



# Intro to Transcriptomics

## Programming for Biologists

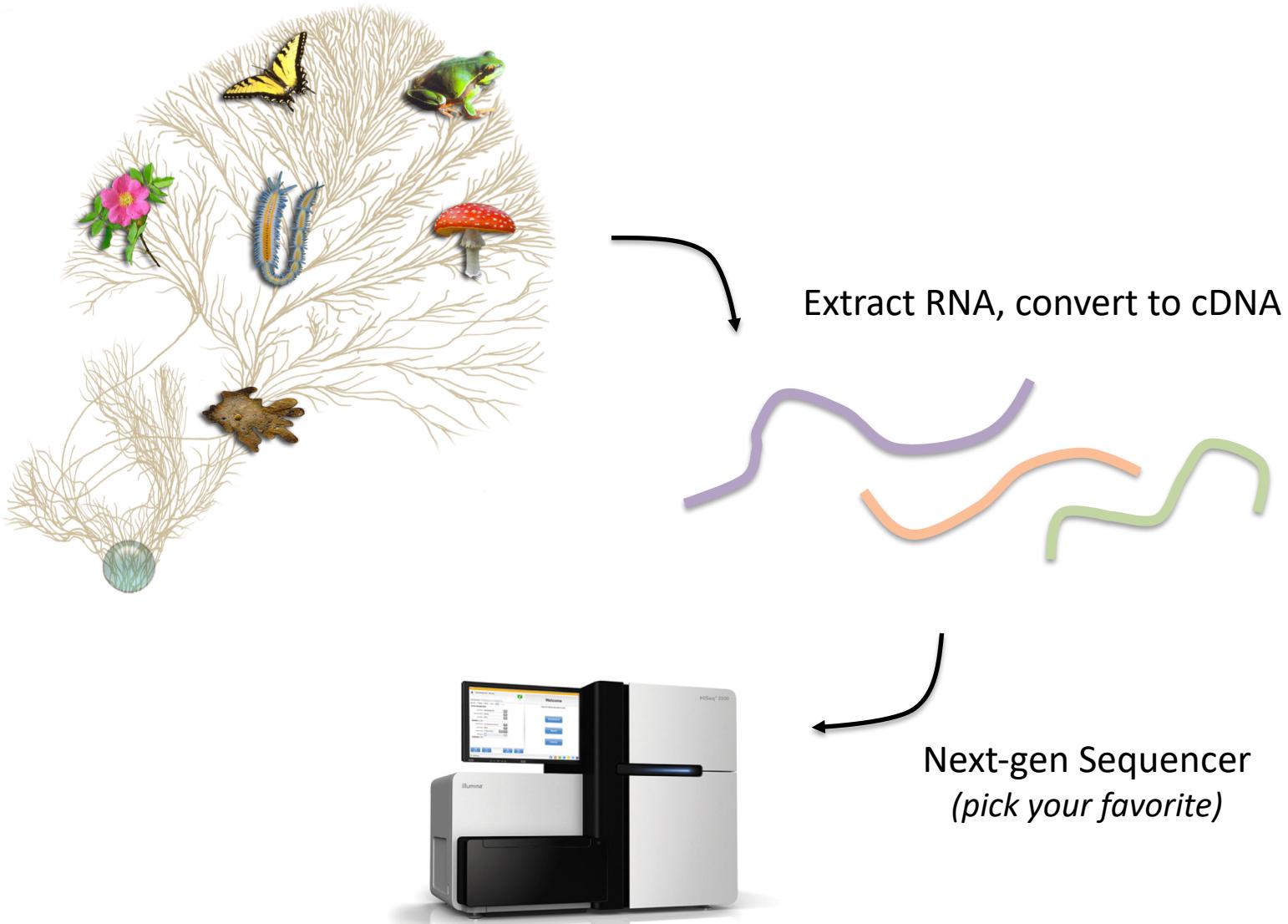
### CSHL 2018

Brian Haas  
Broad Institute

# Transcriptomics Lecture Overview

- Overview of RNA-Seq
- Transcript reconstruction methods
- Expression quantitation
- Differential expression analysis

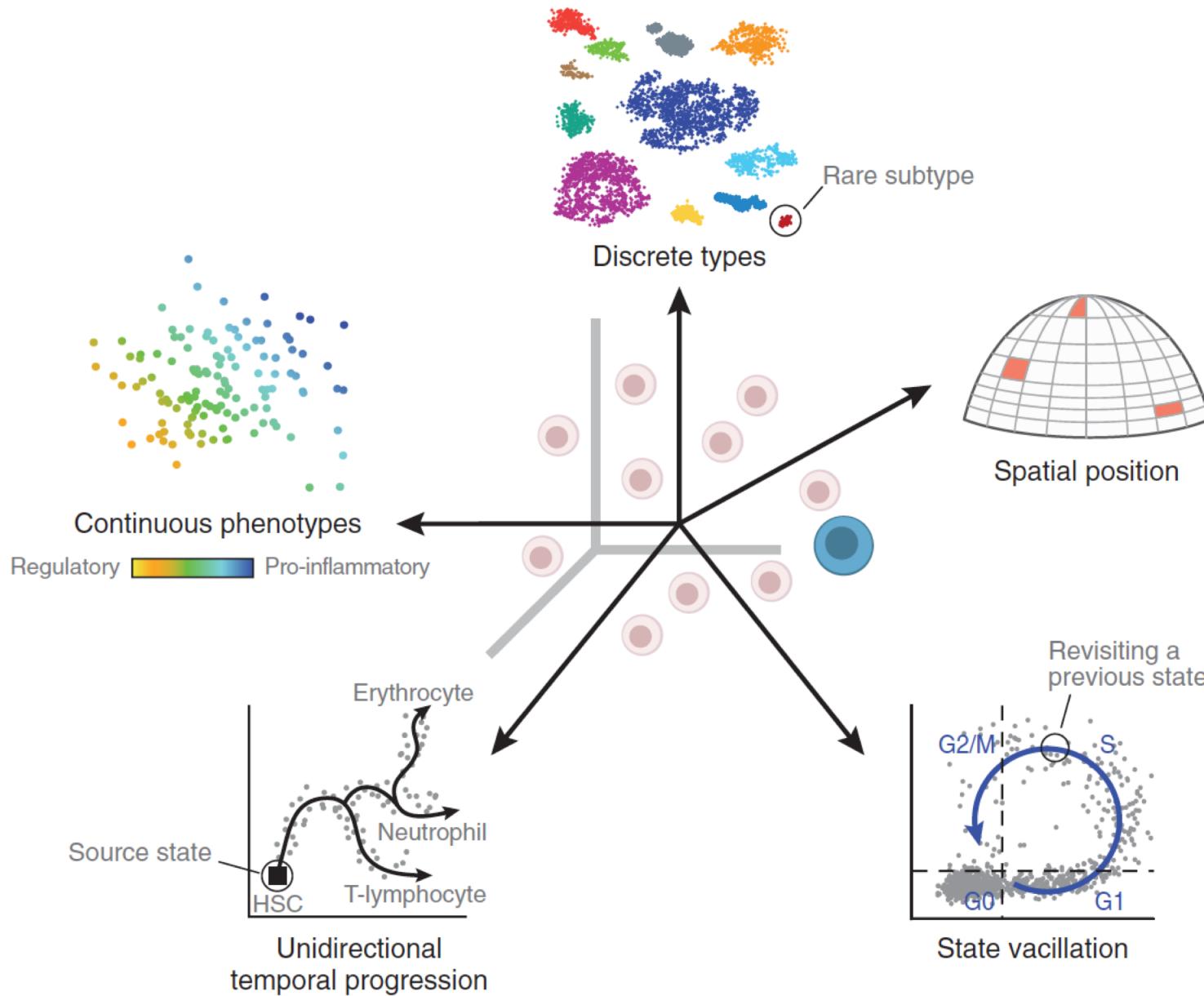
# RNA-Seq Empowers Transcriptome Studies



# RNA-Seq Empowers Many Facets of Biological Investigations

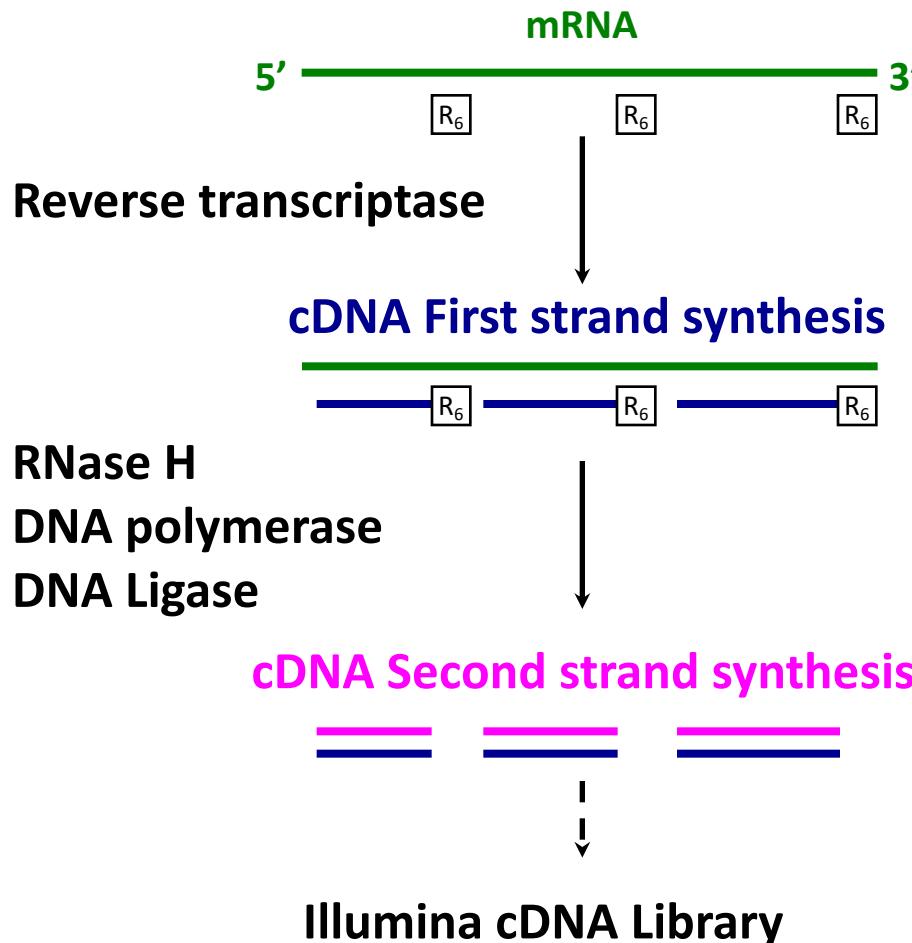
- Transcript identification (ie. which genes active)
- Expression Levels
- Alternative splicing isoforms
- Allelic variants
- Mutations
- Fusion Transcripts
- RNA-editing

# RNA-Seq is Empowering Discovery at Single Cell Resolution



# RNA-Seq: How do we make cDNA?

## Prime with Random Hexamers (R6)



# Generating RNA-Seq: *How to Choose?*

Many different instruments hit the scene in the last decade



Illumina



454



SOLiD



Helicos



Ion Torrent

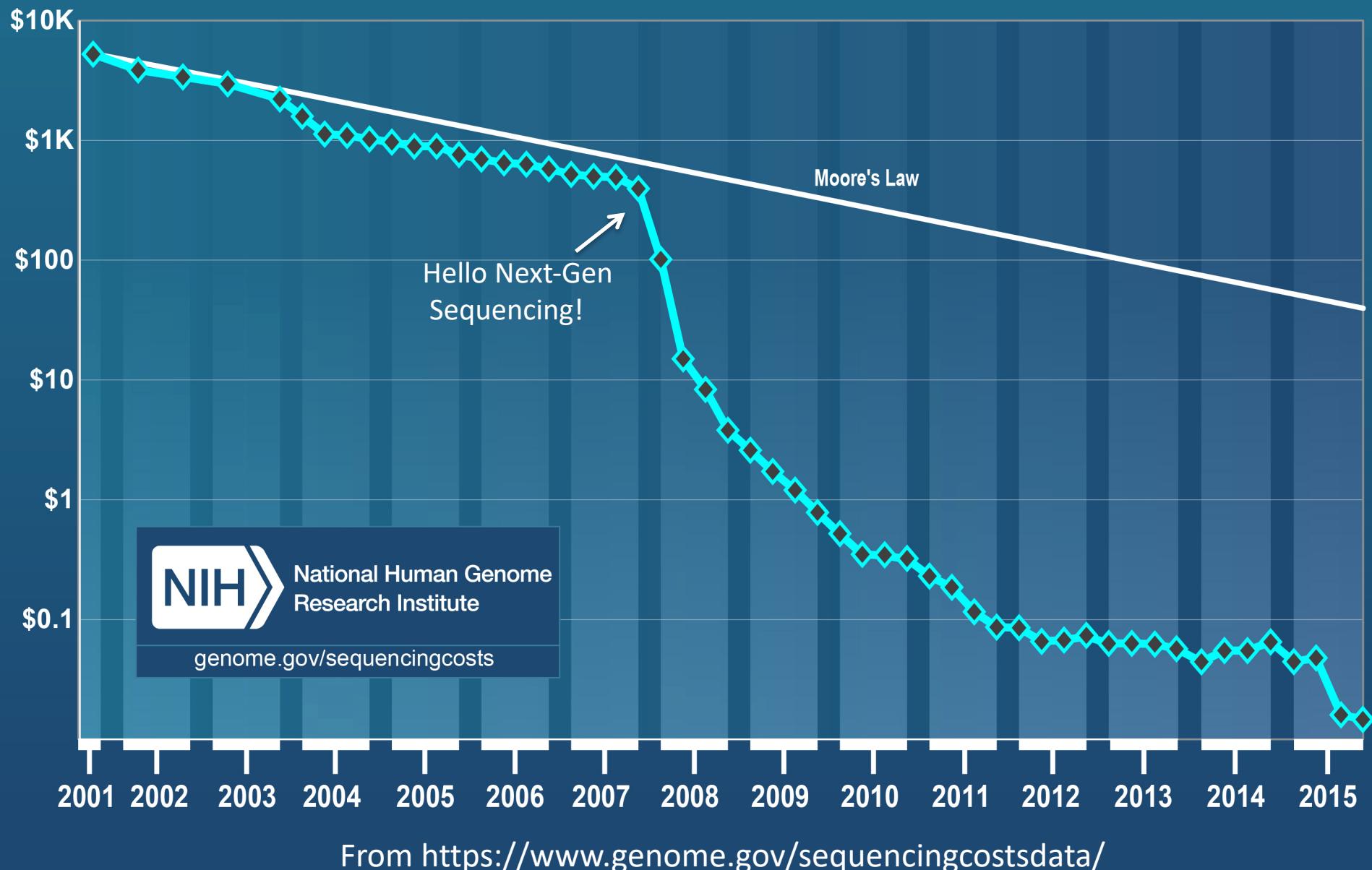


Pacific Biosciences



Oxford Nanopore

# *Cost per Raw Megabase of DNA Sequence*



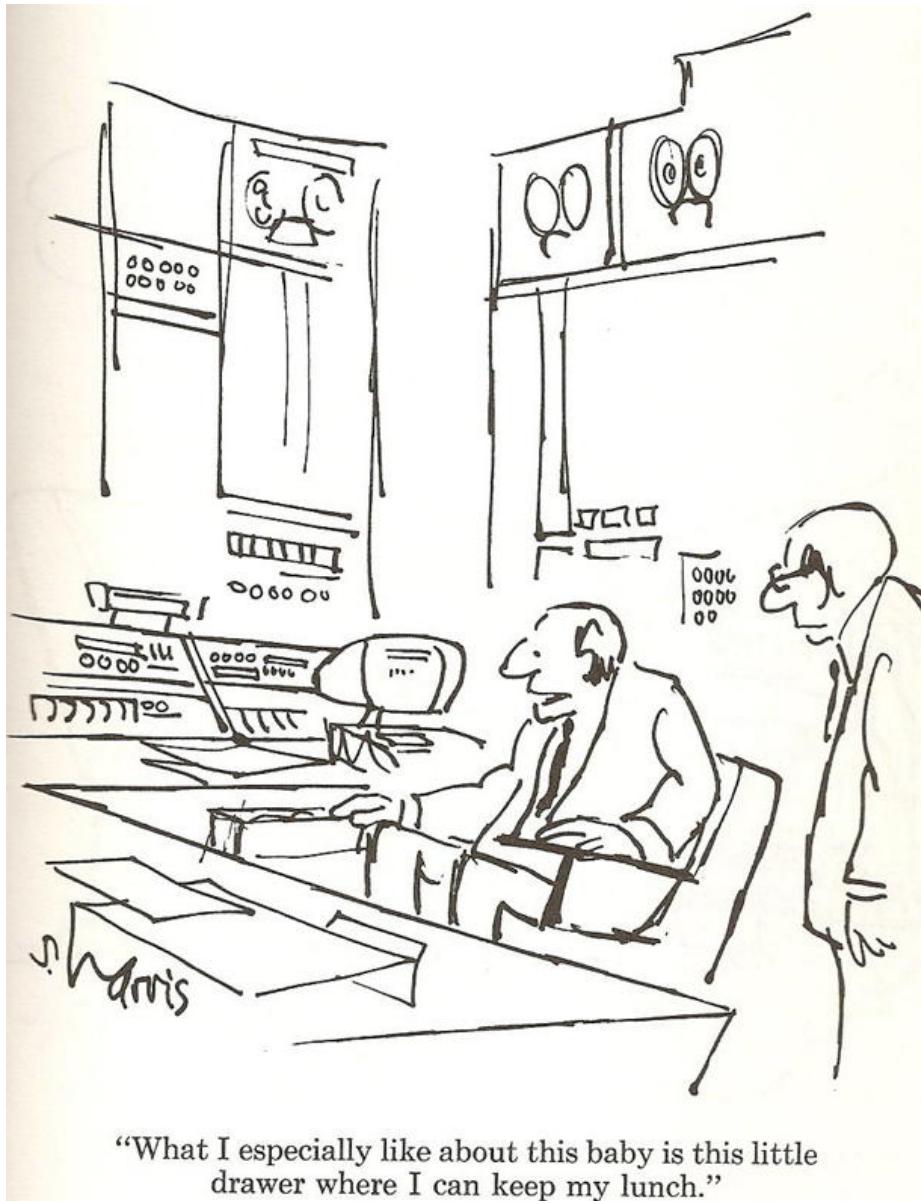
# RNA-Seq: How to Choose?



