Indexing the (compacted) colored de Bruijn graph

UNIVERSITY OF MARYLAND

NOTE: This lecture is being recorded

Scaling up fast reference-based indices

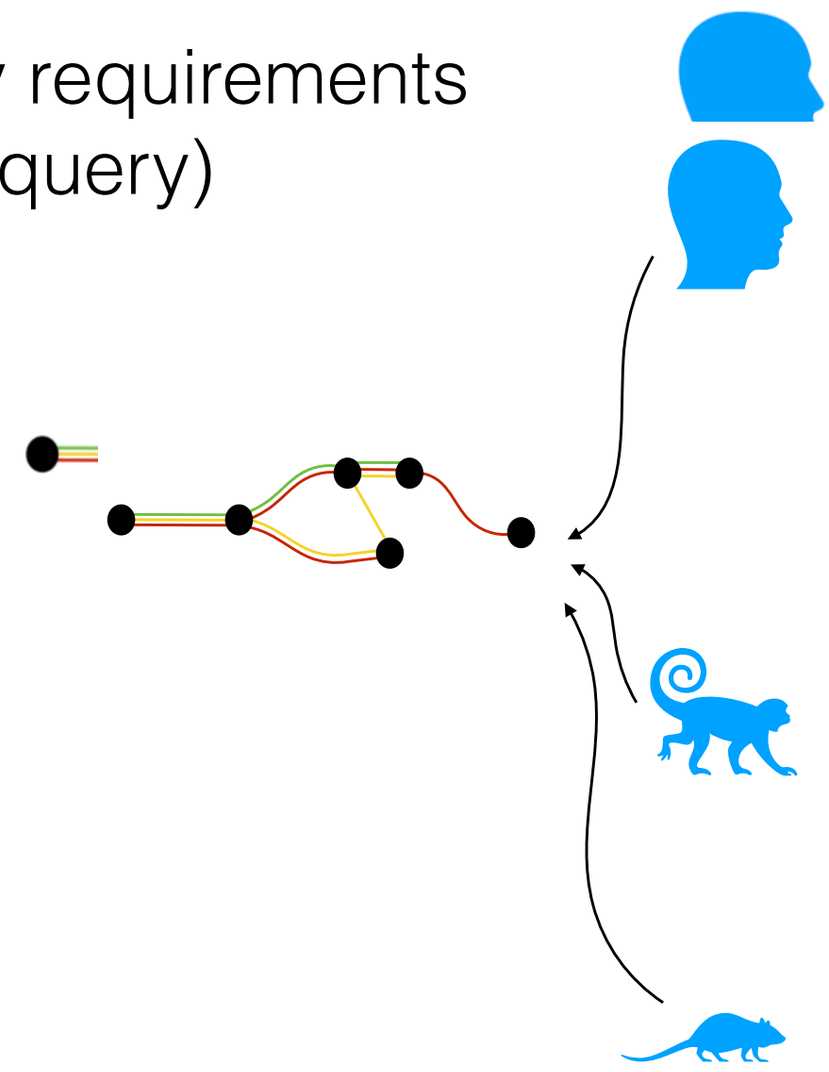
Motivation: Indices used in “ultra-fast” mapping approaches are typically very memory hungry. This is **OK** for transcriptome mapping, but **not scalable** to genomic, metagenomic, pangenomic or population mapping.

Goal: Develop an index with practical memory requirements that maintains the desirable performance (i.e. query) characteristics of the “ultra-fast” indices.

Compacted colored de Bruijn graph (ccdBG)

Built over 1 or more genomes / sequence collections

Index makes use of minimum perfect hashing succinct bit vector representations and (optionally) a new sampling scheme



Pufferfish: An efficient index for the ccdBG

Bioinformatics, 34, 2018, i169–i177
doi: 10.1093/bioinformatics/bty292
ISMB 2018

OXFORD

A space and time-efficient index for the compacted colored de Bruijn graph

Fatemeh Almodaresi†, Hirak Sarkar†, Avi Srivastava and Rob Patro*

Department of Computer Science, Stony Brook University, Stonybrook, NY 11794, USA

Appeared at **ISMB 2018**

- The past decade has largely been dominated by SA/BWT/FM-index-based approaches to reference sequence indexing (e.g. Bowtie, BWA, BWA-MEM, Bowtie2, STAR, etc.)
- There has been a renaissance of sorts for hash-based indexing (deBGA, Brownie, kallisto, mashmap, minimap & minimap2, etc.)
- Pufferfish goes the hashing-based route; *with a twist*.
- Not considering generalized path indices on general seq (e.g. GCSA2 (VG), HISAT2). Interesting, but a different problem.

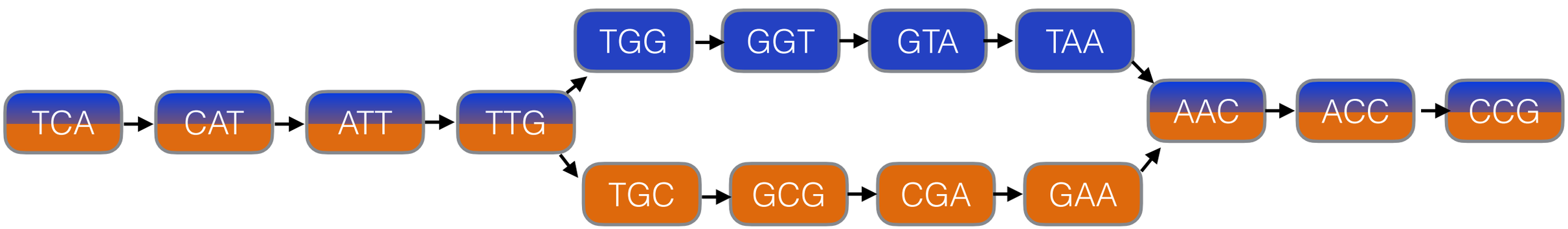
https://github.com/COMBINE-lab/pufferfish

Recall the “colored” de Bruijn Graph

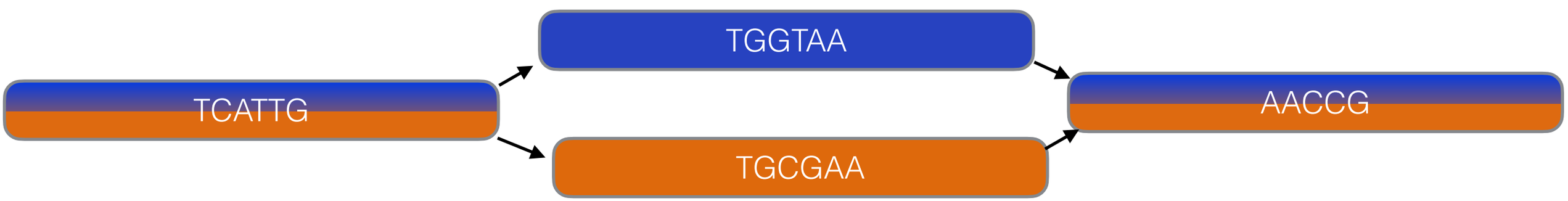
Nodes are k-mers (here k=3)

Edges exist between nodes that overlap by k-1 (in the input)*

Colors encode “origin” of k-mers (e.g., references where they exist)



compacted colored de Bruijn graph



Example from : https://algolab.files.wordpress.com/2016/10/chikhi-milan-18nov.pdf

There are multiple related (but distinct) definitions of the dBG in practice. We adopt the **edge-explicit** version.

The compacted colored dBG as a sequence index

- **Key idea**: represent a collection of sequences using the colored de Bruijn graph (dBG) (Iqbal ’12).

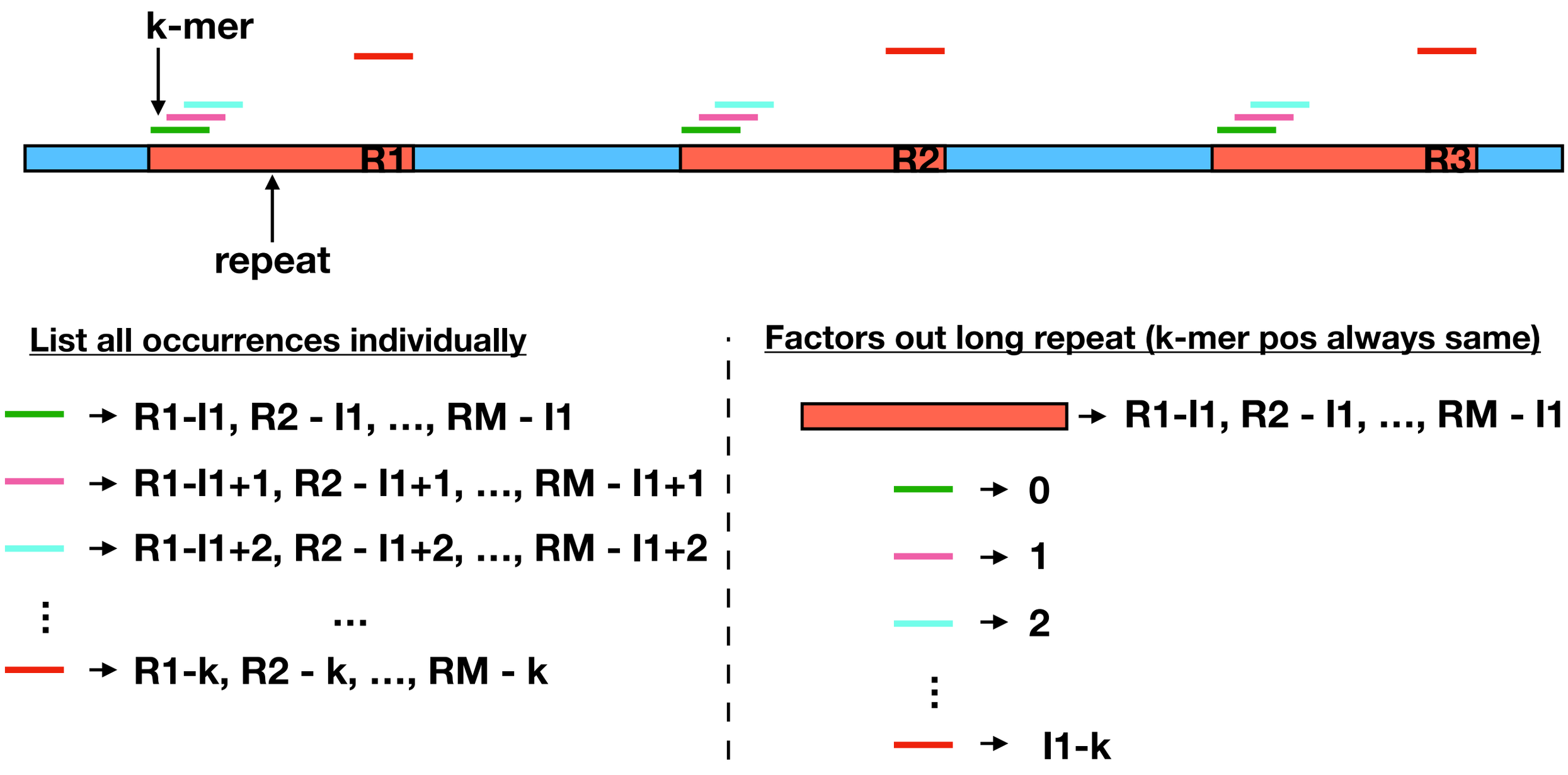
- Each color is an input reference (e.g. genome or transcript).

