Mapping Reads

Dr. Hubert Rehrauer

Purpose of Read Mapping

- Identify the origin of a read
- RNA-seq: Infer the gene that was expressed and that generated the read
- Is it always possible to find the read origin?

Sequence Alignment

- Optimization problem: Find the alignment with the highest score
- Global alignment (end-to-end): Needleman-Wunsch
- Local alignment: Smith-Waterman

Score matrix:

$$\begin{array}{c|cccc} & a & c & g & t \\ \hline a & +2 & -2 & -1 & -2 \\ c & -2 & +2 & -2 & -1 \\ g & -1 & -2 & +2 & -2 \\ t & -2 & -1 & -2 & +2 \end{array}$$

Gap score: -3

```
c  c  t  a  g  t  t  c  a  t
      |  |  x  |  x  |  |
t  g  t  a  a  t  -  c  a  c
     +2 +2 -1 +2 -3 +2 +2
```

Alignment score = 6

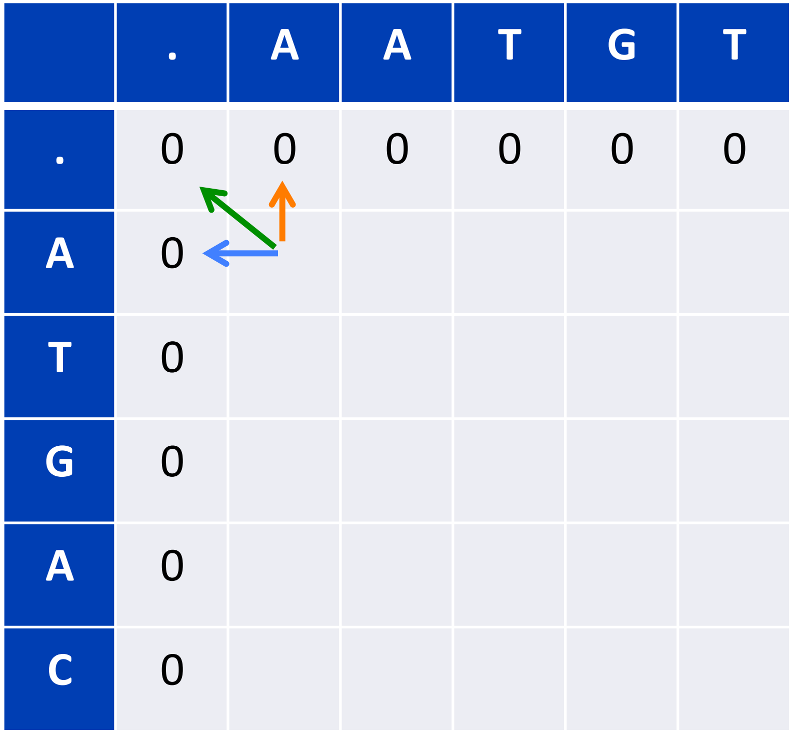
Local alignments (Smith-Waterman)

Generally useful for local alignments, determining regions of similarity between two strings (here, DNA sequence)

Dynamic programming based algorithm: gives optimal alignment, with respect to scoring system

Need to set scores for **match**, penalties for **mismatch** and **gap** (typically, mismatch penalties are set according to evolutionary knowledge)

Match: +1; Mismatch: -1; Gap: -2



	.	A	A	T	G	T
.	0	0	0	0	0	0
A	0					
T	0					
G	0					
A	0					
C	0					

$$H(i,j) = \max \begin{Bmatrix} 0 & \\ H(i-1,j-1) + \ w(a_i,b_j) & \boxed{\text{Match/Mismatch}} \\ H(i-1,j) + \ w(a_i,-) & \boxed{\text{Deletion}} \\ H(i,j-1) + \ w(-,b_j) & \boxed{\text{Insertion}} \end{Bmatrix}, \ 1 \le i \le m, 1 \le j \le n$$

http://en.wikipedia.org/wiki/Smith%E2%80%93Waterman_algorithm

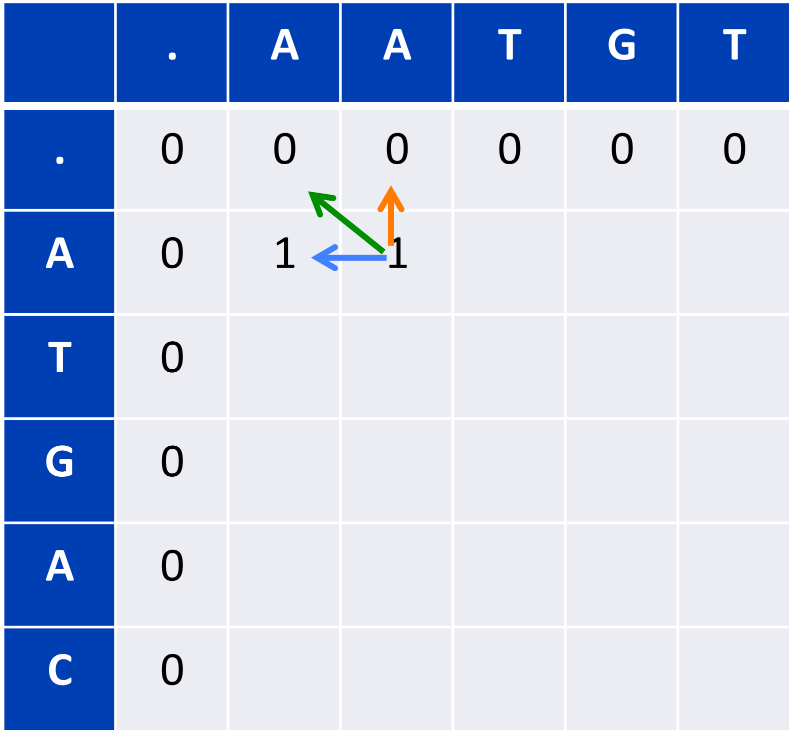
Local alignments (Smith-Waterman)

Generally useful for local alignments, determining regions of similarity between two strings (here, DNA sequence)

Dynamic programming based algorithm: gives optimal alignment, with respect to scoring system

Need to set scores for **match**, penalties for **mismatch** and **gap**

Match: +1; Mismatch: -1; Gap: -2



	.	A	A	T	G	T
.	0	0	0	0	0	0
A	0	1	1			
T	0					
G	0					
A	0					
C	0					

$$H(i,j) = \max \begin{Bmatrix} 0 & \\ H(i-1,j-1) + \; w(a_i,b_j) & \boxed{\text{Match/Mismatch}} \\ H(i-1,j) + \; w(a_i,-) & \boxed{\text{Deletion}} \\ H(i,j-1) + \; w(-,b_j) & \boxed{\text{Insertion}} \end{Bmatrix}, \; 1 \le i \le m, 1 \le j \le n$$

http://en.wikipedia.org/wiki/Smith%E2%80%93Waterman_algorithm

Local alignments (Smith-Waterman)

Generally useful for local alignments, determining regions of similarity between two strings (here, DNA sequence)

Dynamic programming based algorithm: gives optimal alignment, with respect to scoring system

Need to set scores for **match**, penalties for **mismatch** and **gap**

Match: +1; Mismatch: -1; Gap: -2

	.	A	A	T	G	T
.	0	0	0	0	0	0
A	0	↖ 1	↖ 1	0	0	0
T	0	0	0	2	0	1
G	0					
A	0					
C	0					

$$H(i,j) = \max \begin{Bmatrix} 0 & \\ H(i-1,j-1) + \ w(a_i,b_j) & \boxed{\text{Match/Mismatch}} \\ H(i-1,j) + \ w(a_i,-) & \boxed{\text{Deletion}} \\ H(i,j-1) + \ w(-,b_j) & \boxed{\text{Insertion}} \end{Bmatrix}, \ 1 \le i \le m, 1 \le j \le n$$

http://en.wikipedia.org/wiki/Smith%E2%80%93Waterman_algorithm

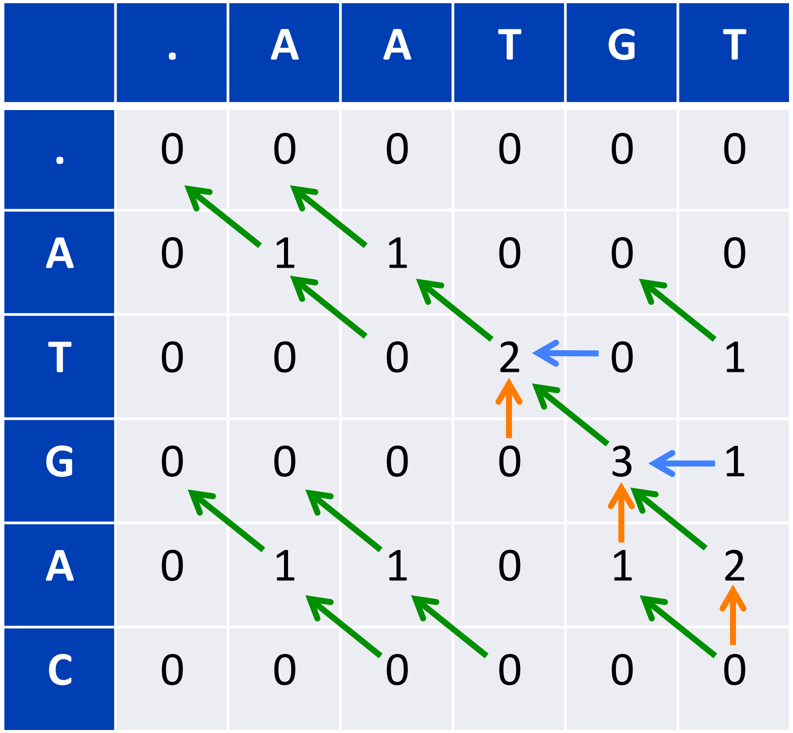
Local alignments (Smith-Waterman)

Generally useful for local alignments, determining regions of similarity between two strings (here, DNA sequence)

Dynamic programming based algorithm: gives optimal alignment, with respect to scoring system

Need to set scores for **match**, penalties for **mismatch** and **gap**

Match: +1; Mismatch: -1; Gap: -2



	.	A	A	T	G	T
.	0	0	0	0	0	0
A	0	1	1	0	0	0
T	0	0	0	2	0	1
G	0	0	0	0	3	1
A	0	1	1	0	1	2
C	0	0	0	0	0	0

$$H(i,j) = \max \begin{Bmatrix} 0 & \\ H(i-1,j-1) + \ w(a_i,b_j) & \boxed{\text{Match/Mismatch}} \\ H(i-1,j) + \ w(a_i,-) & \boxed{\text{Deletion}} \\ H(i,j-1) + \ w(-,b_j) & \boxed{\text{Insertion}} \end{Bmatrix}, \ 1 \le i \le m, 1 \le j \le n$$

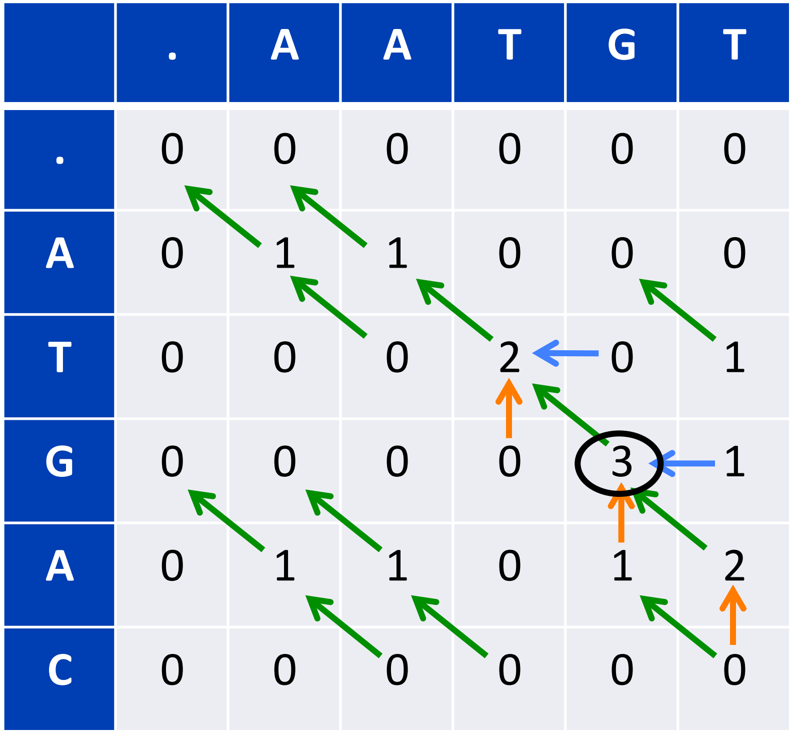
http://en.wikipedia.org/wiki/Smith%E2%80%93Waterman_algorithm

