K-mers and metagenomic sequence classification

Advanced Bioinformatics for NGS

Week 1, Day 3

Abigail Ramsøe

K-mers

S = **A C G T A C G T**

K-mers are substrings of length K that are contained within a String

A sequence of length **L** has **L - k + 1** k-mers and
n^k possible k-mers, where n is the number of possible monomers (1-mers)

K-mers, K = 3

S = **A C G T A C G T**

len(S) = 8

A C G

K-mers, K = 3

S = **A C G T A C G T**

A C G

len(S) = 8

kmer_dict = {}

“ACG” = 1

K-mers, K = 3

S = **A C G T A C G T**

A C G

C G T

len(S) = 8

kmer_dict = {}

“ACG” = 1

“CGT” = 1

K-mers, K = 3

S =	**A**	**C**	**G**	**T**	**A**	**C**	**G**	**T**
	A	C	G					
		C	G	T				
			G	T	A			

len(S) = 8

kmer_dict = {}

“ACG” = 1

“CGT” = 1

“GTA” = 1

K-mers, K = 3

S =	**A**	**C**	**G**	**T**	**A**	**C**	**G**	**T**
	A	C	G					
		C	G	T				
			G	T	A			
				T	A	C		

len(S) = 8

kmer_dict = {}

“ACG” = 1

“CGT” = 1

“GTA” = 1

“TAC” = 1

K-mers, K = 3

S =	**A**	**C**	**G**	**T**	**A**	**C**	**G**	**T**
	A	**C**	**G**					
		C	G	T				
			G	T	A			
				T	A	C		
					A	**C**	**G**	

len(S) = 8

kmer_dict = {}

"ACG" = 1 + 1

"CGT" = 1

"GTA" = 1

"TAC" = 1

K-mers, K = 3

S = **A C G T A C G T**

len(S) = 8

A C G

C G T

G T A

T A C

A C G

C G T

kmer_dict = {}

“ACG” = 1 + 1

“CGT” = 1

“GTA” = 1

“TAC” = 1

“CGT” = 1

K-mers

S = **A C G T A C G T**

A sequence of length **L** has **L - k + 1** k-mers and

L - k + 1 k-mers
8 - 3 + 1 k-mers
6 k-mers

len(S) = 8

kmer_dict = {}

“ACG” = 2

“CGT” = 1

“GTA” = 1

“TAC” = 1

“CGT” = 1

K-mers

S = **A C G T A C G T**

A sequence has n^k possible k-mers, where n is the number of possible monomers (1-mers)

Possible monomers = {A, C, G, T}, len = 4

Possible kmers = 4^3
= 64

len(S) = 8

kmer_dict = {}

“ACG” = 2

“CGT” = 1

“GTA” = 1

“TAC” = 1

“CGT” = 1

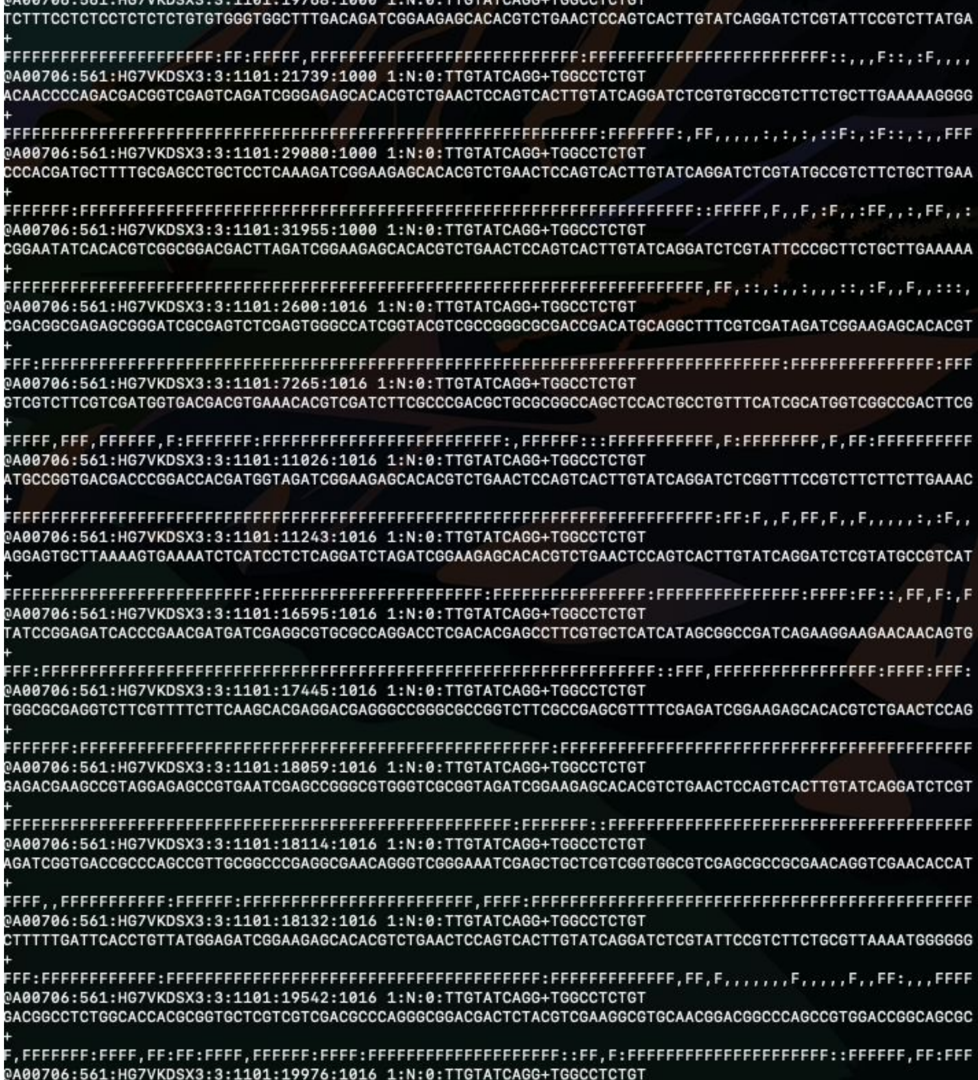
Counting K-mers

Each sequencing run generates ca. 20 BILLION reads

Sequencing errors ALWAYS happen

We can remove these easily using K-mers

If a k-mer has only been seen once, it is likely a sequencing error, and we want to discard it



```
@A00706:561:HG7VKDSX3:3:1101:19768:1000 1:N:0:TTGTATCAGG+TGGCCTCTGT
TCTTTCCTCTCCTCTCTCTGTGTGGGTGGCTTTGACAGATCGGAAGAGCACACGTCTGAACTCCAGTCACTTGTATCAGGATCTCGTATTCCGTCTTATGA
+
FFFFFFFFFFFFFFFFFFFFF:FF:FFFFF,FFFFFFFFFFFFFFFFFFFFFFFFFFFFF:FFFFFFFFFFFFFFFFFFFFFFFFF::,,,F::,:F,,,,
@A00706:561:HG7VKDSX3:3:1101:21739:1000 1:N:0:TTGTATCAGG+TGGCCTCTGT
ACAACCCCAGACGACGGTCGAGTCAGATCGGGAGAGCACACGTCTGAACTCCAGTCACTTGTATCAGGATCTCGTGTGCCGTCTTCTGCTTGAAAAAGGGG
+
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF:FFFFFFF:,FF,,,,,:,:,:,::F:,:F::,:,,FFF
@A00706:561:HG7VKDSX3:3:1101:29080:1000 1:N:0:TTGTATCAGG+TGGCCTCTGT
CCCACGATGCTTTTGCGAGCCTGCTCCTCAAAGATCGGAAGAGCACACGTCTGAACTCCAGTCACTTGTATCAGGATCTCGTATGCCGTCTTCTGCTTGAA
+
FFFFFF:FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF::FFFFF,F,,F,:F,,:FF,,:,FF,,:
@A00706:561:HG7VKDSX3:3:1101:31955:1000 1:N:0:TTGTATCAGG+TGGCCTCTGT
CGGAATATCACACGTCGGCGGACGACTTAGATCGGAAGAGCACACGTCTGAACTCCAGTCACTTGTATCAGGATCTCGTATTCCCGCTTCTGCTTGAAAAA
+
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF,FF,::,:,,:,,,::,:F,,F,,:::,
@A00706:561:HG7VKDSX3:3:1101:2600:1016 1:N:0:TTGTATCAGG+TGGCCTCTGT
CGACGGCGAGAGCGGGATCGCGAGTCTCGAGTGGGCCATCGGTACGTCGCCGGGCGCGACCGACATGCAGGCTTTCGTCGATAGATCGGAAGAGCACACGT
+
FFF:FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF:FFFFFFFFFFFFFF:FFF
@A00706:561:HG7VKDSX3:3:1101:7265:1016 1:N:0:TTGTATCAGG+TGGCCTCTGT
GTCGTCTTCGTCGATGGTGACGACGTGAAACACGTCGATCTTCGCCCGACGCTGCGCGGCCAGCTCCACTGCCTGTTTCATCGCATGGTCGGCCGACTTCG
+
FFFFF,FFF,FFFFFF,F:FFFFFFF:FFFFFFFFFFFFFFFFFFFFFFFFF:,FFFFFF:::FFFFFFFFFF,F:FFFFFFFF,F,FF:FFFFFFFFF
@A00706:561:HG7VKDSX3:3:1101:11026:1016 1:N:0:TTGTATCAGG+TGGCCTCTGT
ATGCCGGTGACGACCCGGACCACGATGGTAGATCGGAAGAGCACACGTCTGAACTCCAGTCACTTGTATCAGGATCTCGGTTTCCGTCTTCTTCTTGAAAC
+
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF:FF:F,,F,FF,F,,F,,,,,:,:F,,
@A00706:561:HG7VKDSX3:3:1101:11243:1016 1:N:0:TTGTATCAGG+TGGCCTCTGT
AGGAGTGCTTAAAAGTGAAAATCTCATCCTCTCAGGATCTAGATCGGAAGAGCACACGTCTGAACTCCAGTCACTTGTATCAGGATCTCGTATGCCGTCAT
+
FFFFFFFFFFFFFFFFFFFFFFFFFF:FFFFFFFFFFFFFFFFFFFFFFFFF:FFFFFFFFFFFFFFFF:FFFFFFFFFFFFFFF:FFFF:FF::,FF,F:,F
@A00706:561:HG7VKDSX3:3:1101:16595:1016 1:N:0:TTGTATCAGG+TGGCCTCTGT
TATCCGGAGATCACCCGAACGATGATCGAGGCGTGCGCCAGGACCTCGACACGAGCCTTCGTGCTCATCATAGCGGCCGATCAGAAGGAAGAACAACAGTG
+
FFF:FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF::FFF,FFFFFFFFFFFFFFF:FFFF:FFF:
@A00706:561:HG7VKDSX3:3:1101:17445:1016 1:N:0:TTGTATCAGG+TGGCCTCTGT
TGGCGCGAGGTCTTCGTTTTCTTCAAGCACGAGGACGAGGGCCGGGCGCCGGTCTTCGCCGAGCGTTTTCGAGATCGGAAGAGCACACGTCTGAACTCCAG
+
FFFFFF:FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF:FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
@A00706:561:HG7VKDSX3:3:1101:18059:1016 1:N:0:TTGTATCAGG+TGGCCTCTGT
GAGACGAAGCCGTAGGAGAGCCGTGAATCGAGCCGGGCGTGGGTCGCGGTAGATCGGAAGAGCACACGTCTGAACTCCAGTCACTTGTATCAGGATCTCGT
+
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF:FFFFFFF::FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
@A00706:561:HG7VKDSX3:3:1101:18114:1016 1:N:0:TTGTATCAGG+TGGCCTCTGT
AGATCGGTGACCGCCCAGCCGTTGCGGCCCGAGGCGAACAGGGTCGGGAAATCGAGCTGCTCGTCGGTGGCGTCGAGCGCCGCGAACAGGTCGAACACCAT
+
FFFF,,FFFFFFFFFFF:FFFFFF:FFFFFFFFFFFFFFFFFFFFFFFF,FFFF:FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
@A00706:561:HG7VKDSX3:3:1101:18132:1016 1:N:0:TTGTATCAGG+TGGCCTCTGT
CTTTTTGATTCACCTGTTATGGAGATCGGAAGAGCACACGTCTGAACTCCAGTCACTTGTATCAGGATCTCGTATTCCGTCTTCTGCGTTAAAATGGGGGG
+
FFF:FFFFFFFFFFFF:FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF:FFFFFFFFFFFF,FF,F,,,,,,,F,,,,,F,,FF:,,,FFFF
@A00706:561:HG7VKDSX3:3:1101:19542:1016 1:N:0:TTGTATCAGG+TGGCCTCTGT
GACGGCCTCTGGCACCACGCGGTGCTCGTCGTCGACGCCCAGGGCGGACGACTCTACGTCGAAGGCGTGCAACGGACGGCCCAGCCGTGGACCGGCAGCGC
+
F,FFFFFFF:FFFF,FF:FF:FFFF,FFFFFF:FFFF:FFFFFFFFFFFFFFFFFFFFF::FF,F:FFFFFFFFFFFFFFFFFFFFF::FFFFFF,FF:FFF
@A00706:561:HG7VKDSX3:3:1101:19976:1016 1:N:0:TTGTATCAGG+TGGCCTCTGT
```

