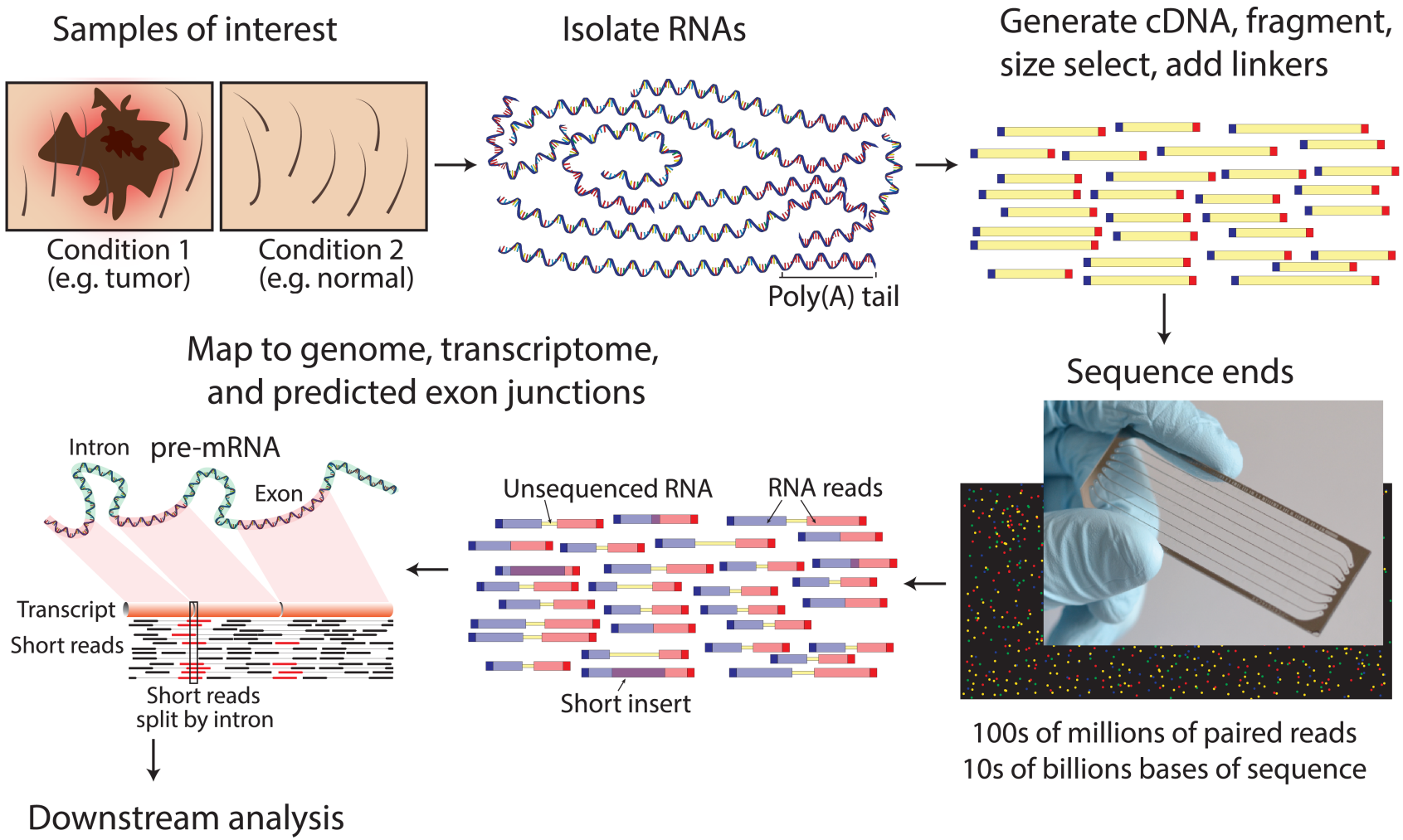
srnaMapper: an optimal mapping tool for sRNA-Seq reads