Local alignments (Smith-Waterman)

Traceback:

Start with maximum score (here, 3)

Follow path that gives multiple score



	.	A	A	T	G	T
.	0	0	0	0	0	0
A	0	1	1	0	0	0
T	0	0	0	2	0	1
G	0	0	0	0	3	1
A	0	1	1	0	1	2
C	0	0	0	0	0	0

$$H(i,j) = \max \begin{Bmatrix} 0 & \\ H(i-1,j-1) + \ w(a_i,b_j) & \boxed{\text{Match/Mismatch}} \\ H(i-1,j) + \ w(a_i,-) & \boxed{\text{Deletion}} \\ H(i,j-1) + \ w(-,b_j) & \boxed{\text{Insertion}} \end{Bmatrix}, \ 1 \le i \le m, 1 \le j \le n$$

http://en.wikipedia.org/wiki/Smith%E2%80%93Waterman_algorithm

University of Zurich^UZH

Institute of Molecular Life Sciences

Local alignments (Smith-Waterman)

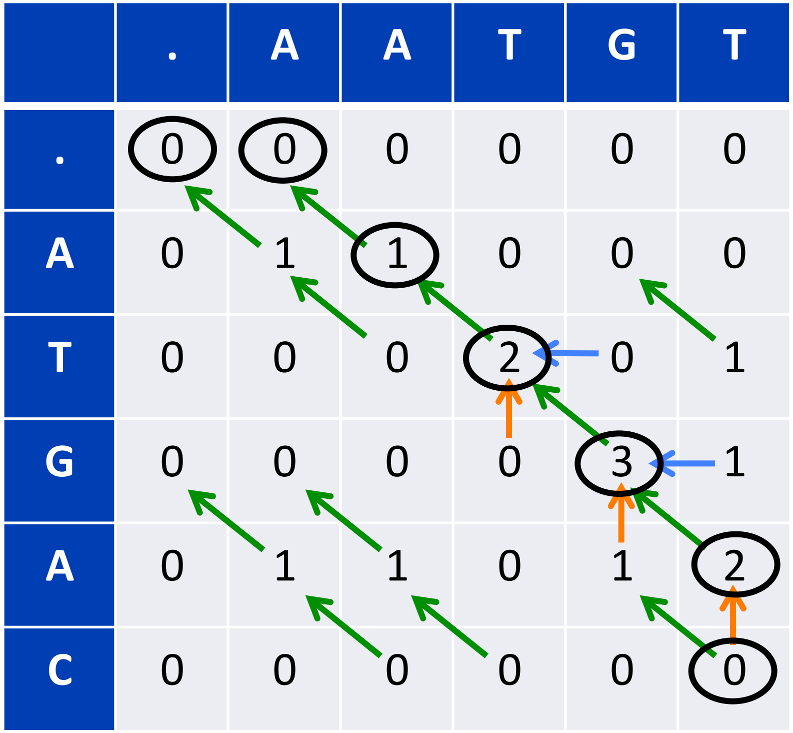
Traceback:

Start with maximum score (here, 3)

Follow path that gives multiple score

This gives:

```
AATGT-
-ATGAC
```



	.	A	A	T	G	T
.	0	0	0	0	0	0
A	0	1	1	0	0	0
T	0	0	0	2	0	1
G	0	0	0	0	3	1
A	0	1	1	0	1	2
C	0	0	0	0	0	0

$$H(i,j) = \max \begin{Bmatrix} 0 & \\ H(i-1,j-1) + \ w(a_i, b_j) & \boxed{\text{Match/Mismatch}} \\ H(i-1,j) + \ w(a_i, -) & \boxed{\text{Deletion}} \\ H(i,j-1) + \ w(-, b_j) & \boxed{\text{Insertion}} \end{Bmatrix}, \ 1 \leq i \leq m, 1 \leq j \leq n$$

http://en.wikipedia.org/wiki/Smith%E2%80%93Waterman_algorithm

Local alignments (Smith-Waterman)

Other considerations:

Natural extension is to have a different *gap opening* and a *gap extension* penalty (former generally being larger)

GO penalty=-2; GE penalty=-1; Mismatch=-1; Match=1

With above penalties, which is the best scoring alignment?

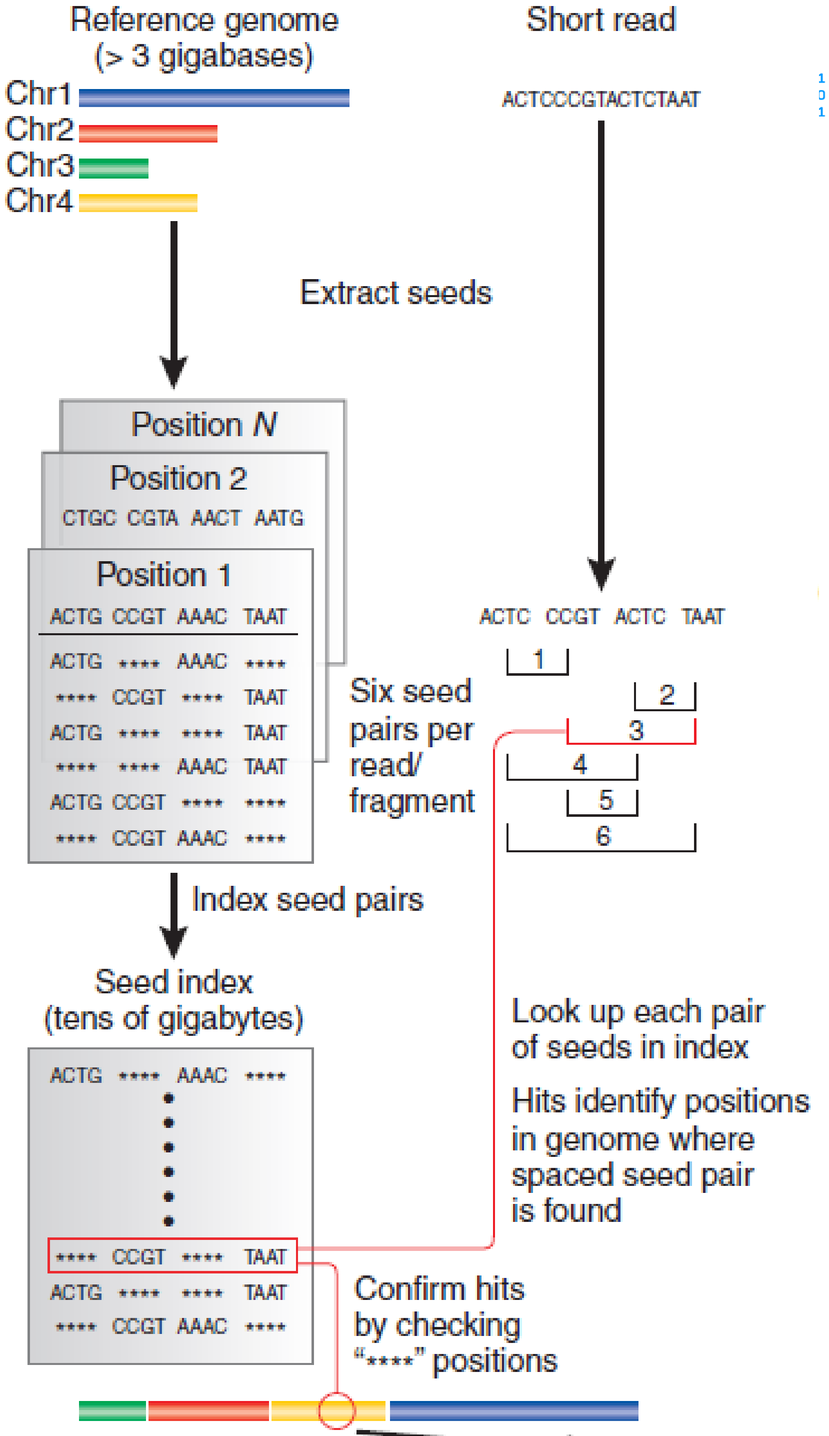
```
AT-C-GT      ATC--GT      AT-C--GT
ATTTTGT      ATTTTGT      ATT-TTGT
```

Alignment Tools

- Dynamic programming is slow
- Speed-up with heuristics
  - e.g. exactly align short subsequences and extend these alignments
    - e.g. BLAST / BLAT
  - no longer guaranteed to find the best alignments
  - exact matches are found by index lookup

Index Genome: Spaced Seeds

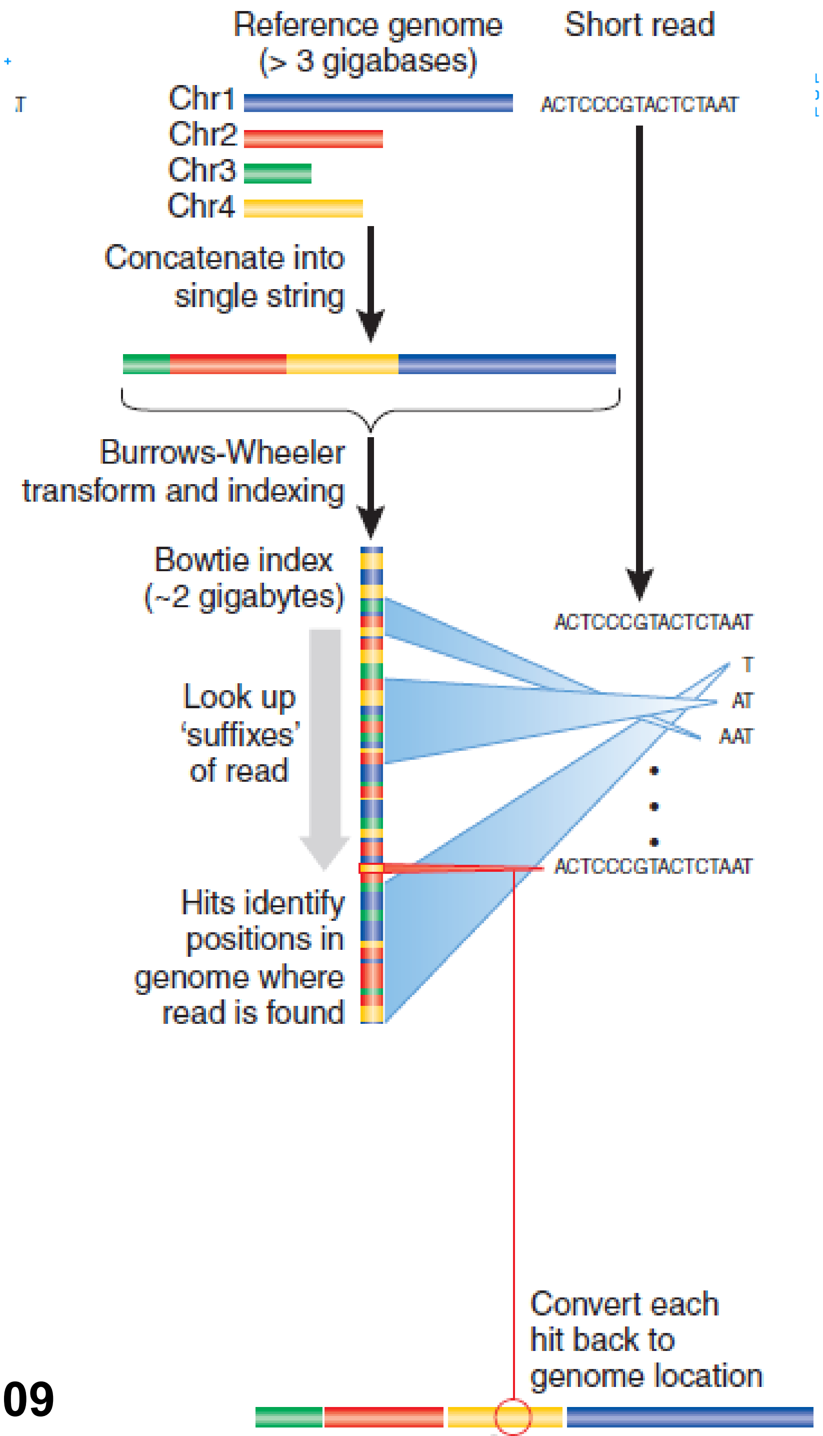
- Tags and tag-sized pieces of reference are cut into small “seeds.”
- Pairs of spaced seeds are stored in an index.
- Look up spaced seeds for each tag.
- For each “hit,” confirm the remaining positions.



