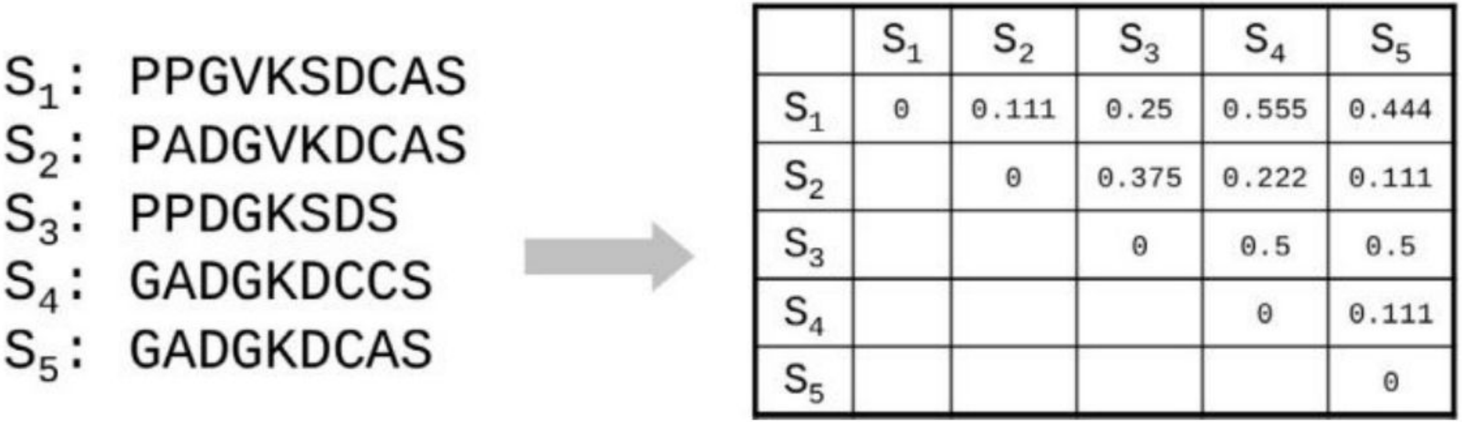
| | - | A | A | C | G | T | T | A | C |
|---|---|---|---|---|---|---|---|---|---|
| - | 0 | -1 | -2 | -3 | -4 | -5 | -6 | -7 | -8 |
| C | -1 | -1 | -2 | -1 | -2 | -3 | -4 | -5 | -4 |
| G | -2 | -2 | -2 | -2 | 0 | -1 | -2 | -3 | -4 |
| A | -3 | -1 | -1 | -2 | -1 | -1 | -2 | -1 | -2 |
| T | -4 | -2 | -2 | -2 | -2 | 0 | 0 | -1 | -2 |
| A | -5 | -3 | -1 | -2 | -3 | -1 | -1 | +1 | 0 |
| A | -6 | -4 | -2 | -2 | -3 | -2 | -2 | 0 | 0 |
| C | -7 | -5 | -3 | -1 | -2 | -3 | -3 | -1 | +1 |



- dynamic programming
- two sequences:
  - two-dimensional matrix
  - $O(n^2)$
- three sequences: $O(n^3)$
- $k$ sequences?
- $O(n^k)$: intractable

# Heuristic: progressive alignment

Step 1: Calculate scores for all pairwise alignments (distance matrix)
Based on the aligned portion (excluding gaps)



| | |
|---|---|
| $S_1$: | PPGVKSDCAS |
| $S_2$: | PADGVKDCAS |
| $S_3$: | PPDGKSDS |
| $S_4$: | GADGKDCCS |
| $S_5$: | GADGKDCAS |

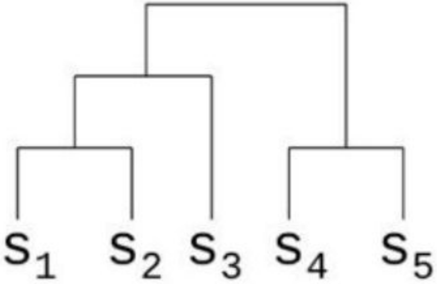
| | $S_1$ | $S_2$ | $S_3$ | $S_4$ | $S_5$ |
|---|---|---|---|---|---|
| $S_1$ | 0 | 0.111 | 0.25 | 0.555 | 0.444 |
| $S_2$ | | 0 | 0.375 | 0.222 | 0.111 |
| $S_3$ | | | 0 | 0.5 | 0.5 |
| $S_4$ | | | | 0 | 0.111 |
| $S_5$ | | | | | 0 |

*ClustalW*: Thompson *et al.*, 1994. 10.1093/nar/22.22.4673.

