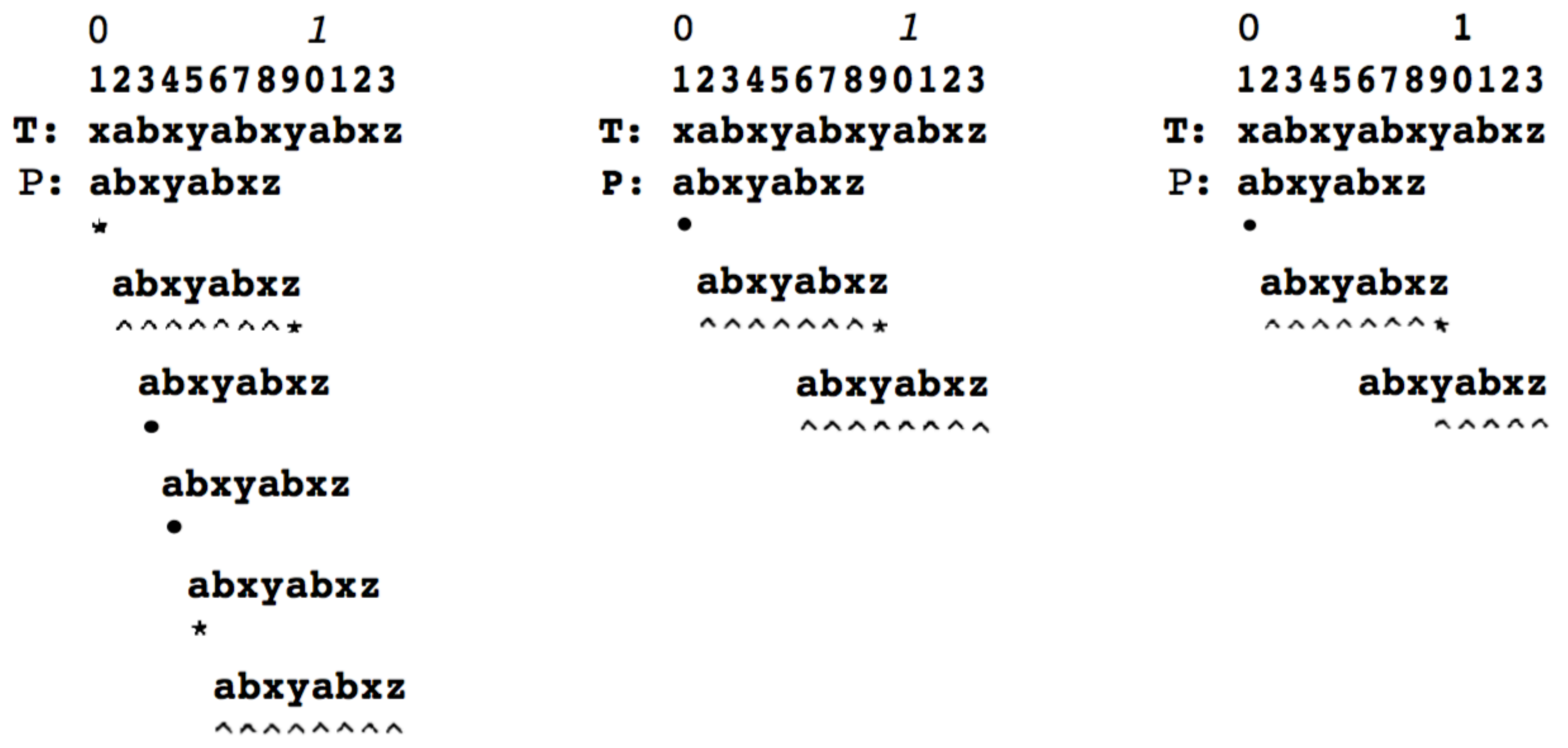
Alignment-free genomics

Sequence Search



```
   0          1
   1234567890123
T: xabxyabxyabxz
P: abxyabxz
   *
    abxyabxz
    ^^^^^^^*
     abxyabxz
     •
      abxyabxz
      •
       abxyabxz
       *
        abxyabxz
        ^^^^^^^^
```

```
   0          1
   1234567890123
T: xabxyabxyabxz
P: abxyabxz
   •
    abxyabxz
    ^^^^^^^*
        abxyabxz
        ^^^^^^^^
```

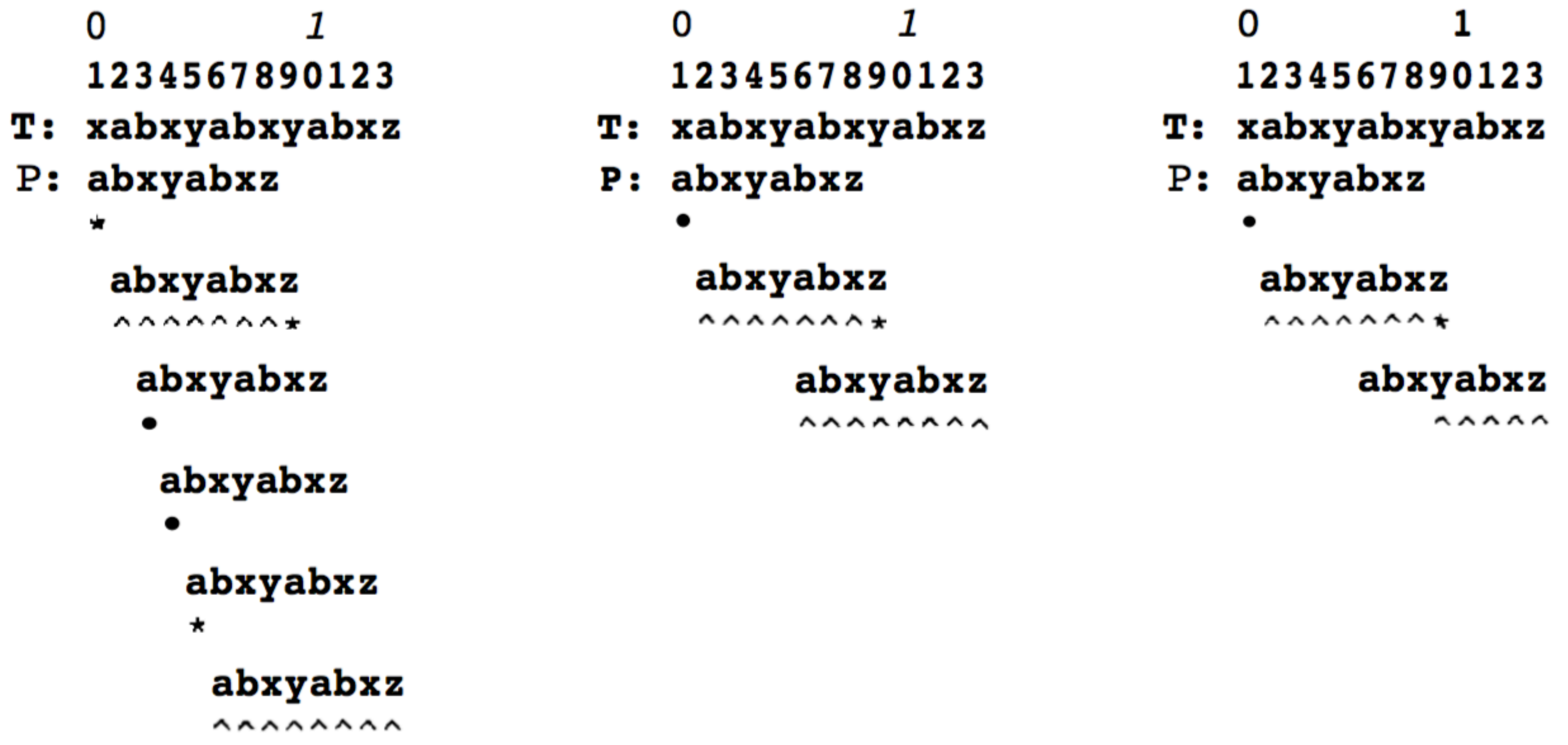
```
   0          1
   1234567890123
T: xabxyabxyabxz
P: abxyabxz
   •
    abxyabxz
    ^^^^^^^*
        abxyabxz
           ^^^^^
```

[1]Boyer, R.S.; Moore, J.S. (October 1977). "A Fast String Searching Algorithm". Comm. ACM. New York, NY, USA: Association for Computing Machinery. 20 (10): 762–772
[2]Knuth, D; Morris, J H.; Pratt, V (1977). "Fast pattern matching in strings". SIAM Journal on Computing. 6 (2): 323–350
image: Gusfield, D. Algorithms on Strings, Trees and Sequences. 1997. Figure 1.1

Sequence Search

Given a pattern `p` and a text `q`, find `p` in `q`

- naive solution is `O(mn)` time, `m=|p|, n=|q|`
- improved to `O(n+m)` by Boyer and More[1]
- later `O(n)` by Knuth, Morris and Pratt[2]

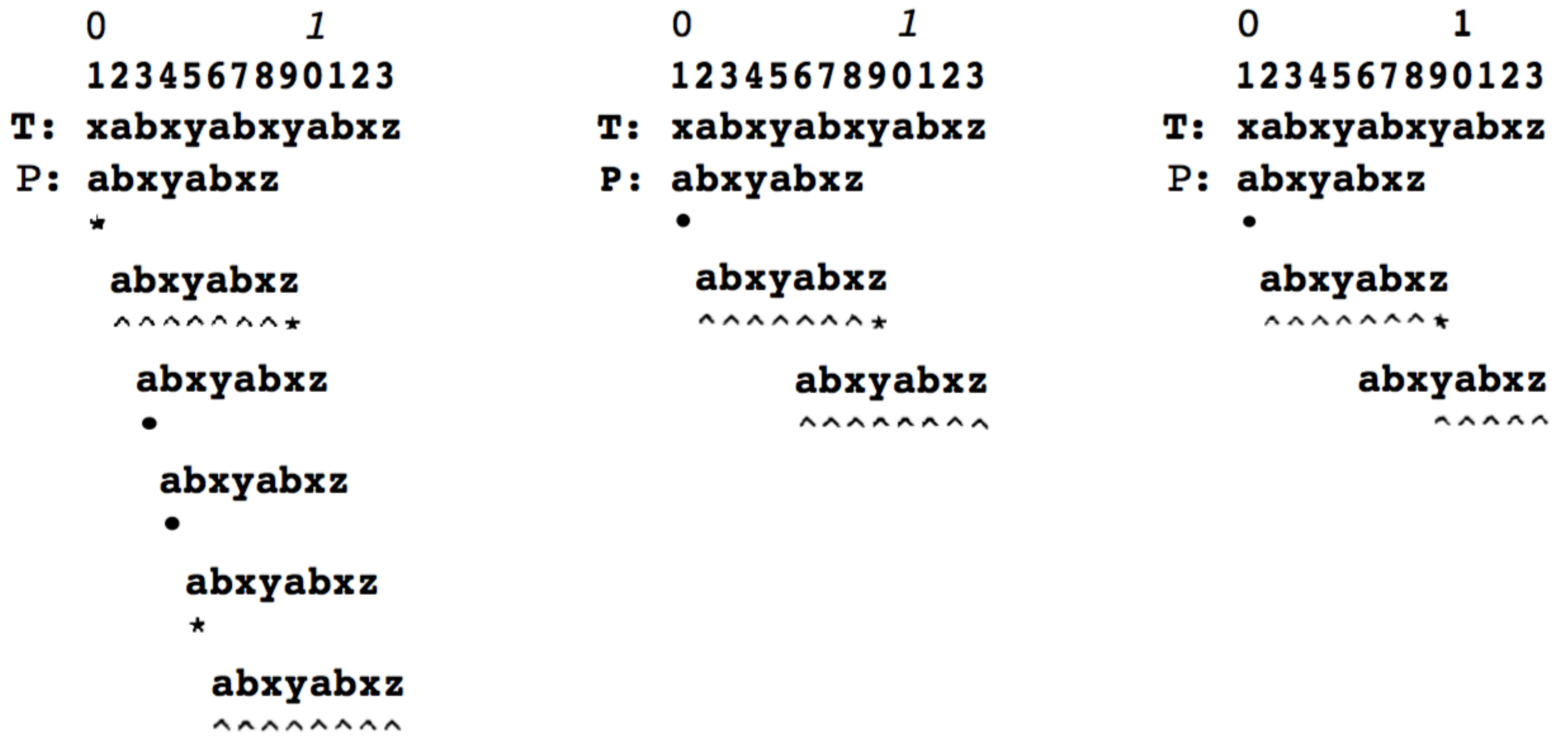


[1]Boyer, R.S.; Moore, J.S. (October 1977). "A Fast String Searching Algorithm". Comm. ACM. New York, NY, USA: Association for Computing Machinery. 20 (10): 762–772
[2]Knuth, D; Morris, J H.; Pratt, V (1977). "Fast pattern matching in strings". SIAM Journal on Computing. 6 (2): 323–350
image: Gusfield, D. Algorithms on Strings, Trees and Sequences. 1997. Figure 1.1

Sequence Search

Given a pattern `p` and a text `q`, find `p` in `q`

- naive solution is `O(mn)` time, `m=|p|, n=|q|`
- improved to `O(n+m)` by Boyer and More[1]
- later `O(n)` by Knuth, Morris and Pratt[2]



What if `n` is very large?

[1]Boyer, R.S.; Moore, J.S. (October 1977). "A Fast String Searching Algorithm". Comm. ACM. New York, NY, USA: Association for Computing Machinery. 20 (10): 762–772
[2]Knuth, D; Morris, J H.; Pratt, V (1977). "Fast pattern matching in strings". SIAM Journal on Computing. 6 (2): 323–350
image: Gusfield, D. Algorithms on Strings, Trees and Sequences. 1997. Figure 1.1

Suffix Trie/Tree

```
P = abx
Q = xabxac
    123456
```

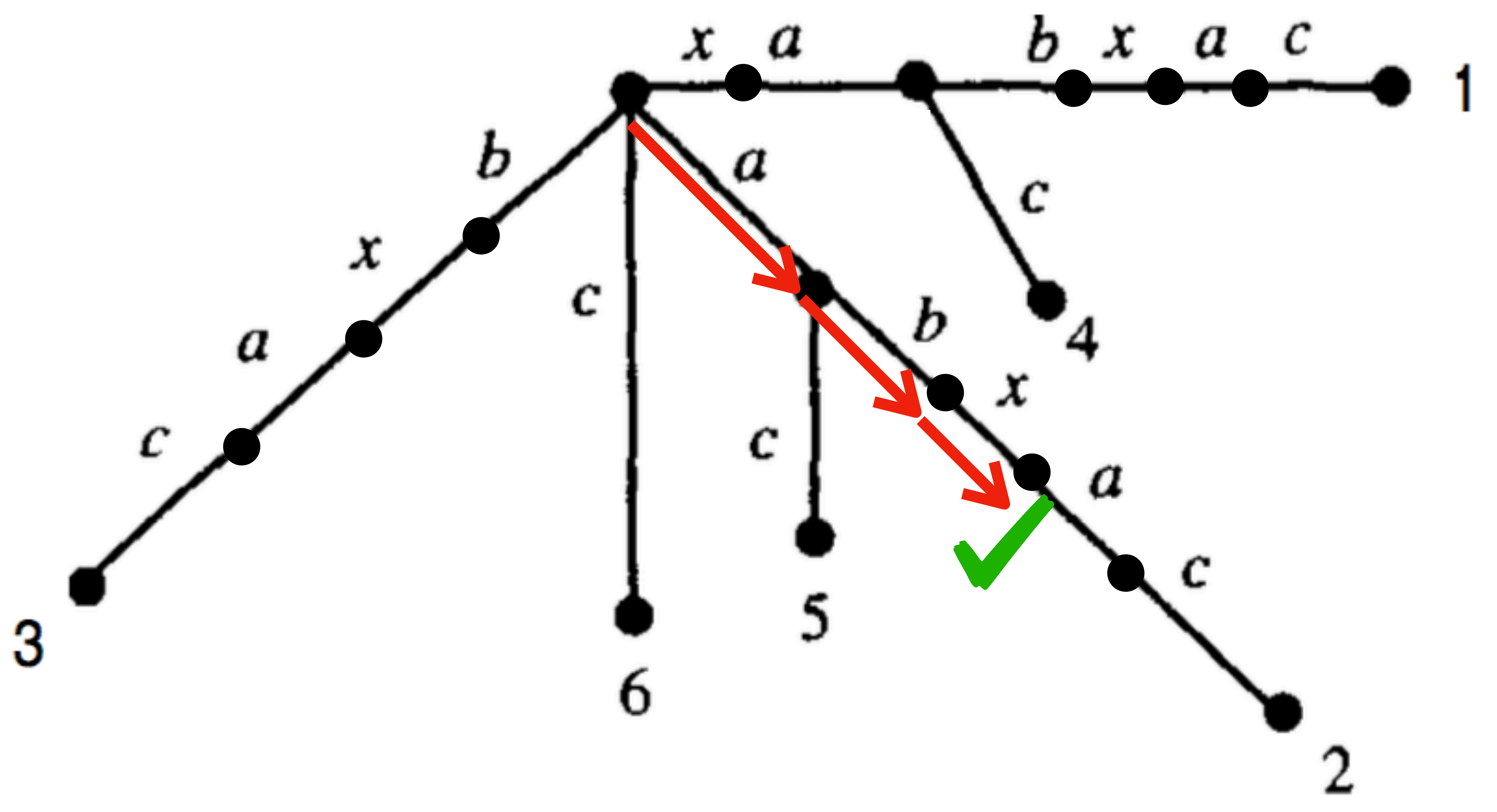


image: Gusfield, D. Algorithms on Strings, Trees and Sequences. 1997. Figure 5.1 (modified)

Suffix Trie/Tree

Let `T` be a rooted tree

- where each edge is labeled by a distinct character `a` ∈ Σ, and
- each leaf `l` labels a suffix of `q` such concatenating the labels of the edges from the root to `p` form the suffix.
- Finding `p` in `T` takes `O(m)`-time.
- Finding all instances is also faster.
- Requires Θ(`n`|Σ|) space!

```
P = abx
Q = xabxac
    123456
```

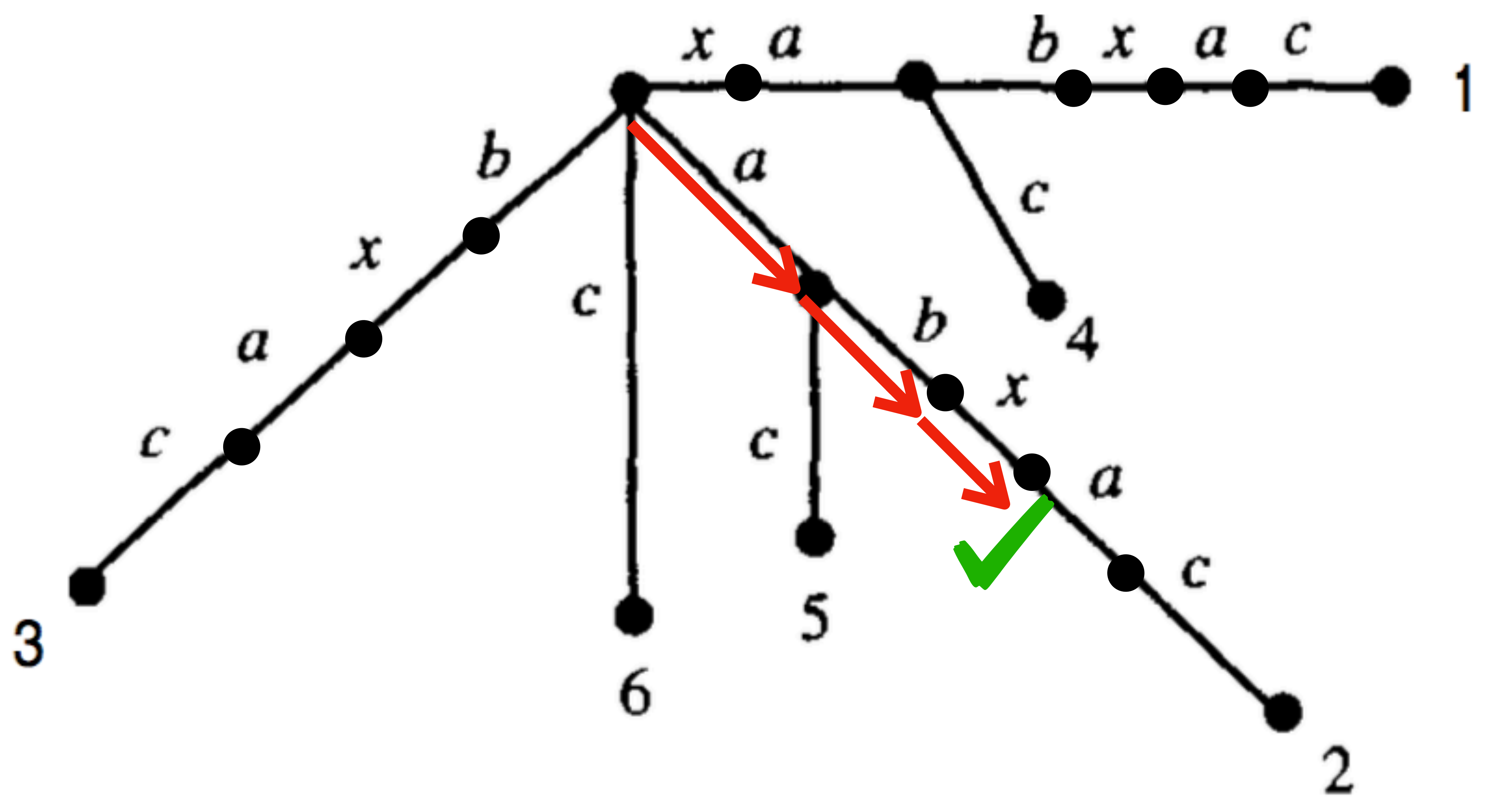


image: Gusfield, D. Algorithms on Strings, Trees and Sequences. 1997. Figure 5.1 (modified)

Suffix Trie/Tree

Let `T` be a rooted tree

- where each edge is labeled by a distinct character `a` ∈ Σ, and
- each leaf `l` labels a suffix of `q` such concatenating the labels of the edges from the root to `p` form the suffix.
- Finding `p` in `T` takes `O(m)`-time.
- Finding all instances is also faster.
- Requires Θ(`n`|Σ|) space!

But, what if `n = 3,000,000,000`?

```
P = abx
Q = xabxac
    123456
```

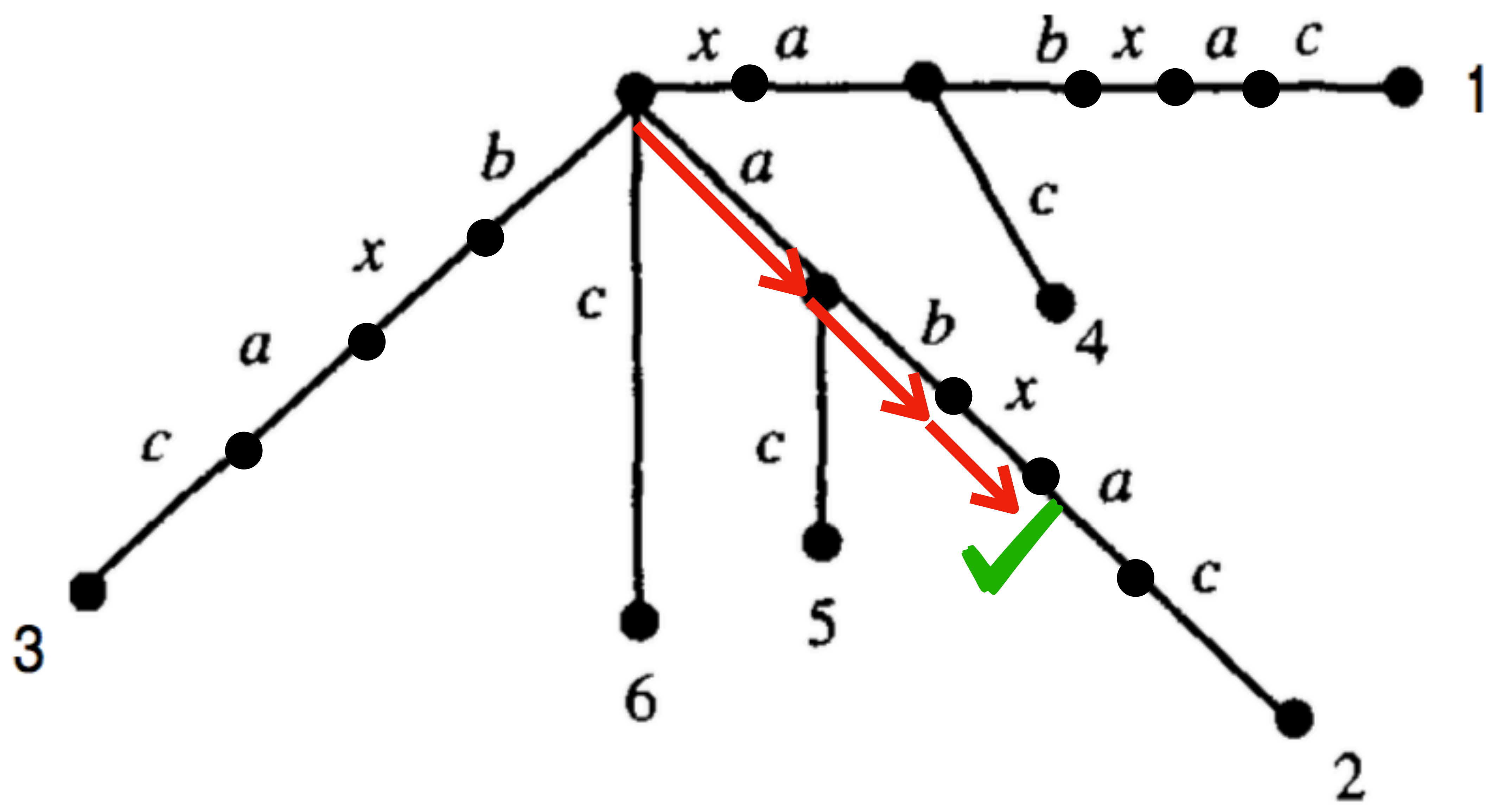


image: Gusfield, D. Algorithms on Strings, Trees and Sequences. 1997. Figure 5.1 (modified)

Suffix Arrays

Store two arrays

- `pos(i)` — which are the start position of suffixes in lexicographic order, and
- `lcp(i,j)` — which stores the longest common prefix between positions i and j.
- Takes O(n) space.
- Search can be conducted in O(m + log n)-time.

	pos	lcp	
0	12		`$`
1	11	0	`i$`
2	8	1	`ippi$`
3	5	1	`issippi$`
4	2	4	`ississippi$`
5	1	0	`missssissippi$`
6	10	0	`pi$`
7	9	1	`ppi$`
8	7	0	`sippi$`
9	4	2	`sissippi$`
10	6	1	`ssippi$`
11	3	3	`ssissippi$`

Burrows-Wheeler Transform

```
$mississippi
i$mississipp
ippi$mississ
issippi$miss
ississippi$m
mississippi$
pi$mississip
ppi$mississi
sippi$missis
sissippi$mis
ssippi$missi
ssissippi$mi
```

Burrows-Wheeler Transform

Store the last column of the rotated sorted suffix list

- Can be easily compressed because of the repetitiveness
- When used along with the genomic sequence can quickly recover the original sequence
- Ferrangina and Manzini later made advances for faster search

```
$mississippi
i$mississipp
ippi$mississ
issippi$miss
ississippi$m
mississippi$
pi$mississip
ppi$mississi
sippi$missis
sissippi$mis
ssippi$missi
ssissippi$mi
```

Burrows-Wheeler Transform

Store the last column of the rotated sorted suffix list

- Can be easily compressed because of the repetitiveness
- When used along with the genomic sequence can quickly recover the original sequence
- Ferrangina and Manzini later made advances for faster search

What if we want to find positions with some changes?

```
$mississippi
i$mississipp
ippi$mississ
issippi$miss
ississippi$m
mississippi$
pi$mississip
ppi$mississi
sippi$missis
sissippi$mis
ssippi$missi
ssissippi$mi
```

Alignment

Given

- two sequences $\mathtt{p}$ and $\mathtt{q}$ over an alphabet Σ, and
- an alignment objective function.

Find an m x 2 matrix ($m > \max(|\mathtt{p}|,|\mathtt{q}|)$)

- where each row represents one of the sequences with inserted gap characters (‘-’ $\notin \Sigma$), and
- is optimal under the objective function.