Trapnell etal., Nature Biotechnology, 2009

Index Genome: Burrows-Wheeler Transform

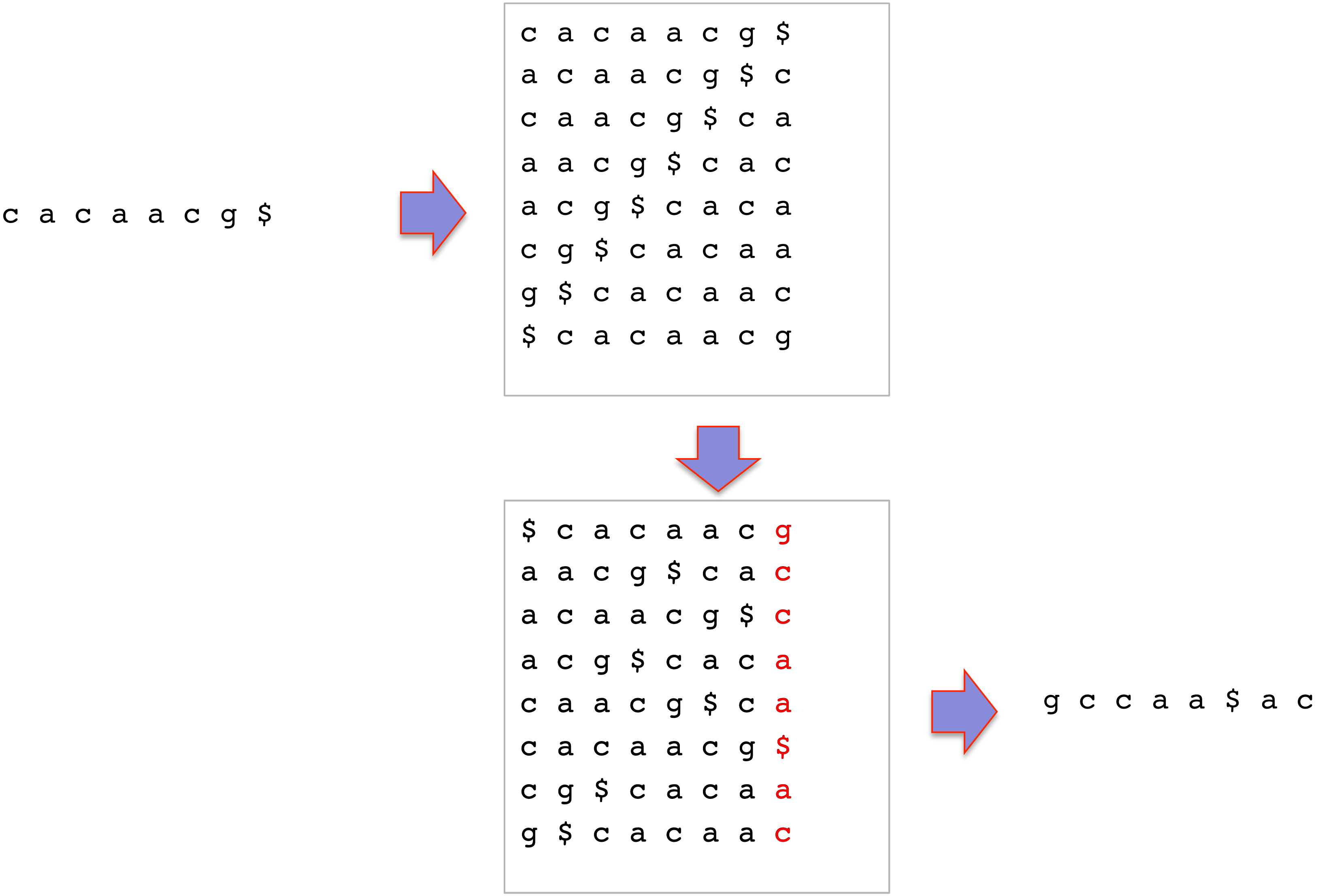
- Store entire reference genome.
- Align tag base by base from the end.
- When tag is traversed, all active locations are reported.
- If no match is found, then back up and try a substitution.



Reference genome
(> 3 gigabases)
Short read
Chr1
Chr2
Chr3
Chr4
ACTCCCGTACTCTAAT
Concatenate into single string
Burrows-Wheeler transform and indexing
Bowtie index (~2 gigabytes)
ACTCCCGTACTCTAAT
T
AT
AAT
Look up 'suffixes' of read
ACTCCCGTACTCTAAT
Hits identify positions in genome where read is found
Convert each hit back to genome location

Trapnell etal., Nature Biotechnology, 2009

The Burrows-Wheeler transform (1994; 1983)



```
c a c a a c g $
```

```
c a c a a c g $
a c a a c g $ c
c a a c g $ c a
a a c g $ c a c
a c g $ c a c a
c g $ c a c a a
g $ c a c a a c
$ c a c a a c g
```

```
$ c a c a a c g
a a c g $ c a c
a c a a c g $ c
a c g $ c a c a
c a a c g $ c a
c a c a a c g $
c g $ c a c a a
g $ c a c a a c
```

```
g c c a a $ a c
```

The “Last-First mapping” property

The relative ordering of a particular character (say c) in column 1 is the same as that in the last column

$c_1 \; a \; c_2 \; a \; a \; c_3 \; g \; \$$

$\$ \; c_1 \; a \; c_2 \; a \; a \; c_3 \; g$
$a \; a \; c_3 \; g \; \$ \; c_1 \; a \; c_2$
$a \; c_2 \; a \; a \; c_3 \; g \; \$ \; c_1$
$a \; c_3 \; g \; \$ \; c_1 \; a \; c_2 \; a$
$c_2 \; a \; a \; c_3 \; g \; \$ \; c_1 \; a$
$c_1 \; a \; c_2 \; a \; a \; c_3 \; g \; \$$
$c_3 \; g \; \$ \; c_1 \; a \; c_2 \; a \; a$
$g \; \$ \; c_1 \; a \; c_2 \; a \; a \; c_3$

The “Last-First mapping” property

The relative ordering of a particular character (say c) in column 1 is the same as that in the last column

$c_1 \; a \; c_2 \; a \; a \; c_3 \; g \; \$$

$$\begin{array}{cccccccc}
\$ & c_1 & a & c_2 & a & a & c_3 & g \\
a & a & c_3 & g & \$ & c_1 & a & c_2 \\
a & c_2 & a & a & c_3 & g & \$ & c_1 \\
a & c_3 & g & \$ & c_1 & a & c_2 & a \\
c_2 & a & a & c_3 & g & \$ & c_1 & a \\
c_1 & a & c_2 & a & a & c_3 & g & \$ \\
c_3 & g & \$ & c_1 & a & c_2 & a & a \\
g & \$ & c_1 & a & c_2 & a & a & c_3
\end{array}$$

Proof:

Suppose c X and c Y are cyclic permutations of the input T.
Suppose c X < c Y (in lexicographical ordering)
Then X c < Y c (in lexicographical ordering)
The LF-mapping property follows.

BWT is reversible

```
$ c a c a a c g
a a c g $ c a c
a c a a c g $ c
a c g $ c a c a
c a a c g $ c a
c a c a a c g $
c g $ c a c a a
g $ c a c a a c
```

g c c a a $ a c → $ a a a c c c g

BWT is reversible

```
$ c a c a a c g
a a c g $ c a c
a c a a c g $ c
a c g $ c a c a
c a a c g $ c a
c a c a a c g $
c g $ c a c a a
g $ c a c a a c
```

g$ ca ca aa ac $c ac cg → $c aa ac ac ca ca cg g$

BWT is reversible

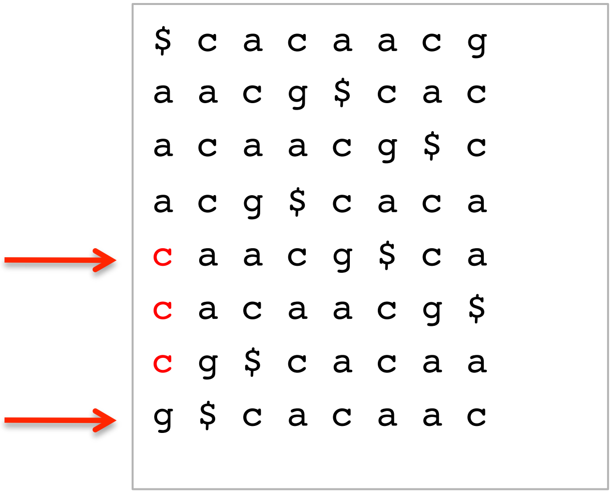
$	c	a	c	a	a	c	g
a	a	c	g	$	c	a	c
a	c	a	a	c	g	$	c
a	c	g	$	c	a	c	a
c	a	a	c	g	$	c	a
c	a	c	a	a	c	g	$
c	g	$	c	a	c	a	a
g	$	c	a	c	a	a	c

Lookup AAC

```
$ c a c a a c g
a a c g $ c a c
a c a a c g $ c
a c g $ c a c a
c a a c g $ c a
c a c a a c g $
c g $ c a c a a
g $ c a c a a c
```

```
Range ← range of last character in 1st column
While characters left (and nonzero range):
        Lookup first and last match to preceding character in final column
        Range ← LF-mapping of first and last match
```

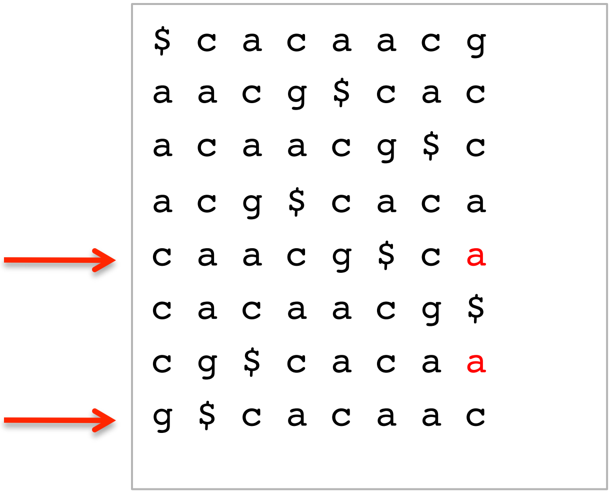
Lookup AAC



$	c	a	c	a	a	c	g
a	a	c	g	$	c	a	c
a	c	a	a	c	g	$	c
a	c	g	$	c	a	c	a
c	a	a	c	g	$	c	a
c	a	c	a	a	c	g	$
c	g	$	c	a	c	a	a
g	$	c	a	c	a	a	c

```
Range ← range of last character in 1st column
While characters left (and nonzero range):
        Lookup first and last match to preceding character in final column
        Range ← LF-mapping of first and last match
```