Counting K-mers

If a k-mer has only been seen once, it is likely a sequencing error, and we want to discard it

We could create a dictionary of number of occurrences

1. Iterate through all k-mers
2. Increment counter
3. Iterate through all counts and find count == 1

But this is two iterations over a large dataset!

```
S = A C G T A C G T

    A C G
      C G T
        G T A
          T A C
            A C G
              C G T
```

kmer_dict = {}

“ACG” = 2

“CGT” = 1

“GTA” = 1

“TAC” = 1

“CGT” = 1

Bloom filters

Set of independent hash functions that map k-mers to values

For each k-mer, we call each hash function

Have we seen this k-mer before?

Hash_function_1

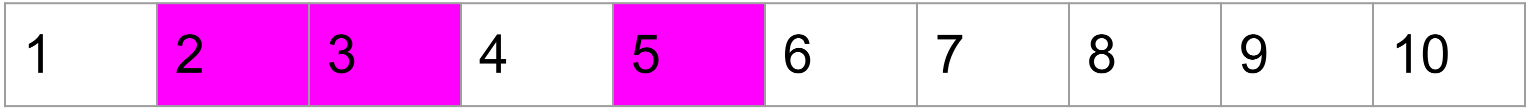
Hash_function_2

Hash_function_3

Bloom filters - loop through our k-mers

		hash_function_1	hash_function_2	hash_function_3
k-mer 1	ADDED	5	2	3

Array of bits



1	2	3	4	5	6	7	8	9	10

Bloom filters - loop through our k-mers

		hash_function_1	hash_function_2	hash_function_3
k-mer 1	ADDED	5	2	3
k-mer 2	ADDED	4	6	8

1	2	3	4	5	6	7	8	9	10

Bloom filters - search for k-mers

		hash_function_1	hash_function_2	hash_function_3
k-mer 1	ADDED	5	2	3
k-mer 2	ADDED	4	6	8
k-mer 1	searching…	5	2	3

Array of bits

1	2	3	4	5	6	7	8	9	10

Bloom filters - search for k-mers

		hash_function_1	hash_function_2	hash_function_3
k-mer 1	ADDED	5	2	3
k-mer 2	ADDED	4	6	8
k-mer 1	TRUE POSITIVE	5	2	3

True positive!

Array of bits

1	2	3	4	5	6	7	8	9	10
	🟪	🟪	🟦	🟪	🟦		🟦		

Bloom filters - search for k-mers

		hash_function_1	hash_function_2	hash_function_3
k-mer 1	ADDED	5	2	3
k-mer 2	ADDED	4	6	8
k-mer 1	TRUE POSITIVE	5	2	3
k-mer 4	searching…	7	1	2

Array of bits

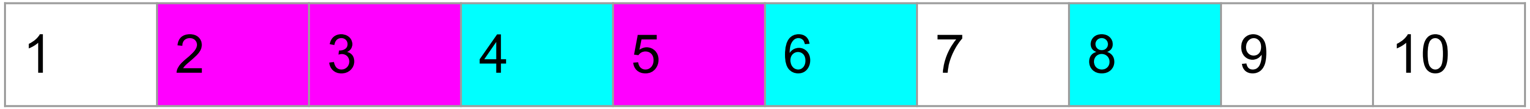
1	2 🟪	3 🟪	4 🟦	5 🟪	6 🟦	7	8 🟦	9	10

Bloom filters - search for k-mers

		hash_function_1	hash_function_2	hash_function_3
k-mer 1	ADDED	5	2	3
k-mer 2	ADDED	4	6	8
k-mer 1	TRUE POSITIVE	5	2	3
k-mer 4	TRUE NEGATIVE	7	1	2

True negative! - only need to check first hash function

Array of bits

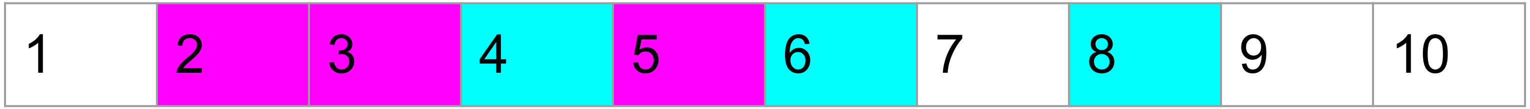


1	2	3	4	5	6	7	8	9	10

Bloom filters - search for k-mers

		hash_function_1	hash_function_2	hash_function_3
k-mer 1	ADDED	5	2	3
k-mer 2	ADDED	4	6	8
k-mer 1	TRUE POSITIVE	5	2	3
k-mer 4	TRUE NEGATIVE	7	1	2
k-mer 5	searching…	2	3	5

Array of bits

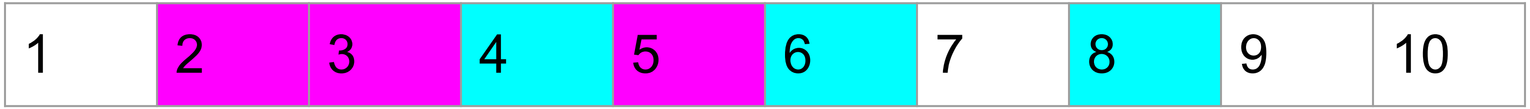


1	2	3	4	5	6	7	8	9	10

Bloom filters - search for k-mers

		hash_function_1	hash_function_2	hash_function_3
k-mer 1	ADDED	5	2	3
k-mer 2	ADDED	4	6	8
k-mer 1	TRUE POSITIVE	5	2	3
k-mer 4	TRUE NEGATIVE	7	1	2
k-mer 5	FALSE POSITIVE	2	3	5

Array of bits

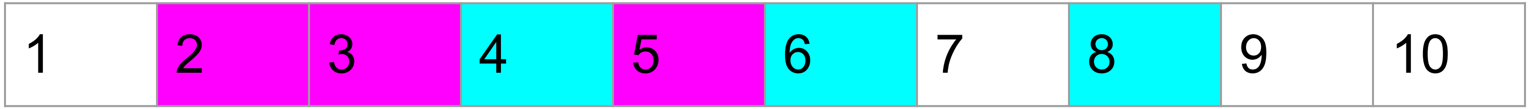


1	2	3	4	5	6	7	8	9	10

Bloom filters - how to discard k-mers

1. Loop though all k-mers
2. Is this k-mer in our bloom filter?
   a. NO - store in filter
   b. YES - increment count

3. Remove k-mers that are in the filter, but have no count



1	2 🟪	3 🟪	4 🟦	5 🟪	6 🟦	7	8 🟦	9	10

Bloom filters

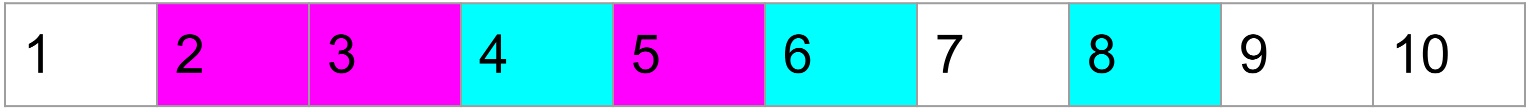
Can false negatives ever occur?

No

What are the factors that reduce false positives?

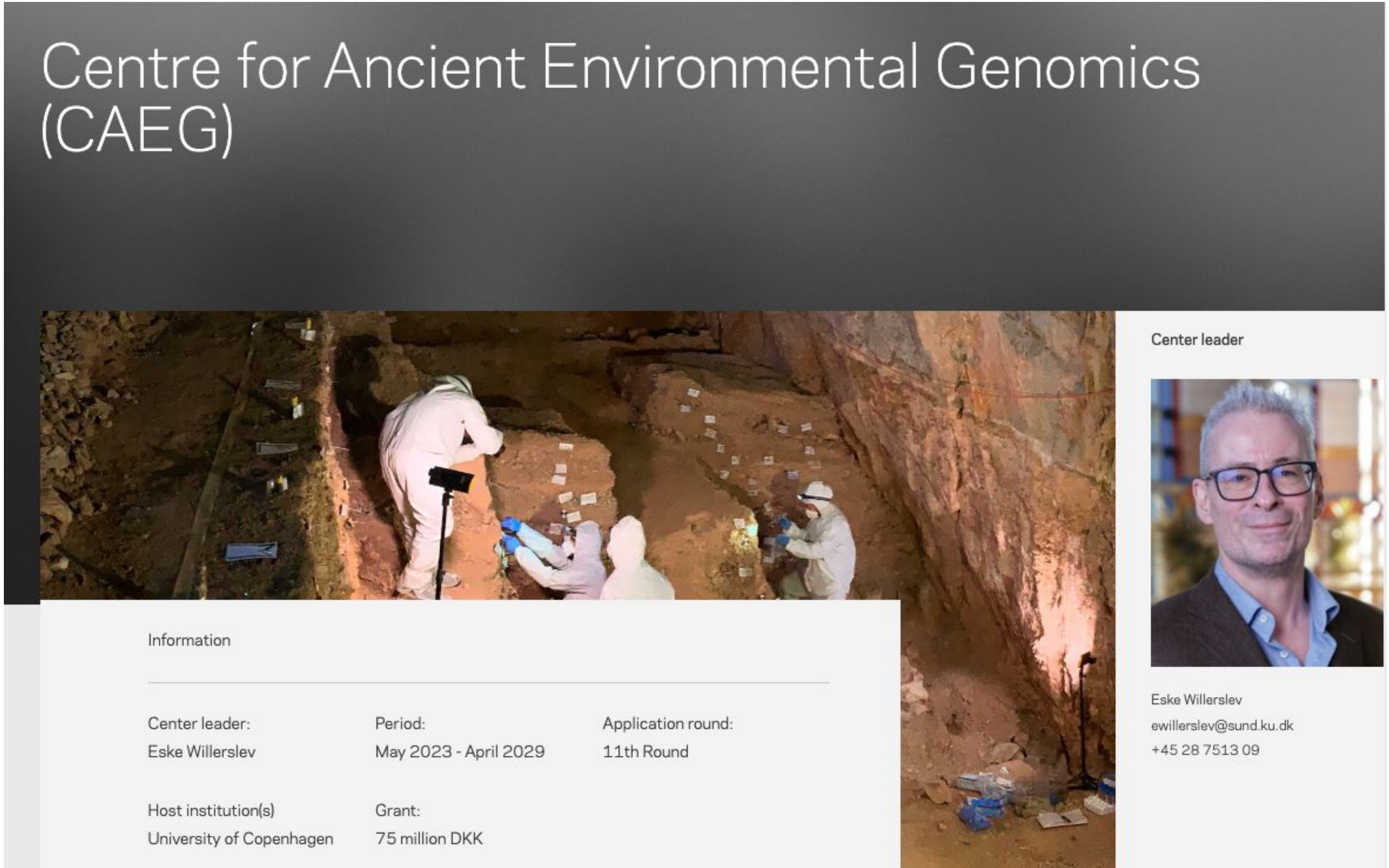
Higher bits

Array of bits



1	2	3	4	5	6	7	8	9	10

K-mers for metagenomics



Centre for Ancient Environmental Genomics (CAEG)