- Use the compacted colored dBG as an index for reference-based sequence search.

- Redundant sequences (repeats) are implicitly collapsed. **Why is this potentially *much* better than a naive hash?**

The compacted colored dBG as a sequence index

- Redundant sequences (repeats) are implicitly collapsed. **Why is this potentially *much* better than a naive hash?**



The cdBG removes redundancy by providing an extra level of indirection

The compacted colored dBG as a sequence index

- Redundant sequences (repeats) are implicitly collapsed. **Why is this potentially *much* better than a naive hash?**

Still, the biggest problem for these schemes, in practice, is *memory usage*

The main culprit is the hash table itself

The cdBG removes redundancy by providing an extra level of indirection

Recall: Minimum Perfect Hashing

Minimum Perfect Hash Function (MPHF)

$$\mathscr{K} \subseteq \mathscr{U}, \quad f : \mathscr{K} \rightarrow \mathbb{N}^{+}$$

if $x \in \mathscr{K}$ **then** $f(x) \in [1, \left|\mathscr{K}\right|]$

if $x \in \mathscr{U} \backslash \mathscr{K}$ **then** $f(x) \in [1, \left|\mathscr{U}\right|]$ (Like "false positives")

f is a complete, injective function from $\mathscr{K} \rightarrow [1, \left|\mathscr{K}\right|]$

Best methods achieve ~2.1 bits/key ***regardless of key size***

Use BBHash :)

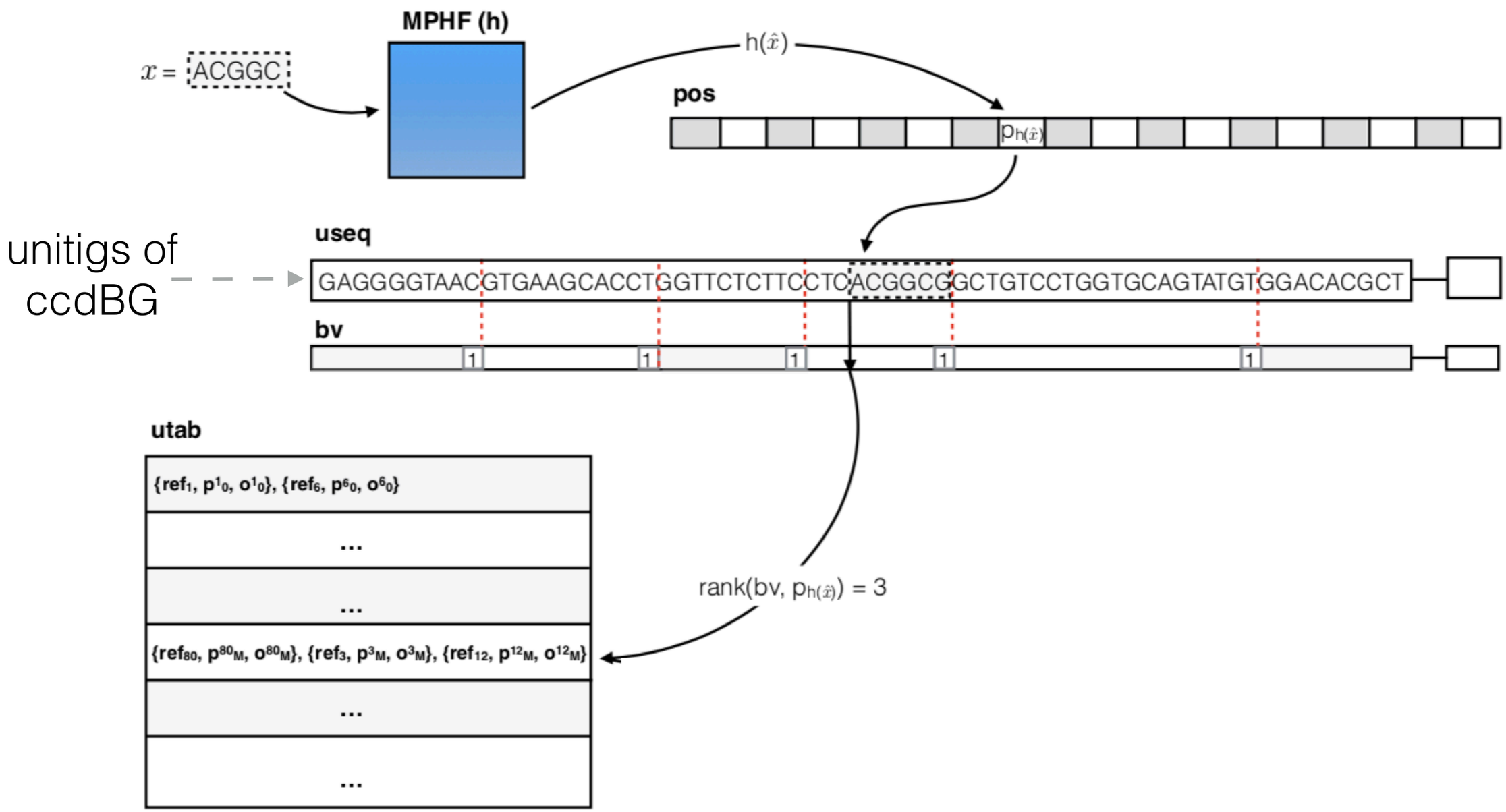
Fast and scalable minimal perfect hashing for massive key sets

Antoine Limasset[1], Guillaume Rizk[1], Rayan Chikhi[2], and Pierre Peterlongo[1]

1 IRISA Inria Rennes Bretagne Atlantique, GenScale team, Campus de Beaulieu 35042 Rennes, France
2 CNRS, CRIStAL, Université de Lille, Inria Lille - Nord Europe, France

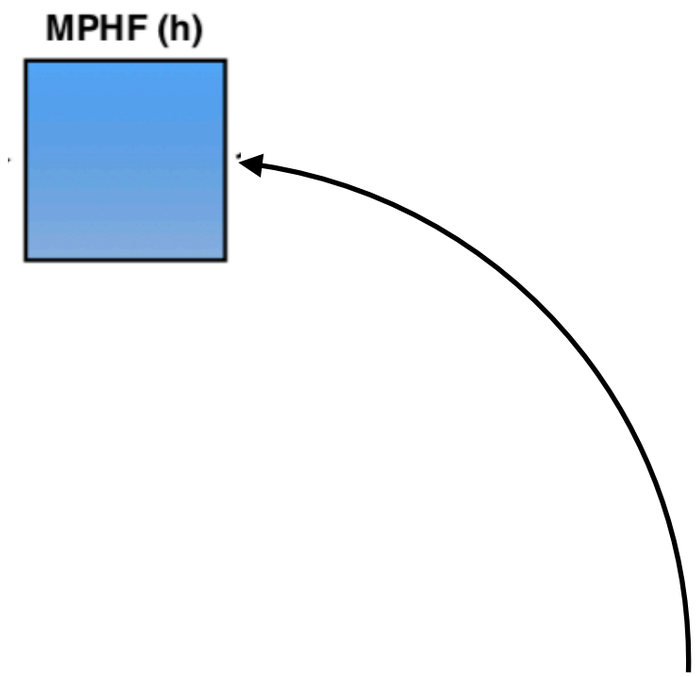
https://github.com/rizkg/BBHash

The **dense** Pufferfish index



Optionally: explicit edge table, equivalence class table

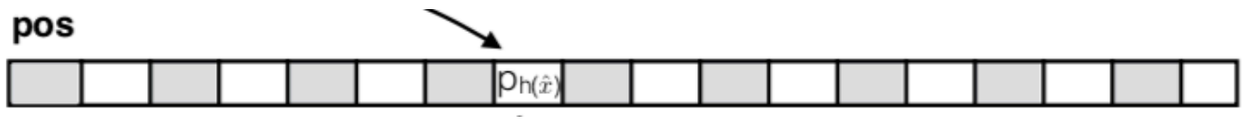
The **dense** Pufferfish index



Maps each valid k-mer to some number in [0,N)

