Hashing and Sketching

CS 4390/5390
Fall 2019

Comparison can be slow

We know calculating local alignments is $O(n^2)$

- in the case of read overlapping if there are say 10^6 reads
- if reads are 10^2 bases each, thats 10^{10} computations!

Even hamming distance ($O(n)$) may be too slow.

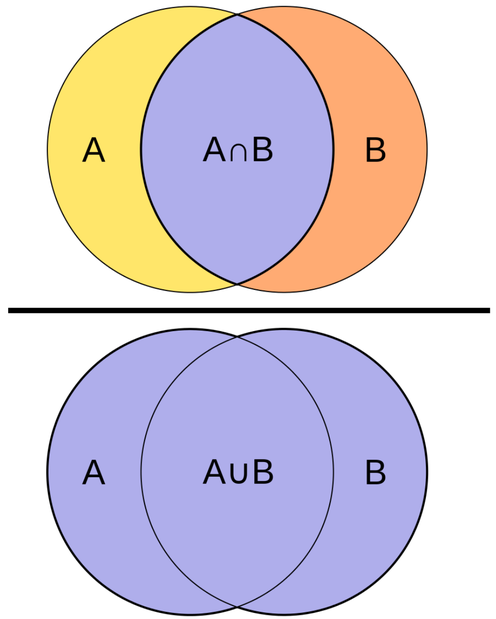
Remember, finding overlaps is just step 1 of assembly!

Jaccard Similarity

Measures the similarity of two sets of items *A* and *B* as:

$$J(A,B) = \frac{\left|A \cap B\right|}{\left|A \cup B\right|} = \frac{\left|A \cap B\right|}{|A| + |B| - \left|A \cap B\right|}$$

$$J(A,B) =$$

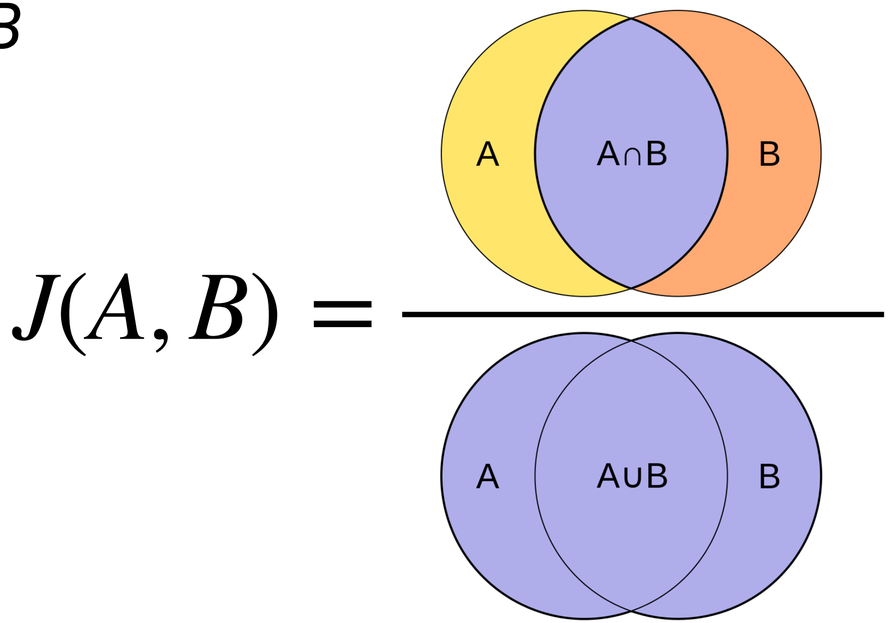


Images courtesy of Wikimedia Commons.

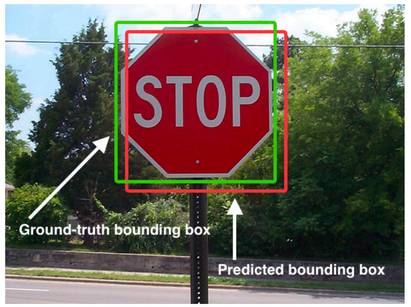
Jaccard Similarity

Measures the similarity of two sets of items *A* and *B* as:

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



Used also used in computer vision, sometimes called the "Intersection over Union" (IoU) metric

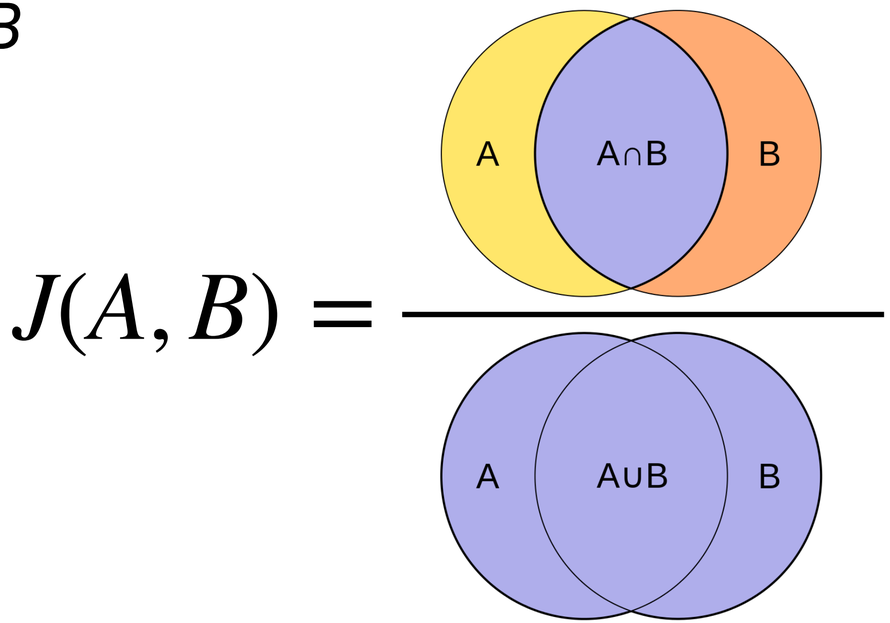


Images courtesy of Wikimedia Commons.

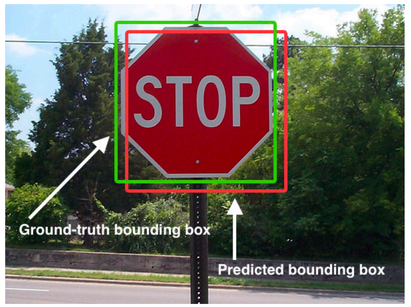
Jaccard Similarity

Measures the similarity of two sets of items *A* and *B* as:

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



Used also used in computer vision, sometimes called the "Intersection over Union" (IoU) metric

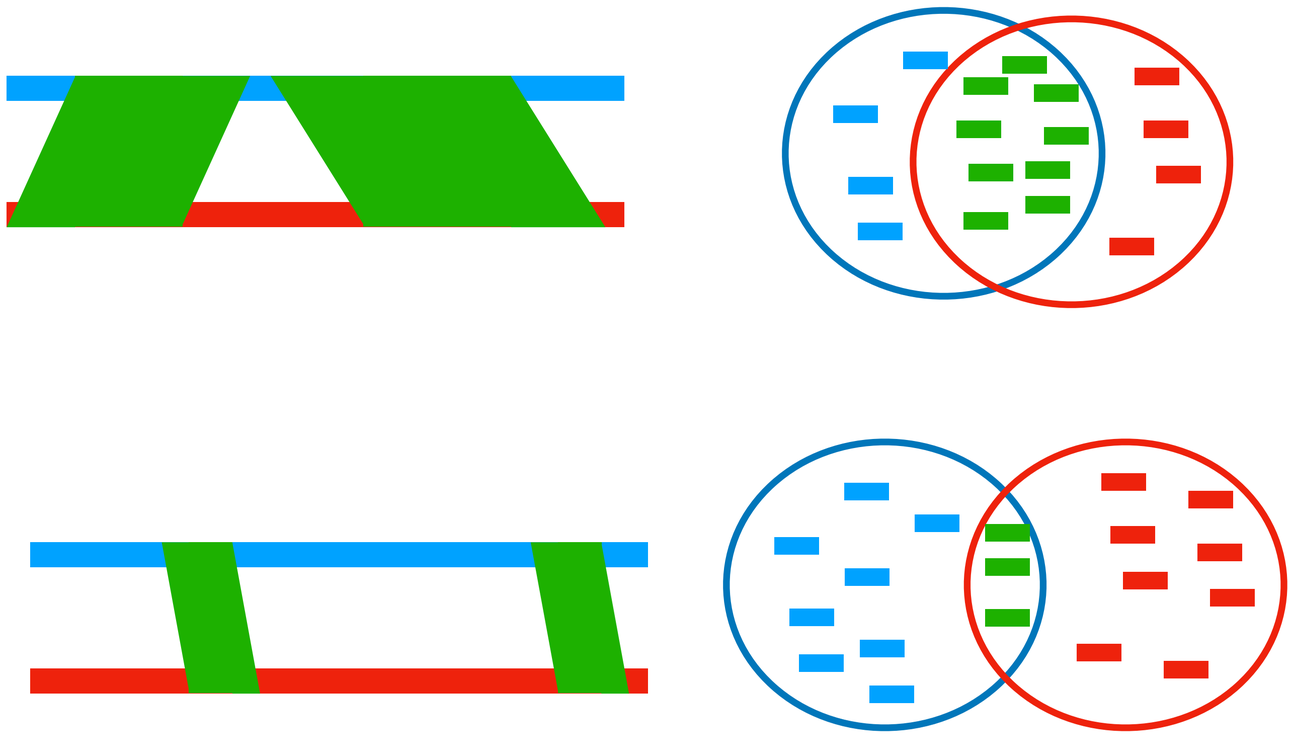


How would we use Jaccard for sequences?

