Aligning (& mapping) RNA-seq reads

Rob Patro
Aug. 15, 2016

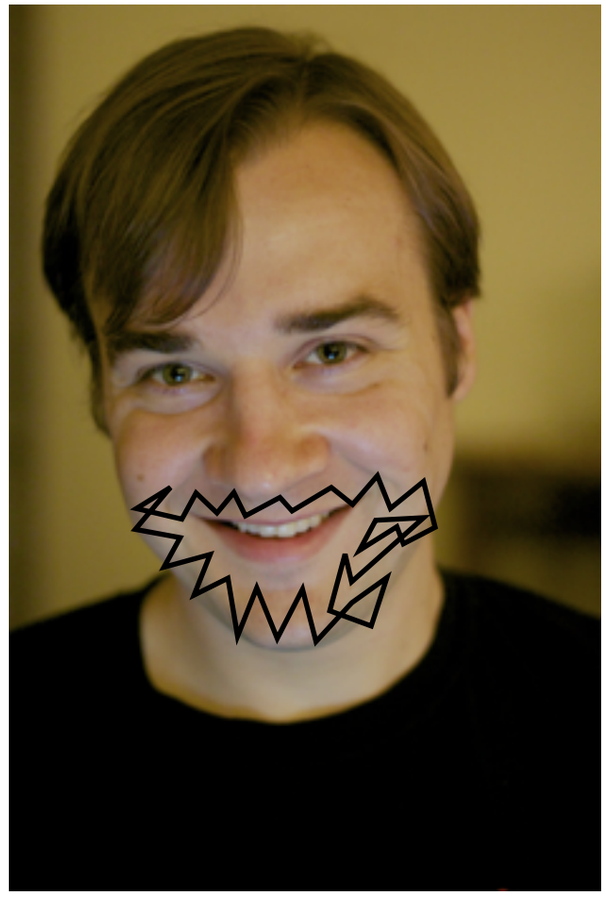
About me:

My Uni: Stony Brook University

My Lab: OMBINE

Twitter: @nomad421 **GitHub**: rob-p **Website**: robpatro.net

Grad student — Postdoc — Asst. Prof



* actually, my son, Isaac ;)

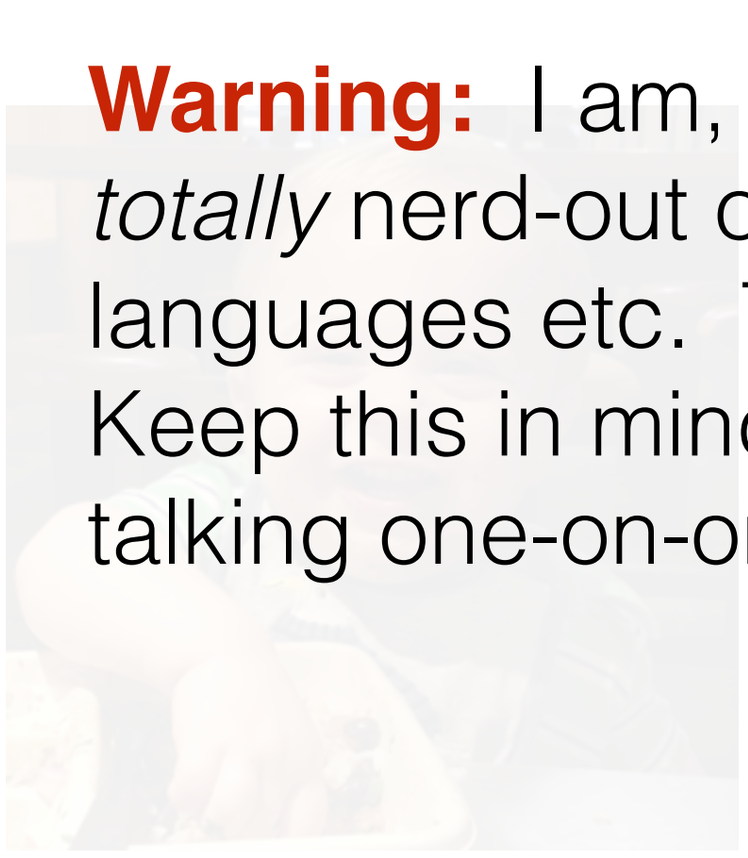
About me:

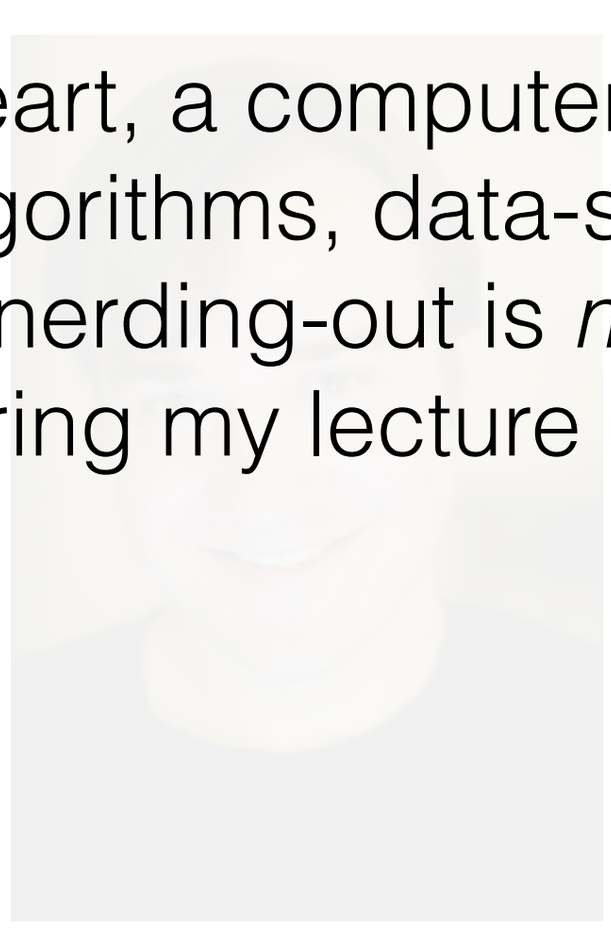
My Uni: Stony Brook University

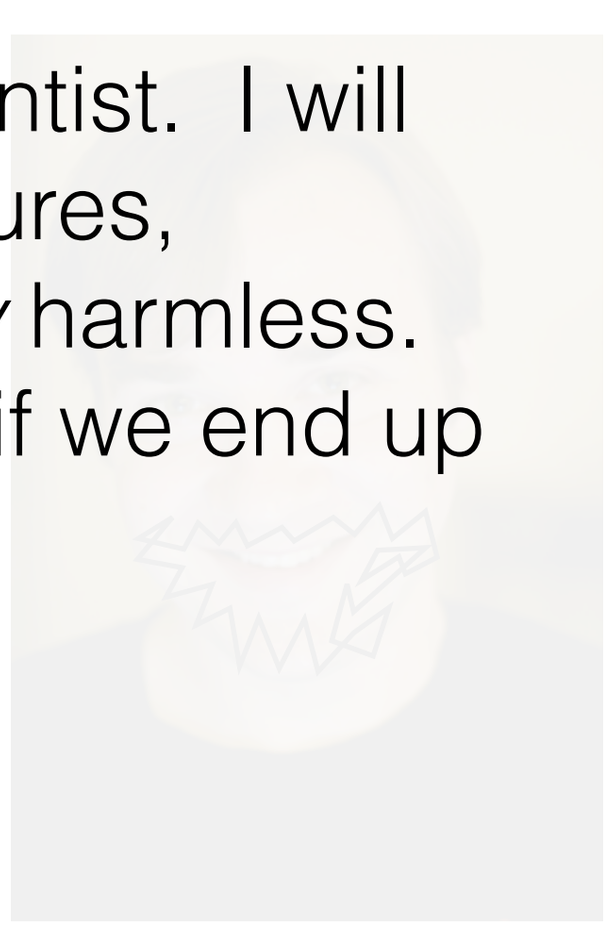
My Lab: OMBINE

Twitter: @nomad421 **GitHub**: rob-p **Website**: robpatro.net

Grad student Postdoc Asst. Prof





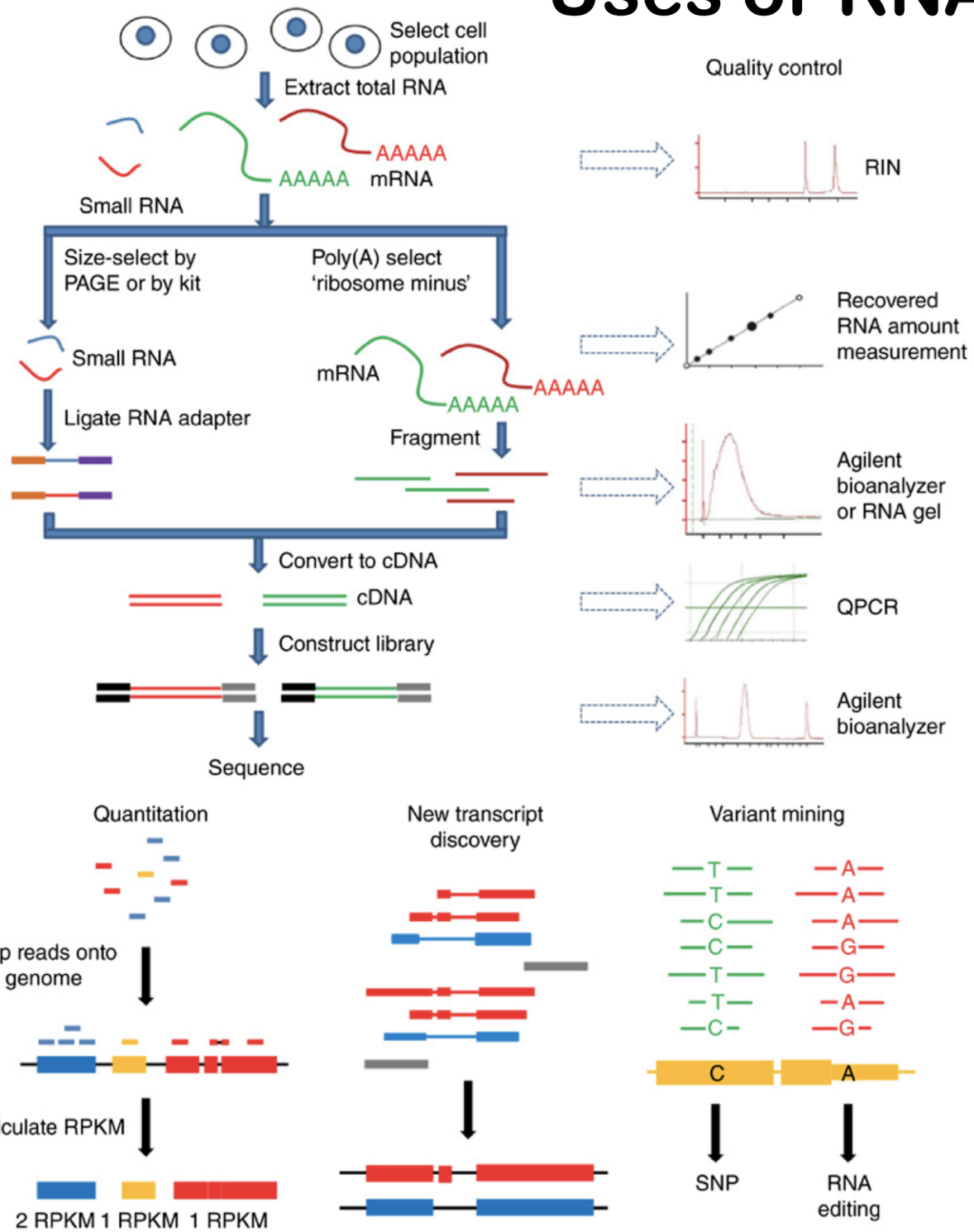


Warning: I am, at heart, a computer scientist. I will *totally* nerd-out on algorithms, data-structures, languages etc. This nerding-out is *mostly* harmless. Keep this in mind during my lecture (and if we end up talking one-on-one).

* actually, my son, Isaac ;)

Zeng & Mortazavi, Nature Immunology 2012

Uses of RNA-Seq are manifold



Whole transcriptome analysis

- Quantification & differential expression
- Novel txp discovery
  - reference-based
  - *de novo*
- Variant detection
  - Genomic SNPs
  - RNA editing

- What is dynamic & changing over time (as disease progresses)?
- What is tissue specific (in fetal development but not after)?
- What is condition specific (under stress conditions vs. not)?

Sequencing Reads

pre-proc.

align / map to ref.

txp. identification

reference-based

pre-proc.

de novo assembly

align / map to assembly

de novo

quantification

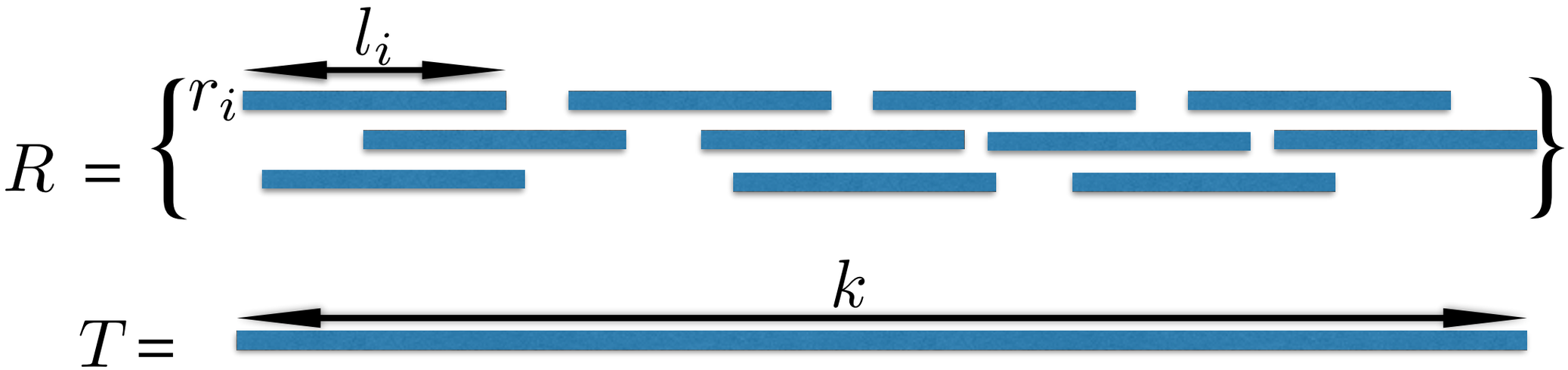
DE, Alternative Splicing, etc.

“Higher-level” analysis

What is the alignment problem?

Given: A collection of sequencing reads, and some target sequence (e.g. a genome)

Find: For each read, all locations where the read is within edit distance ϵ of the reference, and the edits that achieve this distance.



