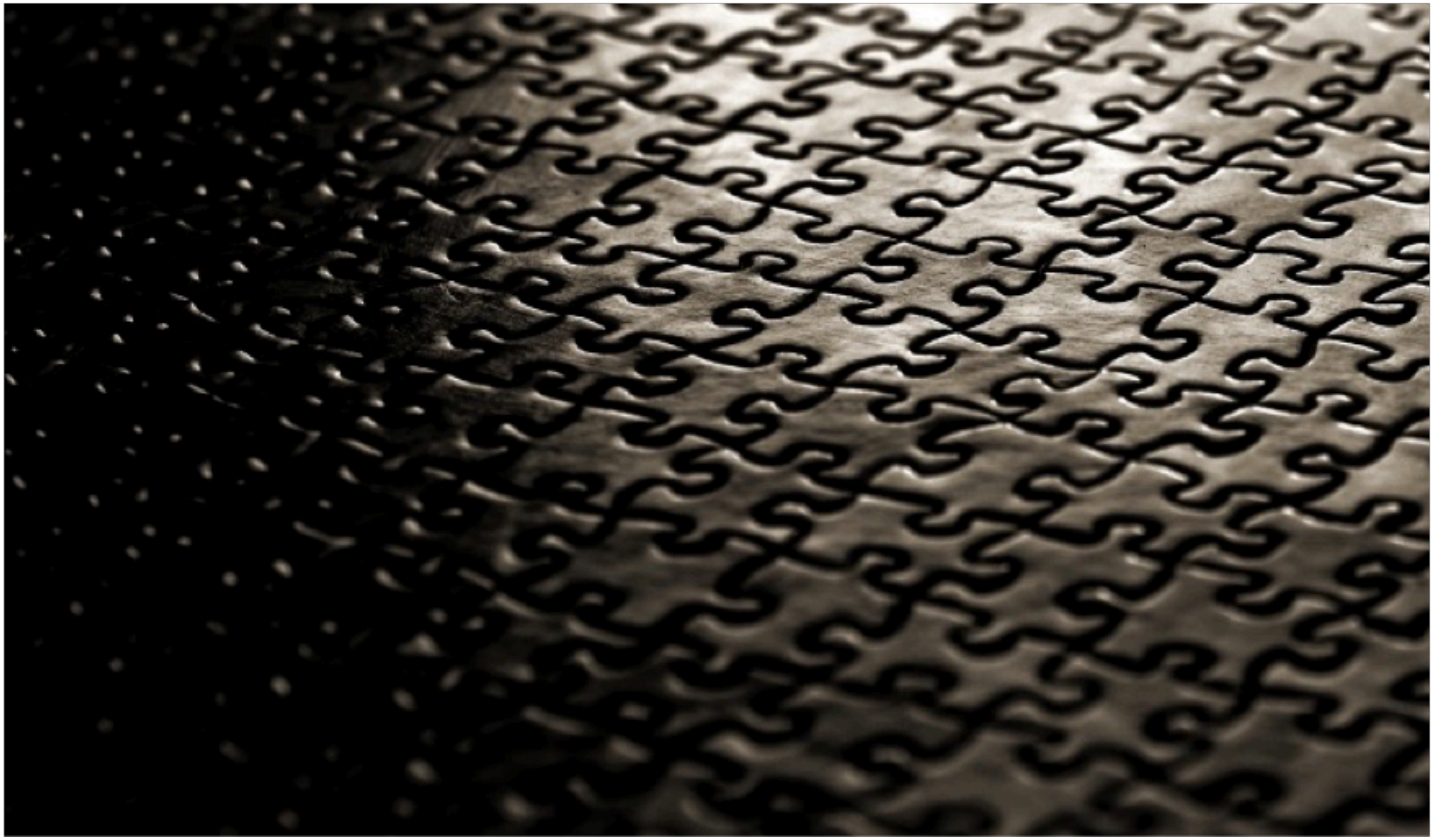