Images courtesy of Wikimedia Commons.

Jaccard Similarity

In sequence analysis we construct a sets of *k*-mers for each of the strings being compared



Min-Hash Sketch

Calculating the union and intersection of a set of anything (in particular k-mers) can be time consuming (*O(n)* time)

Can we calculate it faster?

Min-Hash Sketch

Calculating the union and intersection of a set of anything (in particular k-mers) can be time consuming (*O(n)* time)

Can we calculate it faster?

Consider the following scenario:

- given a hash function on *k*-mers $h: \Sigma^k \rightarrow \mathbb{Z}^+$
- and the sets of *k*-mers for two string *A* and *B,*
- What is the probability that $min_{c \in A}\left\{h(c)\right\} = min_{c \in B}\left\{h(c)\right\}$?

Min-Hash Sketch

Calculating the union and intersection of a set of anything (in particular k-mers) can be time consuming (*O(n)* time)

Can we calculate it faster?

Consider the following scenario:

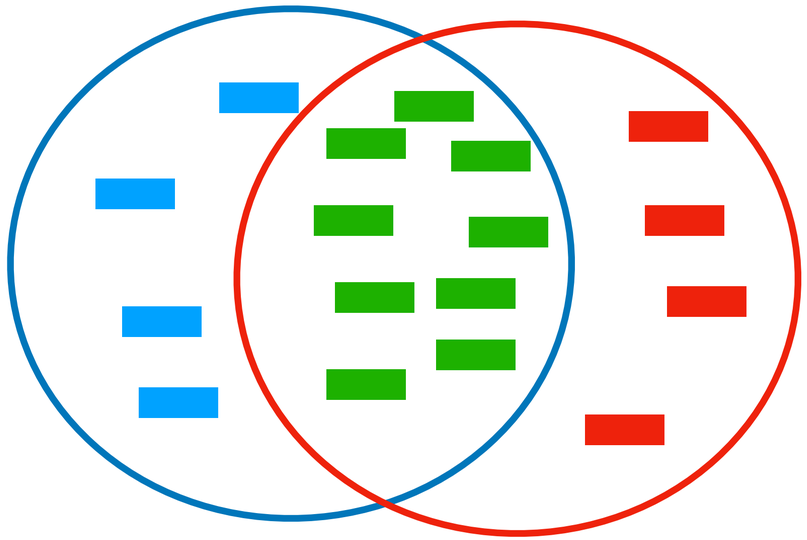
- given a hash function on *k*-mers $h: \Sigma^k \rightarrow \mathbb{Z}^+$
- and the sets of *k*-mers for two string *A* and *B,*
- What is the probability that $min_{c \in A}\left\{h(c)\right\} = min_{c \in B}\left\{h(c)\right\}$?

Turns out that

$$Pr_h\left[min_{c \in A}\left\{h(c)\right\} = min_{c \in B}\left\{h(c)\right\}\right] = J(A, B)$$

Min-Hash Sketch

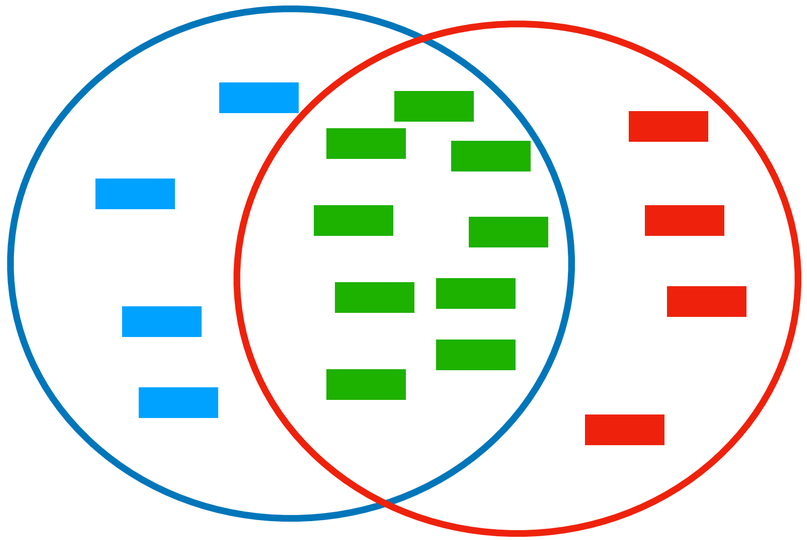
Why is $Pr_h\left[min_{c \in A}\{h(c)\} = min_{c \in B}\{h(c)\}\right] = J(A, B)$?



Min-Hash Sketch

Why is $Pr_h\left[min_{c \in A}\{h(c)\} = min_{c \in B}\{h(c)\}\right] = J(A, B)$?

Think of *h* as applying a randomized ordering on the *k*-mers.

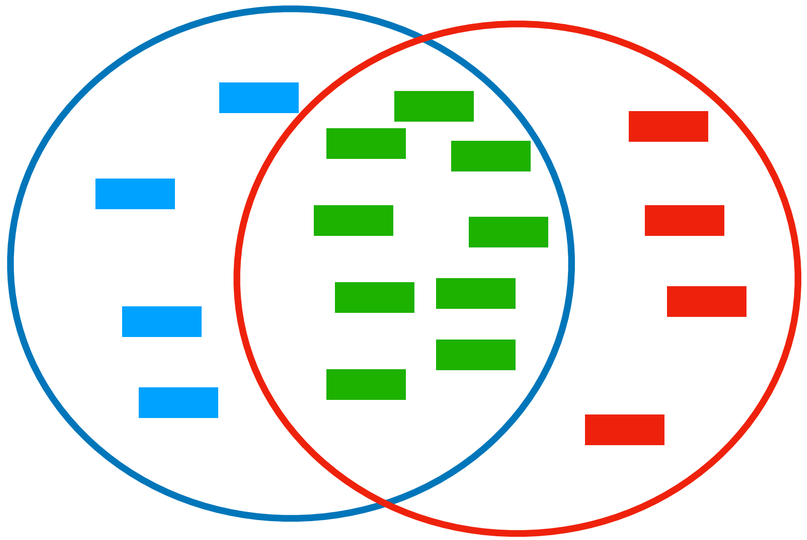


Min-Hash Sketch

Why is $Pr_h\left[min_{c \in A}\{h(c)\} = min_{c \in B}\{h(c)\}\right] = J(A, B)$?

Think of *h* as applying a randomized ordering on the *k*-mers.

If the minimum *k*-mer from the union is in the intersection, it will be minimum for both *A* and *B.*



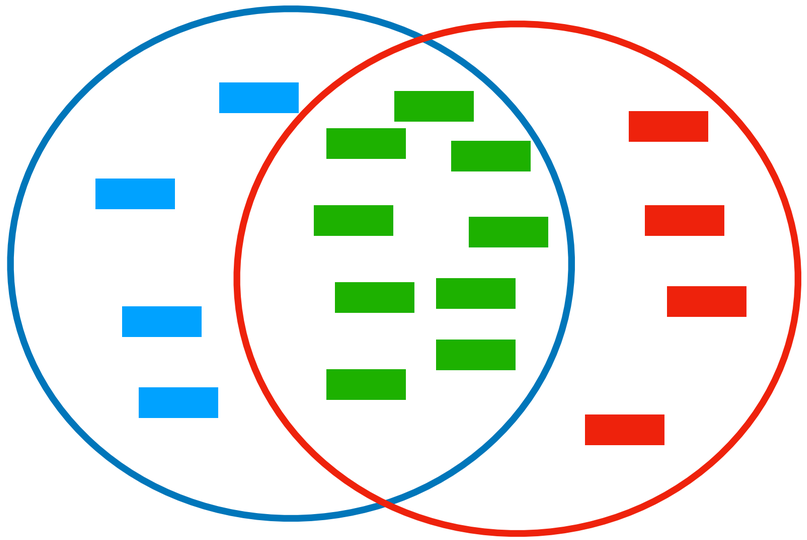
Min-Hash Sketch

Why is $Pr_h\left[min_{c \in A}\{h(c)\} = min_{c \in B}\{h(c)\}\right] = J(A, B)$?