Illumina



Ion Torrent



Helicos



Oxford Nanopore

# Generating RNA-Seq: *How to Choose?*

Popular choices for RNA-Seq today



Illumina



454



SOLiD



Helicos



Ion Torrent



Pacific Biosciences



Oxford Nanopore

# Generating RNA-Seq: *How to Choose?*

Popular choices for RNA-Seq today

[Current RNA-Seq workhorse]



Illumina



Ion Torrent

[Full-length single molecule sequencing]



Pacific Biosciences

[Newly emerging technology for full-length single molecule sequencing]

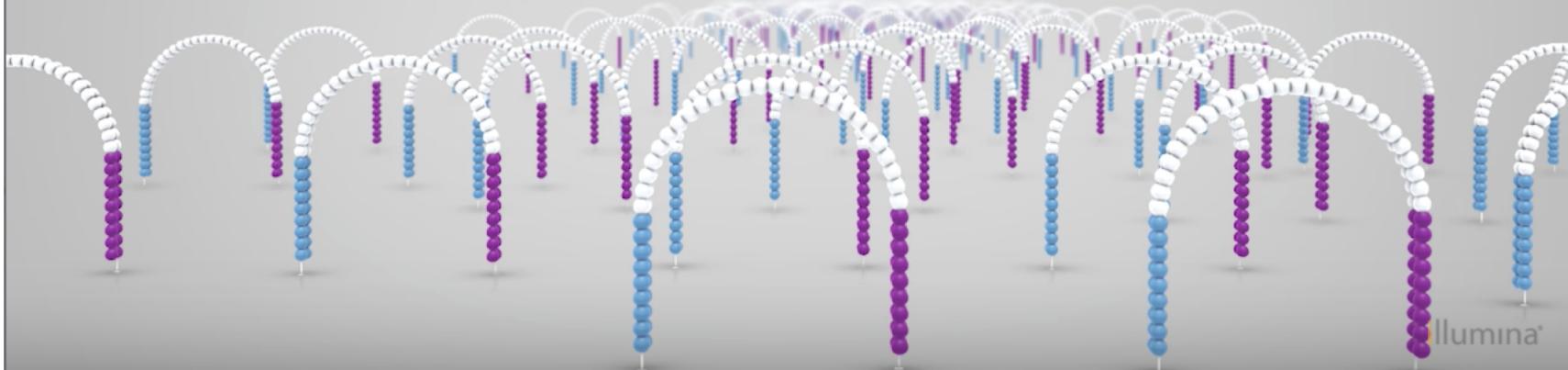


Oxford Nanopore



# Illumina Sequencing by Synthesis

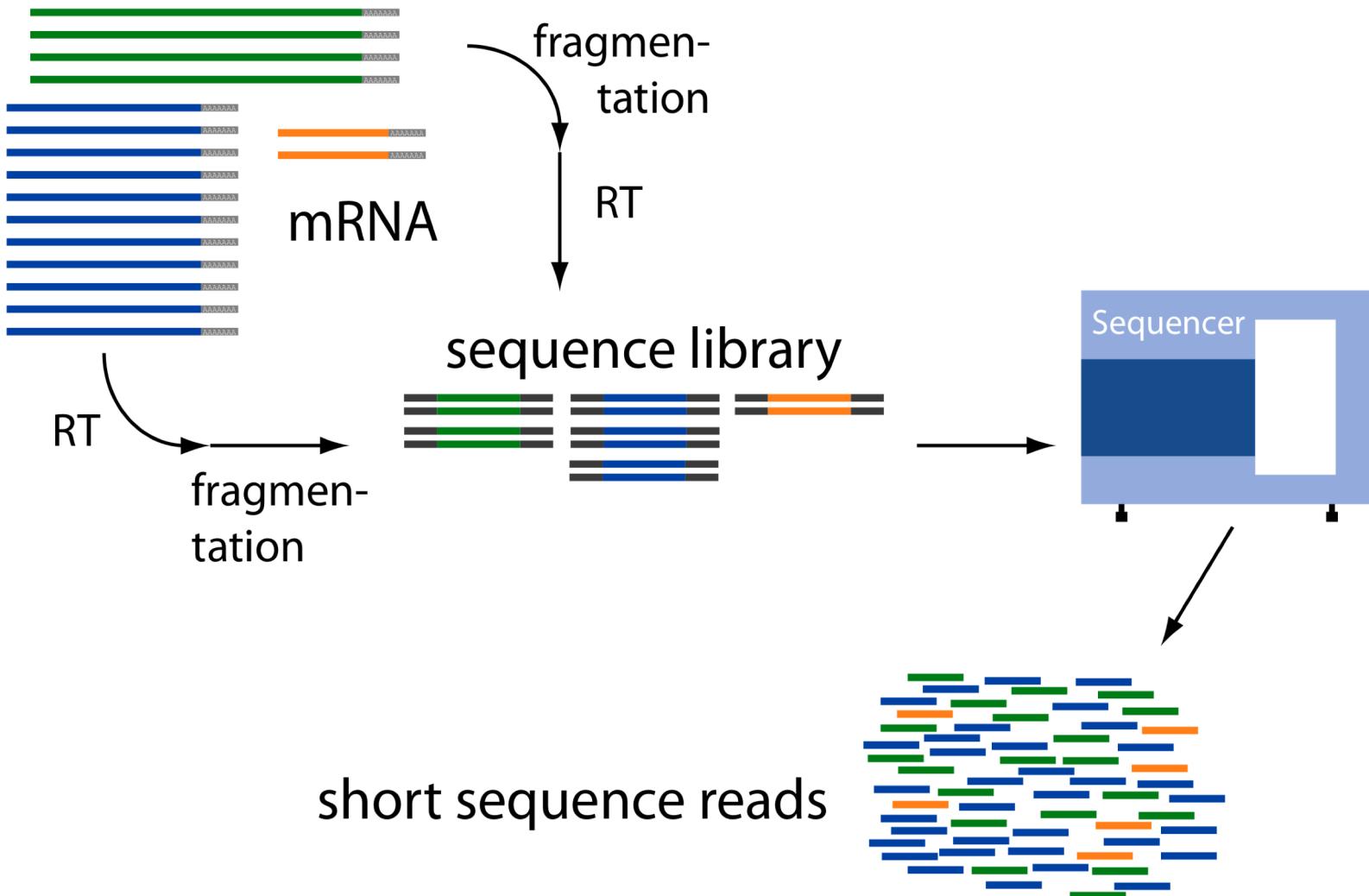
Cluster Generation



2:01 / 5:12



# Overview of RNA-Seq



# Common Data Formats for RNA-Seq

FASTA format:

```
>61DFRAAXX100204:1:100:10494:3070/1  
AAACAAACAGGGCACATTGTCACTCTTGTATTTGAAAAAACACTTCCGGCCAT
```

FASTQ format:

```
@61DFRAAXX100204:1:100:10494:3070/1  
AAACAAACAGGGCACATTGTCACTCTTGTATTTGAAAAAACACTTCCGGCCAT  
+  
ACCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCBC?CCCCCCCC@ @CACCCCCA
```

Read  
Quality values

$$\text{AsciiEncodedQual}(x) = -10 * \log_{10}(\text{Pwrong}(x)) + 33$$



$$\text{AsciiEncodedQual } ('C') = 64$$

$$\text{So, Pwrong('C')} = 10^{(64-33)/(-10)} = 10^{-3.4} = 0.0004$$

# Paired-end Sequences

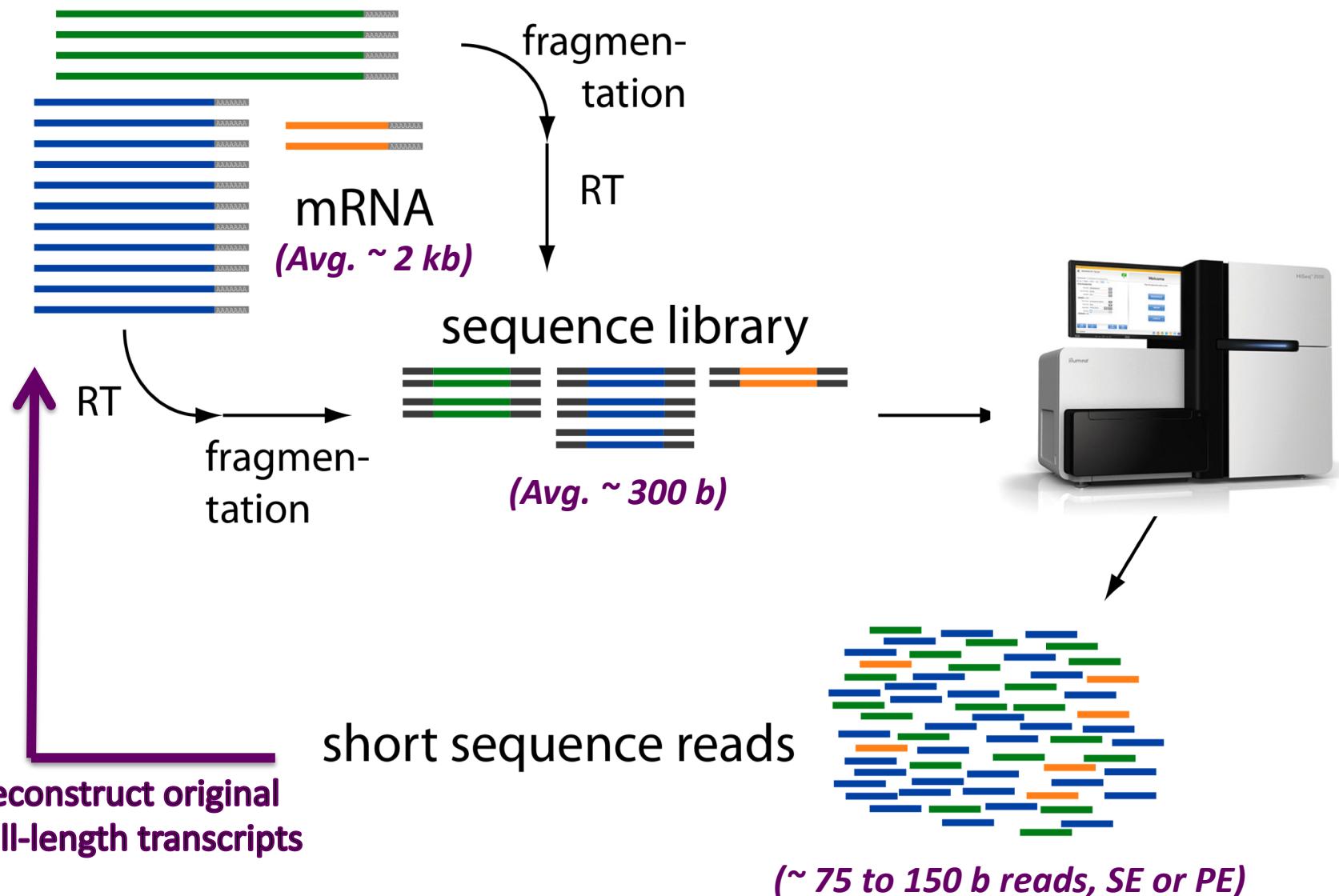


Two FastQ files, read name indicates  
left (/1) or right (/2) read of paired-end

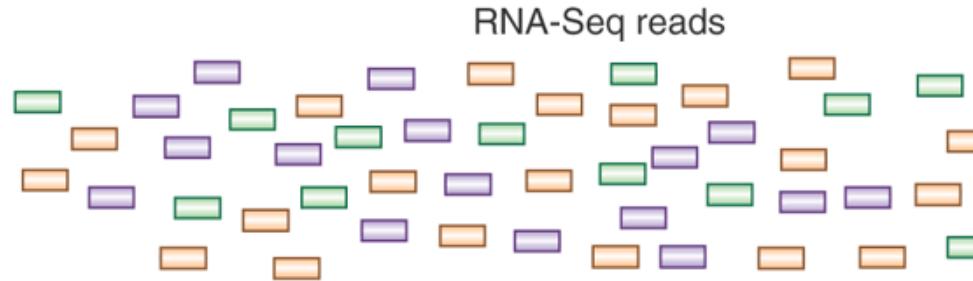
```
@61DFRAAXX100204:1:100:10494:3070/1
AAACAACAGGGCACATTGTCACTCTTGTATTGAAAAACACTTCCGGCCAT
+
ACCCCCCCCCCCCCCCCCCCCCCCCCCCCCCBC?CCCCCCCC@ @CACCCCCA
```

```
@61DFRAAXX100204:1:100:10494:3070/2
CTCAAATGGTTAATTCTCAGGCTGCAAATATTGTTCAAGGATGGAAGAAC
+
C<CCCCCCCCACCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCBCCCC
```

# RNA-Seq Challenge: Transcript Reconstruction



# Transcript Reconstruction from RNA-Seq Reads



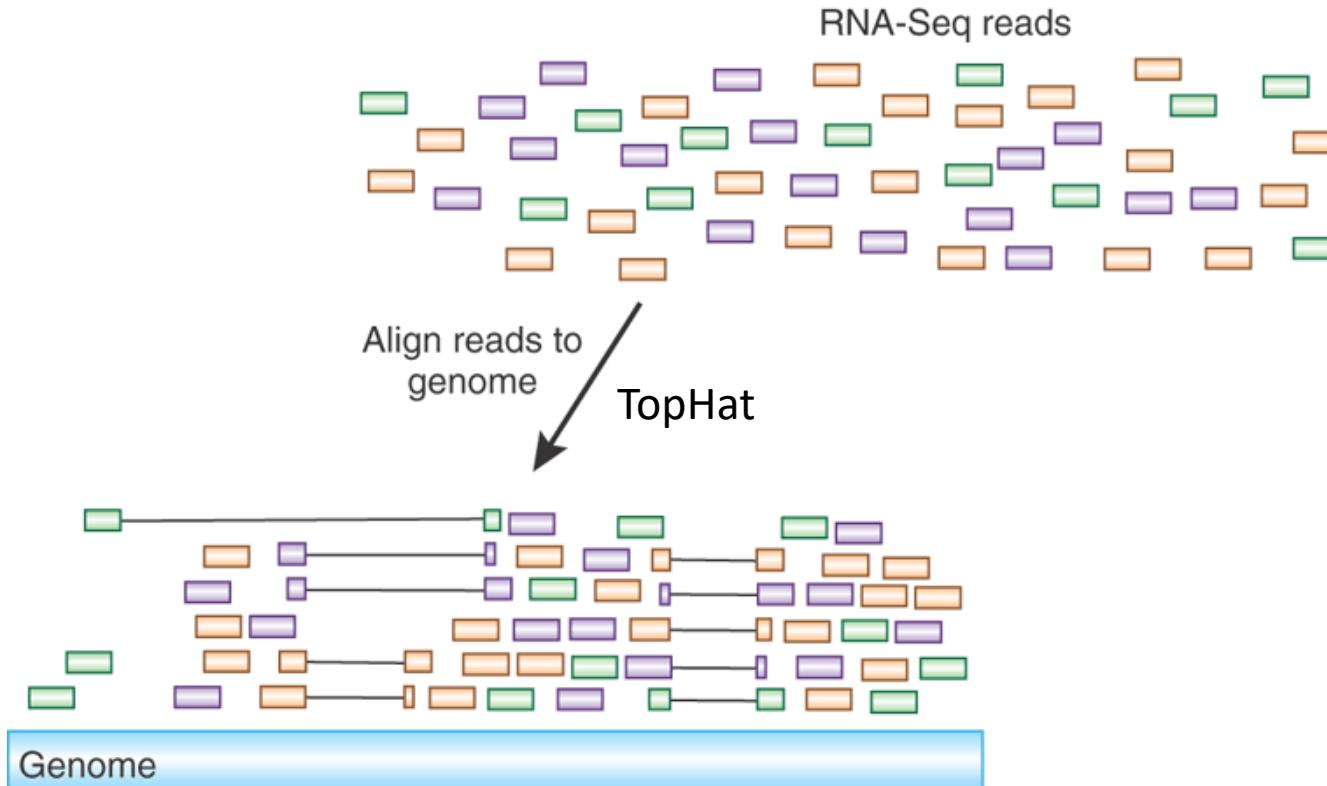
## Advancing RNA-Seq analysis

Brian J Haas & Michael C Zody

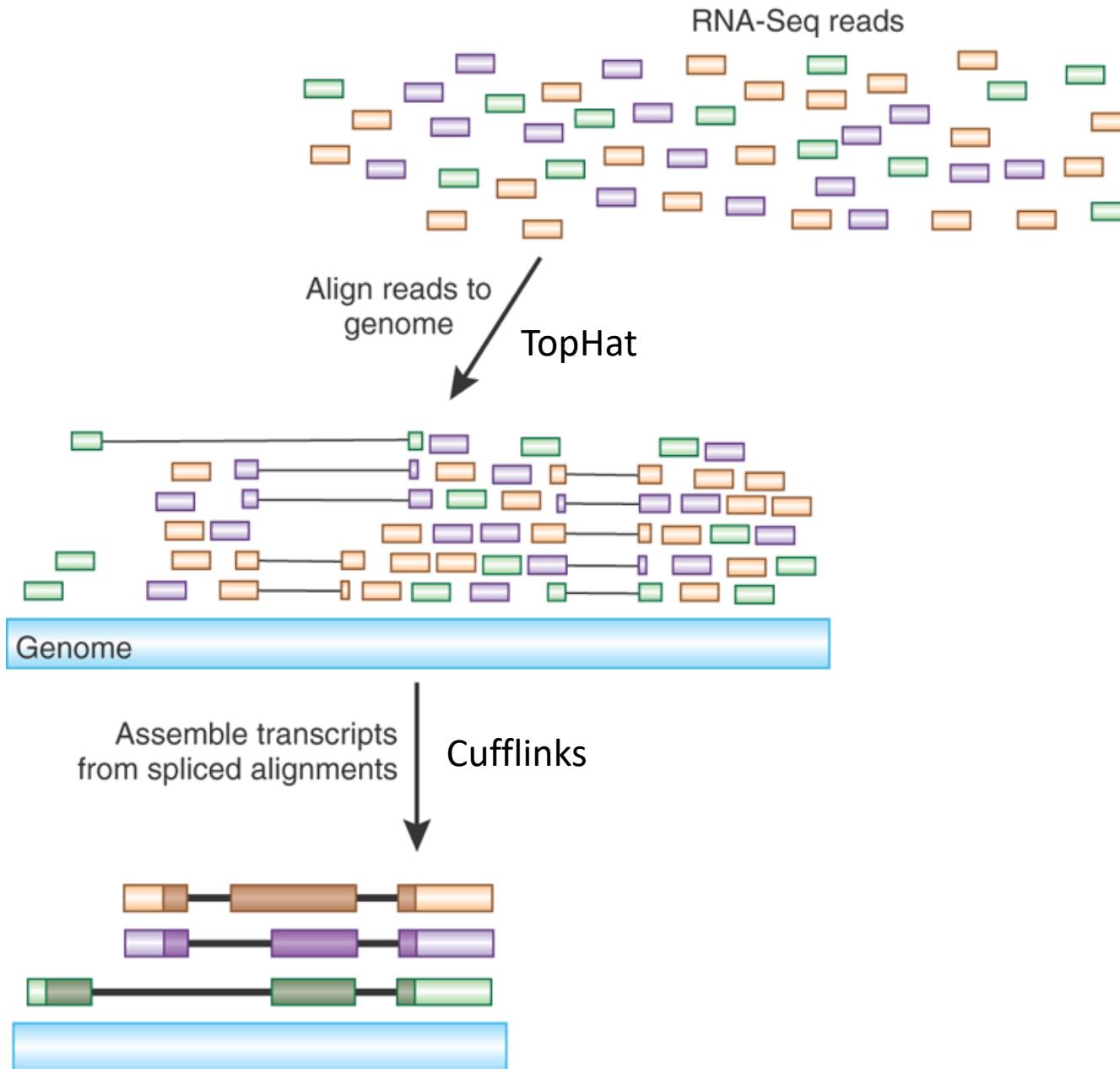
Nature Biotech, 2010

New methods for analyzing RNA-Seq data enable *de novo* reconstruction of the transcriptome.