$d : \Sigma^{|u|} \times \Sigma^{|v|} \rightarrow \mathbb{Z}$ and ϵ (maximum edit distance)

Edit Distance

Given: Two strings

$$a = a_1 a_2 a_3 a_4 \ldots a_m$$
$$b = b_1 b_2 b_3 b_4 \ldots b_n$$

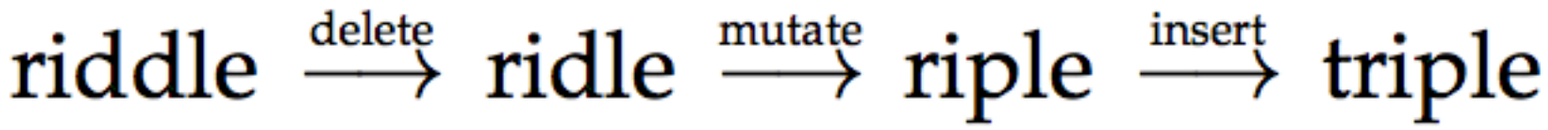
where a_i, b_i are letters from some alphabet, Σ, like {A,C,G,T}.

Compute how similar the two strings are.

What do we mean by "similar"?

Edit distance between strings a and b = the smallest number of the following operations that are needed to transform a into b:

- mutate (replace) a character
- delete a character
- insert a character

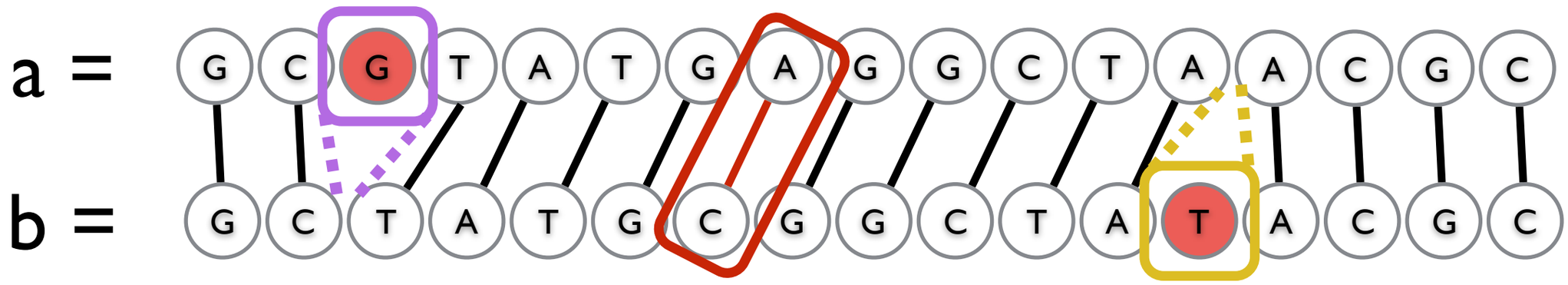


slide courtesy of Carl Kingsford

Another View: Alignment as a Matching

Each string is a set of nodes, one for each character.

Looking for a low-cost matching (pairing) between the sequences.



The operations at our disposal

Insertion (into **a** ~ deletion from **b**)

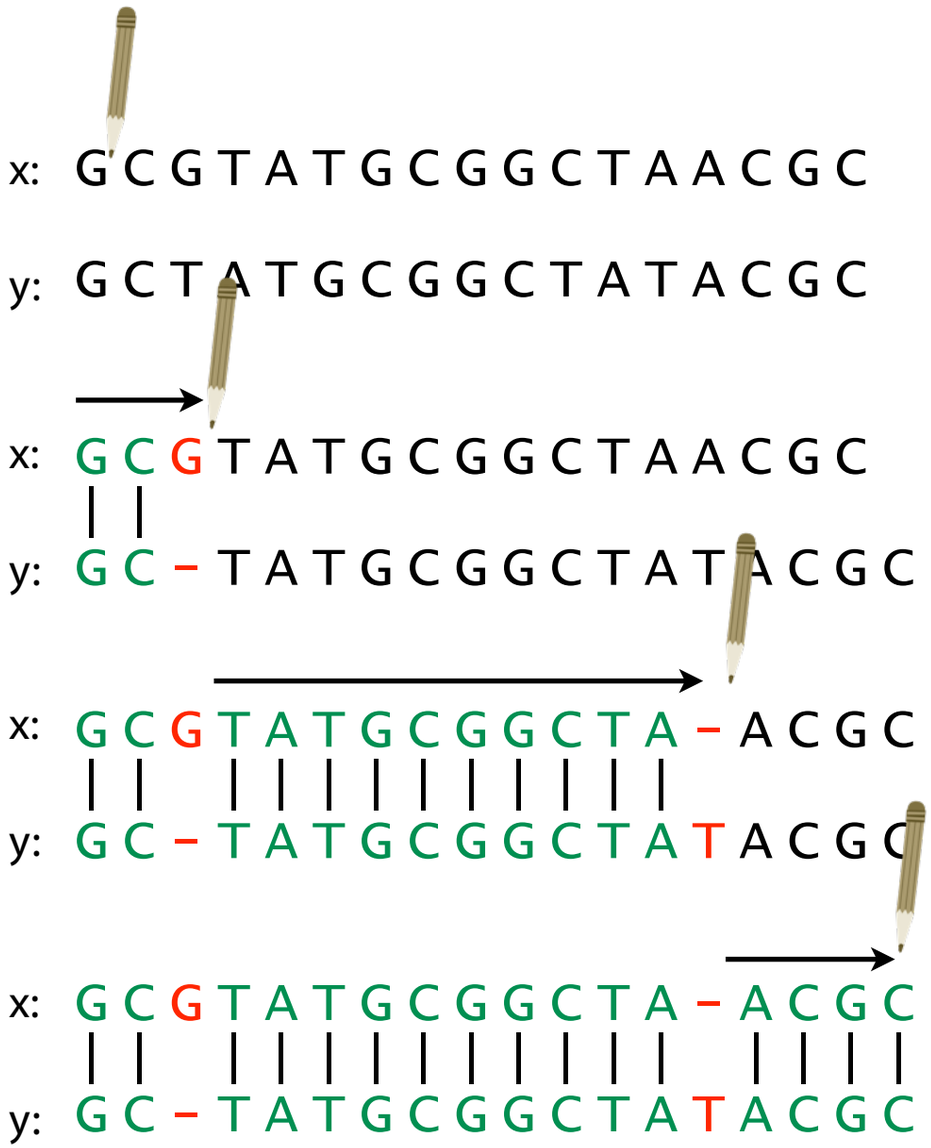
Mutation

Deletion (from **a** ~ insertion into **b**)

When we "delete a" character in **a** this is the same as inserting the character "-" in **b**. Conceptually, you can think of this as aligning the deleted character with "-". Under this model cost(x,'-') = cost('-',x) = gap for any $x \in \Sigma$

Representing alignments as edit transcripts

Can think of edits as being introduced by an *optimal editor* working left-to-right. *Edit transcript* describes how editor turns *x* into *y*.



Operations:

M = match, **R** = replace,

I = insert into *x*, **D** = delete from *x*

MMD

MMDMMMMMMMMMMI

MMDMMMMMMMMMMIMMMM

Slide courtesy of Ben Langmead

Representing edits as alignments

```
prin-ciple
|||| |||xx
prinncipal
(1 gap, 2 mm)
MMMMIMMMRR
```

```
prin-cip-le
|||| ||| |
prinncipal-
(3 gaps, 0 mm)
MMMMIMMMIMD
```

```
misspell
||| ||||
mis-pell
(1 gap)
MMMIMMMM
```

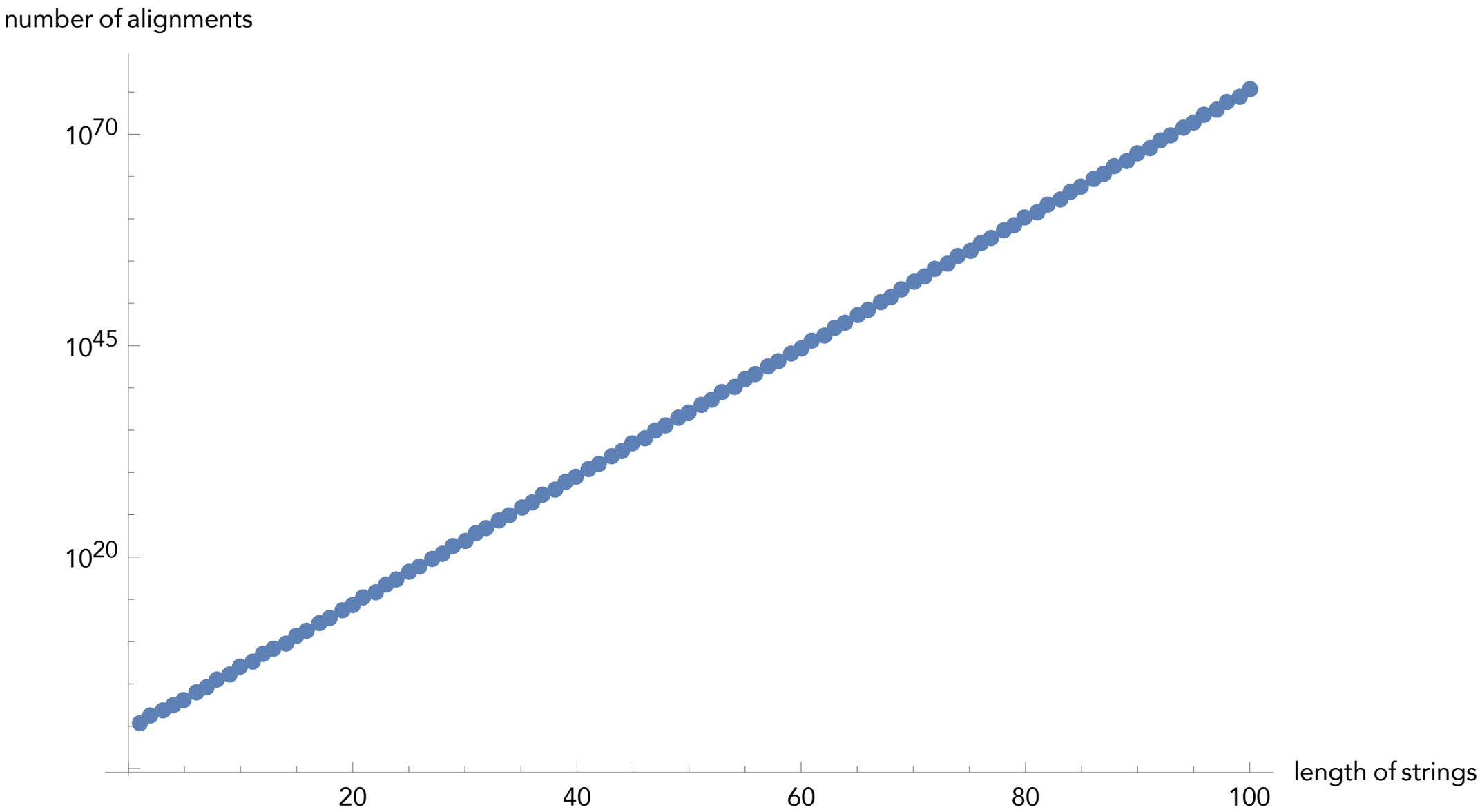
```
prehistoric
   ||||||||
---historic
(3 gaps)
DDDMMMMMMMM
```

```
aa-bb-ccaabb
|x || | | |
ababbbc-a-b-
(5 gaps, 1 mm)
MRIMMIMDMDMD
```

```
al-go-rithm-
|| xx ||x |
alKhwariz-mi
(4 gaps, 3 mm)
MMIRRIMMRDMI
```

slide courtesy of Carl Kingsford

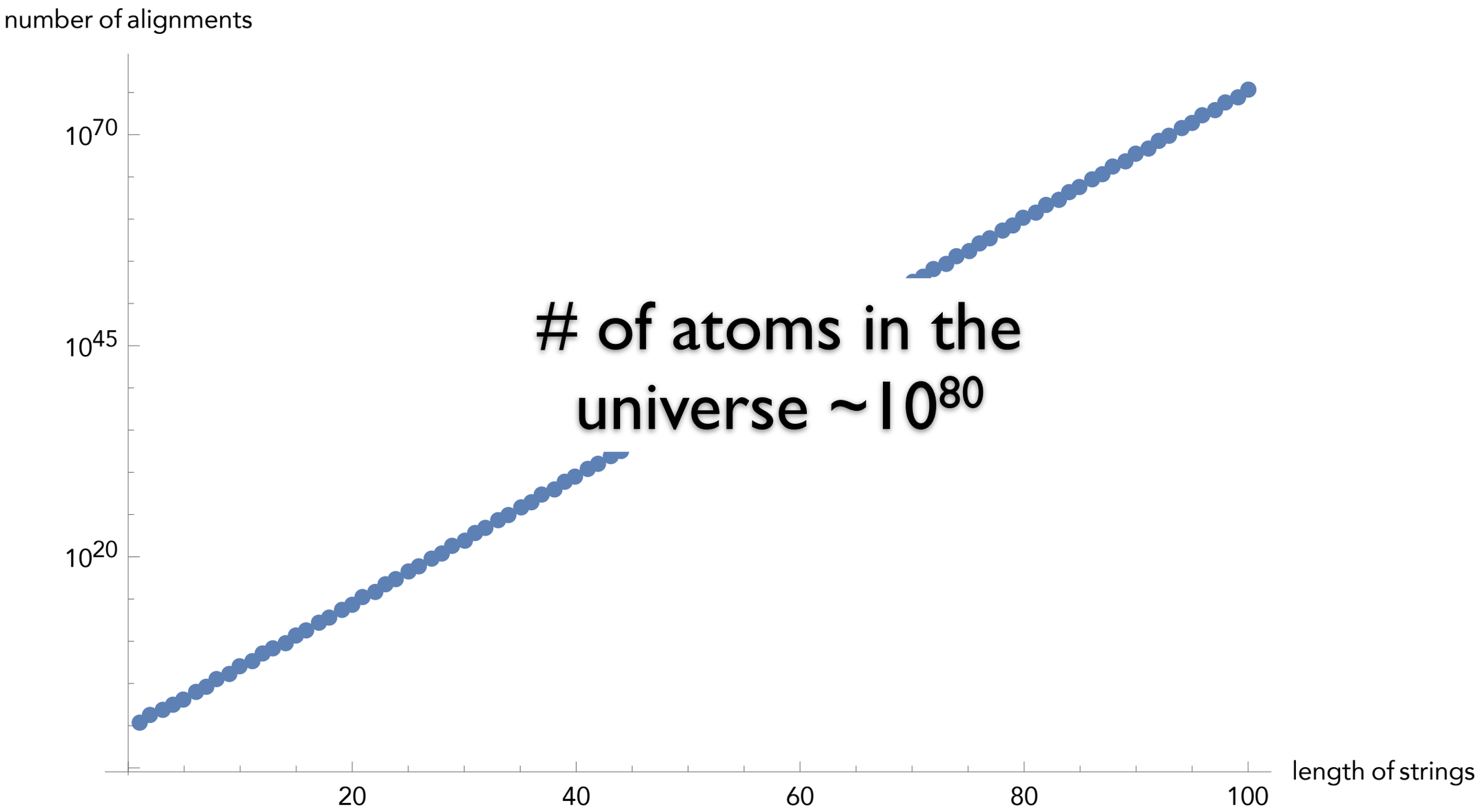
Can’t we just test and choose the best?



$$f(n,m) = \sum_{k=0}^{\min(m,n)} 2^k \binom{m}{k} \binom{n}{k}$$

Andrade, Helena, et al. "The number of reduced alignments between two DNA sequences." BMC bioinformatics 15.1 (2014): 94.