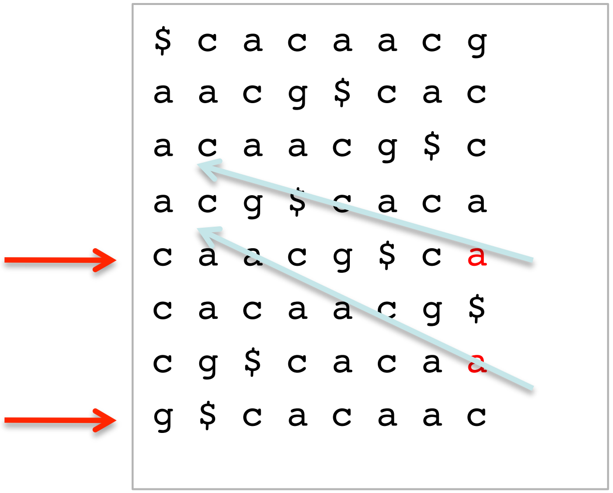
Lookup AAC



```
$ c a c a a c g
a a c g $ c a c
a c a a c g $ c
a c g $ c a c a
c a a c g $ c a
c a c a a c g $
c g $ c a c a a
g $ c a c a a c
```

```
Range ← range of last character in 1st column
While characters left (and nonzero range):
        Lookup first and last match to preceding character in final column
        Range ← LF-mapping of first and last match
```

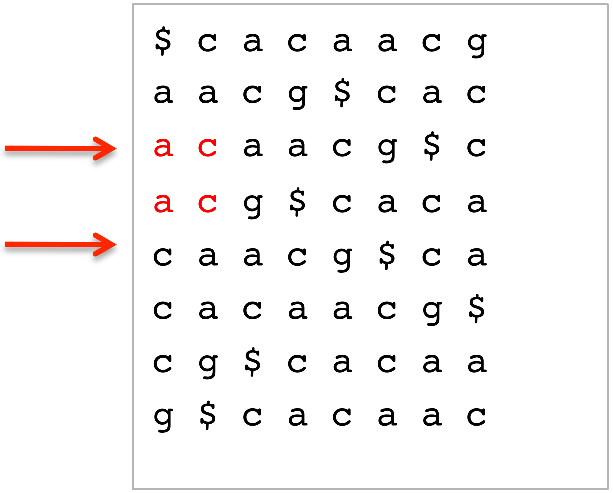
Lookup AAC



```
$ c a c a a c g
a a c g $ c a c
a c a a c g $ c
a c g $ c a c a
c a a c g $ c a
c a c a a c g $
c g $ c a c a a
g $ c a c a a c
```

```
Range ← range of last character in 1st column
While characters left (and nonzero range):
        Lookup first and last match to preceding character in final column
        Range ← LF-mapping of first and last match
```

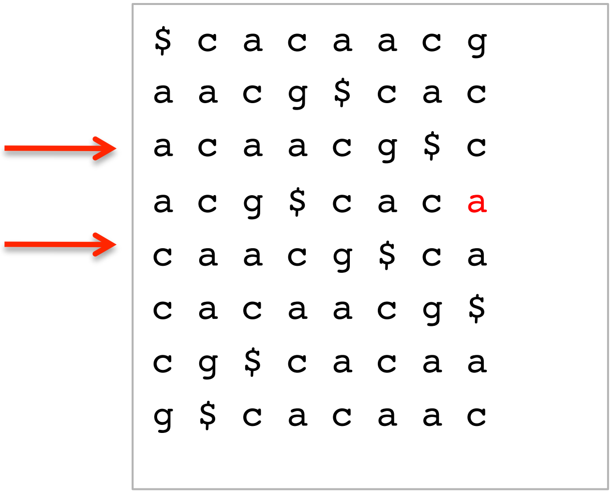
Lookup AAC



```
$ c a c a a c g
a a c g $ c a c
a c a a c g $ c
a c g $ c a c a
c a a c g $ c a
c a c a a c g $
c g $ c a c a a
g $ c a c a a c
```

```
Range ← range of last character in 1st column
While characters left (and nonzero range):
        Lookup first and last match to preceding character in final column
        Range ← LF-mapping of first and last match
```

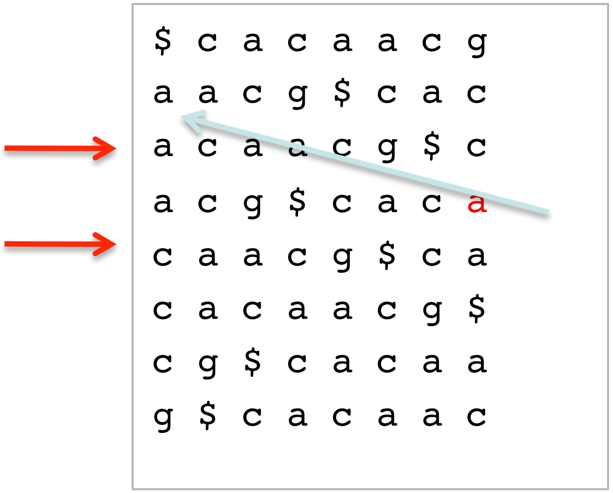
Lookup AAC



```
$ c a c a a c g
a a c g $ c a c
a c a a c g $ c
a c g $ c a c a
c a a c g $ c a
c a c a a c g $
c g $ c a c a a
g $ c a c a a c
```

```
Range ← range of last character in 1st column
While characters left (and nonzero range):
        Lookup first and last match to preceding character in final column
        Range ← LF-mapping of first and last match
```

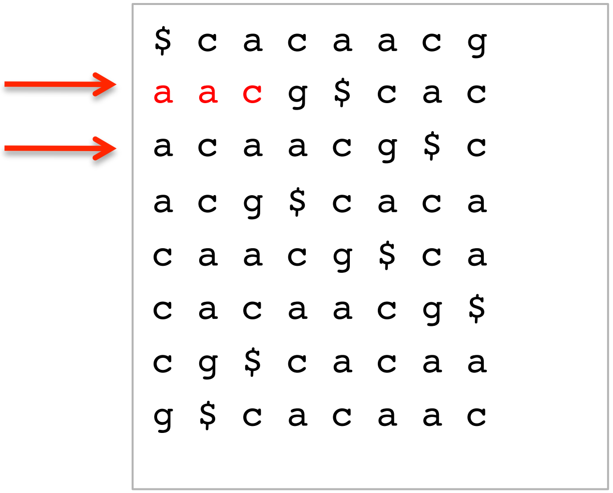
Lookup AAC



```
$ c a c a a c g
a a c g $ c a c
a c a a c g $ c
a c g $ c a c a
c a a c g $ c a
c a c a a c g $
c g $ c a c a a
g $ c a c a a c
```

```
Range ← range of last character in 1st column
While characters left (and nonzero range):
        Lookup first and last match to preceding character in final column
        Range ← LF-mapping of first and last match
```

Lookup AAC



```
$ c a c a a c g
a a c g $ c a c
a c a a c g $ c
a c g $ c a c a
c a a c g $ c a
c a c a a c g $
c g $ c a c a a
g $ c a c a a c
```

```
Range ← range of last character in 1st column
While characters left (and nonzero range):
        Lookup first and last match to preceding character in final column
        Range ← LF-mapping of first and last match
```

Comparison

Spaced seeds

- Requires ~50Gb of memory.
- Runs 30-fold slower.
- More straightforward to program.
- Examples:
  - MAQ
  - Shrimp

- More tolerant to
  - sequence variations
  - sequencing errors

Burrows-Wheeler

- Requires <2Gb of memory.
- Runs 30-fold faster.
- More complicated to program.
- Examples:
  - bowtie
  - BWA
  - tophat (uses bowtie)
  - STAR

Alignment with Mismatches

- Mismatches can occur because of
  - sequencing error (error rate ~1/500)
  - mutation; (human mutation rate ~1/1e4)
  - → if reads are long (> 100nt) reads with mismatches will not be rare; more than ~10% of the reads may have a mismatch
- If there is a sequence mismatch the index lookup fails! How to find nevertheless an alignment?

BWT Alignment with Mismatches

- Strategies:
  - If the BWT lookup fails at position k, try all different bases at position k
    → drawback: computing effort grows exponentially with the number of mismatches
    → implemented e.g. in bowtie
  - Chop reads in segments (seeds) and align those mismatch-free and stitch seed alignments together
    → implemented e.g. in bowtie

Consideration: Sequencing Errors – PHRED quality

- Each called base is given a quality score Q
- Q = 10 * log10(estimated probability that call is wrong)

10	prob = 0.1
20	prob = 0.01
30	prob = 0.001

[Q30 often used as a threshold for useful sequence data]

→down-weight low-quality bases when computing the alignment score

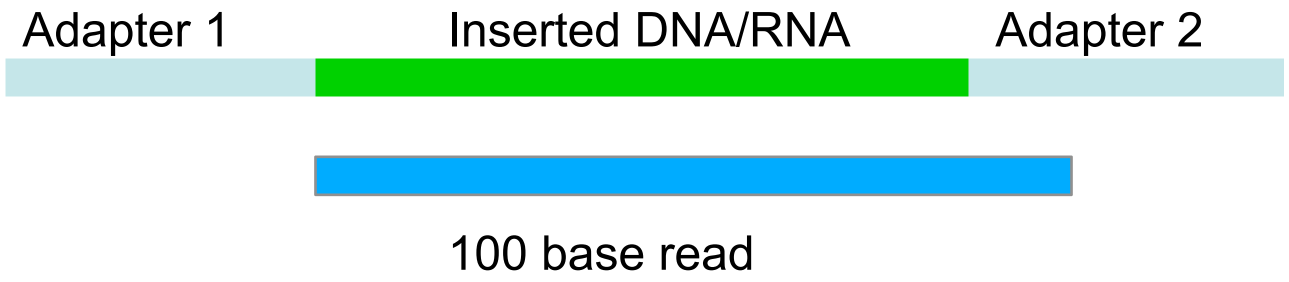
- These quality scores are called PHRED scores. They were first introduced for capillary sequencers.
- PHRED scores are determined by the sequencer that directly rates how reliable the measured signal is

Considerations: Read trimming

- Sequencers have systematic errors
- Illumina sequencers have a higher error rate at the first few bases (1-5)
- Basically all sequencers have increasing error rate towards the end of the read

- Hard trimming: trim a fixed number of bases from the beginning and/or end
- Quality trimming: cut the end of the read as soon as the base quality drops below a threshold

Considerations: Read trimming



- If the inserted DNA/RNA fragment is too short, the read will contain part of the adapter
- Adapter trimming can be challenging if
  - if the insert is 90 – 100 bases
  - if the read has many sequencing errors

Multiple Alignments

- A read may have multiple valid alignments with identical or similarly good alignment scores
- Aligners allow to choose different reporting strategies:
  - Randomly select one alignment from the top-scoring alignments
  - Report all alignments that are within *delta* of the top-scoring alignment; clip if more than *Nmax* alignments are found
  - Report only alignments if they are unique (no other alignment within *delta* of the alignment score)
  - Do not report anything if more than *Nmax* valid alignments are found
  - …
- Whether a read has a **unique alignment** depends on
  - the read sequence and the sequence homology of the organism
  - the search algorithm of the aligner
  - the reporting strategy of the aligner

Read Alignment Summary (I)