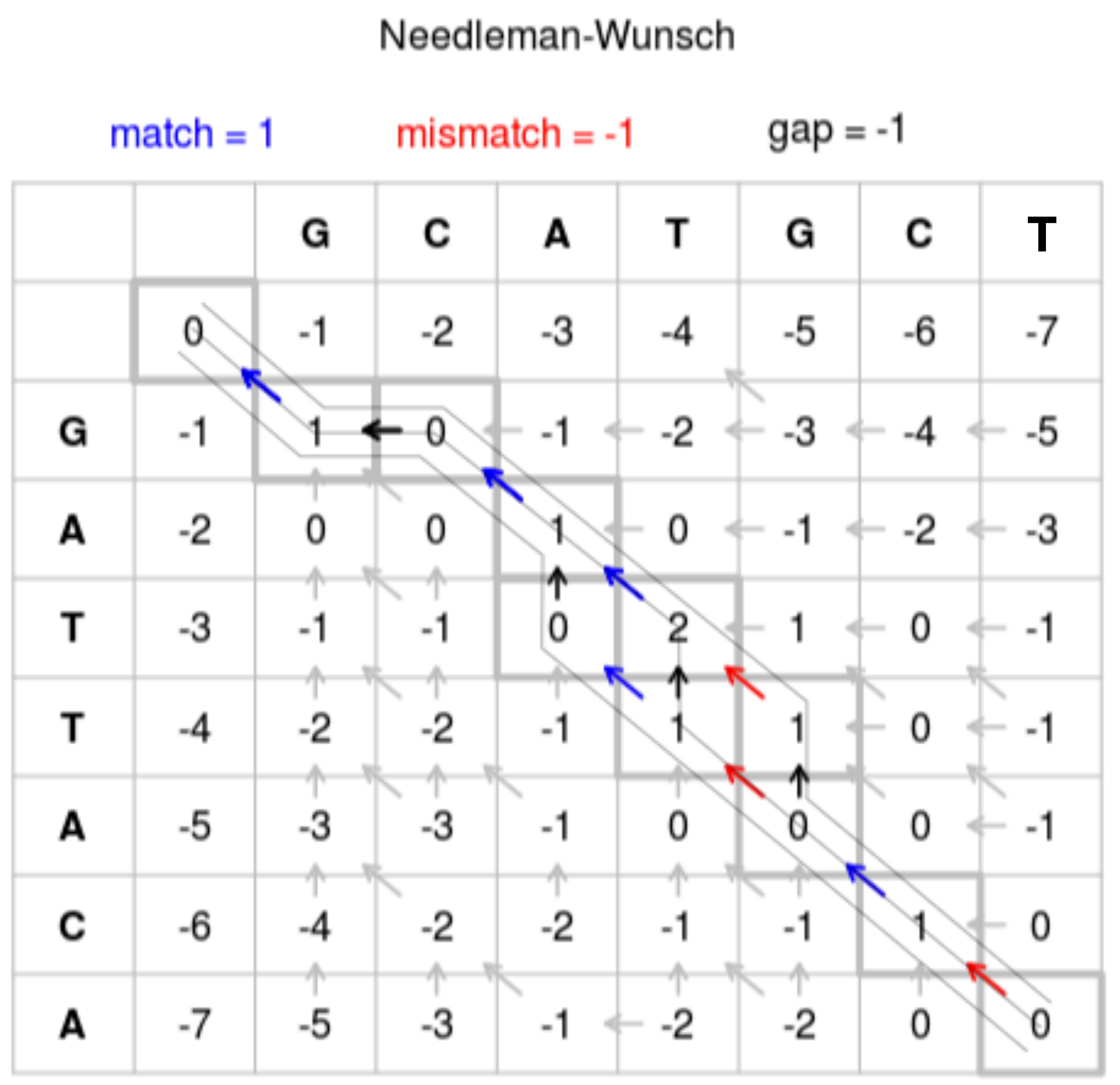
```
p = GATTACA          G-ATTACA
q = GCATGCT    ➡     GCA-TGCT
```

Alignment

Can be solved in

- O(|p| |q|) time using Needleman–Wunsch algorithm[1]
- Extended to local alignment by Smith and Waterman[2]

With local alignment, easily find the best location of a small string within another even if there are errors.



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

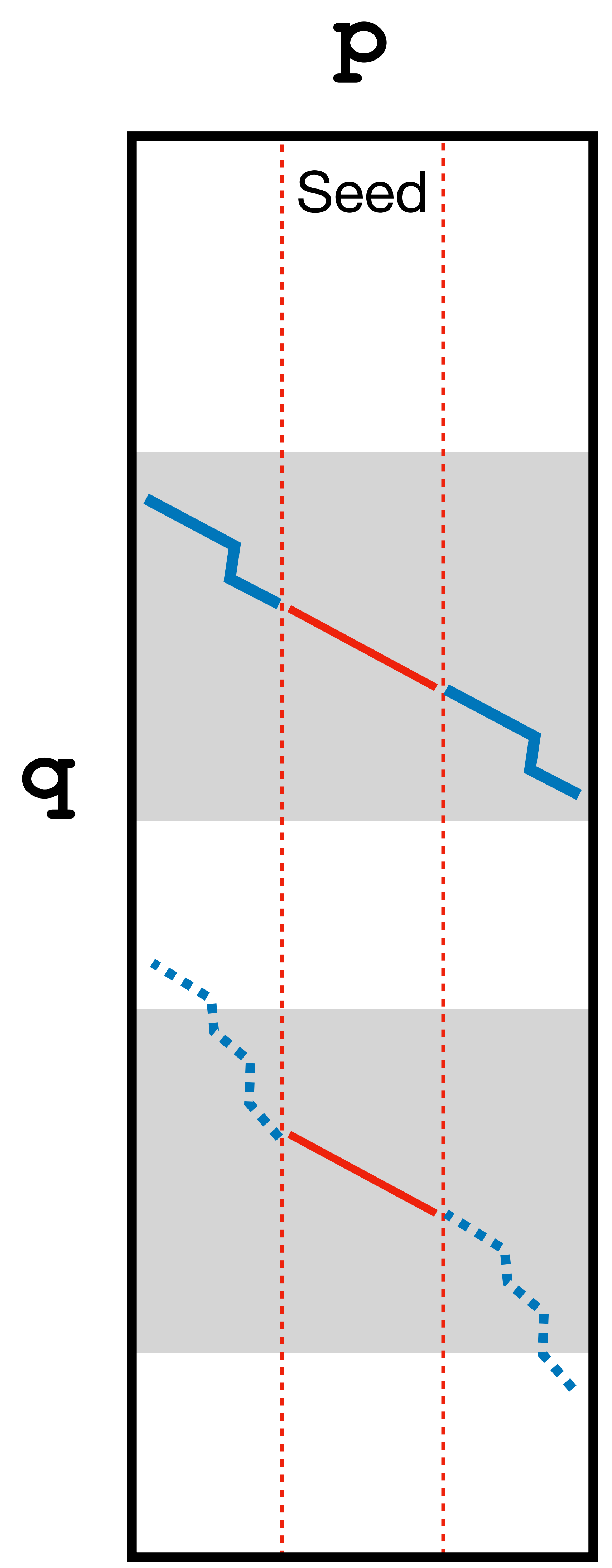
source:wikicommons

[1]Needleman, Saul B. & Wunsch, Christian D. (1970). "A general method applicable to the search for similarities in the amino acid sequence of two proteins". Journal of Molecular Biology. 48 (3): 443–53
[2]Smith, Temple F. & Waterman, Michael S. (1981). "Identification of Common Molecular Subsequences" . Journal of Molecular Biology. 147: 195–197

Seed and Extend

Given a pattern p and a text q, find p in q

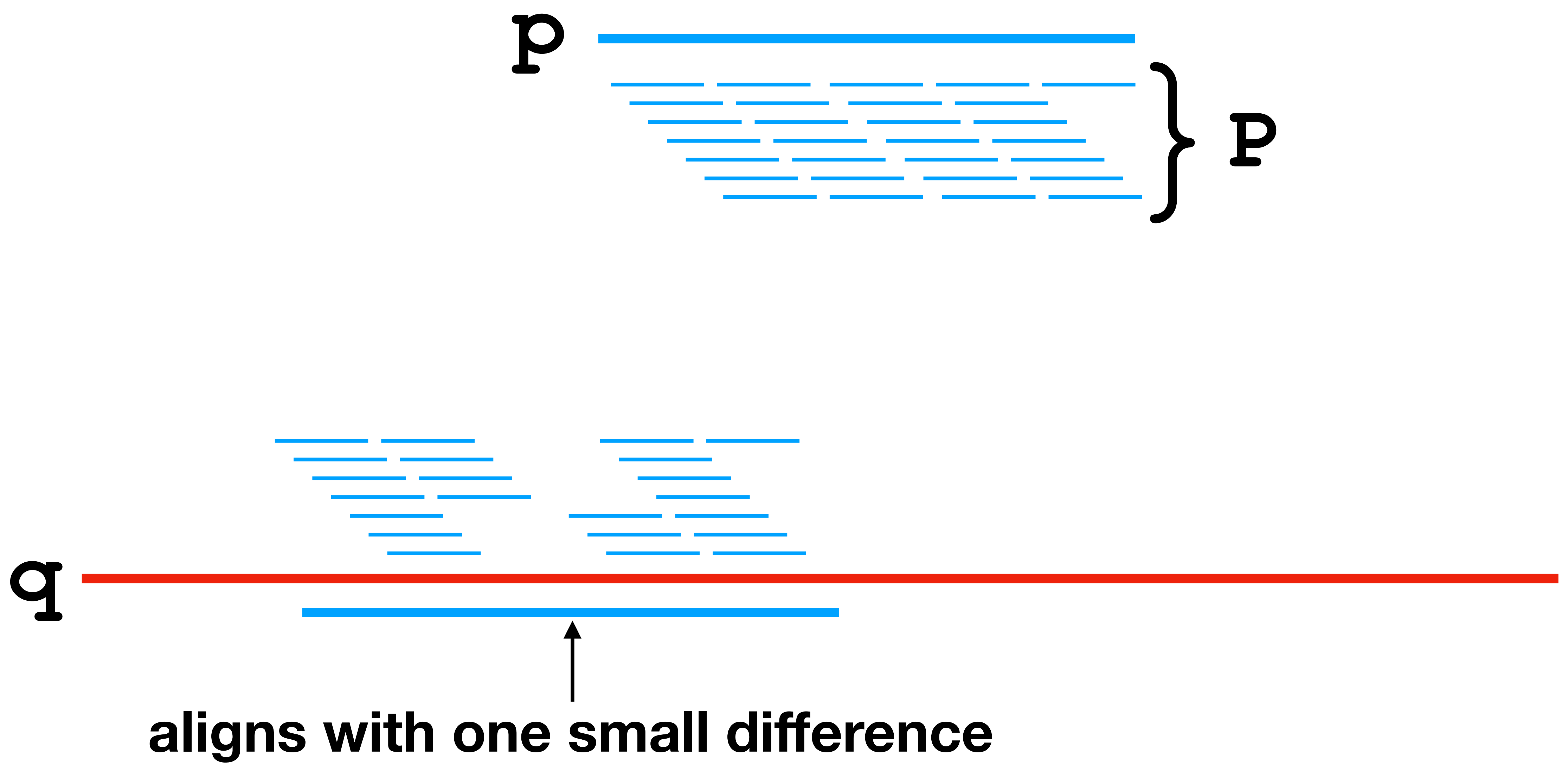
- select a substring p′ from p
- search for p′ in q using an exact search method
- only perform alignment on a small region around locations of p′
- Requiring multiple seeds can further reduce search locations and/or increase the number of errors allowed



Quasi-alignment

Given a pattern `p` and a text `q`, find `p` in `q`

- by first finding the set `P` of all overlapping subsequences of length `k`
- find the locations of `p'` in `P` in `q`
- if there is a region where a large percentage of `P` are found very close call that location a location of `p`.



k-mer Counting

q = xabxyabxyabxz
k = 3

For a given sequence `q` and value `k`

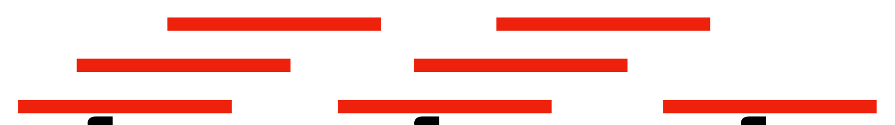
- determine the list of unique `k`-length strings in `q`, and
- count the frequency of each.
- Can be used to quickly compare two sequences.
- Problems arise in keeping the hash table (naively Σ^k entries)

	count
abx	3
bxy	2
bxz	1
xab	1
xya	2
yab	2

Minimizer schemes

Given a string `q`, and values `k` and `w`

- for each substring of `w` `k`-mers
- only select the minimum.
- This reduces the total number of `k`-mers that must be considered.
- Changing the ordering can impact the number of unique `k`-mers.



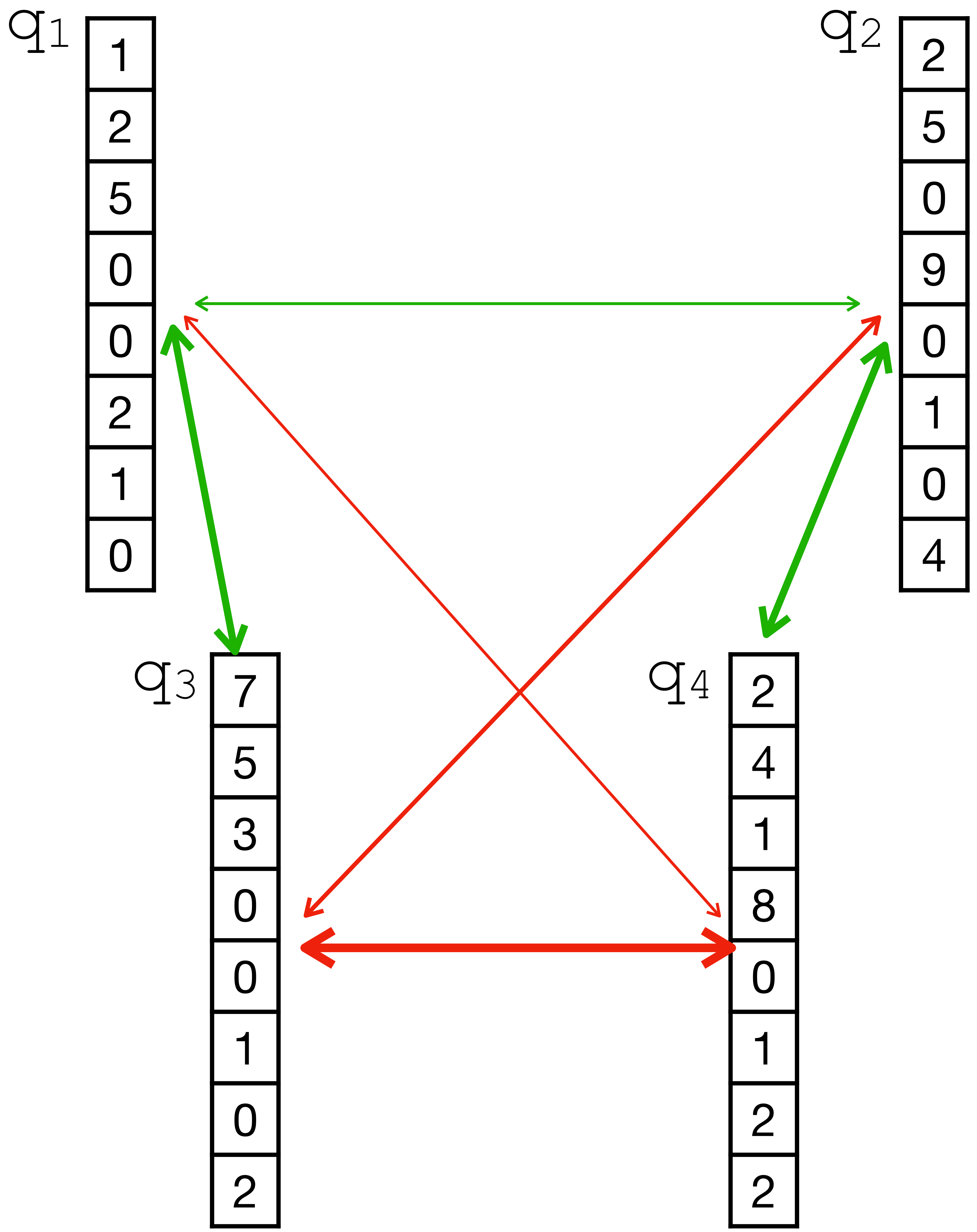
```
q = xabxyabxyabxz
k = 3
w = 2
```

	count	minimizer count
abx	3	3
bxy	2	2
bxz	1	0
xab	1	0
xya	2	2
yab	2	0

Metagenomics

Given a group of sequences `Q`

- group `q` in `Q` so similar sequences are from the same (possibly unknown) organism.
- Similarity can be measured using edit distance (alignment), `k`-mer counts, etc.



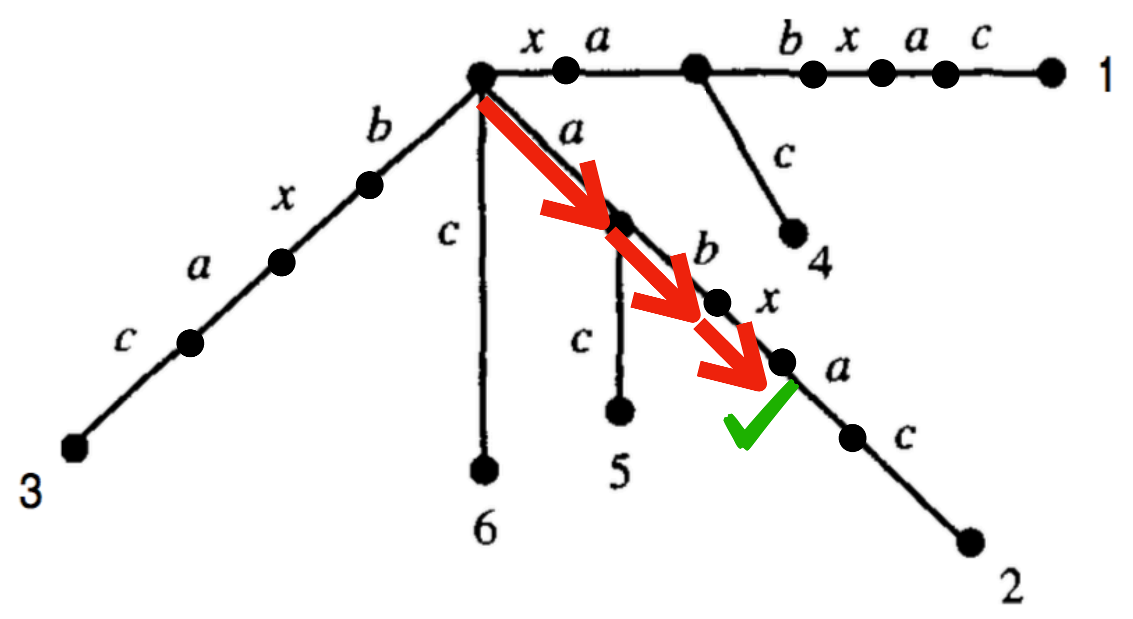
Naive sequence search

```
   0          1
   1234567890123
T: xabxyabxyabxz
P: abxyabxz
   *
    abxyabxz
    ^^^^^^^*
     abxyabxz
     •
      abxyabxz
      •
       abxyabxz
       *
        abxyabxz
        ^^^^^^^^
```

```
   0          1
   1234567890123
T: xabxyabxyabxz
P: abxyabxz
   •
    abxyabxz
    ^^^^^^^*
        abxyabxz
        ^^^^^^^^
```

```
   0         1
   1234567890123
T: xabxyabxyabxz
P: abxyabxz
   •
    abxyabxz
    ^^^^^^^*
        abxyabxz
            ^^^^^
```

Suffix trees



```
P = abx
Q = xabxac
    123456
```

Suffix arrays

	pos	lcp	
0	12		$
1	11	0	i$
2	8	1	ippi$
3	5	1	issippi$
4	2	4	ississippi
5	1	0	mississip
6	10	0	pi$
7	9	1	ppi$
8	7	0	sippi$
9	4	2	sissippi$
10	6	1	ssippi$
11	3	3	ssissippi$

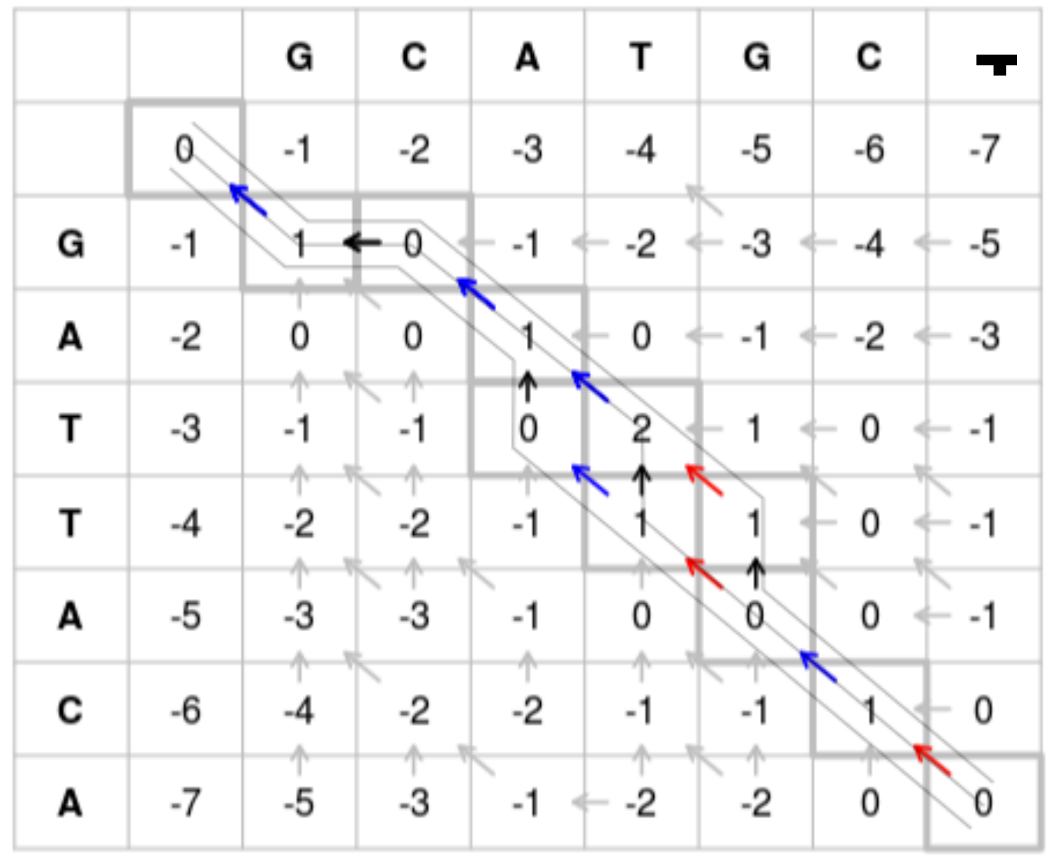
BWT

```
$mississippi
i$mississipp
ippi$mississ
issippi$miss
ississippi$m
mississippi$
pi$mississip
ppi$mississi
sippi$missis
sissippi$mis
ssippi$missi
ssissippi$mi
```

Alignment

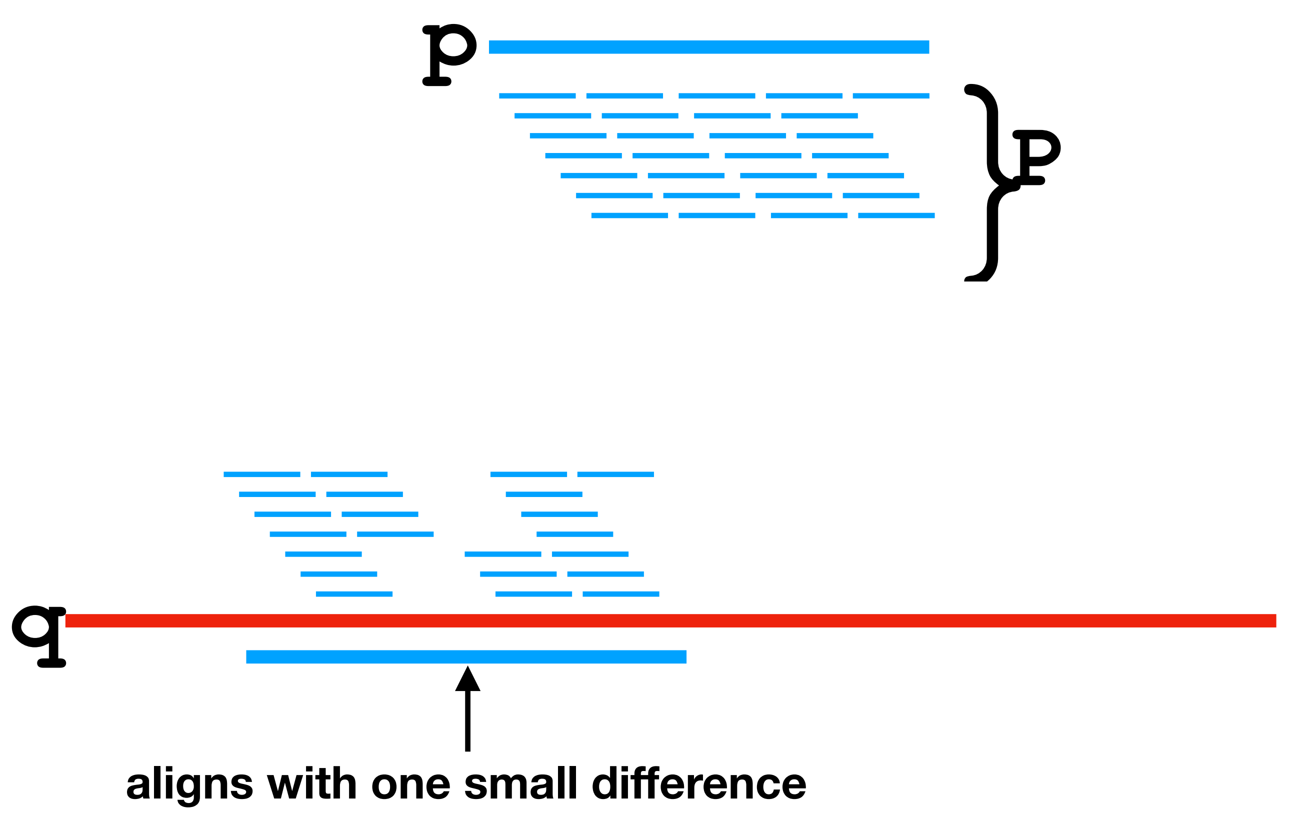
Needleman-Wunsch

match = 1 mismatch = -1 gap = -1



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

Quasi-alignment



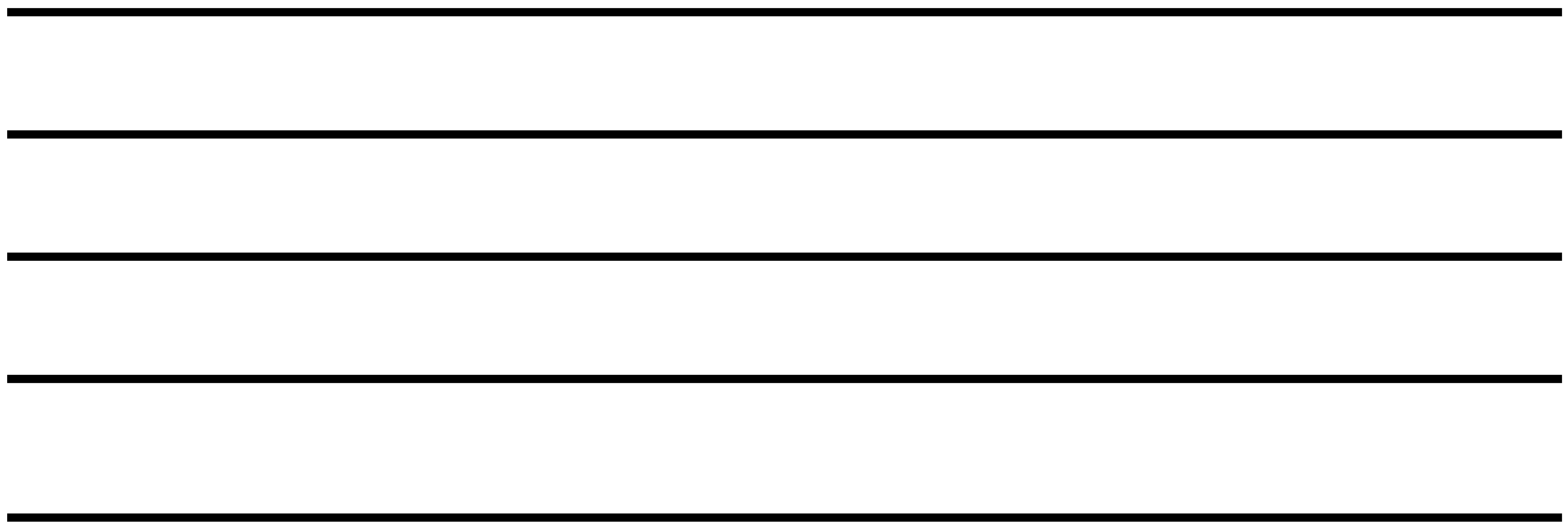
k-mer counting

```
q = xabxyabxyabxz
k = 3
```

abx	3
bxy	2
bxz	1
xab	1
xya	2
yab	2

Minimizers

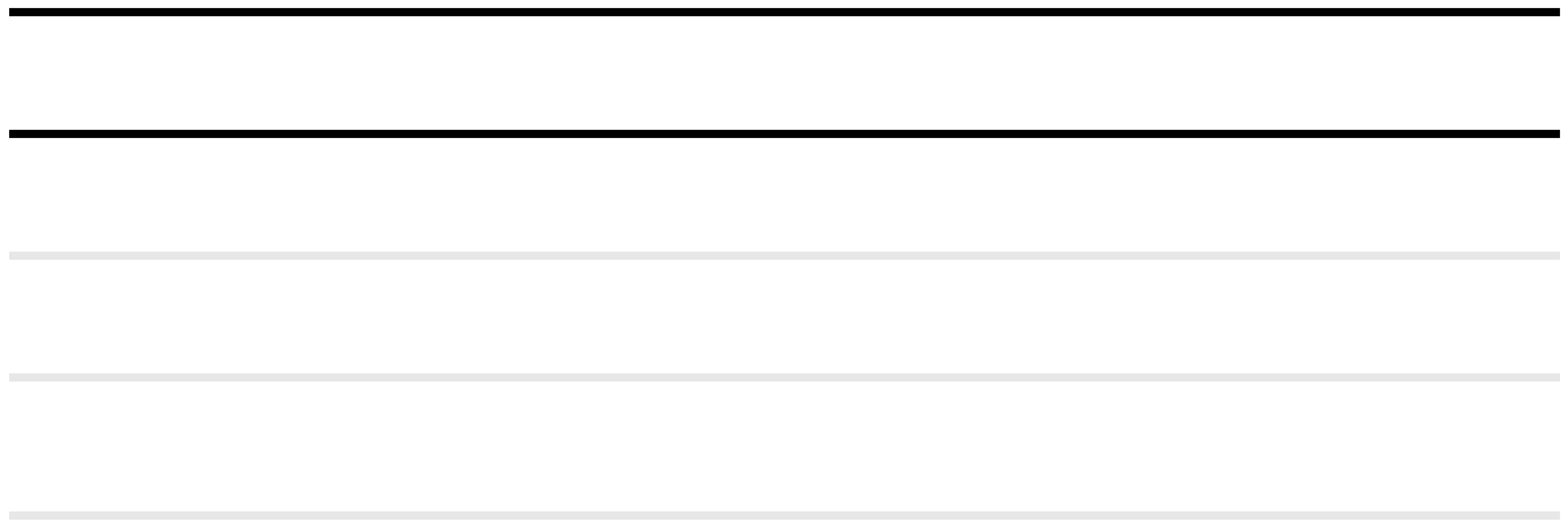
Given a set of sequences, compute all of the pairwise overlaps