Think of *h* as applying a randomized ordering on the *k*-mers.

If the minimum *k*-mer from the union is in the intersection, it will be minimum for both *A* and *B.*

How many minimum *k*-mers from the union can we choose?



Min-Hash Sketch

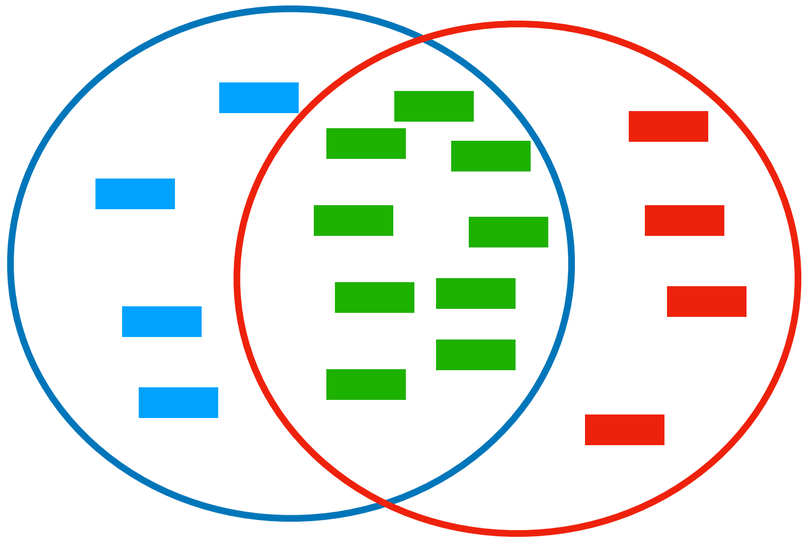
Why is $Pr_h\left[min_{c \in A}\{h(c)\} = min_{c \in B}\{h(c)\}\right] = J(A, B)$?

Think of *h* as applying a randomized ordering on the *k*-mers.

If the minimum *k*-mer from the union is in the intersection, it will be minimum for both *A* and *B.*

How many minimum *k*-mers from the union can we choose?

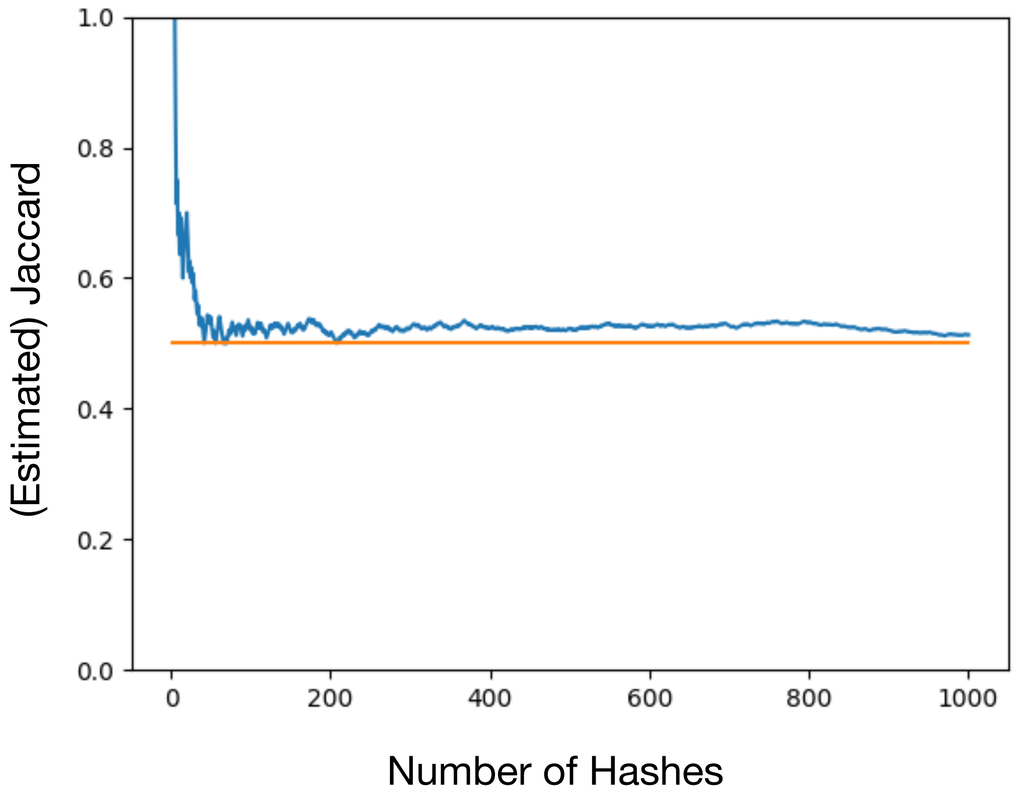
What fraction of those are in the intersection?



Min-Hash Sketch

As you increase the number of hashes, you will get closer to the estimate of the real jaccard value

Finding that many independent hashes may be hard

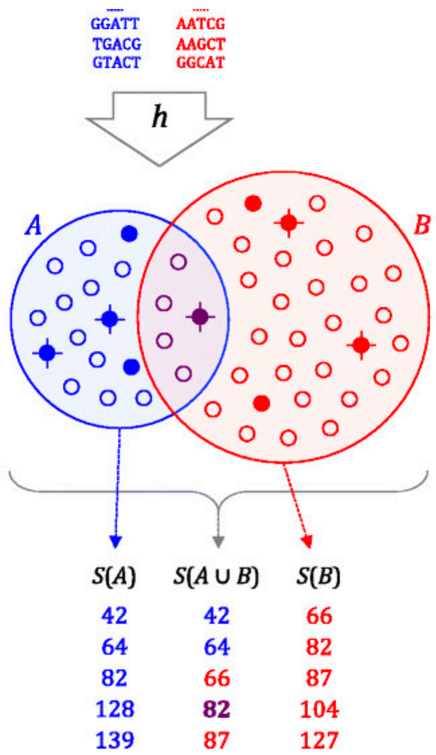


Min Hash Sketch with 1 Hash

The idea is that you choose the minimum *n* elements according to the hash *h*, and compute jaccard on these subsets

This subset of *k*-mers is called a "sketch"

Sometimes called "MinHash bottom sketching"



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

Image credit: Ondov, et al. (2016) *Mash: Fast genome and metagenome distance estimation using MinHash.* **Genome Biology.**

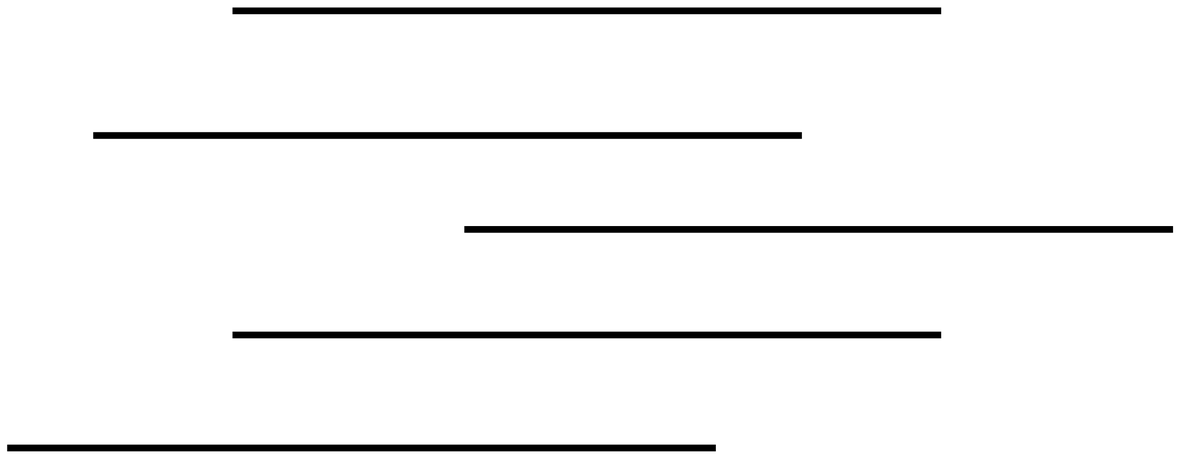
Minimizer Schemes

Another way to *sketch* a sequence is though the use of minimizer schemes

Here a set of *k*-mers for a sequence are selected by finding the minimum *k*-mer in overlapping windows