Matthias Zytnicki, Christine Gaspin

INRAE, MIAT

SeqBIM 2020

RNA-Seq

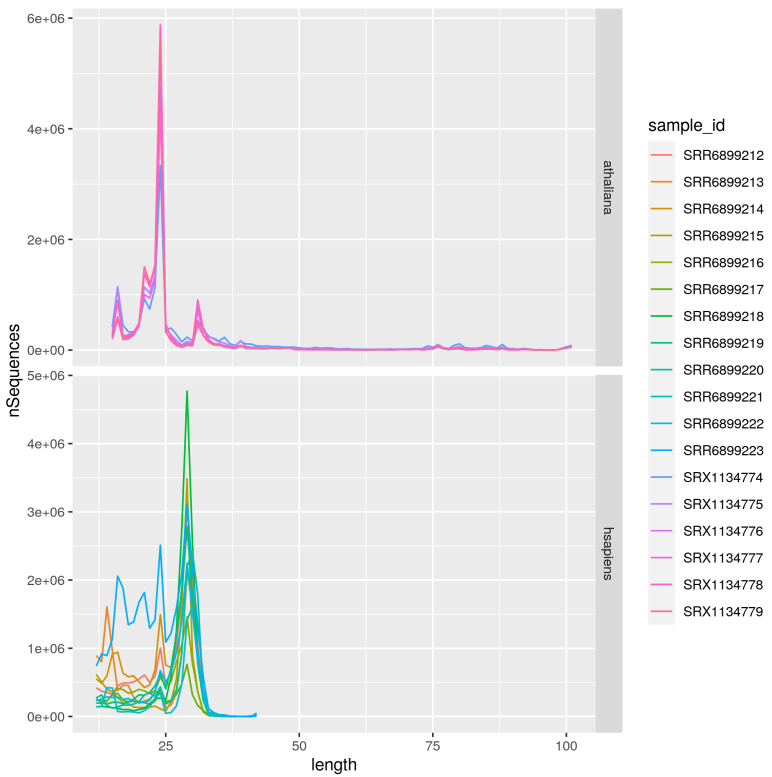


Griffiths et al., PLOS Comp. Biol., 2015

Our question

Particularities of sRNA-Seq

- A population of different classes of small RNAs: miRNAs, tRFs, siRNAs, piRNAs, etc.
- They are short (about 22–24bp, after trimming).
- Sequences are highly duplicated (∼5% the exact same read).
- Most mismatches happen at the ends of the reads.



ID	Accession	RPM	Chromosome	Start	End	Strand
ath-MIR156a	MI0000178	-	chr2	10676451	10676573	-
ath-MIR156b	MI0000179	-	chr4	15074899	15075081	+
ath-MIR156c	MI0000180	-	chr4	15415418	15415521	-
ath-MIR156d	MI0000181	-	chr5	3456632	3456749	-
ath-MIR156e	MI0000182	-	chr5	3867207	3867313	+
ath-MIR156f	MI0000183	-	chr5	9136106	9136237	+
ath-MIR157a	MI0000184	-	chr1	24913202	24913299	-
ath-MIR157b	MI0000185	-	chr1	24921086	24921217	+
ath-MIR157c	MI0000186	-	chr3	6244500	6244716	-
ath-MIR157d	MI0000187	-	chr1	18026811	18027031	-

from miRBase

Our question — Cont.

Observation

- Most mapping tool developments are dedicated to long reads.
- There is no dedicated tool for sRNAs.

Usual (biological) query

For each read, get me *all* the regions with *minimum* number of mismatches n, with $n \leq k$.

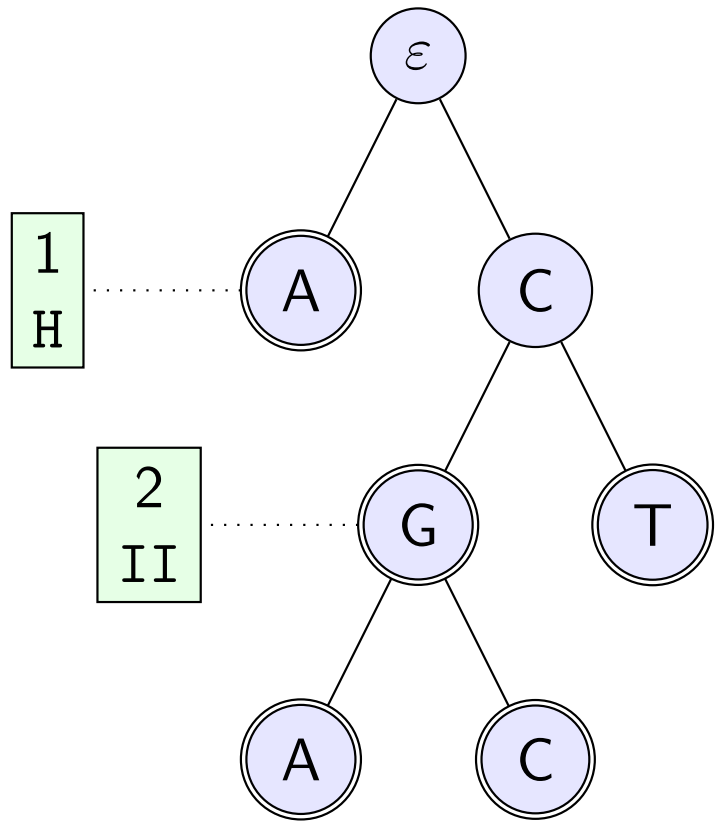
Data

Reads

- Stored in a tree.
- Counts, and best quality is kept.

```
@read1
A
+
H
@read2
CG
+
HI
@read3
CG
+
IH
```

```
@read4
CGA
+
HHI
@read5
CGC
+
IIH
@read6
CT
+
II
```