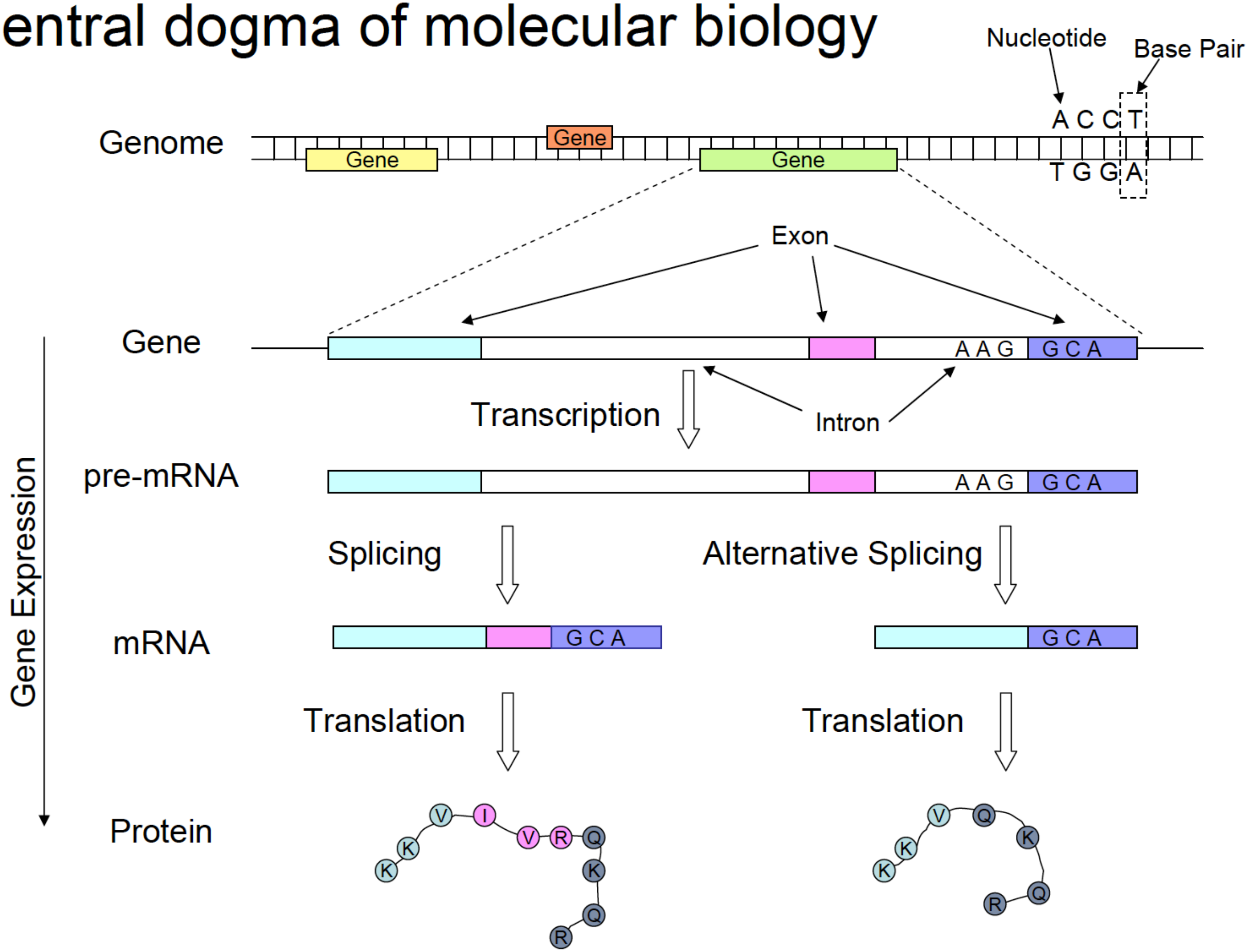
How to map billions of reads?

- The major aligners use the Burrows-Wheeler-Transform to create an index of the reference. Even for the human genome the index fits into 3GB RAM.
- Reads are aligned by **index lookup** not by sequence comparison
- The lookup of a perfect match read is faster than loading the read and writing the alignment coordinates to the output file!
- If the lookup fails because of SNPs or sequencing errors, an actual sequence comparison is done –and this can take time!

Read Alignment Summary (II)

- Alignment is a score optimization problem
- Aligners find the aligment with the **highest alignment score**
- Aligners do not run an extensive search, they use **index lookup and heuristics** to find the alignment with highest score
- If the heuristics are well chosen and the scores are well defined, then the alignment will be very often the correct alignment

Central dogma of molecular biology

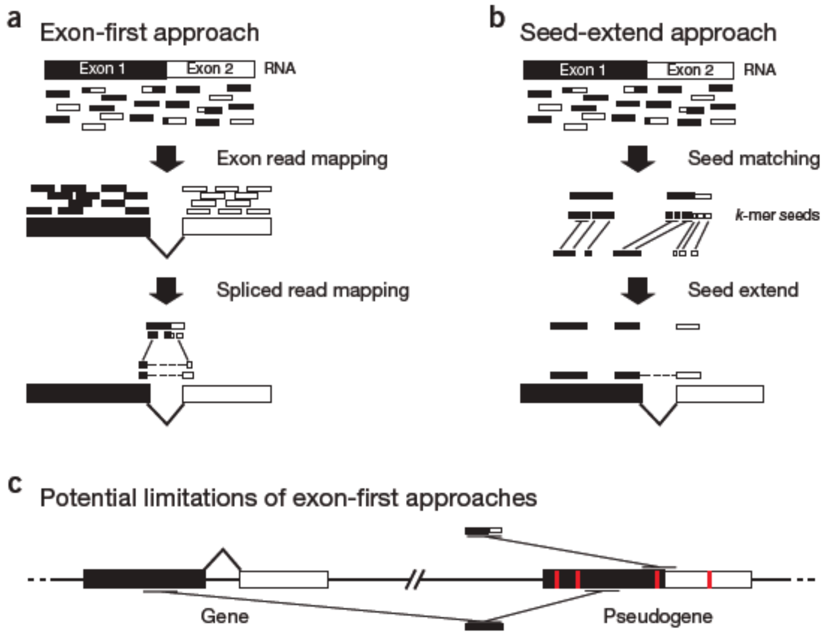


92–94% of human genes undergo alternative splicing,

RNA-seq Mapping

- Mapping targets:
  - transcriptome
  - splice junction library
  - genome

- Mouse retina 60+60bp reads:
  - 41 of 91 Mio map to junctions

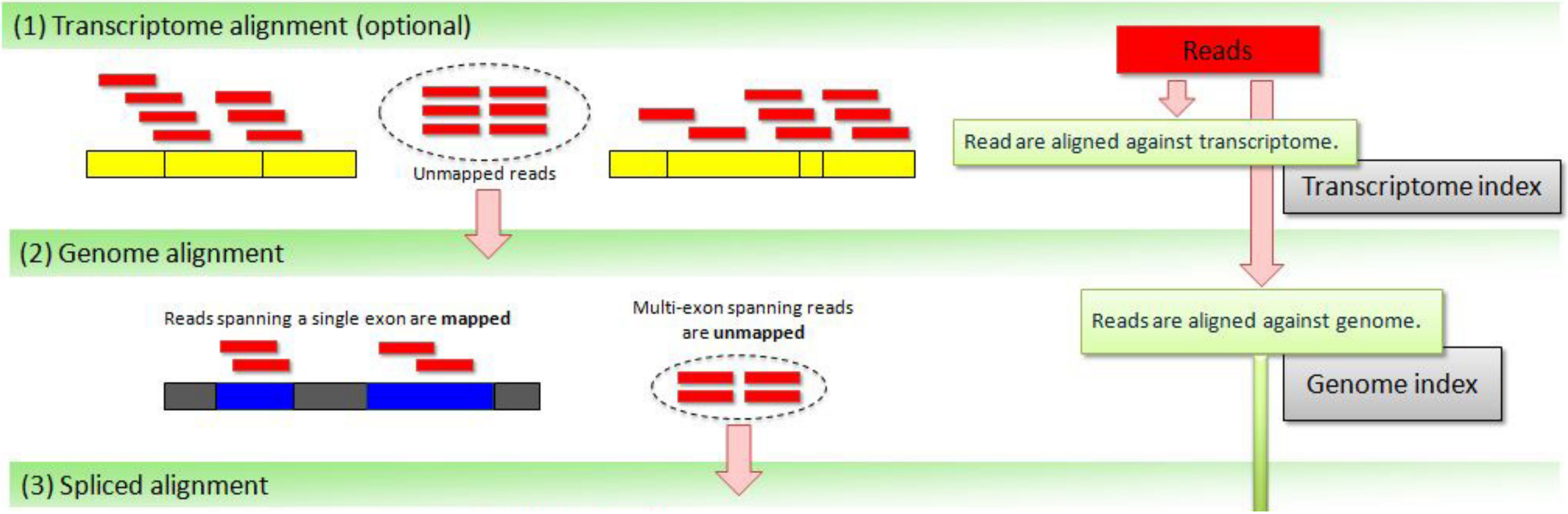


Strategies:

- Exon-first: fast
- Seed-extend:
  - good with polymorphisms
  - simultaneous spliced/ unspliced mapping

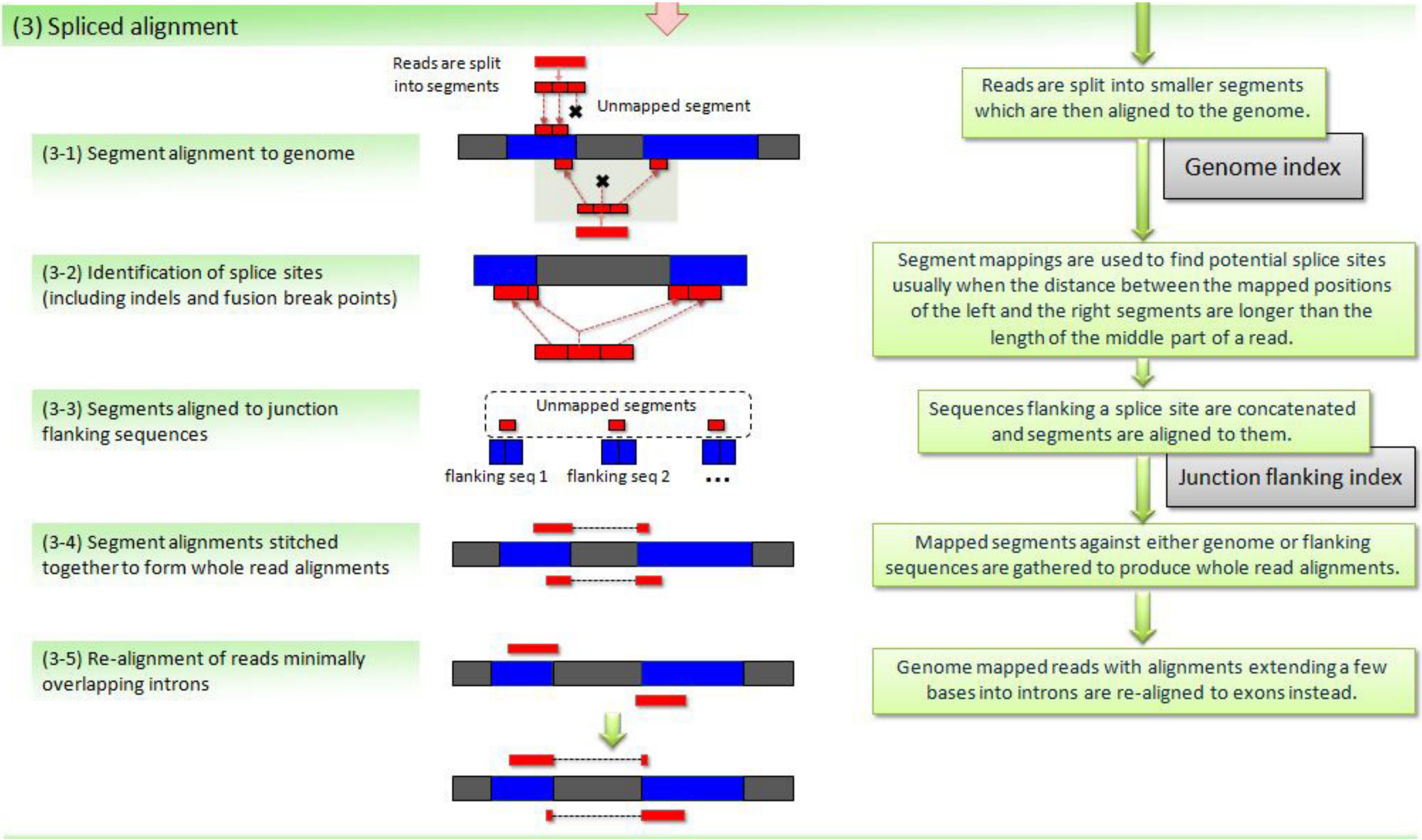
Garber etal., Nature Methods, 2011

Tophat2 pipeline


(1) Transcriptome alignment (optional)
Unmapped reads
Reads
Read are aligned against transcriptome.
Transcriptome index
(2) Genome alignment
Reads spanning a single exon are **mapped**
Multi-exon spanning reads are **unmapped**
Reads are aligned against genome.
Genome index
(3) Spliced alignment

Kim et al. Genome Biology 2013

Tophat2 pipeline



(3) Spliced alignment

(3-1) Segment alignment to genome

Reads are split into smaller segments which are then aligned to the genome.