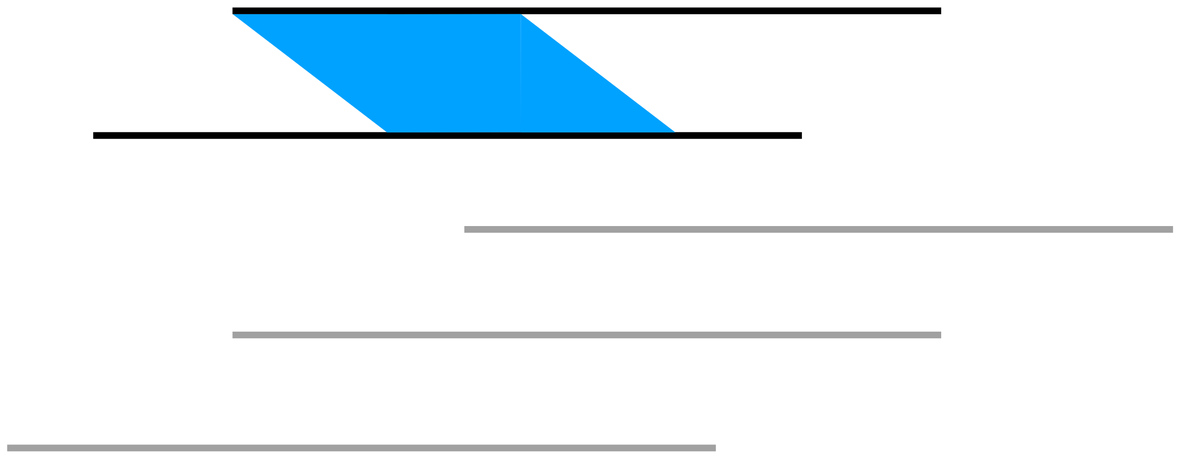
Minimizer Schemes

Roberts, *et al.* (2004) introduced minimizer schemes as a way to decrease the time needed for sequence overlap computation



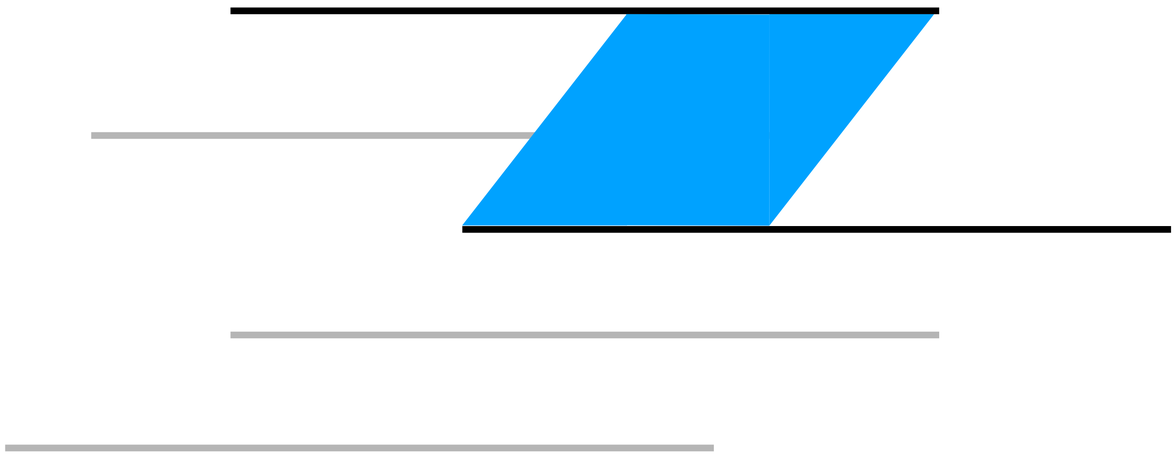
Minimizer Schemes

Roberts, *et al.* (2004) introduced minimizer schemes as a way to decrease the time needed for sequence overlap computation



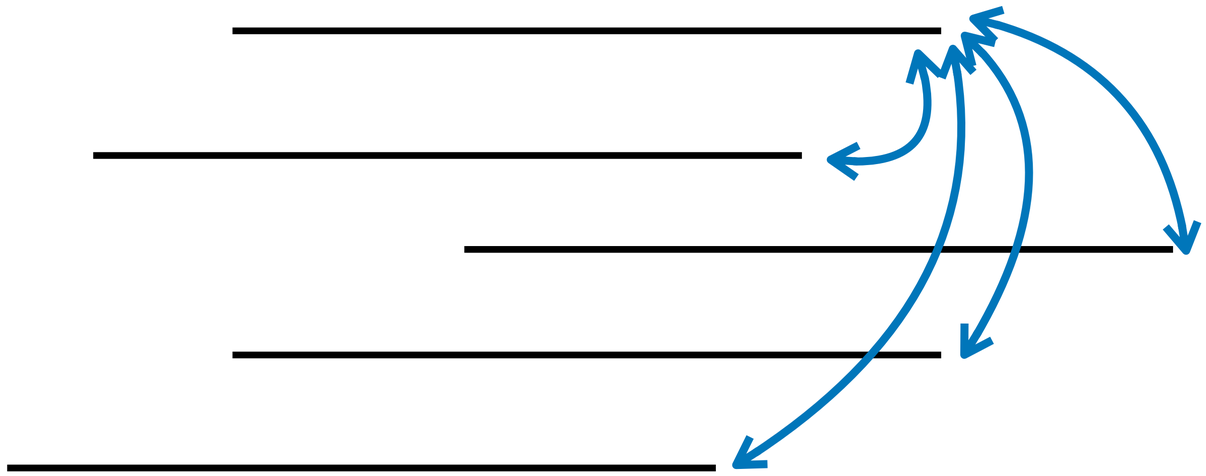
Minimizer Schemes

Roberts, *et al.* (2004) introduced minimizer schemes as a way to decrease the time needed for sequence overlap computation



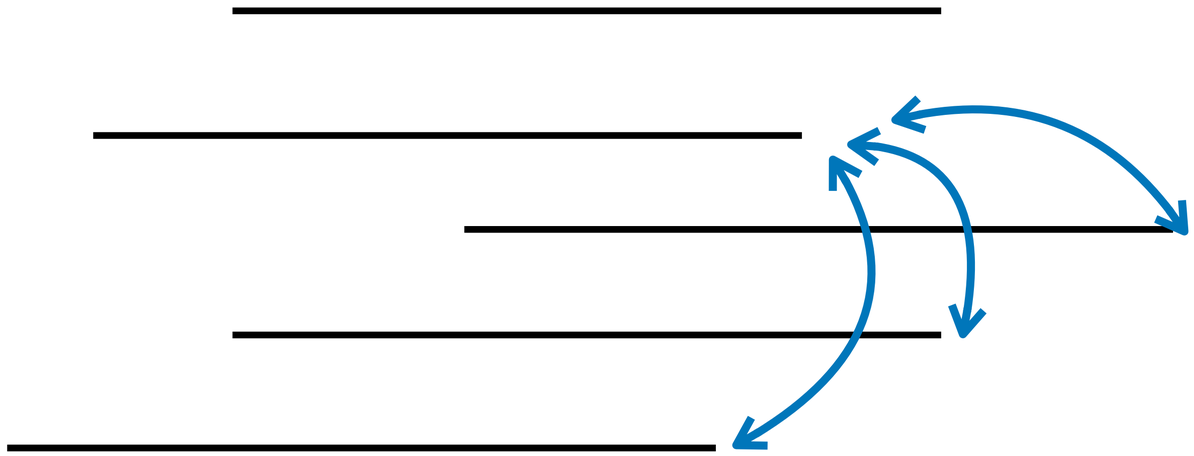
Minimizer Schemes

Roberts, *et al.* (2004) introduced minimizer schemes as a way to decrease the time needed for sequence overlap computation