How many alignments are there?



$$f(n,m) = \sum_{k=0}^{\min(m,n)} 2^k \binom{m}{k} \binom{n}{k}$$

Andrade, Helena, et al. "The number of reduced alignments between two DNA sequences." BMC bioinformatics 15.1 (2014): 94.

Algorithms to the rescue!

Luckily, we can do *much* better than brute force.

Optimal edit distance can be computed in *quadratic* time O(|x|*|y|)[1], using algorithms that rely on dynamic programming.

Unfortunately, solving the alignment problem optimally is often *still* to slow. But *heuristics* get us results that are often very good, and intelligent algorithms and data structures let us find these solutions *fast!*

1: We'll briefly unpack what this means next

A brief primer on “Big O” notation

To understand what is “feasible” or not, in terms of alignment, we have to think about how different approaches perform relative to the *size of the input*.

Computer scientists often use “big O” notation to represent such a notion: i.e.
How does the number of computations required to run an algorithm scale with the input size?

big O notation asks how algorithms perform *asymptotically* (in the limit of infinite input). It is not strictly a measure of performance, but it provides information relevant to performance.

Let’s consider some examples

A brief primer on “Big O” notation

Consider the following “algorithms”

Compute the sum of a list of numbers

Algorithm 1:

```
sum = 0
for elem in L:
    sum += elem
```

How many “steps”

for N = 100	100
for N = 1000	1000
for N = 10000	10,000
for N = 100000	100,000

Compute the sum of *all subranges* of a list of numbers

Algorithm 2:

```
sum = 0
for start in [0, length(L)]:
  subsum = 0
  for stop in [start, length(L)]:
    subsum += elem
  sum += subsum
```

for N = 100	4,950
for N = 1000	499,500
for N = 10000	49,995,000
for N = 100000	4,999,950,000

A brief primer on “Big O” notation

Consider the following “algorithms”

Compute the sum of a list of numbers

Algorithm 1:

```
sum = 0
for elem in L:
    sum += elem
```

How many “steps”

Algorithm 1 is in O(N):
It scales with N

Compute the sum of *all subranges* of a list of numbers

Algorithm 2:

```
sum = 0
for start in [0, length(L)]:
  subsum = 0
  for stop in [start, length(L)]:
    subsum += elem
  sum += subsum
```

Algorithm 1 is in $O(N^2)$:
It scales with N^2

Bonus question: What is the exact # of steps?

A brief primer on “Big O” notation

Consider the following “algorithms”

What is the big-O of **Algorithm 3**?

Algorithm 3:

```
sum = 0
for start in [0, length(L)]:
  subsum = 0
  for stop in [start, length(L)]:
    subsum += elem
  sum += subsum

sum2 = 0
for elem in L:
    sum2 += elem
return sum + sum2
```

A brief primer on “Big O” notation

Consider the following “algorithms”

Algorithm 3:

```
sum = 0
for start in [0, length(L)]:
  subsum = 0
  for stop in [start, length(L)]:
    subsum += elem
  sum += subsum

sum2 = 0
for elem in L:
    sum2 += elem
return sum + sum2
```

What is the big-O of **Algorithm 3**?

Algorithm 3 is still in $O(N^2)$:

As N grows large, N^2 dominates N, so the final for loop contributes negligibly to the total number of steps. Big-O is only concerned with the highest-order terms so that e.g.:

N^3 dominates N^2 dominates N etc.

Primer over; back to alignment

So, how is this relevant to alignment?

The best algorithms we have (and like the best that could exist) to compute the optimal alignment of two strings are *quadratic*

If we have N reads, each of length ℓ, and the genome is of length L, then applying the optimal algorithm at each possible position (to test the edit distance) is **O(N·ℓ·L)**

Consider a dataset with:

N = 20 x 10^6 reads

ℓ = 100

L = 3x10^9 nucleotides

and a processor that can do X = 3 x 10^9 operations / sec.

You'd wait about (N·ℓ·L) / X = 200,000,000 sec = 6.34 **years** to align your reads. If you think a Ph.D. is slow now . . .

Primer over; back to alignment

Note: This analysis is a worst-case scenario, where we apply the most naive algorithm possible to solve this problem.

Still, solving the alignment problem as stated above takes too long to be used in practice.

Thus, most tools resort to *heuristics* — approaches that usually work well, but may not return optimal results.

Specifically, there may be positions on the genome to which a read can be aligned with edit distance $\leq \epsilon$, that are not found (i.e. false negatives). False positives are also possible, but usually resort from finding sub-optimal alignments where better ones exist, and rely on a slightly different formulation of the alignment "problem".

Keep this in mind as we discuss the methods below.

Phylogeny of Read-Alignment

Aligning (Mapping) NGS Reads

DNA-sequencing

Genome (Spliced)

- Aligns RNA-seq reads to genome
- Challenge of **Spliced Alignment**
- Example: topHat, STAR, HISAT(1/2)

RNA-sequencing

Transcriptome

- Aligns RNA-seq reads to transcriptome
- Challenge of **high multi-mapping rate**
- Example: Bowtie(1/2), BWA(SW/MEM)

Aligner

- Base-to-Base Alignment (CIGAR string)

Mapper

- **NO** CIGAR string

RNA-Seq Read Alignment

Given an RNA-seq read, where *might* it come from?

Two main “regimes”

Align to transcriptome

Align reads directly to txps

No “split” alignments — transcripts contain spliced exons directly.

Typically *a lot* of multi-mapping (80-90% of reads may map to multiple places)

Does *not r*equire target *genome*

Can be used in *de novo* context (i.e. after *de novo* assembly)

Align to genome

Align reads to target genome

Reads spanning exons will be “split” (gaps up to 10s of kb)

Typically little multi-mapping (most reads have single genomic locus of origin)

Requires target *genome*

Can be used to find new transcripts

RNA-Seq Read Alignment

Given an RNA-seq read, where does it come from?

Two main “regimes”

Align to transcriptome

Main computational challenge comes from ubiquitous multi-mapping.

Bowtie

Bowtie 2

BWA

STAR

HISAT (1&2)

…

Align to genome

Main computational challenge comes from spliced alignments.

Top Hat

STAR

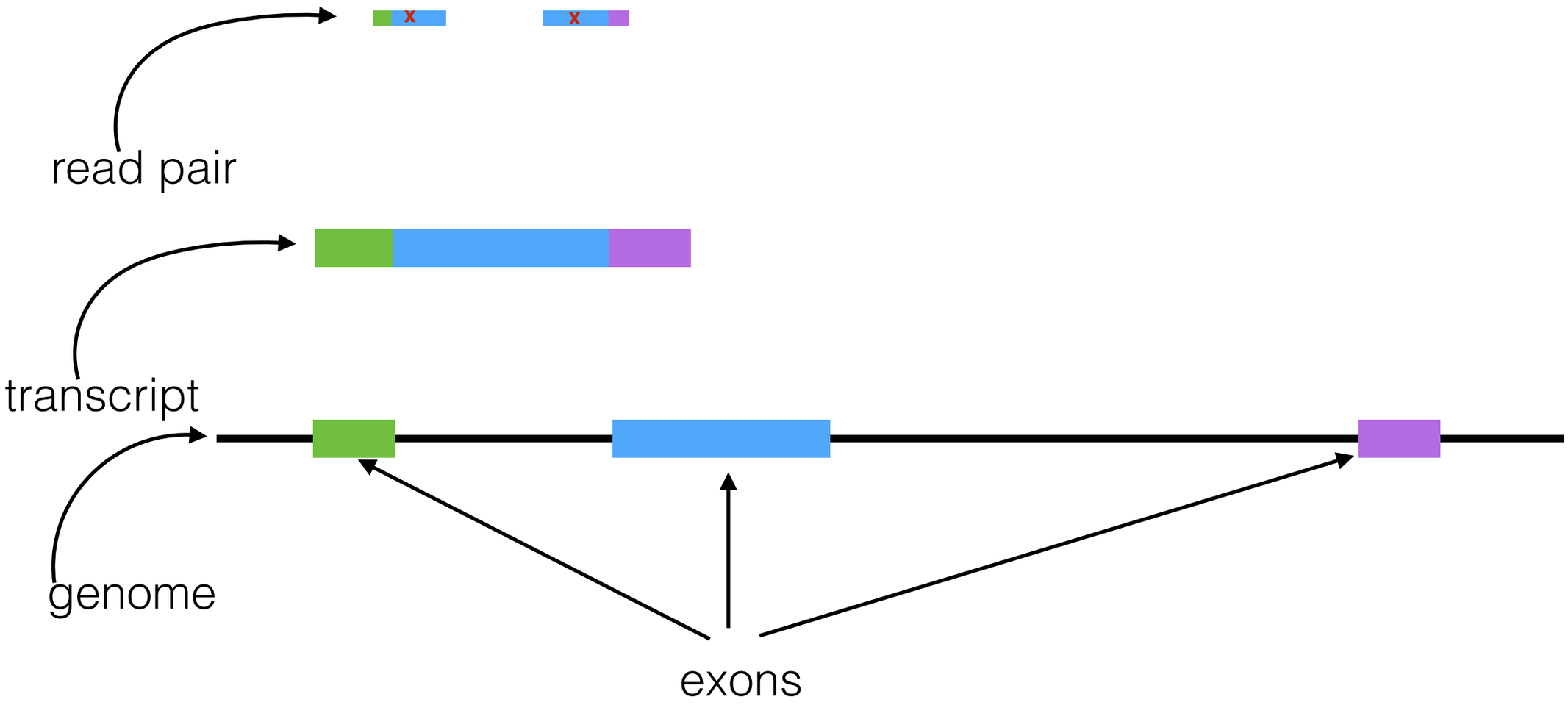
HISAT (1&2)

Map Splice

Subread Aligner

…

Spliced Alignment

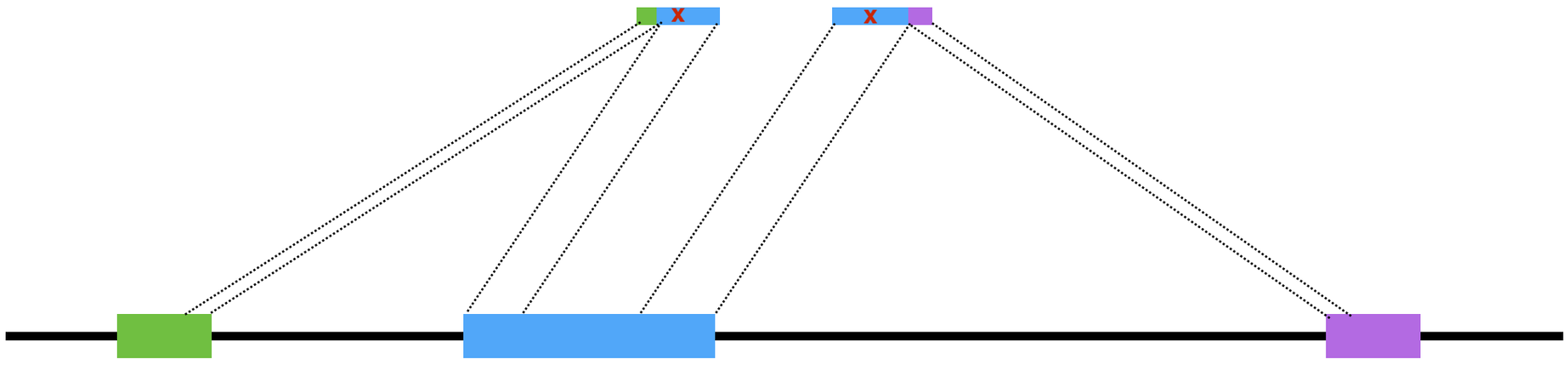


Spliced Alignment

Alight this

back to this

Spliced Alignment

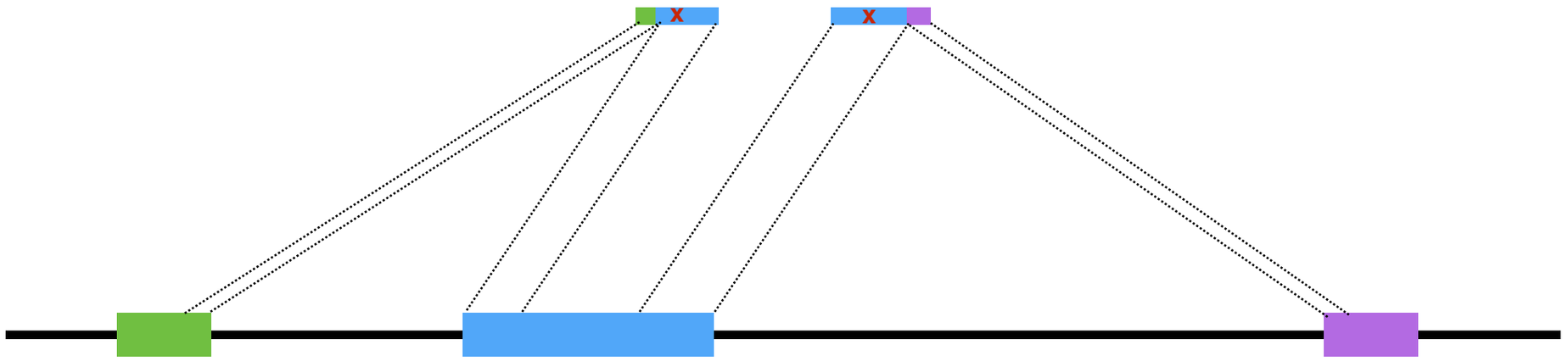


Splice junctions might be known, or *unknown.*

Overlap of read with exon may be *very short*, sequence is ambiguous (e.g. 10 bases).

Sequence of read might be repetitive in the genome.

STAR



Finds long, exact matches using a suffix array.

Potential alignment represented by a chain of MMPs (Maximum Mappable Prefixes).

Alignment validated and processed using dynamic programming algorithm.