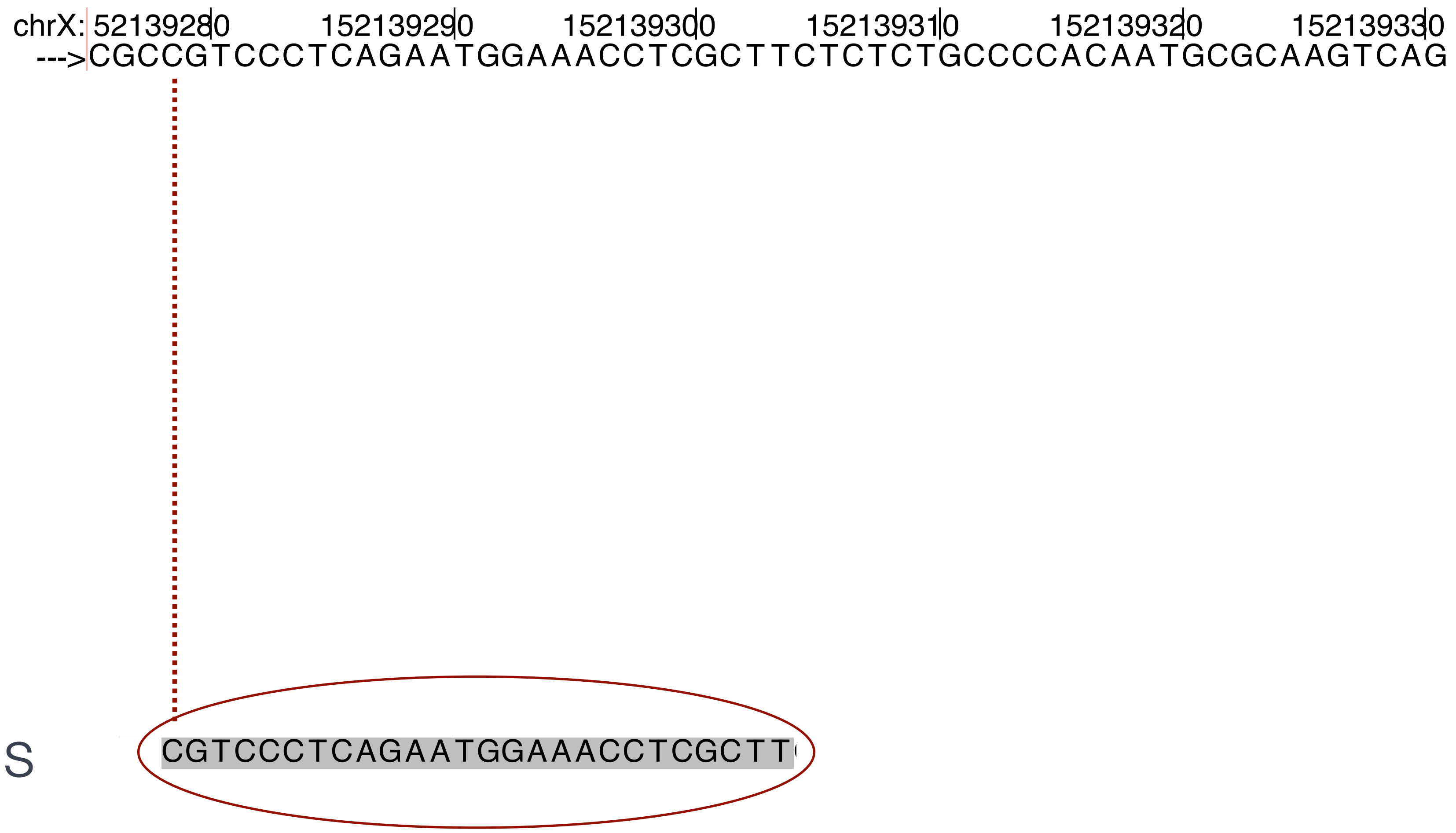