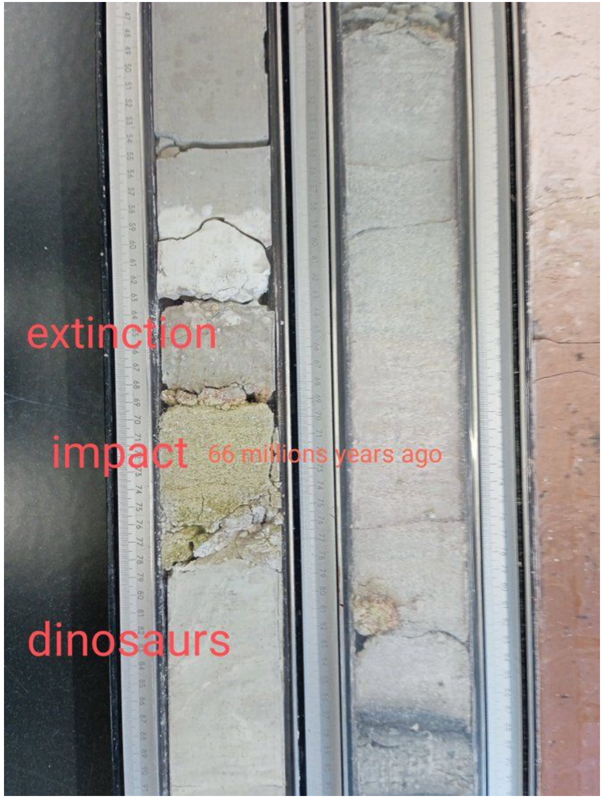
Center leader

Eske Willerslev
ewillerslev@sund.ku.dk
+45 28 7513 09

Information

Center leader:	Period:	Application round:
Eske Willerslev	May 2023 - April 2029	11th Round
Host institution(s)	Grant:	
University of Copenhagen	75 million DKK	

K-mers for metagenomics



Metagenomic sequence classification

But first, **lowest common ancestor (LCA)**

What is the lowest common ancestor of cat and tiger?

They belong to different species, so we check the genus level

Species

Tiger
(*Panthera tigris*)

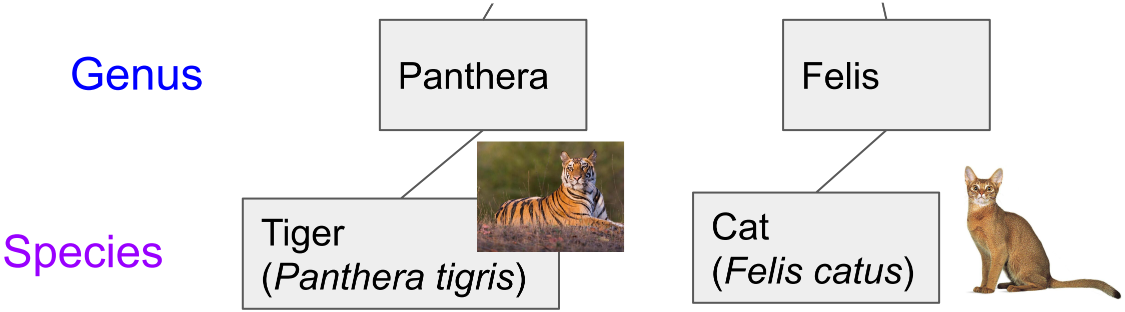
Cat
(*Felis catus*)

Metagenomic sequence classification

But first, **lowest common ancestor (LCA)**

What is the lowest common ancestor of cat and tiger?

They also belong to different genus



Genus

Panthera

Felis

Species

Tiger
(*Panthera tigris*)

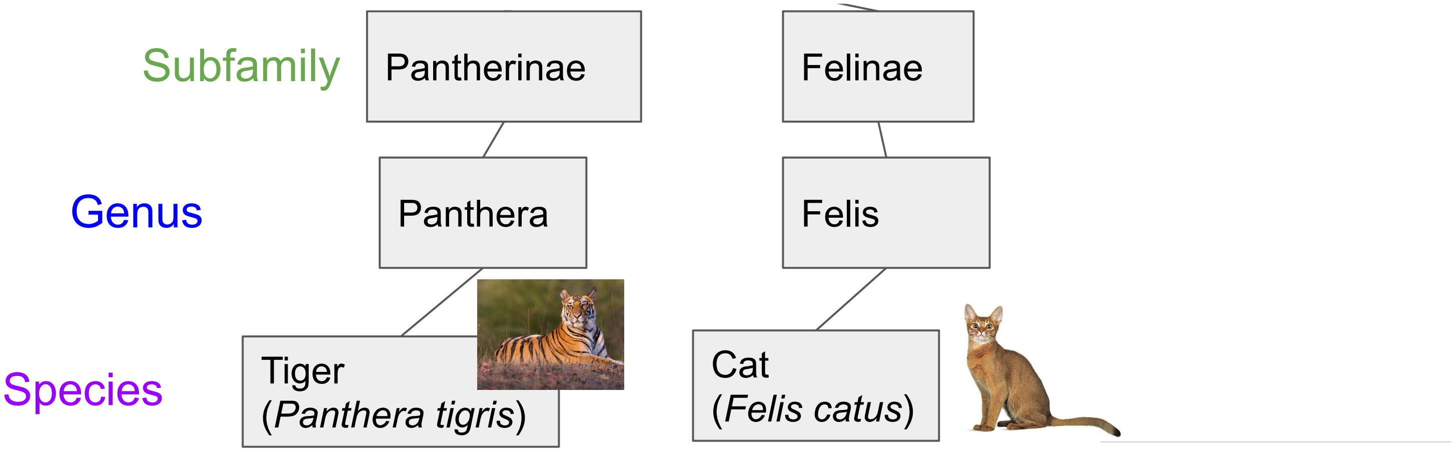
Cat
(*Felis catus*)

Metagenomic sequence classification

But first, **lowest common ancestor (LCA)**

What is the lowest common ancestor of cat and tiger?

They also belong to different subfamilies



Subfamily

Pantherinae

Felinae

Genus

Panthera

Felis

Species

Tiger
(*Panthera tigris*)

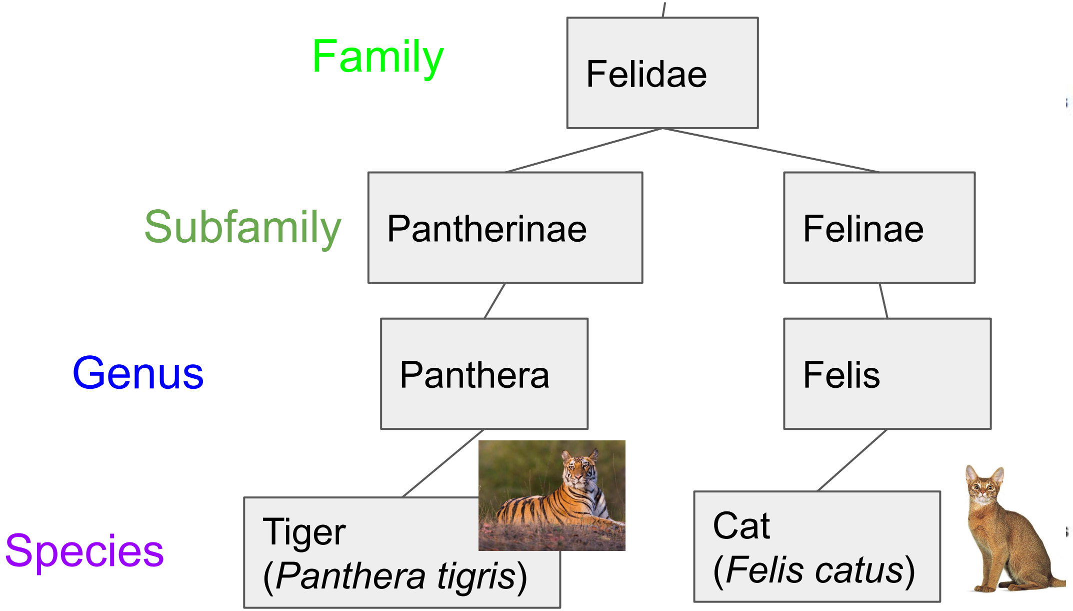
Cat
(*Felis catus*)

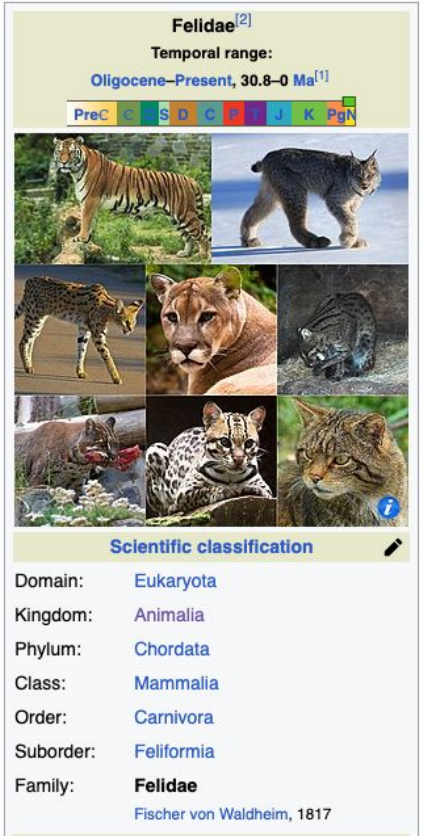
Metagenomic sequence classification

But first, **lowest common ancestor (LCA)**

What is the lowest common ancestor of cat and tiger?