# Heuristic: progressive alignment

**Step 2:** Build a guide tree of similar sequences based on the pairwise alignments
Agglomerative clustering (neighbour join - greedy heuristic approach).
Joins at each step, the two closest sub-trees.

|  | $S_1$ | $S_2$ | $S_3$ | $S_4$ | $S_5$ |
|---|---|---|---|---|---|
| $S_1$ | 0 | 0.111 | 0.25 | 0.555 | 0.444 |
| $S_2$ |  | 0 | 0.375 | 0.222 | 0.111 |
| $S_3$ |  |  | 0 | 0.5 | 0.5 |
| $S_4$ |  |  |  | 0 | 0.111 |
| $S_5$ |  |  |  |  | 0 |



序列之间的距离越小，说明它们之间的相似性越高。例如，S1和S2之间的距离为0.111

*ClustalW*: Thompson *et al.*, 1994. 10.1093/nar/22.22.4673.

# Heuristic: progressive alignment

Step 3: Progressive alignment following the guide tree

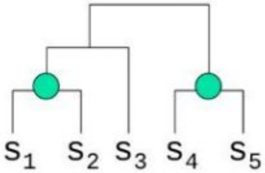
Align the two most closely-related sequences first.
This alignment is then 'fixed' and will never change.

If a gap is to be introduced subsequently it will be introduced in the same place in both sequences, but their relative alignment remains unchanged.

Select the next pair to be merged.

Alignment progressively built, with each step being treated as a pairwise alignment.

Each member of a 'pair' might have more than one sequence.



*ClustalW*: Thompson *et al.*, 1994. 10.1093/nar/22.22.4673.

第三步：按照指导树进行渐进式比对

首先比对最相似的两个序列：根据前一步骤生成的指导树，选出最近亲的两个序列首先进行比对。

比对结果是固定的：一旦这两个序列比对完成，比对结果就固定下来，以后不会改变。这意味着如果在后续的步骤中需要在序列中引入空位（gap），这个空位会被加入到所有相关的序列中相同的位置。

选择下一对进行合并：接着选择下一对最相关的序列或者已经比对过的序列组，并进行比对合并。

渐进式构建比对结果：每进行一步比对，就将新的比对结果作为一个整体，用于与下一个序列或者序列组进行比对。这个过程会不断重复，直到所有序列都被合并到比对结果中。

对于‘一对’可能包含不止一个序列：在进行比对的时候，‘一对’可能指的是单一序列，也可能是已经比对好的一组序列。

图中的树状图展示了蓝色的节点代表了合并点，指导了序列比对的合并顺序。

ClustalW，这是一个用于多序列比对的计算程序，
常用在生物信息学中比对DNA或蛋白质序列。

# ClustalW

不保证收敛到最优解：ClustalW作为一个启发式方法，并不保证能够找到序列比对的最优解。启发式算法是基于直觉或经验设计的，旨在解决复杂的问题，在合理的时间内给出一个足够好的解，但不保证是最好的。

一旦创建了间隔（gap），唯一允许的改变是它的扩展：在ClustalW中，当在序列中引入了一个间隔后，随后只允许对这个间隔进行扩展，而不能再次改变其位置或将其从序列中移除。

- Does not guarantee convergence to optimal solutions (it is a heuristic)
- Once a gap is created the only allowed change is its extension
- The final alignment is affected by the quality of the initial alignment
- Time complexity
  - $O(k^2 \log k)$

最终的比对结果受初始比对质量的影响：这意味着如果初始序列的比对不准确，那么最终结果也可能不理想。初始比对的质量对整个比对过程非常关键。

*ClustalW*: Thompson *et al.*, 1994. 10.1093/nar/22.22.4673.

# MSA methods

- Optimal alignment 2↑
  - Extension of the dynamic programming approaches
- Heuristic
  - Progressive alignment (ClustalW)
    - Build a tree of similar sequences based on pairwise alignments
    - Perform agglomerative clustering via neighbor joining
  - Iterative methods (MAFFT, MUSCLE)
    - Genetic Algorithms
  - Probabilistic (*e.g.* Hidden Markov Models)

# Thank you!

Don’t forget assignment 1!
***Due tomorrow!!***

**Today:** Comparing Sequences

**Next time:** Advanced Indexing