Optionally: explicit edge table, equivalence class table

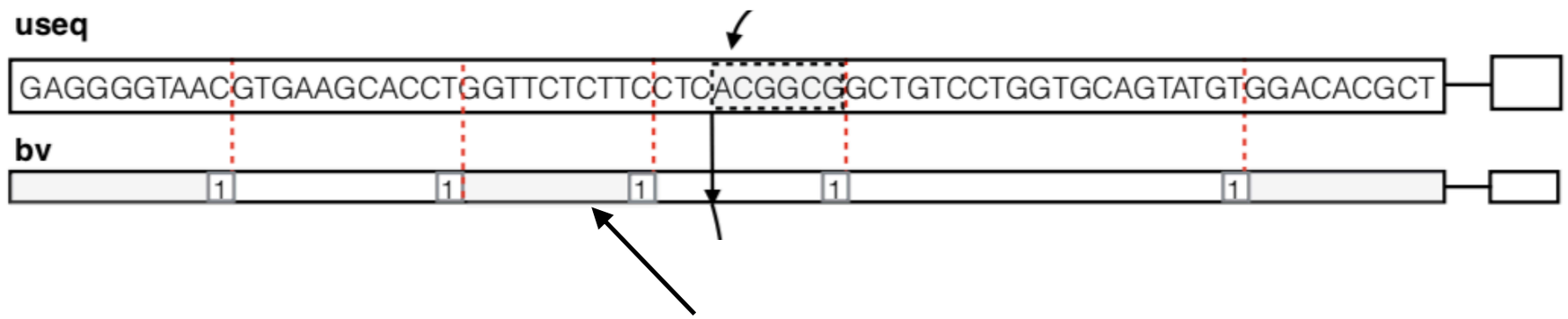
The **dense** Pufferfish index



At index h(x), this table contains the position, in the list of unitigs, of this k-mer

Optionally: explicit edge table, equivalence class table

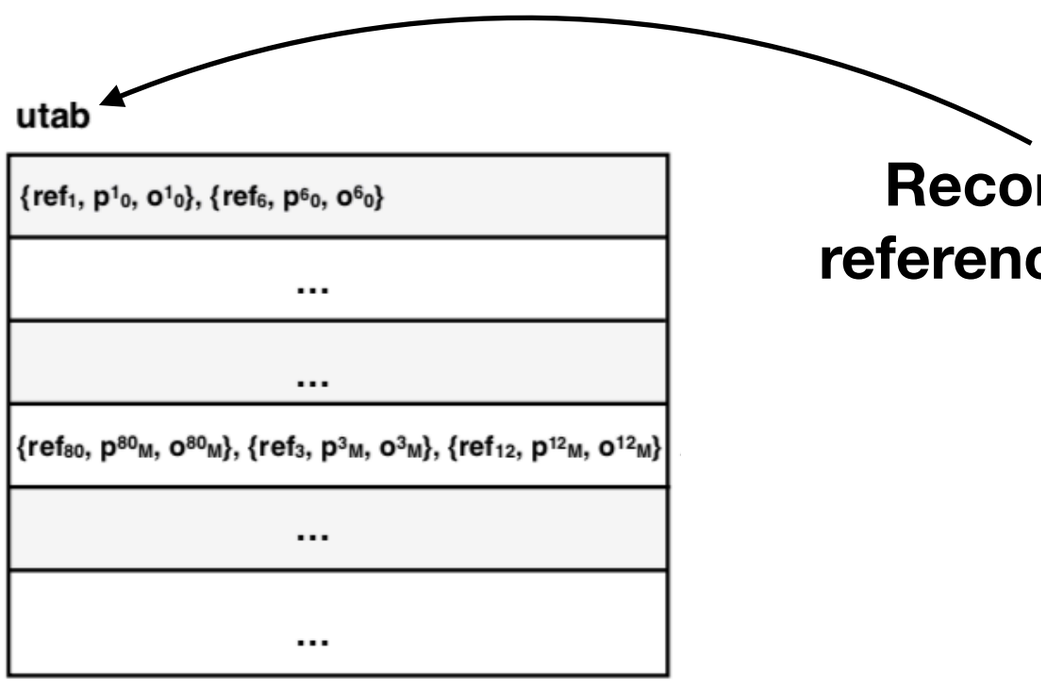
The **dense** Pufferfish index



- **useq contains the uniting sequences concatenated together**
- **bv is a boundary vector that records a 1 at the end of each uniting, and a 0 elsewhere**

Optionally: explicit edge table, equivalence class table

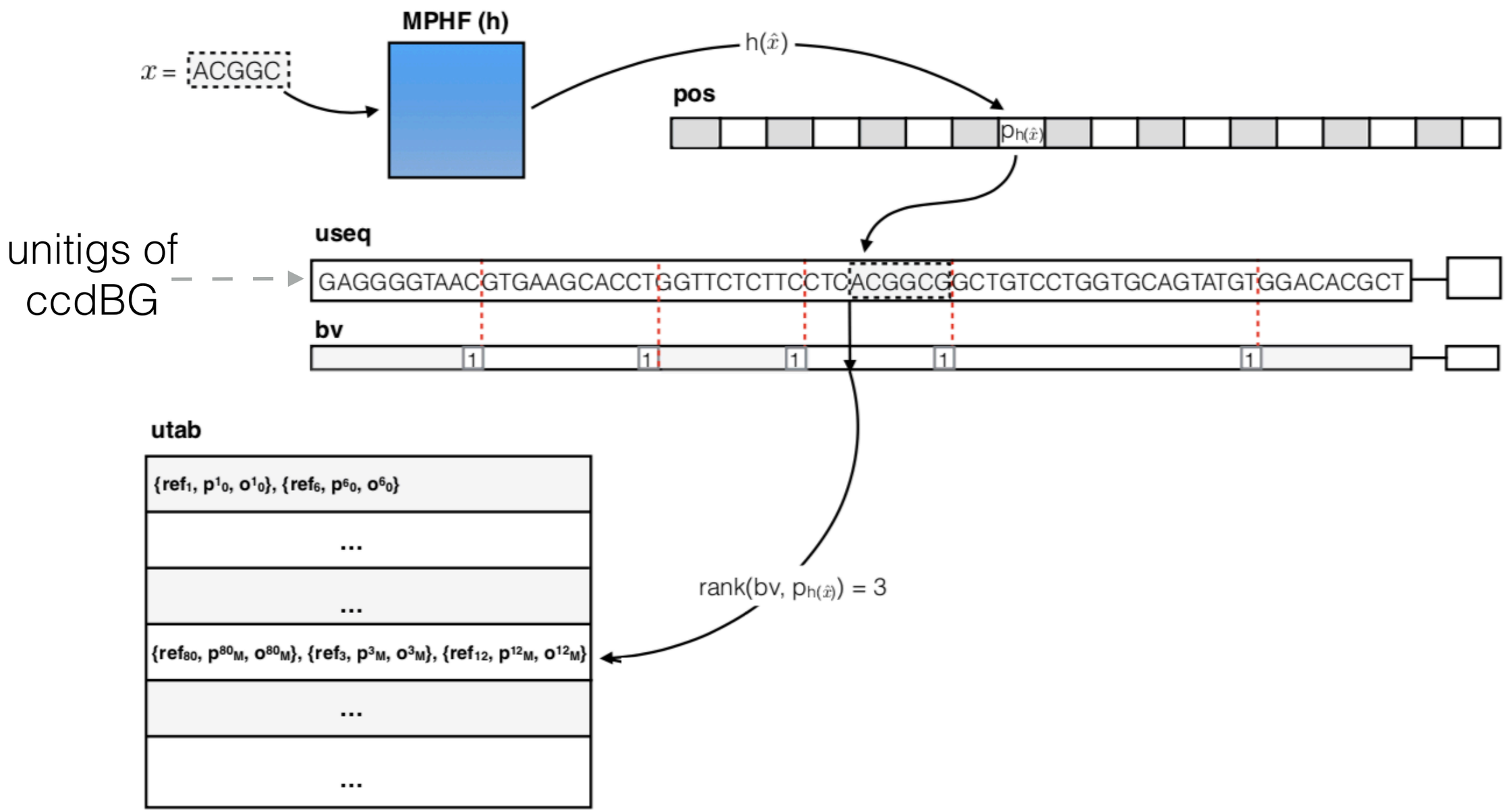
The **dense** Pufferfish index



Records, for each uniting, the list of references, positions and orientations in which it occurs

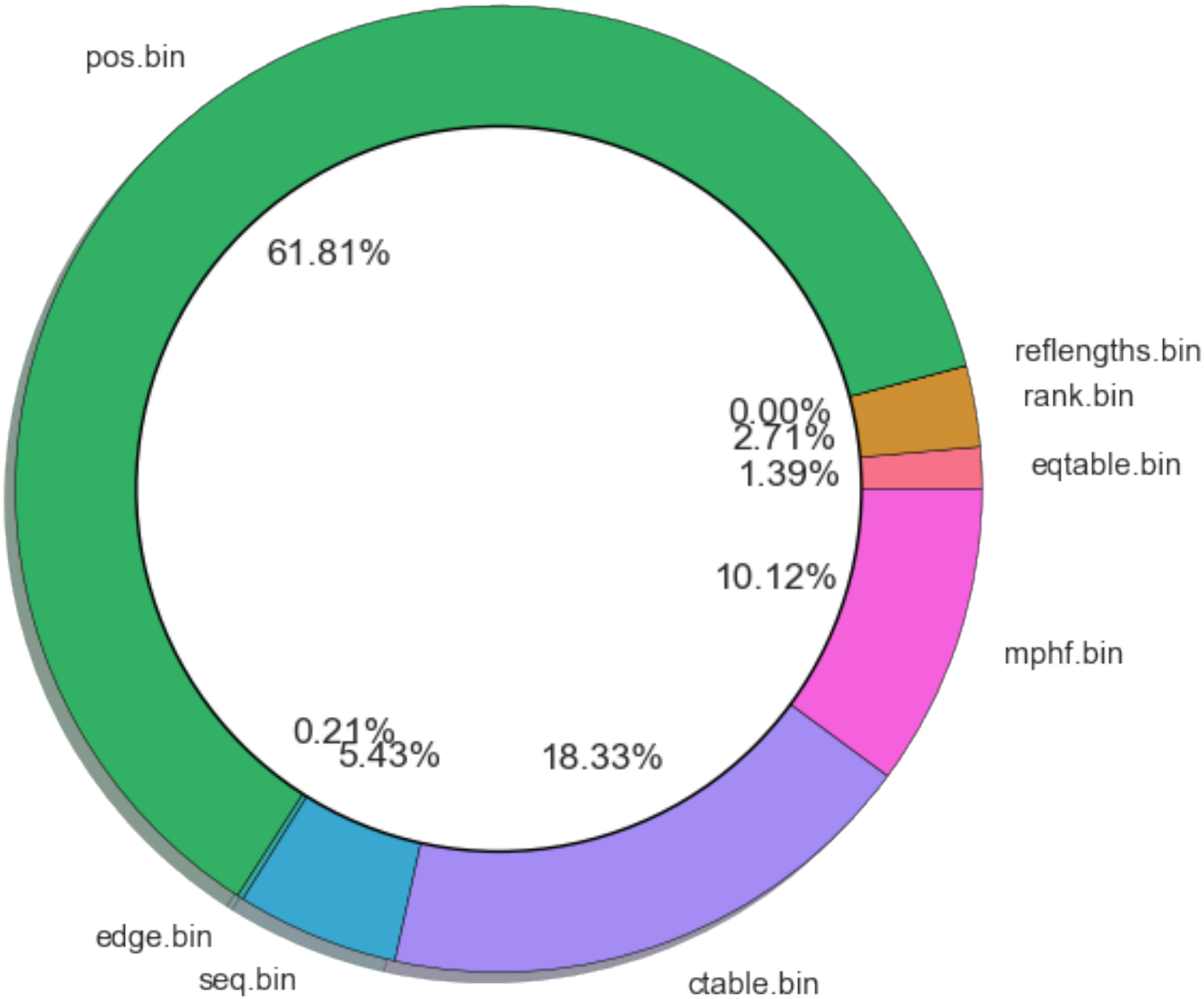
Optionally: explicit edge table, equivalence class table