STAR

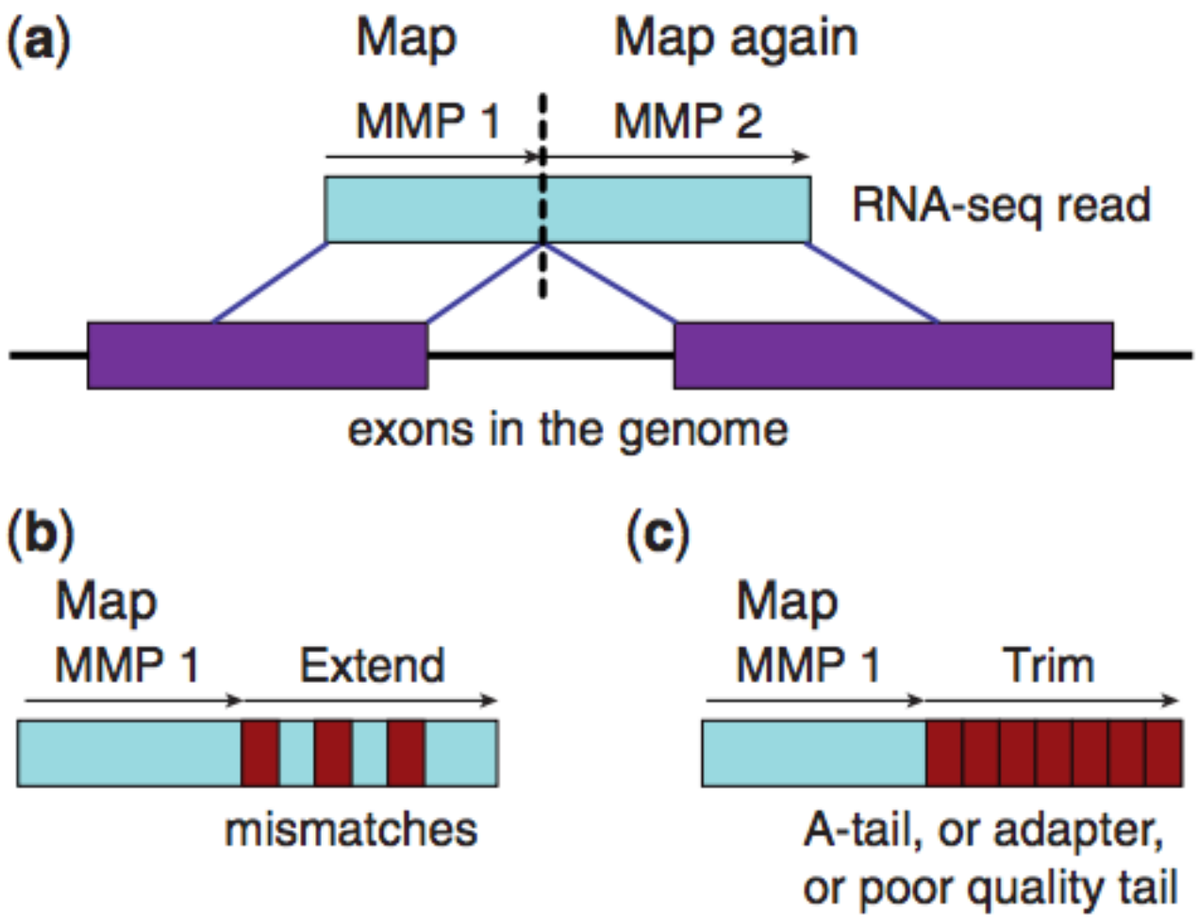


Fig. 1. Schematic representation of the Maximum Mappable Prefix search in the STAR algorithm for detecting (**a**) splice junctions, (**b**) mismatches and (**c**) tails

Figure from: Dobin, Alexander, et al. "STAR: ultrafast universal RNA-seq aligner." Bioinformatics 29.1 (2013): 15-21.

STAR

Accuracy

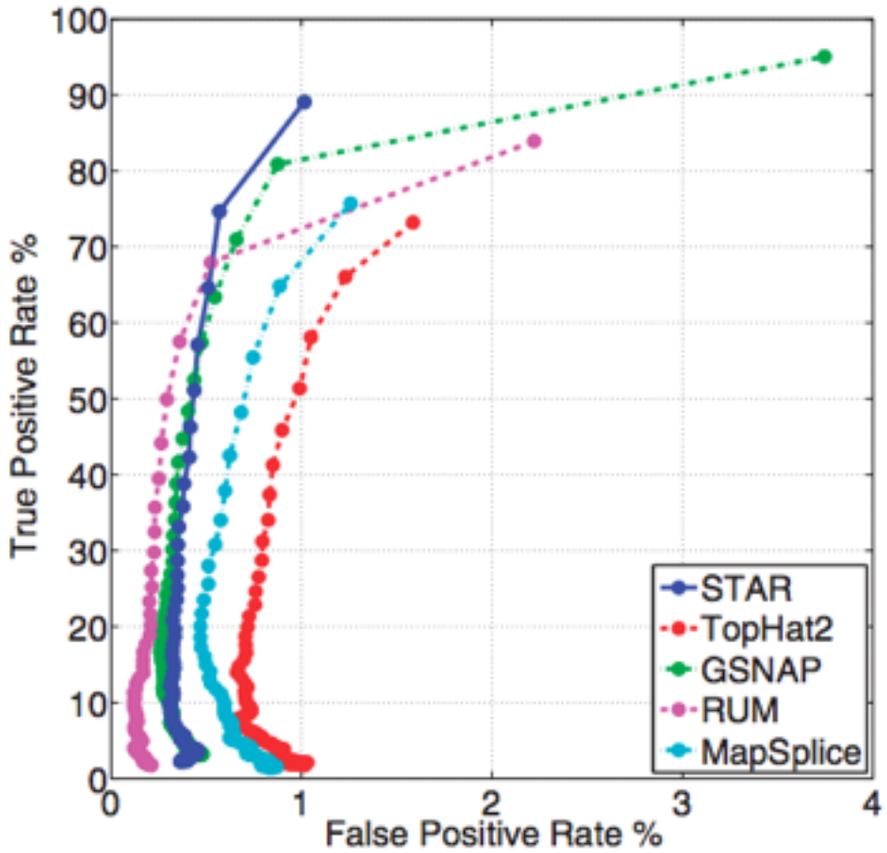


Fig. 2. True-positive rate versus false-positive rate (ROC-curve) for simulated RNA-seq data for STAR, TopHat2, GSNAP, RUM and MapSplice

Speed

Table 1. Mapping speed and RAM benchmarks on the experimental RNA-seq dataset

Aligner	Mapping speed: million read pairs/hour		Peak physical RAM, GB	
	6 threads	12 threads	6 threads	12 threads
STAR	309.2	549.9	27.0	28.4
STAR sparse	227.6	423.1	15.6	16.0
TopHat2	8.0	10.1	4.1	11.3
RUM	5.1	7.6	26.9	53.8
MapSplice	3.0	3.1	3.3	3.3
GSNAP	1.8	2.8	25.9	27.0

Figure from: Dobin, Alexander, et al. "STAR: ultrafast universal RNA-seq aligner." Bioinformatics 29.1 (2013): 15-21.

Modern RNA-Seq *Alignment*

When introduced, STAR was *much* faster than the alternatives. Recently, some new approaches have been developed.

HISAT — *Hierarchical FM-index*

- speed of STAR in an order-of-magnitude less memory
- 1.5-pass alignment to improve align to novel junctions (in STAR now too)

HISAT2 — *Hierarchical Graph FM-index*

- Improved sensitivity
- Very cool / elegant representation of genome, transcripts (etc.)
- Can align to a *population* of genomes

Take home: TopHat (1/2) used to be the *de facto* aligner for RNA-seq -> genome alignment. Now there are *much* better solutions. Also, TopHat 2 has been deprecated in favor of HISAT[1].

1: Pertea, Mihaela, et al. "Transcript-level expression analysis of RNA-seq experiments with HISAT, StringTie and Ballgown." Nature Protocols 11, 1650–1667 (2016)

Aligning to the transcriptome

What if we know the transcripts ahead of time?

What if we don't have the target genome (*de novo) txome assembly*?

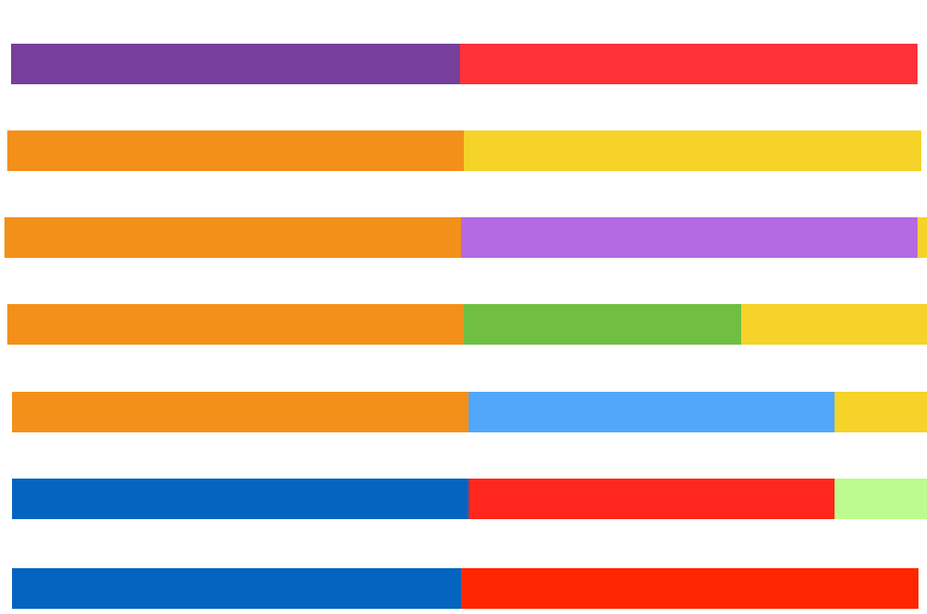
What if there are other transcripts that we know will be present in our sample, not from the target organism?

One way to handle such cases is to align *directly* to the transcriptome!

Aligning reads to a Transcriptome

Consider the following scenario:

Transcripts



Read

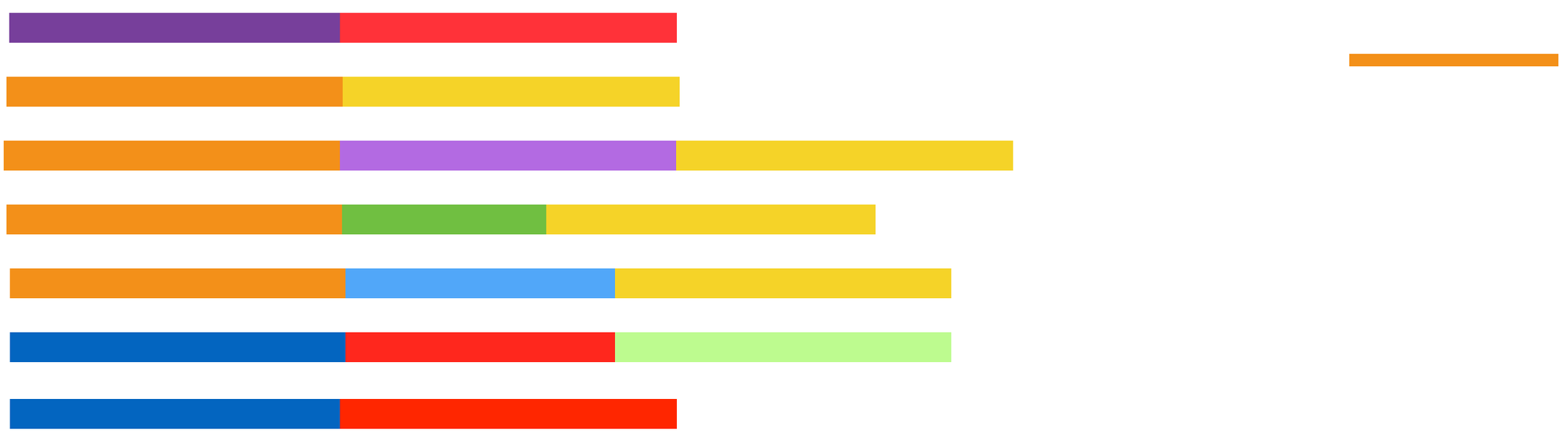
Aligning reads to a Transcriptome

Consider the following scenario:

Say that colors represent exonic sequence.
Intuitively, from where does the read originate?

Transcripts

Read



No more spliced alignment, but **prevalent** multi-mapping. Due, in part, to alternative splicing present in the transcriptome.

Problem: RNA-Seq Read ~~Alignment~~ Mapping to the Transcriptome

What if we don't *need* alignment?

> **Claim:** Some (but not all) of the analyses we're interested in performing may ***not actually require the read alignment***

How much more efficient may a solution be if we only care about ***where*** and not exactly ***how*** a read corresponds to the reference?

Validation: For a *very common analysis*, RNA-seq-based quantification and differential expression testing, we can replace alignment with mapping with virtually **no loss in accuracy.** We'll talk about Salmon, the tool that uses this idea, tomorrow.

RNA-Seq Read Alignment

Alignment is ***fast*** . . . but not always as fast as our data is ***big***

A single ***sample*** may contain 10s of millions of reads

An ***experiment*** may consist of many samples
e.g. conditions, time course samples, etc.

Condition A	Condition B	Condition C	Condition D	Condition E
Replicate 1	Replicate 1	Replicate 1	Replicate 1	Replicate 1
Replicate 2	Replicate 2	Replicate 2	Replicate 2	Replicate 2
Replicate 3	Replicate 3	Replicate 3	Replicate 3	Replicate 3
Replicate 4	Replicate 4	Replicate 4	Replicate 4	Replicate 4

A single *experiment* may easily consist of **100s of millions of reads.**

Quasi-mapping: A stand-in for alignment

Concept:

For a given fragment, a quasi-mapping specifies the ***target*** where a fragment "matches well", and the ***position***, and ***orientation*** of the fragment w.r.t the target, but *not details of the alignment*.

Algorithm:

Relies on a suffix array to compute the *Maximum Mappable Prefix* (MMP) and *Next Informative Position* (NIP) when mapping a read.

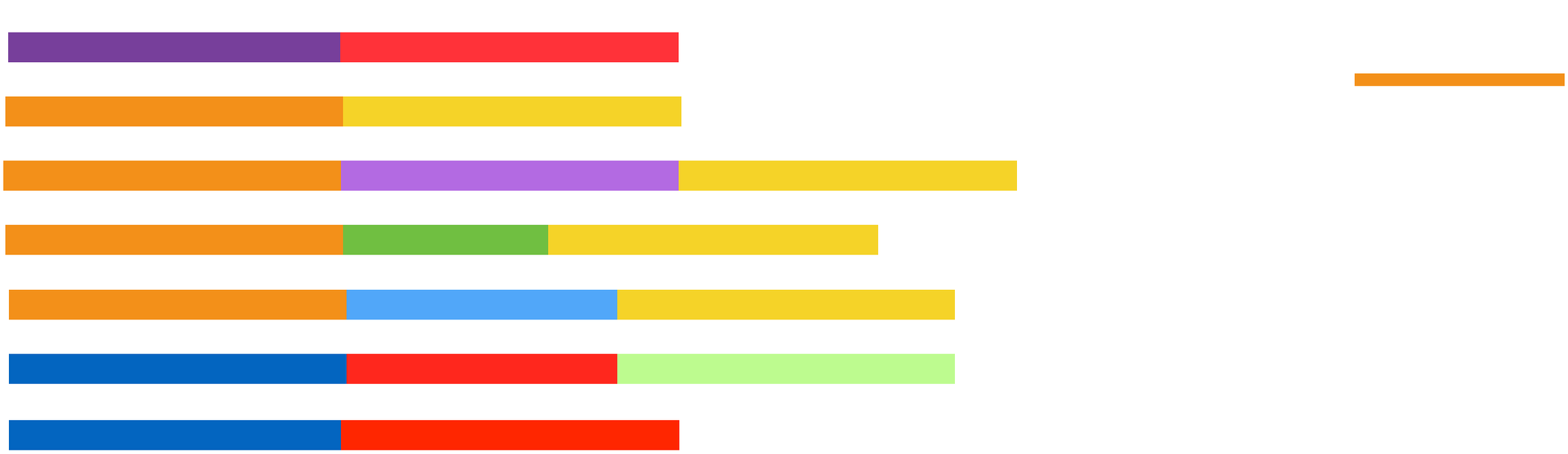
Given a carefully-designed algorithm, quasi-mapping information can be obtained *very* quickly.