Data

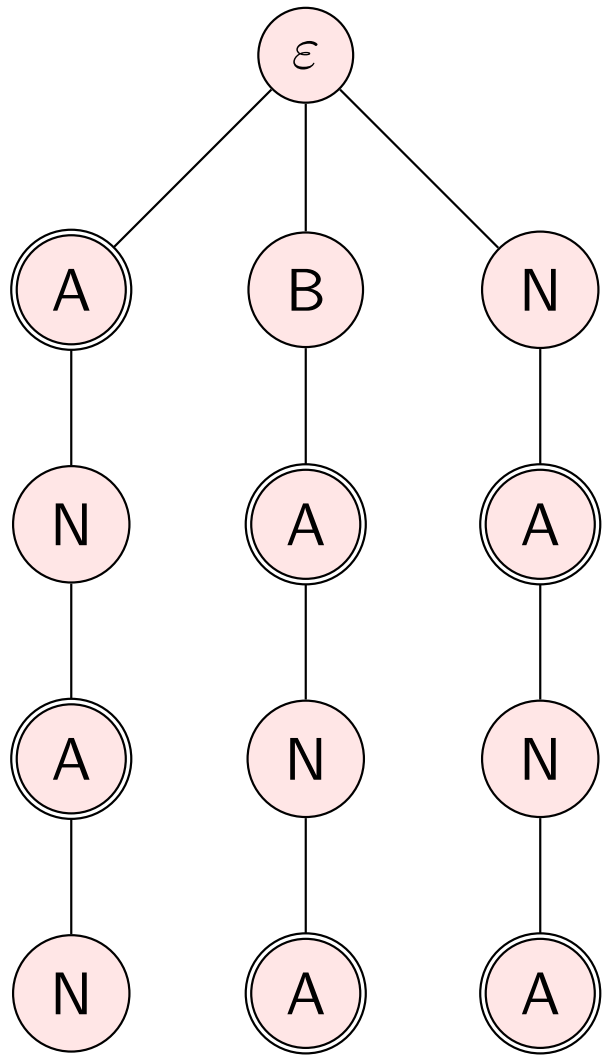
Genome

- Stored in a suffix array.
- Using BWA implementation.

Example

BANANA

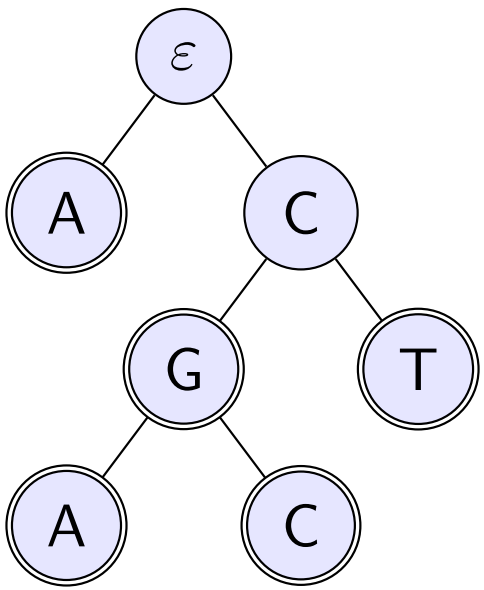
Suffix tree

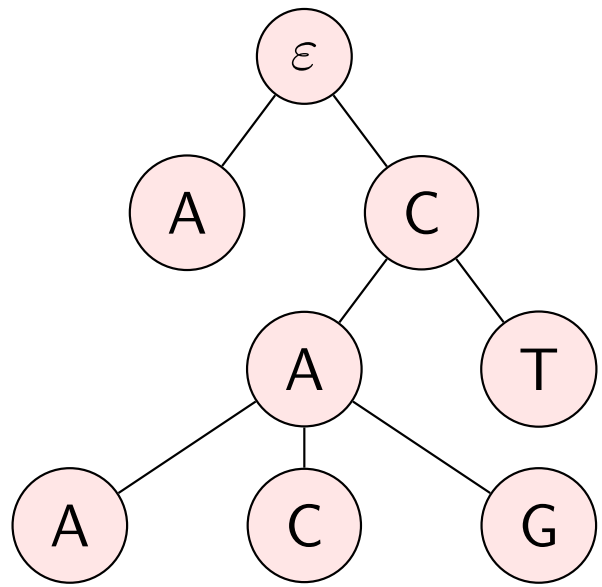


Main idea

Aim

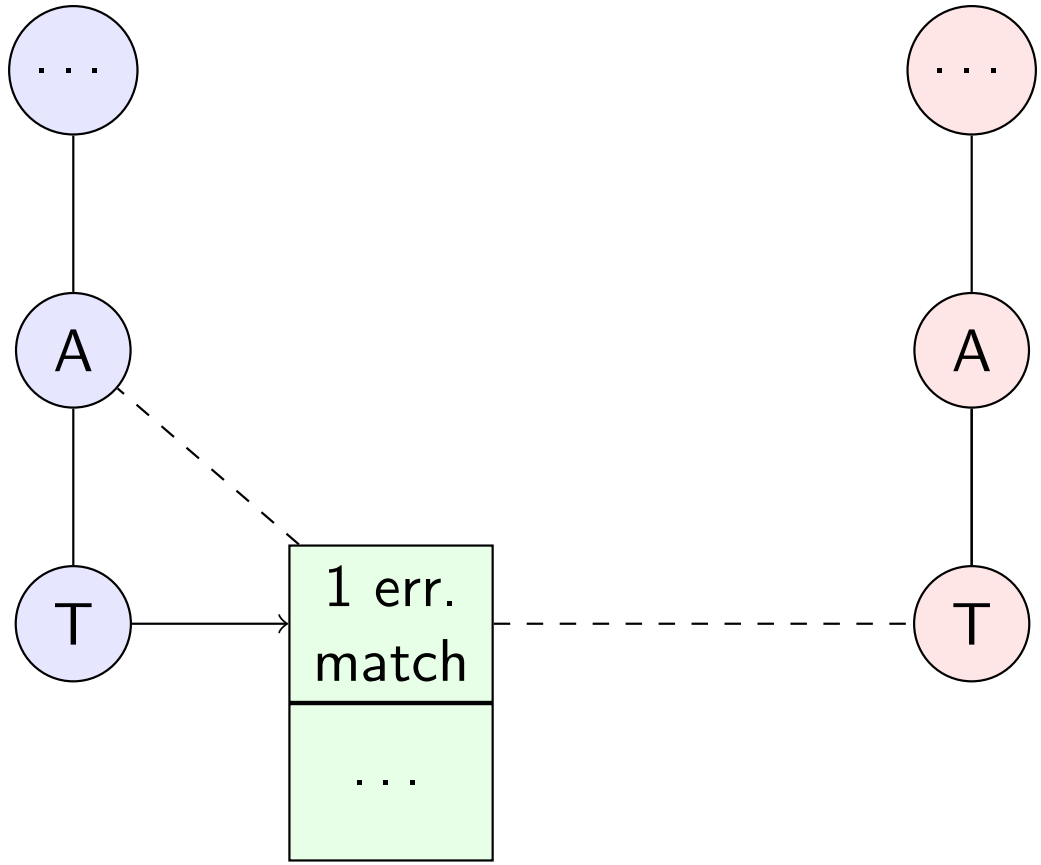
- For each accepting "read node," compute the all the "genome nodes" with minimum distance not greater than k.
- For each "reads node," compute recursively the all the "genome nodes" with distance not greater than k.