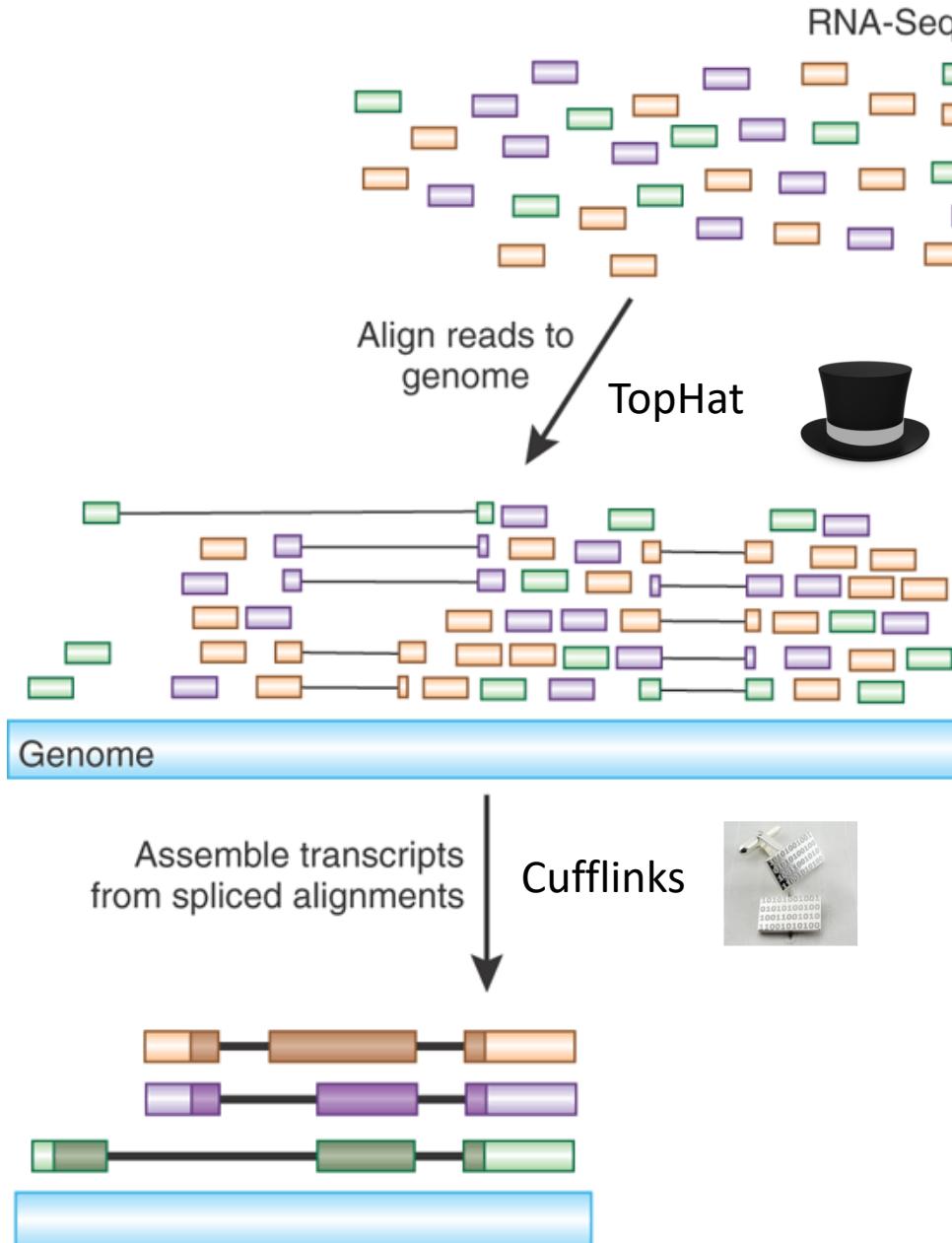
# Transcript Reconstruction from RNA-Seq Reads



# Transcript Reconstruction from RNA-Seq Reads



# Transcript Reconstruction from RNA-Seq Reads



**The Tuxedo Suite:**  
End-to-end **Genome**-based  
RNA-Seq Analysis  
Software Package

*NATURE PROTOCOLS* | PROTOCOL

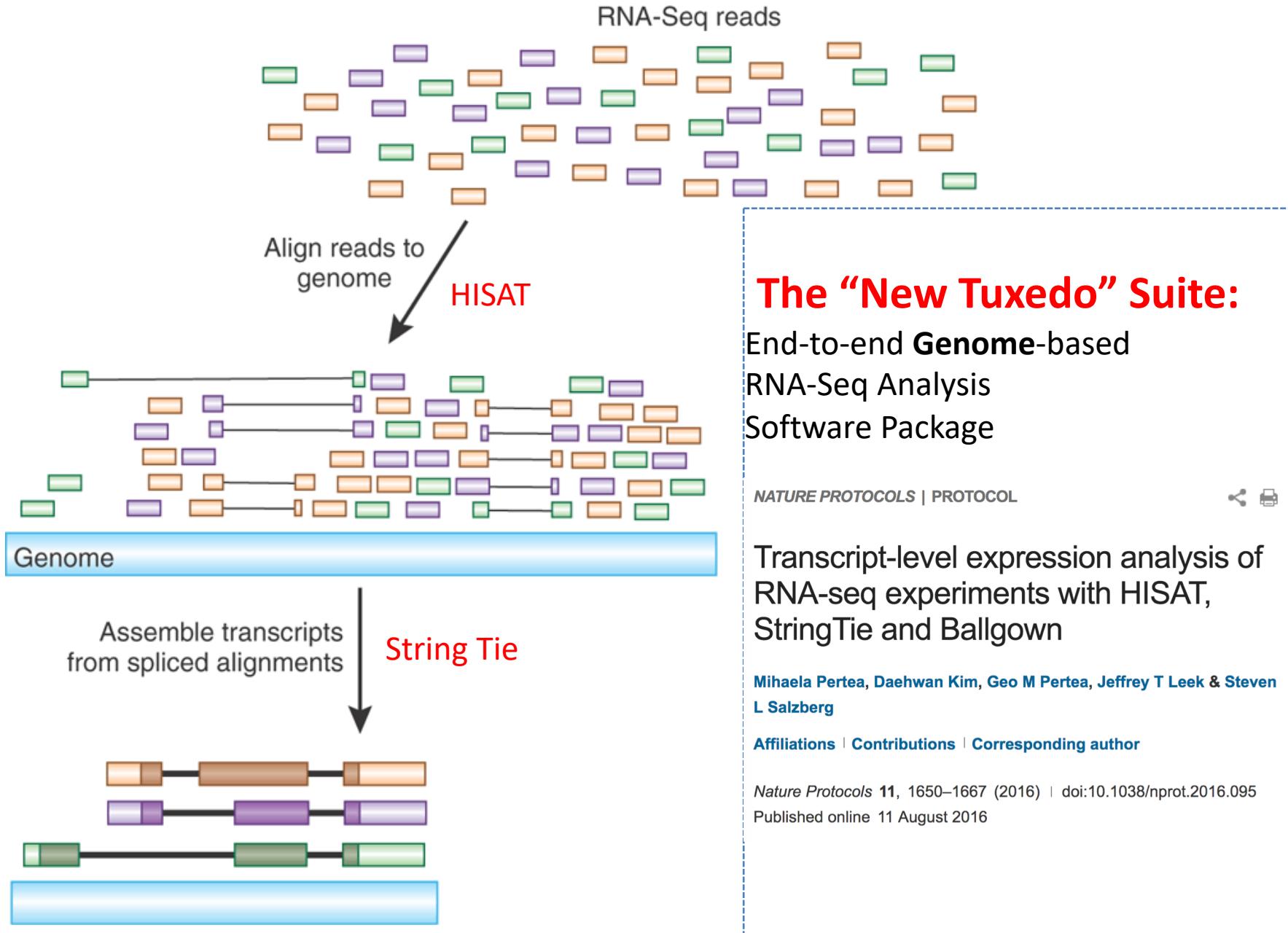
Differential gene and transcript expression analysis of RNA-seq experiments with TopHat and Cufflinks

Cole Trapnell, Adam Roberts, Loyal Goff, Geo Pertea, Daehwan Kim, David R Kelley, Harold Pimentel, Steven L Salzberg, John L Rinn & Lior Pachter

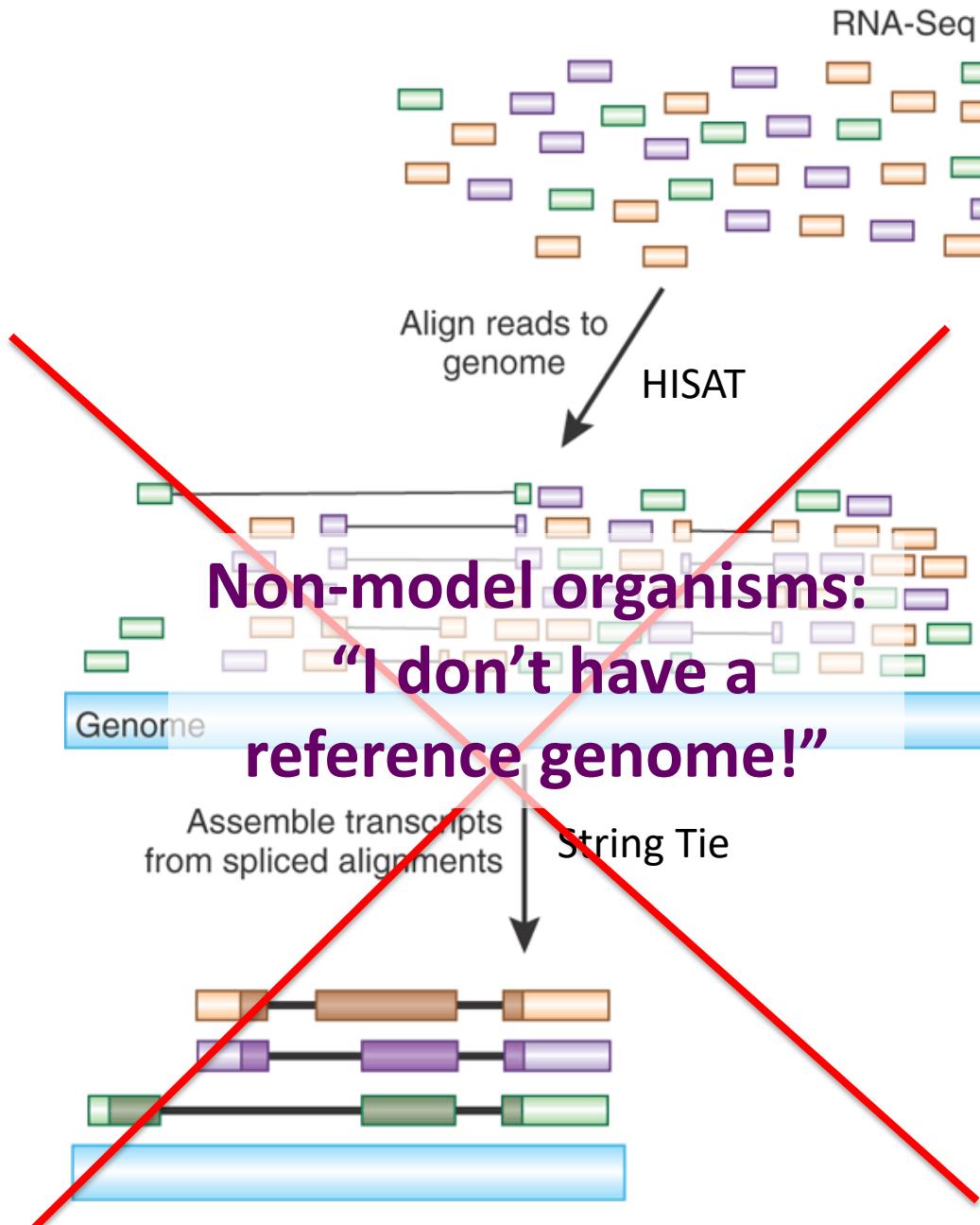
Affiliations | Contributions | Corresponding author

*Nature Protocols* 7, 562–578 (2012) | doi:10.1038/nprot.2012.016  
Published online 01 March 2012

# Transcript Reconstruction from RNA-Seq Reads



# Transcript Reconstruction from RNA-Seq Reads



**The “New Tuxedo” Suite:**  
End-to-end Genome-based  
RNA-Seq Analysis  
Software Package

NATURE PROTOCOLS | PROTOCOL



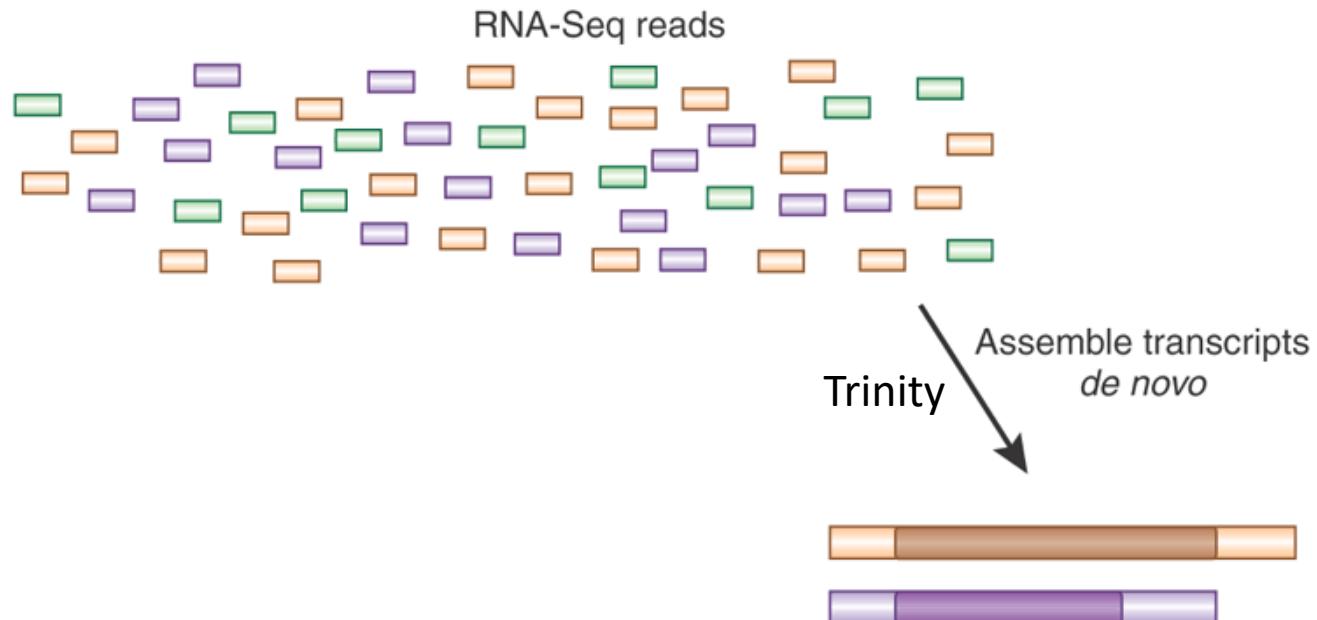
Transcript-level expression analysis of  
RNA-seq experiments with HISAT,  
StringTie and Ballgown