Mapping reads to a Transcriptome

Consider the following scenario:

Transcripts

Read



Mapping reads to a Transcriptome

Consider the following scenario:

Say that colors represent exonic sequence.
Intuitively, from where does the read originate?

Transcripts

Read

Mapping reads to a Transcriptome

Consider the following scenario:

Say that colors represent exonic sequence.
Intuitively, from where does the read originate?
What about *this* read?

Transcripts

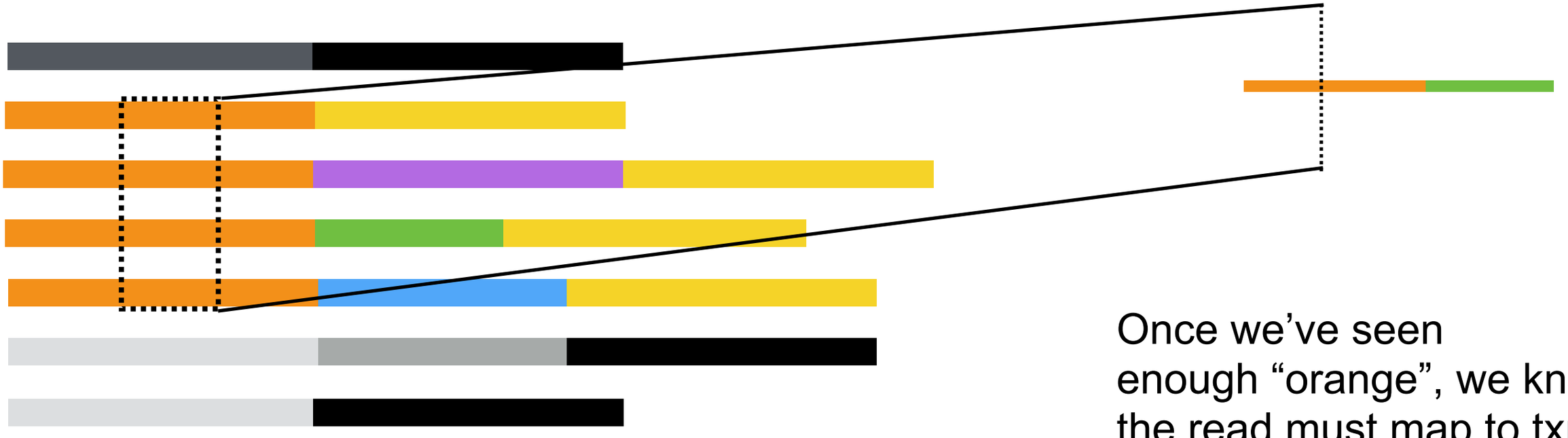
Read

Mapping reads to a Transcriptome

Consider the following scenario:

Transcripts

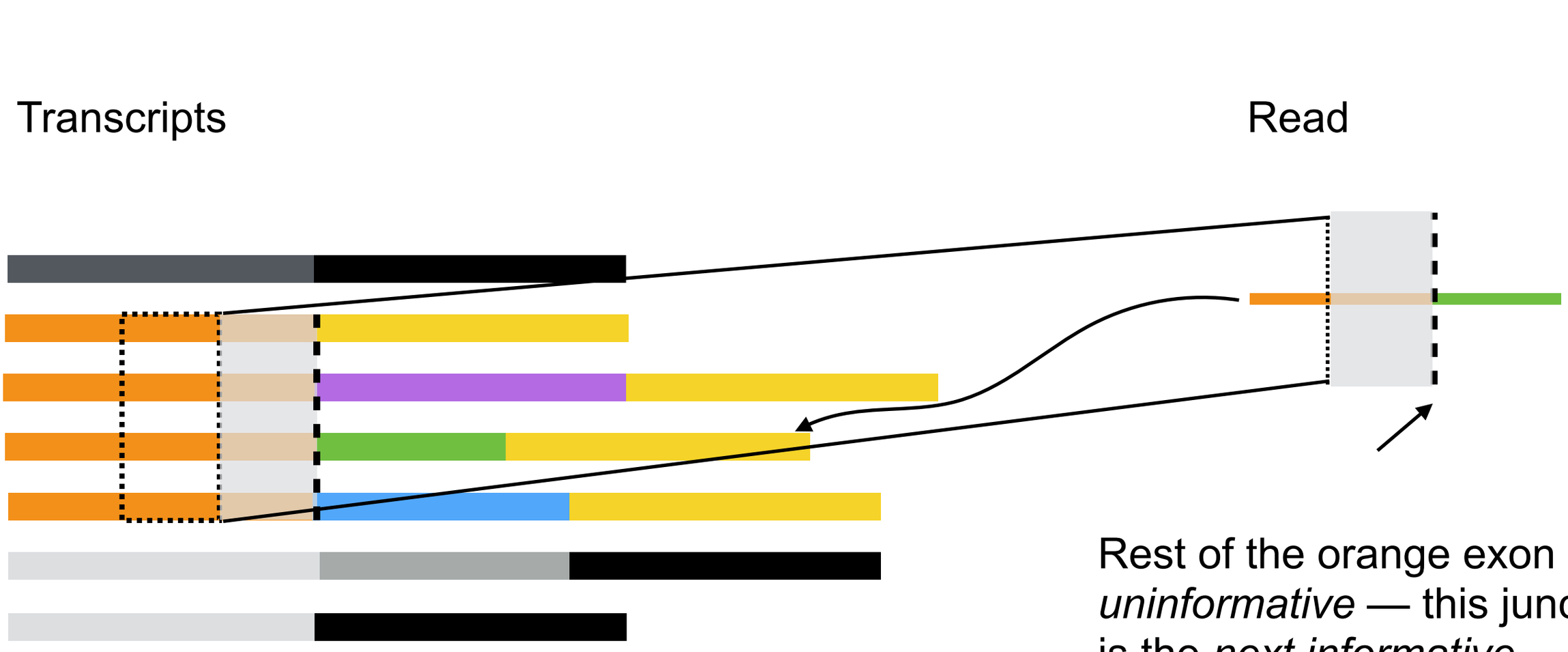
Read



Once we've seen enough "orange", we know the read must map to txps with this exon; but which one(s)?

Mapping reads to a Transcriptome

Consider the following scenario:

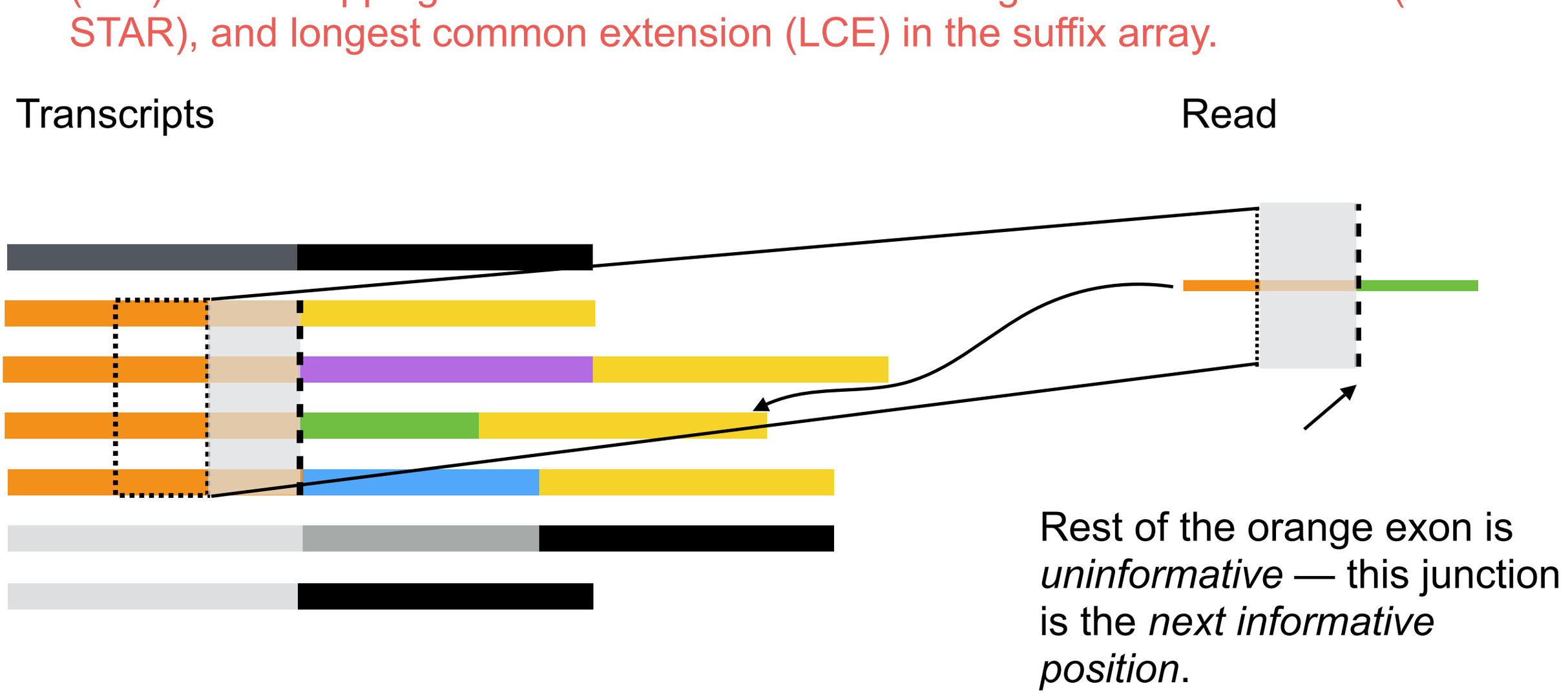


Rest of the orange exon is *uninformative* — this junction is the *next informative position*.

Mapping reads to a Transcriptome

Consider the following scenario:

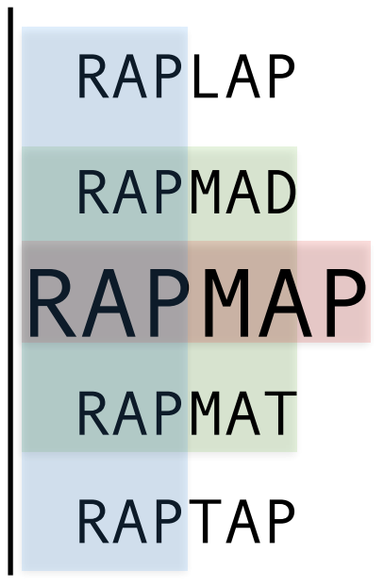
> Is there some ***general/formal*** way to always find the next informative position (NIP) when mapping a read? Yes — we do this using the notions of MMP (as in STAR), and longest common extension (LCE) in the suffix array.



Note: This idea (NIP) shares motivation with the k-mer “skipping” of pseudoalignment[1], though there are differences in the results themselves and in how the information is obtained.

1:Bray,N.L. et al. (2016) Near-optimal probabilistic RNA-seq quantification. Nature Biotech., 34(5), 525–527

RapMap: A Rapid, Sensitive and Accurate Tool for Mapping RNA-seq Reads to Transcriptomes



GitHub repository: https://github.com/COMBINE-lab/RapMap

Paper: http://bioinformatics.oxfordjournals.org/content/32/12/i192.full.pdf

(appeared at ISMB 16)

co-authors (students): Avi Srivastava, Hirak Sarkar, Nitish Gupta

RapMap Index

Generalized suffix array on transcriptome ($ character separating transcripts)

Hash from k-mers to SA intervals (for speed) (can be **dense** or **minimum perfect hash**)

Very fast bit-vector rank — rank9*— allow constant time access to transcript start positions in generalized suffix array

Benefits of this indexing structure

The suffix array allows us to encode / find the NIPs ***dynamically*** (and guided by the length of matching context)

Allows us to efficiently deal with *intervals* of exact matches (efficient).

Length of context changes *dynamically* with quality of data (errors).

Moving from mapping to full alignment becomes very efficient (*ongoing work*).

*Sebastiano Vigna. Broadword implementation of rank/select queries. In Proc. InternationalConference on Experimental Algorithms, WEA'08, pages 154–168, 2008

RapMap Index

Technical details in bonus slides if we have time (and if you're curious)

*Sebastiano Vigna. Broadword implementation of rank/select queries. In Proc. InternationalConference on Experimental Algorithms, WEA'08, pages 154–168, 2008

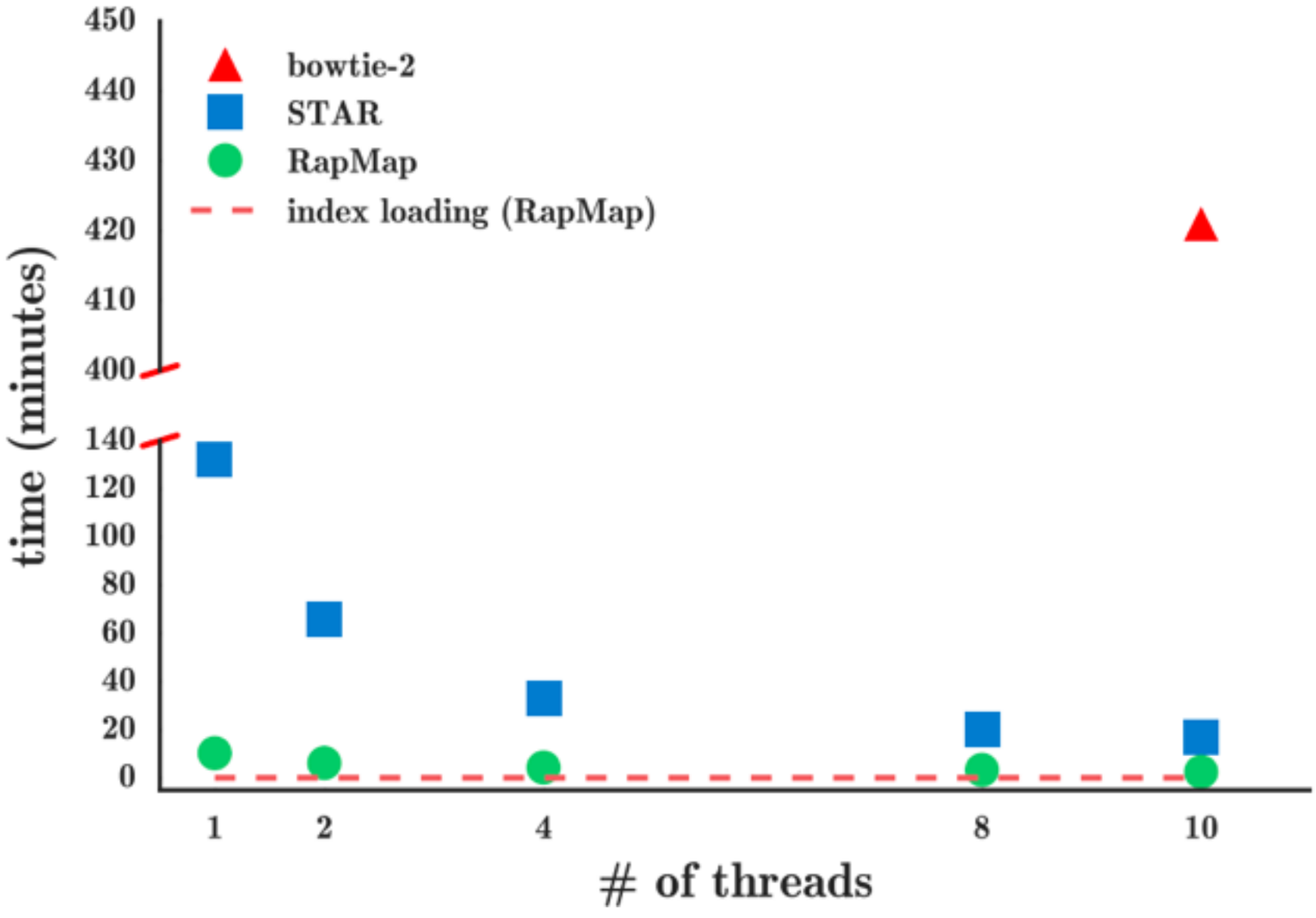
Simulated Data

- Simulated Data using flux-simulator (configs are in the paper)

- Analysis on 48Million (76bp) PE reads mapped to human transcriptome;

- Comparison of 4 tools Bowtie2, STAR and RapMap, kallisto*.