Note: The genome tree here is not an actual suffix tree. It is just presented as an illustration.

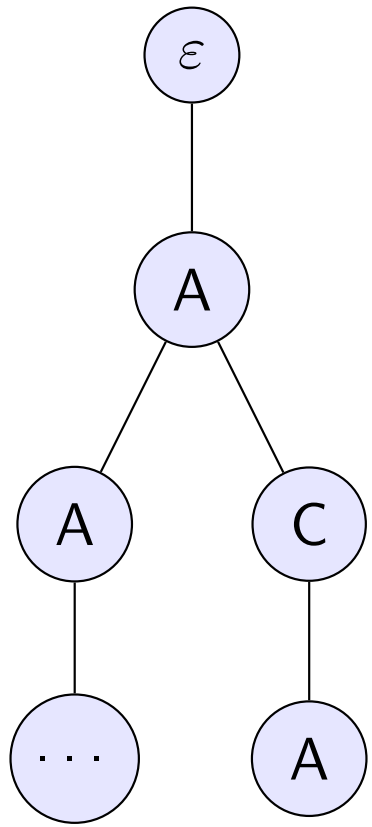
Implementation



Optimization 1

Expect a 0-error mapping first

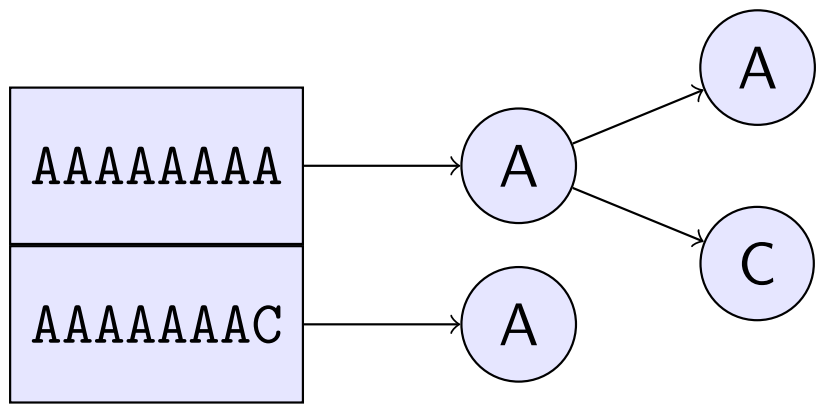
- Map with no error first.
- In case of error at depth d, add an error up to depth d.



Optimization 2

The genome tree is a vector of 4^8 trees

- The first tree is labelled `AAAAAAAA`.
- The second tree is labelled `AAAAAAAC`.
- etc.
- Each tree starts at depth 8.