Slide adapted from those of Guillaume Marçais

Minimizers

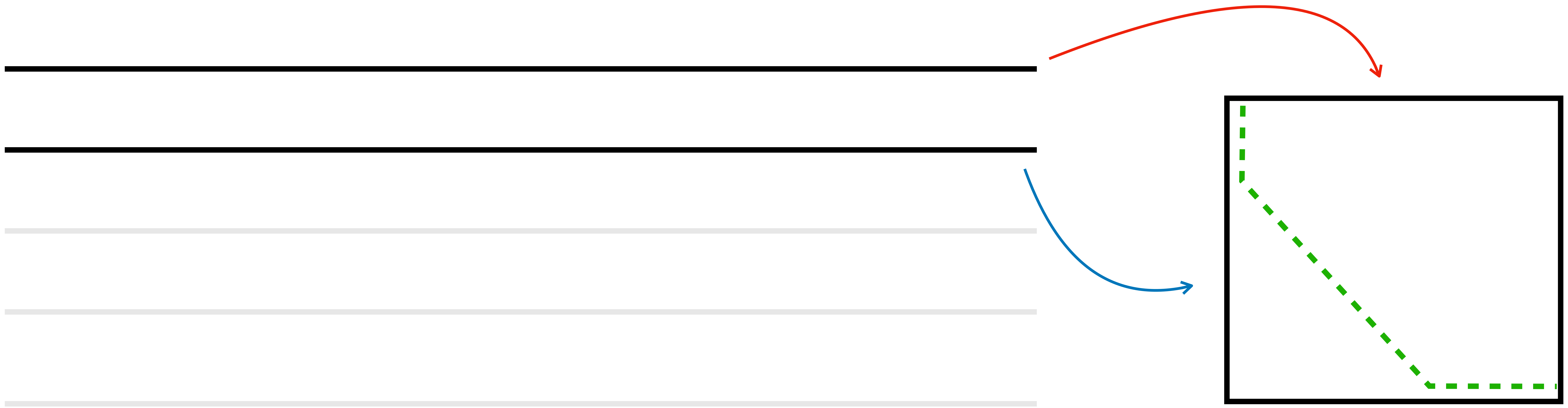
Given a set of sequences, compute all of the pairwise overlaps



Slide adapted from those of Guillaume Marçais

Minimizers

Given a set of sequences, compute all of the pairwise overlaps

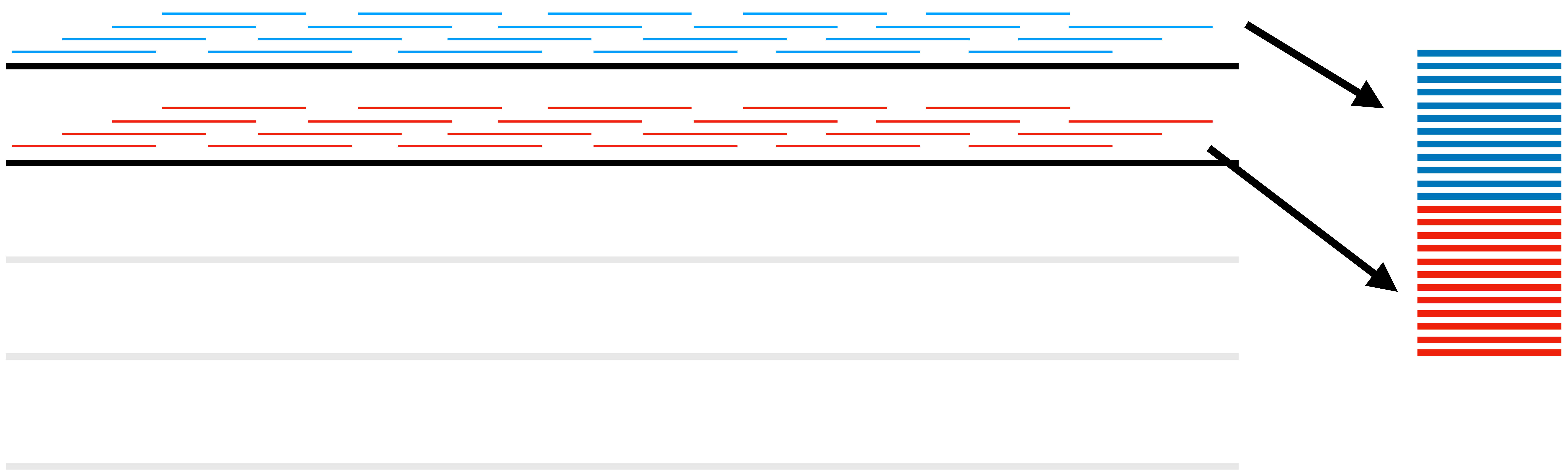


O(mn) time for each pair!

Slide adapted from those of Guillaume Marçais

Minimizers

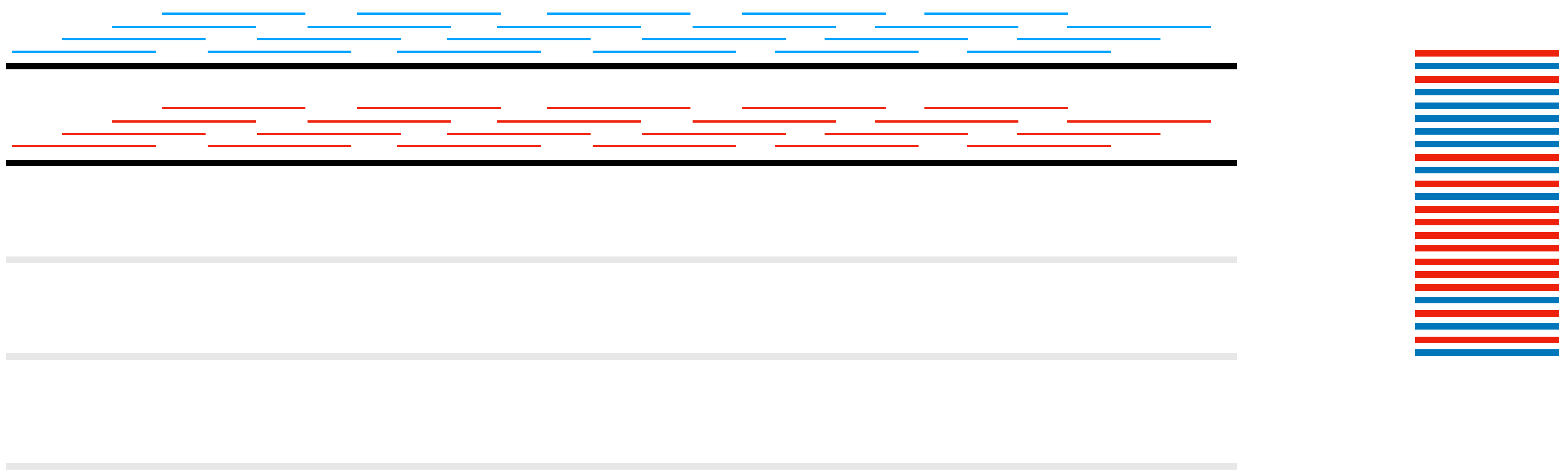
Given a set of sequences, compute all of the pairwise overlaps



Slide adapted from those of Guillaume Marçais

Minimizers

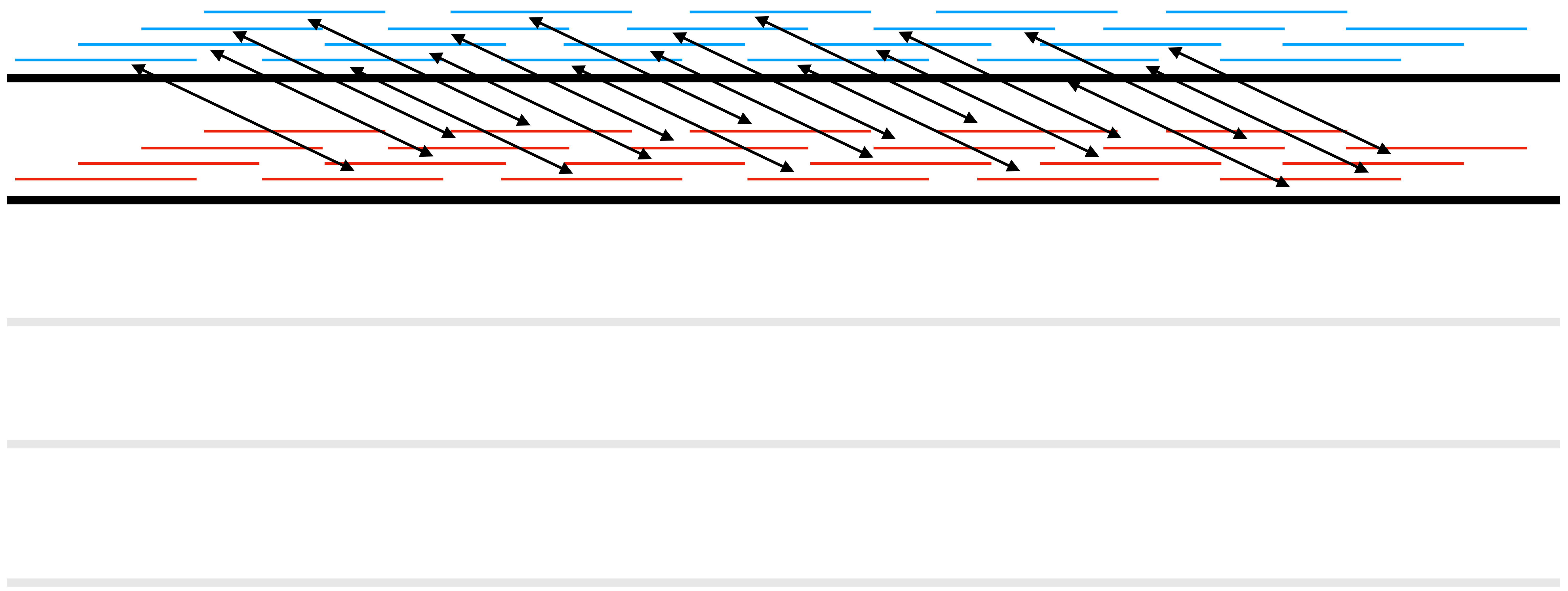
Given a set of sequences, compute all of the pairwise overlaps

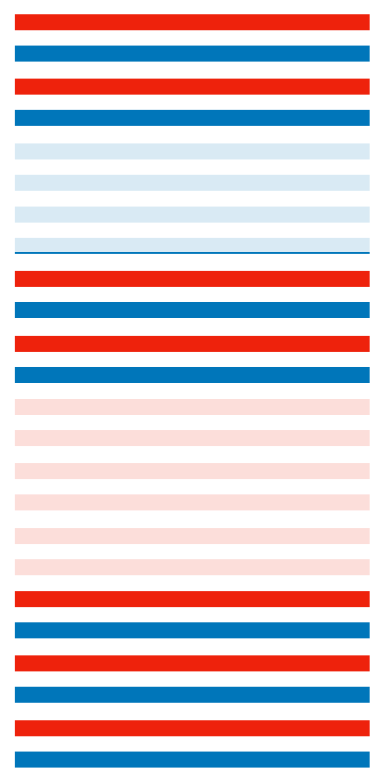


Slide adapted from those of Guillaume Marçais

Minimizers

Given a set of sequences, compute all of the pairwise overlaps





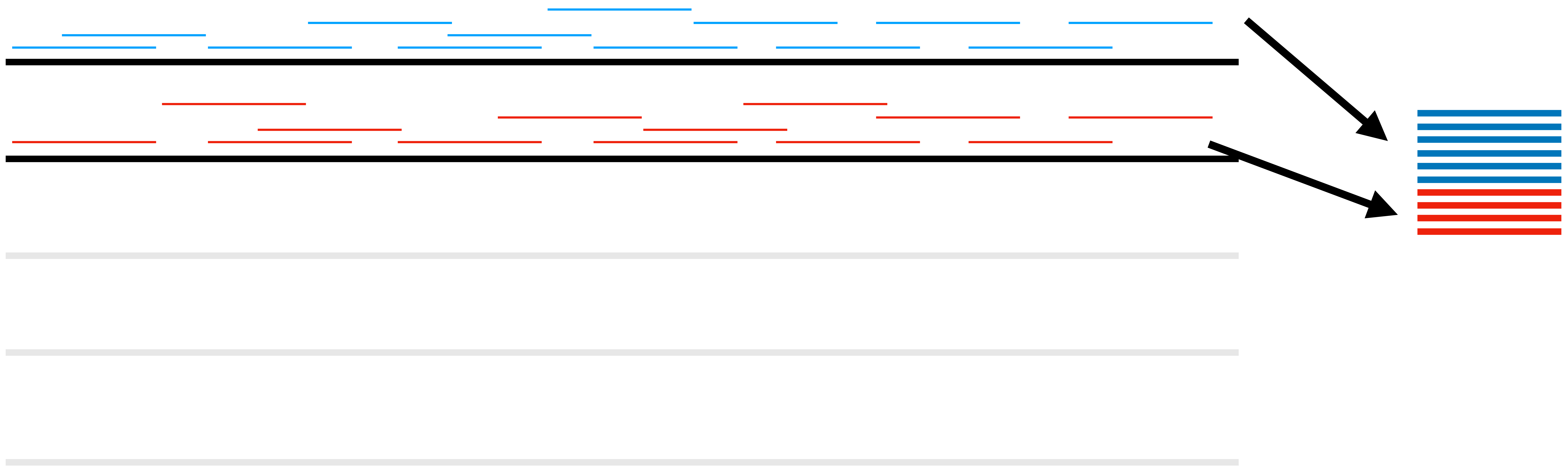
O(n) k-mers per sequence!

When sorted exact matches appear together and can be mapped easily

Slide adapted from those of Guillaume Marçais

Minimizers

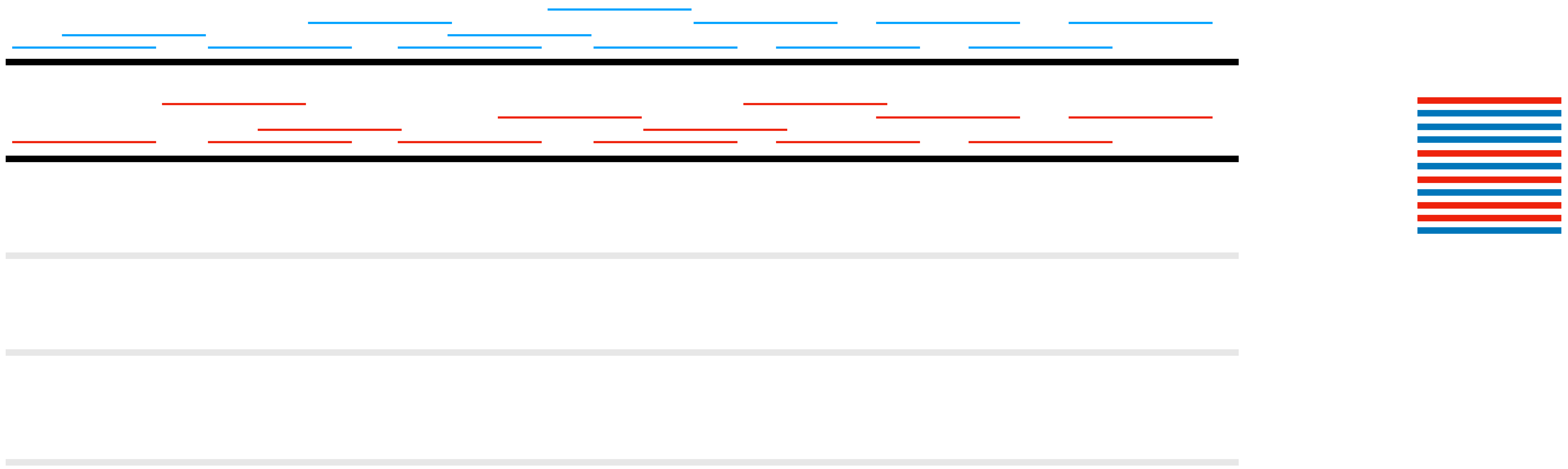
Given a set of sequences, compute all of the pairwise overlaps



Slide adapted from those of Guillaume Marçais

Minimizers

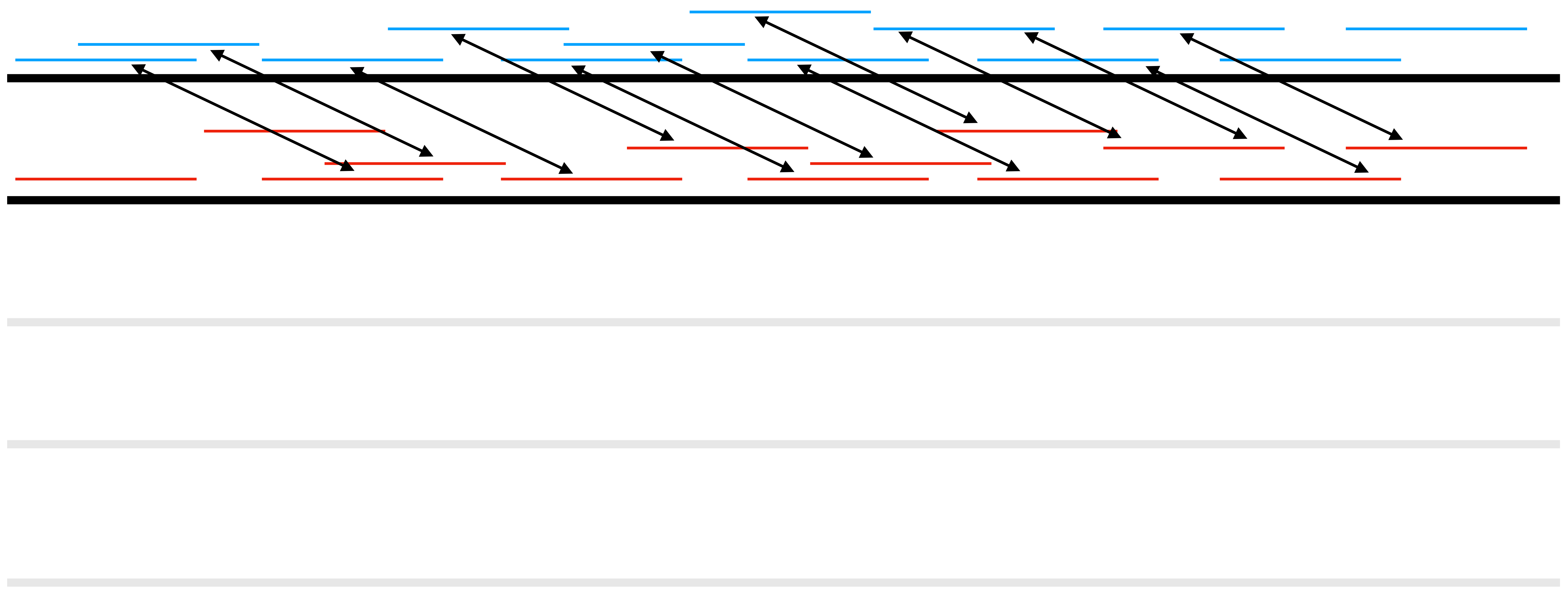
Given a set of sequences, compute all of the pairwise overlaps

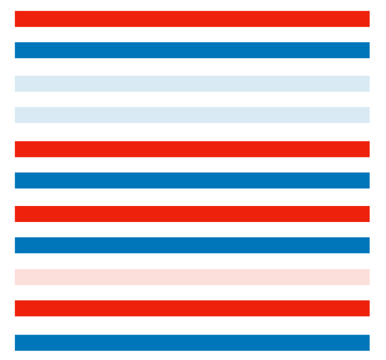


Slide adapted from those of Guillaume Marçais

Minimizers

Given a set of sequences, compute all of the pairwise overlaps





Slide adapted from those of Guillaume Marçais

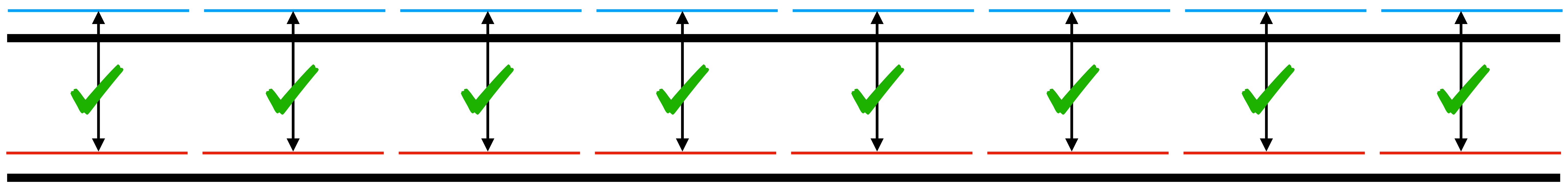
Minimizers

How should you choose the k-mers from a string

- Small number of k-mers
- Cover the whole string
- Long overlaps should have large numbers of matched k-mers

Store exactly every wth mer

Storage and comparisons reduced by factor of 1/w



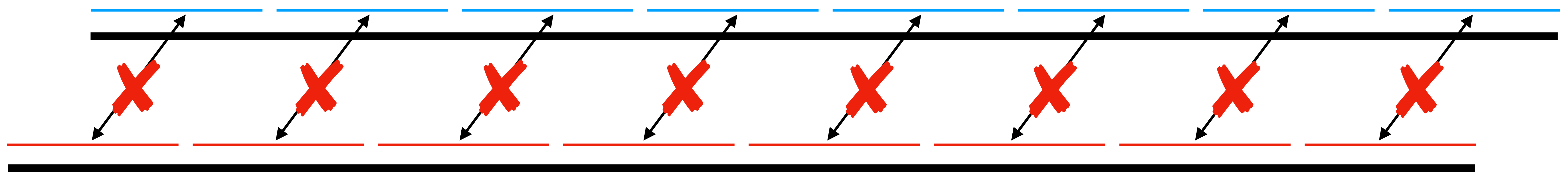
Slide adapted from those of Guillaume Marçais

Minimizers

How should you choose the k-mers from a string

- Small number of k-mers
- Cover the whole string
- Long overlaps should have large numbers of matched k-mers

Store exactly every wth mer



Storage and comparisons reduced by factor of 1/w

Large overlap, no matched k-mers

Slide adapted from those of Guillaume Marçais

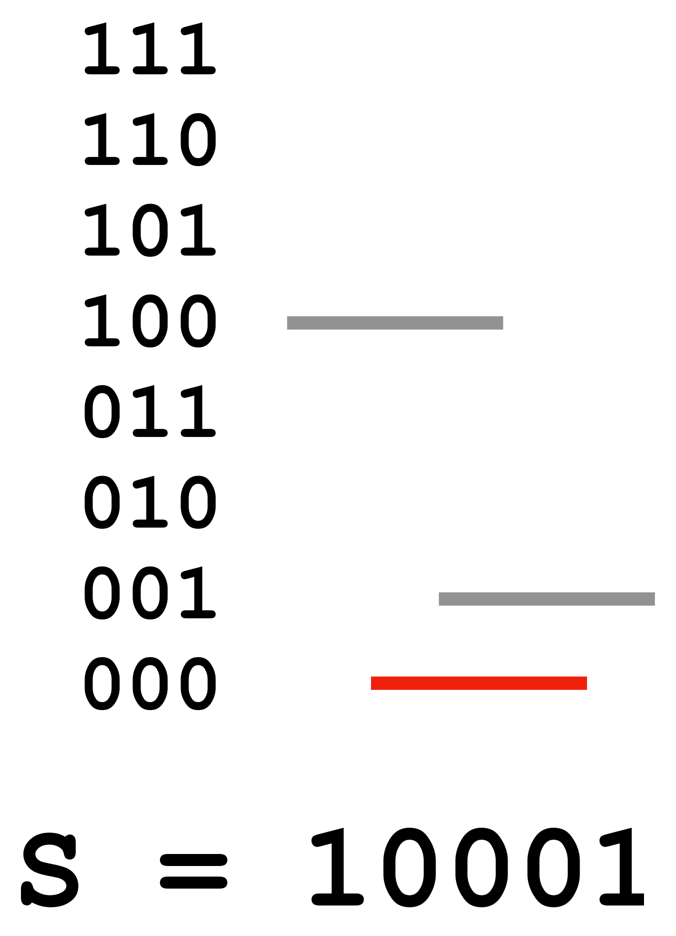
Minimizers

How should you choose the `k`-mers from a string

- Small number of `k`-mers
- Cover the whole string
- Long overlaps should have large numbers of matched `k`-mers

For each window of `w` `k`-mers

- choose the smallest `k`-mer as the fingerprint.



k = 3
w = 3

Slide adapted from those of Guillaume Marçais

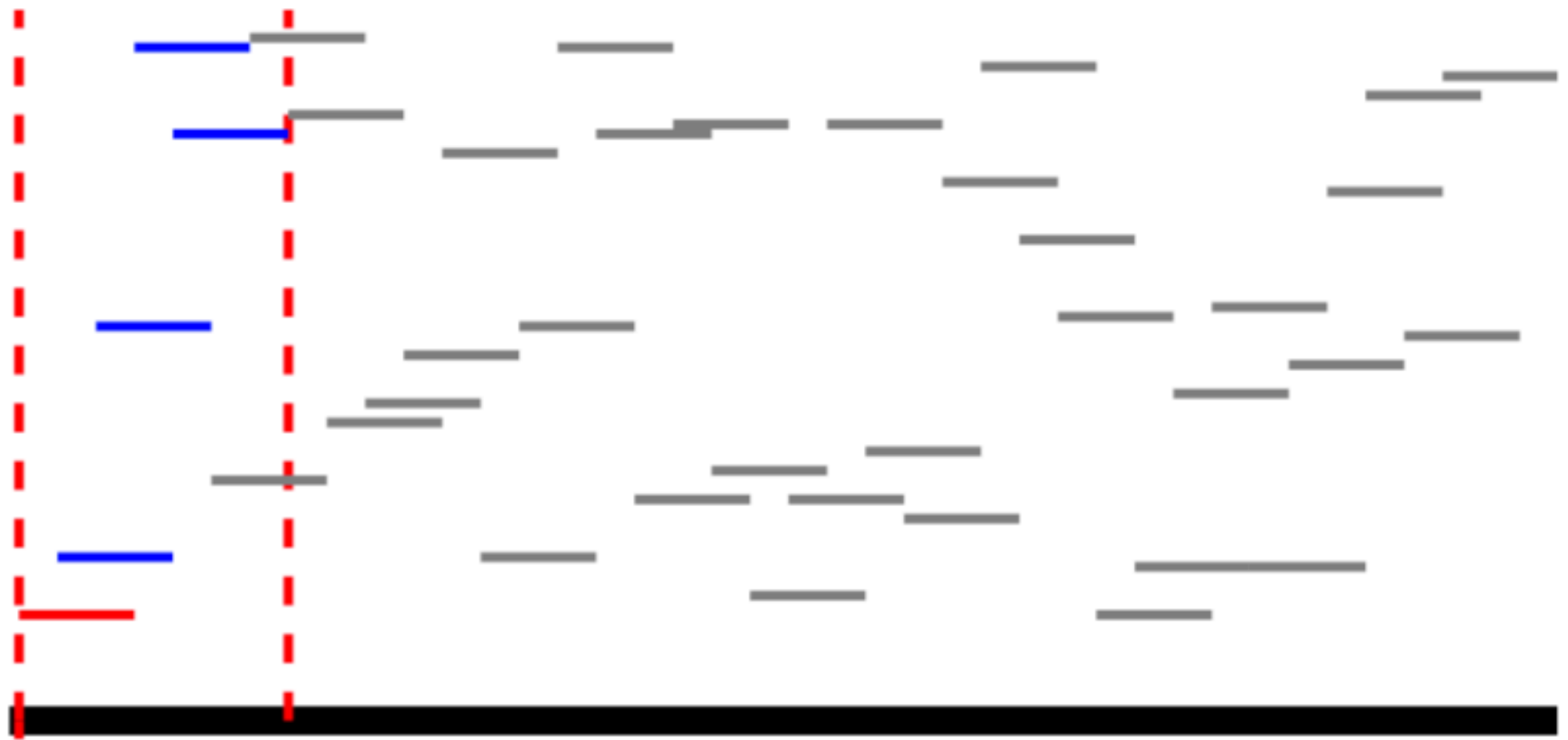
Minimizers

How should you choose the `k`-mers from a string

- Small number of `k`-mers
- Cover the whole string
- Long overlaps should have large numbers of matched `k`-mers

For each window of `w` `k`-mers

- choose the smallest `k`-mer as the fingerprint.