Quasi-mapping is Fast



Can map **75 million paired-end reads** (76 bp) to the human transcriptome in matter of **minutes**; even with few threads.

Note: High degree of multi-mapping and inability to report top "**stratum**" means Bowtie2 is often reporting more than the "best" mapping (though it's commonly used in this context).

Quasi-mapping is Accurate

	Bowtie 2	Kallisto	RapMap	STAR
reads aligned	47579567	44804857	47613536	44711604
recall	97.41	91.60	97.49	91.35
precision	98.31	97.72	98.48	97.02
F1-score	97.86	94.56	97.98	94.10
FDR	1.69	2.28	1.52	2.98
hits per read	5.98	5.30	4.30	3.80

Bowtie 2: BWT-based aligner

Kallisto: dBG-based pseudoaligner

RapMap: SA-based quasi-mapper

STAR: SA-based aligner

TP = True transcript of origin was in the set returned by the method

FP = Mappings were returned for the read, none of which were to the true transcript

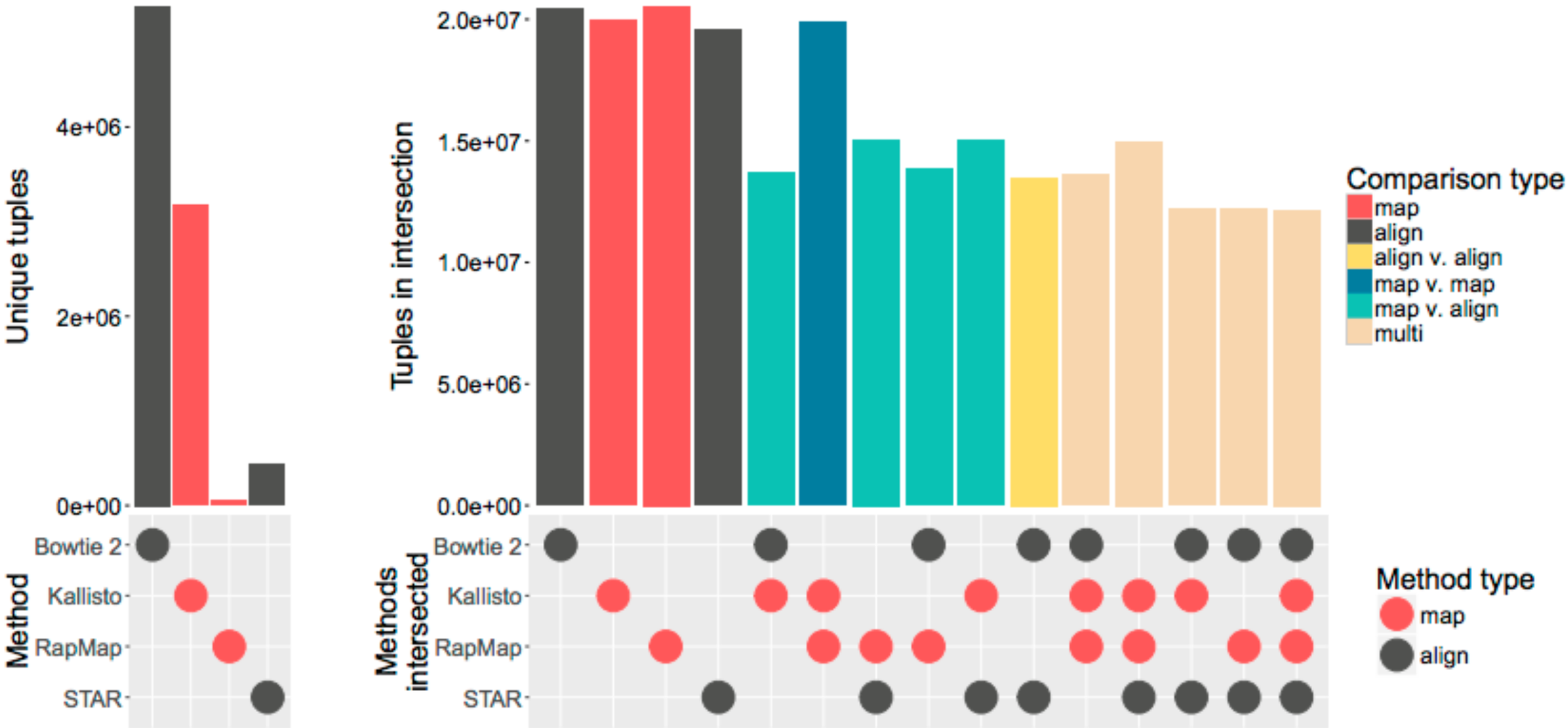
FN = Read is un-mapped, but derives from the transcriptome

Hits per read = Avg. # of mappings returned for the reads
How many *extra* mappings did we report?

Experimental Data

- Analysis on 25Million (76bp) PE reads mapped to human transcriptome;
- Comparison of 4 tools Bowtie2, STAR, RapMap and kallisto.
- Every method reports a tuple $(r, (t_i, t_j))$.
- Concordance: if tuple is exactly same.

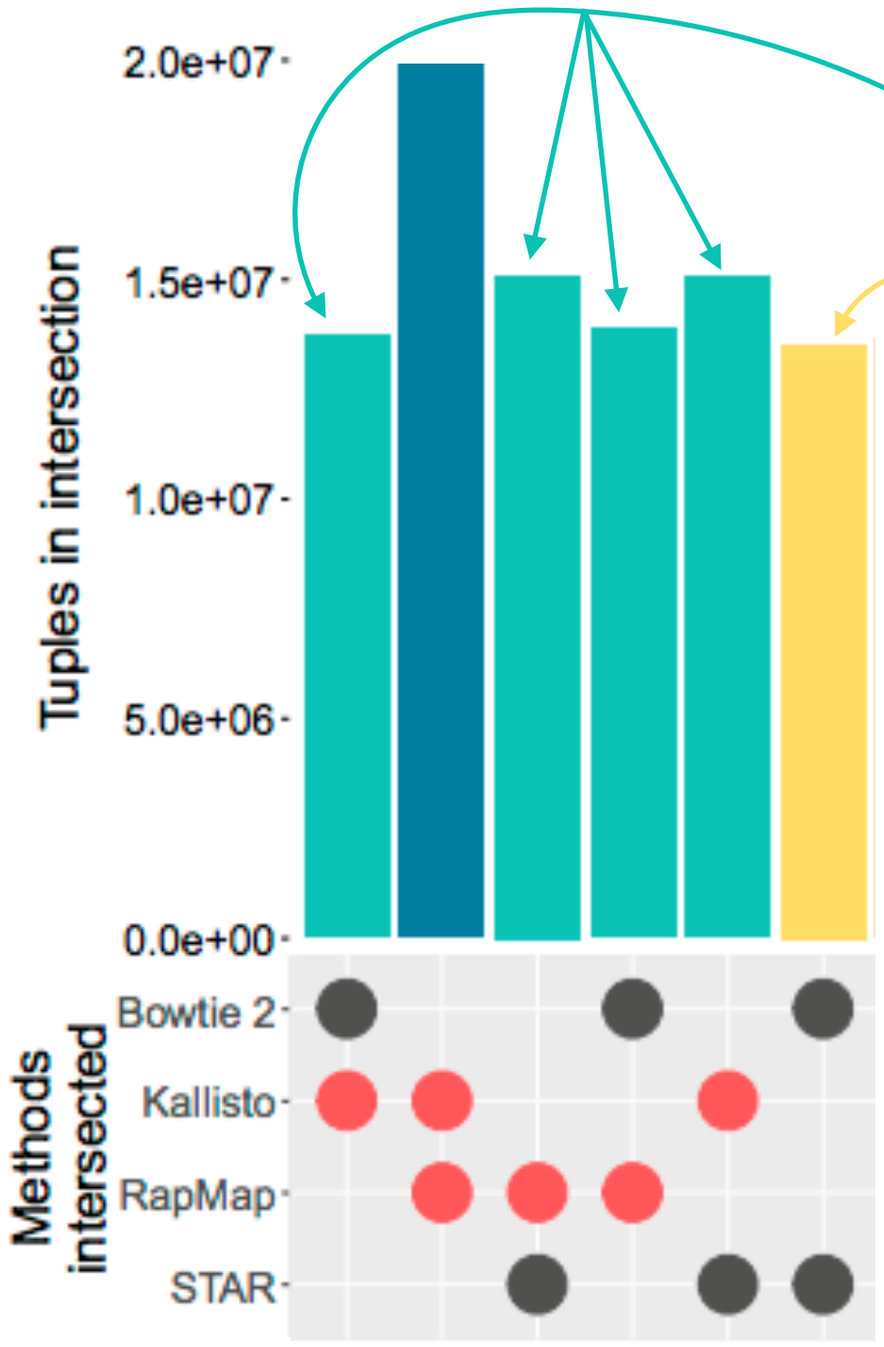
Quasi-mapping and Alignment Agree Well



A *tuple* consists of a read id and set of transcripts e.g. (r_i, {t_1, t_2, t_6})

Two methods *agree* on the mappings of a read if they return the same tuple; otherwise they disagree

Quasi-mapping and Alignment Agree Well



Mappers agree with **Aligners**, at least as often as **aligners** agree with **eachother**.

A *tuple* consists of a read id and set of transcripts e.g. (r_i, {t_1, t_2, t_6})

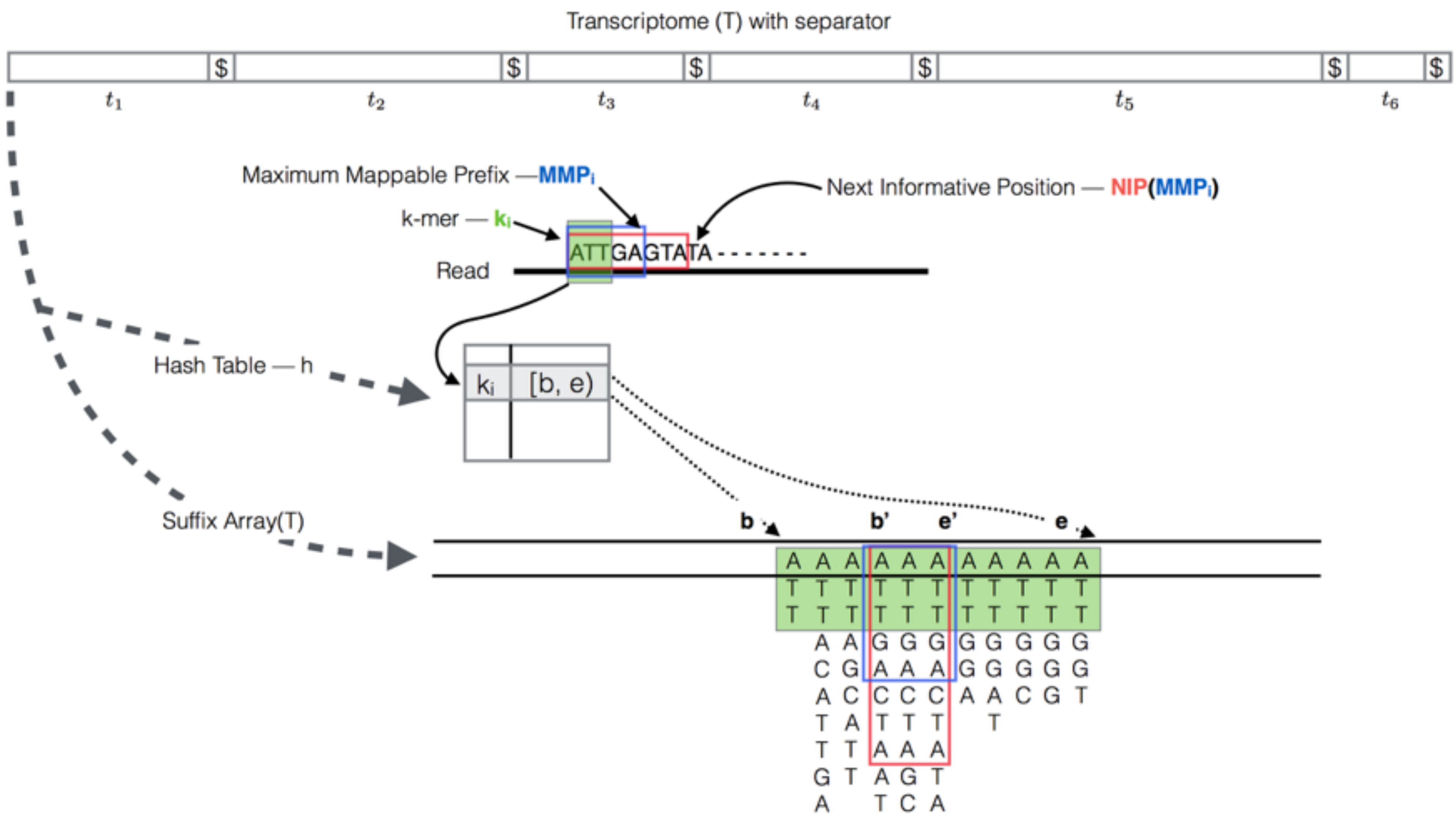
Two methods *agree* on the mappings of a read if they return the same tuple; otherwise they disagree