They belong to the same family - Felidae





Felidae[2]

Temporal range:

Oligocene–Present, 30.8–0 Ma[1]

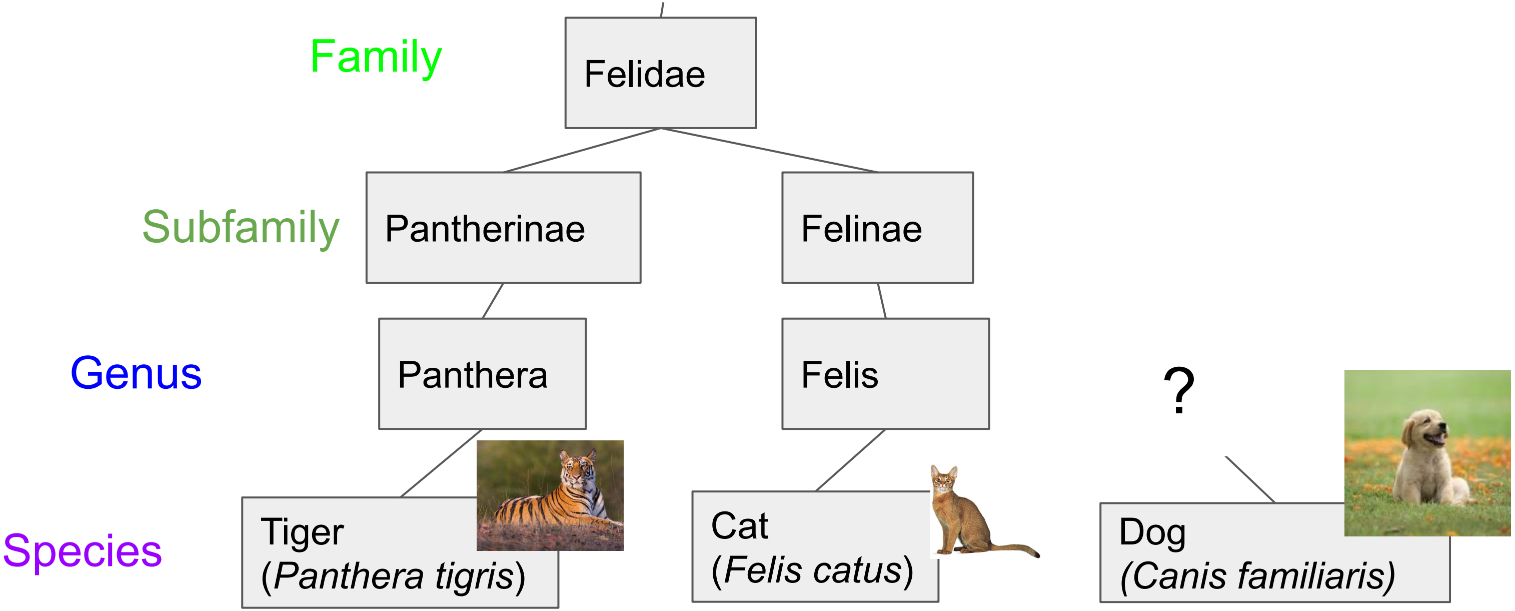
Scientific classification

Domain:	Eukaryota
Kingdom:	Animalia
Phylum:	Chordata
Class:	Mammalia
Order:	Carnivora
Suborder:	Feliformia
Family:	**Felidae** Fischer von Waldheim, 1817

Metagenomic sequence classification

But first, **lowest common ancestor (LCA)**

What about dogs? Is the LCA lower (more specific), or higher (less specific) than that of cat and tiger?



Family

Felidae

Subfamily

Pantherinae

Felinae

Genus

Panthera

Felis

?

Species

Tiger
(Panthera tigris)

Cat
(Felis catus)

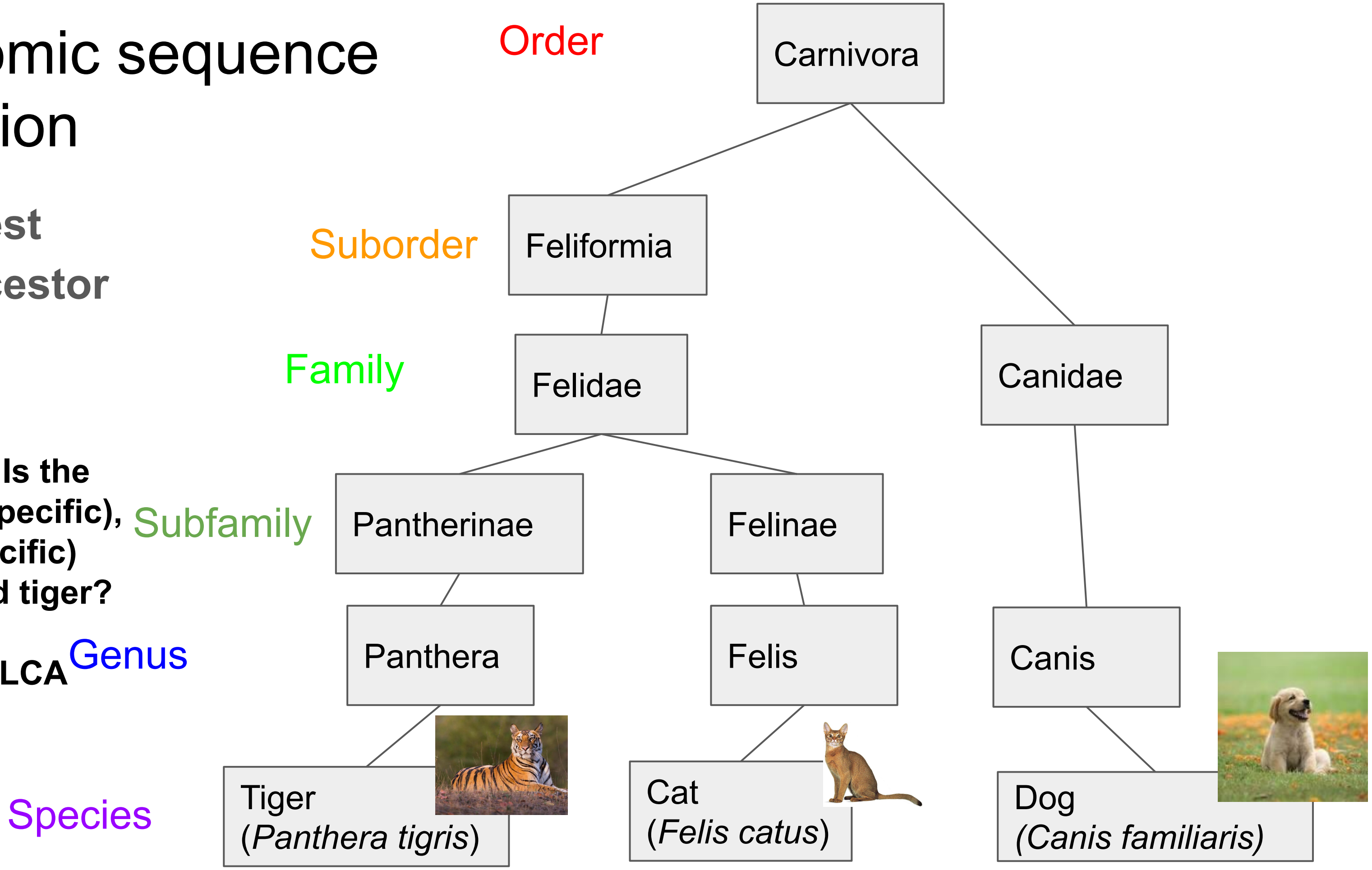
Dog
(Canis familiaris)

Metagenomic sequence classification

But first, **lowest common ancestor (LCA)**

What about dogs? Is the LCA lower (more specific), or higher (less specific) than that of cat and tiger?

Carnivora - higher LCA



Order

Carnivora

Suborder

Feliformia

Family

Felidae

Canidae

Subfamily

Pantherinae

Felinae

Genus

Panthera

Felis

Canis

Species

Tiger
(*Panthera tigris*)

Cat
(*Felis catus*)

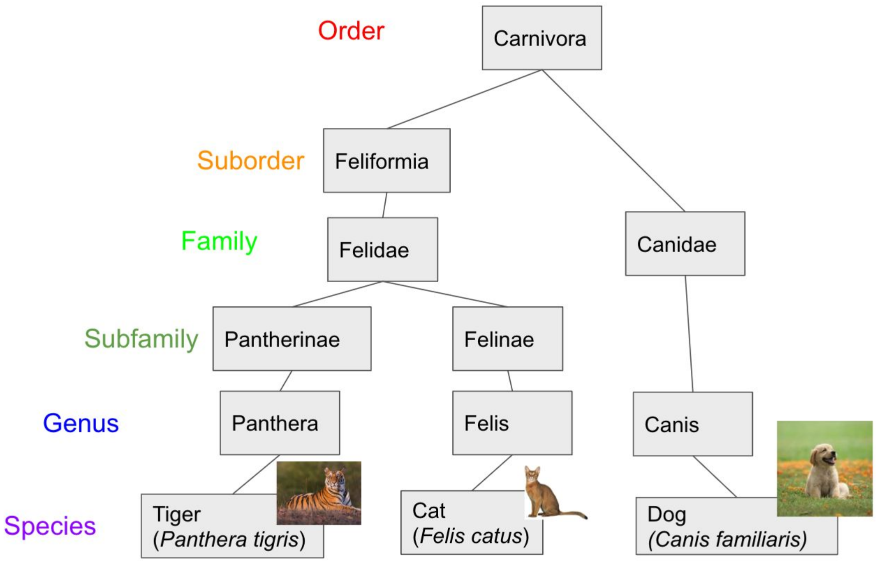
Dog
(*Canis familiaris*)

Metagenomic sequence classification - brute force

You could simply map each read to ALL genomes of interest.

Then from there, figure out the LCA for each read

E.g. if any reads map to the tiger reference genome, but NOT the cat reference, that read likely comes from an animal in the Pantherinae subfamily



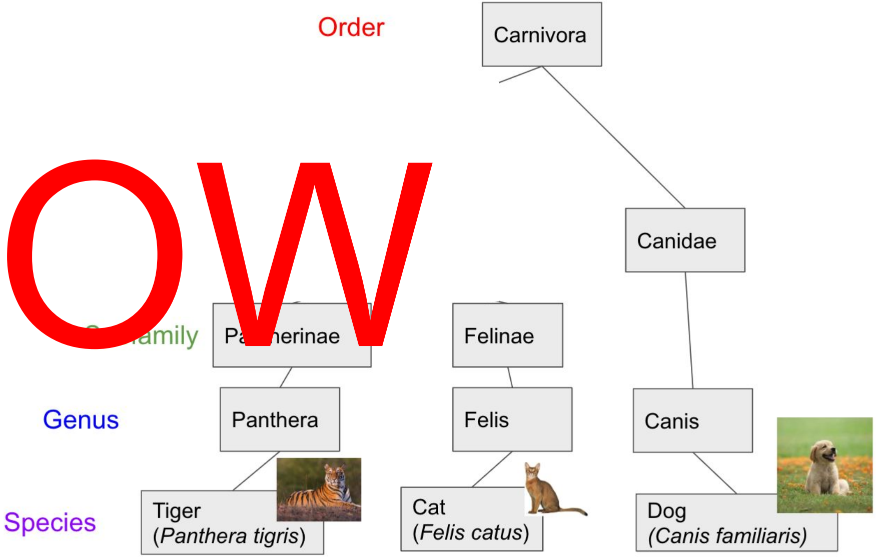
Metagenomic sequence classification - brute force

You could simply map each read to ALL genomes of interest.

Then from there, figure out for each read

SLOW

E.g. if any reads map to the tiger reference genome, but NOT the cat reference, that read likely comes from an animal in the Pantherinae subfamily



Order

Carnivora

Canidae

amily

Pa herinae

Felinae

Genus

Panthera

Felis

Canis

Species

Tiger
(*Panthera tigris*)

Cat
(*Felis catus*)

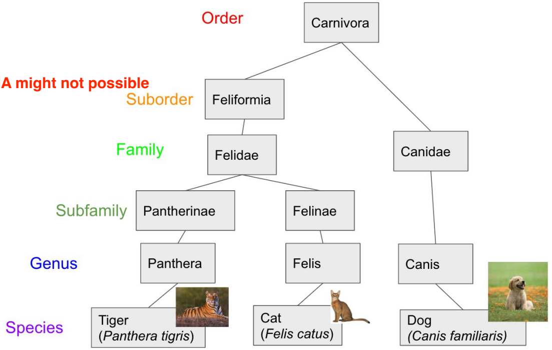
Dog
(*Canis familiaris*)

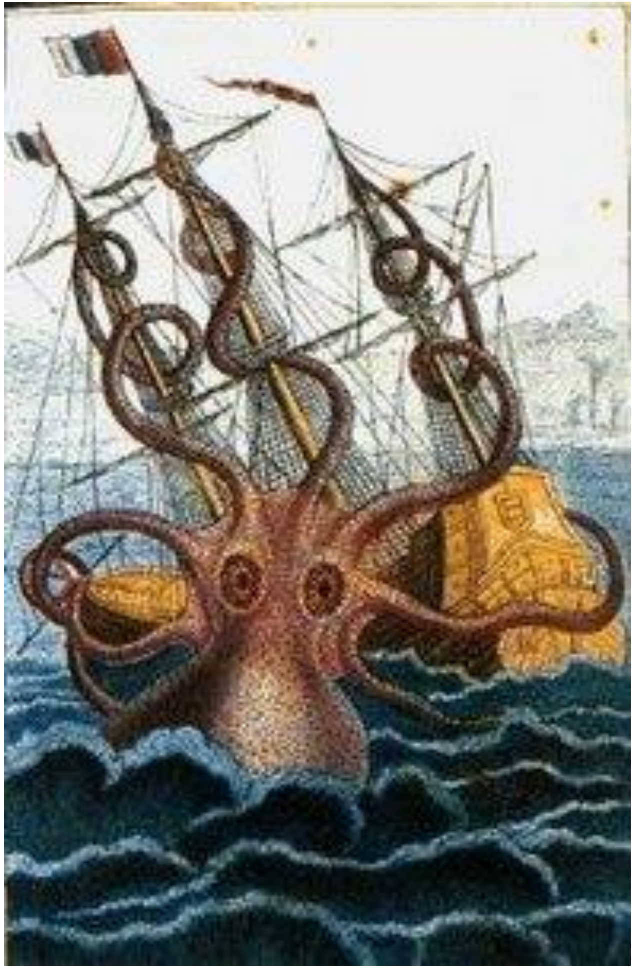
Kraken - metagenomic sequence classification

Kraken has a database of k-mers along with their lowest common ancestor

The default K is 31 - why ?