Mihaela Pertea, Daehwan Kim, Geo M Pertea, Jeffrey T Leek & Steven L Salzberg

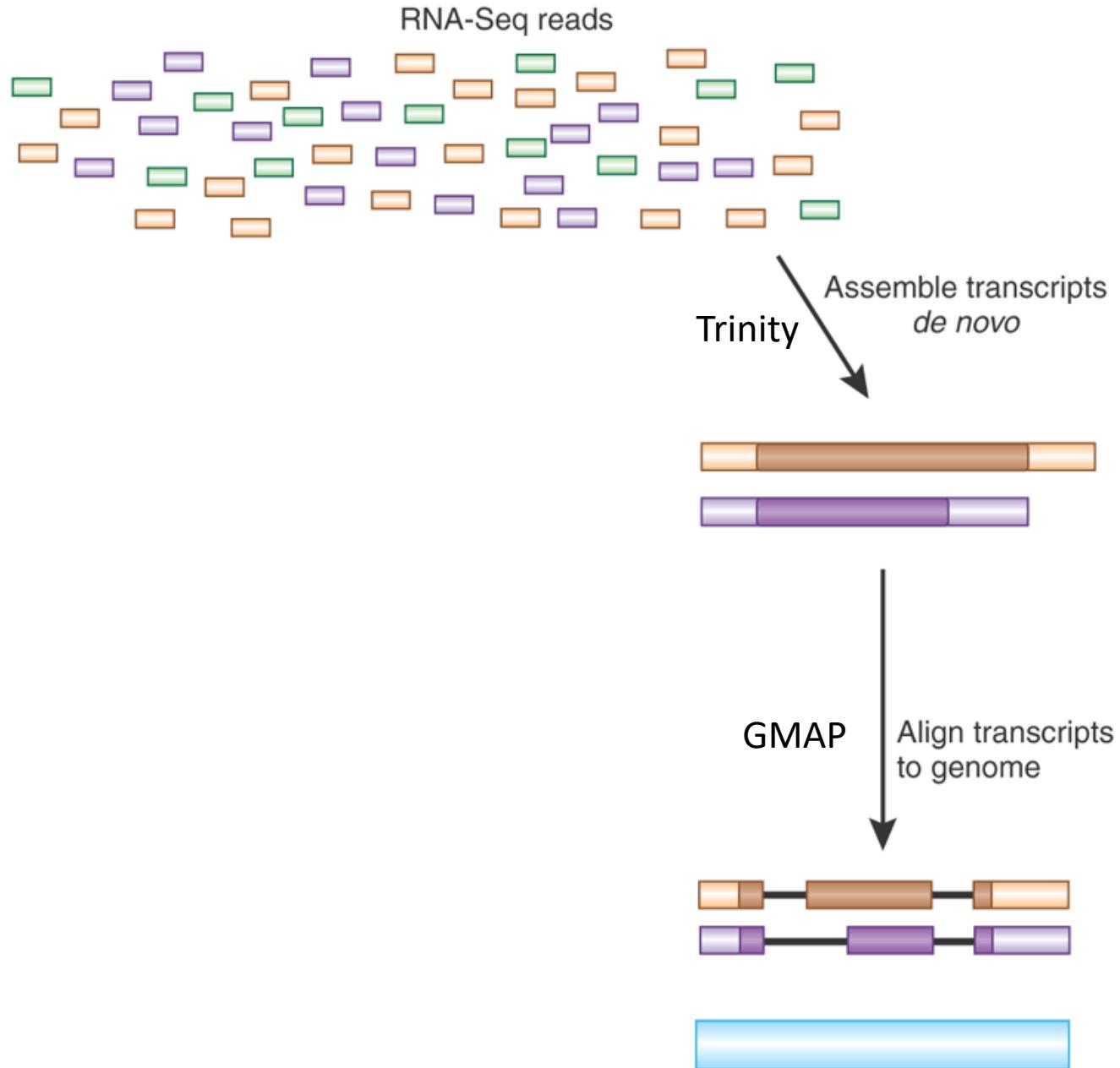
Affiliations | Contributions | Corresponding author

*Nature Protocols* 11, 1650–1667 (2016) | doi:10.1038/nprot.2016.095  
Published online 11 August 2016

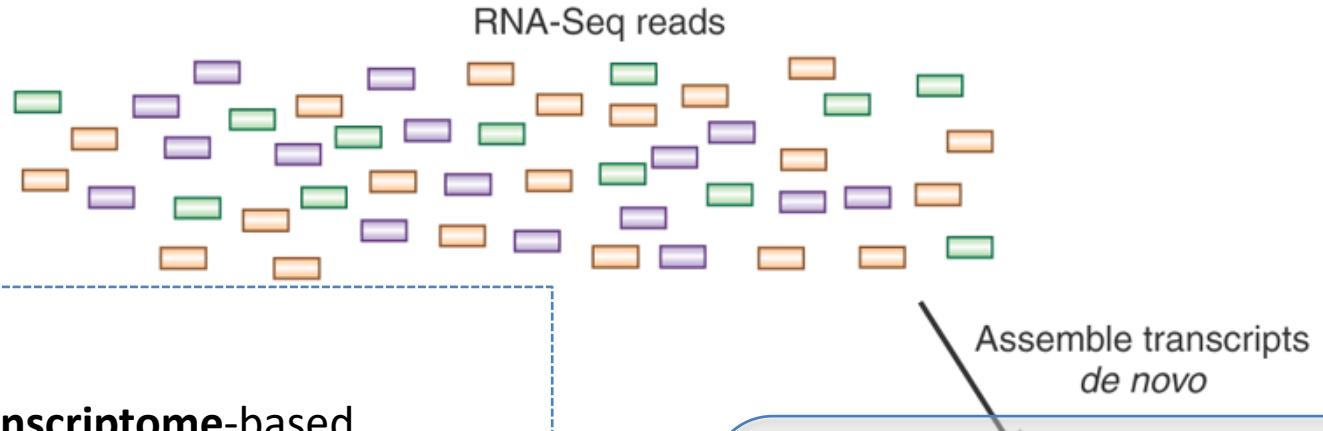
# Transcript Reconstruction from RNA-Seq Reads



# Transcript Reconstruction from RNA-Seq Reads



# Transcript Reconstruction from RNA-Seq Reads



End-to-end Transcriptome-based  
RNA-Seq Analysis  
Software Package

NATURE PROTOCOLS | PROTOCOL

*De novo* transcript sequence reconstruction from RNA-seq using the Trinity platform for reference generation and analysis

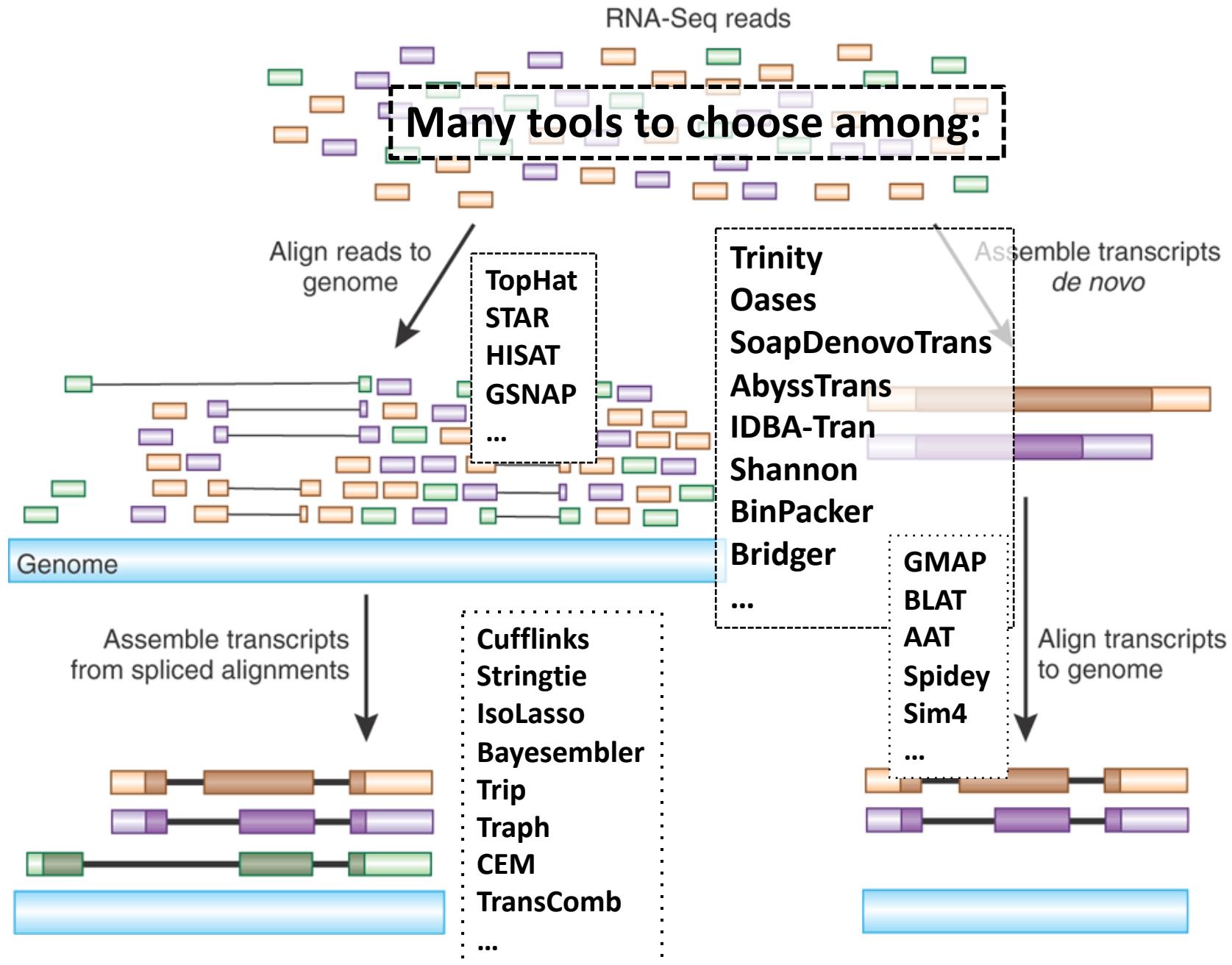
Brian J Haas, Alexie Papanicolaou, Moran Yassour, Manfred Grabherr, Philip D Blood, Joshua Bowden, Matthew Brian Couger, David Eccles, Bo Li, Matthias Lieber, Matthew D MacManes, Michael Ott, Joshua Orvis, Nathalie Pochet, Francesco Strozzi, Nathan Weeks, Rick Westerman, Thomas William, Colin N Dewey, Robert Henschel, Richard D LeDuc, Nir Friedman & Aviv Regev

[Affiliations](#) | [Contributions](#) | [Corresponding authors](#)

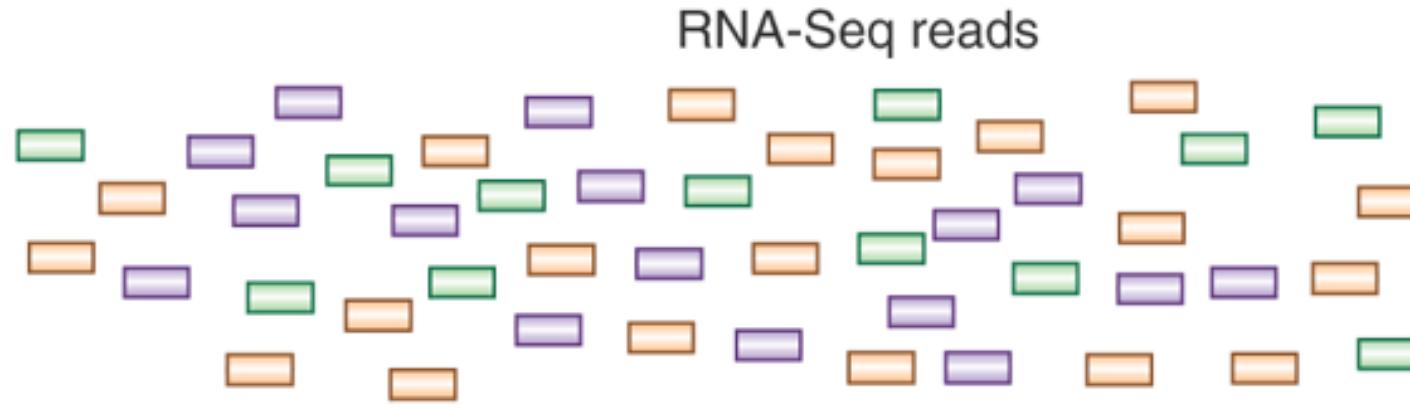
*Nature Protocols* 8, 1494–1512 (2013) | doi:10.1038/nprot.2013.084

Published online 11 July 2013

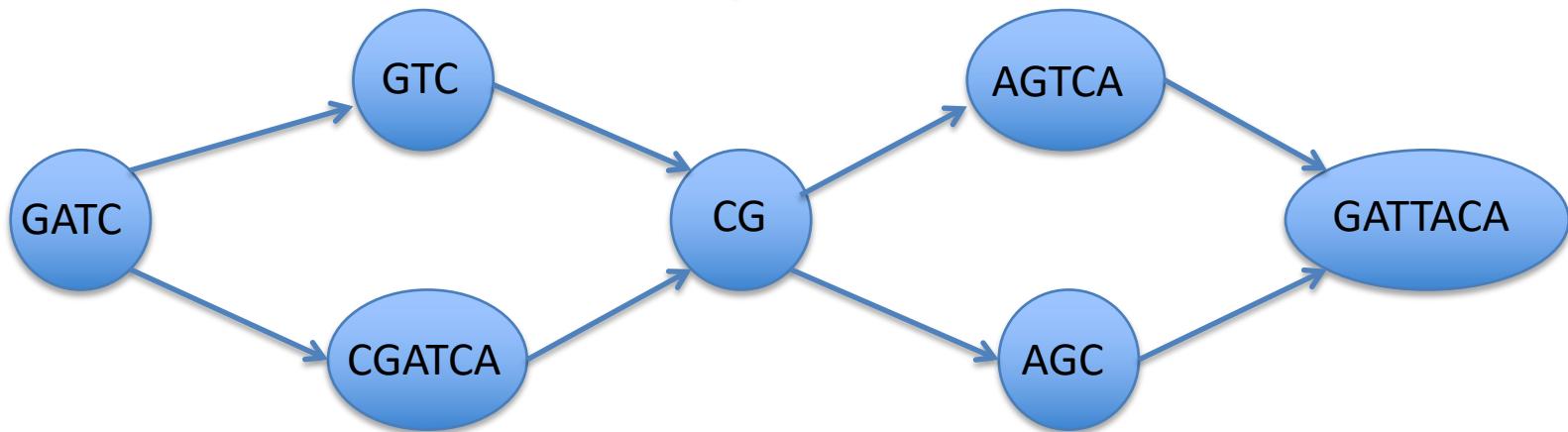
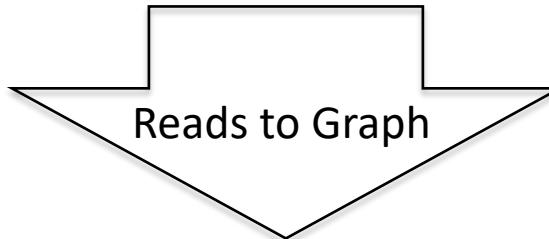
# Transcript Reconstruction from RNA-Seq Reads



# Graph Data Structures Commonly Used For Assembly

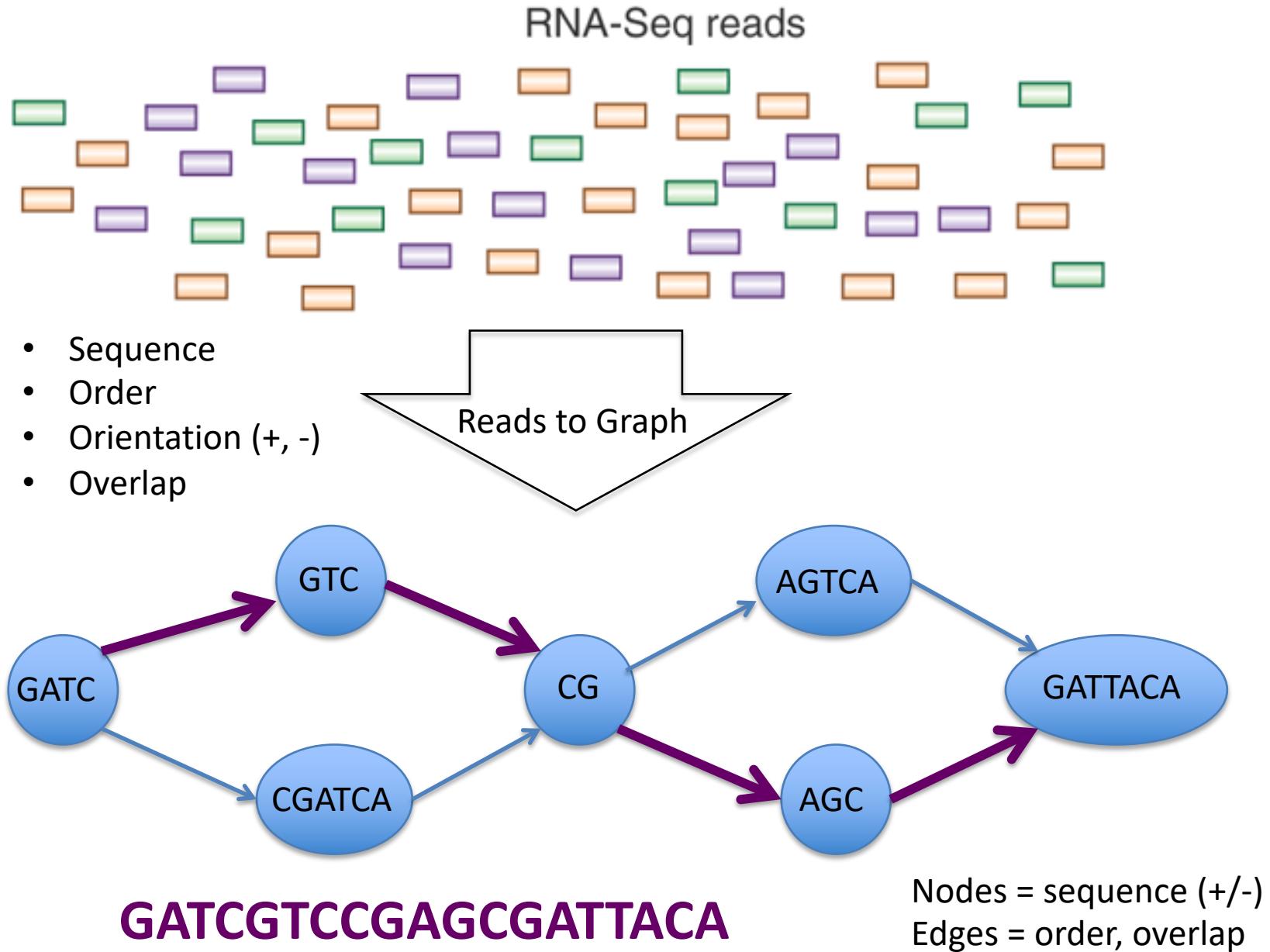


- Sequence
- Order
- Orientation (+, -)
- Overlap



Nodes = sequence (+/-)  
Edges = order, overlap

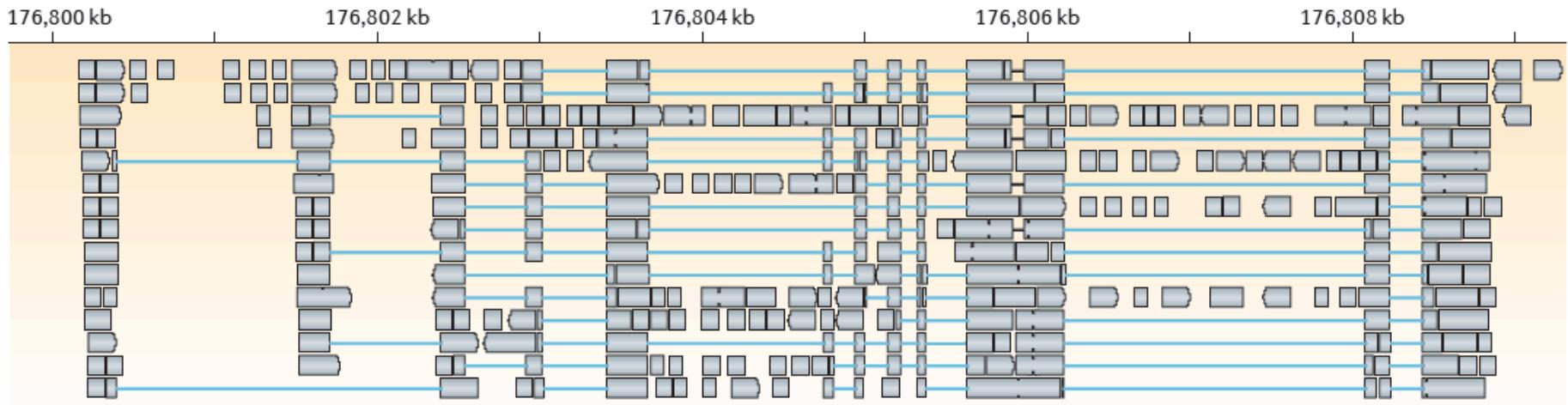
# Graph Data Structures Commonly Used For Assembly