Expression quantification: tools and theory

What does it mean to “align” reads to a reference genome/transcriptome?

Reference

Sequence reads

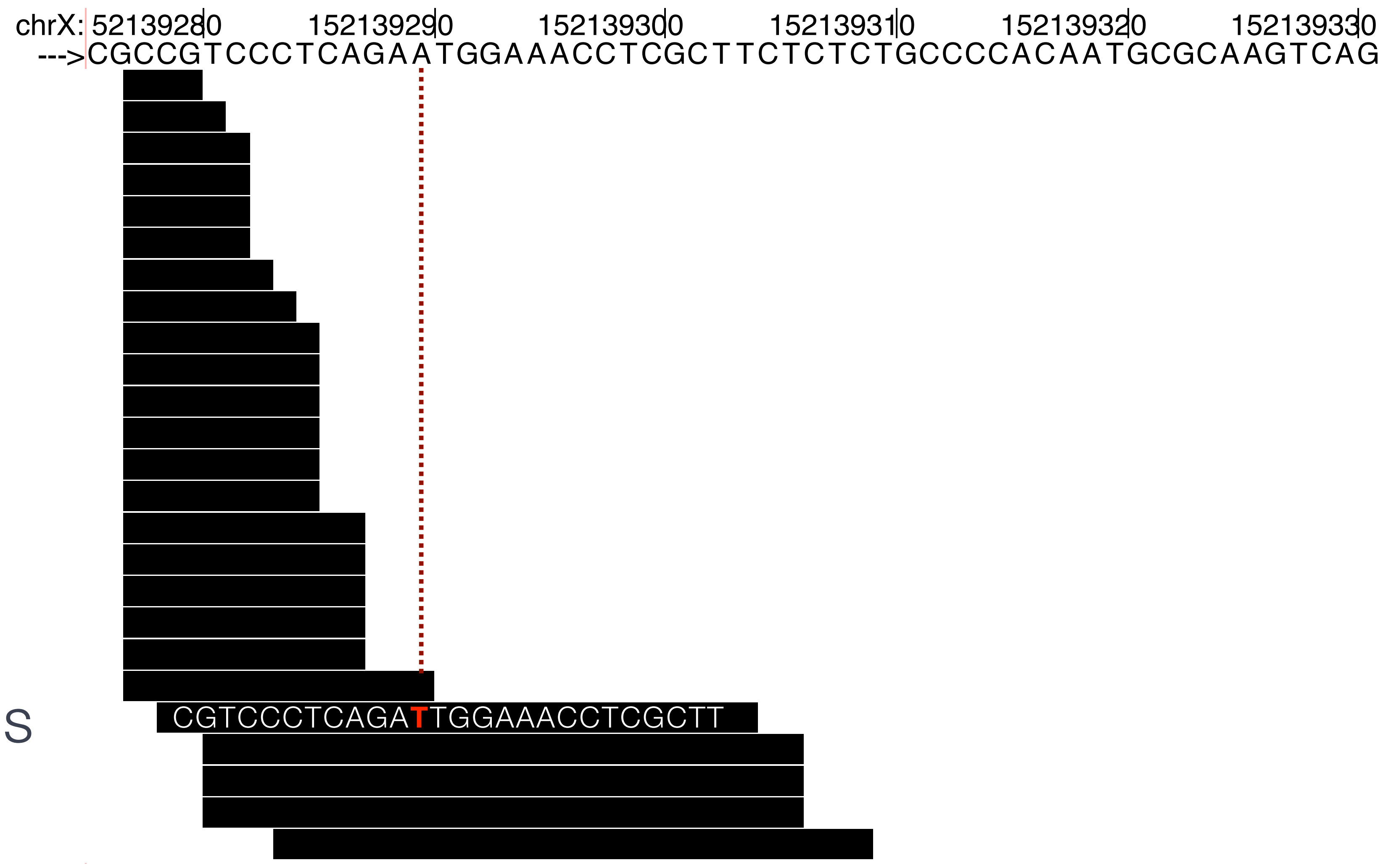


```
chrX: 52139280      152139290      152139300      152139310      152139320      152139330
 --->CGCCGTCCCTCAGAATGGAAACCTCGCTTCTCTCTGCCCCACAATGCGCAAGTCAG

         CGTCCCTCAGAATGGAAACCTCGCTT
```

A simple case of string matching

Reference


chrX: 52139280
152139290
152139300
152139310
152139320
152139330
--->CGCCGTCCCTCAGAATGGAAACCTCGCTTCTCTCTGCCCCACAATGCGCAAGTCAG
CGTCCCTCAGATTGGAAACCTCGCTT

Sequence reads

A simple case of string matching?

Non-comprehensive list of challenges

- Large, incomplete and repetitive genomes
- Short reads: 50-150 bp
    - Non-unique alignment
    - Sensitive to non-exact matching (variants, sequencing errors)
- Massive number of short reads
- Compute capacity for efficient base-to-base mapping

How do we overcome these challenges?

Building an index

- Having an index of the reference sequence provides an efficient way to search

Reference data files

- **Genome** assembly gives us the nucleotide sequence of the reference genome.
    - It is in the form of a Fasta file
    - **Genome build/release** represents a version of the genome with improvements (i.e. gaps filled, mistakes corrected).
- **Transcriptome** gives us the complete set of transcripts
    - Fasta file format for direct alignment
    - GTF (gene transfer format) file as a companion to genome alignment (same source and version as genome build)