Bonus slides

An algorithm for quasi-mapping

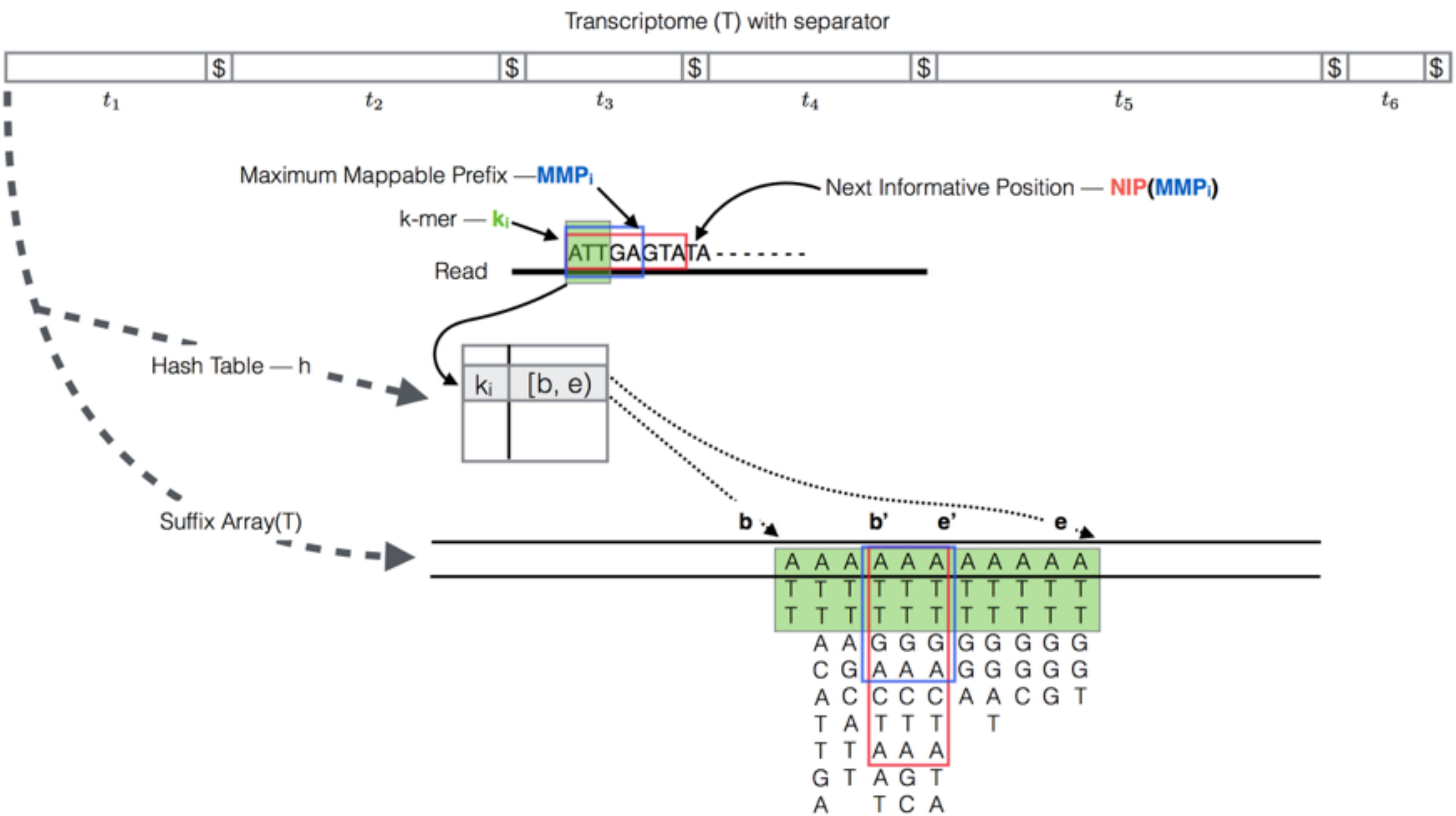
Move from left to right along read, until we find a k-mer with non-empty SA interval.

Compute Maximum Mappable Prefix (**MMP**) starting with this k-mer — logarithmic in k-mers SA interval



An algorithm for quasi-mapping

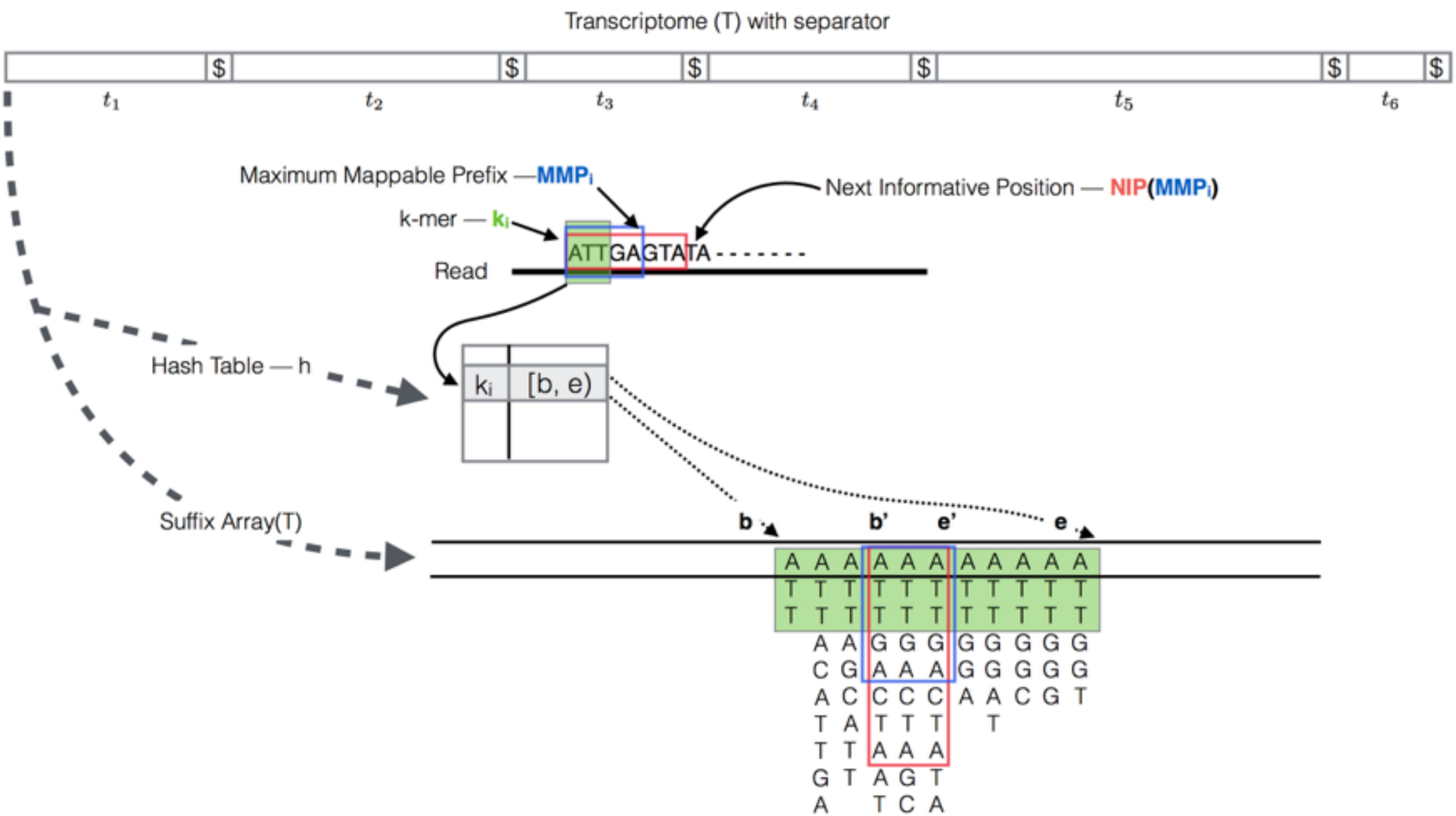
Compute **NIP** of this **MMP** — (fast) linear in read length



An algorithm for quasi-mapping

Compute **NIP** of this **MMP** — (fast) linear in read length

intuitively: **NIP** jumps you to the next exon boundary overlapping the read (need not be an actual exon boundary)



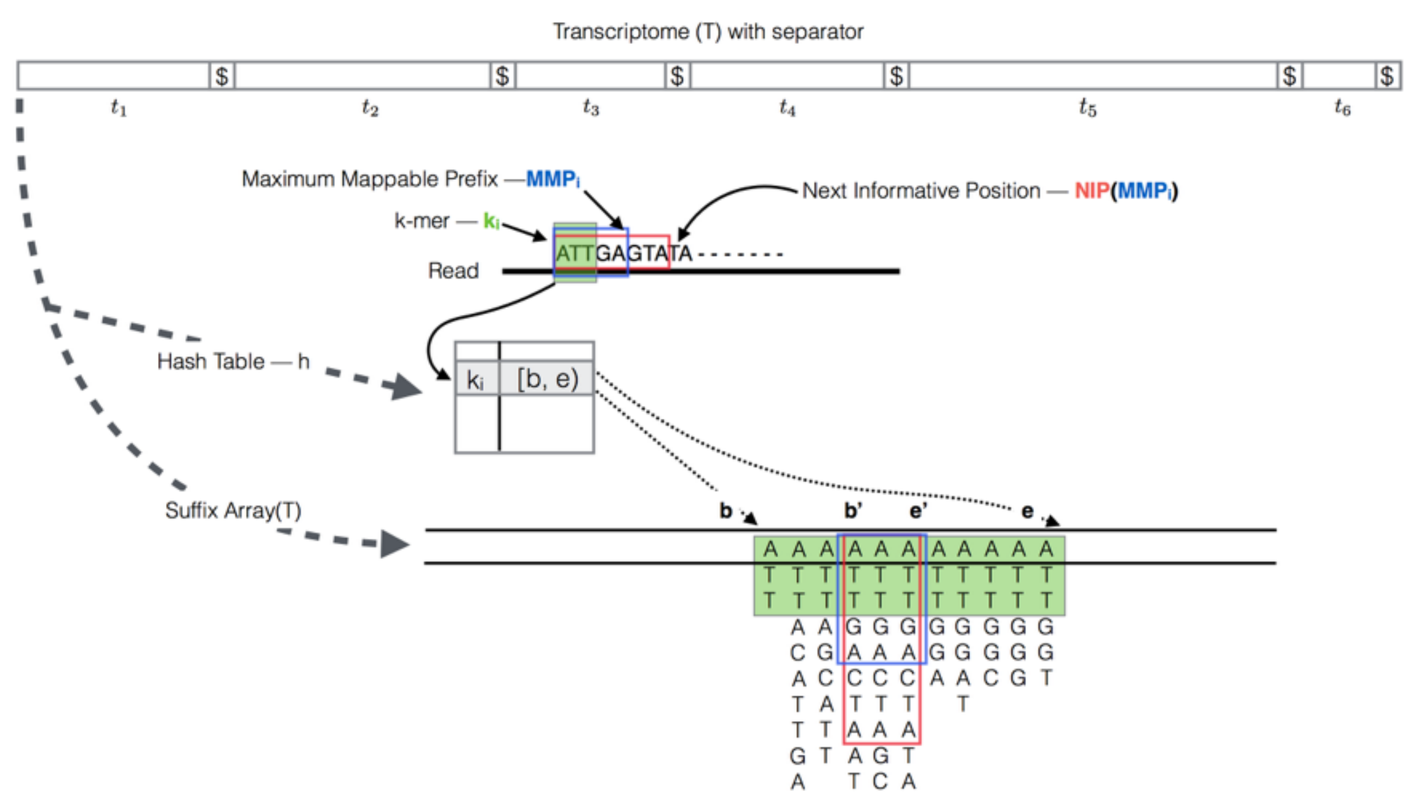
An algorithm for quasi-mapping

Produces a set of disjoint *hits* over each query (read).

A *hit* is a tuple — (query offset, orientation, length, SA-interval)

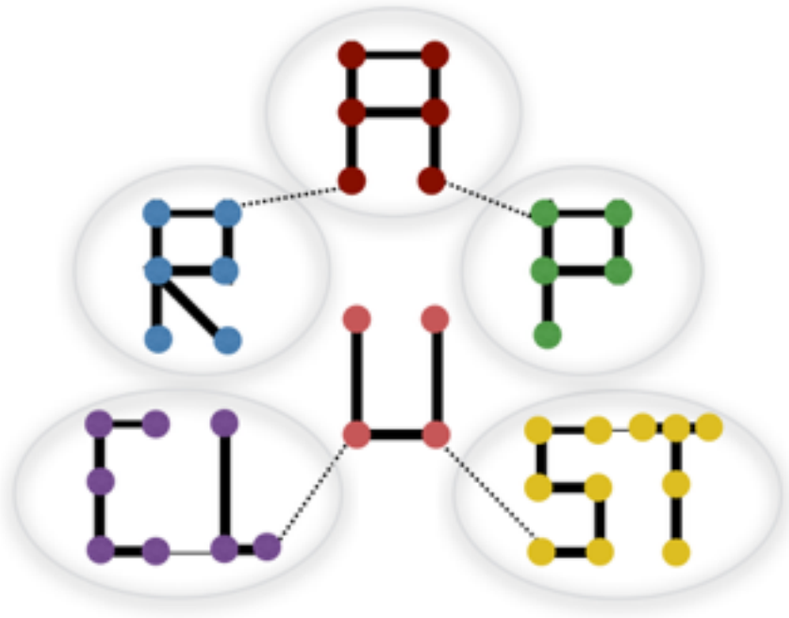
Mappings are determined by a *consensus* mechanism over hits:

- *default:* a read maps to a transcript if that transcript appears in **every hit for that read**.
- other (stricter or looser) mechanisms are trivial to enforce (e.g. co-linearity of hits wrt read & reference).



De novo transcriptome clustering

RapClust: Fast, Lightweight Clustering of de novo Transcriptomes using Fragment Equivalence Classes



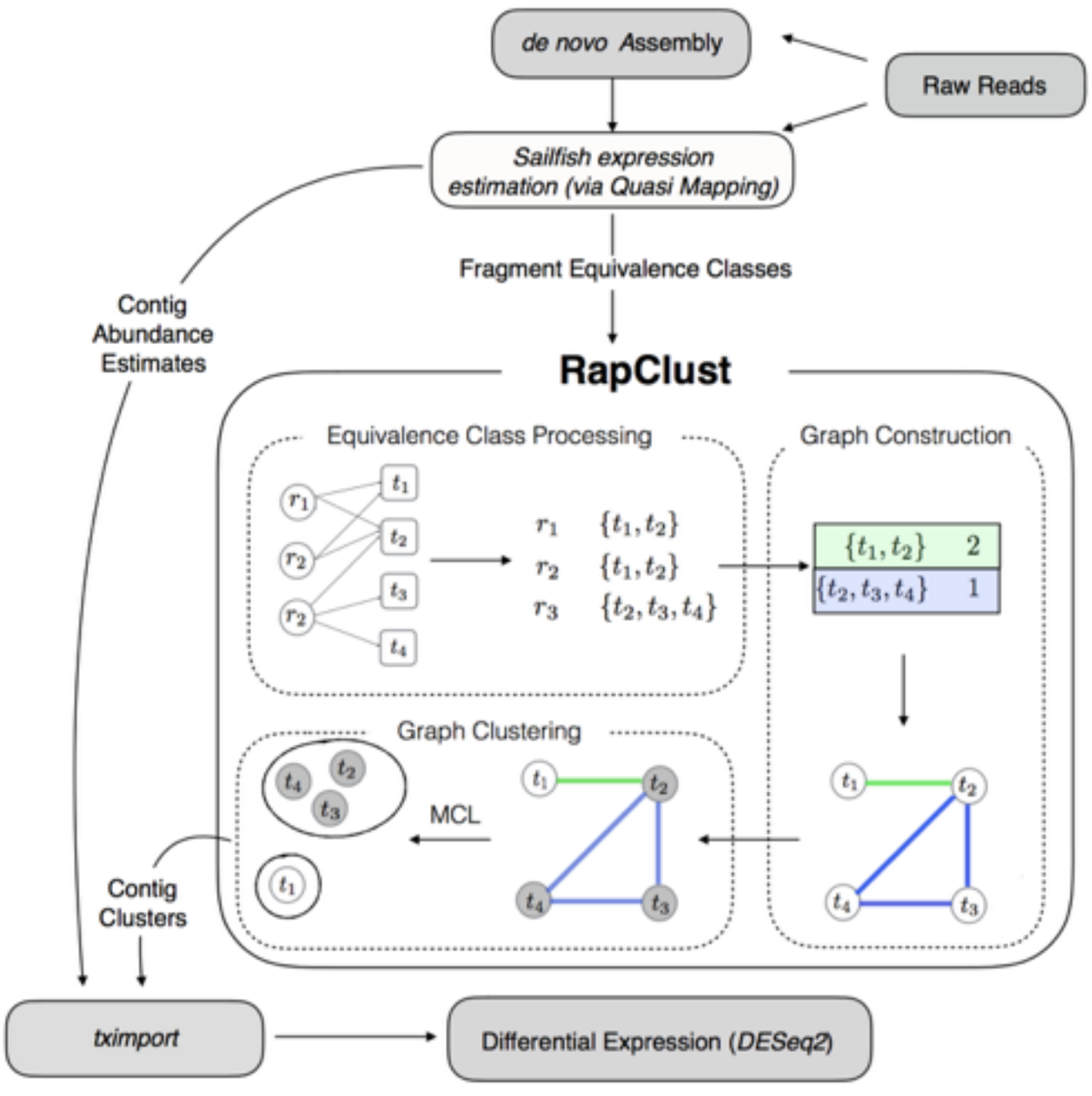
GitHub repository: https://github.com/COMBINE-lab/rapclust

Paper: https://arxiv.org/abs/1604.03250

RapClust: clustering contigs in de novo assemblies

Uses the fragment equivalence classes discussed above to cluster contigs in *de novo* assemblies.