(3-2) Identification of splice sites (including indels and fusion break points)

Segment mappings are used to find potential splice sites usually when the distance between the mapped positions of the left and the right segments are longer than the length of the middle part of a read.

(3-3) Segments aligned to junction flanking sequences

Sequences flanking a splice site are concatenated and segments are aligned to them.

(3-4) Segment alignments stitched together to form whole read alignments

Mapped segments against either genome or flanking sequences are gathered to produce whole read alignments.

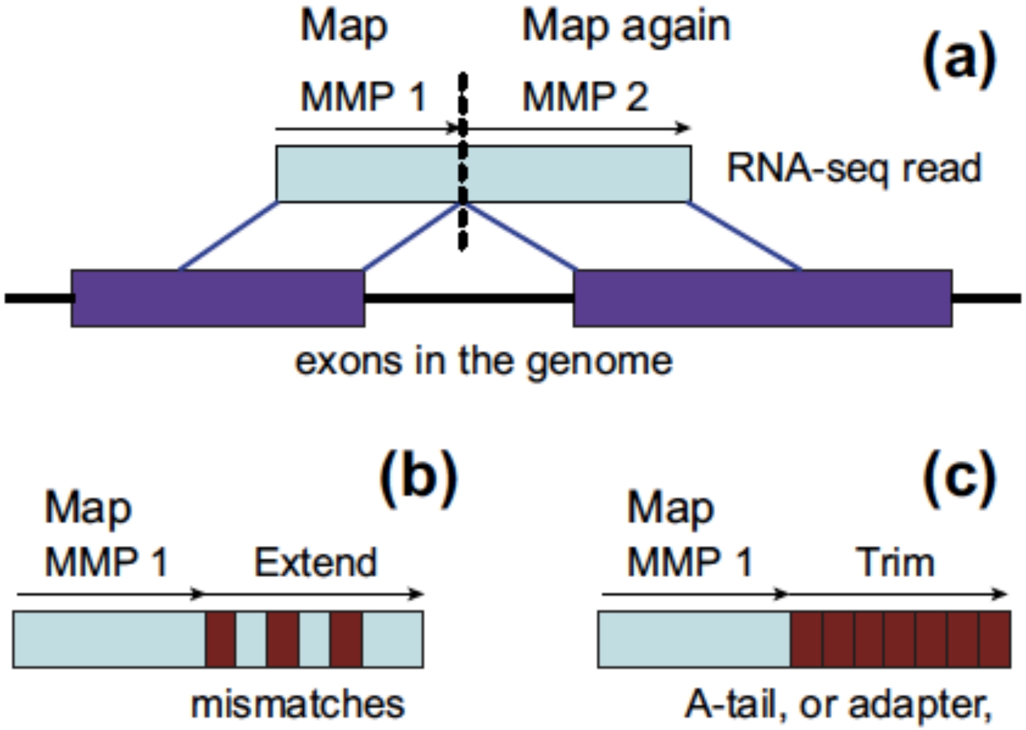
(3-5) Re-alignment of reads minimally overlapping introns

Genome mapped reads with alignments extending a few bases into introns are re-aligned to exons instead.

Kim et al. Genome Biology 2013

STAR: universal RNA-seq aligner

- Designed to align the non-contiguous sequences directly to the reference genome
- Steps:
  - Search Maximal Mappable Prefix (MMP)
  - clustering/stitching/scoring



Comparison of Aligners

- Tradeoff:
  - speed
  - RAM
  - accuracy
  - sensitivity

- http://www.ecseq.com/support/benchmark

Performance comparison of RNA-seq mappers

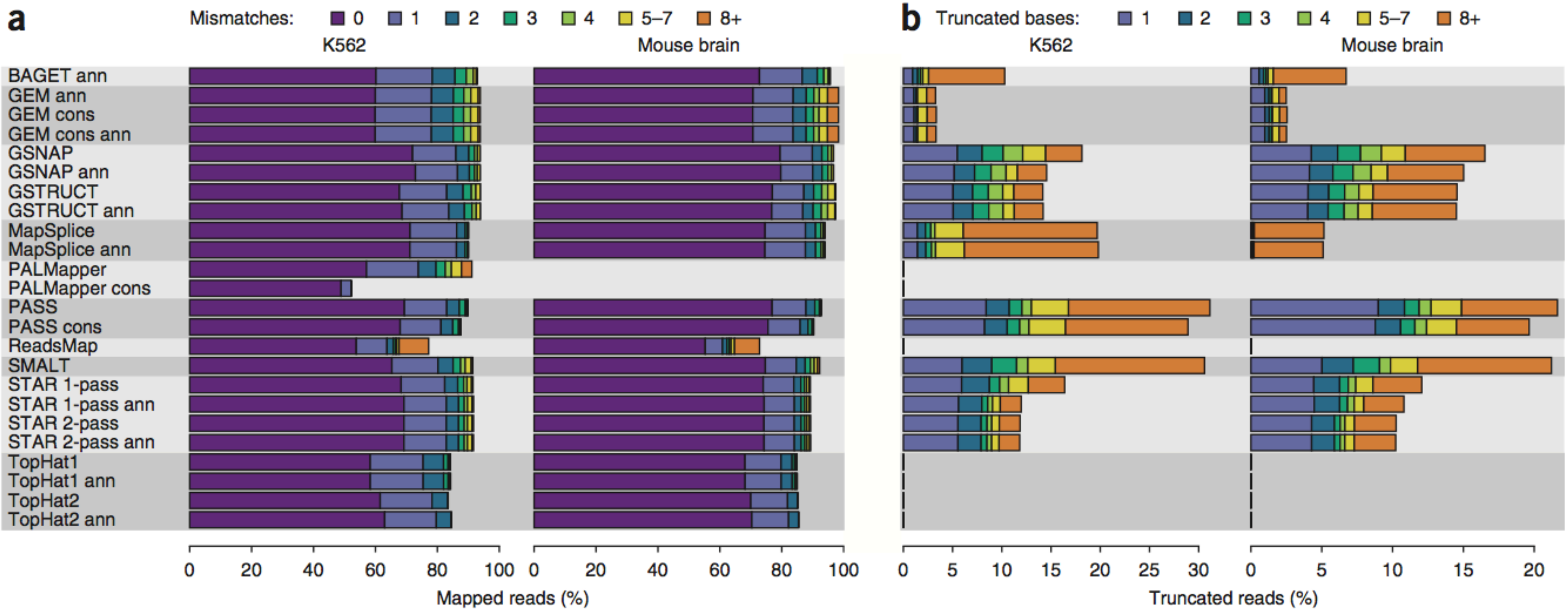


Figure 2 | Mismatch and truncation frequencies. (**a**) Percentage of sequenced reads mapped with the indicated number of mismatches. (**b**) Percentage of sequenced reads truncated at either or both ends. Bar colors indicate the number of bases removed.

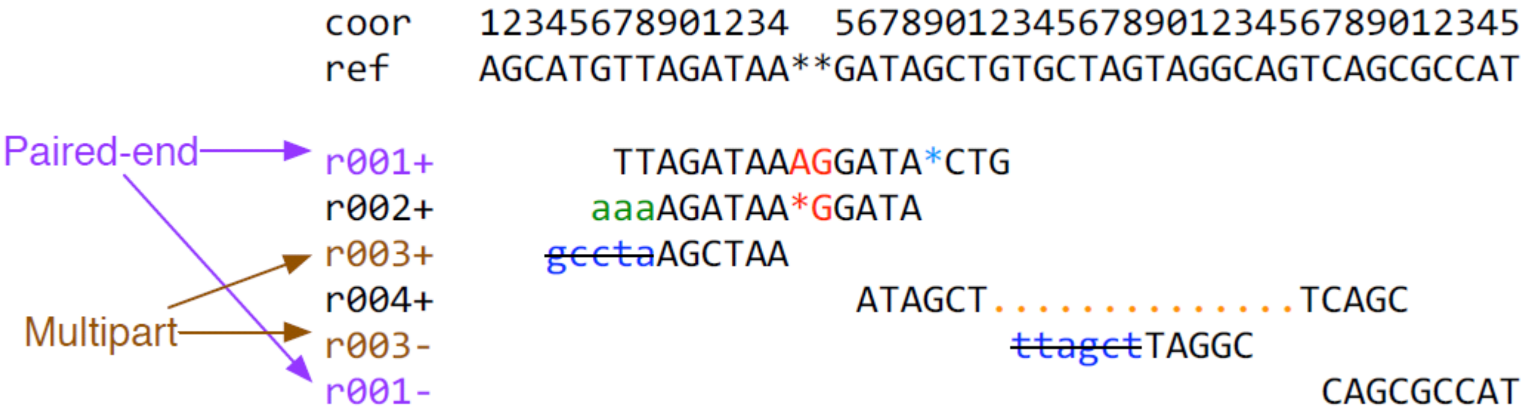
Engström et al. Systematic evaluation of spliced alignment programs for RNA-seq data. Nat Methods. 2013.