Building an index

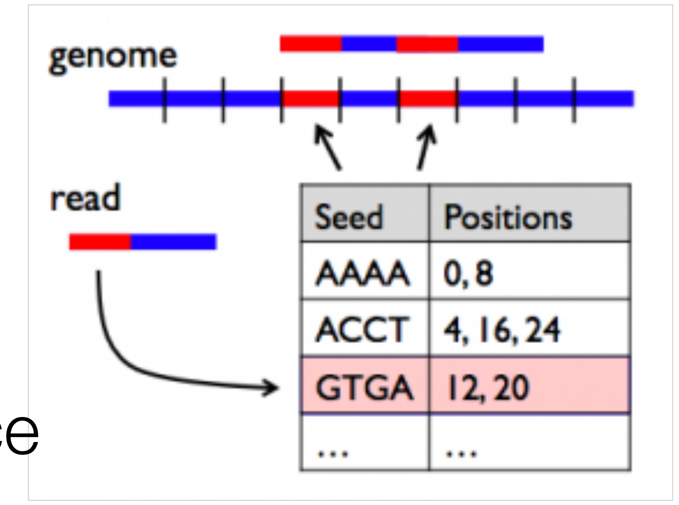
- Having an index of the reference sequence provides an efficient way to search
- Many algorithmic solutions exist for this speedup

Commonly used indexing methods

- Hash-based (Salmon, Kallisto)
- Suffix arrays (Salmon, STAR)
- Burrows-Wheeler Transform (BWA, Bowtie2)

Hash-based alignment (circa 1990)

- Pick k-mer size, build lookup of every k-mer in the reference mapped to its positions (the index)
- Break the query into k-mers
- Seed-and-extend strategy
- For BLAST, 100% match the query k-mer to reference then extend until score drops below 50%
- 0.1 - 1 sec per query; not feasible for NGS data



Seed	Positions
AAAA	0, 8
ACCT	4, 16, 24
GTGA	12, 20
...	...

Hash-based alignment (present day)

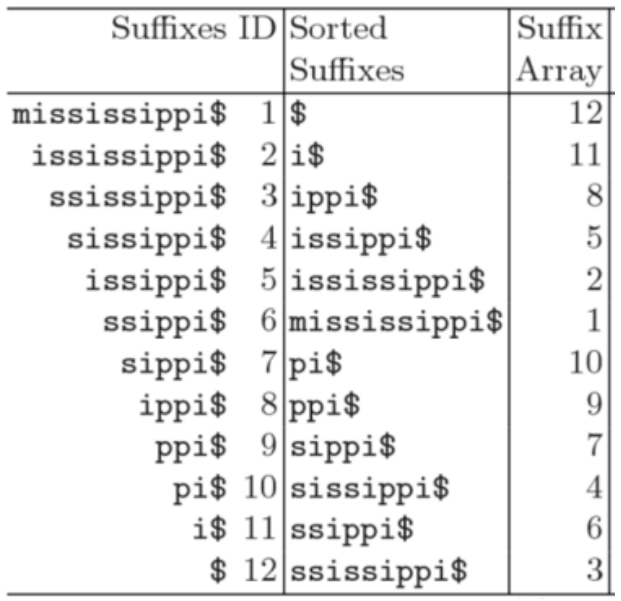
- Need to make some concessions on sensitivity by making adaptations for use on NGS data:
    - allow for mismatches and/or gaps (ELAND, MAQ, SOAP)
    - using multiple seeds (BLAT, ELAND2)
- Memory intensive and slower (~16GB RAM required for hg19)
- Simpler in design but more sensitive

Suffix arrays

- A sorted table of all suffixes (substrings) of a given string
- A suffix array will contain integers that represent the starting indexes of the all the suffixes of a given string, after the aforementioned suffixes are sorted
- Requires large amount of memory to load the suffix array and genome sequence prior to alignment
- Popular Tools:

  STAR (2012), Salmon

Let the given string be “`mississippi`”