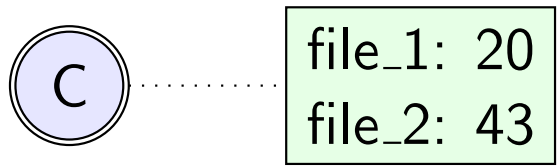


Other optimizations

Remove low complexity reads

`ACACACACACA`

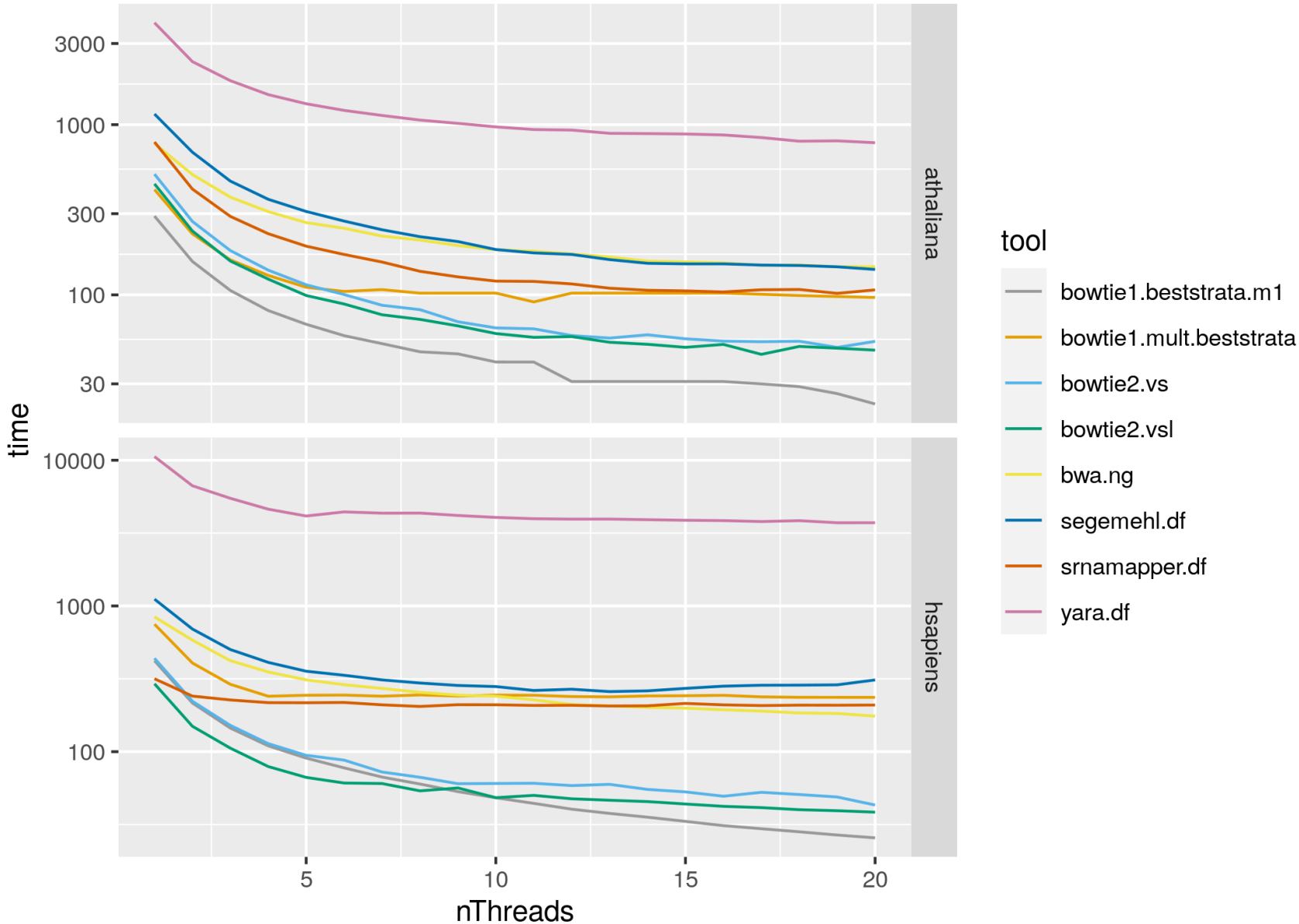
Can process several reads files



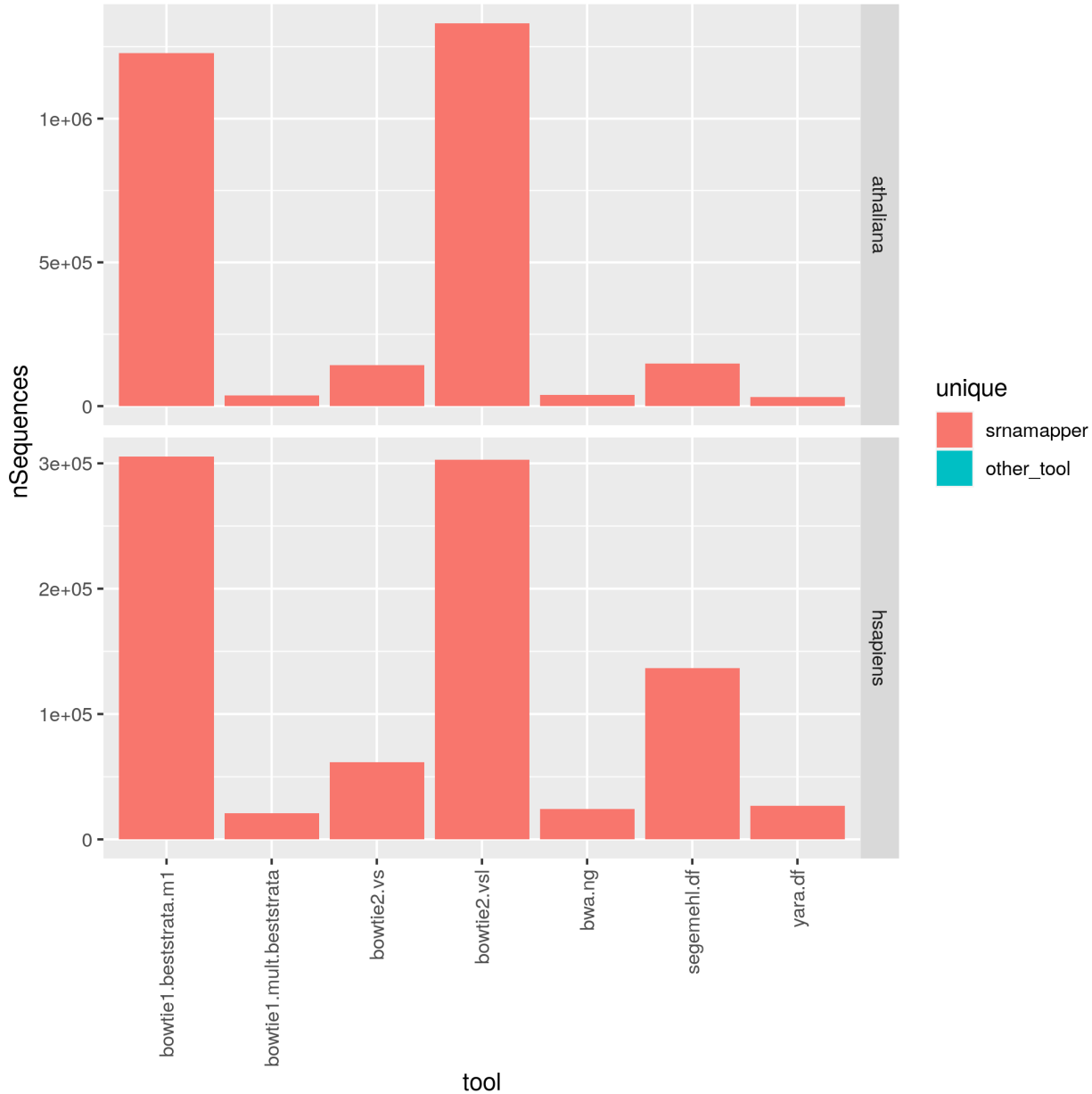
Can use several threads

- Mapping: The mapping threads traverse distinct parts of the reads trees.
- Reads tree construction: 1 tree for each thread, which are merged afterwards.