Minimizer Schemes

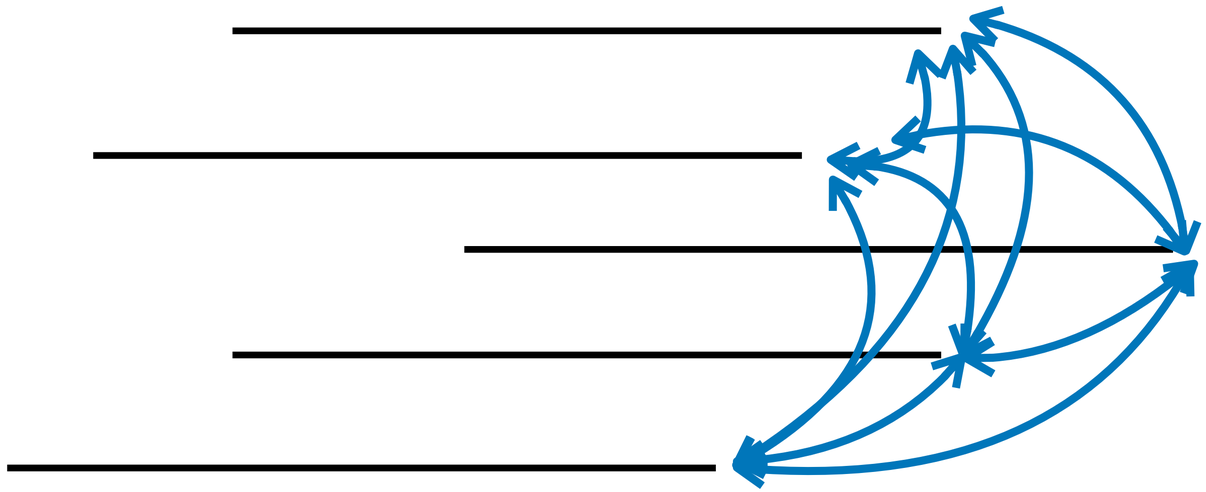
Roberts, *et al.* (2004) introduced minimizer schemes as a way to decrease the time needed for sequence overlap computation



Minimizer Schemes

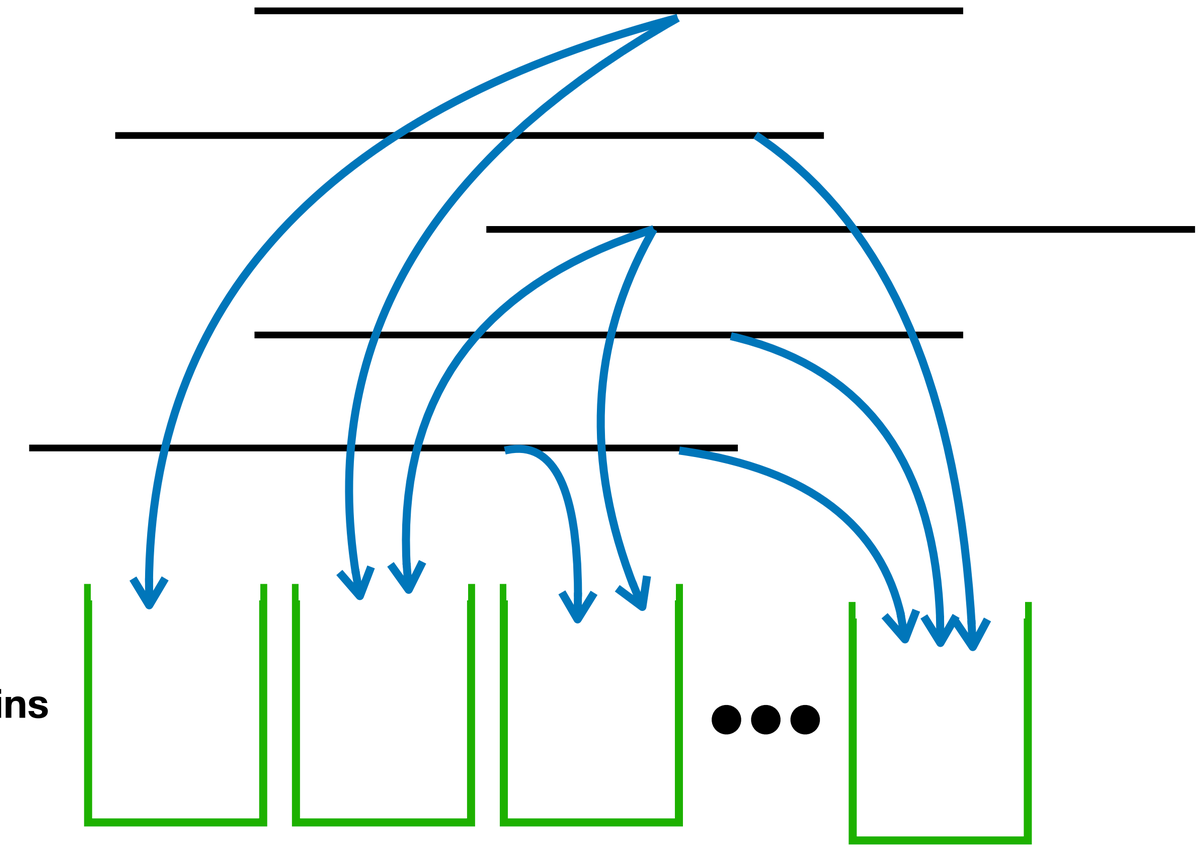
$O(n^2)$ alignments!

Roberts, *et al.* (2004) introduced minimizer schemes as a way to decrease the time needed for sequence overlap computation



Minimizer Schemes

Roberts, *et al.* (2004) introduced minimizer schemes as a way to decrease the time needed for sequence overlap computation



Only compare within bins

Minimizer Schemes

- Minimizer schemes have two special properties:
  - two sequences with a long exact match must select the same k-mers
  - there are no large gap between selected k-mers

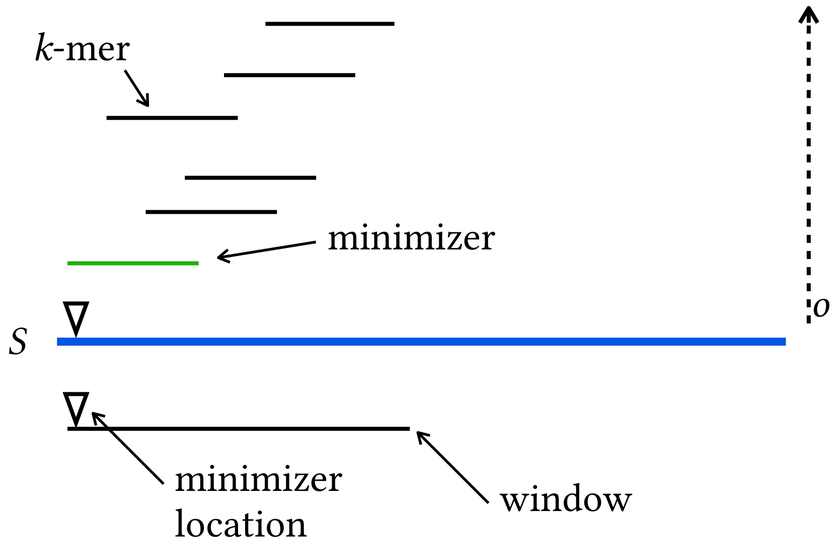
Minimizer Schemes

- Minimizer schemes have two special properties:
  - two sequences with a long exact match must select the same k-mers
  - there are no large gap between selected k-mers

- Use in k-mer counting, *de Brujin* graph construction, data structure sparsification, etc.

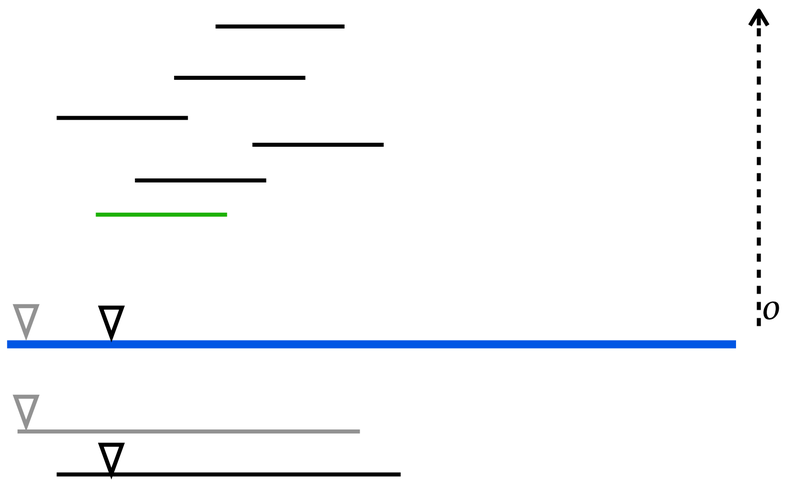
Minimizer Schemes

For a windows of w consecutive k-mers from a sequence S, a minimizer scheme selects the minimum according to an ordering o as a representative