Slide adapted from those of Guillaume Marçais

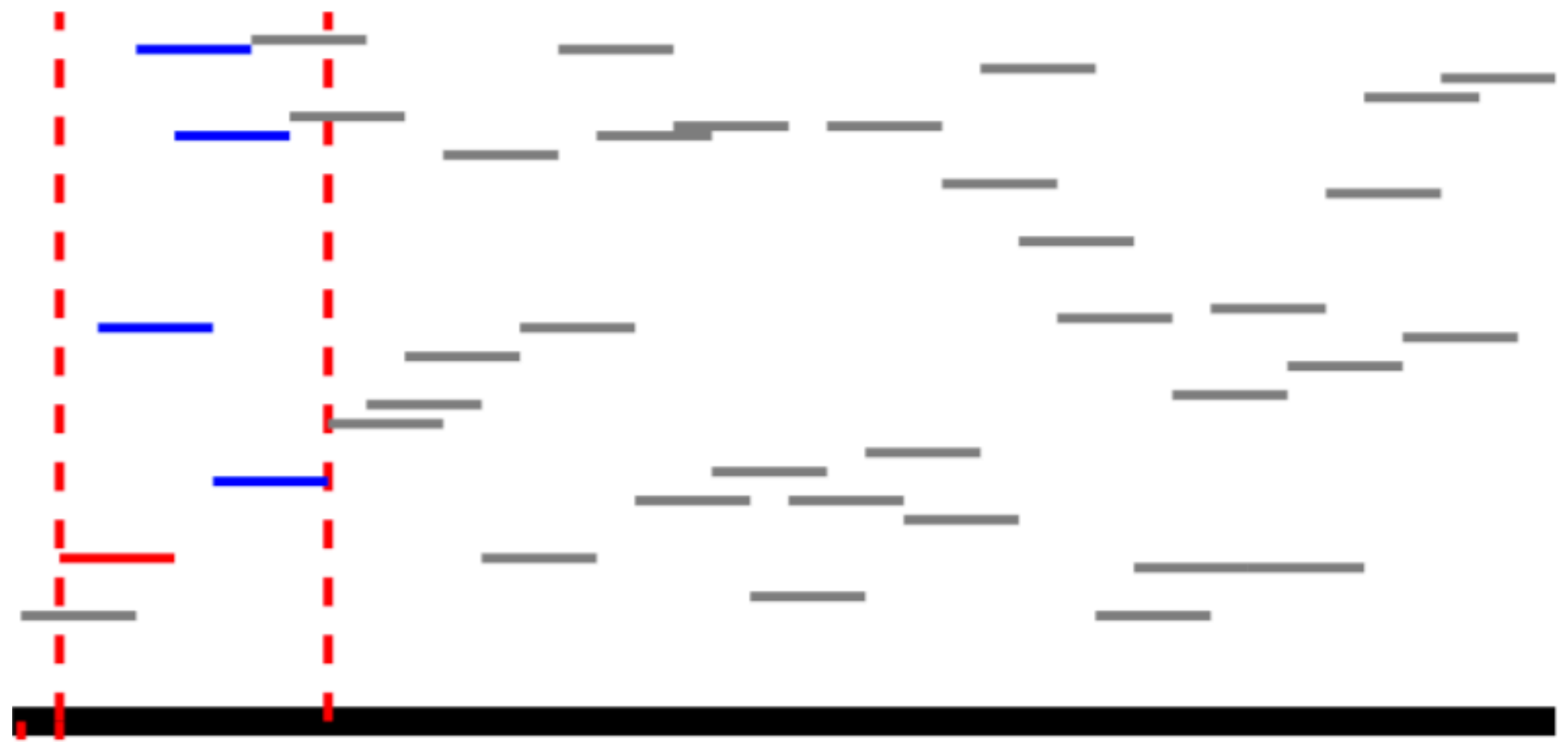
Minimizers

How should you choose the k-mers from a string

- Small number of k-mers
- Cover the whole string
- Long overlaps should have large numbers of matched k-mers

For each window of w k-mers

- choose the smallest k-mer as the fingerprint.



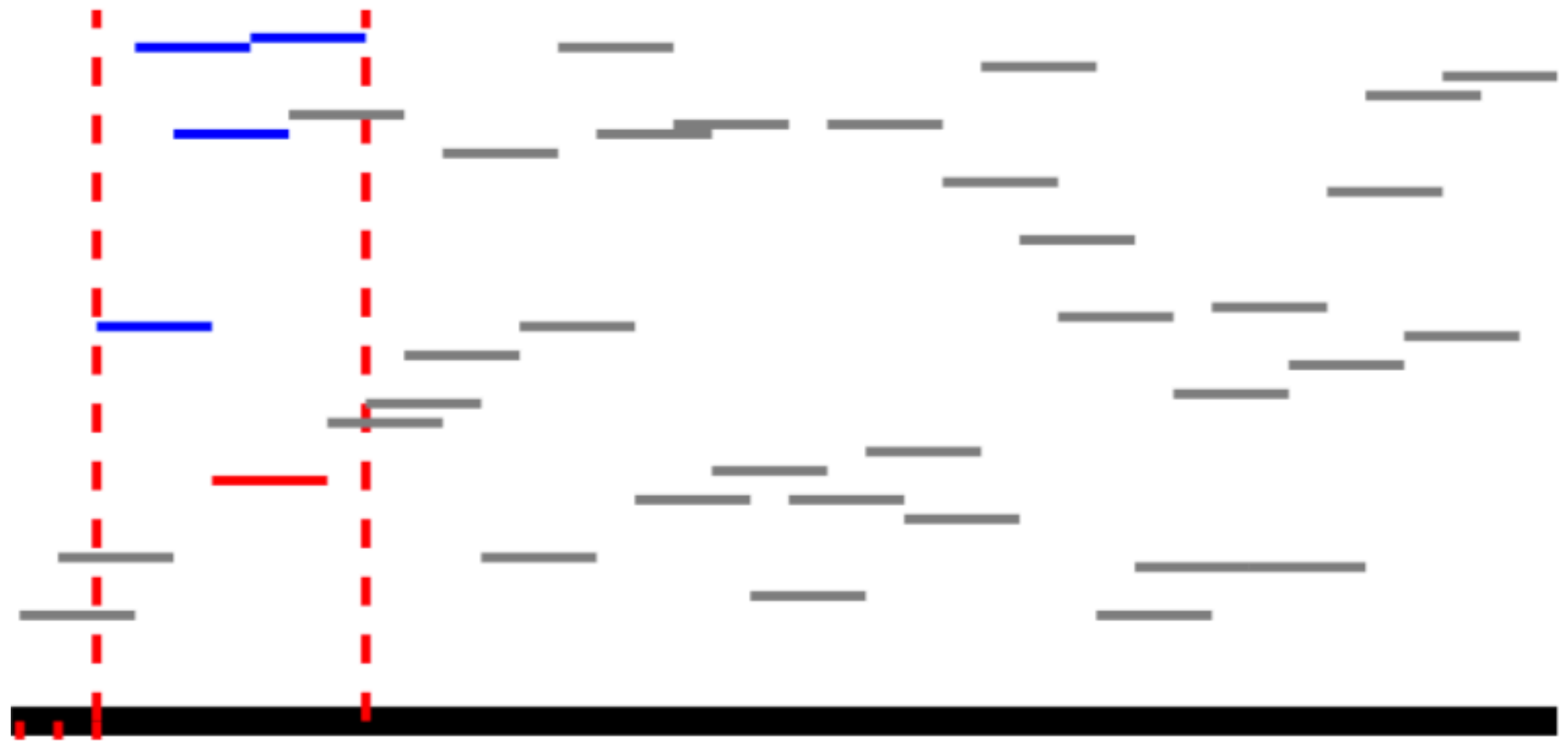
Minimizers

How should you choose the `k`-mers from a string

- Small number of `k`-mers
- Cover the whole string
- Long overlaps should have large numbers of matched `k`-mers

For each window of `w` `k`-mers

- choose the smallest `k`-mer as the fingerprint.



Slide adapted from those of Guillaume Marçais

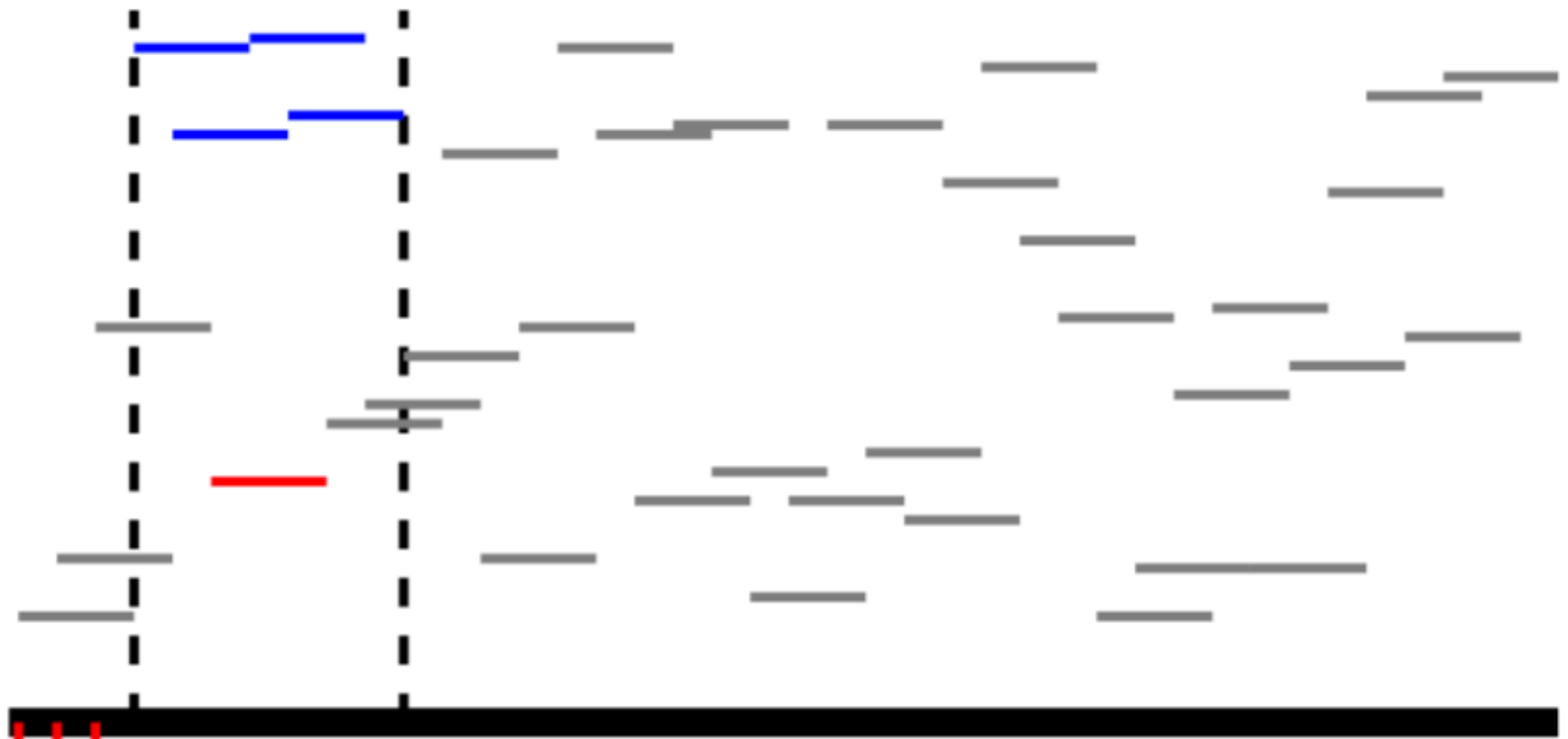
Minimizers

How should you choose the `k`-mers from a string

- Small number of `k`-mers
- Cover the whole string
- Long overlaps should have large numbers of matched `k`-mers

For each window of `w` `k`-mers

- choose the smallest `k`-mer as the fingerprint.



Slide adapted from those of Guillaume Marçais

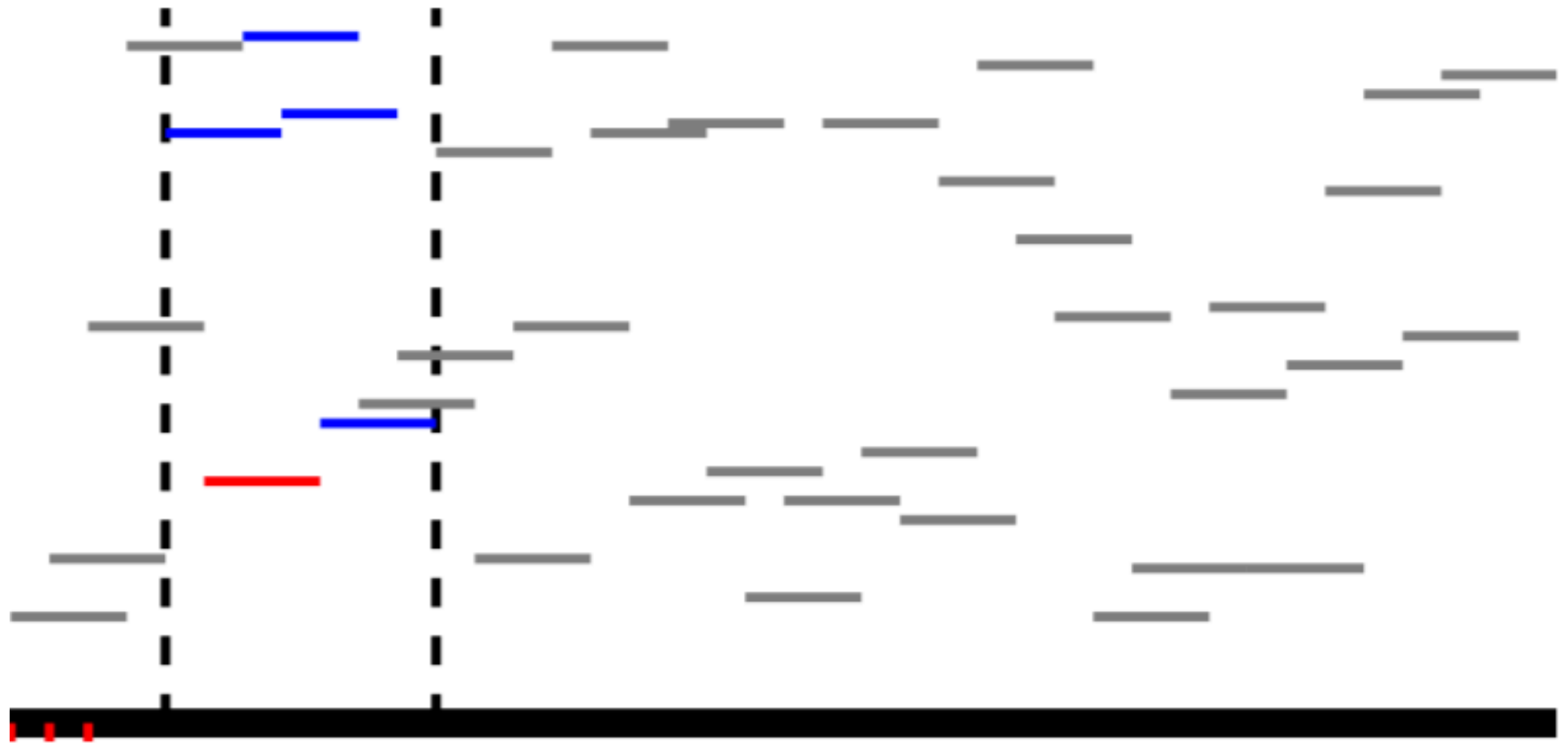
Minimizers

How should you choose the `k`-mers from a string

- Small number of `k`-mers
- Cover the whole string
- Long overlaps should have large numbers of matched `k`-mers

For each window of `w` `k`-mers

- choose the smallest `k`-mer as the fingerprint.



Slide adapted from those of Guillaume Marçais

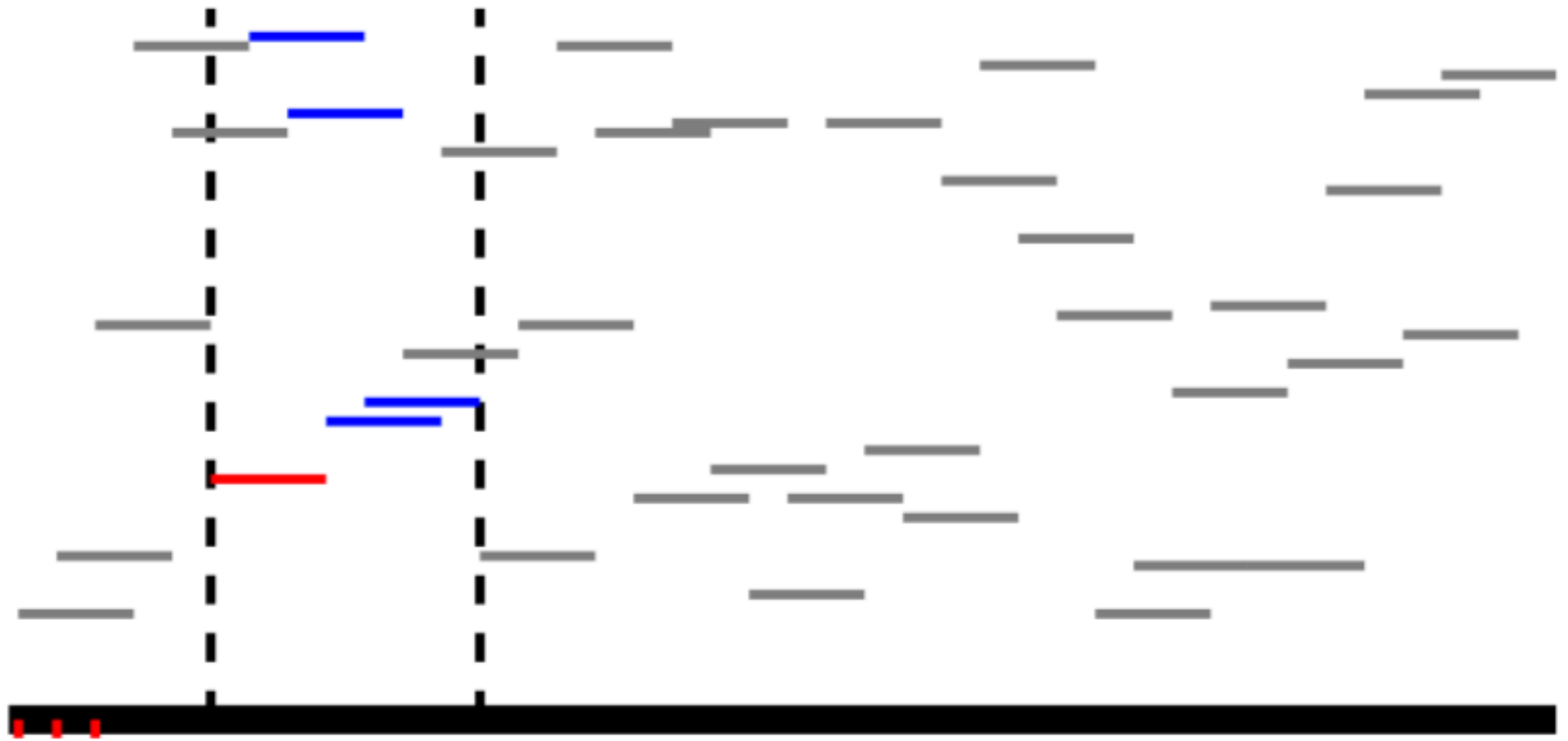
Minimizers

How should you choose the `k`-mers from a string

- Small number of `k`-mers
- Cover the whole string
- Long overlaps should have large numbers of matched `k`-mers

For each window of `w` `k`-mers

- choose the smallest `k`-mer as the fingerprint.



Slide adapted from those of Guillaume Marçais

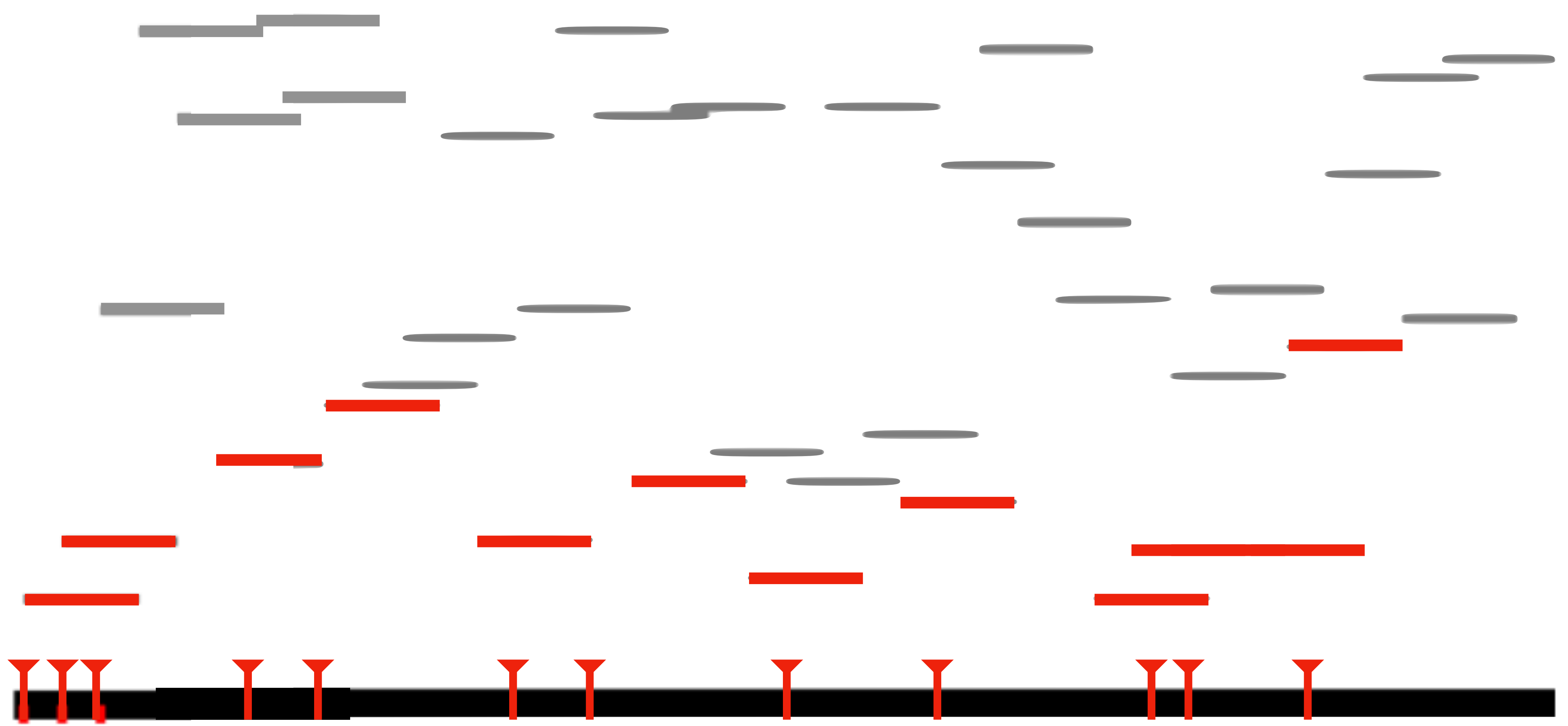
Minimizers

How should you choose the `k`-mers from a string

- Small number of `k`-mers
- Cover the whole string
- Long overlaps should have large numbers of matched `k`-mers

For each window of `w` `k`-mers

- choose the smallest `k`-mer as the fingerprint.



Slide adapted from those of Guillaume Marçais

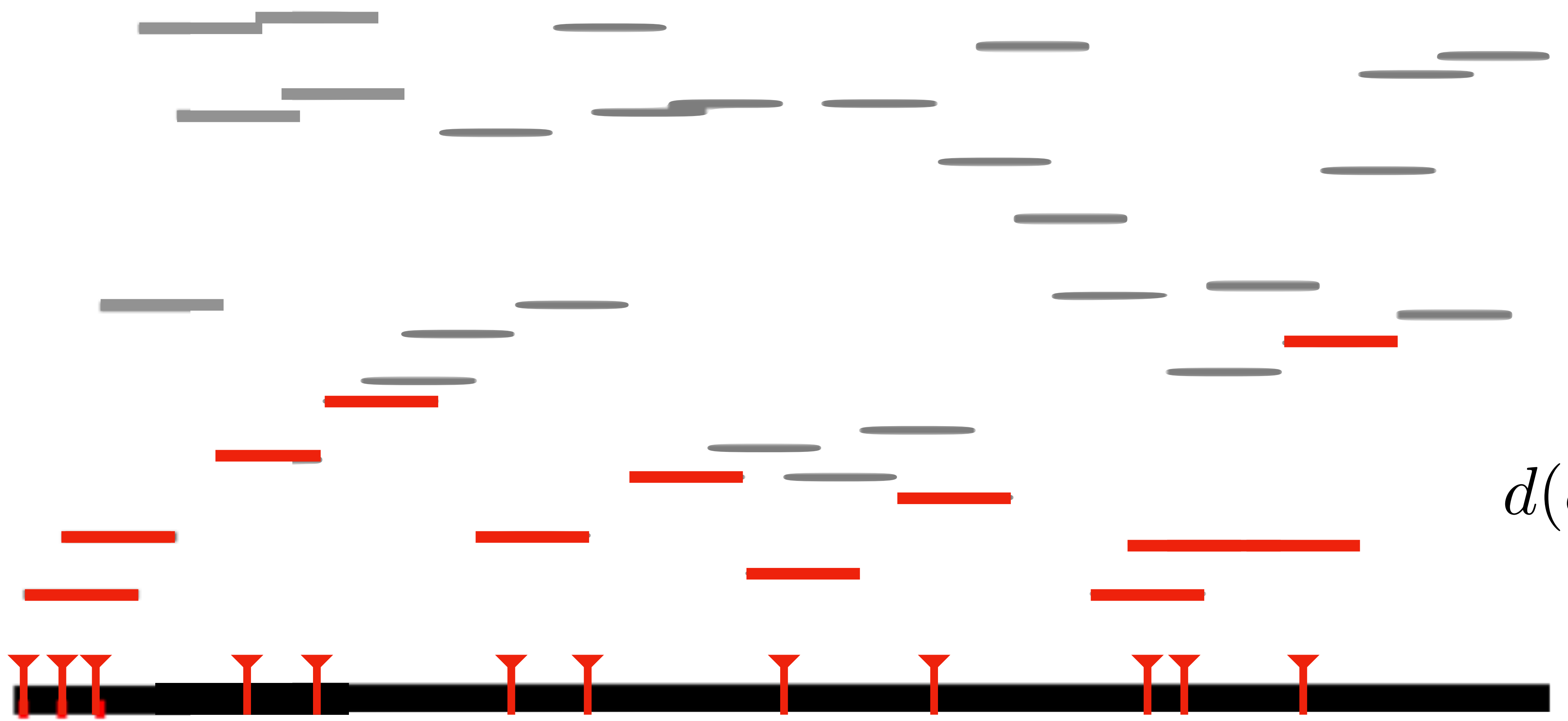
Minimizers

How should you choose the `k`-mers from a string

- Small number of `k`-mers
- Cover the whole string
- Long overlaps should have large numbers of matched `k`-mers

For each window of `w` `k`-mers

- choose the smallest `k`-mer as the fingerprint.