The General Approach to  
*De novo* RNA-Seq Assembly  
Using De Bruijn Graphs

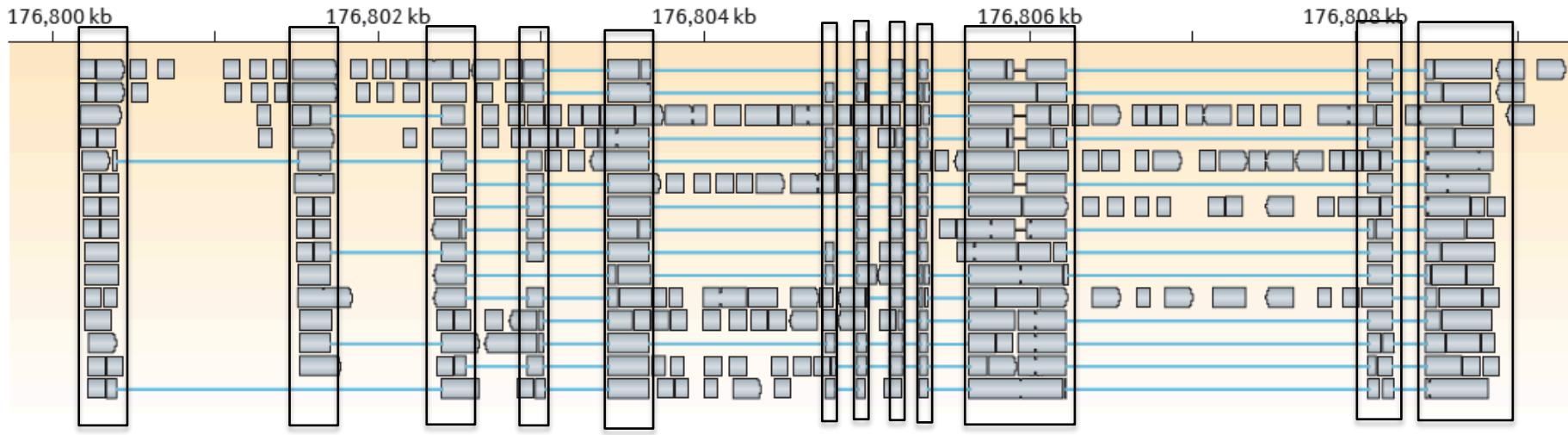
# Genome-Guided Transcript Reconstruction

## Splice-align reads to the genome



# Genome-Guided Transcript Reconstruction

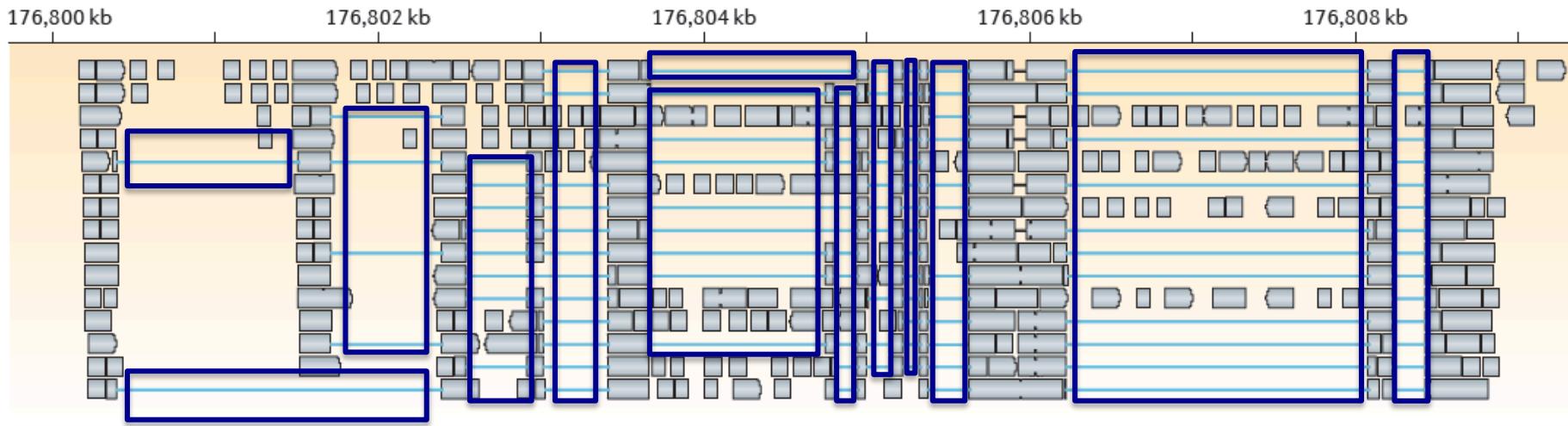
Splice-align reads to the genome



Alignment segment piles => exon regions

# Genome-Guided Transcript Reconstruction

## Splice-align reads to the genome



Large alignment gaps => introns

# Genome-Guided Transcript Reconstruction

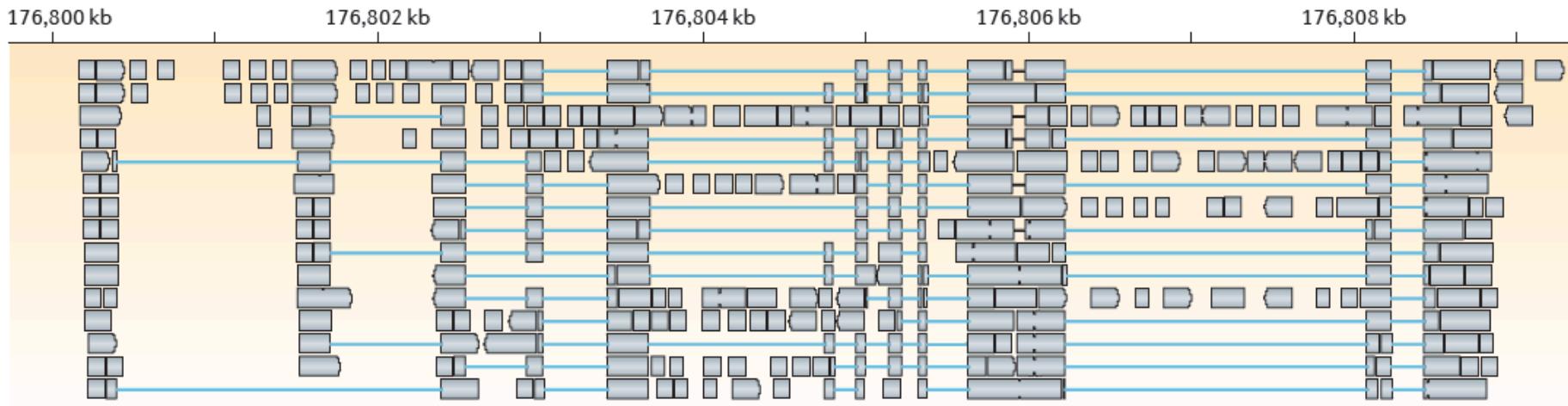
## Splice-align reads to the genome



Overlapping but different introns = evidence of alternative splicing

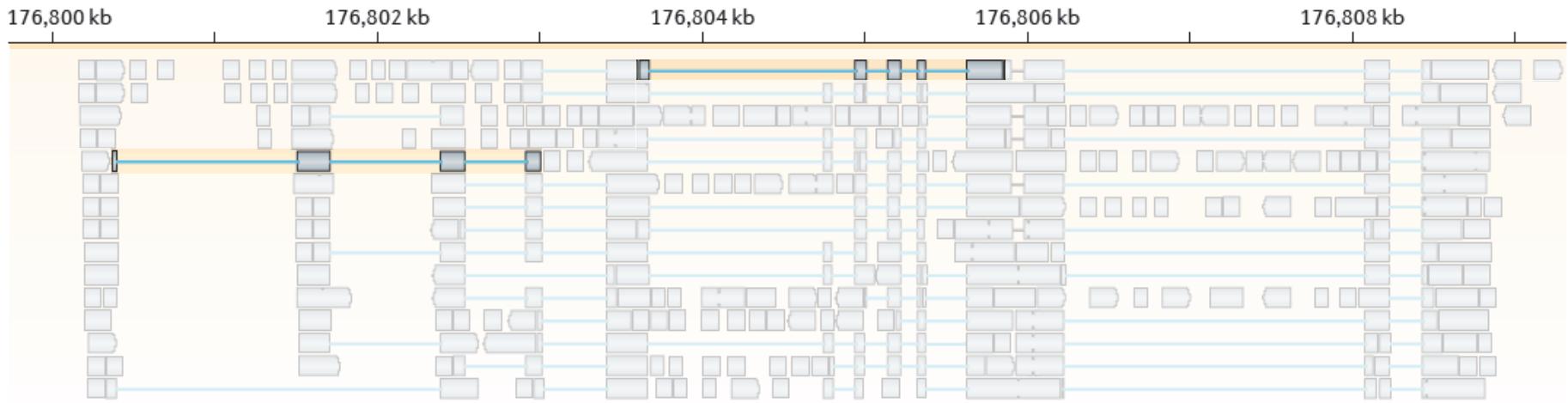
# Genome-Guided Transcript Reconstruction

## Splice-align reads to the genome



# Genome-Guided Transcript Reconstruction

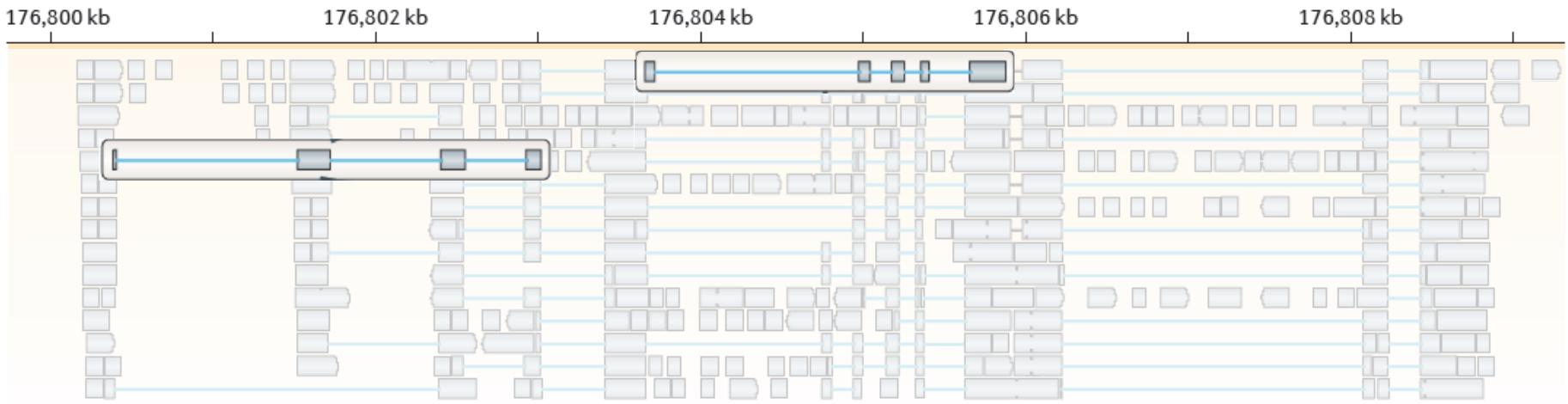
## Splice-align reads to the genome



Individual reads can yield multiple exon and intron segments (splice patterns)

# Genome-Guided Transcript Reconstruction

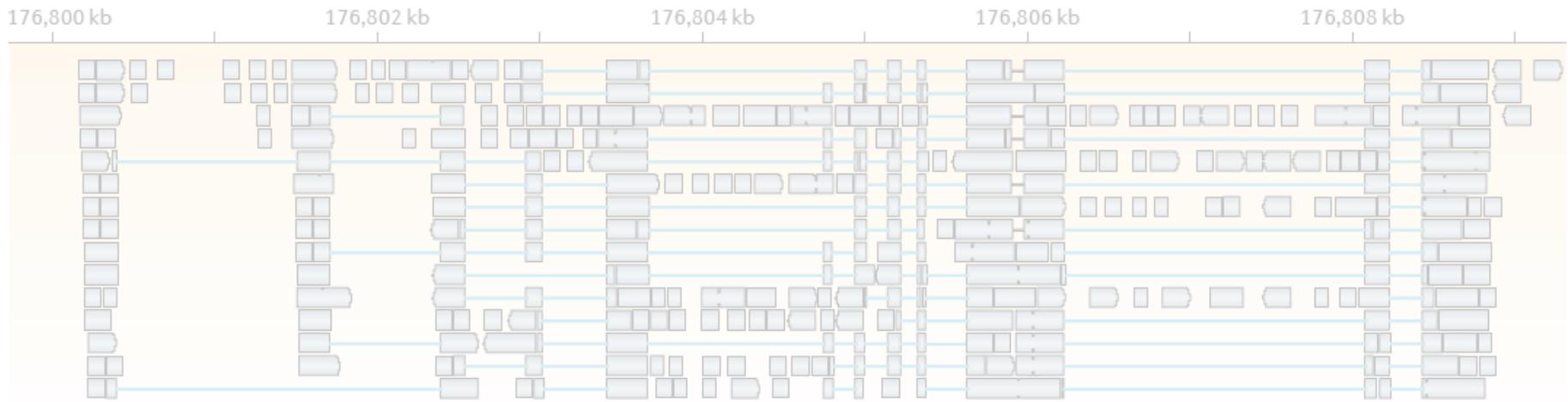
Splice-align reads to the genome



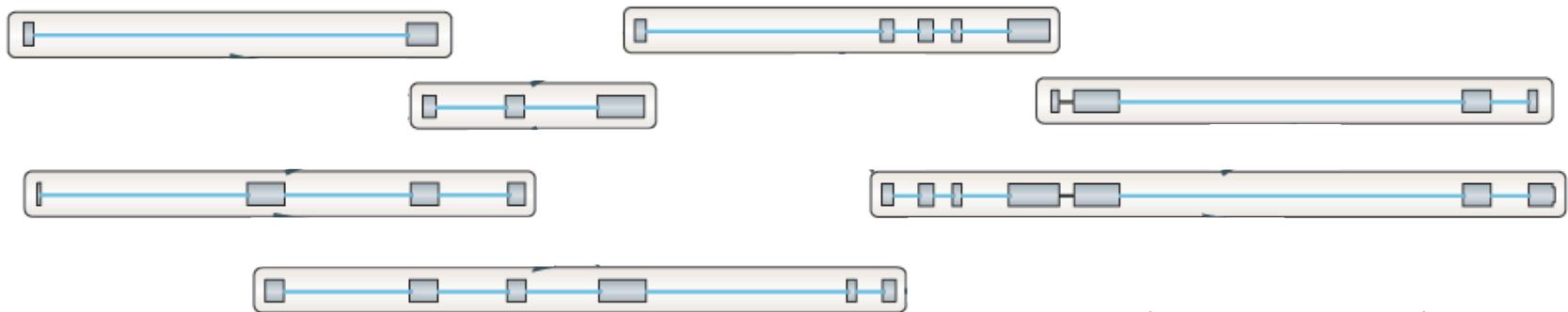
Nodes = unique splice patterns

# Genome-Guided Transcript Reconstruction

Splice-align reads to the genome

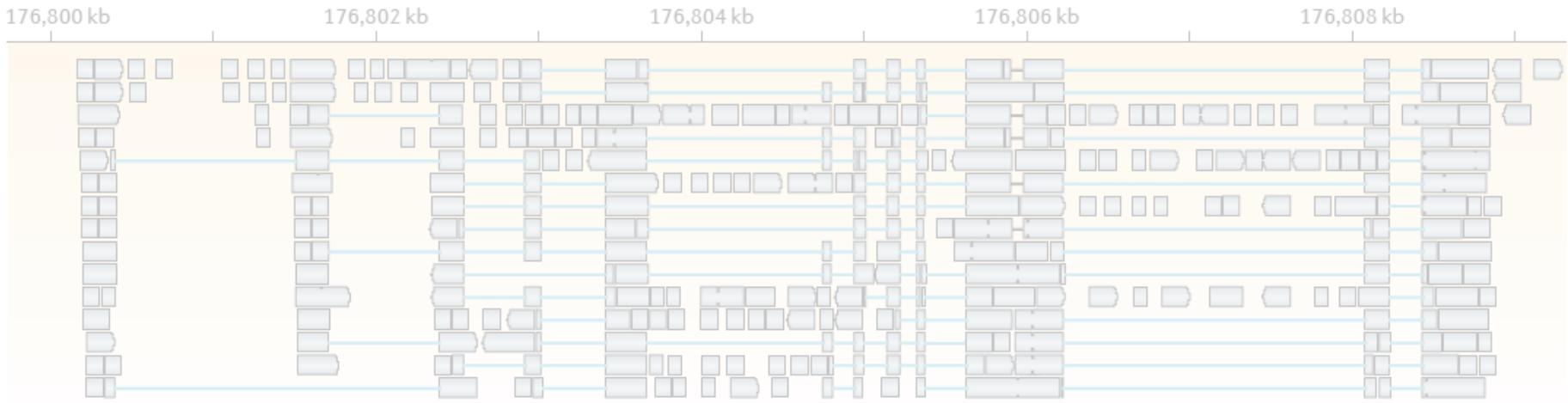


Construct graph from unique splice patterns of aligned reads.