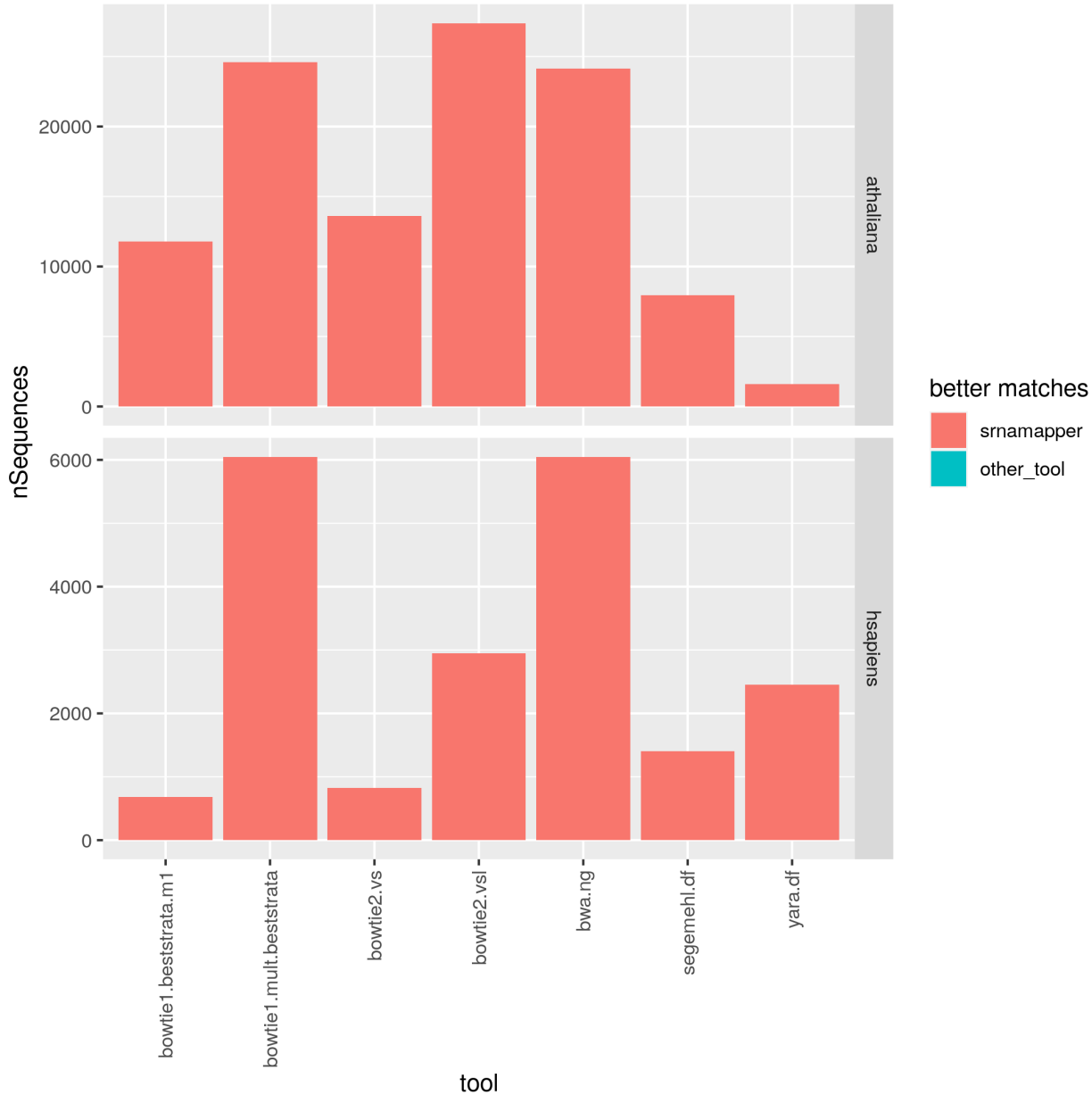
Results — Time



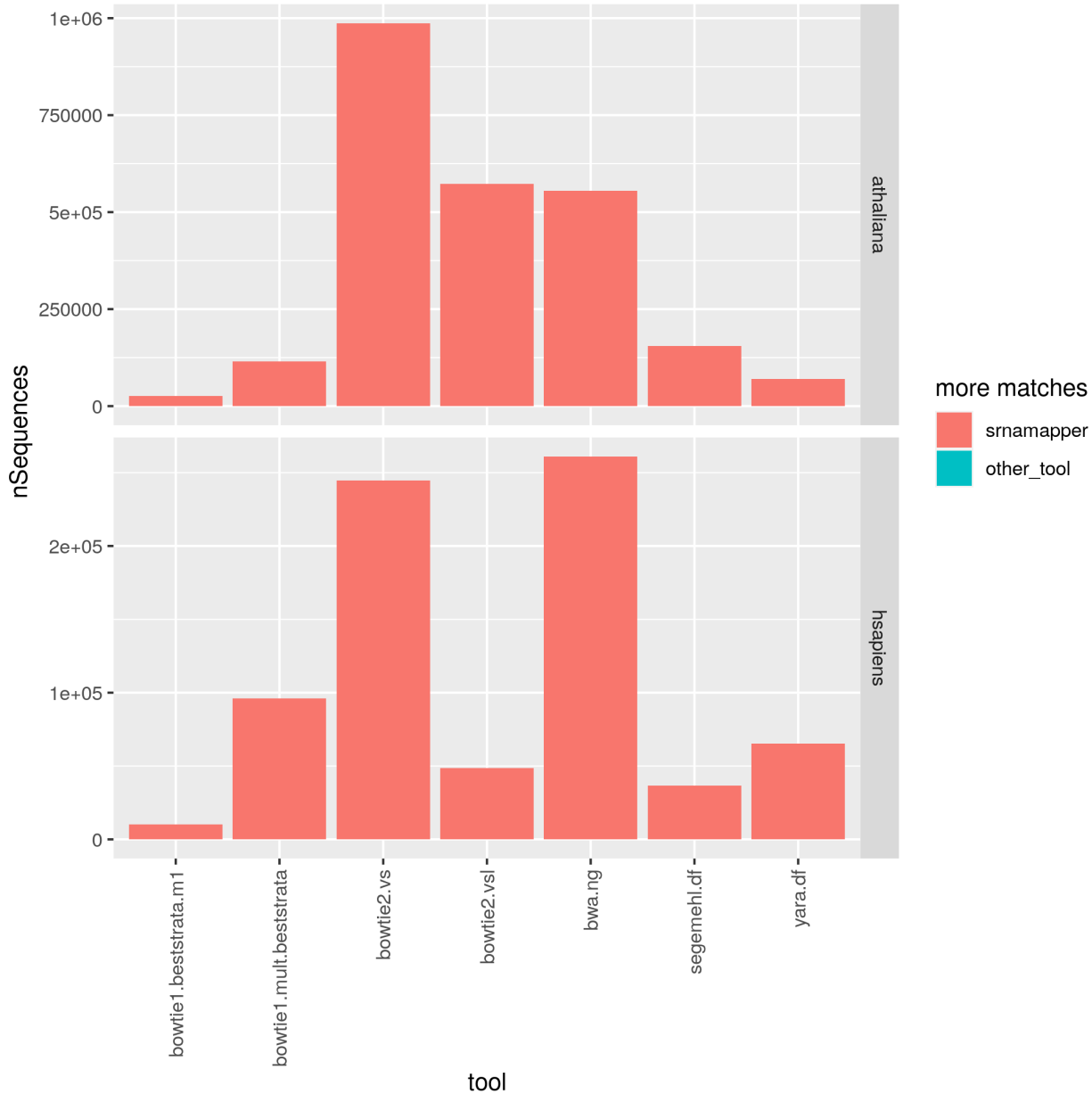
Results — comparison of # sequences



Results — comparison of # better hits

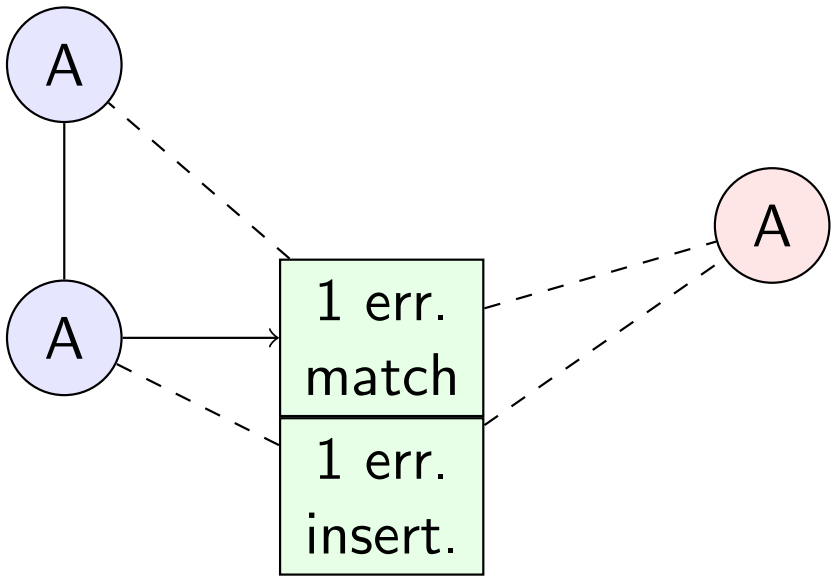


Results — comparison of # hits



Problem

states increase



Compare to dynamic programming

	ε		A		A
ε	0	$\rightarrow$	1	$\rightarrow$	2