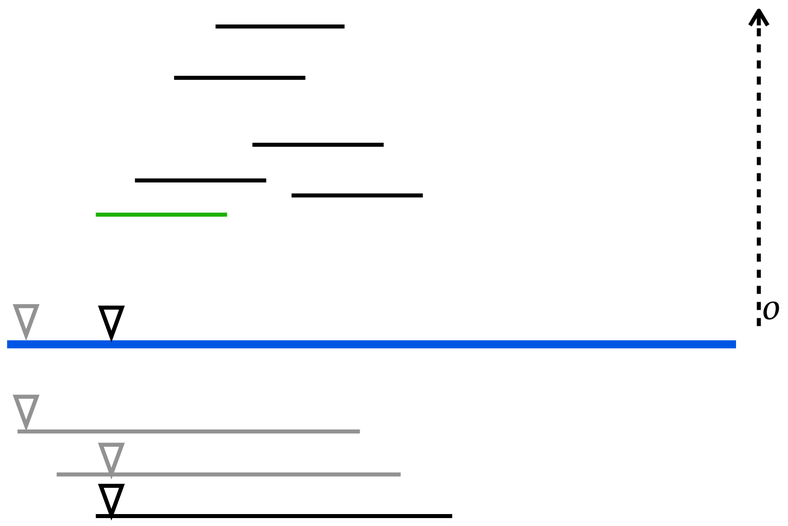
Minimizer Schemes

For a windows of w consecutive k-mers from a sequence S, a minimizer scheme selects the minimum according to an ordering o as a representative



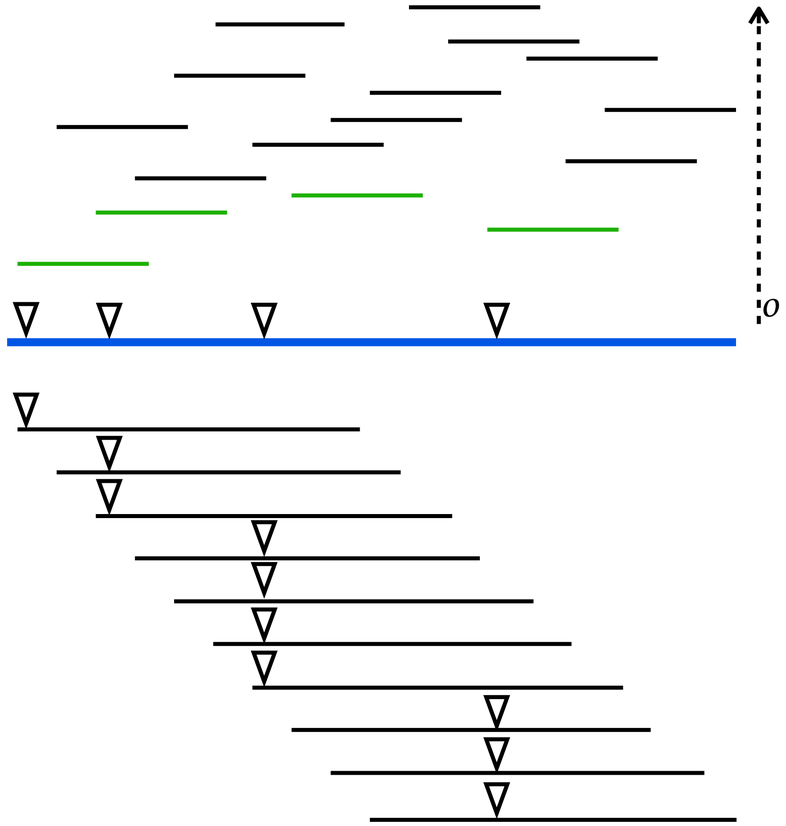
Minimizer Schemes

For a windows of w consecutive k-mers from a sequence S, a minimizer scheme selects the minimum according to an ordering o as a representative



Minimizer Schemes

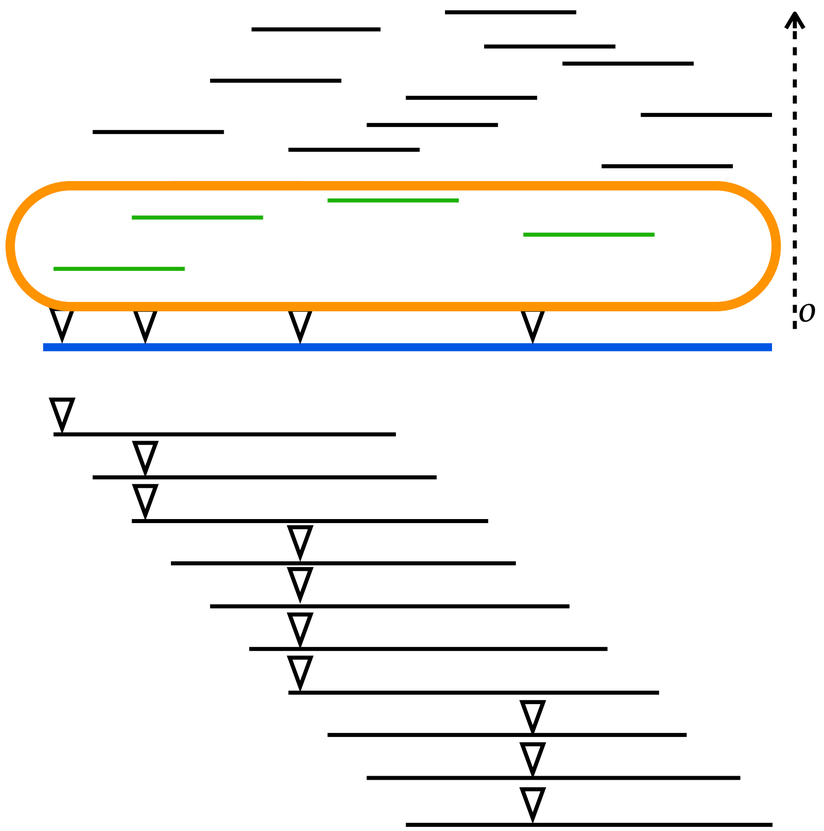
- Changing the ordering used can greatly impact the number of unique minimizers
- Can we find an order that minimizes the number of minimizer locations



Minimizer Schemes

- Changing the ordering used can greatly impact the number of unique minimizers
- Can we find an order that minimizes the number of minimizer locations

Only some k-mers are used a minimizers

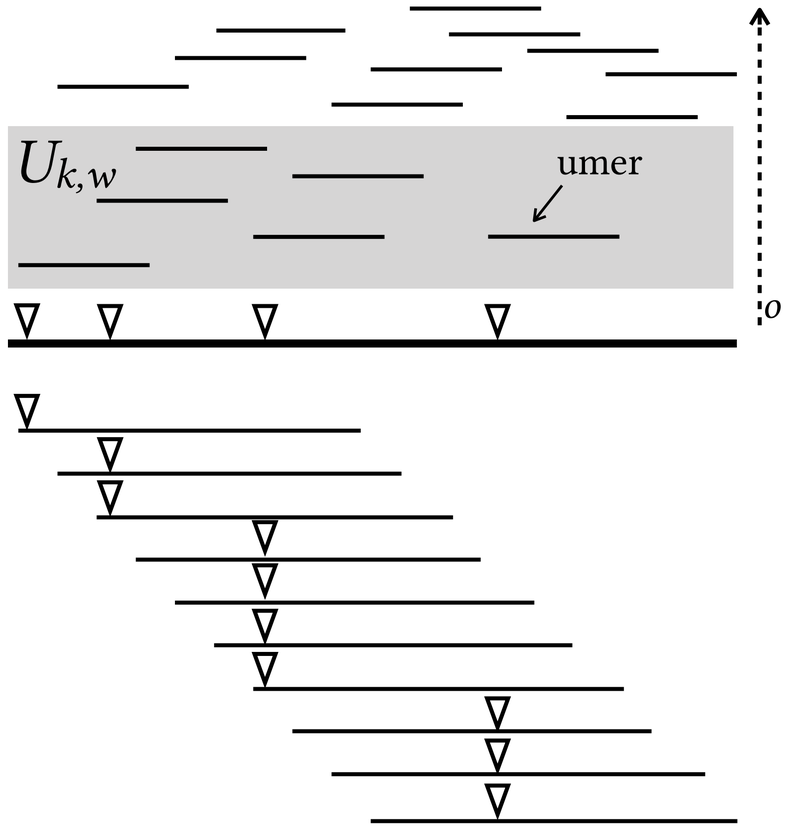


Universal k-mer Set

A universal k-mer set $U_{k,w} \subseteq \Sigma^k$ is a set of k-mers such that any window of w consecutive k-mers must contain at least one element from the set