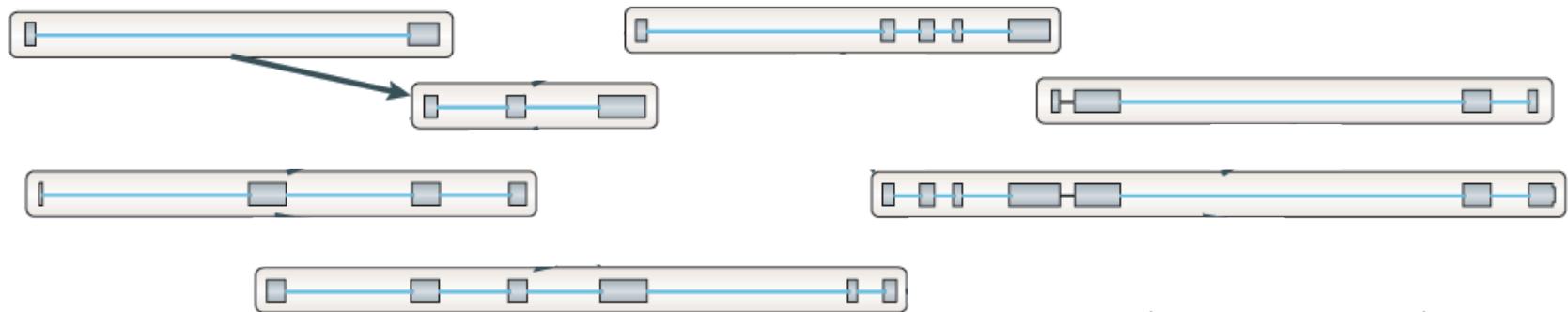


# Genome-Guided Transcript Reconstruction

Splice-align reads to the genome



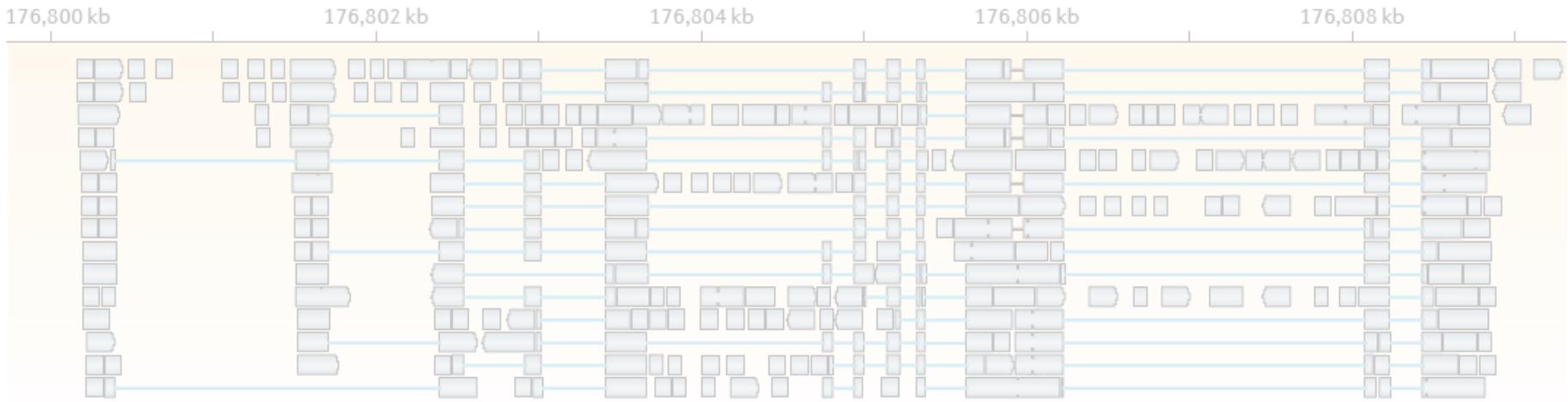
Construct graph from unique splice patterns of aligned reads.



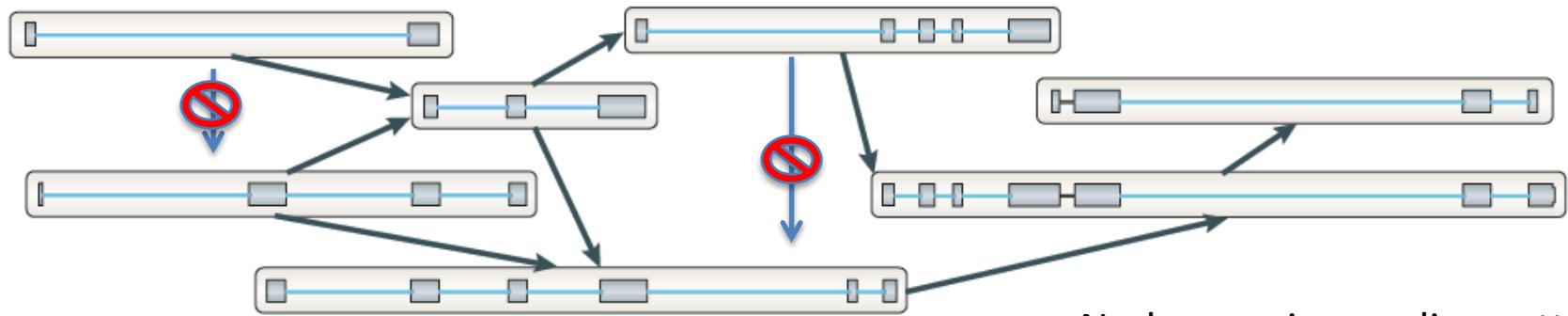
Nodes = unique splice patterns  
Edges = compatible patterns

# Genome-Guided Transcript Reconstruction

Splice-align reads to the genome

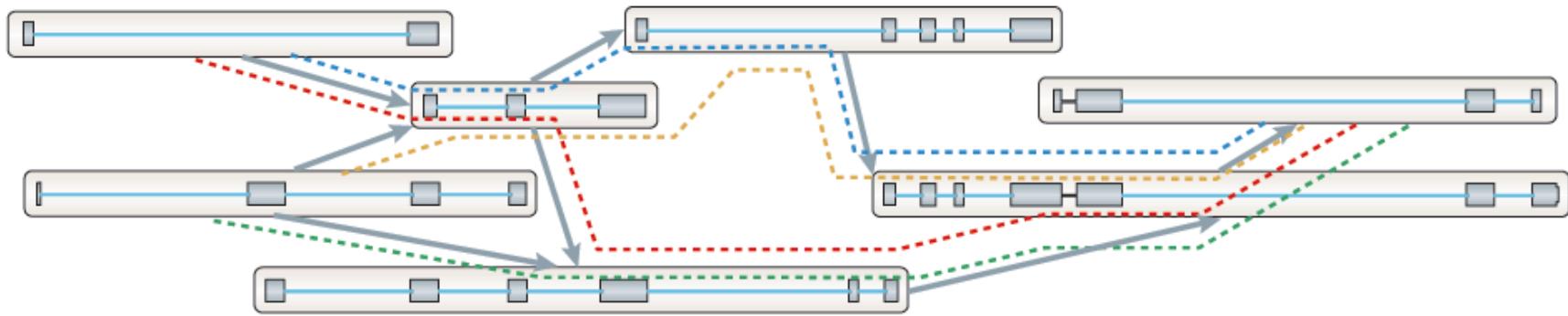


Construct graph from unique splice patterns of aligned reads.



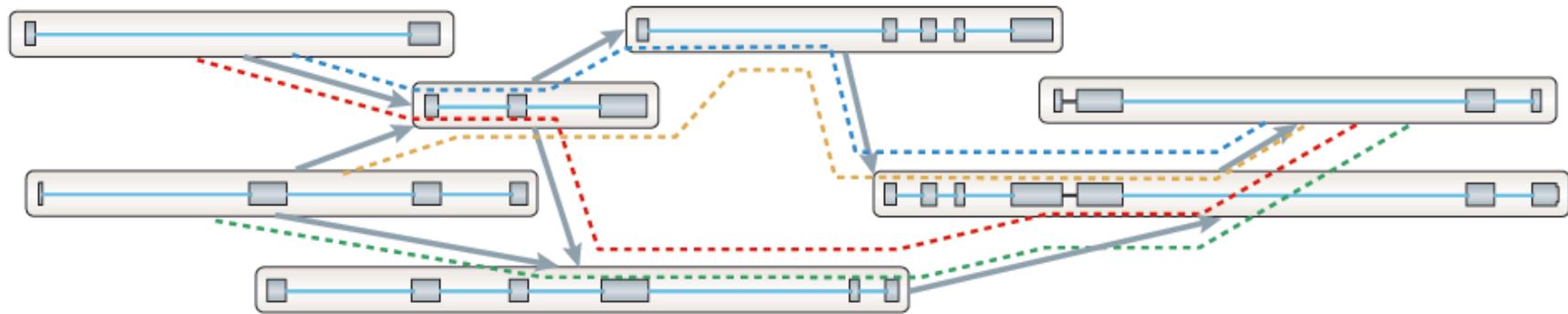
# Genome-Guided Transcript Reconstruction

Traverse paths through the graph to assemble transcript isoforms

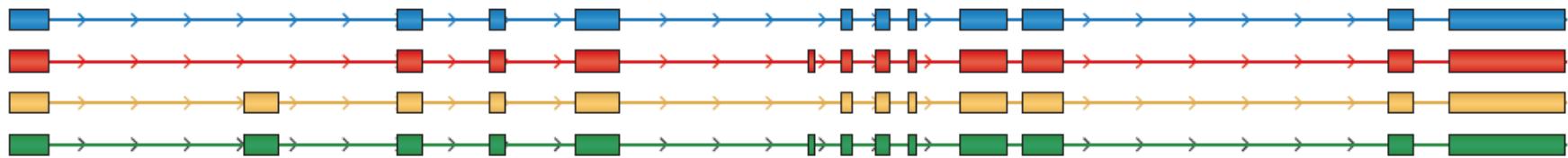


# Genome-Guided Transcript Reconstruction

Traverse paths through the graph to assemble transcript isoforms

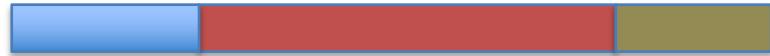


Reconstructed isoforms

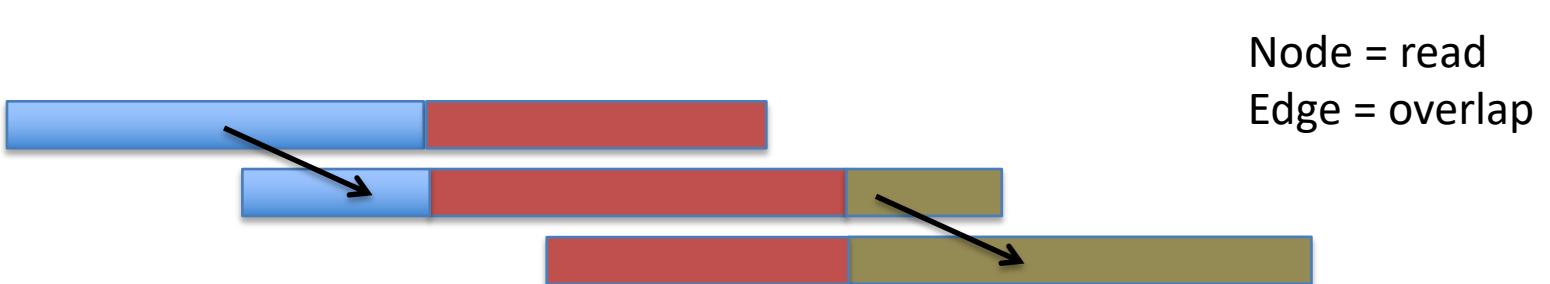
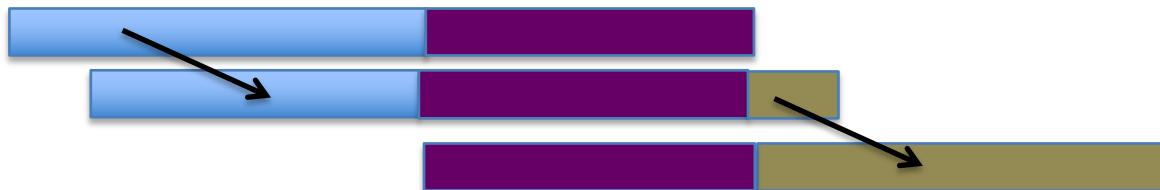


What if you don't have a high quality reference genome sequence?

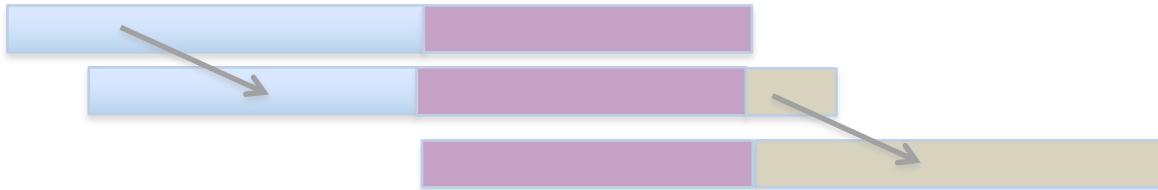
## Read Overlap Graph: Reads as nodes, overlaps as edges



## Read Overlap Graph: Reads as nodes, overlaps as edges



## Read Overlap Graph: Reads as nodes, overlaps as edges

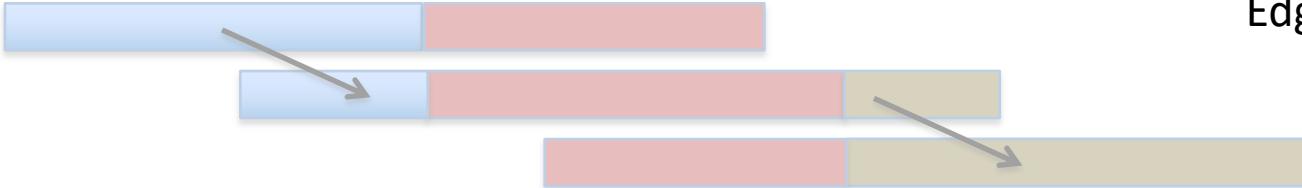


Transcript A



Generate consensus sequence where reads overlap

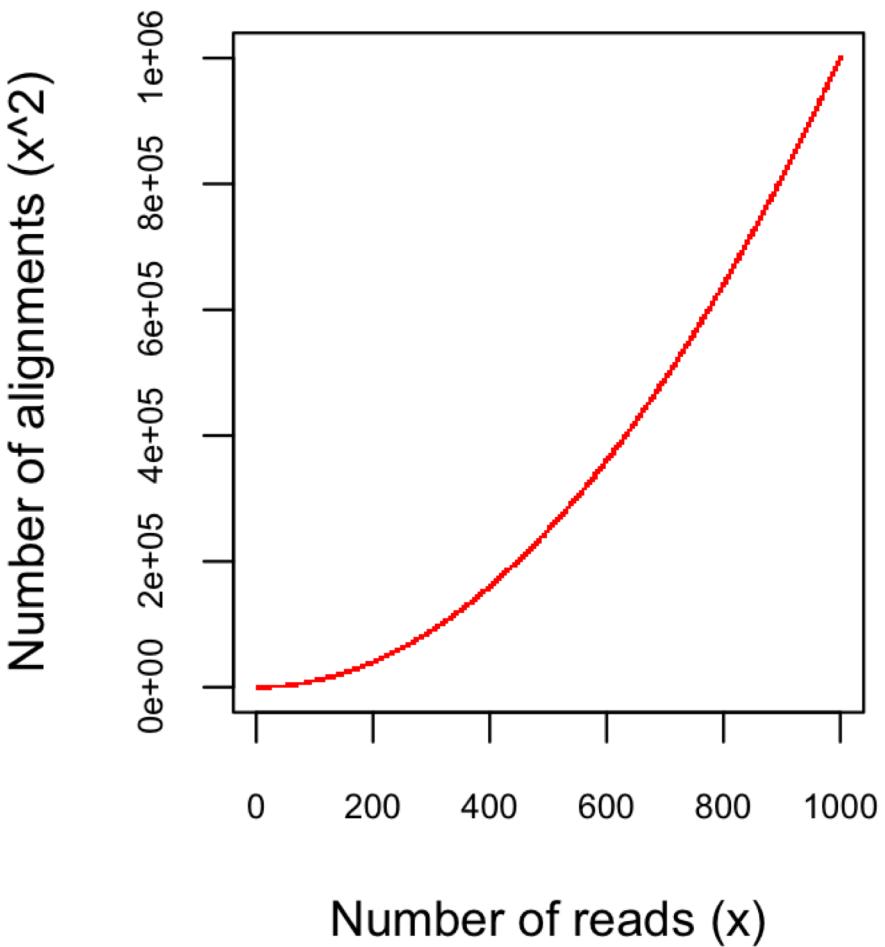
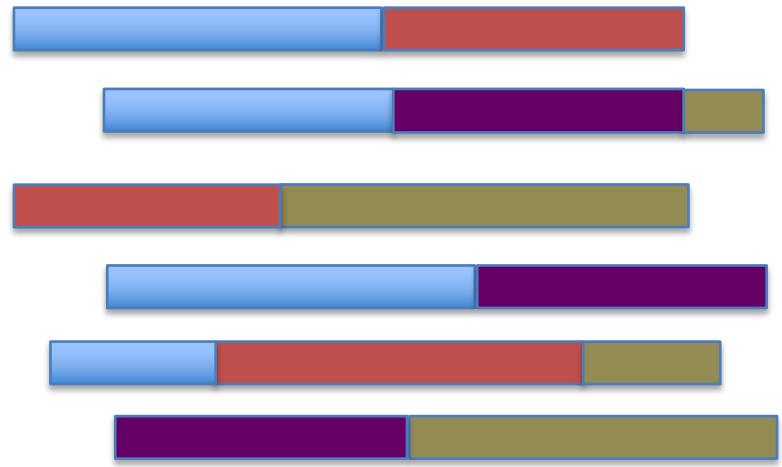
Node = read  
Edge = overlap



Transcript B

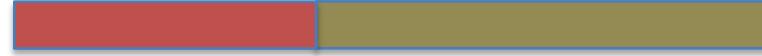


Finding pairwise overlaps between  $n$  reads involves  $\sim n^2$  comparisons.