The **dense** Pufferfish index

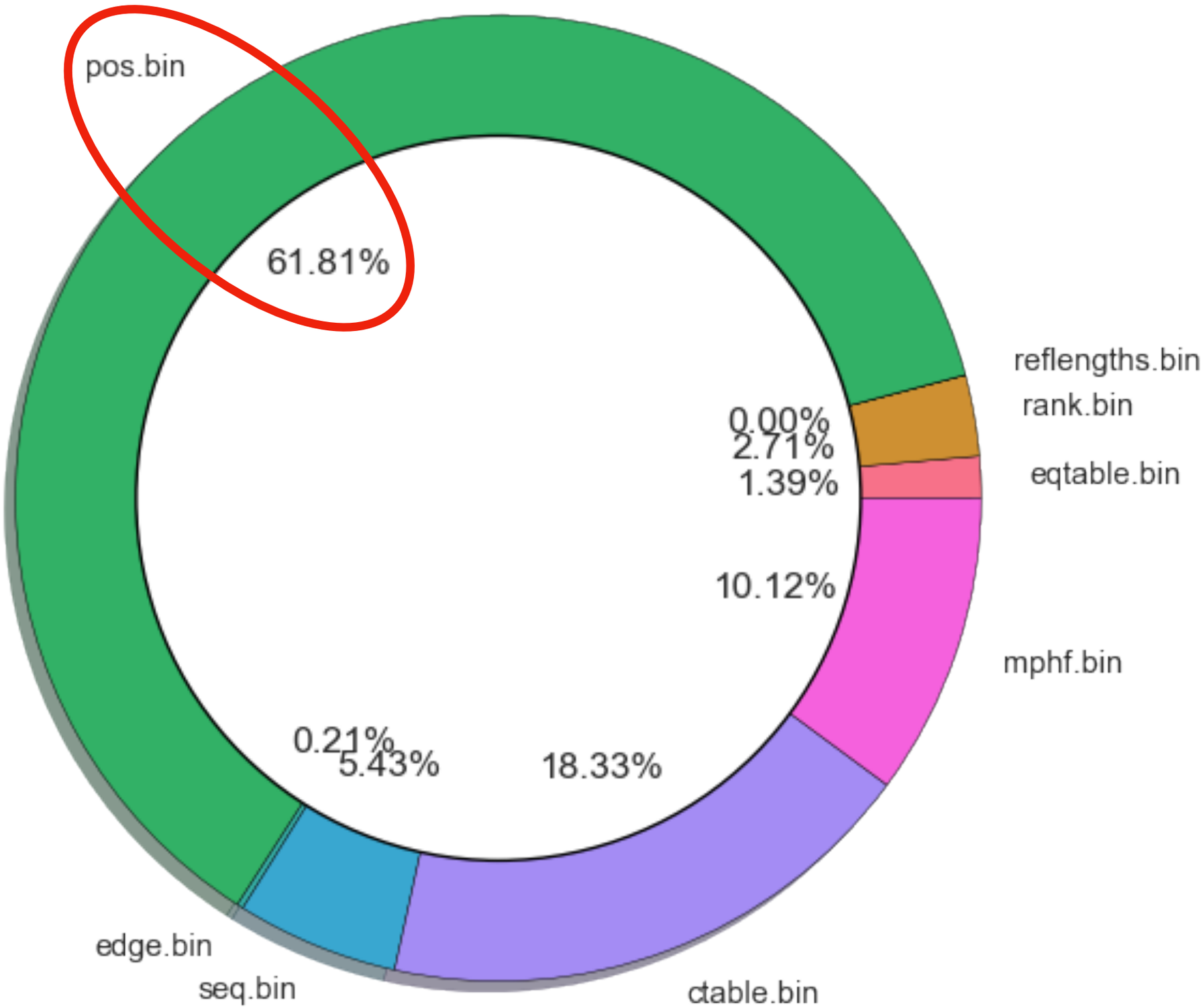


Optionally: explicit edge table, equivalence class table

Who's the culprit?

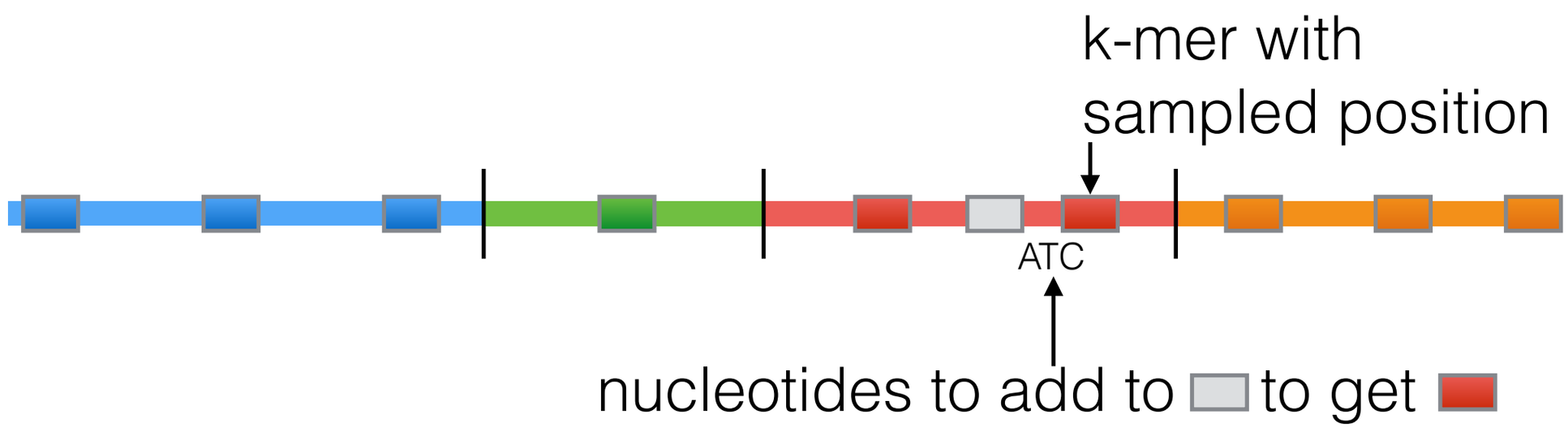


Who's the culprit?