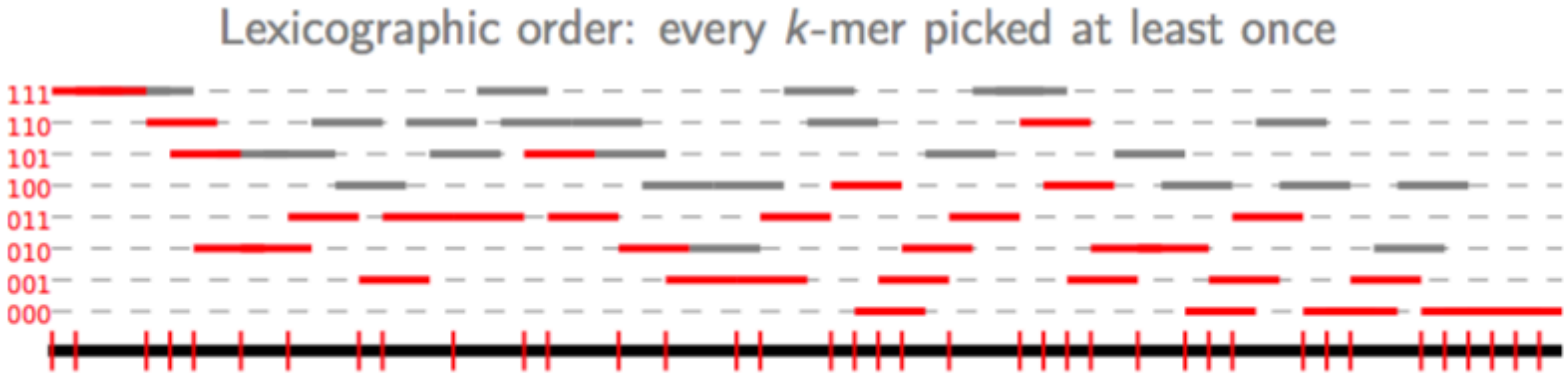


Density of an order o

$$d(o, S, k) = \frac{\#\text{ of selected positions}}{|S| - k + 1}$$

Slide adapted from those of Guillaume Marçais

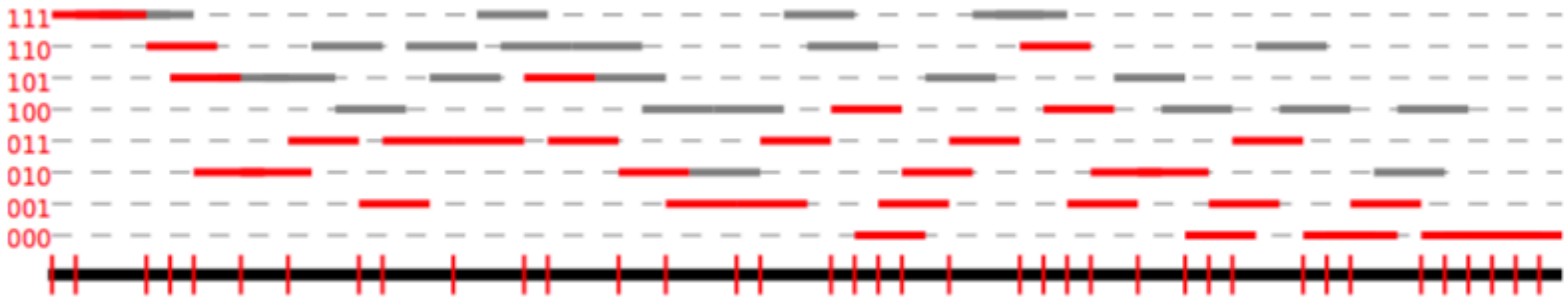
Minimizers



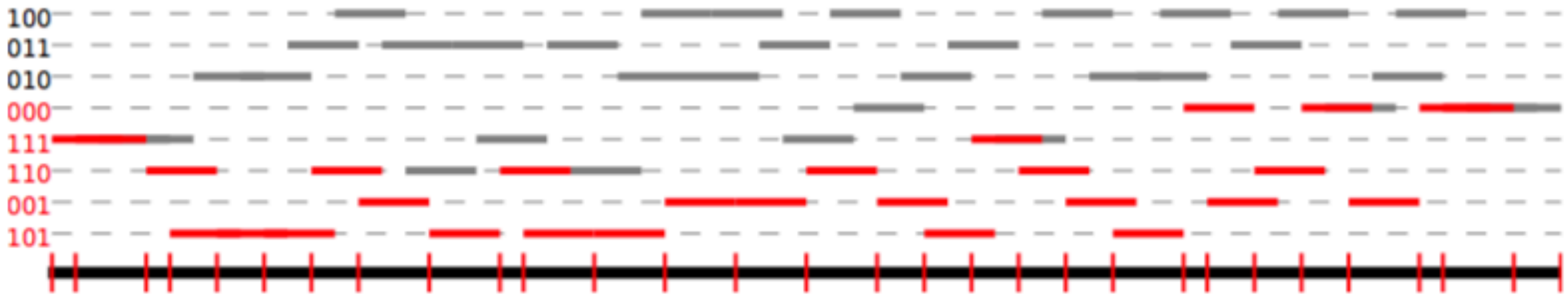
Slide adapted from those of Guillaume Marçais

Minimizers

Lexicographic order: every *k*-mer picked at least once



Optimized order: top 3 *k*-mers never picked



Slide adapted from those of Guillaume Marçais

Universal hitting set

A universal set $\mathtt{U_{k,w}}$ is

- a set of `k`-mers such that
- all string of length `w+k-1` contains one `k`-mers from the set
- that minimizes the size of the set.
- Found using the de Brujin graph of `k`-mers by first selecting nodes that intersect all cycles (decycling)
- then additional nodes to intersect long paths.

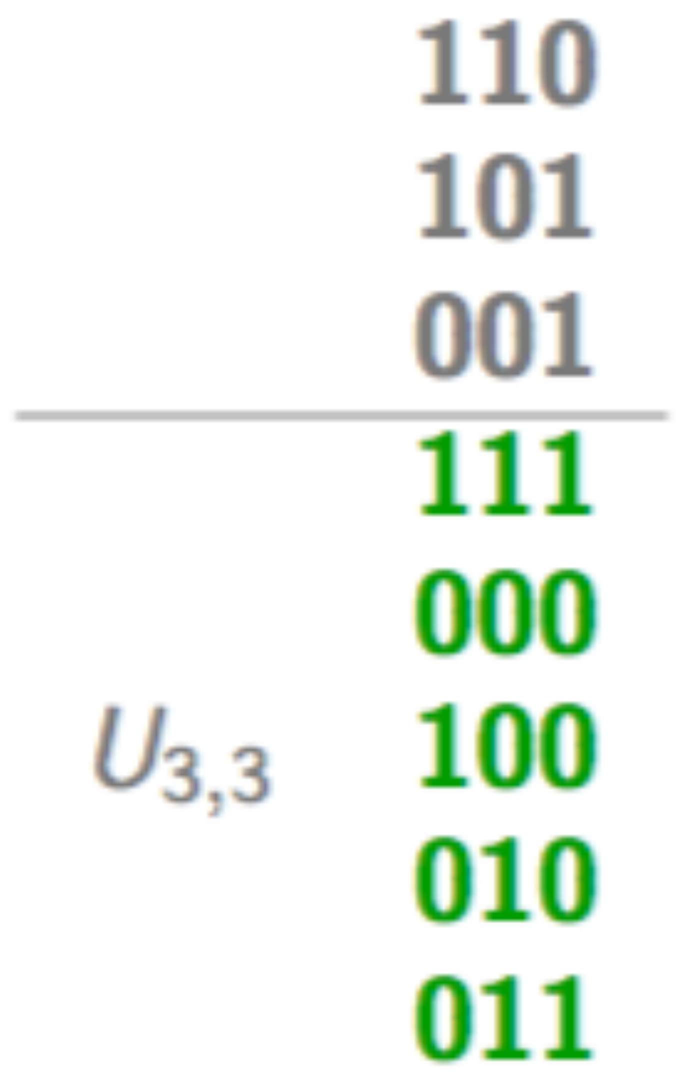
Universal hitting set

A universal set $U_{k,w}$ is

- a set of k-mers such that
- all string of length $w+k-1$ contains one k-mers from the set
- that minimizes the size of the set.
- Found using the de Brujin graph of k-mers by first selecting nodes that intersect all cycles (decycling)
- then additional nodes to intersect long paths.

Universal set ordering

- Given a universal set $U_{k,w}$
- rank all k-mers from the set lower than any other.

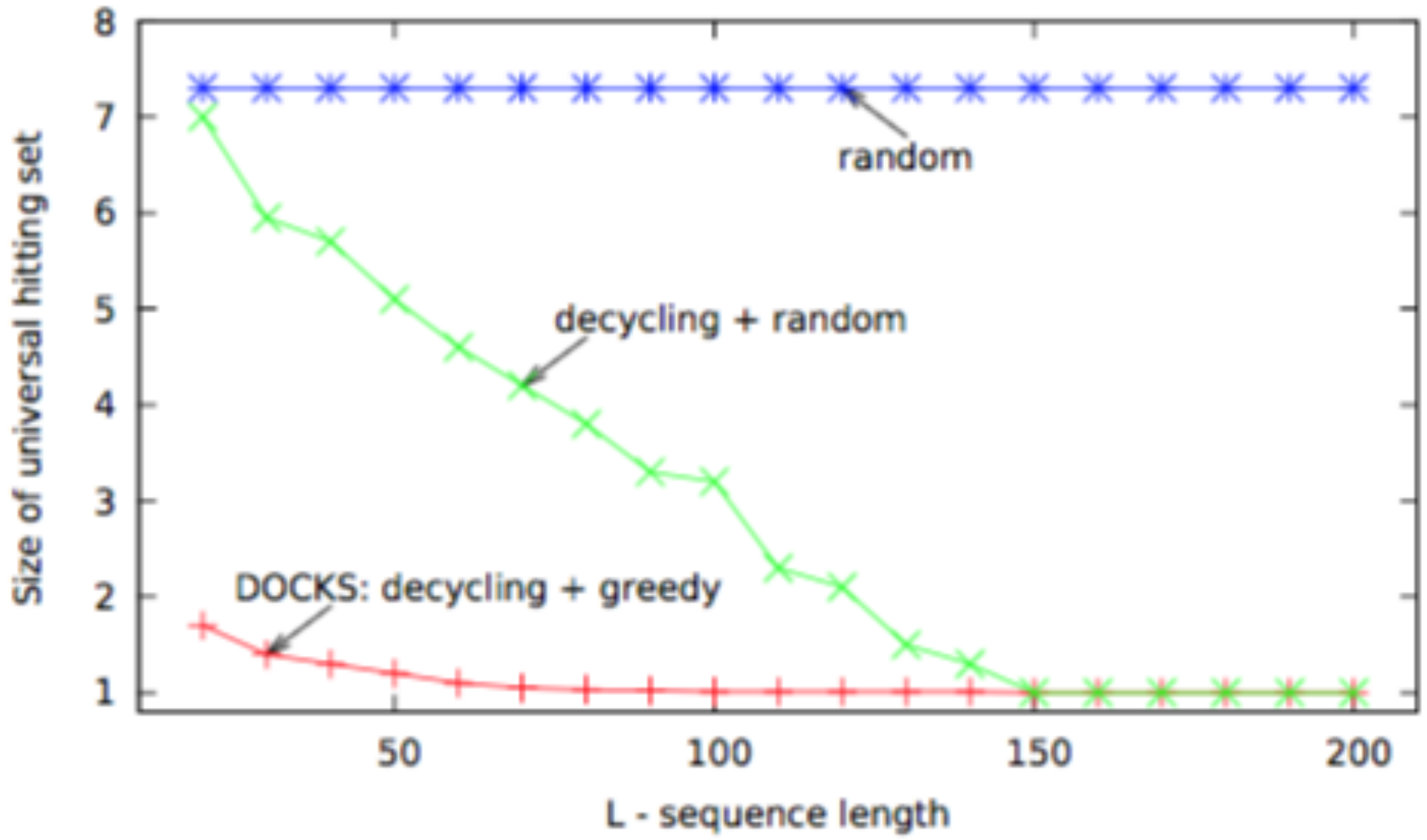


110
101
001

$U_{3,3}$
111
000
100
010
011

Universal hitting set

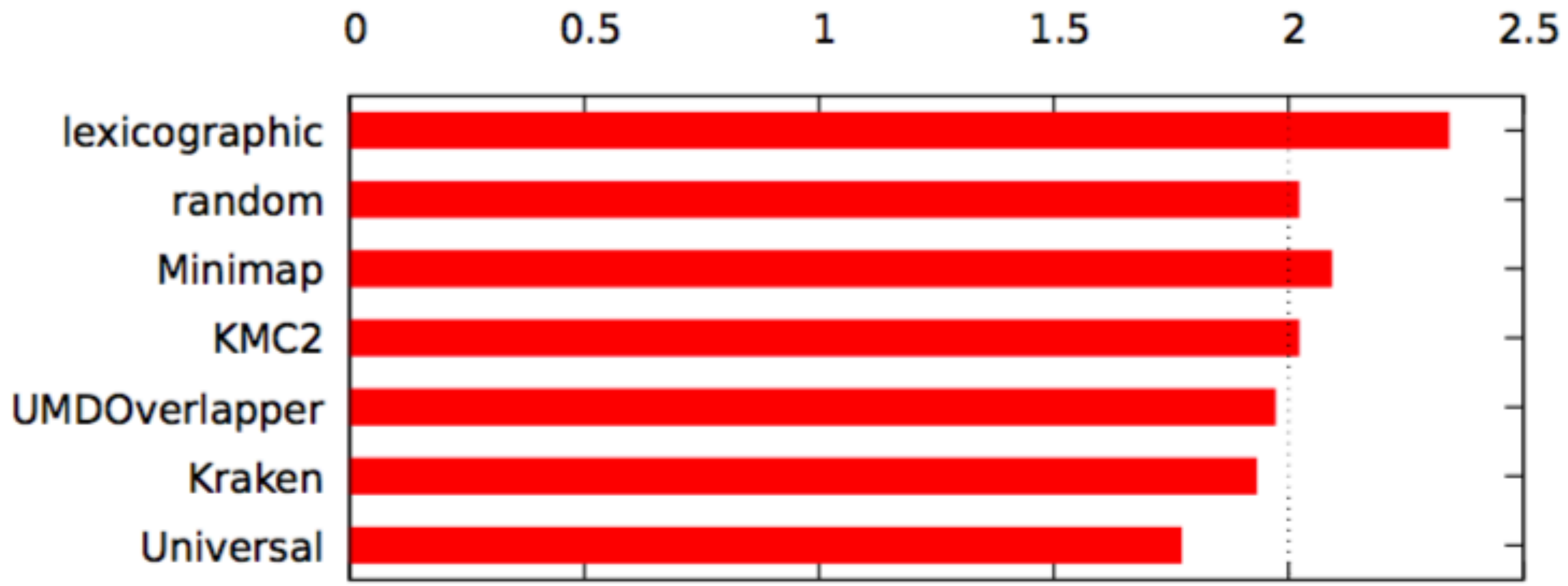
$k = 8$, size of universal hitting set compared to decycling set



Slide adapted from those of Guillaume Marçais

Minimizers

Density for human chromosome 19, k=7, w=11

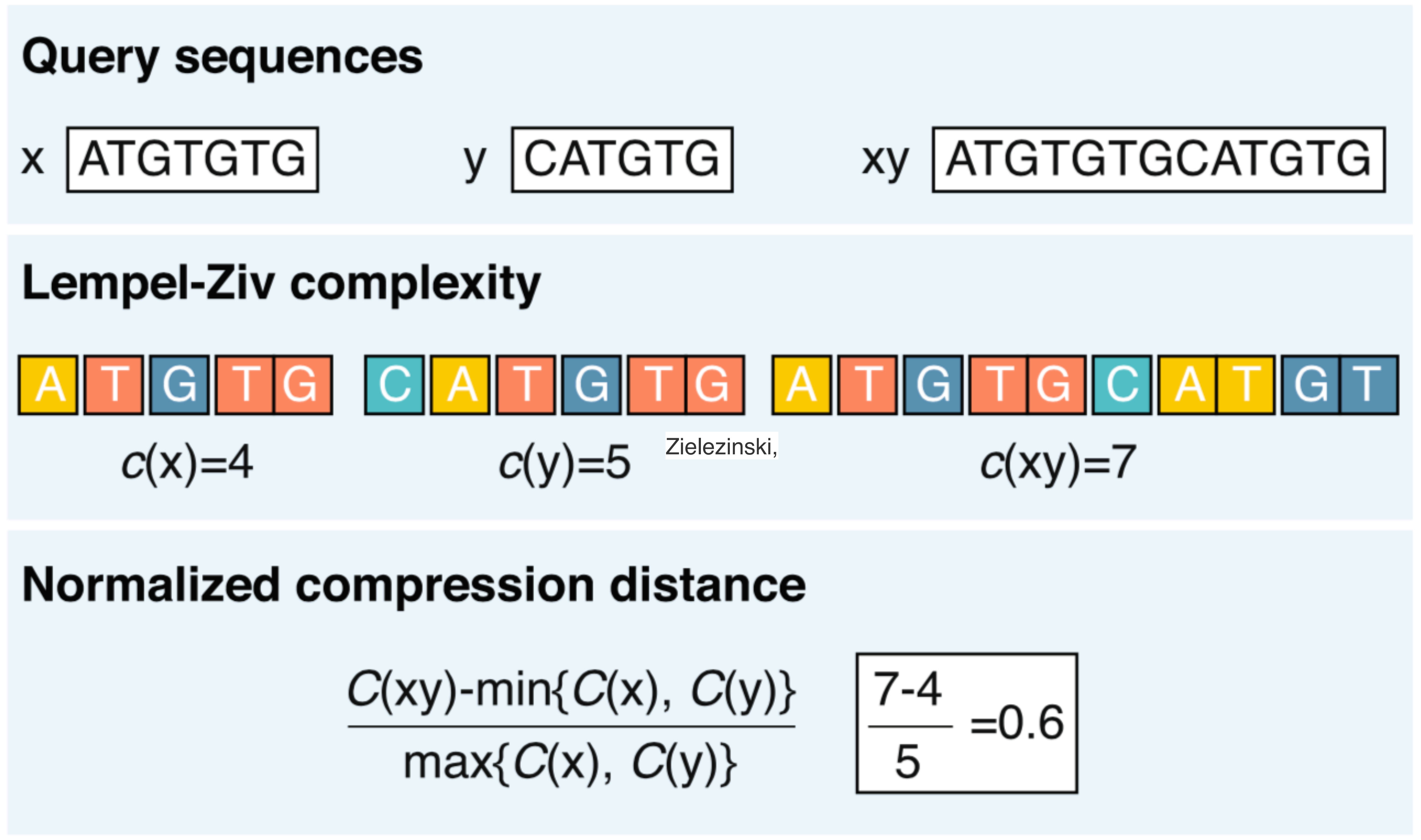


Slide adapted from those of Guillaume Marçais

Lempel-Ziv Complexity

Let the distance between strings be

- compressibility of the concatenated string
- compared to them individually compressed.
- Similar strings will have a higher compression ratio.



$$\frac{C(xy)-\min\{C(x), C(y)\}}{\max\{C(x), C(y)\}} \quad \boxed{\frac{7-4}{5}=0.6}$$

image source: Zielezinski, et al. Alignment-free sequence comparison: benefits, applications, and tools. Genome Biology. 2017.

Kullback-Leibler distance

Use the difference in information content to determine the distance between strings.

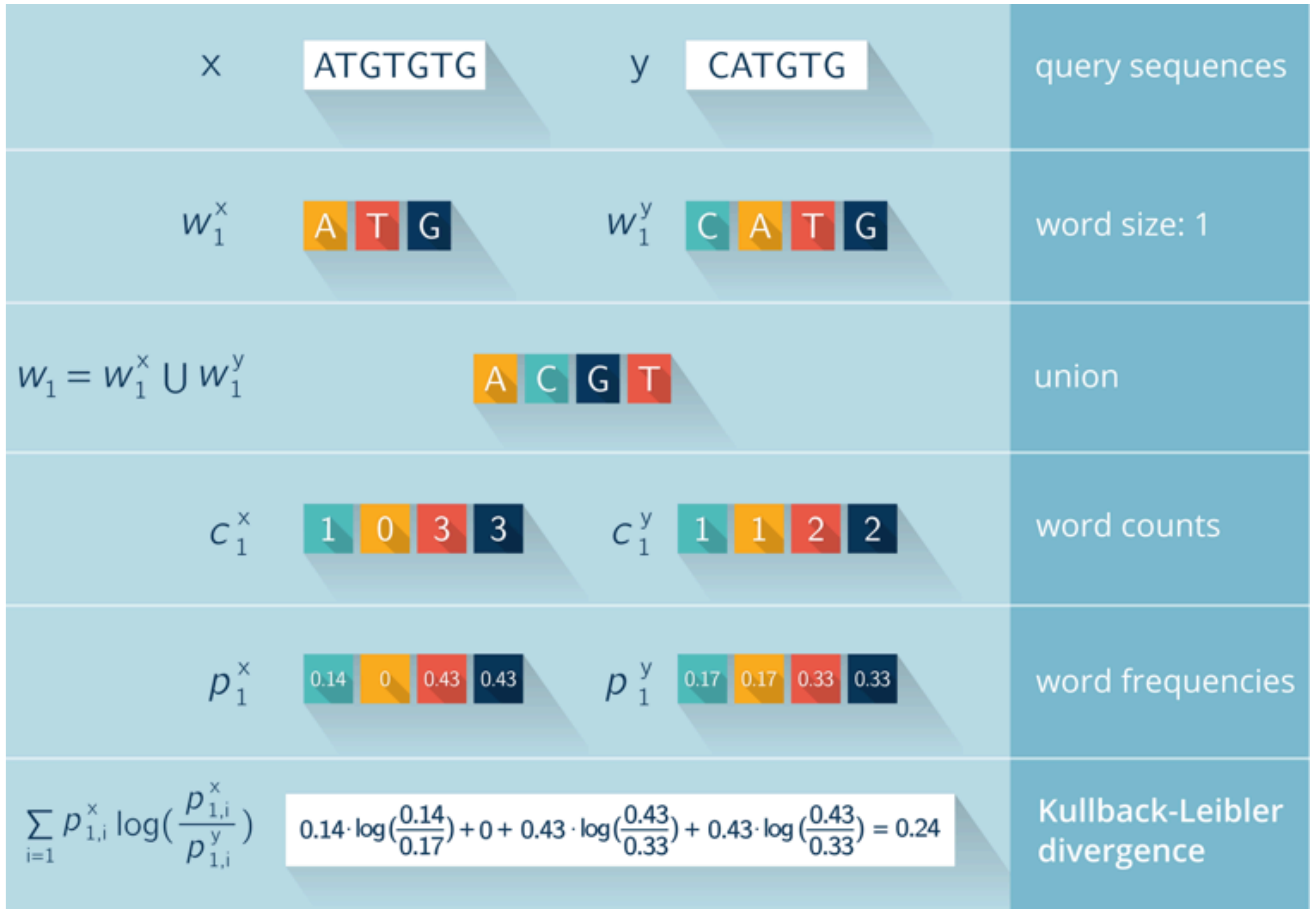


image source: Zielezinski, et al. Alignment-free sequence comparison: benefits, applications, and tools. Genome Biology. 2017.

Metagenomics

- Normally one experiment -> one organism
- Multiple organisms makes for a harder problem
- Not knowing what possible species are there makes it harder still

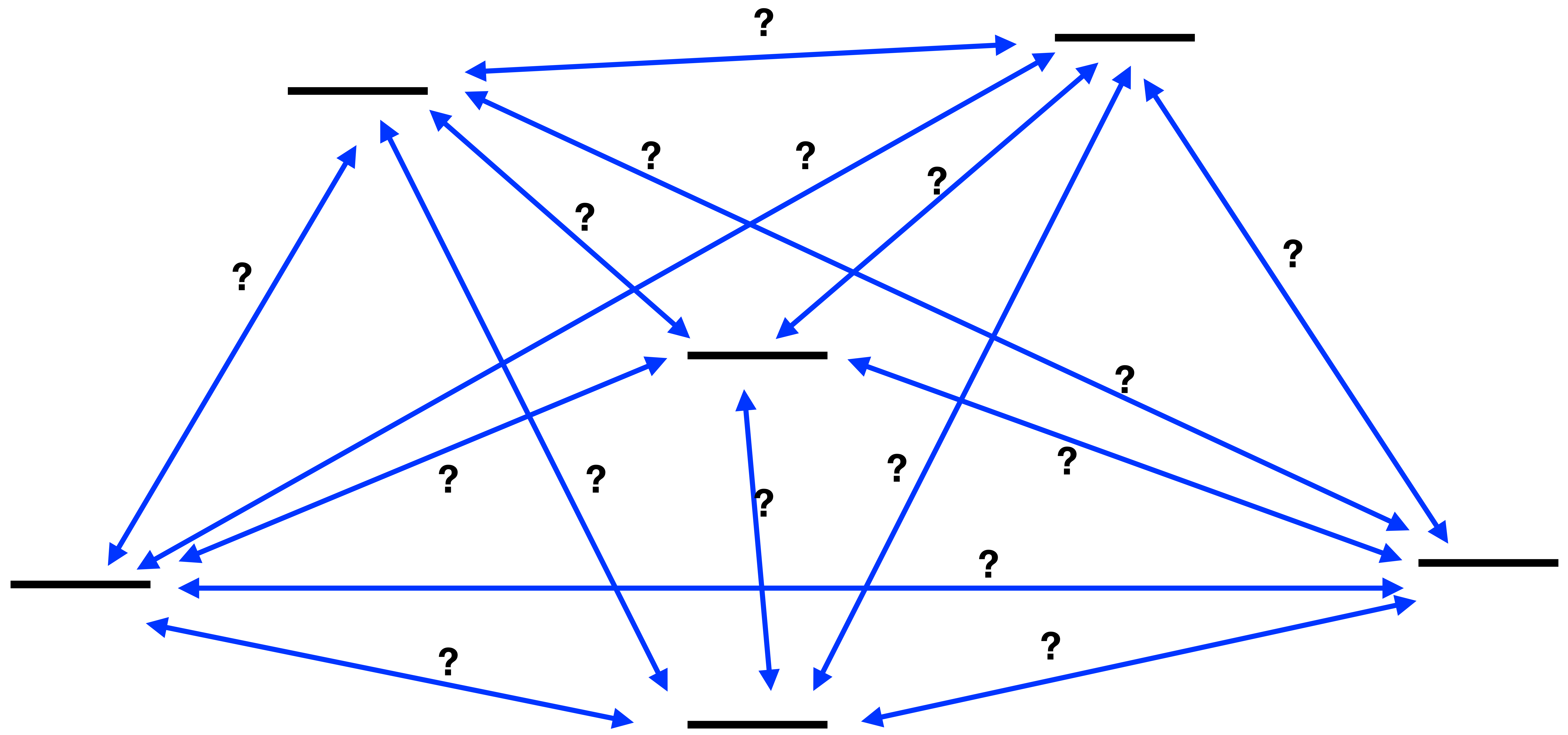
Metagenomics

Given a set of sequences `Q`

- group similar sequences together.
- Equivalent to the classical problem of clustering.
- Distance metrics become important.