*Impractical for typical RNA-Seq data (50M reads)*

# No genome to align to... De novo assembly required



Want to avoid  $n^2$  read alignments to define overlaps

Use a de Bruijn graph

# Sequence Assembly via de Bruijn Graphs

Generate all substrings of length k from the reads



# Sequence Assembly via De Bruijn Graphs

Generate all substrings of length k from the reads



# Sequence Assembly via De Bruijn Graphs

Generate all substrings of length k from the reads



Construct the de Bruijn graph



Nodes = unique k-mers

# Sequence Assembly via De Bruijn Graphs

Generate all substrings of length k from the reads



Construct the de Bruijn graph



Nodes = unique k-mers  
Edges = overlap by (k-1)

# Sequence Assembly via De Bruijn Graphs

Generate all substrings of length k from the reads



Construct the de Bruijn graph



Nodes = unique k-mers  
Edges = overlap by (k-1)

# Sequence Assembly via De Bruijn Graphs

Generate all substrings of length k from the reads



Construct the de Bruijn graph



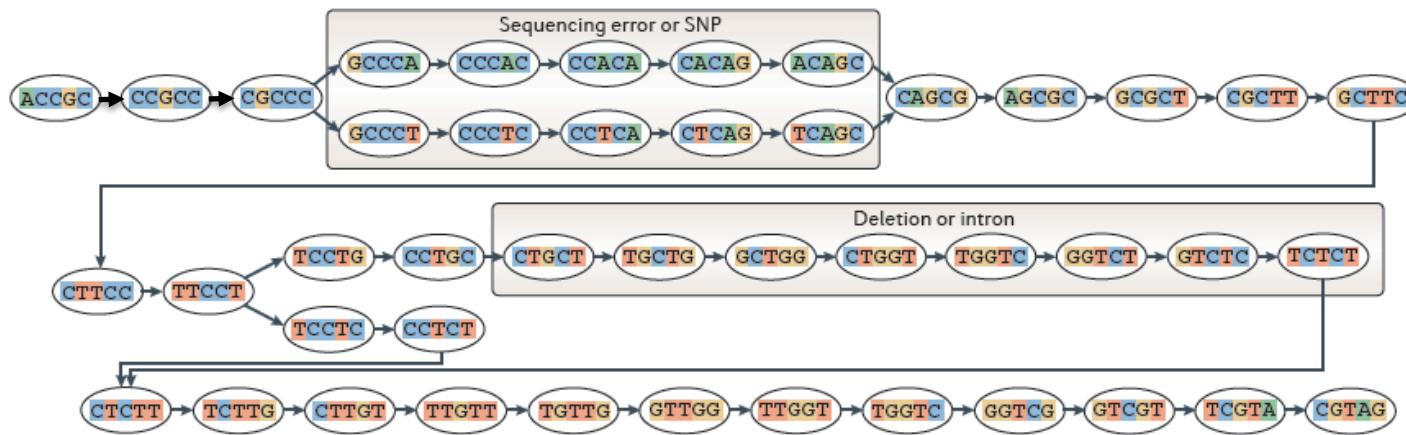
Nodes = unique k-mers  
Edges = overlap by (k-1)

# Sequence Assembly via De Bruijn Graphs

Generate all substrings of length k from the reads

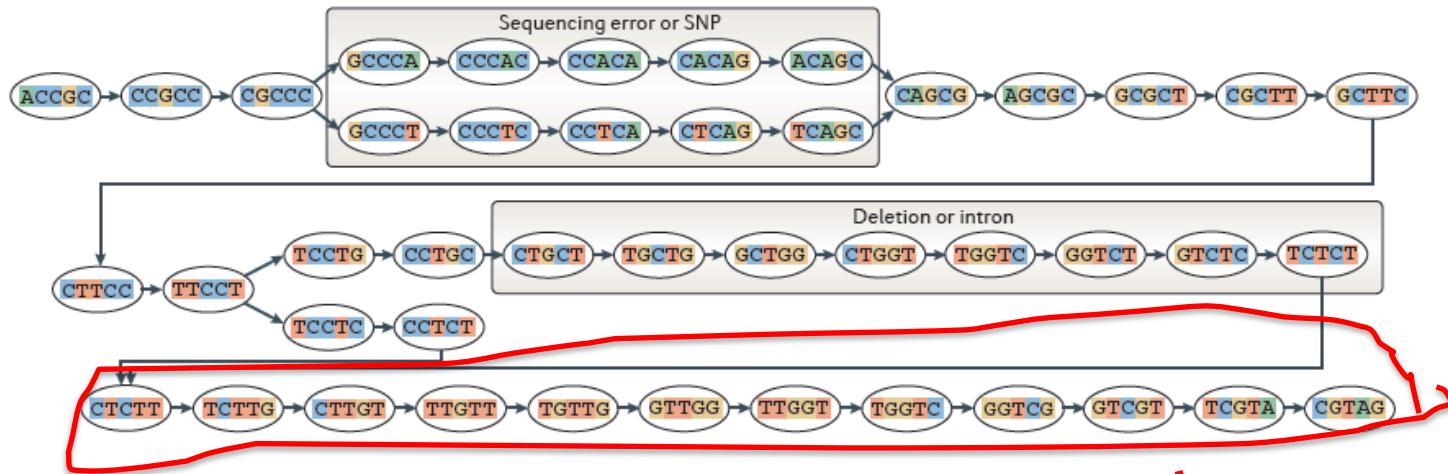
ACAGC	TCCTG	GTCTC		AGCGC	CTCTT	GGTCG	k-mers (k=5)
CACAG	TTCCT	GGTCT		CAGCG	CCTCT	TGGTC	
CCACA	CTTCC	TGGTC	TGTTG	TCAGC	TCCTC	TTGGT	
CCCAC	GCTTC	CTGGT	TTGTT	CTCAG	TTCCT	GTTGG	
GCCCA	CGCTT	GCTGG	CTTGT	CCTCA	CTTCC	TGTTG	
CGCCC	GCGCT	TGCTG	TCTTG	CCCTC	GCTTC	TTGTT	
CCGCC	AGCGC	CTGCT	CTCTT	GCCCT	CGCTT	CTTGT	
ACCGC	CAGCG	CCTGC	TCTCT	CGCCC	GCGCT	TCTTG	
ACCGCCCCACAGCGCTTCCTGCTGGTCTCTTGTG				CGCCCTCAGCGCTTCCTCTTGTGGTCGTAG			
							Reads

Construct the de Bruijn graph

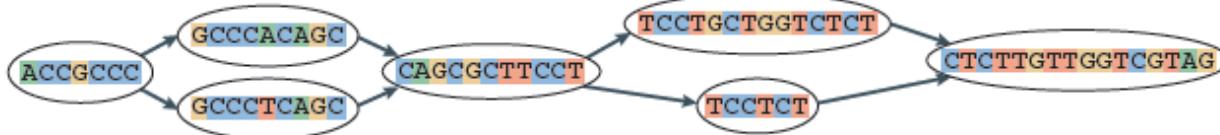


Nodes = unique k-mers  
Edges = overlap by (k-1)

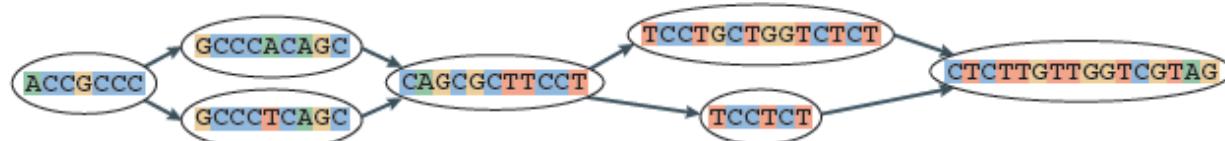
## Construct the de Bruijn graph



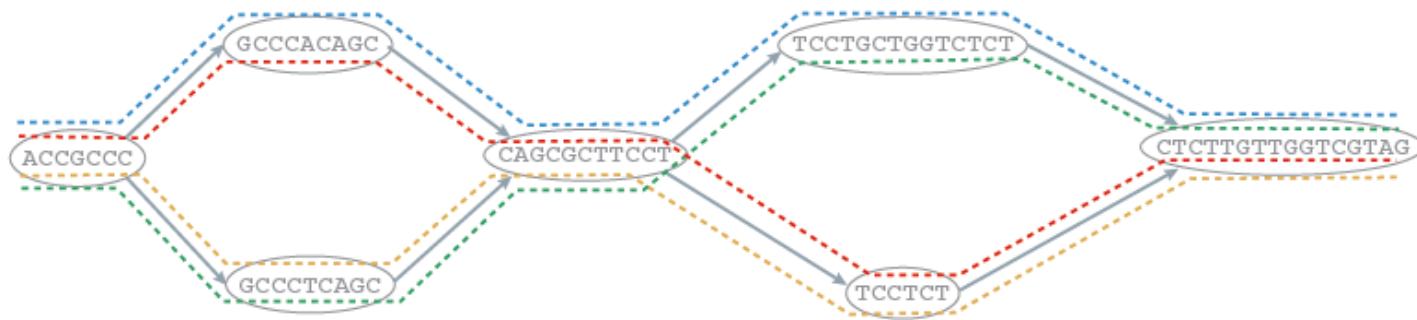
## Collapse the de Bruijn graph



## Collapse the de Bruijn graph



## Traverse the graph



## Assemble Transcript Isoforms

— ACCGCCCCACAGCGCTTCC  
— ACCGCCCCACAGCGCTTCCT — CTTGGTGGTCGTAG  
— ACCGCCCCACAGCGCTTCCT — CTTGGTGGTCGTAG  
— ACCGCCCCACAGCGCTTCCT — CTTGGTGGTCGTAG

# Contrasting Genome and Transcriptome *De novo* Assembly

## Genome Assembly

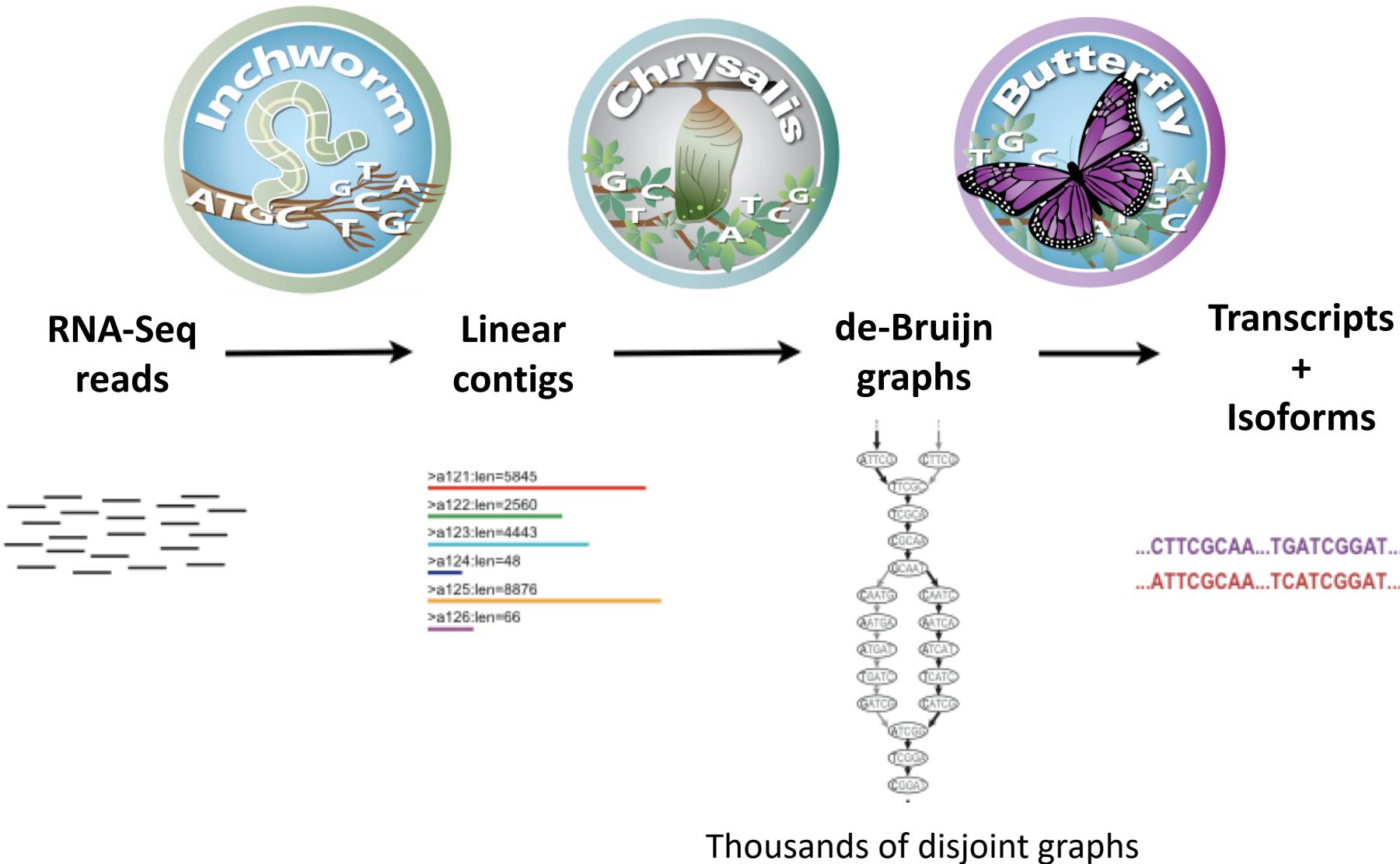
- Uniform coverage
- Single contig per locus
- Assemble small numbers of large Mb-length chromosomes
- Double-stranded data

## Transcriptome Assembly

- Exponentially distributed coverage levels
- Multiple contigs per locus (alt splicing)
- Assemble many thousands of Kb-length transcripts
- Strand-specific data available



# Trinity – How it works:





# Inchworm Algorithm

- Decompose all reads into overlapping Kmers => hashtable(kmer, count)

Read: **AATGTGAAACTGGATTACATGCTGGTATGTC...**

**AATGTGA**

**ATGTGAA**

Overlapping kmers of length (k)

**TGTGAAA**

...

**Kmer Catalog (hashtable)**

Kmer	Count among all reads
<b>AATGTGA</b>	<b>4</b>
<b>ATGTGAA</b>	<b>2</b>
<b>TGTGAAA</b>	<b>1</b>
<b>GATTACA</b>	<b>9</b>



# Inchworm Algorithm

- Decompose all reads into overlapping Kmers => hashtable(kmer, count)
- Identify seed kmer as most abundant Kmer, ignoring low-complexity kmers.

**GATTACA**  
9

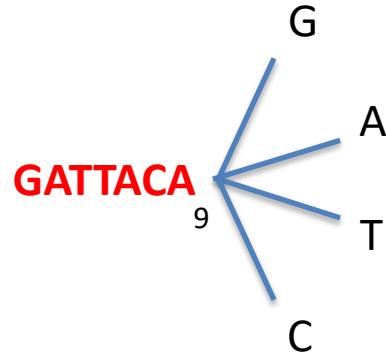
**Kmer Catalog (hashtable)**

Kmer	Count among all reads
AATGTGA	4
ATGTGAA	2
TGTGAAA	1
<b>GATTACA</b>	<b>9</b>



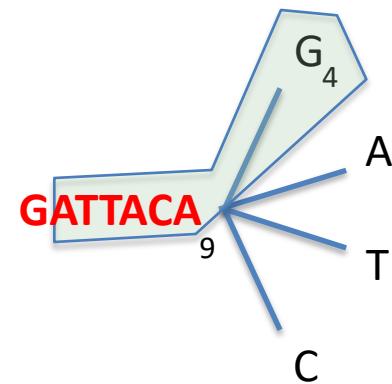
# Inchworm Algorithm

- Decompose all reads into overlapping Kmers => hashtable(kmer, count)
- Identify seed kmer as most abundant Kmer, ignoring low-complexity kmers.
- Extend kmer at 3' end, guided by coverage.