Sequence / Alignment (SAM) files

- **SAM (Sequence Alignment/Map)**
  - Single unified format for storing read alignments to a reference genome
  - Large plain text file

- **BAM (Binary Alignment/Map)**
  - Binary equivalent of SAM
  - Compressed data plus index (bai)
  - Developed for fast processing/indexing

An example of read mapping

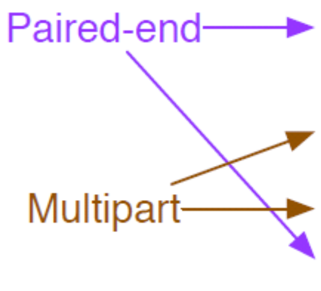


```
coor    12345678901234  5678901234567890123456789012345
ref     AGCATGTTAGATAA**GATAGCTGTGCTAGTAGGCAGTCAGCGCCAT

r001+         TTAGATAAAGGATA*CTG
r002+        aaaAGATAA*GGATA
r003+      ~~gccta~~AGCTAA
r004+                    ATAGCT..............TCAGC
r003-                           ~~ttagct~~TAGGC
r001-                                         CAGCGCCAT
```

Slide modified from Heng Li's presentation

Corresponding SAM file



```
coor    12345678901234  5678901234567890123456789012345
ref     AGCATGTTAGATAA**GATAGCTGTGCTAGTAGGCAGTCAGCGCCAT

r001+         TTAGATAAAGGATA*CTG
r002+        aaaAGATAA*GGATA
r003+      ~~gccta~~AGCTAA
r004+                    ATAGCT..............TCAGC
r003-                           ~~ttagct~~TAGGC
r001-                                         CAGCGCCAT
```

```
                @SQ SN:ref LN:45
Ins & padding   r001 163 ref  7 30 8M2I4M1D3M = 37  39 TTAGATAAAGGATACTA *
Soft clipping   r002   0 ref  9 30 3S6M1P1I4M *  0   0 AAAAGATAAGGATA    *
                r003   0 ref  9 30 5H6M       *  0   0 AGCTAA     *    NM:i:1
Splicing        r004   0 ref 16 30 6M14N5M    *  0   0 ATAGCTTCAGC       *
Hard clipping   r003  16 ref 29 30 6H5M       *  0   0 TAGGC      *    NM:i:0
                r001  83 ref 37 30 9M         =  7 -39 CAGCGCCAT         *
```

QNAME RNAME MAPQ? MRNM ISIZE QUAL

FLAG POS CIGAR? MPOS SEQ

Slide modified from Heng Li's presentation

CIGAR string - compact representation of an alignment

- M - match or mismatch
- I – insertion
- D – deletion
- S - soft clip
  - Clipped sequences stored in SAM
- H - hard clip
  - Clipped sequences not stored in SAM
- N – skipped reference bases, splicing

```
Match/mismatch,indels

Ref:   ACGCAGTG--GT
Read:  ATGCA-TGCAGT
Cigar:5M1D2M2I2M
```

```
Soft clipping

REF:   ATCGTGTAACCTGACTAGTTAA
READ:  gggGTGTAACC-GACTAGgggg
Cigar: 3S8M1D5M4S
```

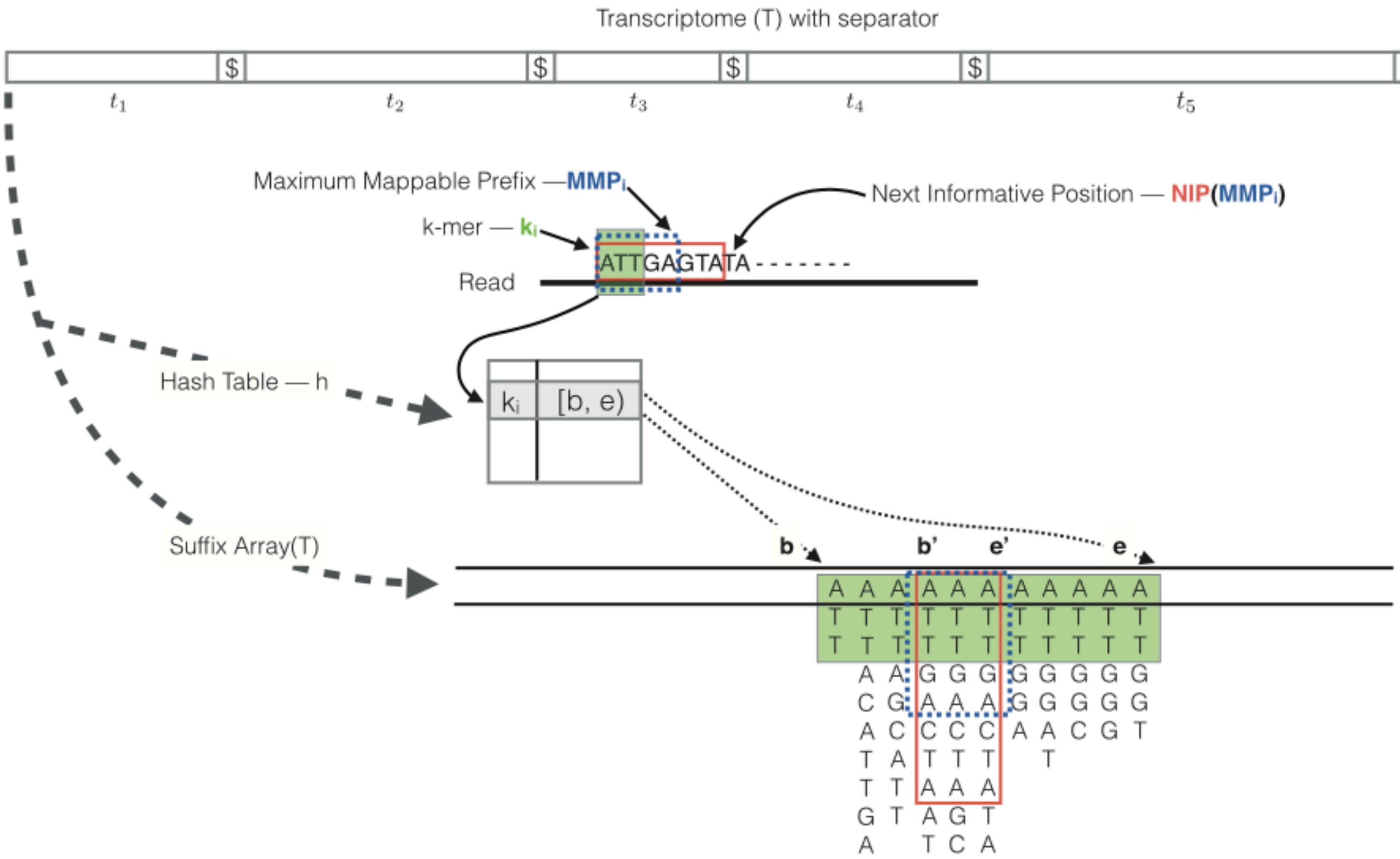
```
Hard clipping

REF:   ATCGTGTAACCTGACTAGTTAA
READ:  gggGTGTAACC-GACTAGgggg
Cigar: 3H8M1D5M4H
```

Quasi-Alignments or Pseudo-Alignments

- compare reads only against a hashed transcriptome index
- use only perfect alignments of shore read fragments (k-mers, e.g. with k=31).

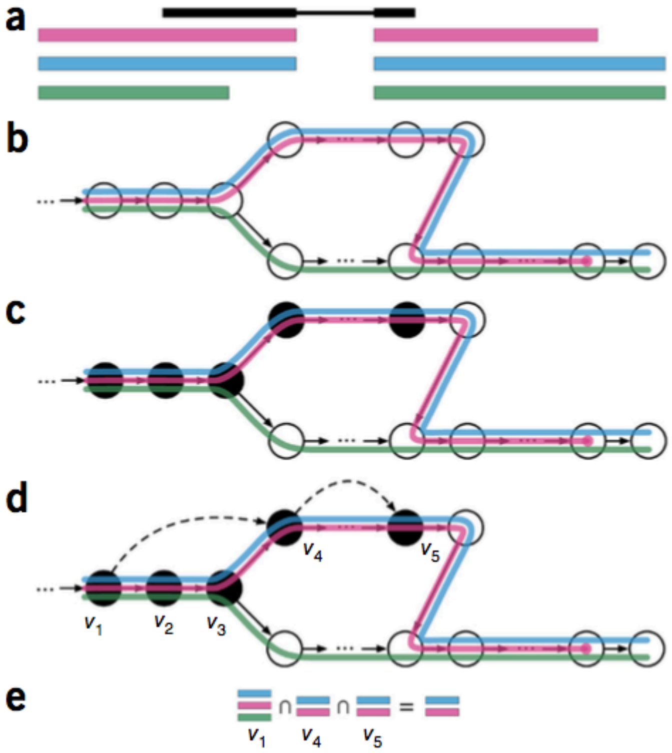
RapMap / Salmon

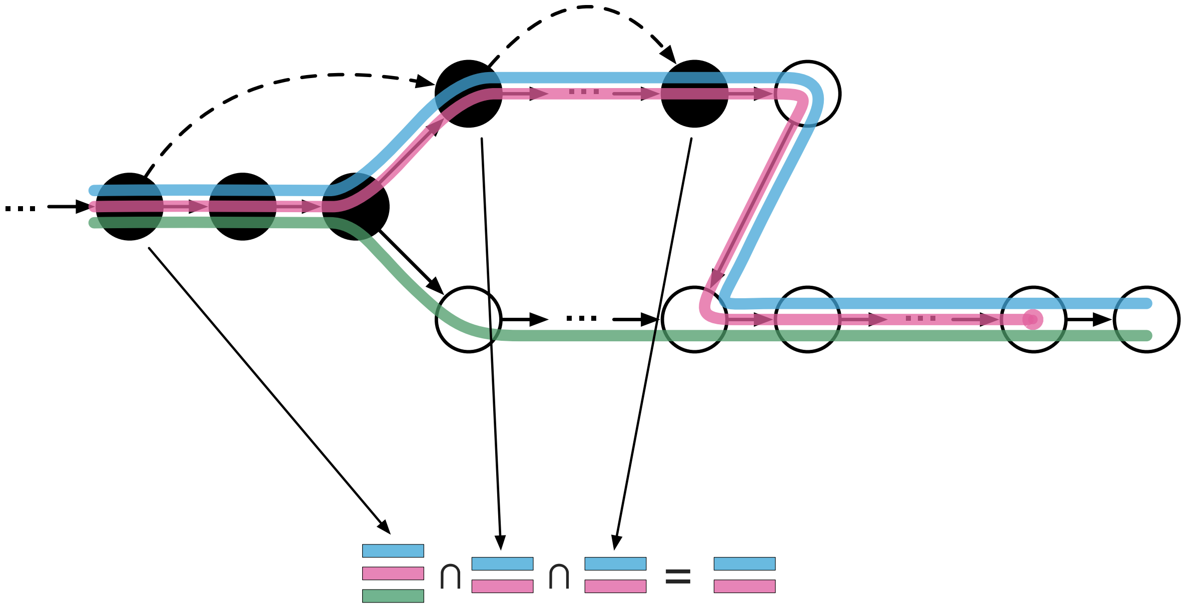


Srivastava, Bioinformatics 2016

kallisto pseudo-alignments

- Create every k-mer in the transcriptome (k=31), build de Bruin Graph and color each k-mer
- Preprocess the transcriptome to create the T-DBG





Each k-mer appears in a set of transcripts

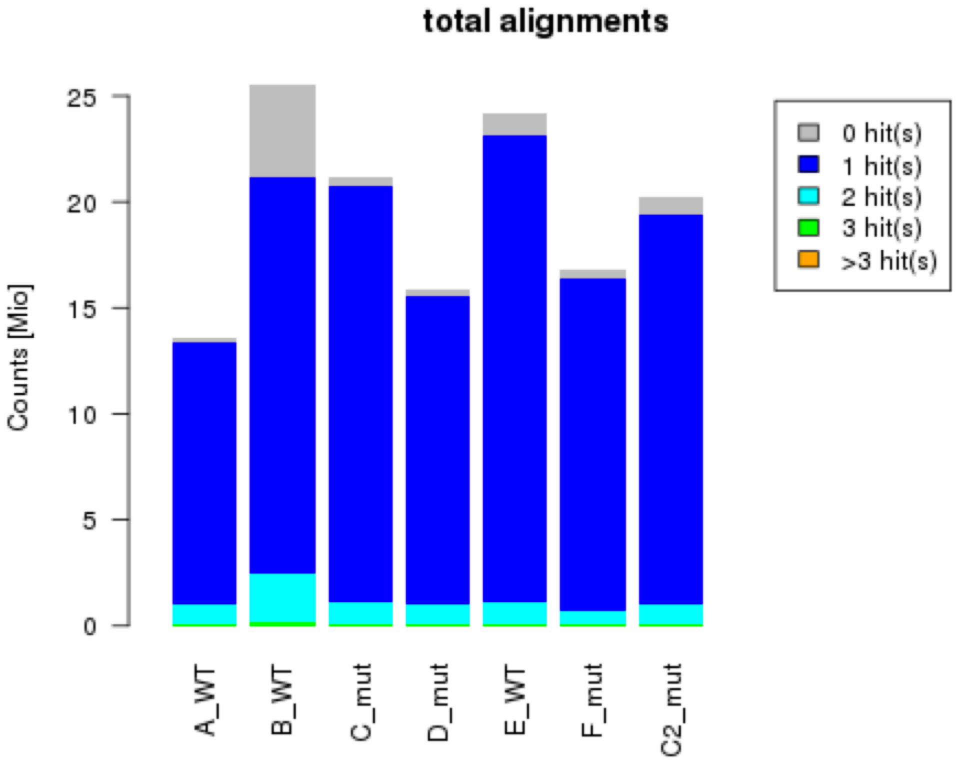
The intersection of all sets is our pseudoalignment