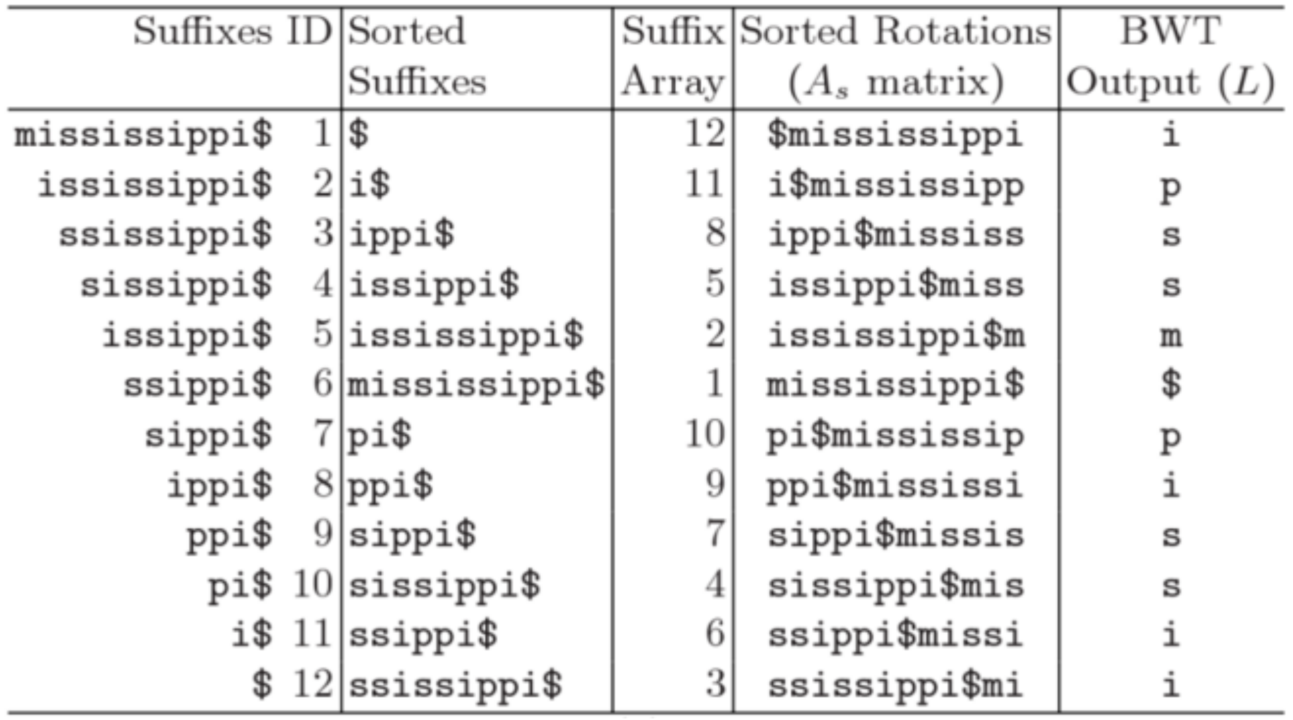


Suffixes	ID	Sorted Suffixes	Suffix Array
mississippi$	1	$	12
ississippi$	2	i$	11
ssissippi$	3	ippi$	8
sissippi$	4	issippi$	5
issippi$	5	ississippi$	2
ssippi$	6	mississippi$	1
sippi$	7	pi$	10
ippi$	8	ppi$	9
ppi$	9	sippi$	7
pi$	10	sissippi$	4
i$	11	ssippi$	6
$	12	ssissippi$	3

The suffix array will be:
{12, 11, 8, 5, 2, 1, 10, 9, 7, 4, 6, 3}

Burrows-Wheeler transform

- A compressed form of suffix arrays
- Tends to put runs of the same character together rather than alphabetically, which makes the compression work well



Suffixes	ID	Sorted Suffixes	Suffix Array	Sorted Rotations (A_s matrix)	BWT Output (L)
mississippi$	1	$	12	$mississippi	i
ississippi$	2	i$	11	i$mississipp	p
ssissippi$	3	ippi$	8	ippi$mississ	s
sissippi$	4	issippi$	5	issippi$miss	s
issippi$	5	ississippi$	2	ississippi$m	m
ssippi$	6	mississippi$	1	mississippi$	$
sippi$	7	pi$	10	pi$mississip	p
ippi$	8	ppi$	9	ppi$mississi	i
ppi$	9	sippi$	7	sippi$missis	s
pi$	10	sissippi$	4	sissippi$mis	s
i$	11	ssippi$	6	ssippi$missi	i
$	12	ssissippi$	3	ssissippi$mi	i