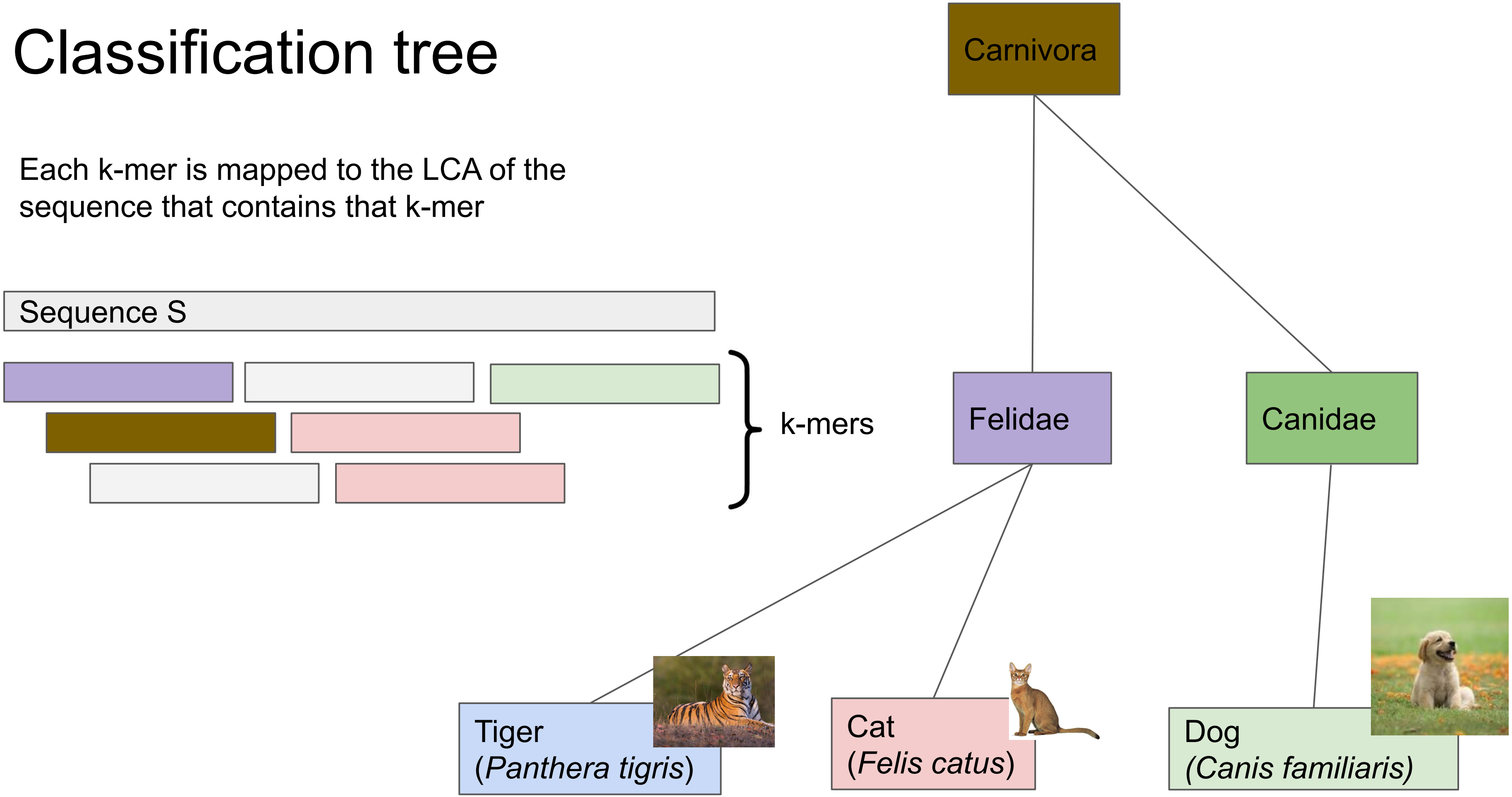
31 is good enough to map into all species. Higher value for aDNA might not possible due to its damage which affects fragment size





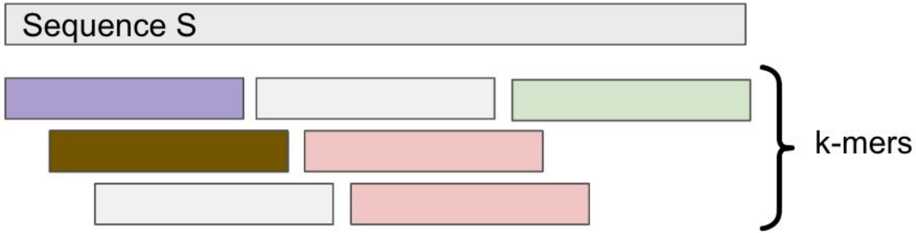
Classification tree

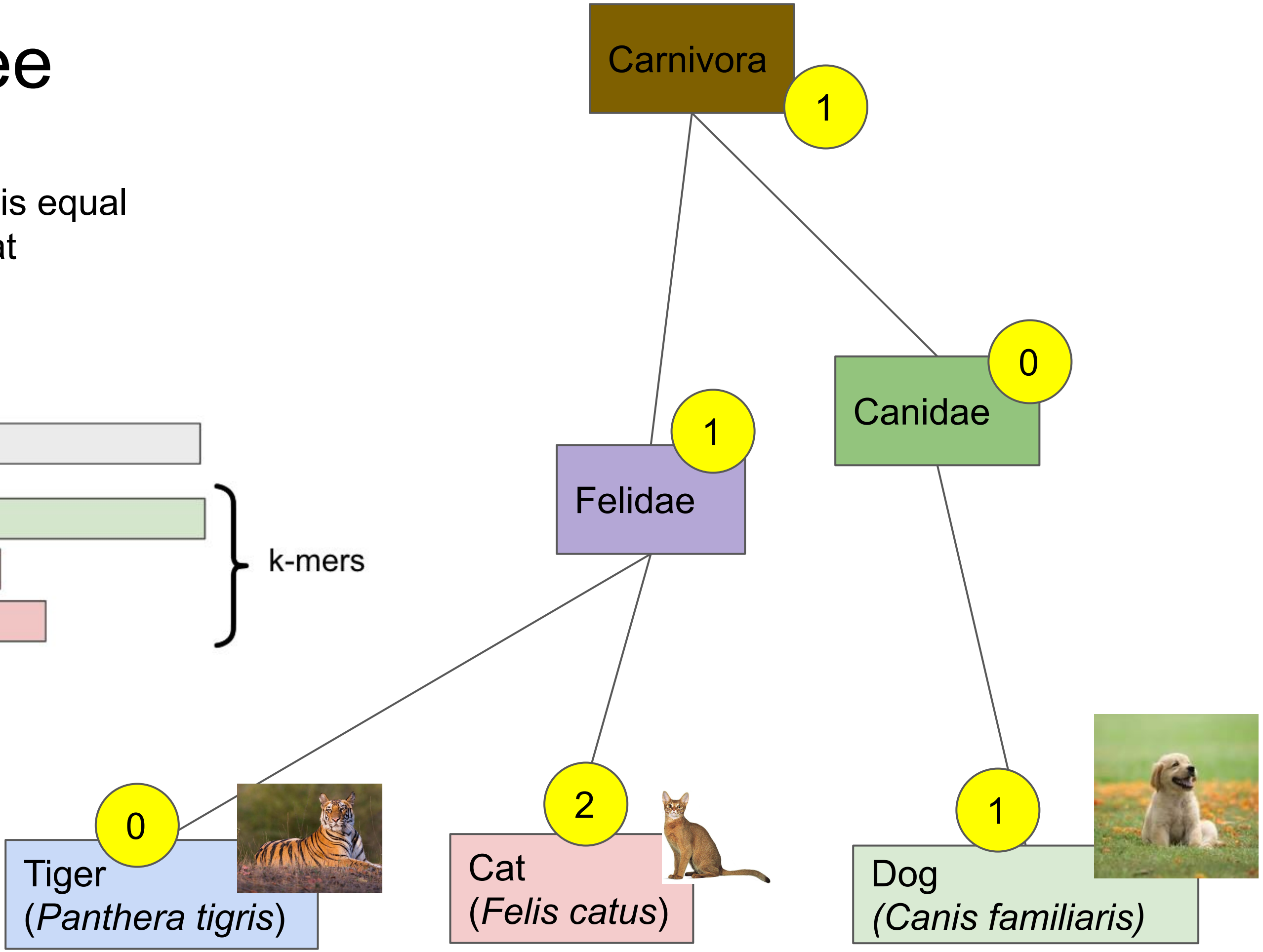
Each k-mer is mapped to the LCA of the sequence that contains that k-mer



Classification tree

Each node gets a weight X that is equal to the number of k-mers in S that classifies to the node’s taxon