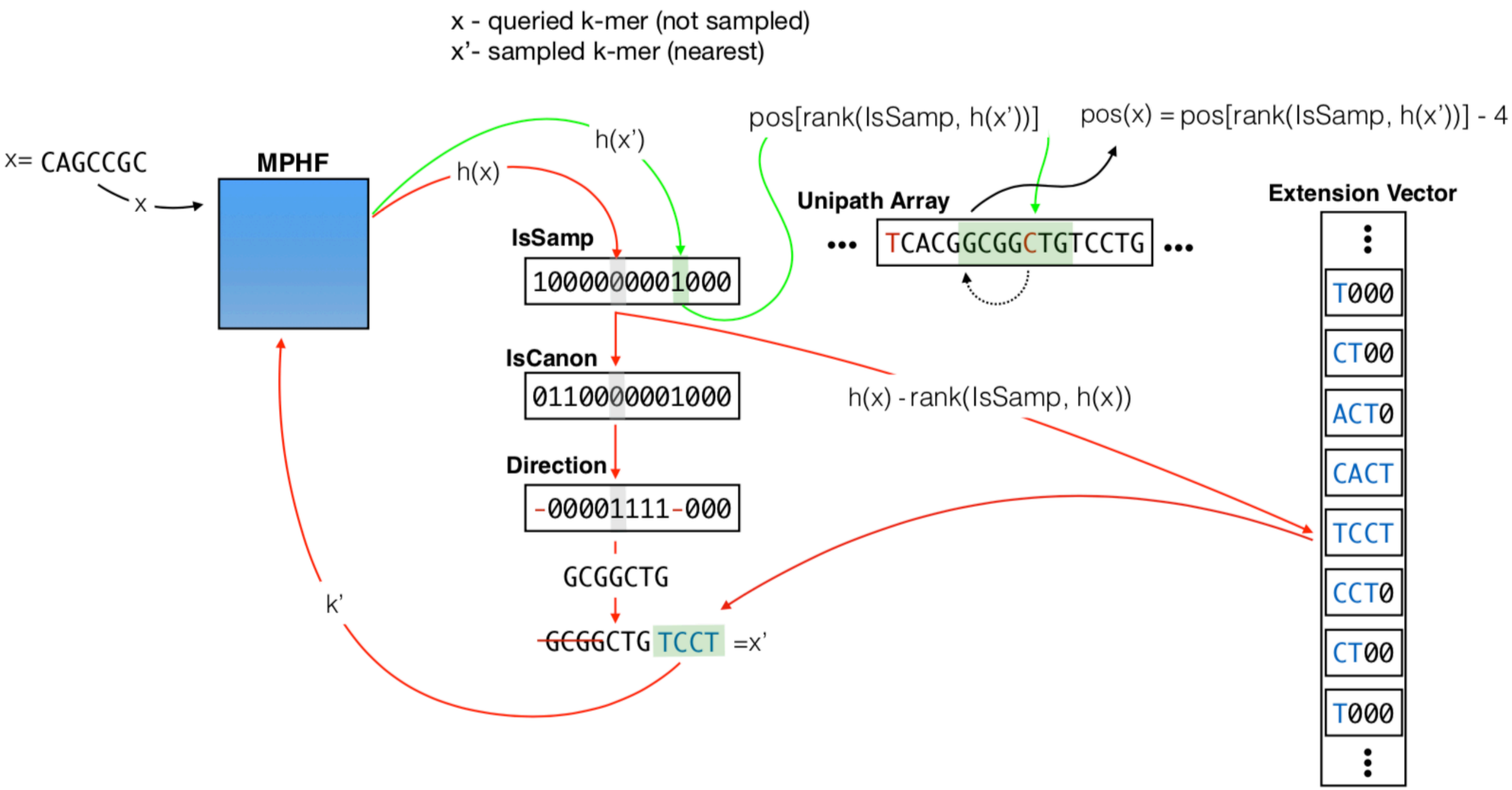
The **sparse** Pufferfish index

In large indices, the position table *dominates* index size



Intuition: Successors and predecessors in unipaths are *globally unique*, instead of storing position information for all k-mers, store positions only at sampled "landmarks" and say how to navigate to these landmarks (similar to bi-directional sampling in the FM-index).

The **sparse** Pufferfish index (in detail)

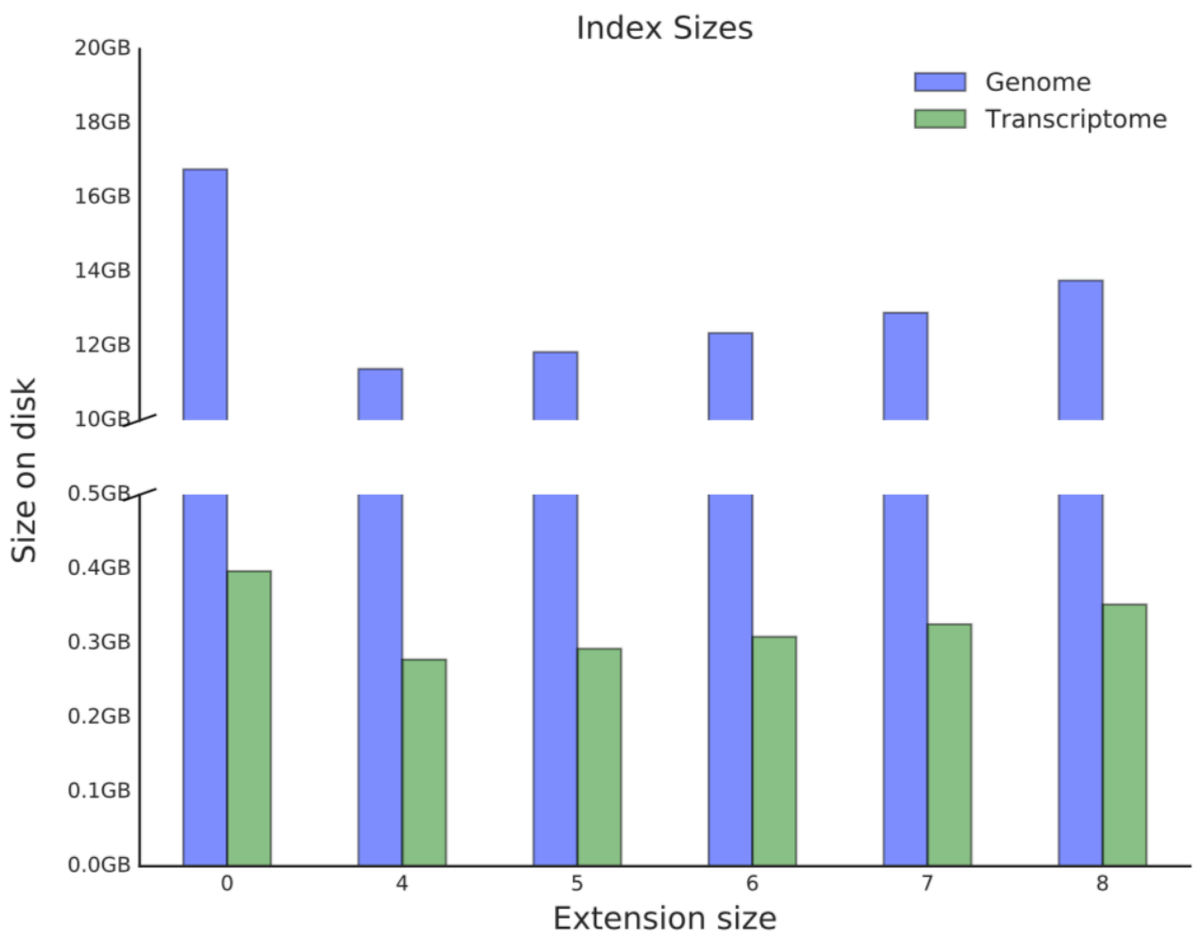


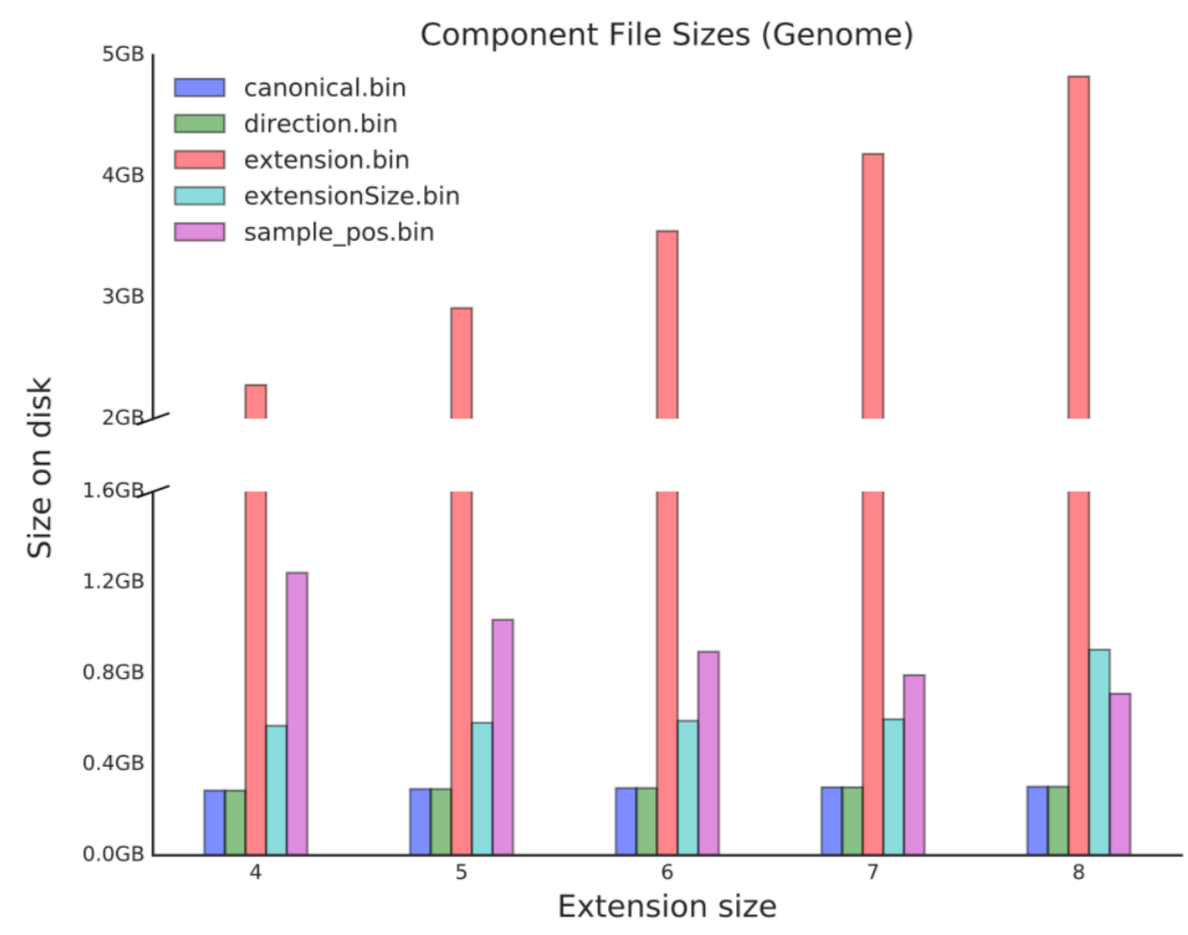
What sampling factor is right?

Tradeoff : Sparser sampling → less space but slower lookup

Fastest : Sampling factor s > 2·e+1 (Still a range of sizes)

Smallest : Extension size = 1, sampling = s





Index space & K-mer query time

Space of index + query in RAM

Tool	Memory (MB)		
	Human Transcriptome	Human Genome	Bacterial Genome
BWA	308	4,439	27,535
kallisto	3,336	110,464	232,353
pufferfish dense	454	17,684	41,532
pufferfish sparse	341	12,533	30,565

#Li, H. (2013). Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM. arXiv Preprint arXiv:1303.3997.