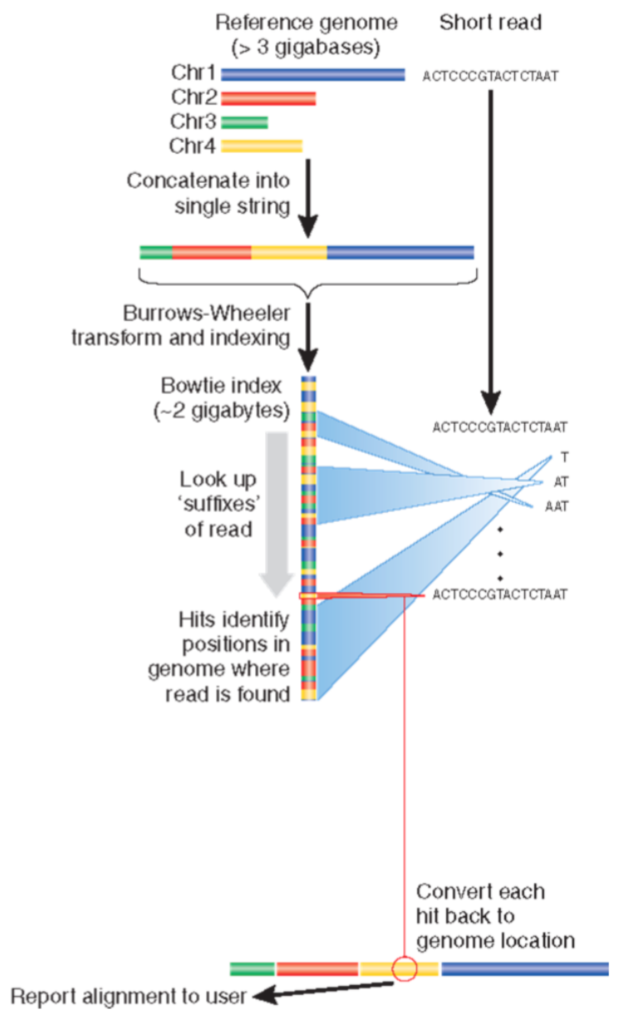
Burrows-Wheeler transform

- Much less memory because of compression; ~1.5 GB of RAM required for hg19 index
- But compression results in diminished efficiency of the string search operations
- Popular Tools:

    Bowtie2 (2012)

    SOAP2

    BWA-MEM (2013)



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
Report alignment to user

Building an index

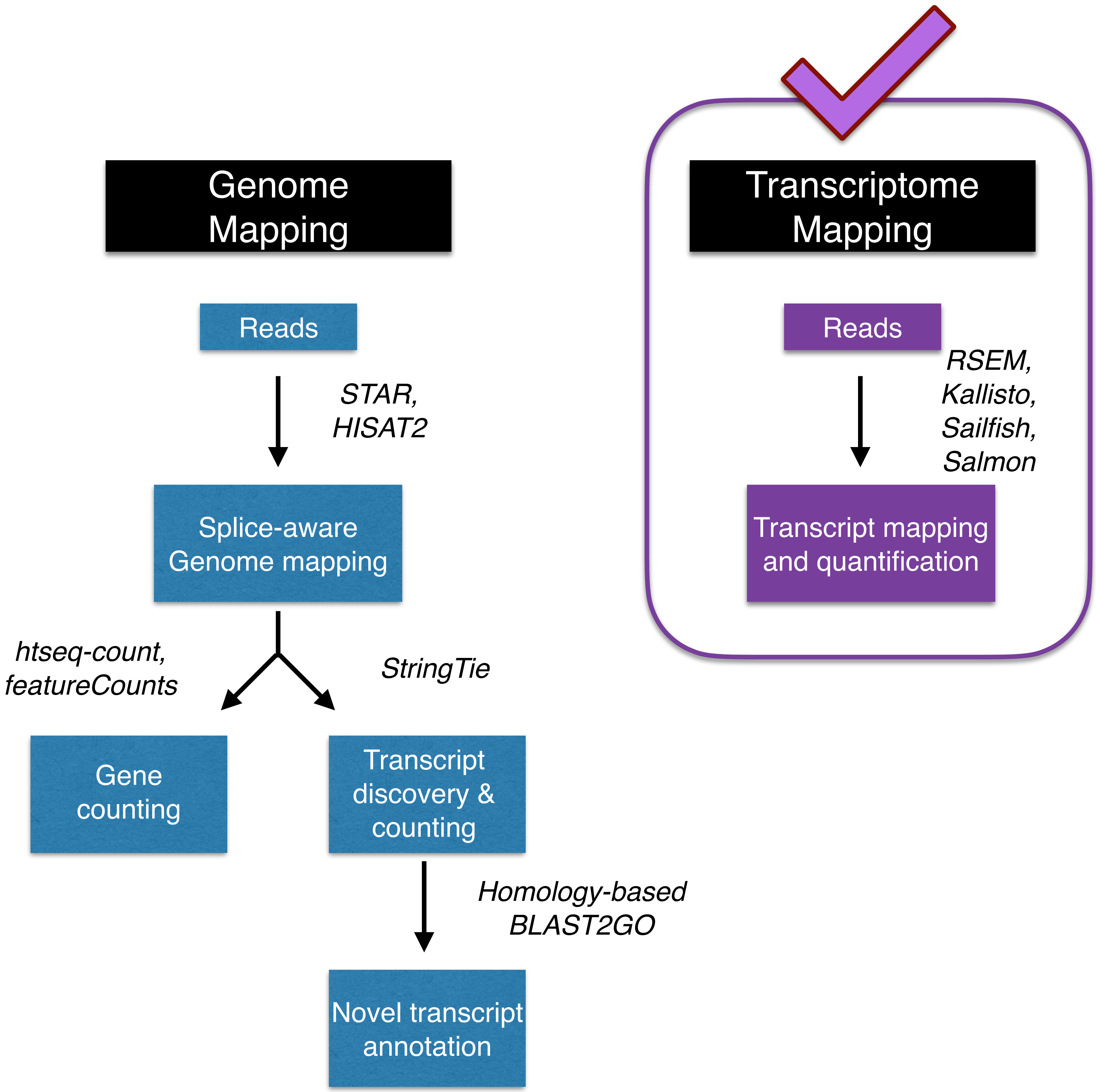
- Having an index of the reference sequence provides an efficient way to search
- Many algorithmic solutions exist for this speedup
- Once an index is built, it can be queried any number of times - e.g. you build the mm10 genome once for any number of mouse RNA-seq experiments.
- Every genome or transcriptome build requires a new index for the specific tool in question.