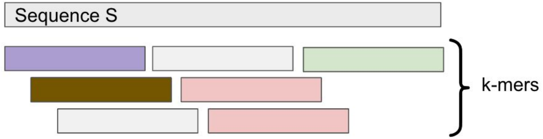
Classification tree

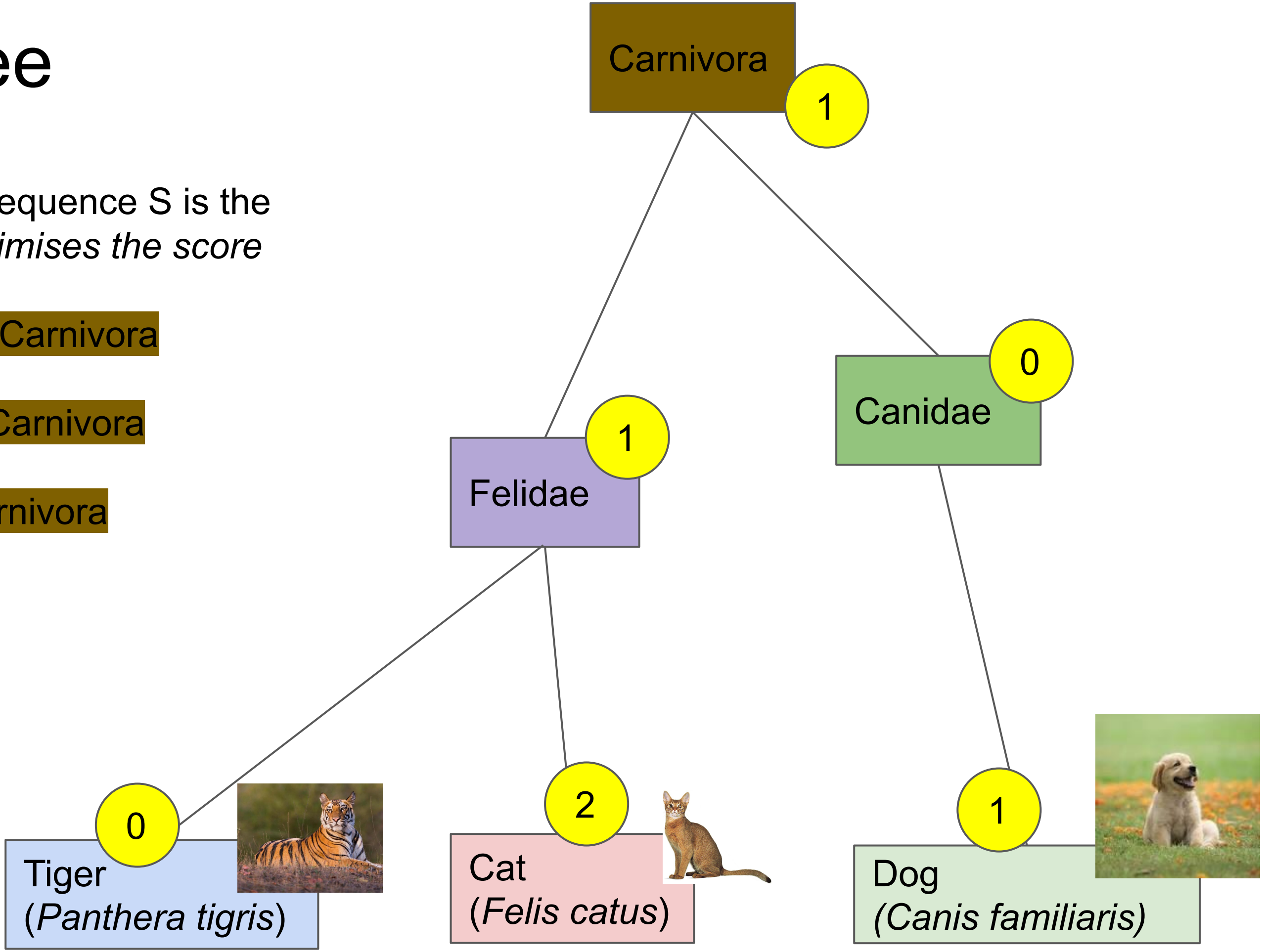
The classification used for the sequence S is the root-to-leaf (RTL) path that *maximises the score*

Tiger Score = Tiger + Felidae + Carnivora
= 0 + 1 + 1 = **2**

Dog Score = Dog + Canidae + Carnivora
= 1 + 0 + 1 = **2**

Cat Score = Cat + Felidae + Carnivora
= 2 + 1 + 1 = **4**



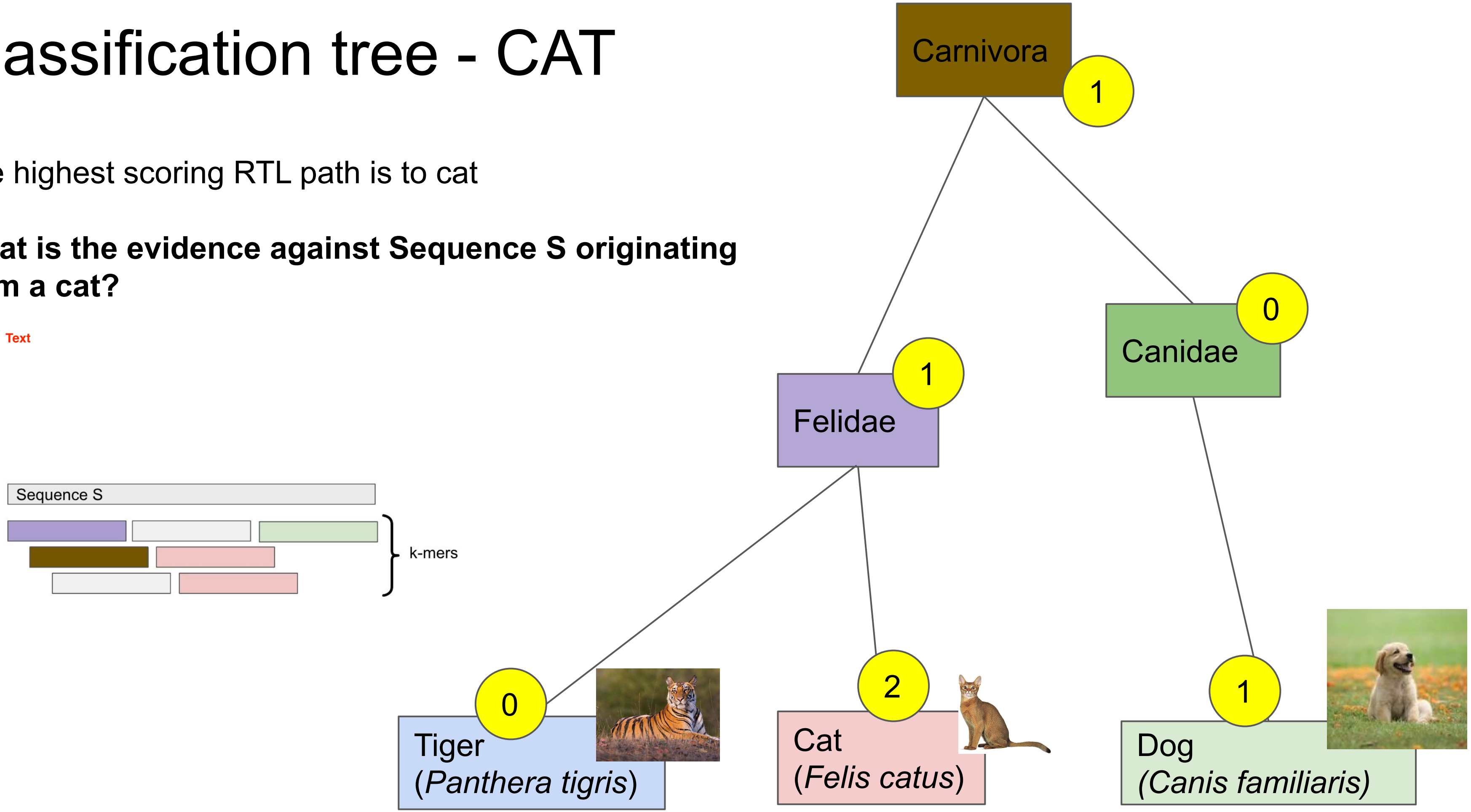


Classification tree - CAT

The highest scoring RTL path is to cat

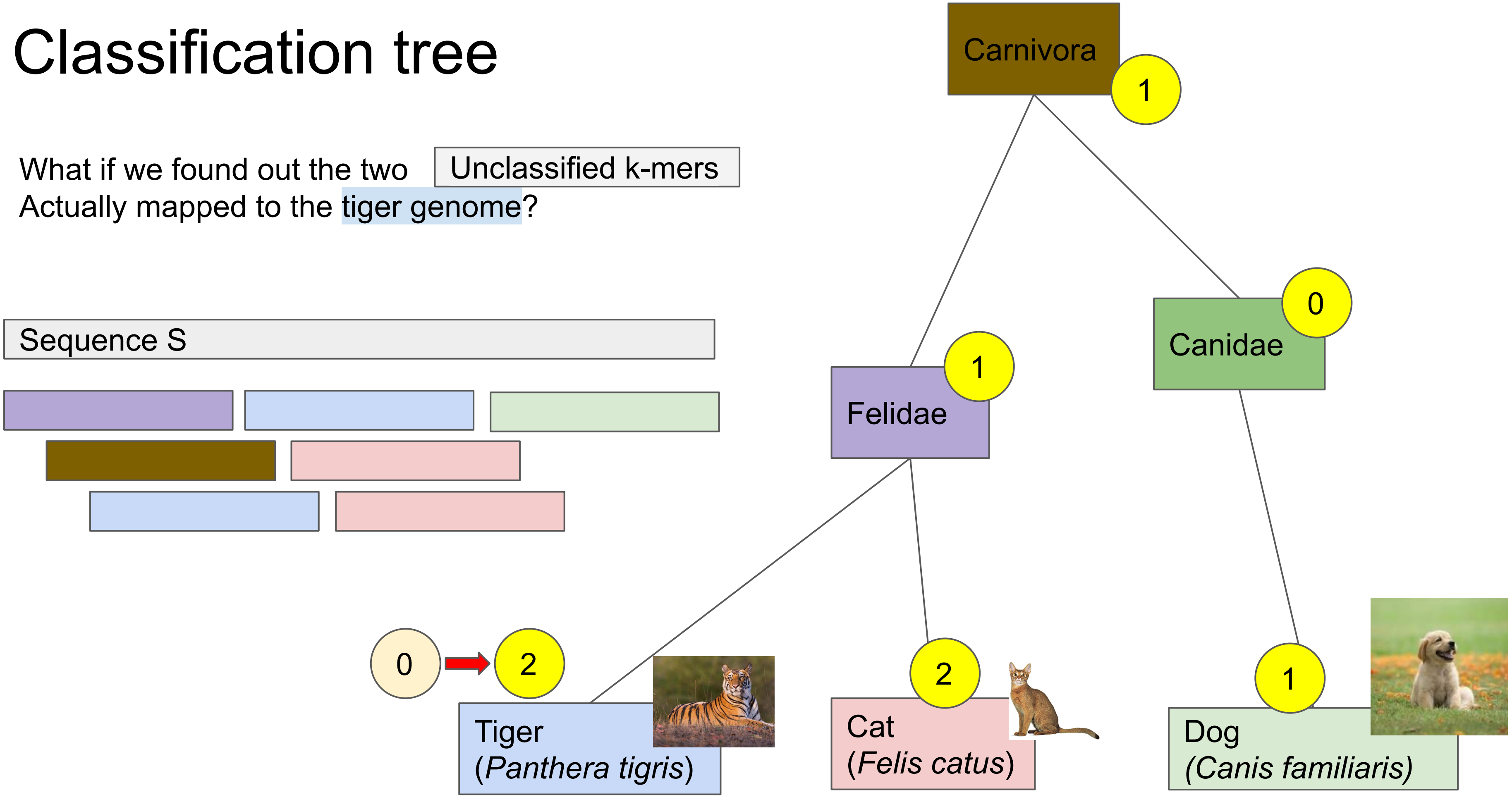
What is the evidence against Sequence S originating from a cat?