Metagenomics

Given a set of sequences, compute all of the pairwise *distance*



Metagenomics

Once the data is clustered

- identify certain groups
- assemble clusters
- generate phylogeny
- ….

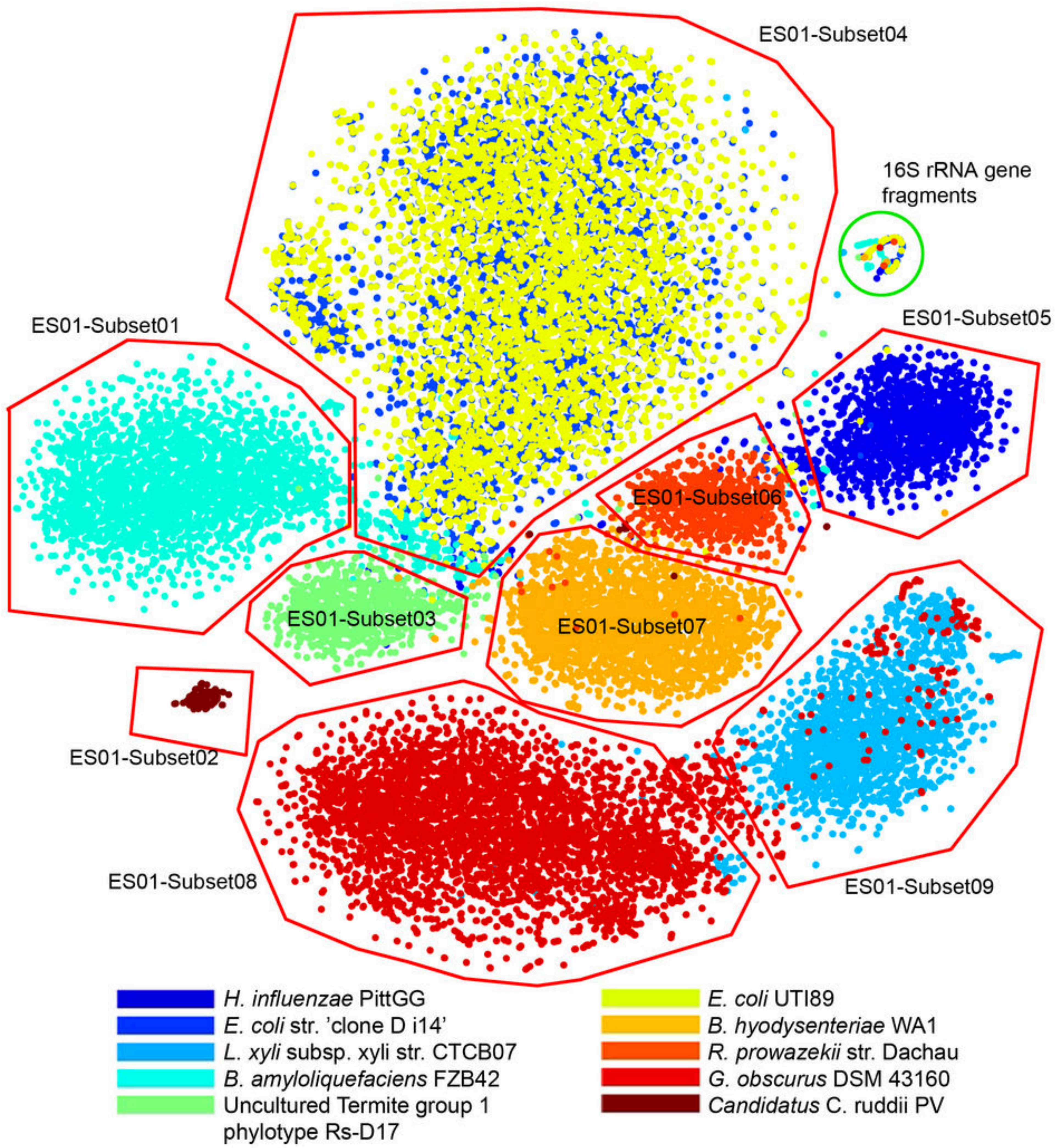
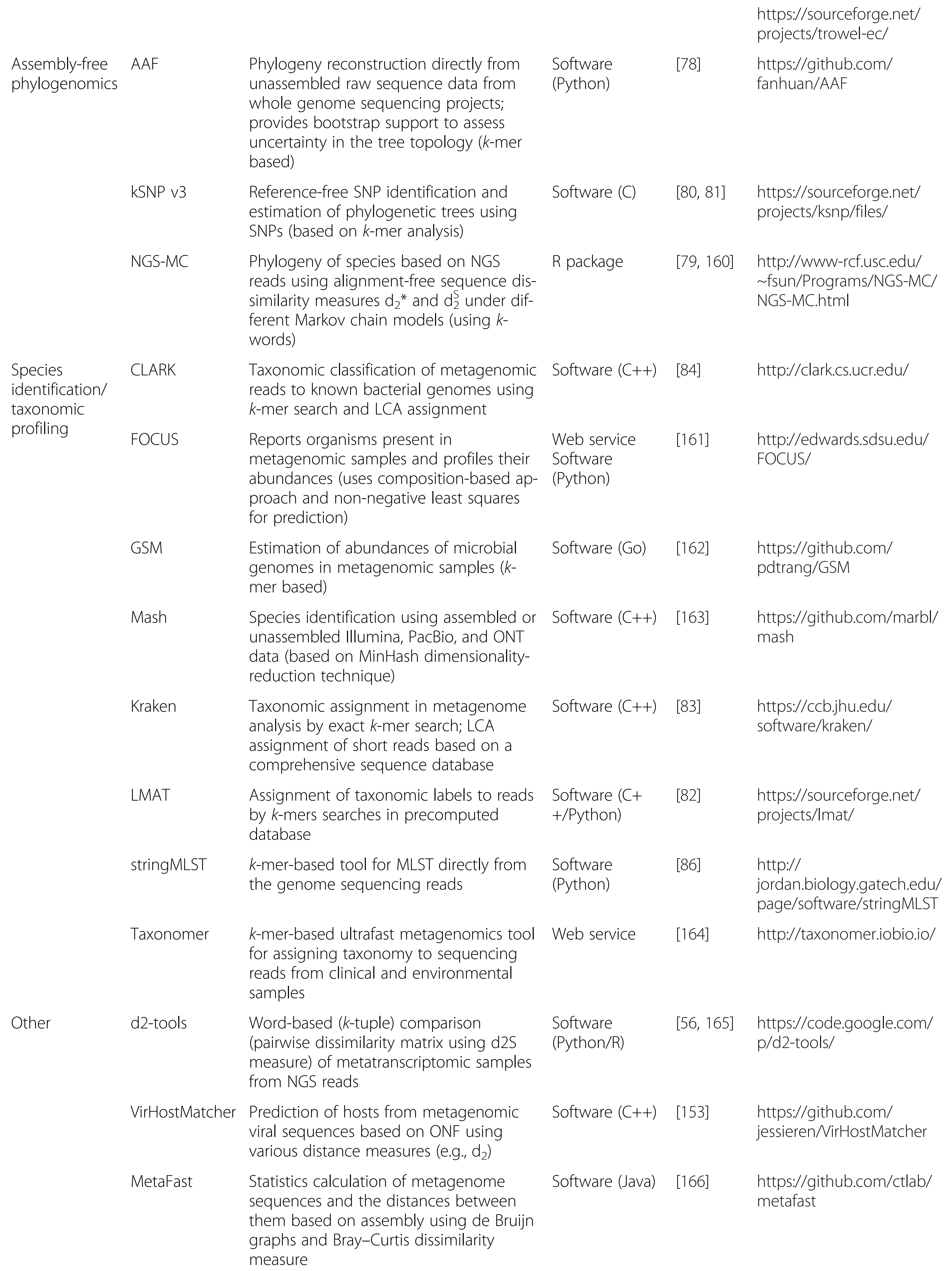


image source: Laczny, et al. Alignment-free Visualization of Metagenomic Data by Nonlinear Dimension Reduction Scientific Reports, 2014

Metagenomics Tools



					https://sourceforge.net/projects/trowel-ec/
Assembly-free phylogenomics	AAF	Phylogeny reconstruction directly from unassembled raw sequence data from whole genome sequencing projects; provides bootstrap support to assess uncertainty in the tree topology (*k*-mer based)	Software (Python)	[78]	https://github.com/fanhuan/AAF
	kSNP v3	Reference-free SNP identification and estimation of phylogenetic trees using SNPs (based on *k*-mer analysis)	Software (C)	[80, 81]	https://sourceforge.net/projects/ksnp/files/
	NGS-MC	Phylogeny of species based on NGS reads using alignment-free sequence dissimilarity measures d_2* and d_2^S under different Markov chain models (using *k*-words)	R package	[79, 160]	http://www-rcf.usc.edu/~fsun/Programs/NGS-MC/NGS-MC.html
Species identification/taxonomic profiling	CLARK	Taxonomic classification of metagenomic reads to known bacterial genomes using *k*-mer search and LCA assignment	Software (C++)	[84]	http://clark.cs.ucr.edu/
	FOCUS	Reports organisms present in metagenomic samples and profiles their abundances (uses composition-based approach and non-negative least squares for prediction)	Web service Software (Python)	[161]	http://edwards.sdsu.edu/FOCUS/
	GSM	Estimation of abundances of microbial genomes in metagenomic samples (*k*-mer based)	Software (Go)	[162]	https://github.com/pdtrang/GSM
	Mash	Species identification using assembled or unassembled Illumina, PacBio, and ONT data (based on MinHash dimensionality-reduction technique)	Software (C++)	[163]	https://github.com/marbl/mash
	Kraken	Taxonomic assignment in metagenome analysis by exact *k*-mer search; LCA assignment of short reads based on a comprehensive sequence database	Software (C++)	[83]	https://ccb.jhu.edu/software/kraken/
	LMAT	Assignment of taxonomic labels to reads by *k*-mers searches in precomputed database	Software (C++/Python)	[82]	https://sourceforge.net/projects/lmat/
	stringMLST	*k*-mer-based tool for MLST directly from the genome sequencing reads	Software (Python)	[86]	http://jordan.biology.gatech.edu/page/software/stringMLST
	Taxonomer	*k*-mer-based ultrafast metagenomics tool for assigning taxonomy to sequencing reads from clinical and environmental samples	Web service	[164]	http://taxonomer.iobio.io/
Other	d2-tools	Word-based (*k*-tuple) comparison (pairwise dissimilarity matrix using d2S measure) of metatranscriptomic samples from NGS reads	Software (Python/R)	[56, 165]	https://code.google.com/p/d2-tools/
	VirHostMatcher	Prediction of hosts from metagenomic viral sequences based on ONF using various distance measures (e.g., d_2)	Software (C++)	[153]	https://github.com/jessieren/VirHostMatcher
	MetaFast	Statistics calculation of metagenome sequences and the distances between them based on assembly using de Bruijn graphs and Bray–Curtis dissimilarity measure	Software (Java)	[166]	https://github.com/ctlab/metafast

image source: Zielezinski, et al. Alignment-free sequence comparison: benefits, applications, and tools. Genome Biology. 2017.

Sailfish

02-715

6 February 2018

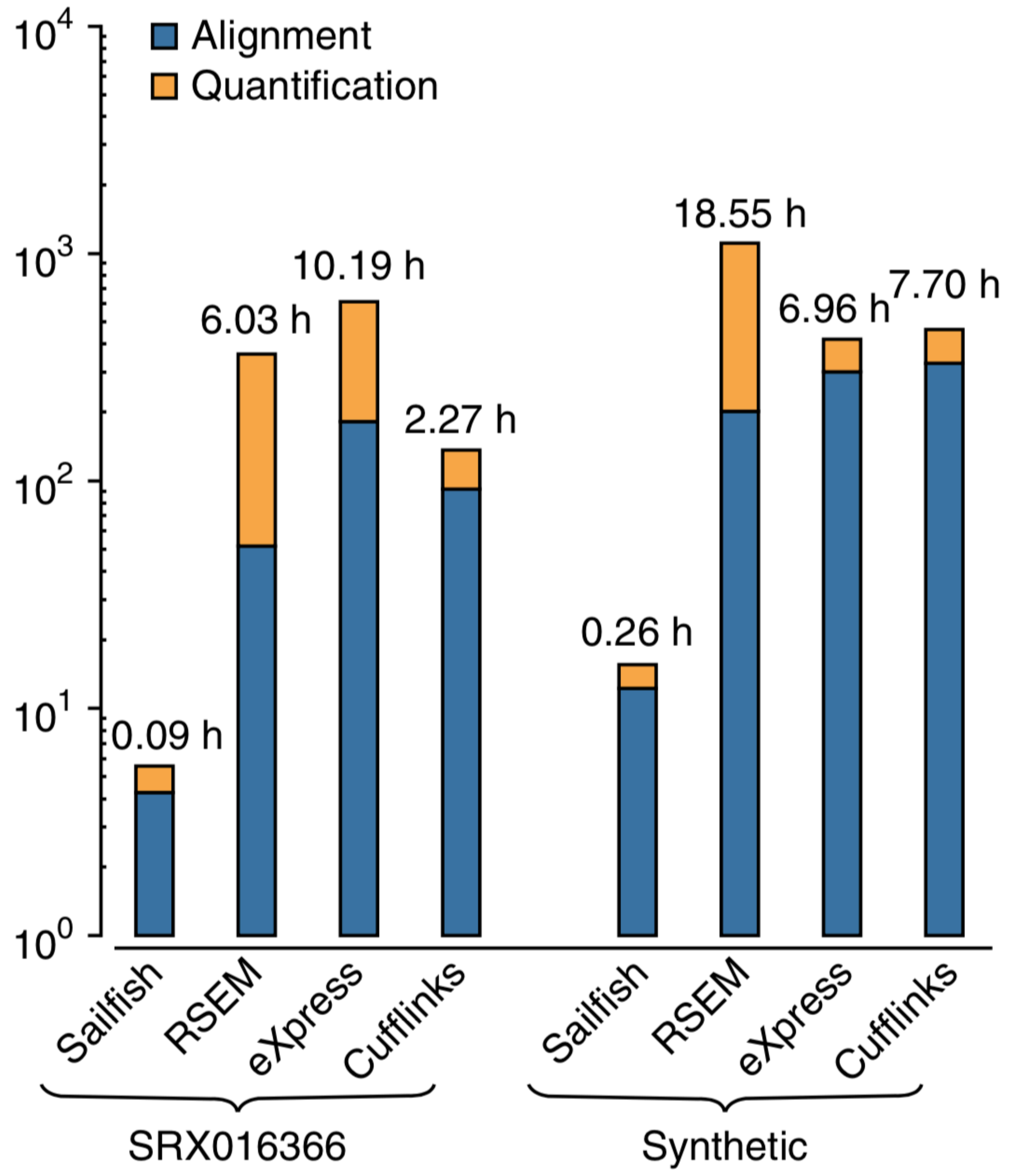
Why?

Quantification is used for

- differential expression
- disease sub-typing, and
- cancer progression analysis

Alignment-free

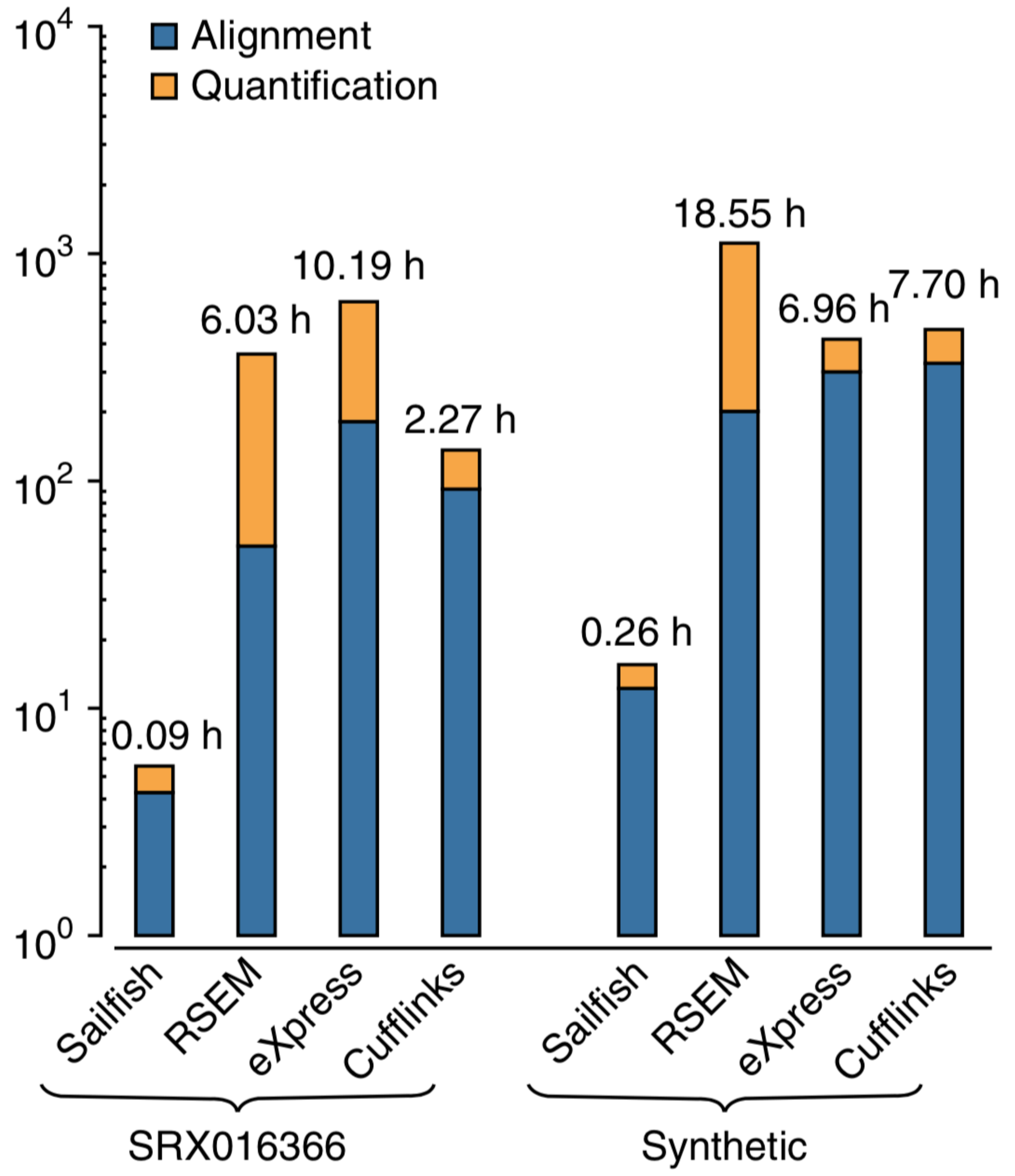
Alignment is slow



Alignment-free

Alignment is slow

Why else?

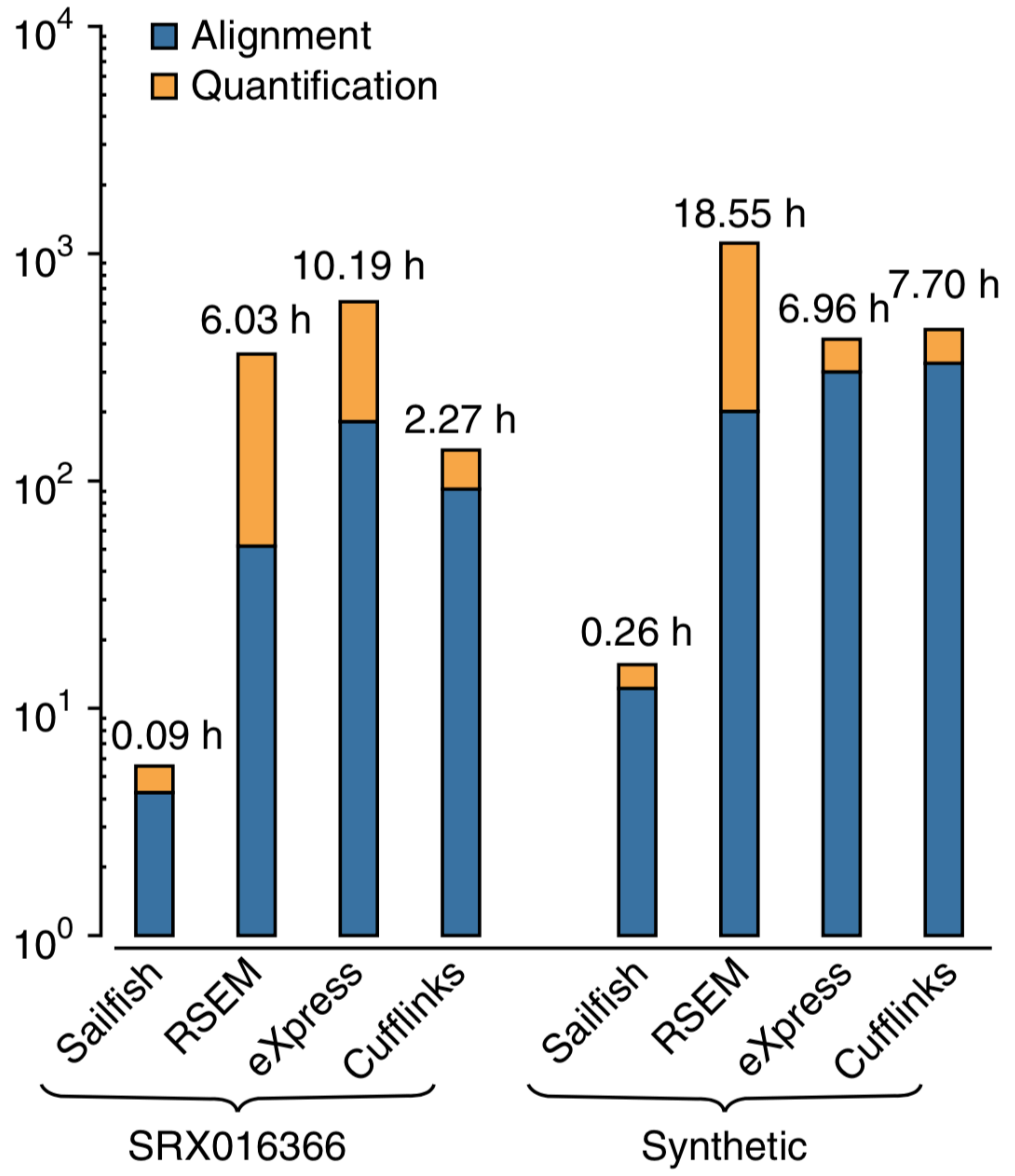


Alignment-free

Alignment is slow

Why else?

- Error tolerance

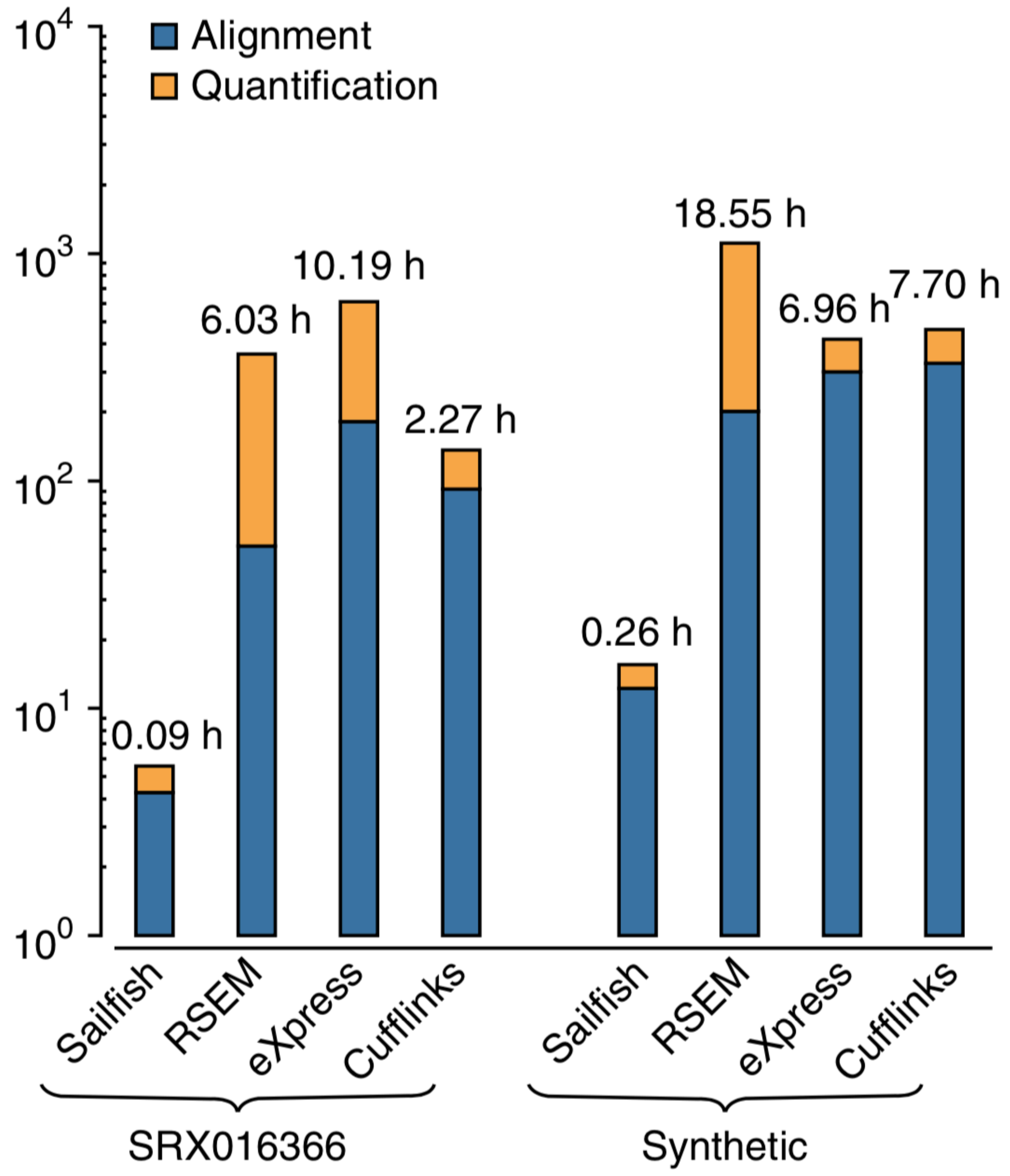


Alignment-free

Alignment is slow

Why else?