Can jump over k-mers in the T-DBG that provide same information Jumping provides ~8x speedup over chekcing all k-mers

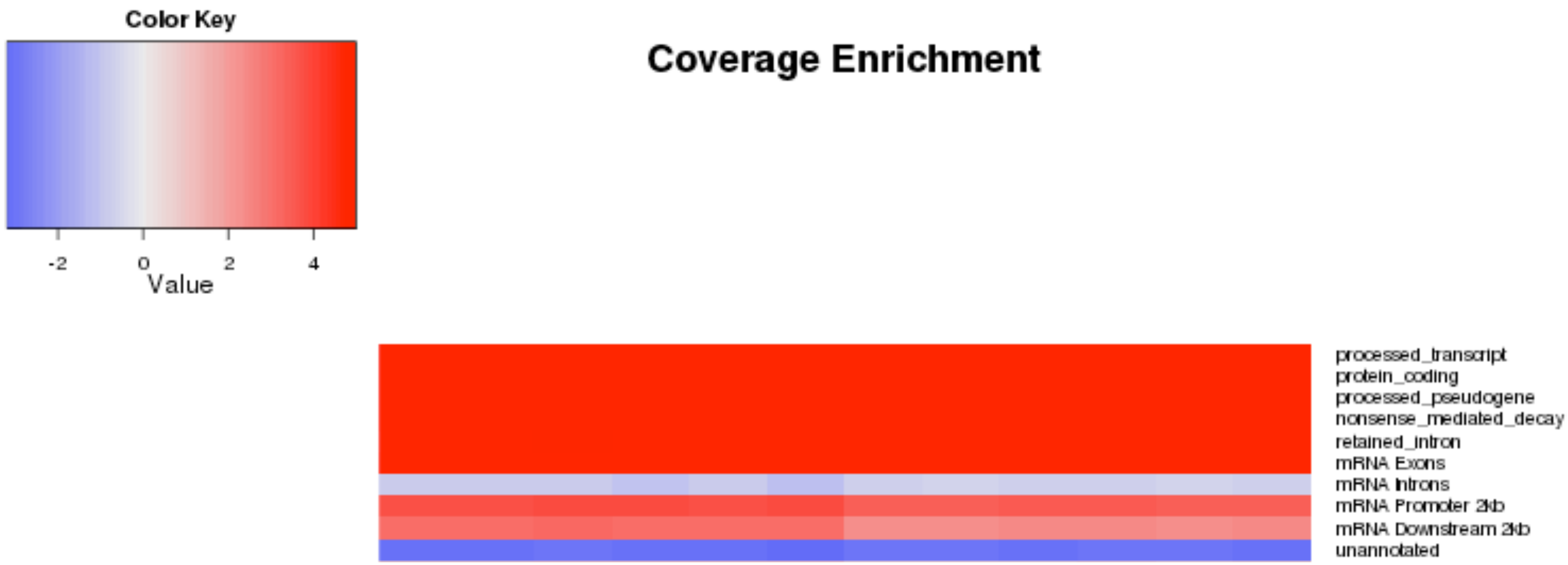
Mapping QC: Mapping statistics

- How well did my sequence library align to my reference?

- Summary statistics (per read library)
  - % reads with unique alignmen
  - % reads with multiple alignme
  - % reads with no alignment
  - % reads properly paired
    (for paired end libraries)



Mapping QC: Profiling efficiency by transcript annotation



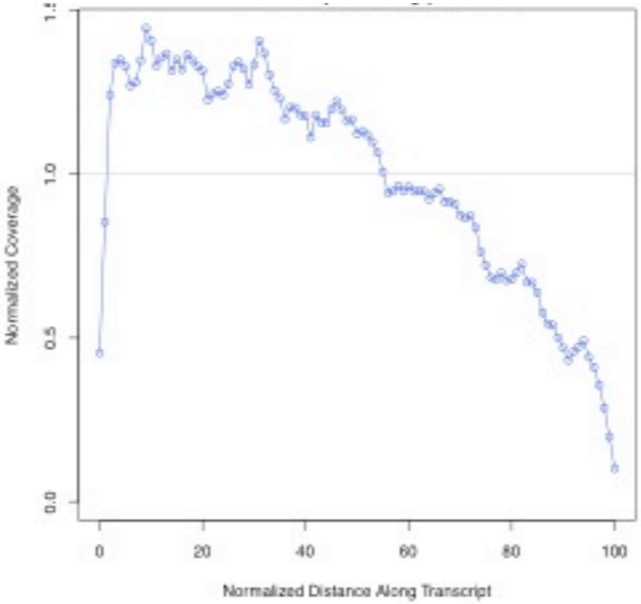
- Transcript annotation: intron, exon, up/down stream, unannotated
- Expression profiling efficiency

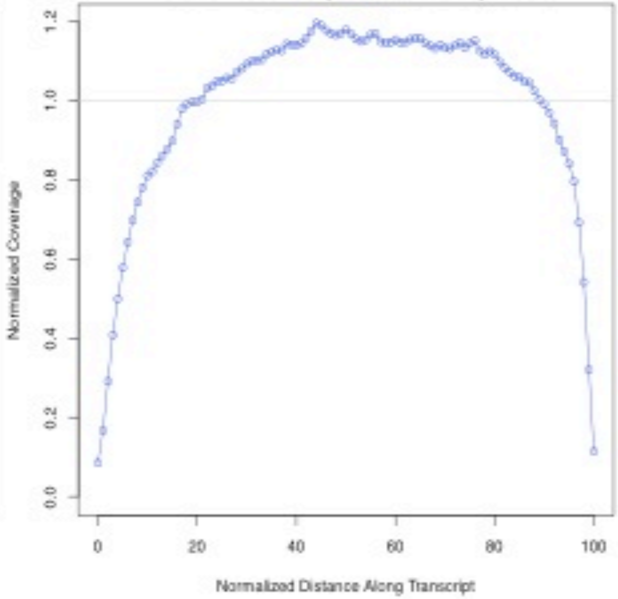
Mapping QC: Coverage bias

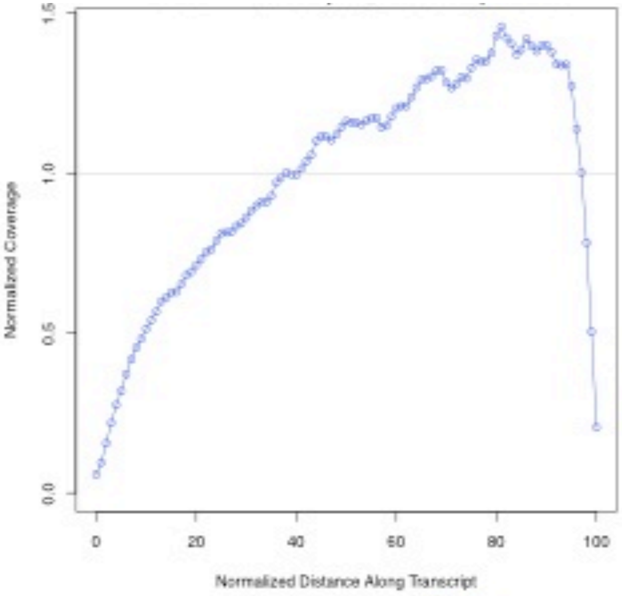
3’ bias

No bias

5’ bias

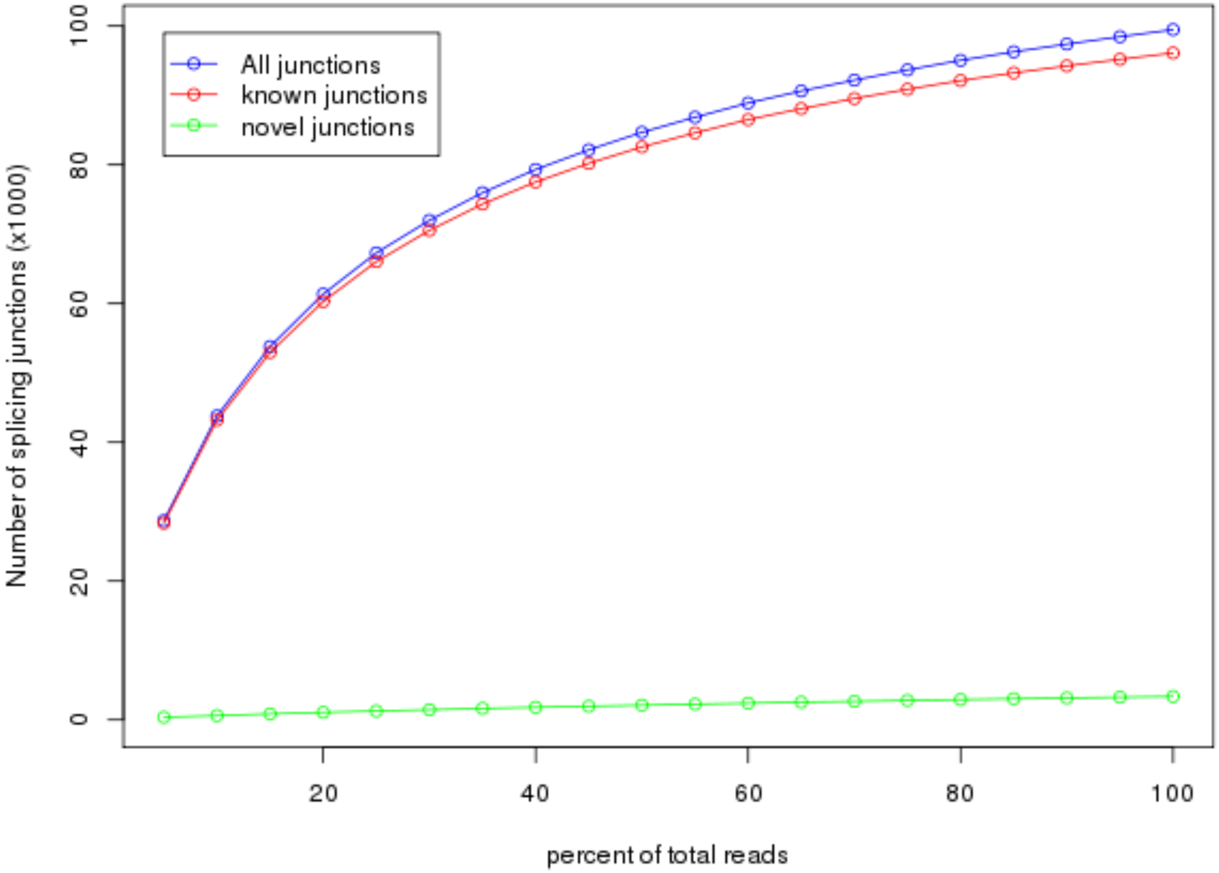






Zuojian Tang NYU Langone MEDICAL CENTER

Mapping QC: Junction saturation



- Depth needed for alternative splicing analysis
- All annotated splice junctions are detected - a saturated RNA-seq dataset

RNA-seq mapping QC tools

- RNA-SeQC
  - https://confluence.broadinstitute.org/display/CGATools/RNA-SeQC
- EVER-seq (RSeQC)
  - http://code.google.com/p/rseqc/
- Qualimap
  - http://qualimap.bioinfo.cipf.es/