Text



Classification tree

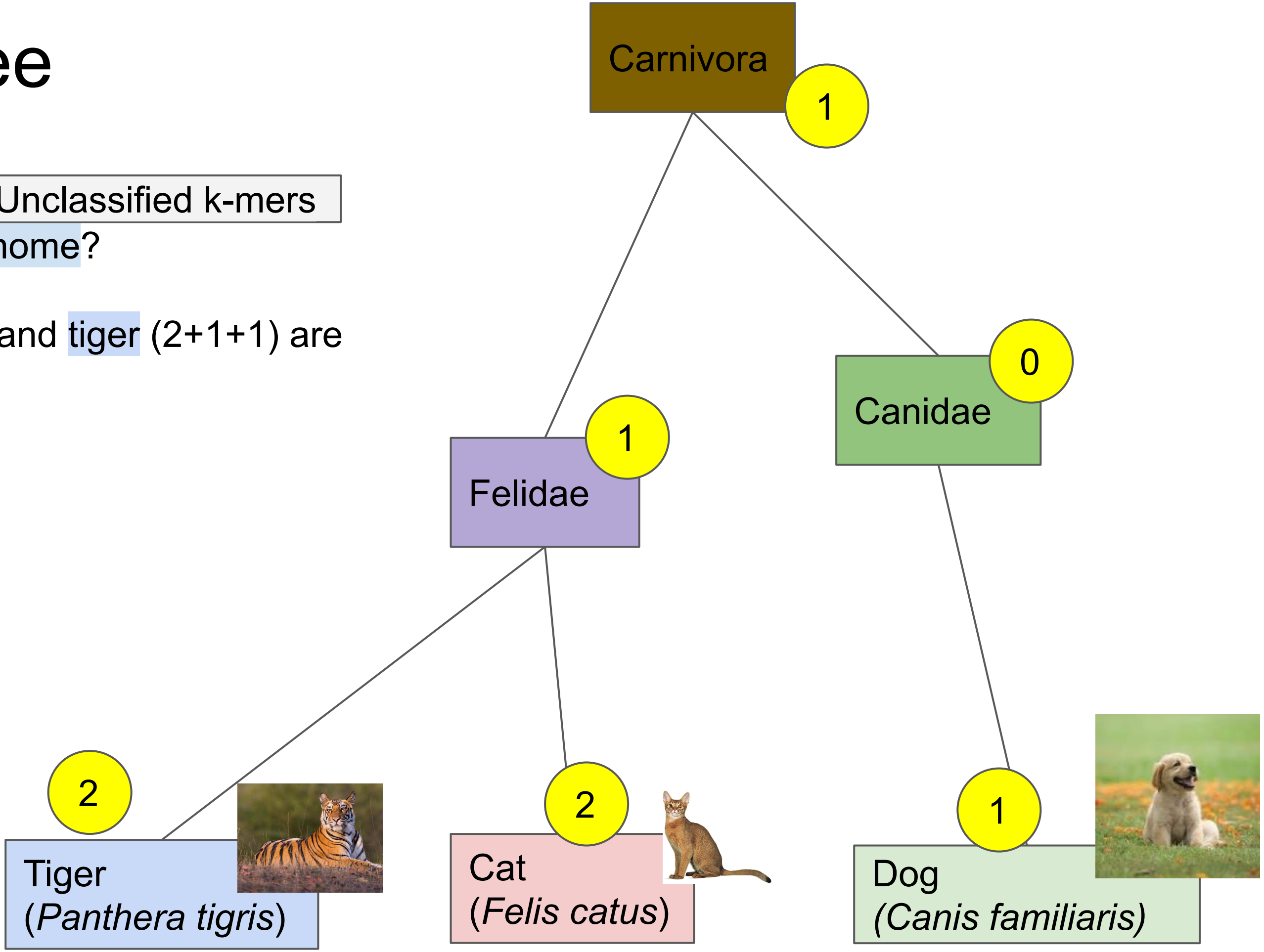
What if we found out the two Unclassified k-mers Actually mapped to the tiger genome?

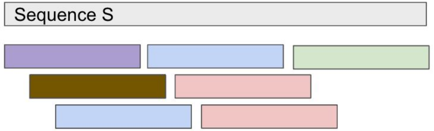


Classification tree

What if we found out the two Unclassified k-mers
Actually mapped to the tiger genome?

Now the scores for cat (2+1+1) and tiger (2+1+1) are equal



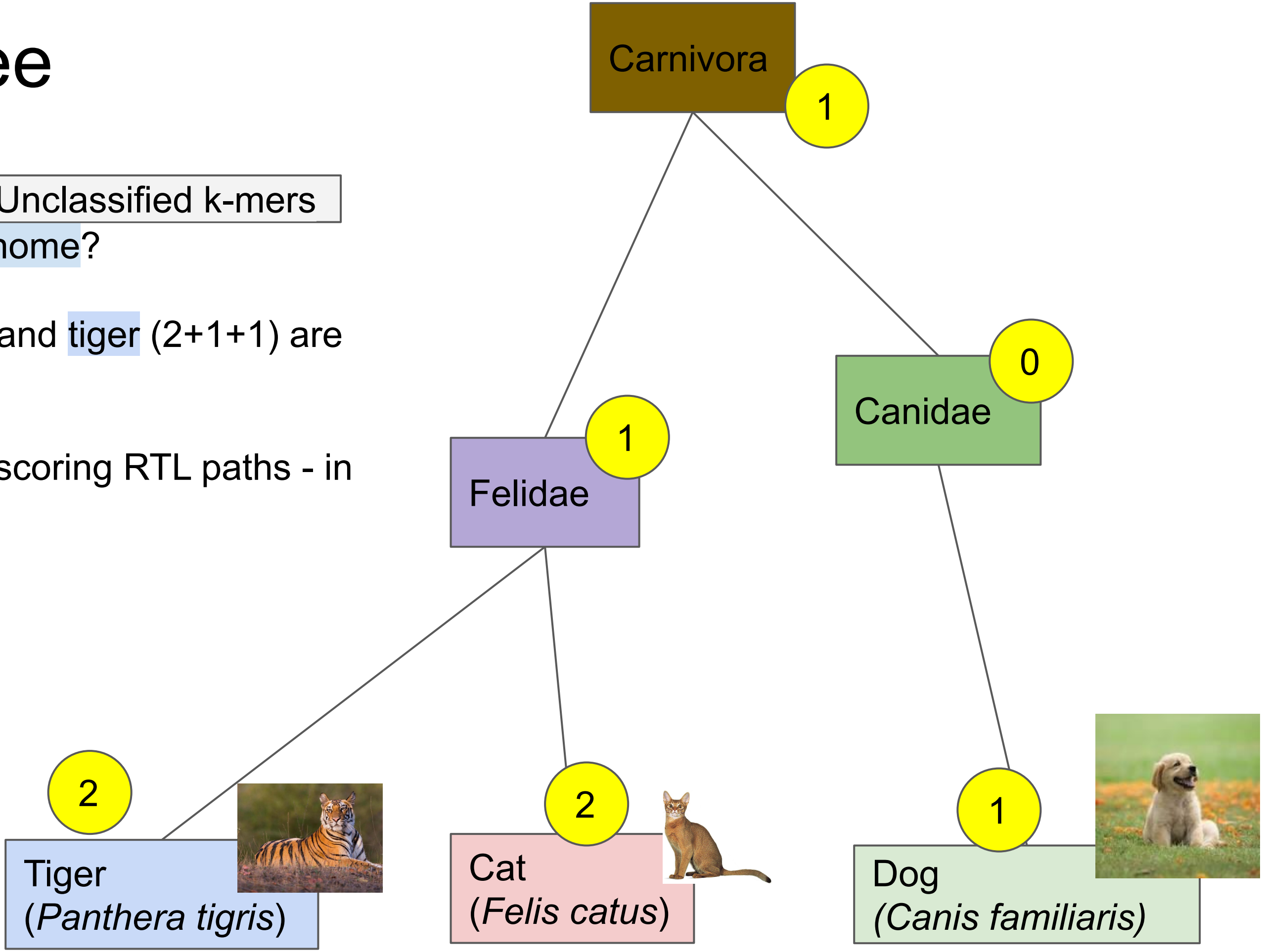


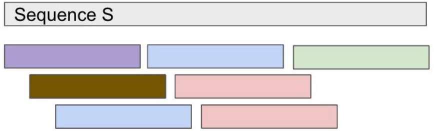
Classification tree

What if we found out the two Unclassified k-mers
Actually mapped to the tiger genome?

Now the scores for cat (2+1+1) and tiger (2+1+1) are equal

Use the LCA of the two equally scoring RTL paths - in this case Felidae



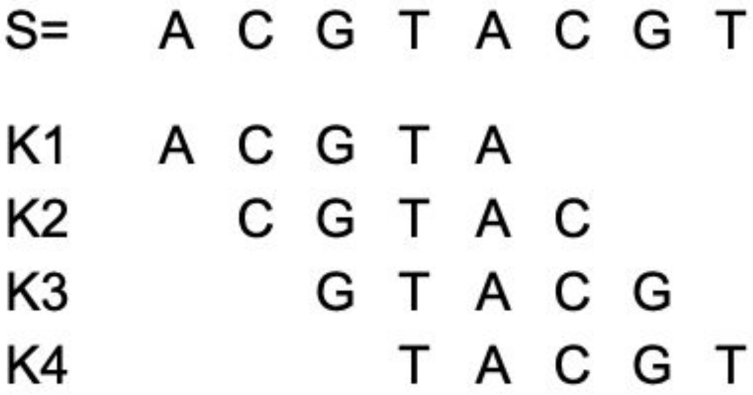


Minimizers

K-mers that are adjacent to each other are very similar, so we waste time looking them up

Kraken uses minimizers to optimise cache usage

len(S) = 8
K = 5



S=	A	C	G	T	A	C	G	T
K1	A	C	G	T	A			
K2		C	G	T	A	C		
K3			G	T	A	C	G	
K4				T	A	C	G	T

Minimizers

M-mers are substrings of k-mers of length M,
Where M < K

The minimizer of a k-mer is the first M-mer, if they all are arranged in alphabetical order (i.e. the *lexicographically smallest* m-mer)

Thus, the minimizer of K1 (k-mer 1) is A C G

S=	A	C	G	T	A	C	G	T
K1	A	C	G	T	A			
K2		C	G	T	A	C		
K3			G	T	A	C	G	
K4				T	A	C	G	T

len(S) = 8
K = 5
M = 3

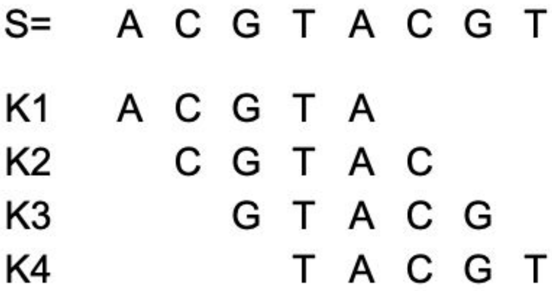
K1	A	C	G	T	A
M1	A	C	G		
M2		C	G	A	
M3			G	A	T

Minimizers

Compute the minimizers (M=3) for the rest of the K-mers

For each K-mer

1. Find all M-mers
2. Sort the M-mers alphabetically
3. Find the first one - this is the minimizer

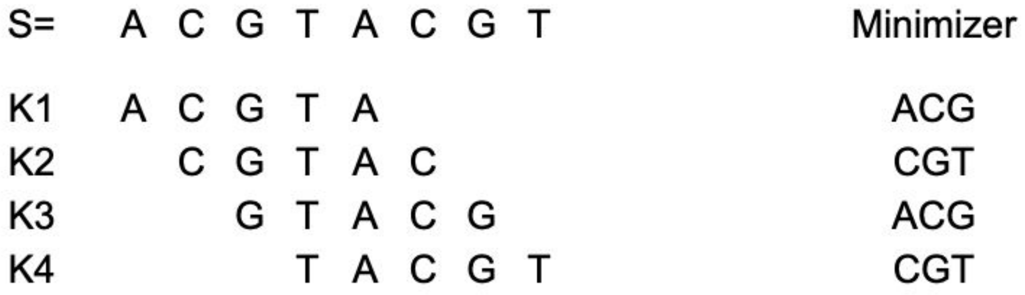


S=	A	C	G	T	A	C	G	T
K1	A	C	G	T	A			
K2		C	G	T	A	C		
K3			G	T	A	C	G	
K4				T	A	C	G	T

Minimizers

Compute the minimizers (M=3) for the rest of the K-mers

Are the minimizers for K1..4 similar?