Universal k-mer Set and Minimizer Ordering

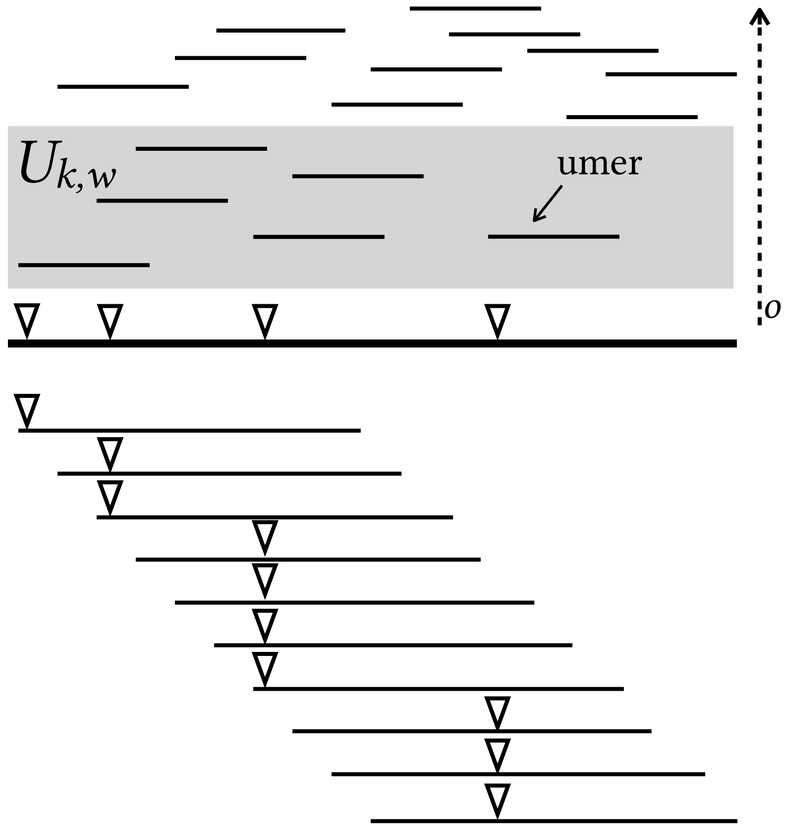
- A universal k-mer set induces a family of compatible orderings
- Orderings based on universal sets have better performance then lexicographic or random orders (Marçais, et al., 2017)



Universal k-mer Set and Minimizer Ordering

- A universal k-mer set induces a family of compatible orderings

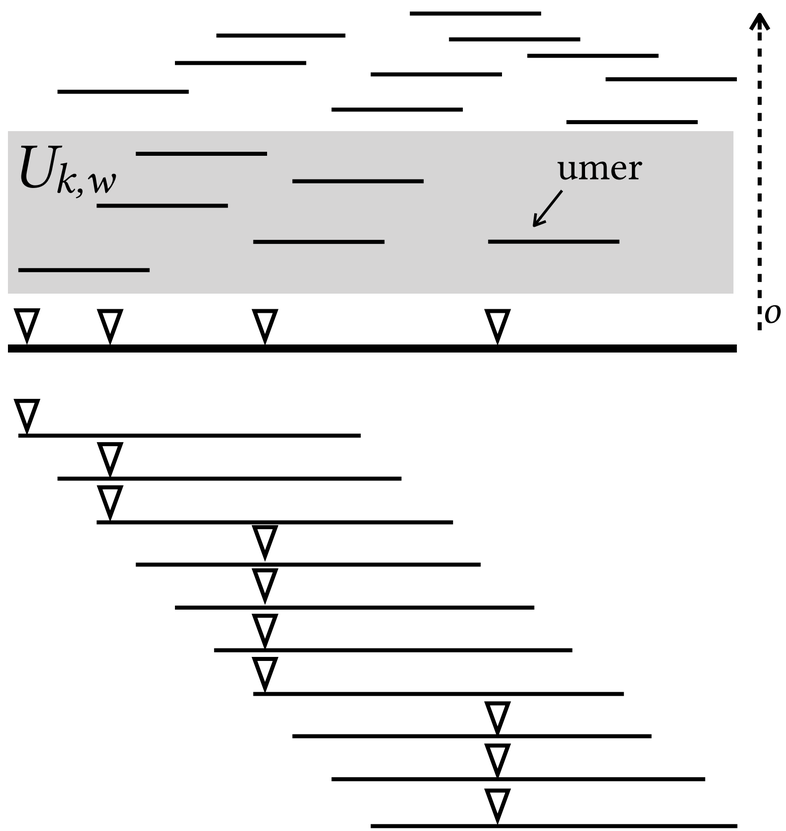
- Orderings based on universal sets have better ***performance*** then lexicographic or random orders (Marçais, et al., 2017)



Universal k-mer Set and Minimizer Ordering

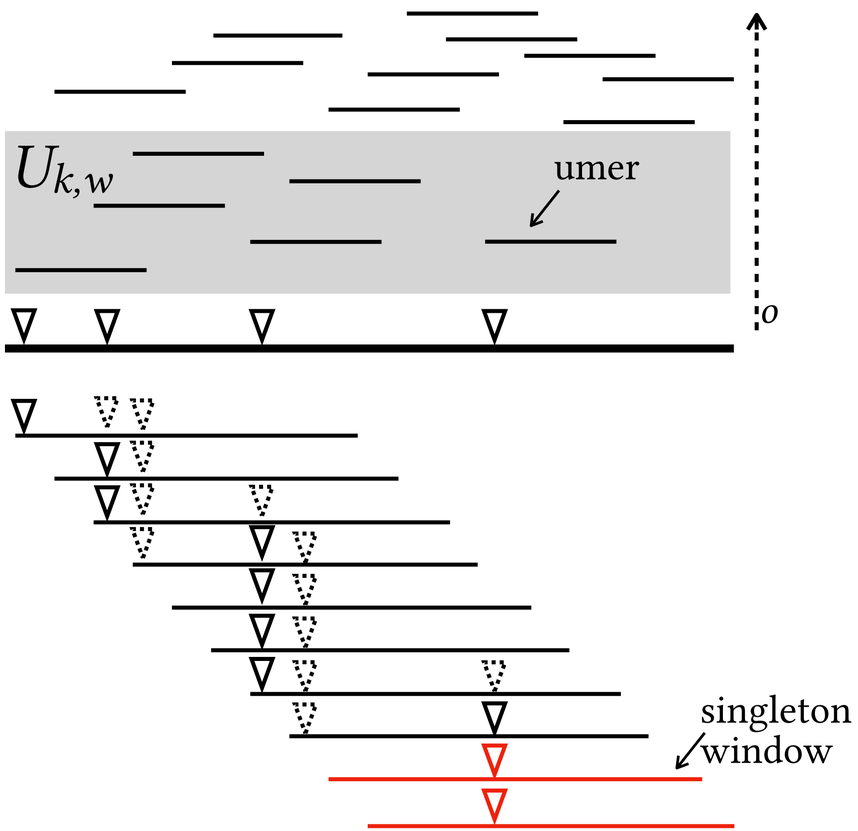
- Set Size
  - Fraction of all k-mers in the universal set

- Density
  - Normalized count of minimizer locations in S



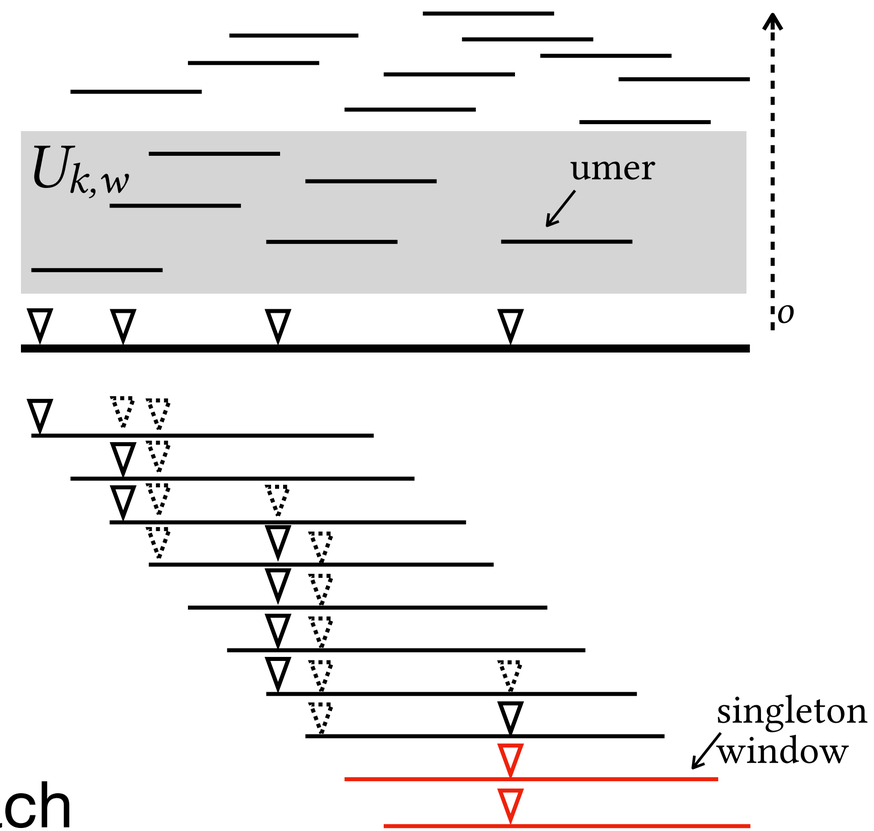
Universal k-mer Set and Minimizer Ordering