^Bray, N. L., Pimentel, H., Melsted, P., and Pachter, L. (2016). Near-optimal probabilistic RNA-seq quantification. Nature Biotechnology, 34(5), 525–527.

Index space & K-mer query time

Time to look up all fixed-length substrings in an experiment

Tool	Time (h:m:s)		
	Human Transcriptome	Human Genome	Bacterial Genome
BWA	0:17:35	0:50:31	0:14:05
kallisto	0:02:01	0:19:11	0:22:25
pufferfish dense	0:02:46	0:10:37	0:06:03
pufferfish sparse	0:08:34	0:22:11	0:08:26
# queries:	747,842,900	7,508,576,020	509,143,360

Pufferfish summary (part 1)

- To keep memory usage reasonable, we have to be quite careful about our hashing-based schemes.

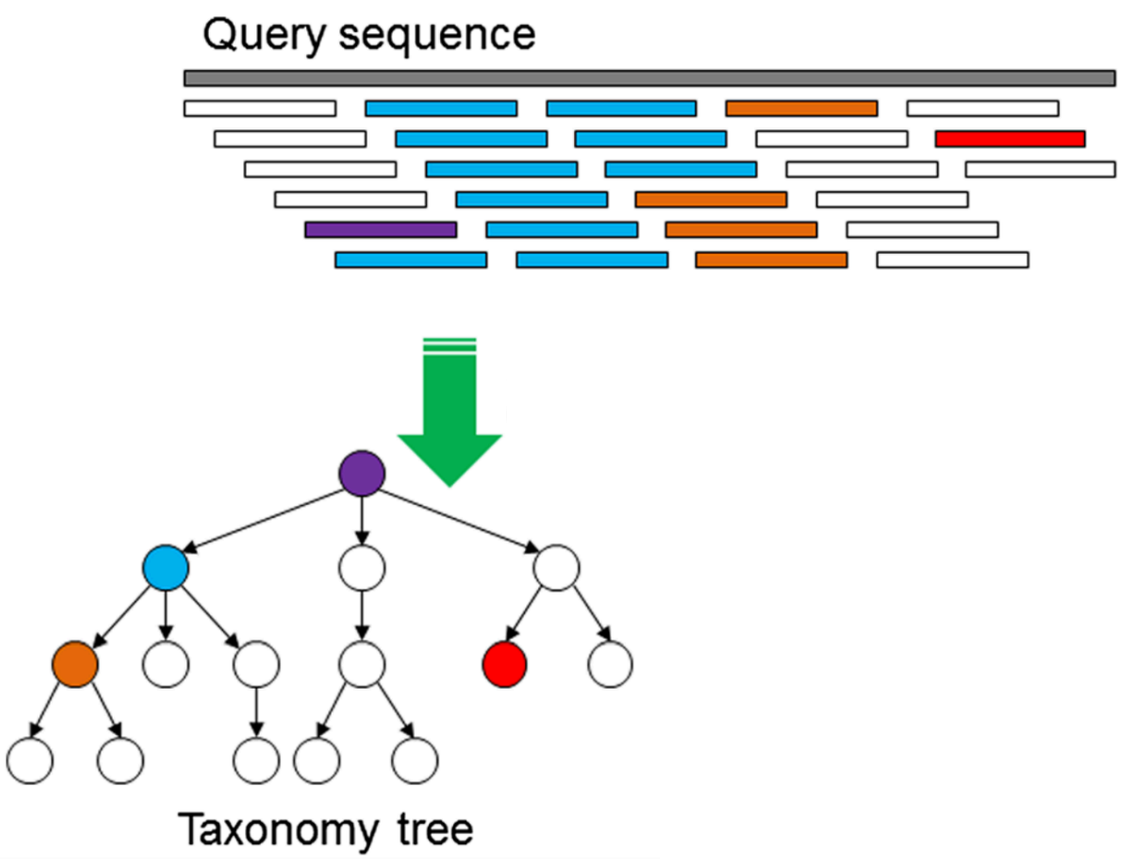
- The dense pufferfish index strikes a good balance between index space and raw query speed.

- At a constant factor (though not asymptotic) cost, index size is tunable with our sampling scheme.

- At least for fixed-length patterns, a good hashing approach can be *much faster* than (still asymptotically-optimal) full-text indexes.

An example application of Pufferfish

- Taxonomic read classification — for each read, assign it to the taxon (strain, species, genus) from which we think it derived. Related to, ***but distinct from***, taxonomic abundance estimation.



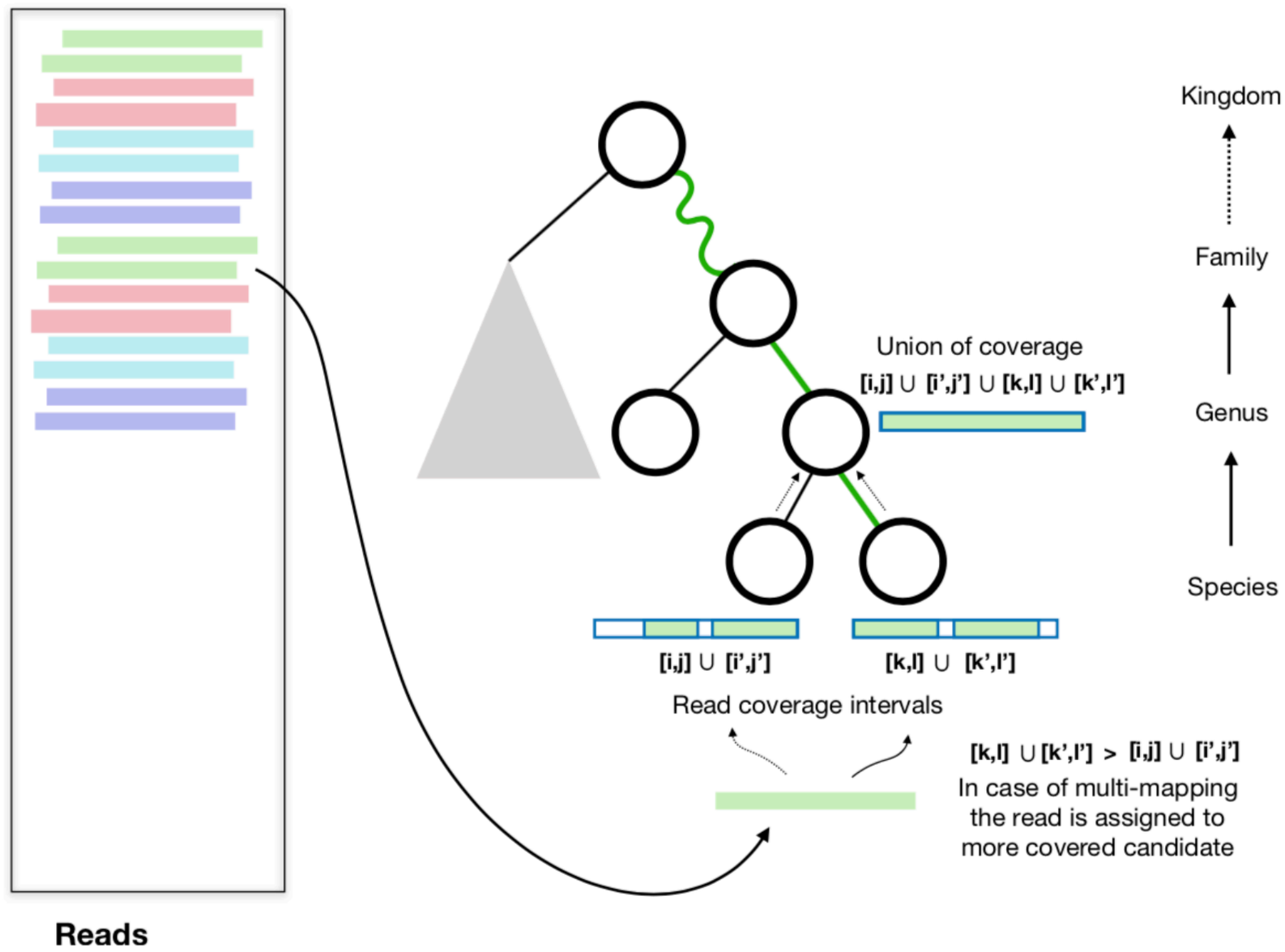
Figures adapted from: Wood, D.E. and Salzberg, S.L., 2014. Kraken: ultrafast metagenomic sequence classification using exact alignments. *Genome biology*, *15*(3), p.R46.

Pufferfish taxonomic assignment

We adopt what is essentially the algorithm of *Kraken**, but replace k-mer counting with lightweight mapping.