We will be using *Salmon*,

(a) a “lightweight alignment” tool

(b) that performs alignment and quantification together (Hash-based alignment + suffix arrays)

(c) using the reference transcriptome fasta file (not the whole genome).


Genome Mapping
Reads
STAR, HISAT2
Splice-aware Genome mapping
htseq-count, featureCounts
StringTie
Gene counting
Transcript discovery & counting
Homology-based BLAST2GO
Novel transcript annotation
Transcriptome Mapping
Reads
RSEM, Kallisto, Sailfish, Salmon
Transcript mapping and quantification

Why use lightweight alignment?

- Approaches avoid base-to-base alignment
- Faster, more efficient (~ >20x faster than alignment-based)
- Improved accuracy for transcript-level quantification
- Improvements in accuracy for gene-level quantification**
- Tools include: Kallisto (quasi-aligner), Sailfish (kmer-based), Salmon (quasi-aligner), RSEM

**doi: 10.12688/f1000research.7563.2

Lightweight alignment and quantification using Salmon

- Reference = FASTA file of all **transcript sequences** for the organism
- Reference **Index:** (2 components)
  - Suffix array
  - Hash table (mapping each transcript to its location in the SA)
- Output:
  - **Abundance estimates** of reads mapping to each transcript listed in the reference

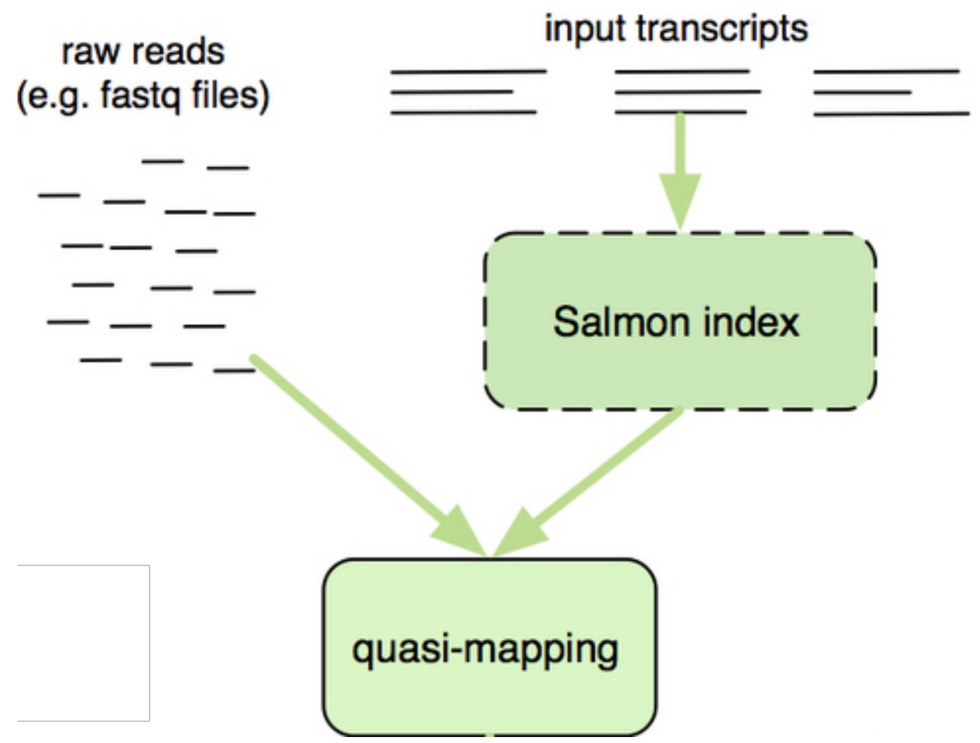
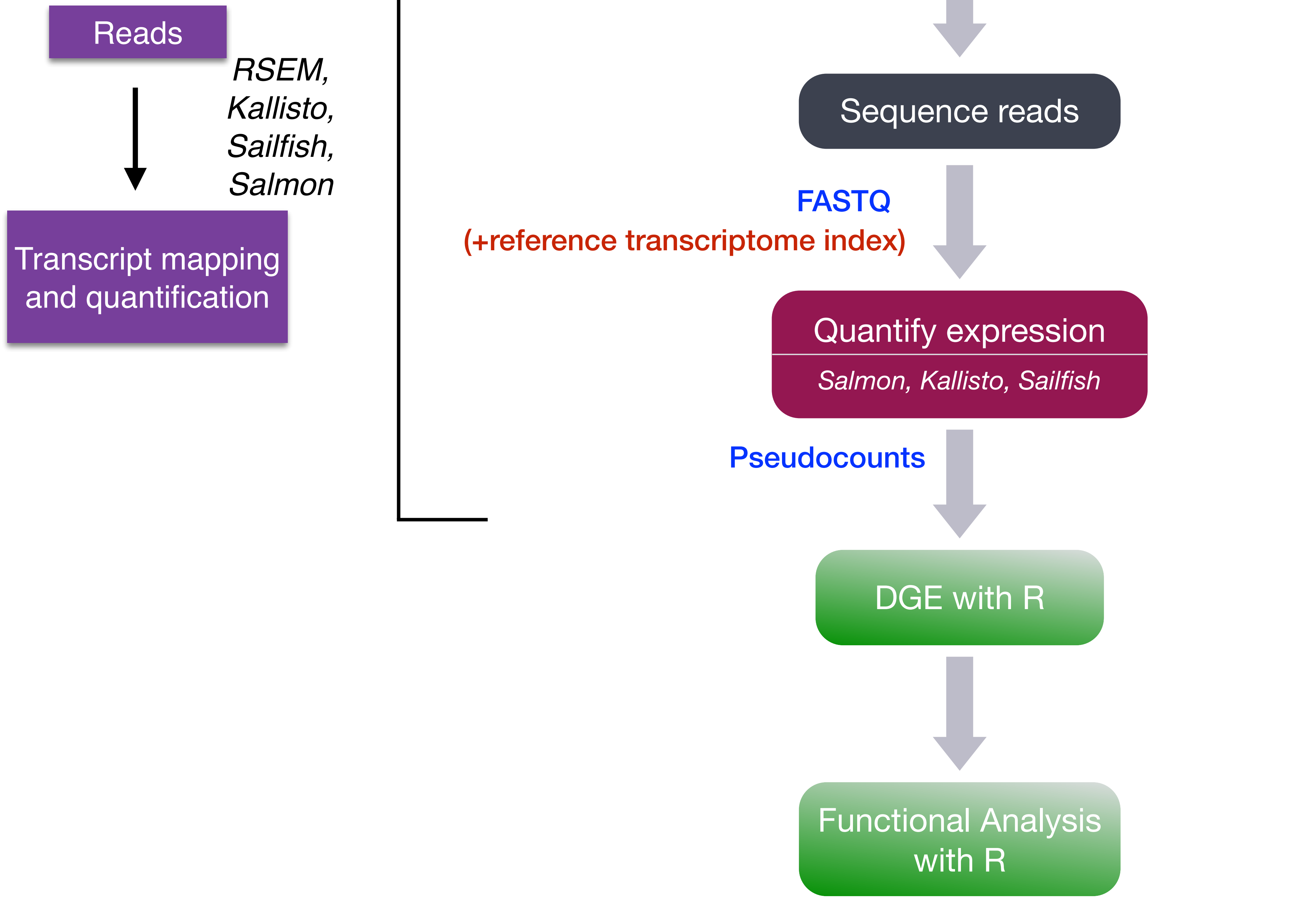


Image source: RNA-seq blog

Note that we don't get the genomic coordinates of where each read is mapping with this approach!

Transcriptome Mapping


Reads
RSEM, Kallisto, Sailfish, Salmon
Transcript mapping and quantification
Biological samples/Library preparation
Sequence reads
FASTQ
(+reference transcriptome index)
Quantify expression
Salmon, Kallisto, Sailfish
Pseudocounts
DGE with R
Functional Analysis with R