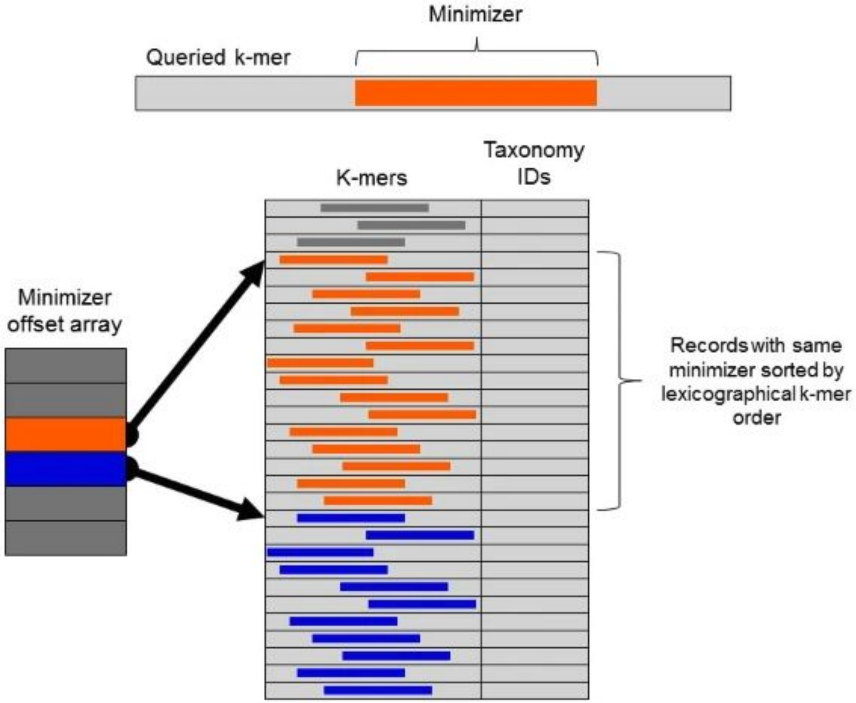
S=	A	C	G	T	A	C	G	T	Minimizer
K1	A	C	G	T	A				ACG
K2		C	G	T	A	C			CGT
K3			G	T	A	C	G		ACG
K4				T	A	C	G	T	CGT

Database structure and search

Kraken stores k-mers with the same minimizer adjacent to each other

This means that when one k-mer with a certain minimizer is queried, the rest are *loaded into CPU cache*

Because adjacent k-mers are likely to have the same minimizer, this speeds up computation

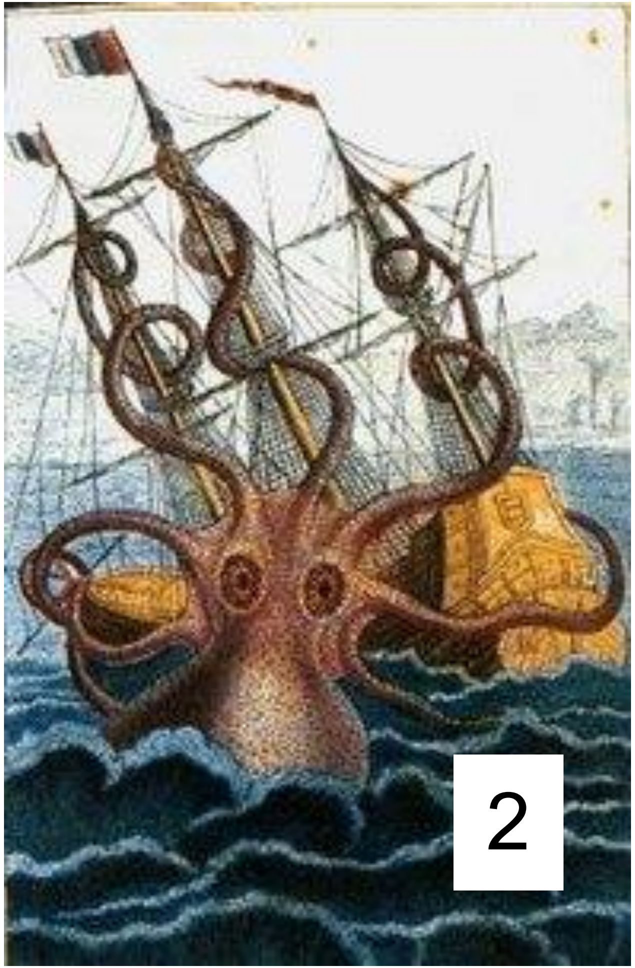

Minimizer
Queried k-mer
Taxonomy IDs
K-mers
Minimizer offset array
Records with same minimizer sorted by lexicographical k-mer order

Kraken 2 - 85% faster than Kraken 1 and 100% more complicated to explain

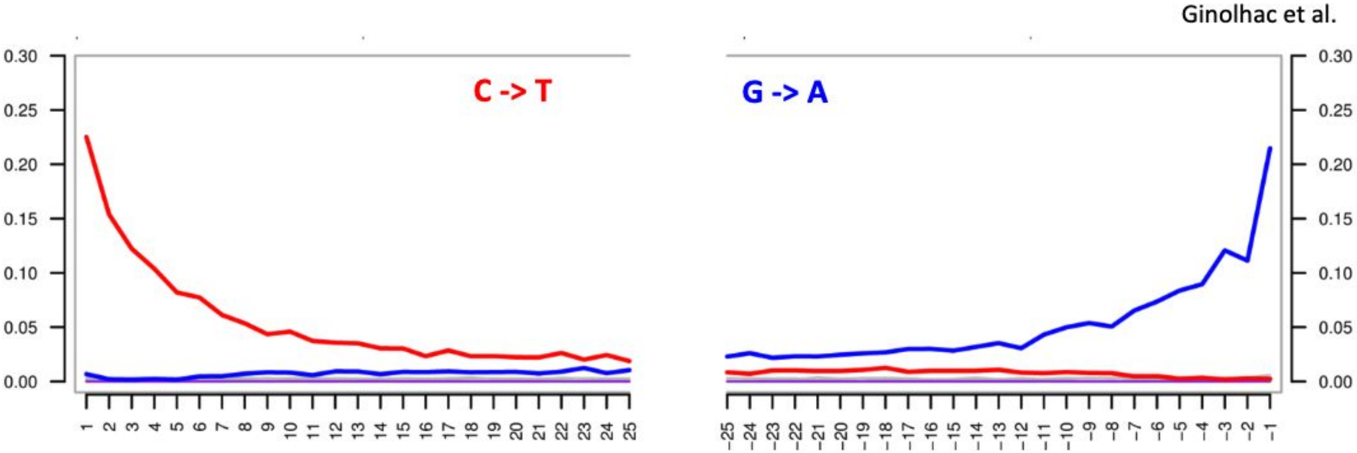
Kraken 1 uses a sorted list indexed by minimizers to store k-mers

Kraken 2 uses a compact hash table (faster, less memory intensive, a bit less accurate)

K2 only stores (big) minimizers, whereas K1 stored (big) k-mers and used (smaller) minimizers



What about ancient DNA?



Sequencing ancient cats

Take the sequencing read

S= T T A A A A A

Cat Reference Genome

A A C C A A A G G A A

Break into k-mers (6-mers)

A A C C A A

A C C A A A

C C A A A G

C A A A G G

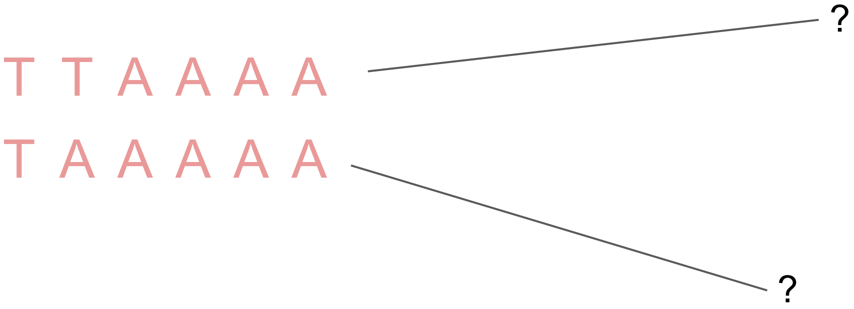
A A A G G A

A A G G A A

Make K-mers from S and query reference genome

Take the sequencing read

S= T T A A A A A



T T A A A A

T A A A A A

?

?

NO MATCH IN REFERENCE GENOME

Cat Reference Genome

A A C C A A A G G A A

Break into k-mers (6-mers)

A A C C A A

A C C A A A

C C A A A G

C A A A G G

A A A G G A

A A G G A A

What about if we “repair the damage”

Old read

S= T T A A A A A

S_fix= C C A A A G G

C C A A A G

C A A A G G

Cat Reference Genome

A A C C A A A G G A A

Break into k-mers (6-mers)

A A C C A A

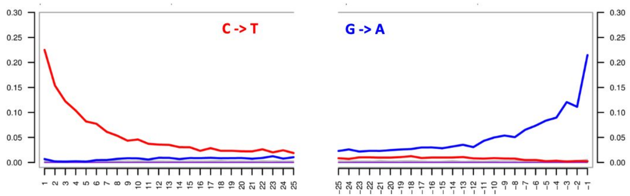
A C C A A A

C C A A A G

C A A A G G

A A A G G A

A A G G A A



Ginolhac et al.

What about if we “repair the damage”

Old read

S= T T A A A A A

S_fix= C C A A A G G

C C A A A G

C A A A G G

Perfect match to cat genome!

Cat Reference Genome

A A C C A A A G G A A

Break into k-mers (6-mers)

A A C C A A

A C C A A A

C C A A A G

C A A A G G

A A A G G A

A A G G A A

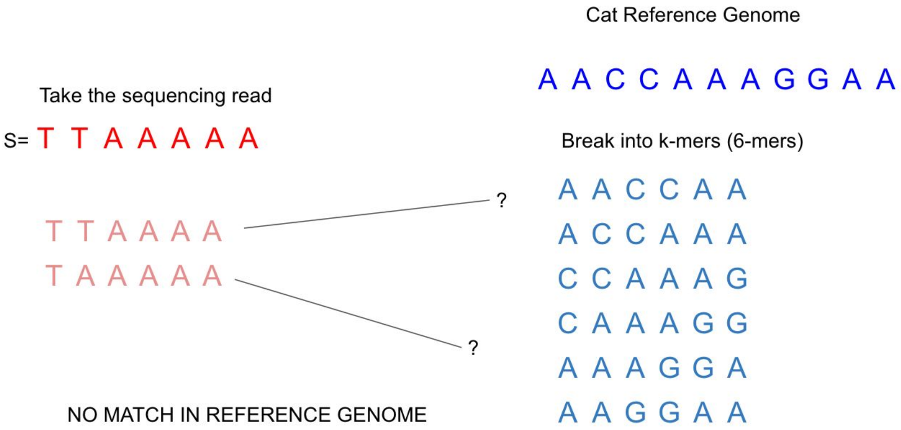
How can we handle ancient damage?

1. Repair the C-T and G-A transitions bioinformatically, like we did on the last slide?

   Trim end of bases from the read

2. Repair the transitions enzymatically?

3. Something else?



Cat Reference Genome

A A C C A A A G G A A

Take the sequencing read

S= T T A A A A A

T T A A A A

T A A A A A

?

?

Break into k-mers (6-mers)

A A C C A A

A C C A A A

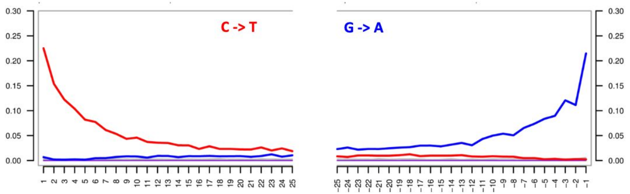
C C A A A G

C A A A G G

A A A G G A

A A G G A A

NO MATCH IN REFERENCE GENOME



Ginolhac et al.

Summary

1. What is a k-mer
2. Why do we want to count k-mers
3. What is a bloom filter
4. What is the lowest common ancestor (LCA)
5. How are k-mers used for metagenomic sequence classification?
6. How does metagenomic sequence classification handle ancient DNA?