This leads to improved downstream analysis (e.g. DE calls)



RapClust is fast

Time *including* quantification (4 threads)

	Yeast		Human		Chicken	
	`RapClust`	`Corset`	`RapClust`	`Corset`	`RapClust`	`Corset`
Time(min)	5.12	37.25	22.67	211.67	64.18	453
Space(Gb)	0.005	5.7	0.092	22	0.49	145
% of reads	88.17	62.32	93.04	77.94	88.80	60.99

Time *excluding* quantification

	Yeast			Human			Chicken		
	RC	CD	CT	RC	CD	CT	RC	CD	CT
Time(min)	0.04	0.2	2.8	0.82	4.02	16.25	5.29	36.5	87

RapClust is Fast & Lightweight

Time & Space comparison of RapClust with Corset, for *all* phases (raw reads through quantified clusters — using 4 threads).

	Yeast		Human		Chicken	
	`RapClust`	`Corset`	`RapClust`	`Corset`	`RapClust`	`Corset`
Time(min)	5.12	37.25	22.67	211.67	64.18	453
Space(Gb)	0.005	5.7	0.092	22	0.49	145
% of reads	88.17	62.32	93.04	77.94	88.80	60.99

Not having to output / rely on BAM files means the space footprint of RapClust is *orders of magnitude* smaller than that of Corset

Time comparison of RapClust (RC), Corset (CT), and CD-HIT EST (CD) for *just clustering* (using 1 thread).

	Yeast			Human			Chicken		
	RC	CD	CT	RC	CD	CT	RC	CD	CT
Time(min)	0.04	0.2	2.8	0.82	4.02	16.25	5.29	36.5	87

RapClust is accurate

Variation of Information[#] distance between the *true* clustering and the clustering computed by each method (lower is better).

#: Meila, M. (2007). "Comparing clusterings—an information based distance". Journal of Multivariate Analysis 98 (5): 873–895.

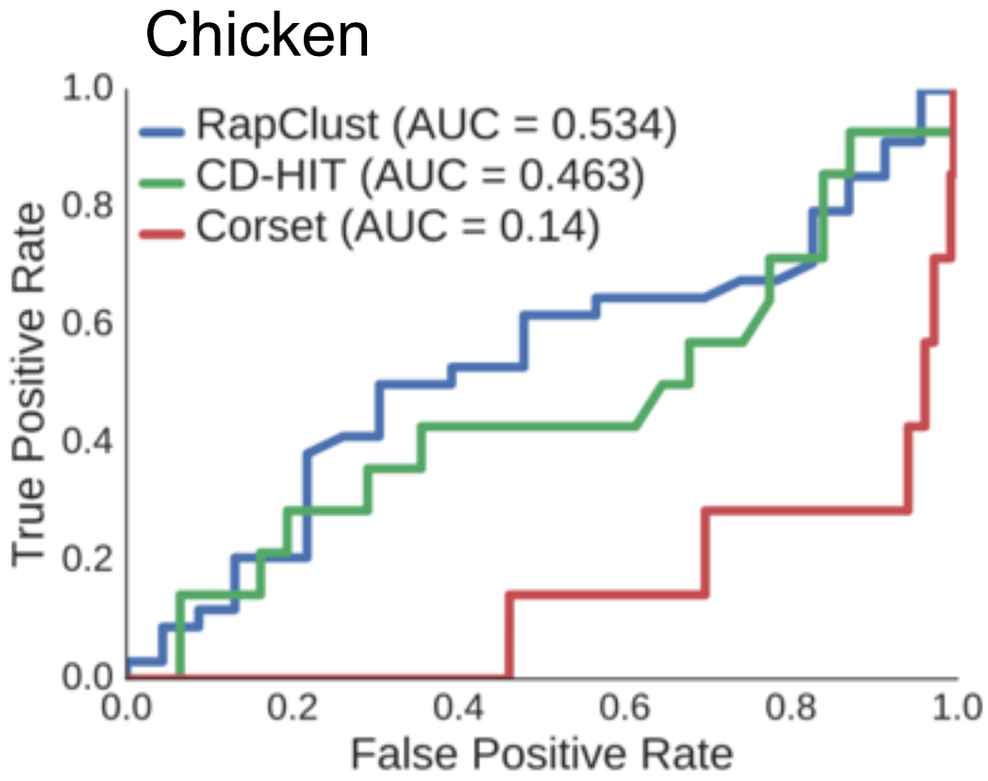
VI Distance	RapClust	CORSET	CD-HIT EST
Chicken	0.127	0.191	2.01
Human	0.712	0.735	1.24
Yeast	0.176	0.178	0.216

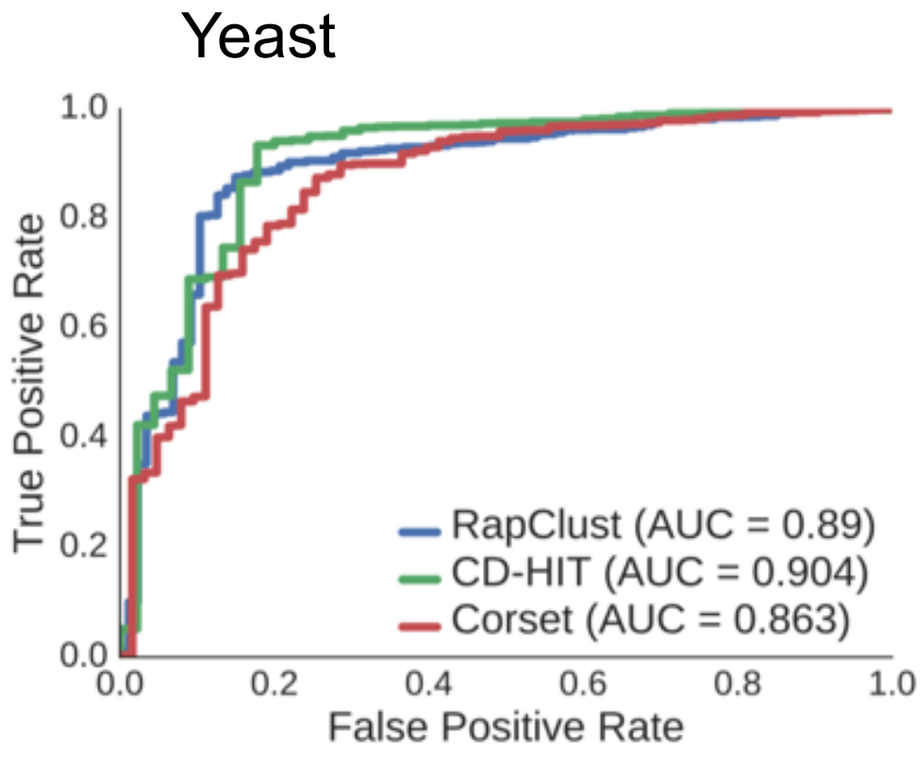
F1-Score of correct classification (i.e. co-clustering) of contigs from the same gene (higher is better).

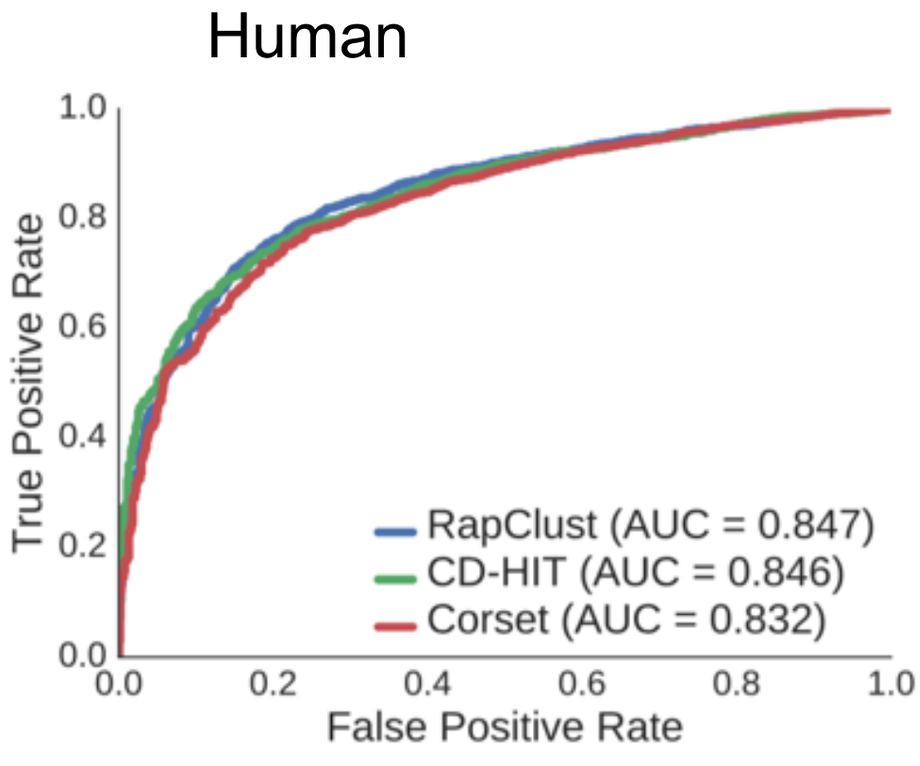
F1-Score	RapClust	CORSET	CD-HIT EST*
Chicken	97.17	95.02	13.27
Human	72.23	70.58	23.97
Yeast	46.24	45.40	21.48

*Note: RapClust & CORSET only predict clusters on an expressed subset of the data; CD-HIT EST is not directly comparable.

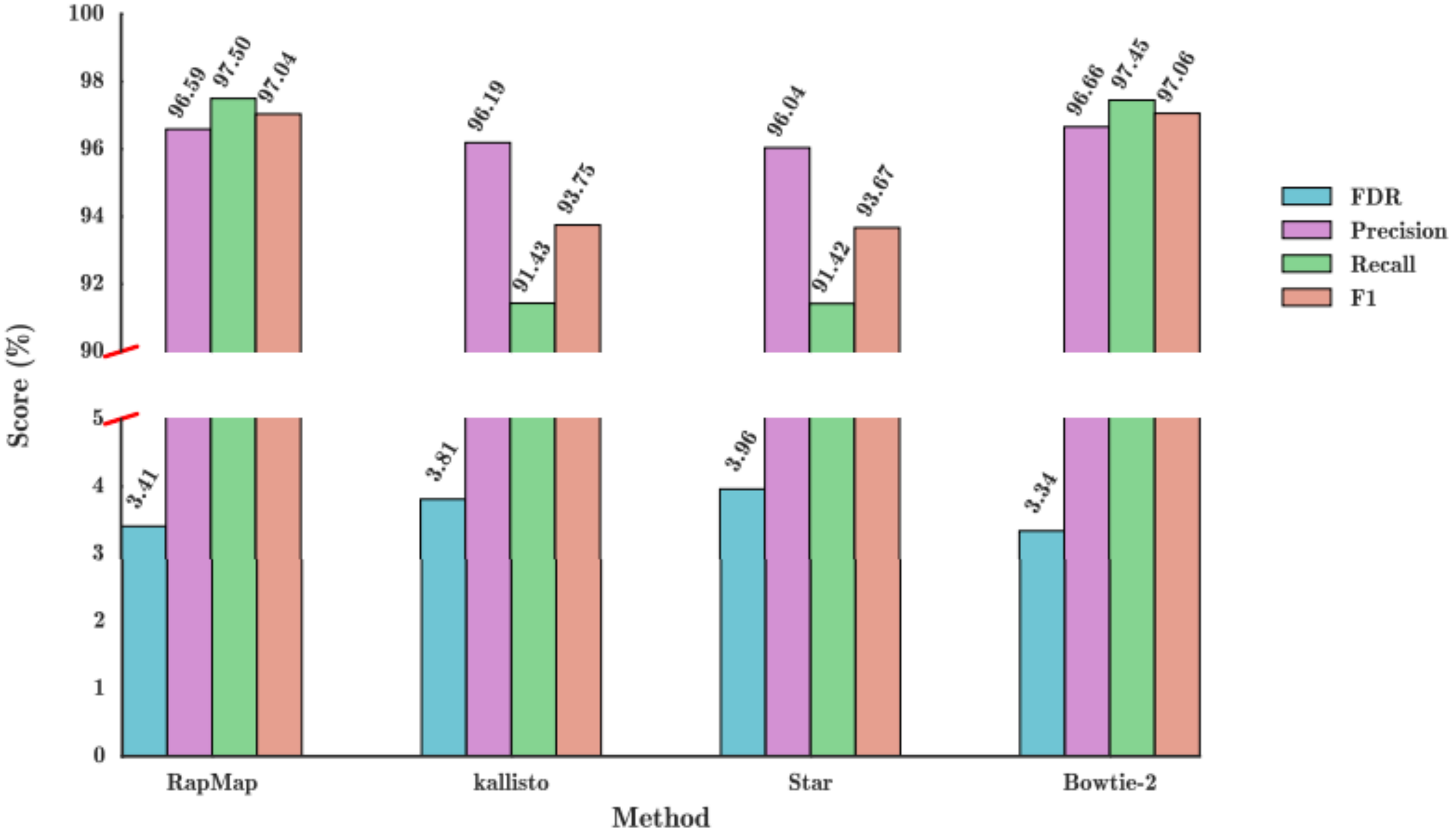
RapClust improves differential expression testing







Trends continue, even with “noisy” transcriptomes



Include “noise” reads from unspliced / nascent transcripts.