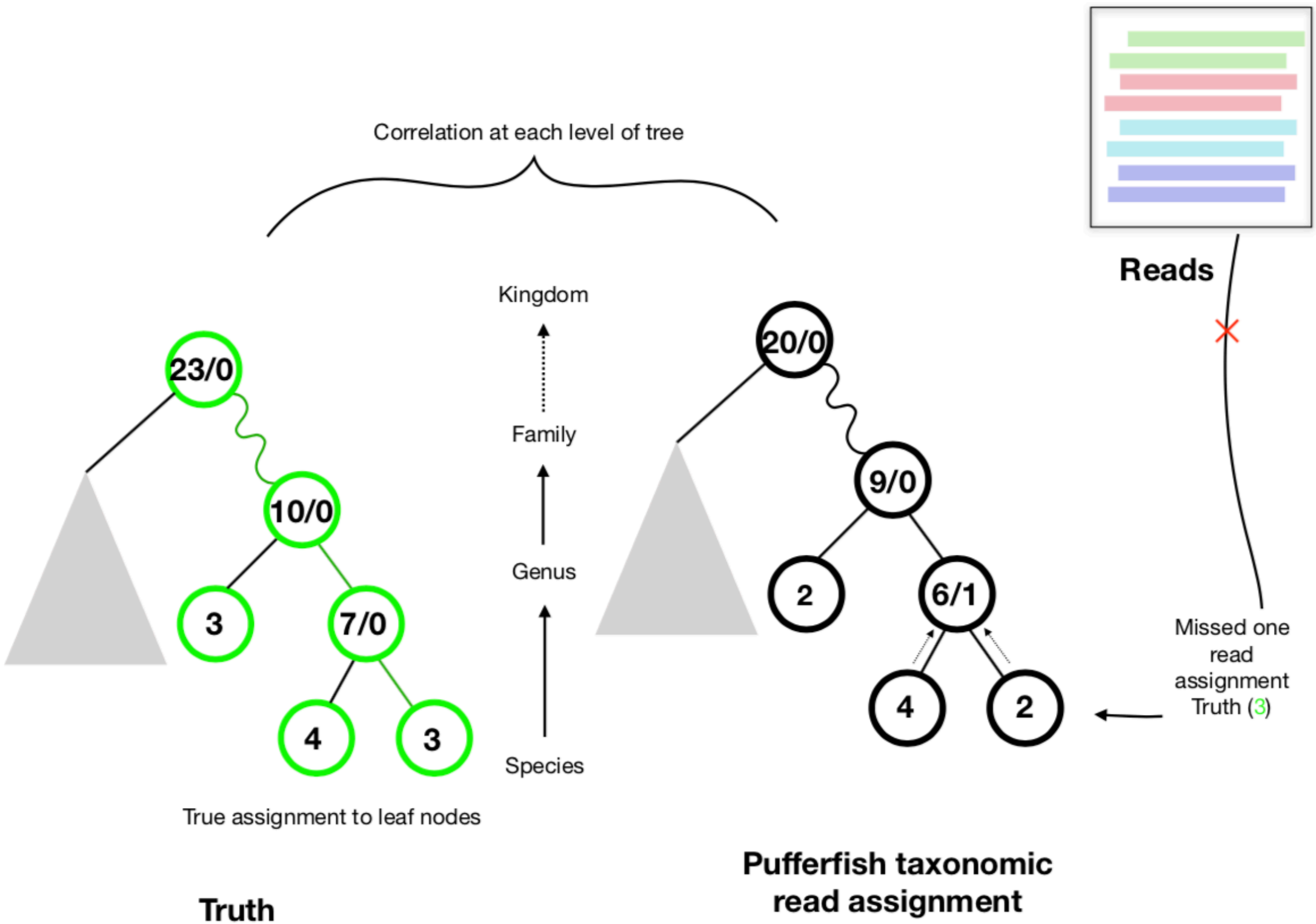
This *enforces positional & orientation consistency* of matches

- Score all root-to-leaf (RTL) paths
- Assign read to leaf of highest-scoring path
- In case of tie, assign read to LCA of all highest-scoring paths.

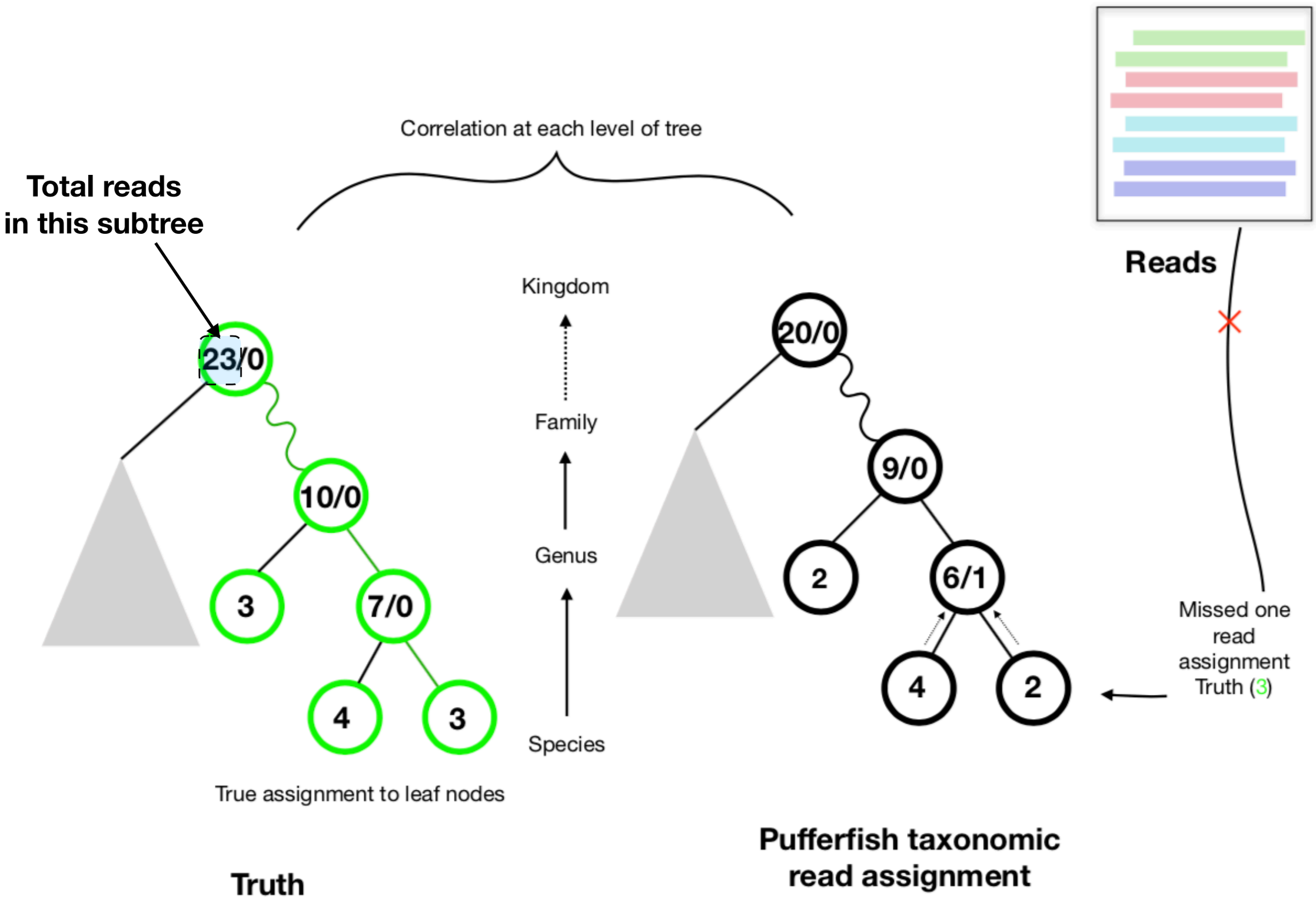


*Wood, D.E. and Salzberg, S.L., 2014. Kraken: ultrafast metagenomic sequence classification using exact alignments. *Genome biology*, *15*(3), p.R46.