	$\downarrow$	$\searrow$		$\searrow$	
A	1		0	$\rightarrow$	1

Bottom line

- You do not want *all* the mappings.
- How to implement a good # states *vs* states elimination balance?

Implementation details — Reads

First pass

- Edges contain the nucleotides (and the size), and the address to the following node.
- No predefined order.
- Each node contains 4 edges, the read counts, and the qualities.

Second pass

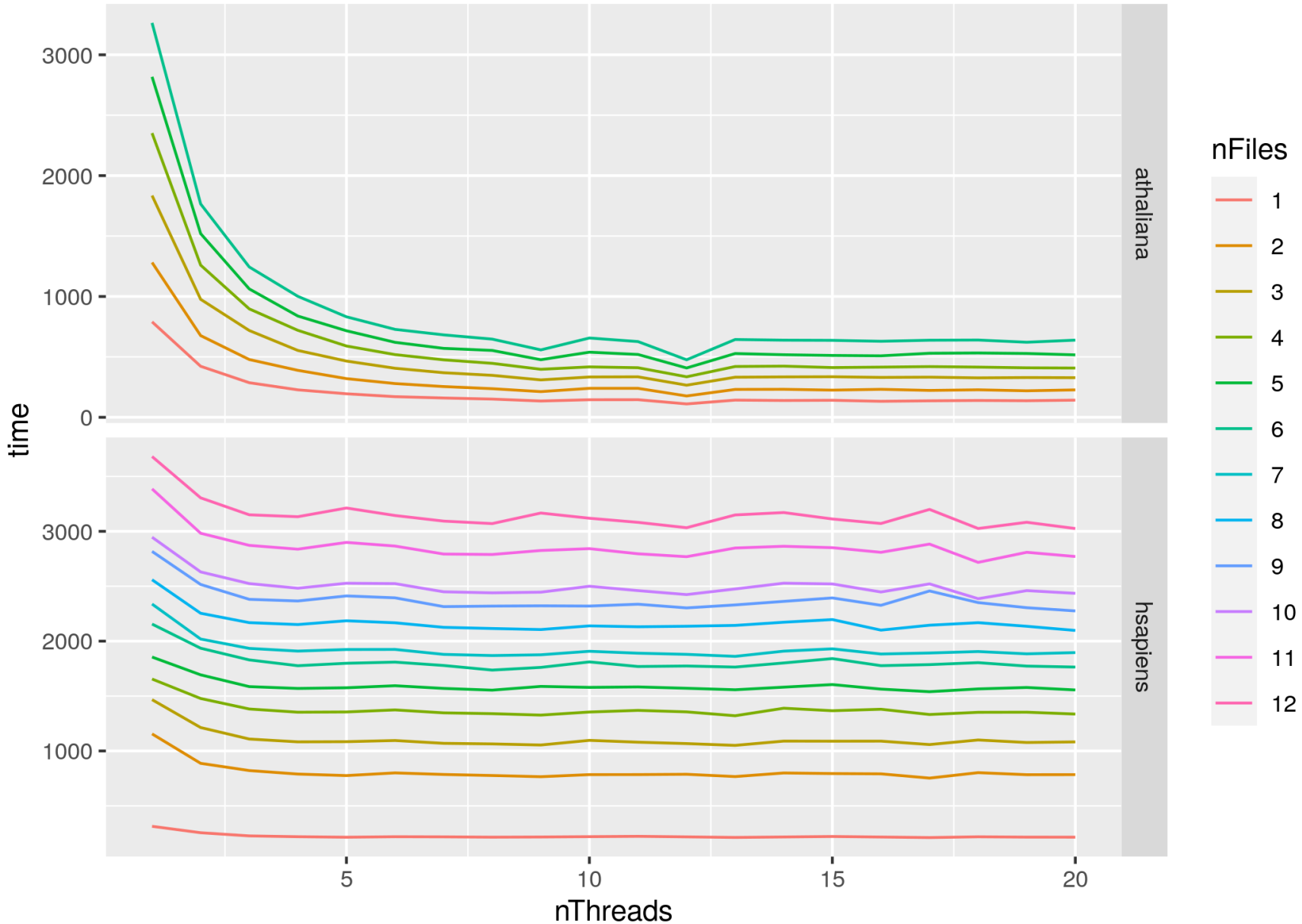
- Nodes are sorted in a depth-first fashion.
- Read counts and qualities are stored in a parallel vector.

That's all, folks!

Available at `https://github.com/mzytnicki/srnaMapper`

Thank you for your attention!

Results — time *vs* # files



Results — # mismatches