- Error tolerance
- Storage size
  - Sailfish on human = 3.1G
  - Bowtie alignment = 15.5G

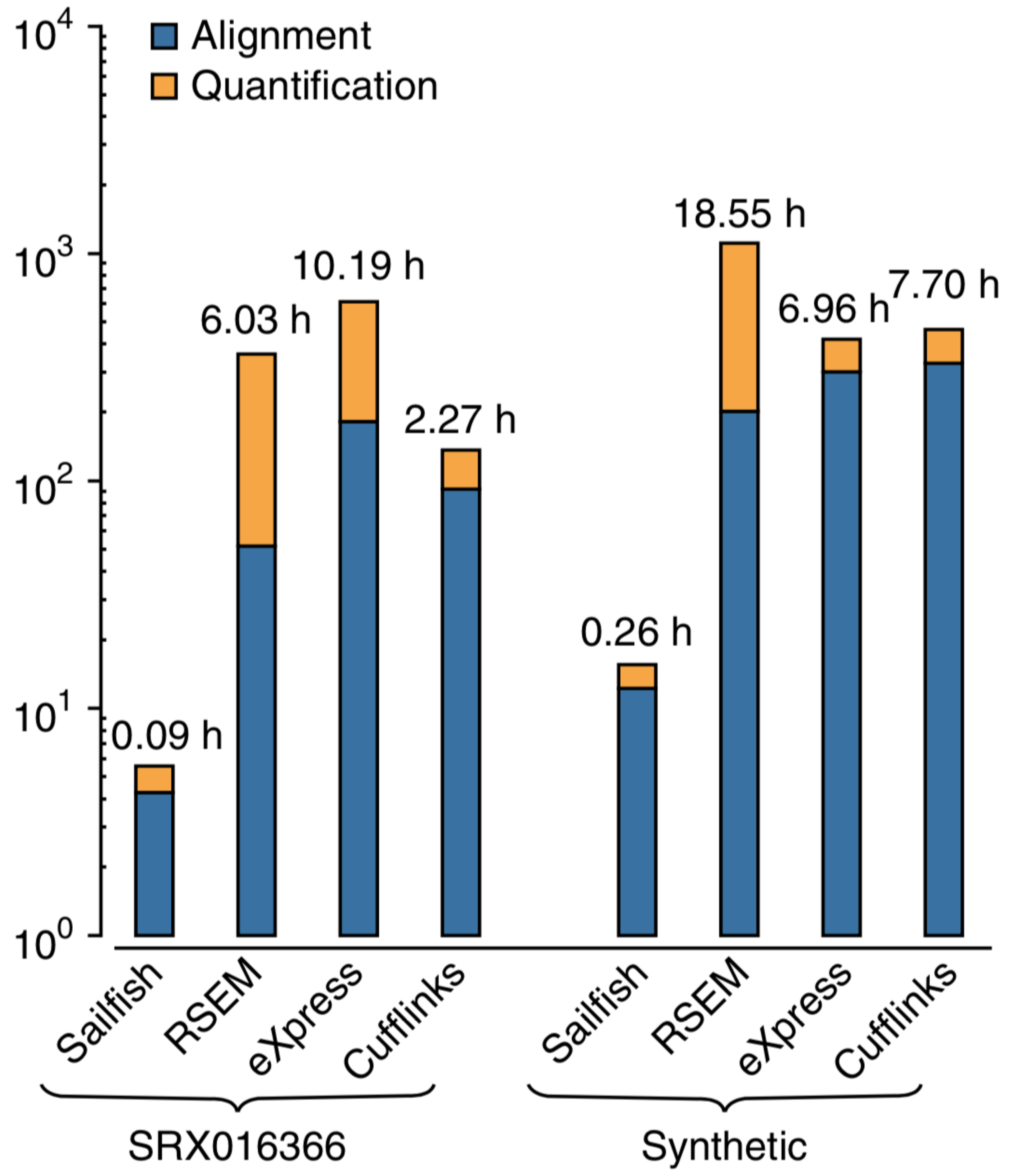


Alignment-free

Alignment is slow

Why else?

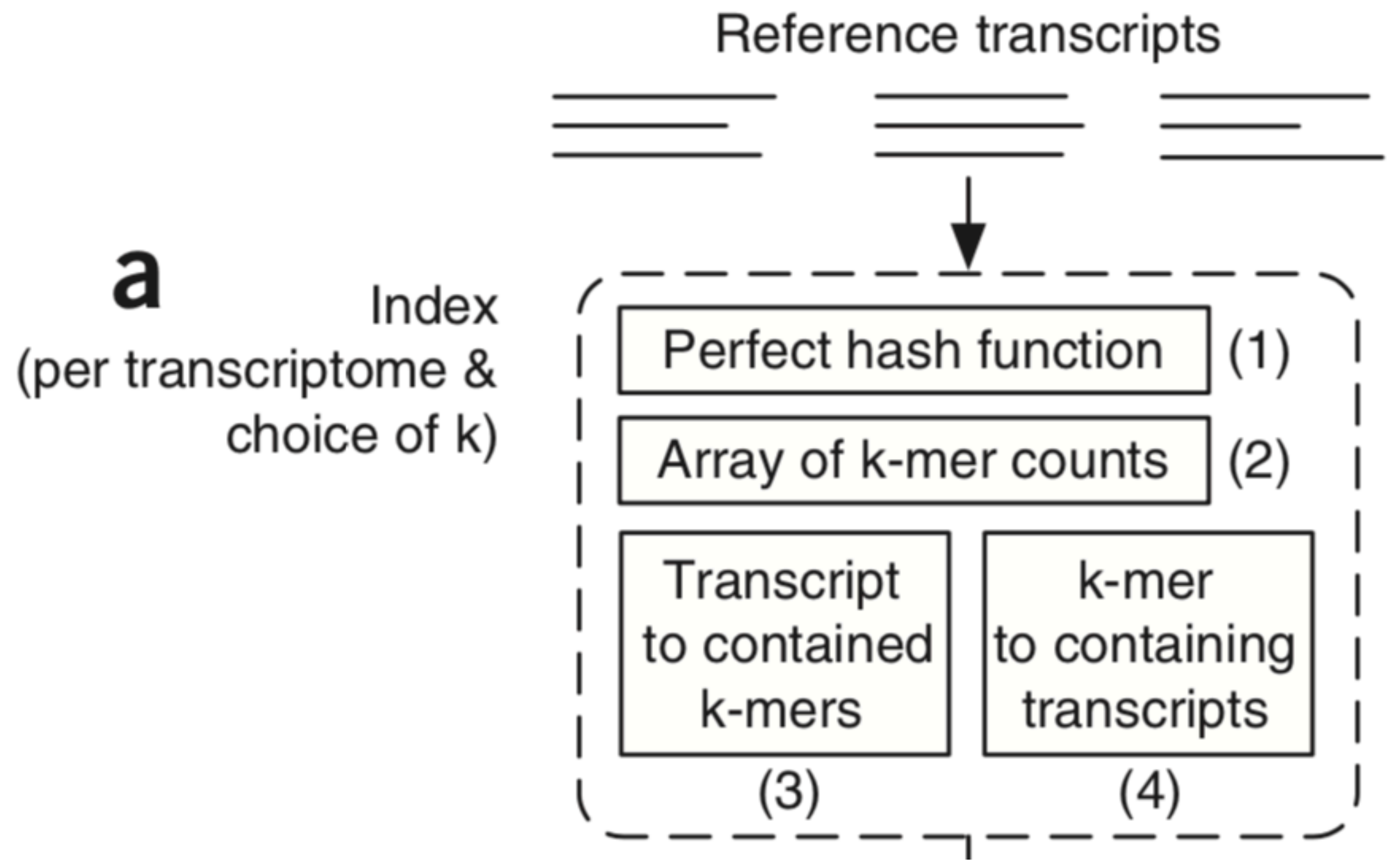
- Error tolerance
- Storage size
  - Sailfish on human = 3.1G
  - Bowtie alignment = 15.5G
- Memory usage



The Index

The index is the key it contains

- a perfect hash of k-mers (in T) to indices
- a count of k-mer use
- maps from T to k-mers and vice versa



a
Reference transcripts
Index (per transcriptome & choice of k)
Perfect hash function (1)
Array of k-mer counts (2)
Transcript to contained k-mers (3)
k-mer to containing transcripts (4)

Equivalence Classes

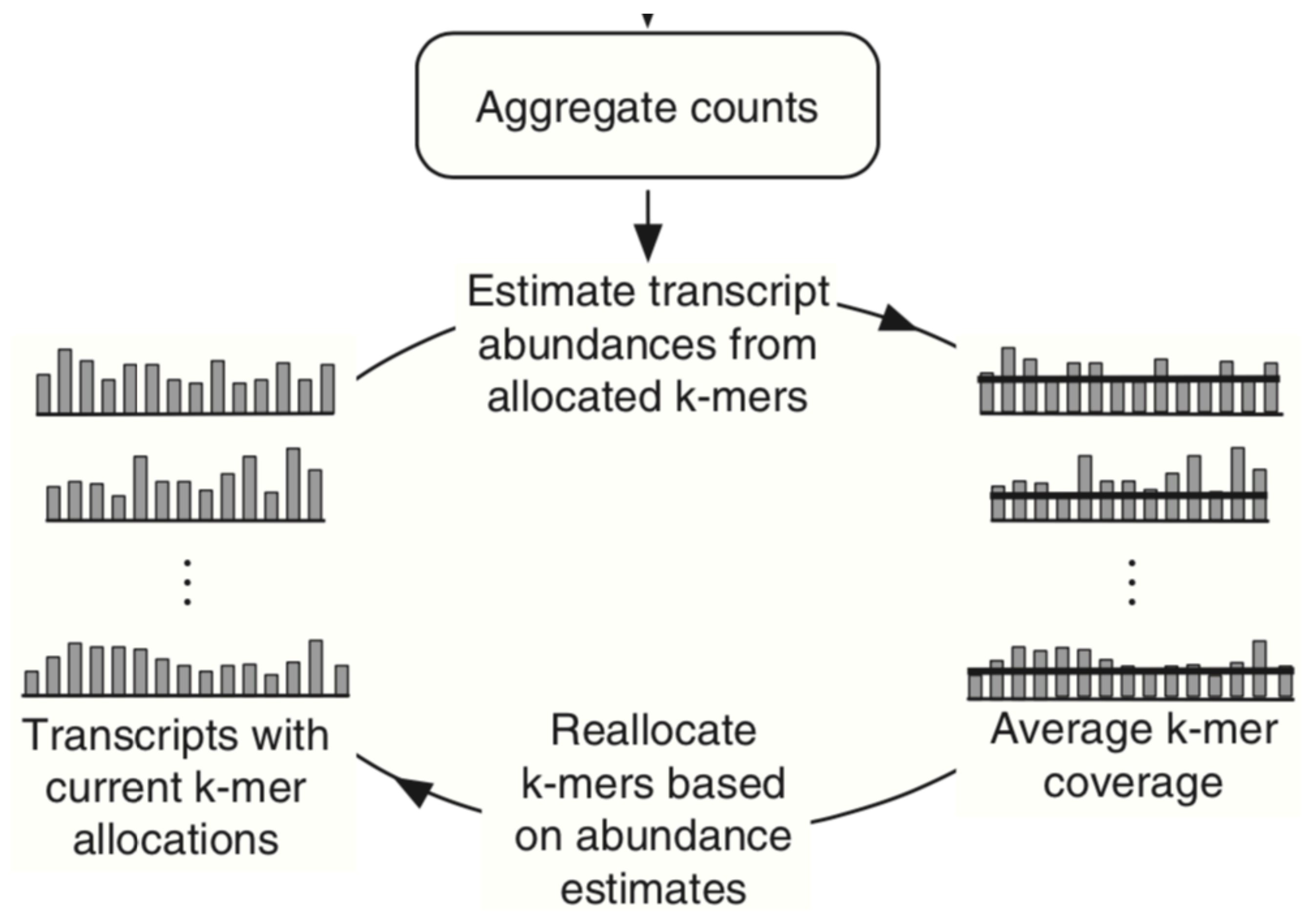
An equivelence class is

- all k-mers
- that appear in the same set of transcripts
- with the same frequency

For k=20 and the human experiment

•number of k-mers: 4^20 = 2^40 =	1,099,511,627,776
•number of k-mers in the human transcriptome =	60,504,111
•number of k-mers in the human read set =	39,393,132
•number of distinct equivalence classes =	151,385
	(0.38%)

EM


Aggregate counts
Estimate transcript abundances from allocated k-mers
Average k-mer coverage
Reallocate k-mers based on abundance estimates
Transcripts with current k-mer allocations

Initial μ’

$$\alpha(j,i) = \frac{\mu_i' \mathrm{L}(s_j)}{\sum_{t \supseteq [s_j]} \mu_t'}$$

$$\mu_i' = \frac{\mu_i}{\sum_{t_j \in T} \mu_j}$$

$$\mu_i = \frac{\sum_{[s_j] \subseteq t_i} \alpha(j,i)}{l_i'}$$

$$\mathrm{L}(s_i) = \sum_{s_j \in [s_i]} C_{\mathfrak{R}}\left(h(s_j)\right)$$

Why is $l_i' = l - k + 1$?

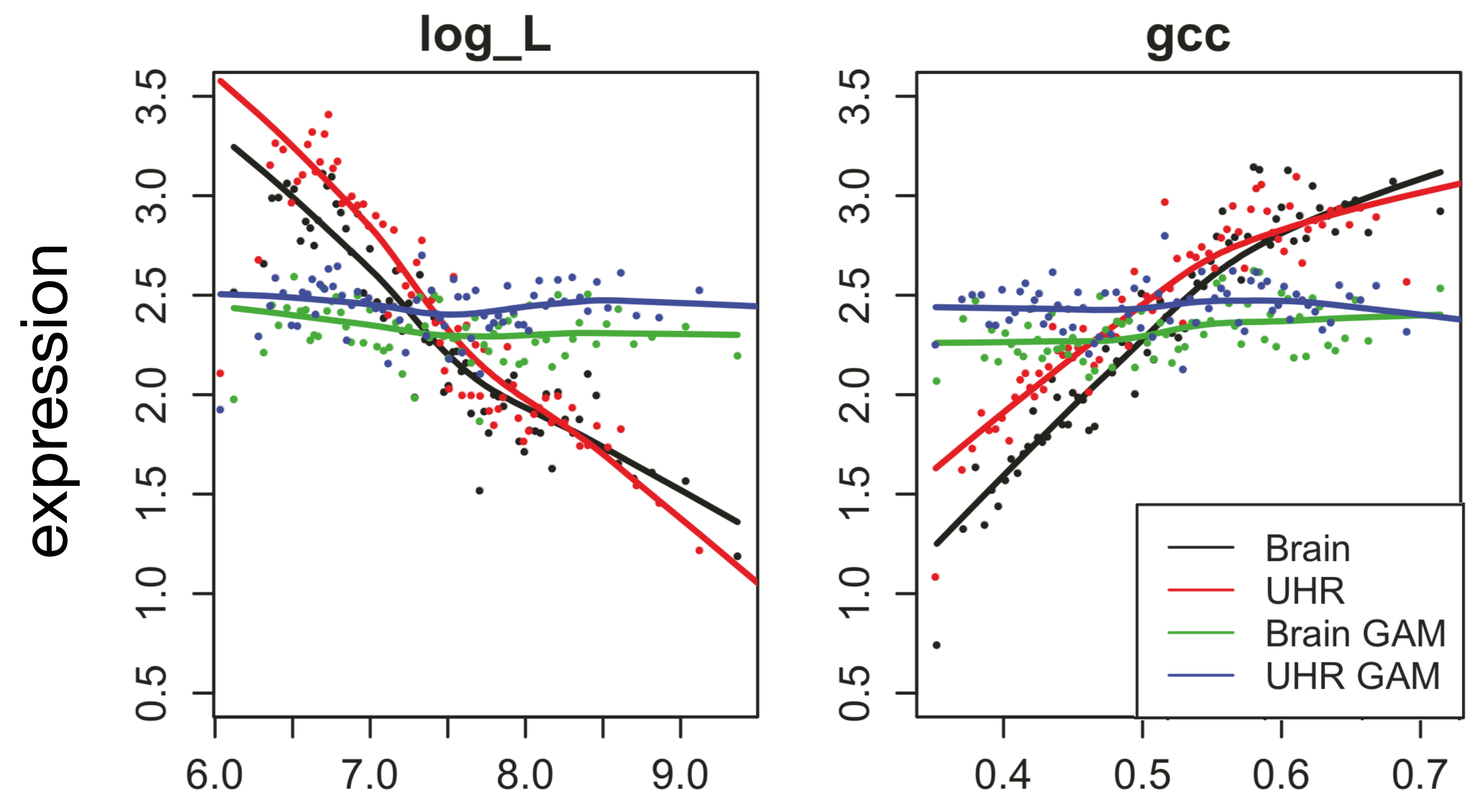
Bias Correction

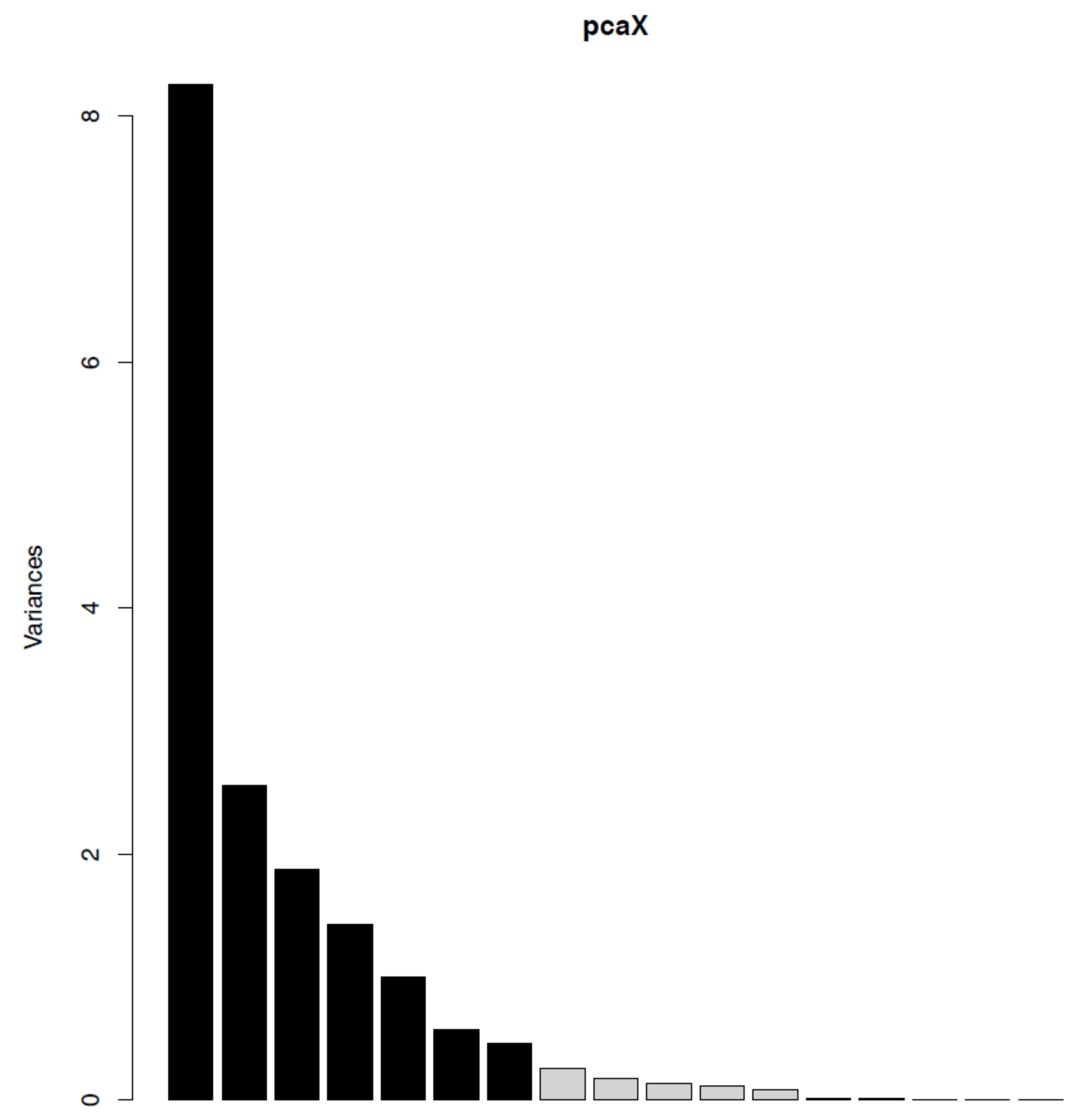
Expression prediction can be impacted by

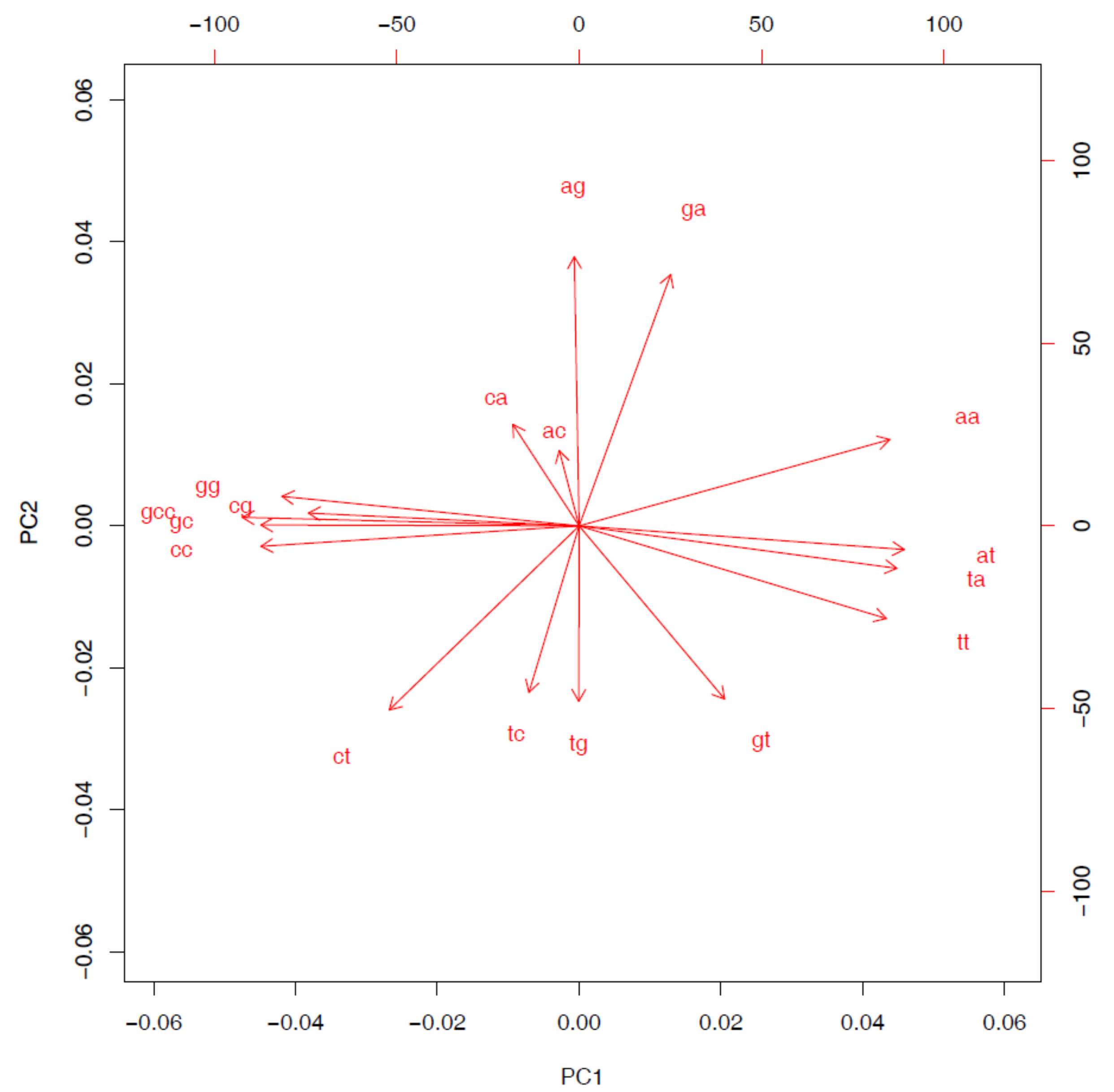
- fragment length
- CG content
- dinucleotide frequency

Corrections are made following procedures in Zeng, *et al.* (2011)

- measure length, GC and DN frequency
- find PCs for GC and DN
- fit a correction function





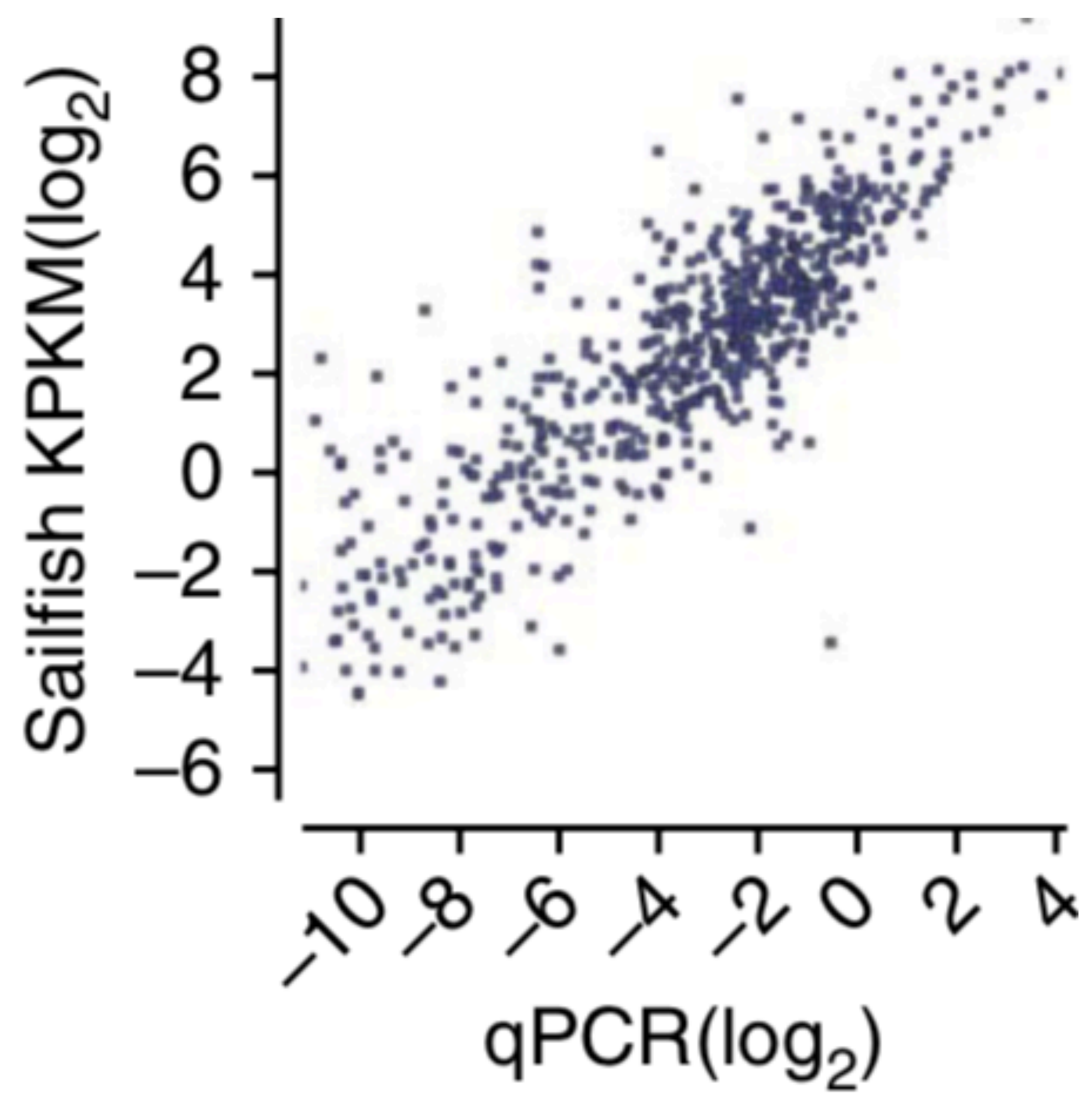


Real Data

$$\text{TPM}_i = 10^6 \mu_i'$$

$$\text{KPKM}_i = \frac{\left.\frac{C_i}{l_i}\right/ 10^3}{\left.N\right/ 10^6} = \frac{10^9 \frac{C_i}{l_i}}{N} \approx \frac{10^9 \mu_i}{N}$$

k-mers per kilobase, per million mapped k-mers

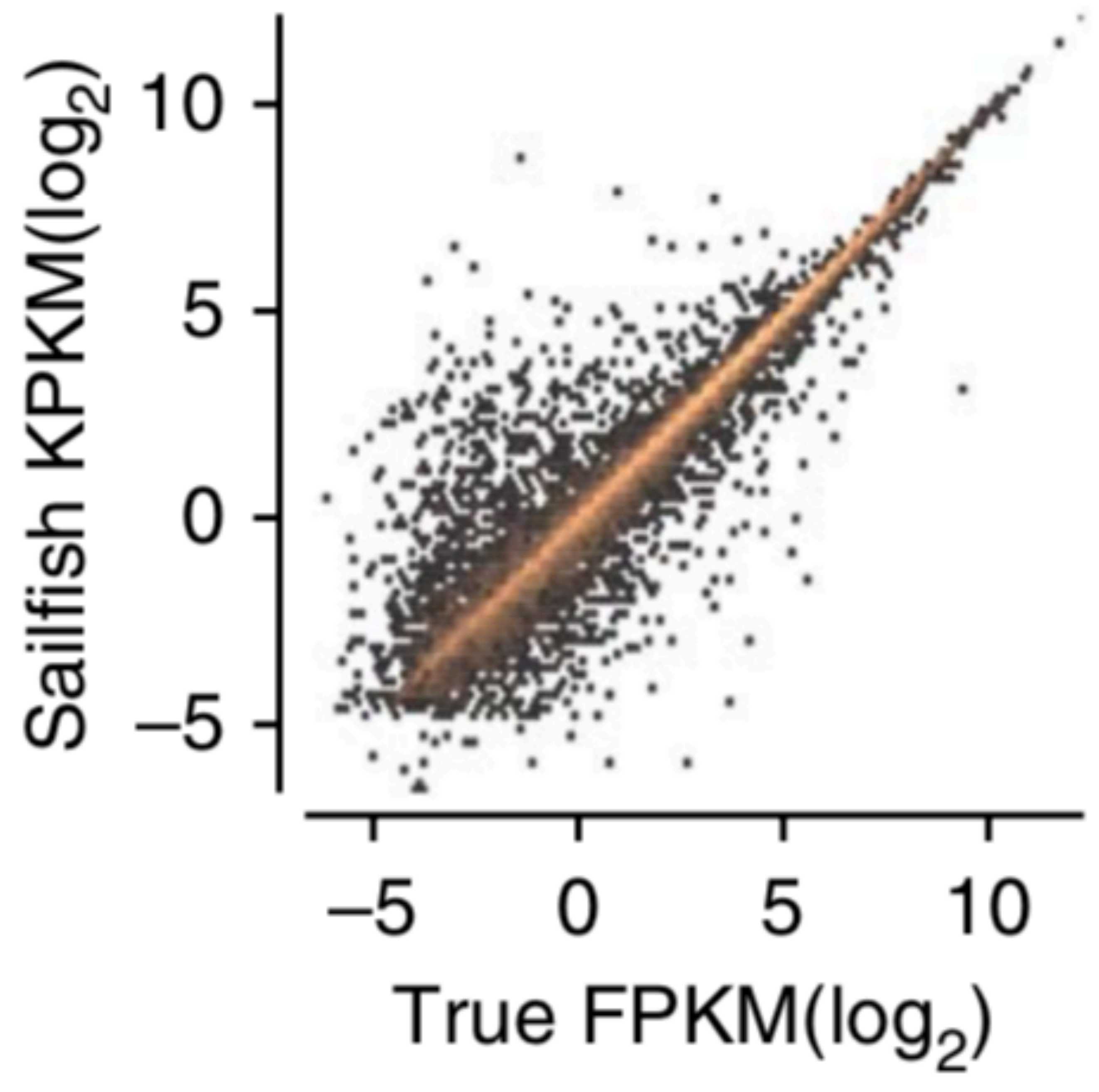


	Human brain tissue			
	Sailfish	RSEM	eXpress	Cufflinks
Pearson	0.85	0.82	0.85	0.85
Spearman	0.84	0.80	0.85	0.85