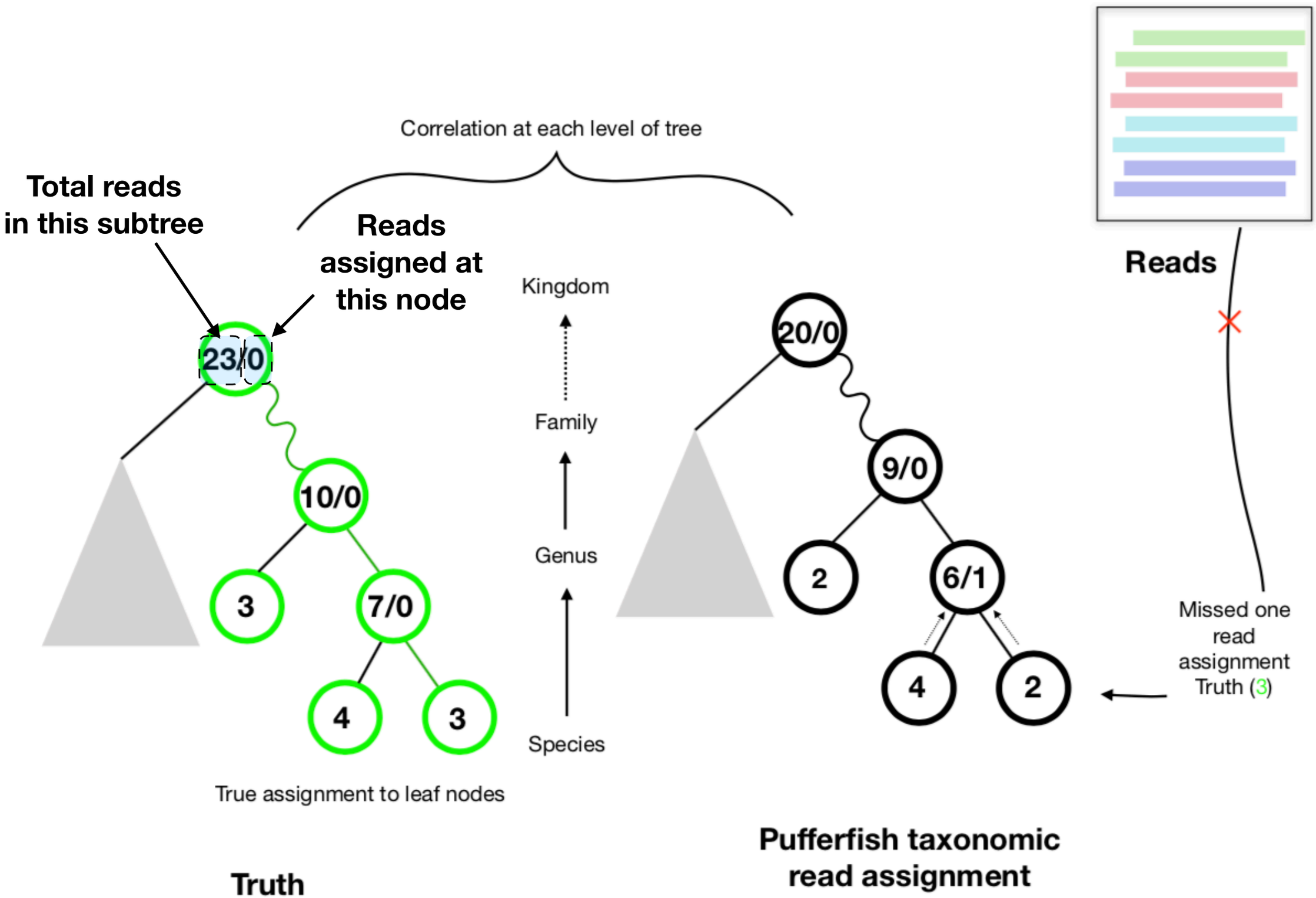
“Whole taxonomy” accuracy assessment



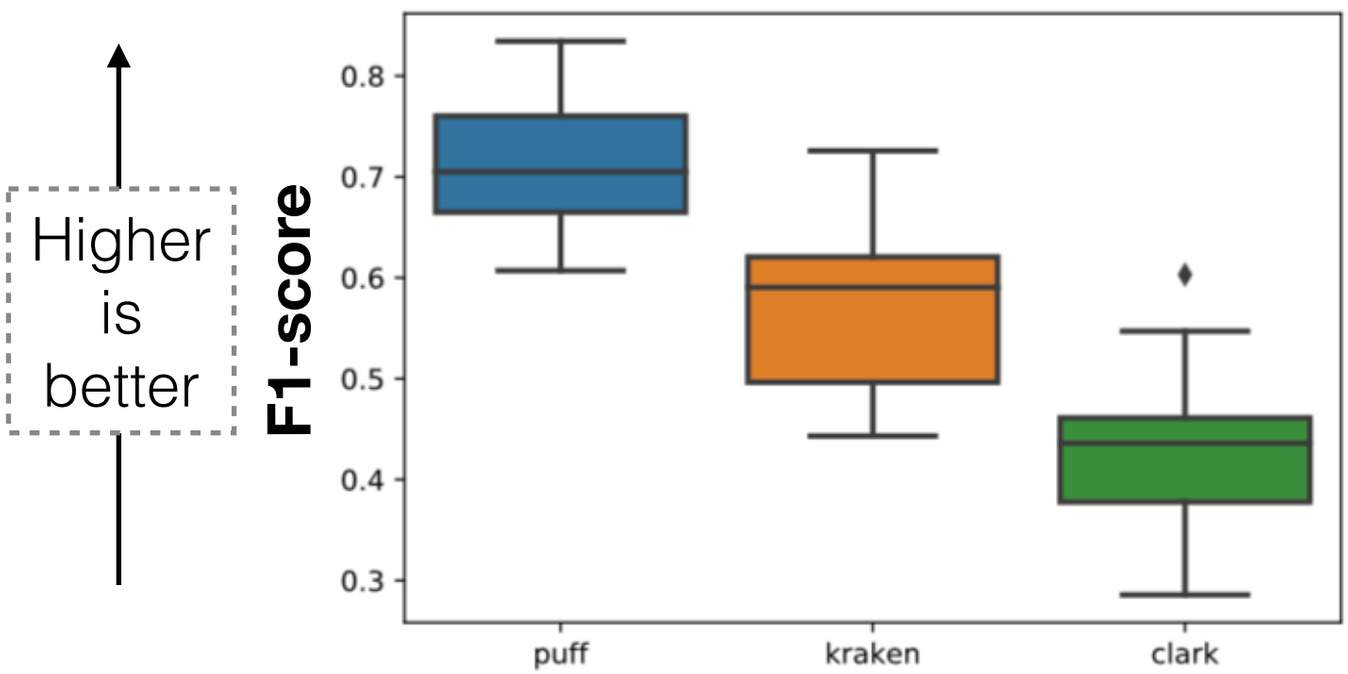
“Whole taxonomy” accuracy assessment

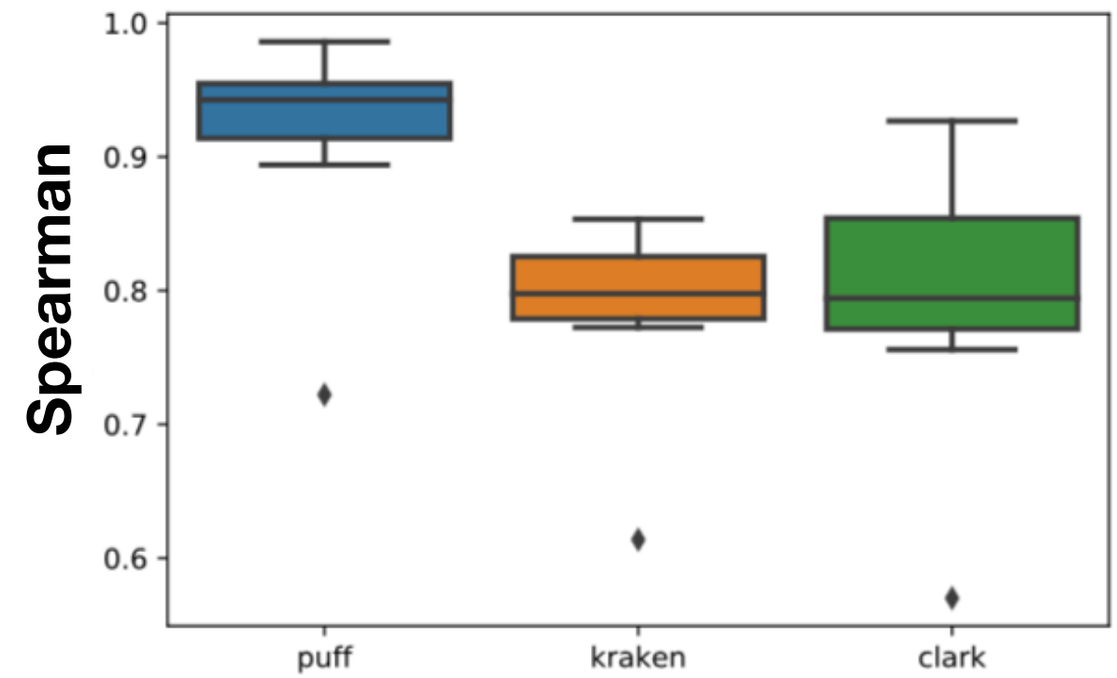


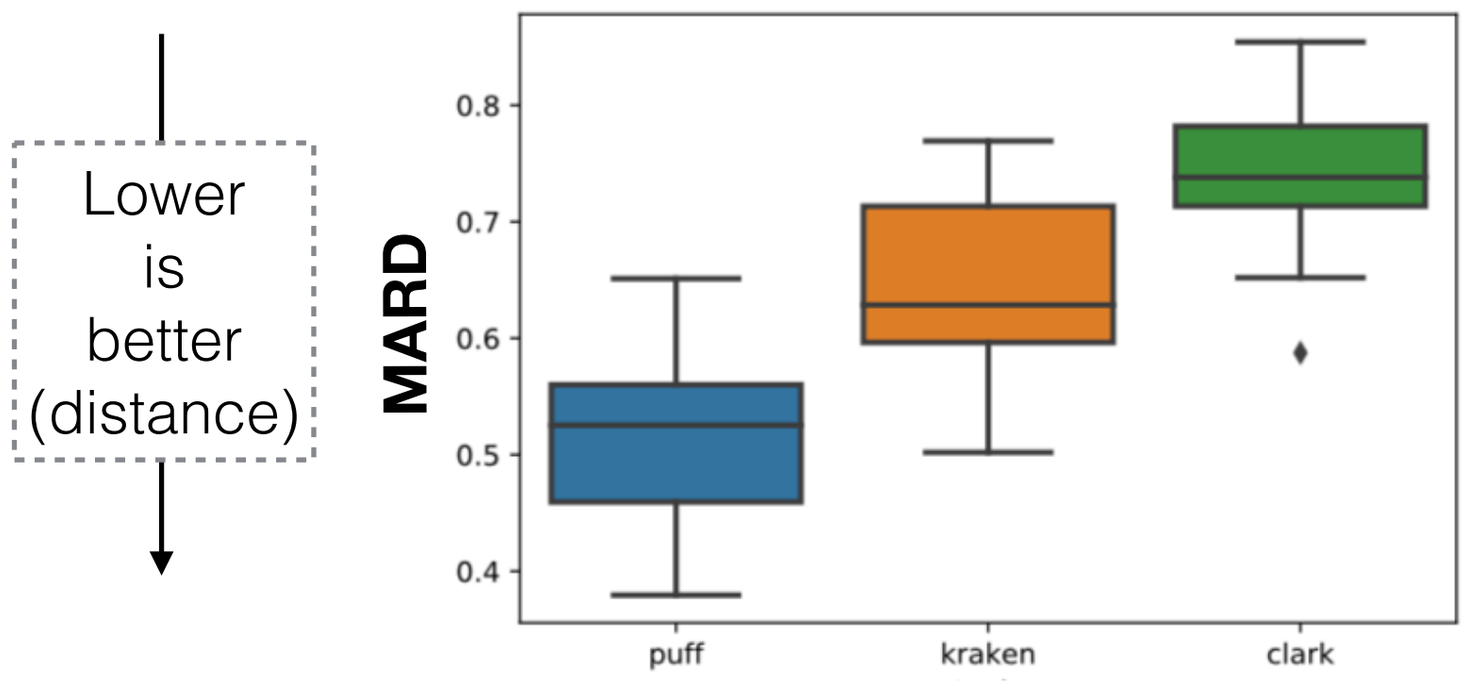
“Whole taxonomy” accuracy assessment



Pufferfish taxonomic assignment







Simulated data from : McIntyre, et al. (2017).
Comprehensive benchmarking and ensemble approaches for metagenomic classifiers. Genome Biology, 18(1).

Simulations:
(LC1-8, HC1, HC2)