- Set Size
  - Fraction of all k-mers in the universal set
- Density
  - Normalized count of minimizer locations in S
- Sparsity
  - Normalized count of windows in S with only one umer (universal k-mer)



Universal k-mer Set and Minimizer Ordering

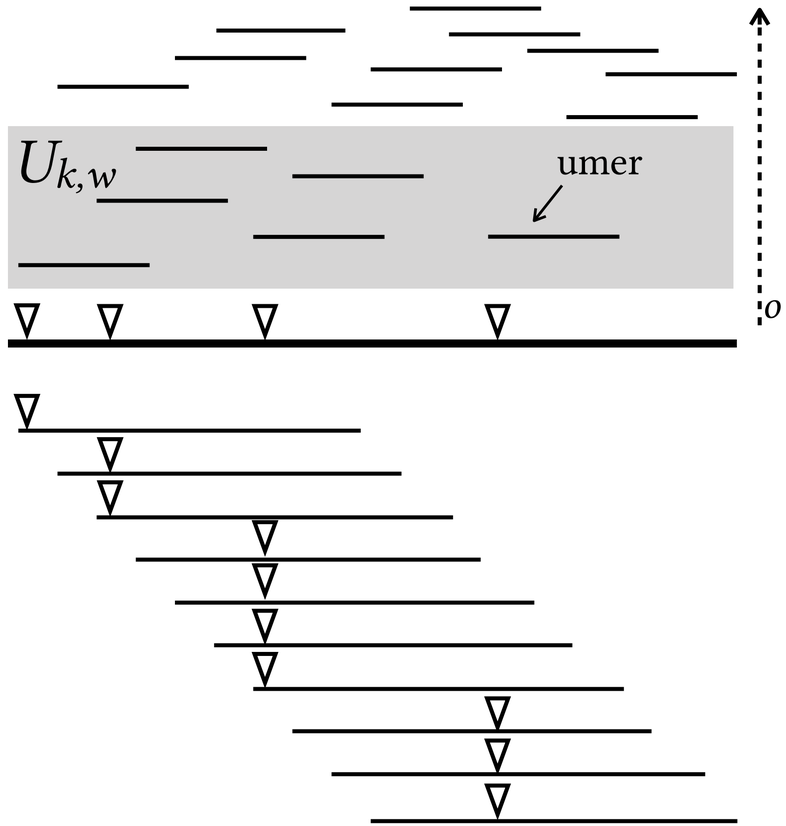
- Set Size
  - Fraction of all k-mers in the universal set
- ***Expected*** Density
  - Normalized count of minimizer locations in B_L
- ***Expected*** Sparsity
  - Normalized count of windows in B_L with only one umer (universal k-mer)



B_L is the **de Brujin** sequence of order L, it contains each window exactly once

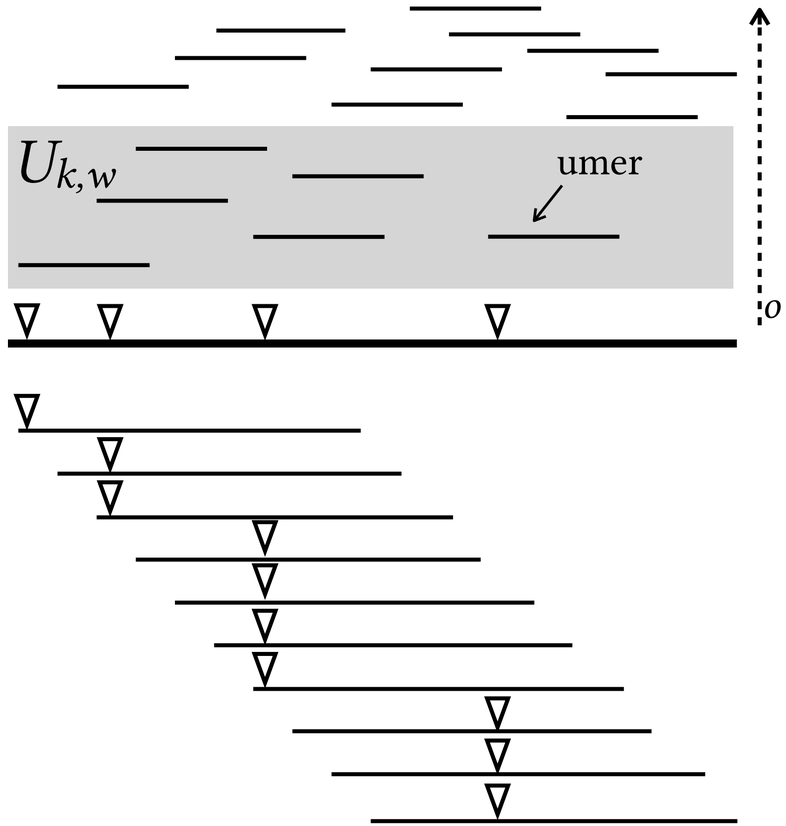
Universal k-mer Set and Minimizer Ordering

- A universal k-mer set induces a family of compatible orderings
- Orderings based on universal sets have better performance then lexicographic or random orders (Marçais, et al., 2017)



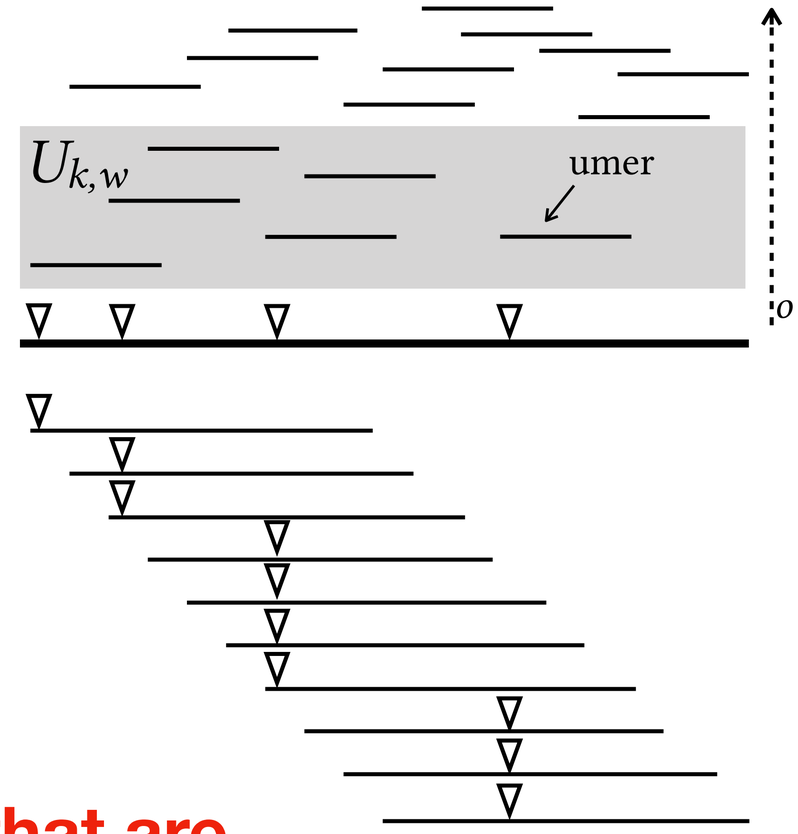
Universal k-mer Set and Minimizer Ordering

- A universal k-mer set induces a family of compatible orderings
- Orderings based on universal sets have better performance then lexicographic or random orders (Marçais, et al., 2017)
- Current methods cannot construct sets for values of k and w used in practice



Universal k-mer Set and Minimizer Ordering

- A universal k-mer set induces a family of compatible orderings
- Orderings based on universal sets have better performance then lexicographic or random orders (Marçais, et al., 2017)
- Current methods cannot construct sets for values of k and w used in practice



Can we construct universal k-mer sets that are practical for use in minimizer schemes?

Problems with Jaccard