Simulation

$$\mathrm{PE}(x_i, y_i) = 100 \times \frac{|x_i - y_i|}{x_i}$$

$$\mathrm{RMSE}(\mathbf{x}, \mathbf{y}) = \sqrt{\frac{\sum_{i=1}^{n}(x_i - y_i)^2}{n}}$$



	Synthetic			
	Sailfish	RSEM	eXpress	Cufflinks
Pearson	0.96	0.96	0.95	0.94
Spearman	0.76	0.77	0.77	0.75
RMSE	6.06	8.90	8.83	10.05
medPE	6.50	12.48	14.06	12.42

Take Aways

What was new

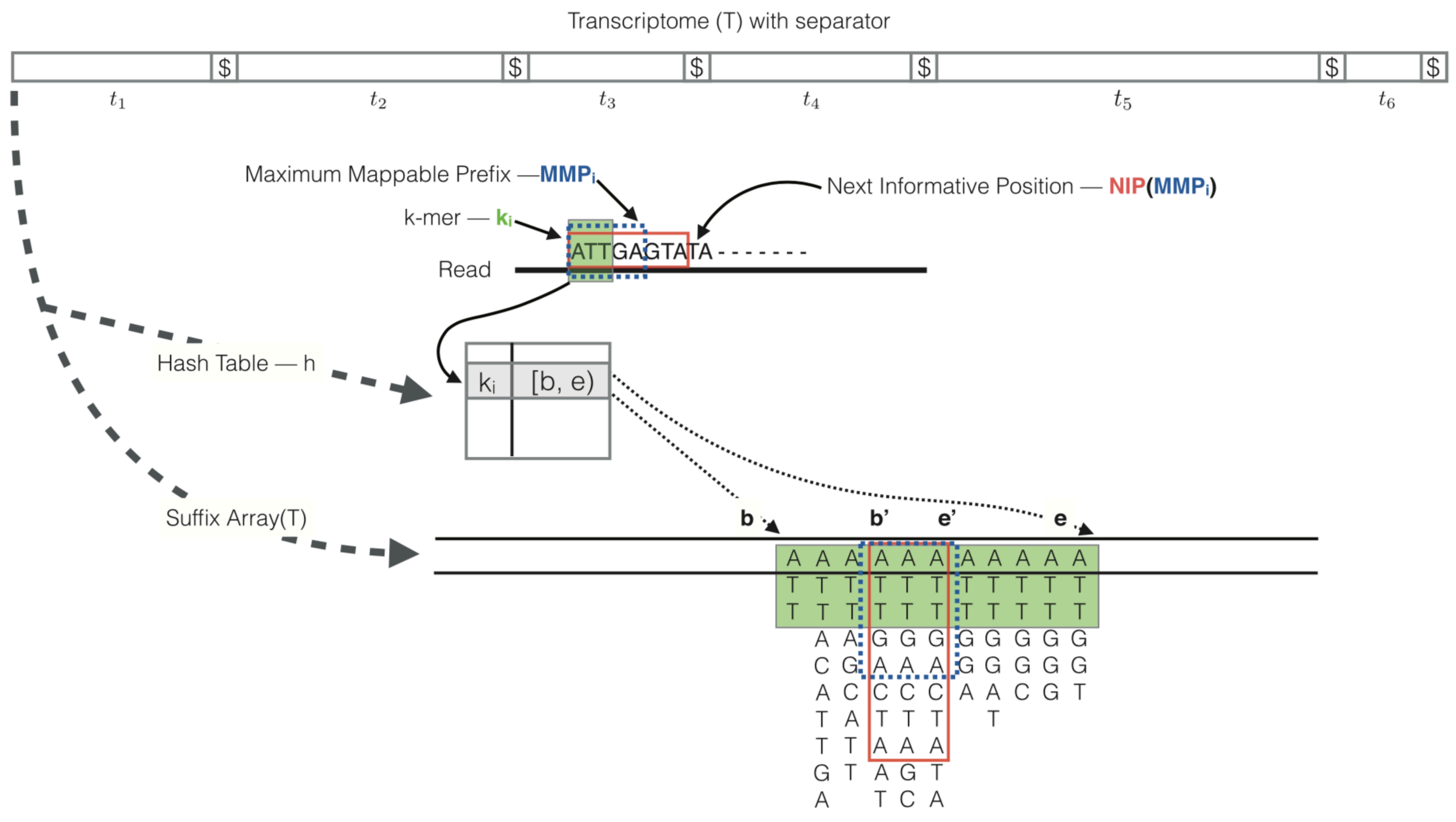
- alignment-free quantification is fast
- similar accuracy to alignment-based methods
- EM efficiently takes care of multi-mapped reads
- single input parameter

Problems

- biases are not addressed fully

Going beyond k-mer counts

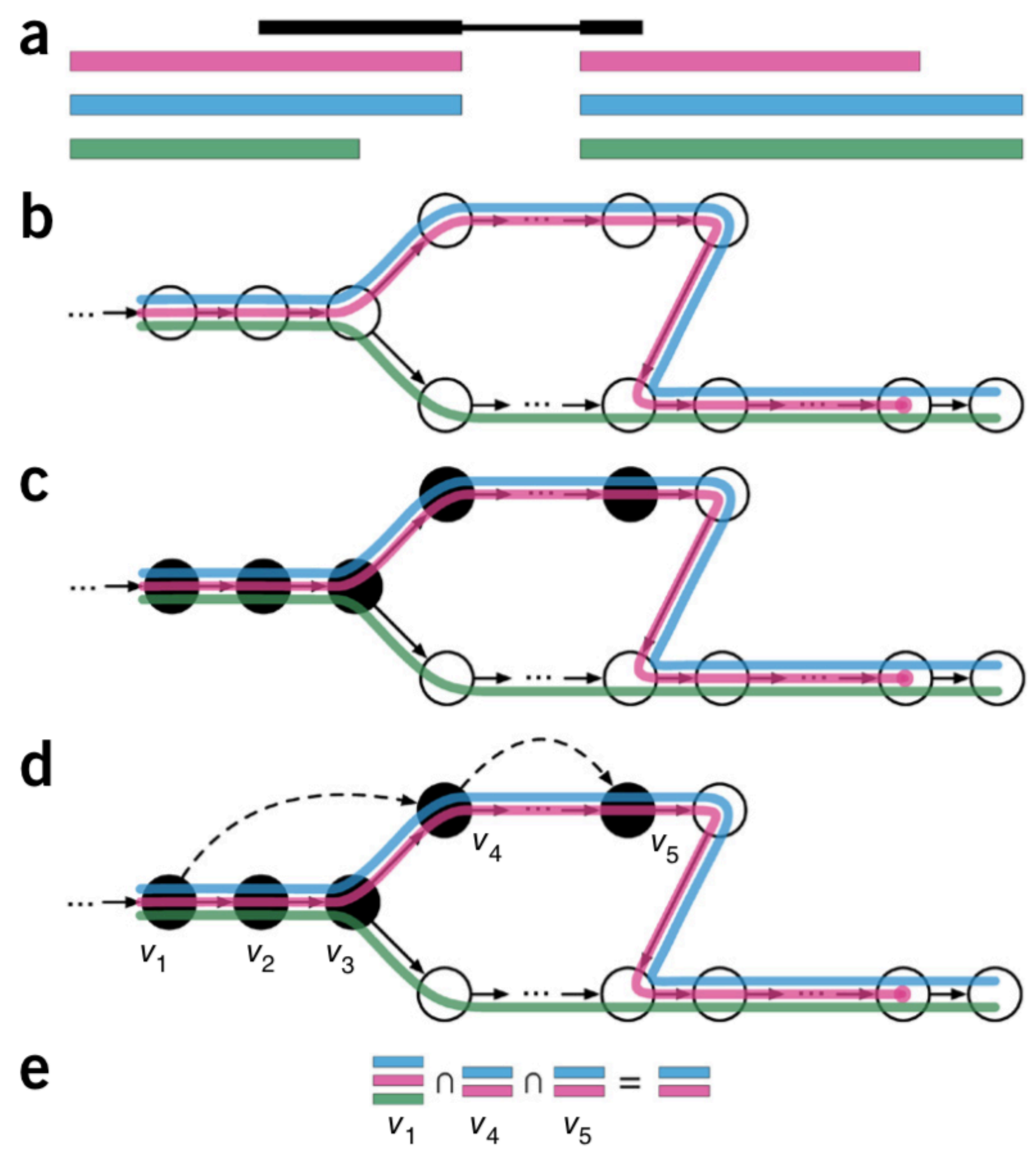
Salmon



Transcriptome (T) with separator
$
t_1 t_2 t_3 t_4 t_5 t_6
Maximum Mappable Prefix — MMPi
Next Informative Position — NIP(MMPi)
k-mer — ki
ATTGAGTATA
Read
Hash Table — h
ki [b, e)
Suffix Array(T)
b b' e' e

Patro, *et al.* Nature Biotechnology, 2017
Srivastava, *et al.* Bioinformatics, 2016

kallisto



Bray, *et al.* Nature Biotechnology, 2016

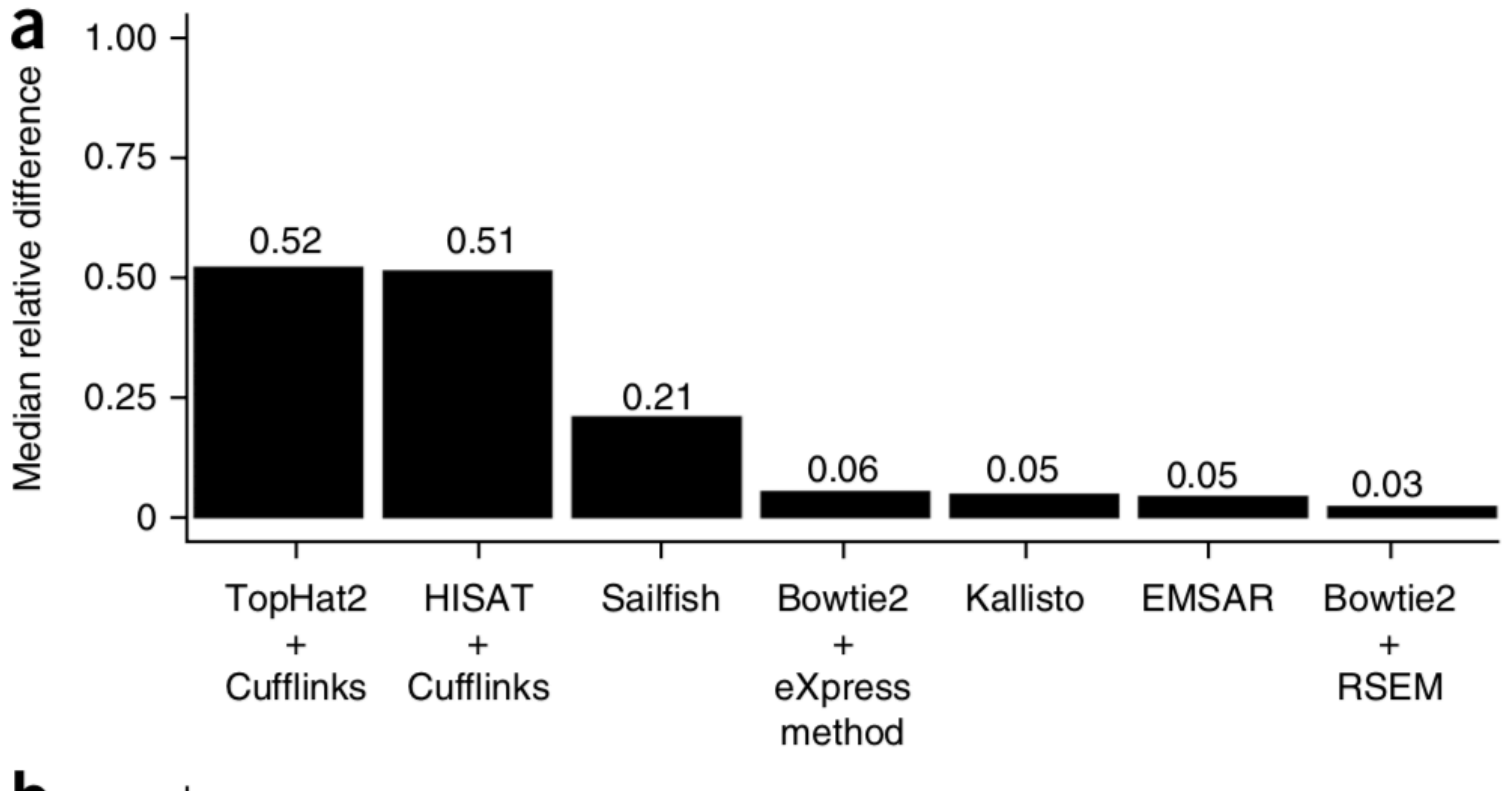
Advanced Bias Correction

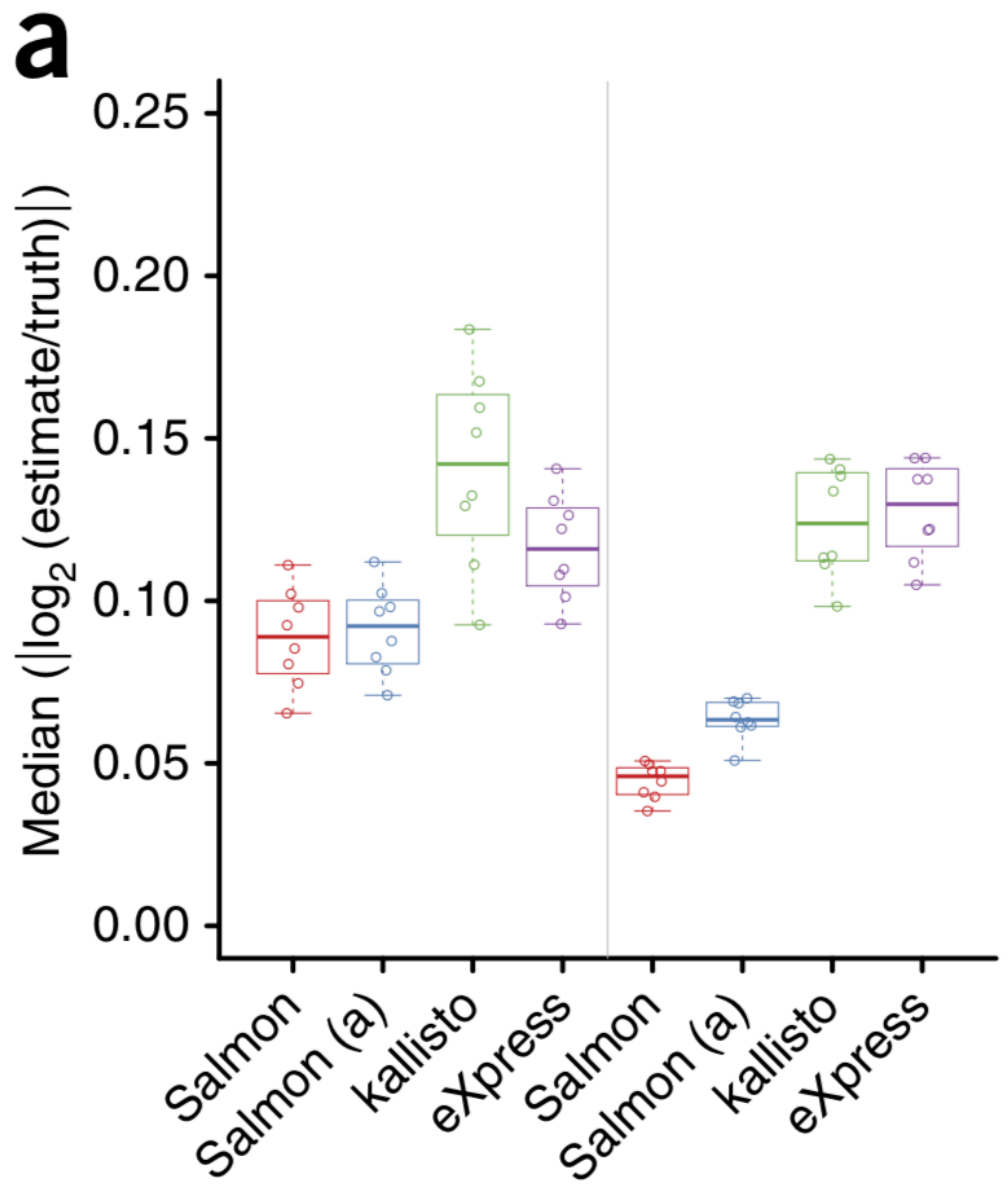
Salmon extends the BC from Sailfish

- considers 5’- and 3’- sequencing bias
- fragment-level GC bias
- length-bias

$$\widetilde{\ell'_i} = \sum_{j=1}^{\ell_i} \sum_{k=1}^{f_i(j,L)} \frac{b_{gc^+}(t_i, j, j+k)}{b_{gc^-}(t_i, j, j+k)} \cdot \frac{b_{s^+}^{5'}(t_i, j)}{b_{s^-}^{5'}(t_i, j)} \cdot \frac{b_{s^+}^{3'}(t_i, j+k)}{b_{s^-}^{3'}(t_i, j+k)} \cdot \Pr\{X = j\}$$

Accuracy of new methods





Suggested Reading

Metagenomics

- G. G. Z. Silva, D. A. Cuevas, B. E. Dutilh, and R. A. Edwards, “FOCUS: an alignment-free model to identify organisms in metagenomes using non-negative least squares,” PeerJ, vol. 2, no. 4, p. e425, Jun. 2014.
- **W. Liao, J. Ren, K. Wang, S. Wang, F. Zeng, Y. Wang, and F. Sun, “Alignment-free Transcriptomic and Metatranscriptomic Comparison Using Sequencing Signatures with Variable Length Markov Chains,” Scientific Reports, vol. 6, no. 1, p. 37243, Nov. 2016.**
- Y. Y. Lu, K. Tang, J. Ren, J. A. Fuhrman, M. S. Waterman, and F. Sun, “CAFE: aCcelerated Alignment-FrEe sequence analysis,” Nucleic Acids Research, vol. 45, no. 1, pp. W554–W559, Jul. 2017.
- S. Gao, D.-T. Pham, and V. Phan, “Alignment-free methods for metagenomic profiling,” BMC Bioinformatics, vol. 16, no. 15, p. P4, 2015.
- **V. Popic, V. Kuleshov, M. Snyder, and S. Batzoglou, “GATTACA: Lightweight Metagenomic Binning With Compact Indexing Of Kmer Counts And MinHash-based Panel Selection,” bioRxiv, p. 130997, Apr. 2017.**
- B. D. Ondov, T. J. Treangen, P. Melsted, A. B. Mallonee, N. H. Bergman, S. Koren, and A. M. Phillippy, “Mash: fast genome and metagenome distance estimation using MinHash,” Genome Biol., vol. 17, no. 1, p. 403, Jun. 2016.
- **D. E. Wood and S. L. Salzberg, “Kraken: ultrafast metagenomic sequence classification using exact alignments,” Genome Biol., vol. 15, no. 3, p. R46, 2014.**
- *Y. Y. Lu, T. Chen, J. A. Fuhrman, and F. Sun, “COCACOLA: binning metagenomic contigs using sequence COmposition, read CoverAge, CO-alignment and paired-end read LinkAge,” Bioinformatics, vol. 33, no. 6, pp. 791–798, Mar. 2017.*
- **Y. Luo, Y. W. Yu, J. Zeng, B. Berger, and J. Peng, “Metagenomic binning through low density hashing,” bioRxiv, p. 133116, May 2017.**
- J. Kawulok and S. Deorowicz, “CoMeta: Classification of Metagenomes Using k-mers,” PLoS ONE, vol. 10, no. 4, p. e0121453, Apr. 2015.
- D.-T. Pham, S. Gao, and V. Phan, “An accurate and fast alignment-free method for profiling microbial communities,” Journal of Bioinformatics and Computational Biology, vol. 15, no. 3, p. 1740001, Mar. 2017. (see me for the PDF)
- J. Pell, A. Hintze, R. Canino-Koning, A. Howe, J. M. Tiedje, and C. T. Brown, “Scaling metagenome sequence assembly with probabilistic de Bruijn graphs.,” Proceedings of the National Academy of Sciences, vol. 109, no. 33, pp. 13272–13277, Aug. 2012.

Suggested Reading

<u>Quantification</u>

- **R. Patro, S. M. Mount, and C. Kingsford, “Sailfish enables alignment-free isoform quantification from RNA-seq reads using lightweight algorithms,” Nature Biotechnology, vol. 32, no. 5, pp. 462–464, Apr. 2014.**
- *R. Patro, G. Duggal, M. I. Love, R. A. Irizarry, and C. Kingsford, “Salmon provides fast and bias-aware quantification of transcript expression,” Nat. Methods, vol. 14, no. 4, pp. 417–419, Mar. 2017.*
- Nicolas L Bray, Harold Pimentel, Páll Melsted and Lior Pachter, "Near-optimal probabilistic RNA-seq quantification", Nature Biotechnology vol. 34, pp. 525–527, 2016.
- D. C. Wu, J. Yao, K. S. Ho, A. M. Lambowitz, C. O. Wilke, "Limitation of alignment-free tools in total RNA-seq quantification", bioRxiv, 10.1101/246967 (review)

<u>Minimizers</u>

- H. Li, “Minimap2: fast pairwise alignment for long nucleotide sequences.” 04-Aug-2017.
- Y. Orenstein, D. Pellow, G. Marçais, R. Shamir, and C. Kingsford, “Compact Universal k-mer Hitting Sets,” in Algorithms in Bioinformatics, vol. 9838, no. 10, Cham: Springer, Cham, 2016, pp. 257–268.
- G. Marçais, D. Pellow, D. Bork, Y. Orenstein, R. Shamir, and C. Kingsford, “Improving the performance of minimizers and winnowing schemes,” bioRxiv, p. 104075, Jan. 2017.

Suggested Reading

<u>**Storage/Search**</u>

- **S. Grabowski and M. Raniszewski, “Sampled suffix array with minimizers,” Software: Practice and Experience, vol. 47, no. 11, pp. 1755–1771, Nov. 2017.**
- C.-A. Leimeister, T. Dencker, and B. Morgenstern, “Anchor points for genome alignment based on Filtered Spaced Word Matches.” 26-Mar-2017.
- L. Noé, “Best hits of 11110110111: model-free selection and parameter-free sensitivity calculation of spaced seeds,” Algorithms Mol Biol, vol. 12, no. 1, p. 1, Dec. 2017.
- B. Solomon and C. Kingsford, “Fast search of thousands of short-read sequencing experiments,” Nature Biotechnology, vol. 34, no. 3, pp. 300–302, Mar. 2016.
- G. Holley, R. Wittler, J. Stoye, and F. Hach, “Dynamic Alignment-Free and Reference-Free Read Compression,” in Research in Computational Molecular Biology, vol. 10229, no. 7, S. C. Sahinalp, Ed. Cham: Springer International Publishing, 2017, pp. 50–65.
- B. H. Bloom, “Space/time trade-offs in hash coding with allowable errors,” Commun. ACM, vol. 13, no. 7, pp. 422–426, Jul. 1970.
- S. Grabowski, S. Deorowicz, and Ł. Roguski, “Disk-based compression of data from genome sequencing,” Bioinformatics, vol. 31, no. 9, pp. 1389–1395, May 2015.
- **G. Holley, R. Wittler, and J. Stoye, “Bloom Filter Trie – A Data Structure for Pan-Genome Storage,” in Algorithms in Bioinformatics, vol. 9289, no. 16, Berlin, Heidelberg: Springer, Berlin, Heidelberg, 2015, pp. 217–230.**
- **C. Jain, A. Dilthey, S. Koren, S. Aluru, and A. M. Phillippy, “A Fast Approximate Algorithm for Mapping Long Reads to Large Reference Databases,” in Research in Computational Molecular Biology, vol. 10229, no. 17, S. C. Sahinalp, Ed. Cham: Springer International Publishing, 2017, pp. 66–81.**

Suggested Reading

<u>k-mer Counting</u>

- **S. Deorowicz, M. Kokot, S. Grabowski, and A. Debudaj-Grabysz, "KMC 2: fast and resource-frugal k-mer counting," Bioinformatics, vol. 31, no. 10, pp. 1569–1576, May 2015.**
- M. Erbert, S. Rechner, and M. Müller-Hannemann, "Gerbil: a fast and memory-efficient k -mer counter with GPU-support," Algorithms Mol Biol, vol. 12, no. 1, p. 9, Dec. 2017.
- T. Pan, P. Flick, C. Jain, Y. Liu, and S. Aluru, Kmerind: A Flexible Parallel Library for K-mer Indexing of Biological Sequences on Distributed Memory Systems. New York, New York, USA: ACM, 2016, pp. 422–433.
- G. Marçais and C. Kingsford, "A fast, lock-free approach for efficient parallel counting of occurrences of k-mers," Bioinformatics, vol. 27, no. 6, pp. 764–770, Mar. 2011.
- N. Sivadasan, R. Srinivasan, and K. Goyal, "Kmerlight: fast and accurate k-mer abundance estimation." 19-Sep-2016.
- P. Pandey, M. A. Bender, R. Johnson, and R. Patro, "Squeakr: An Exact and Approximate k-mer Counting System," bioRxiv, p. 122077, Mar. 2017.

Suggested Reading

Phylogeny

- M. Bogusz, S. W. S. biology, 2017, “Phylogenetic tree estimation with and without alignment: new distance methods and benchmarking,” academic.oup.com, p. syw074, Sep. 2016.
- **S. V. Thankachan, S. P. Chockalingam, Y. Liu, A. Krishnan, and S. Aluru, “A greedy alignment-free distance estimator for phylogenetic inference,” BMC Bioinformatics, vol. 18, no. 8, p. 238, Jun. 2017.**

Biological Prediction

- Y. Guo, K. Tian, H. Zeng, X. Guo, and D. K. Gifford, “A novel k-mer set memory (KSM) motif representation improves regulatory variant prediction,” bioRxiv, p. 130815, Apr. 2017.
- F.-D. Pajuste, L. Kaplinski, M. Möls, T. Puurand, M. Lepamets, and M. Remm, “FastGT: an alignment-free method for calling common SNVs directly from raw sequencing reads,” Scientific Reports, vol. 7, no. 1, p. 589, May 2017.

Other

- A. Zielezinski, S. Vinga, J. Almeida, and W. M. Karlowski, “Alignment-free sequence comparison: benefits, applications, and tools,” Genome Biol., vol. 18, no. 1, p. 186, Dec. 2017. (survey)