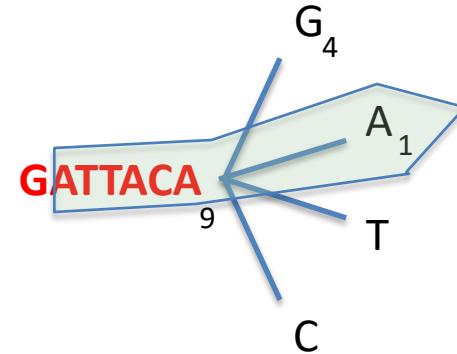


# Inchworm Algorithm



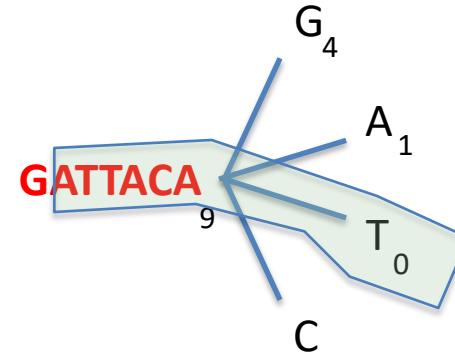


# Inchworm Algorithm



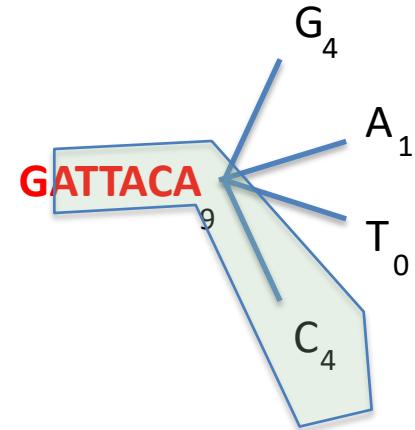


# Inchworm Algorithm



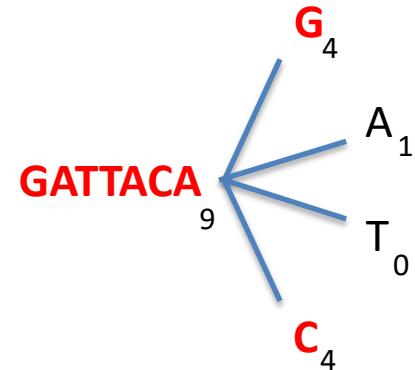


# Inchworm Algorithm



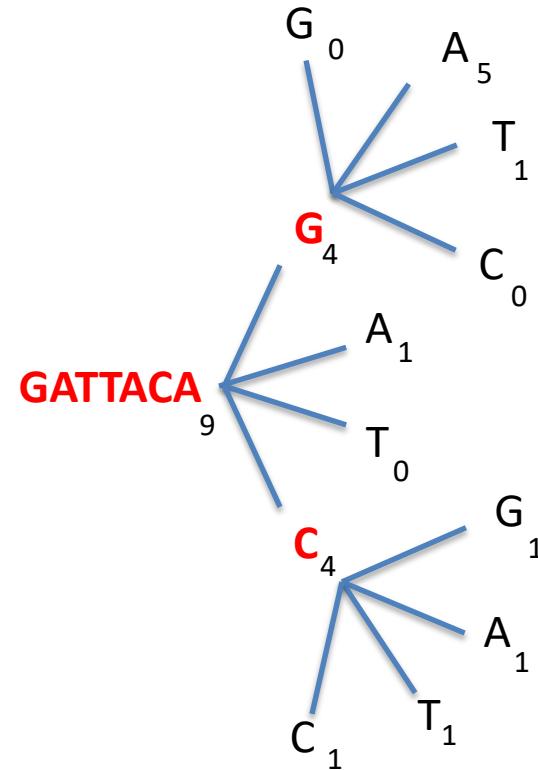


# Inchworm Algorithm



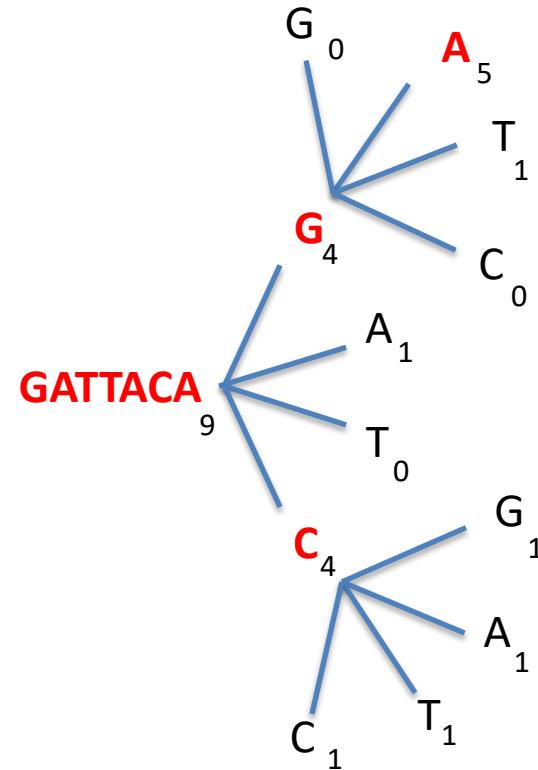


# Inchworm Algorithm



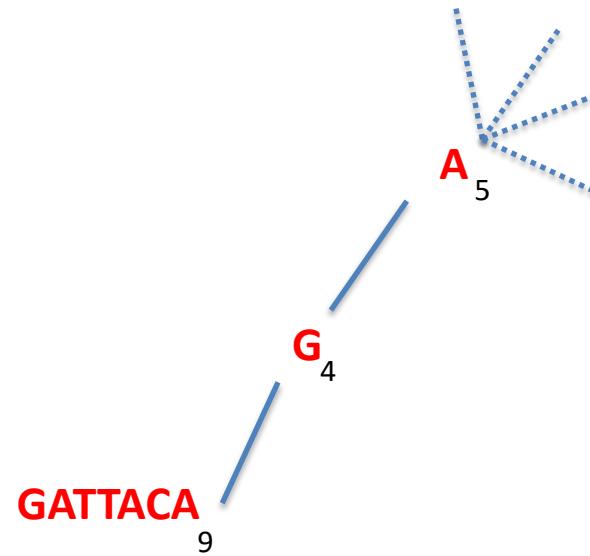


# Inchworm Algorithm



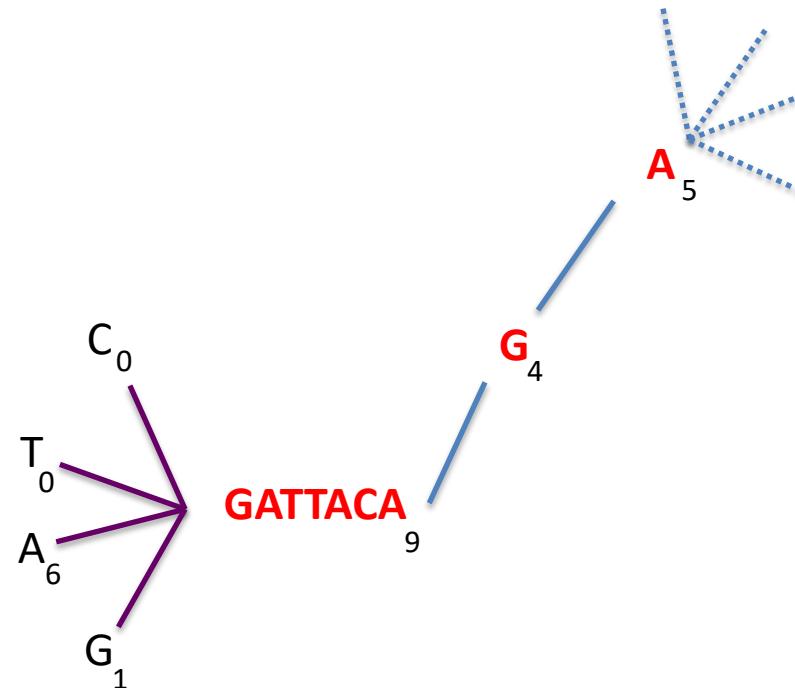


# Inchworm Algorithm



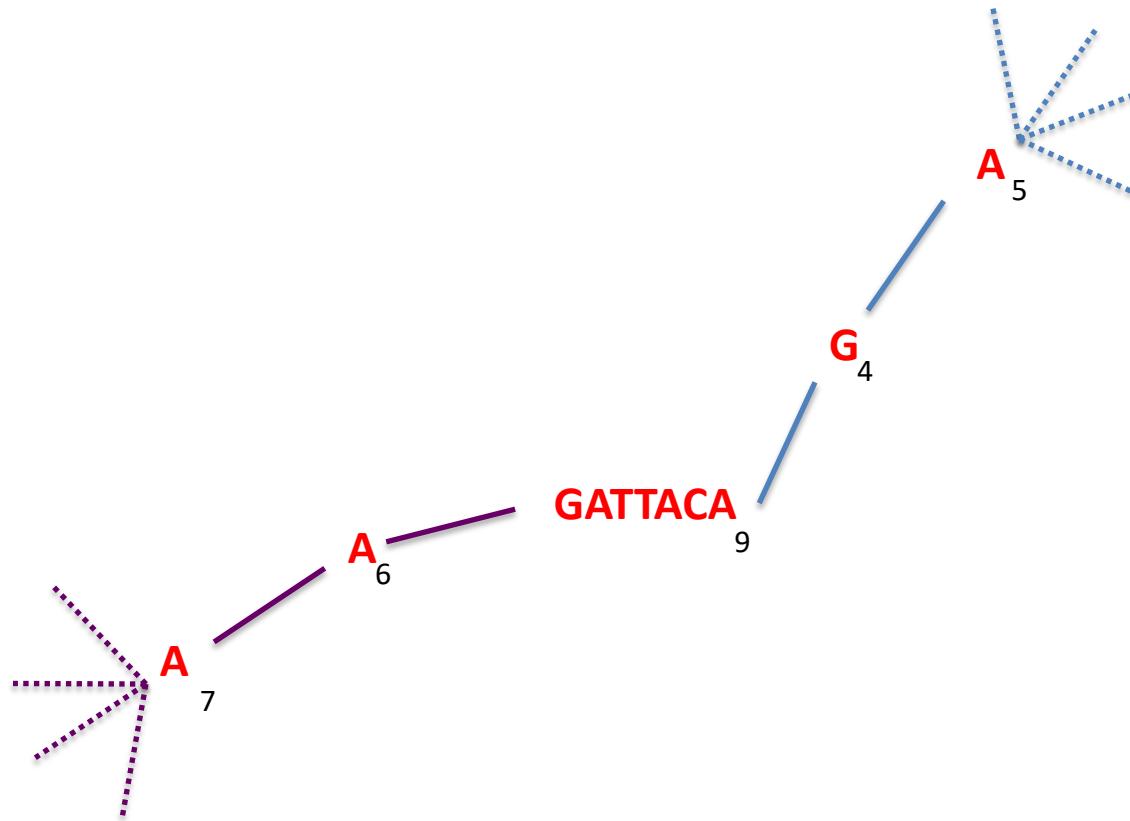


# Inchworm Algorithm





# Inchworm Algorithm



Report contig: ....**AAGATTACAGA**....

Remove assembled kmers from catalog, then repeat the entire process.

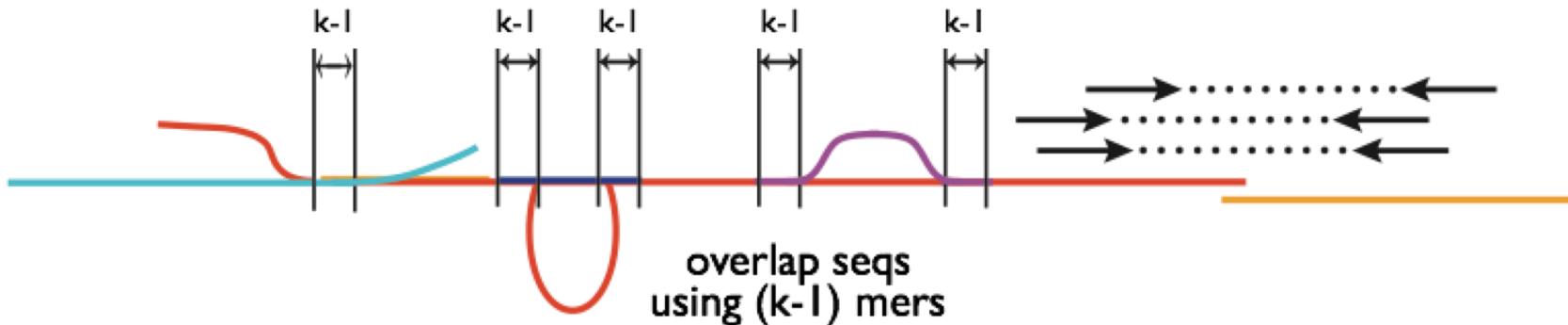
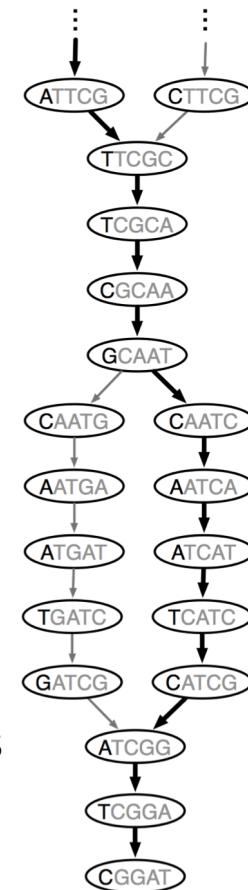
# Chrysalis

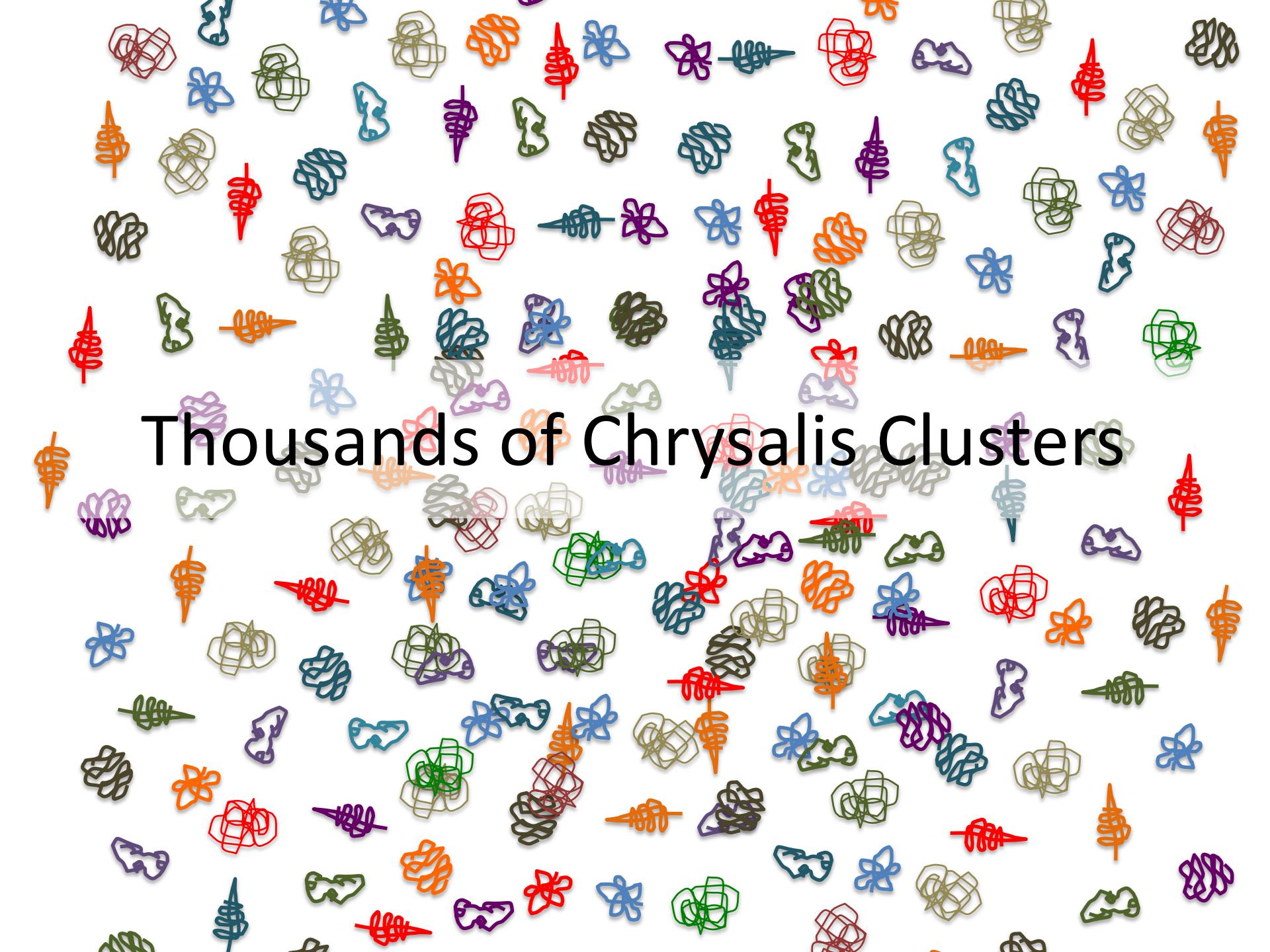
>a121:len=5845  
>a122:len=2560  
>a123:len=4443  
>a124:len=48  
>a125:len=8876  
>a126:len=66

Integrate isoforms via k-1 overlaps

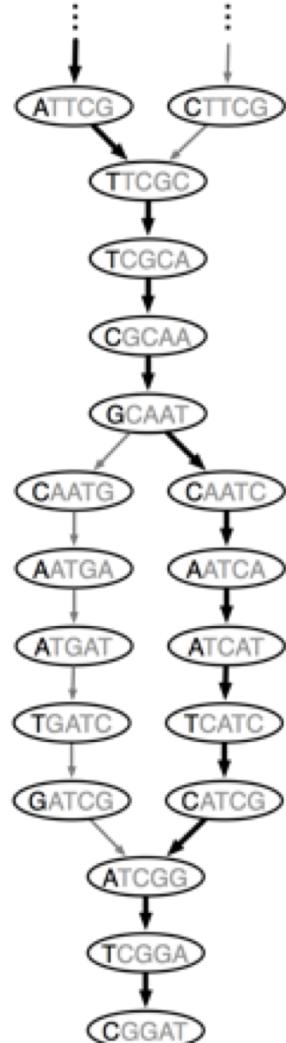


Build de Bruijn Graphs (ideally, one per gene)



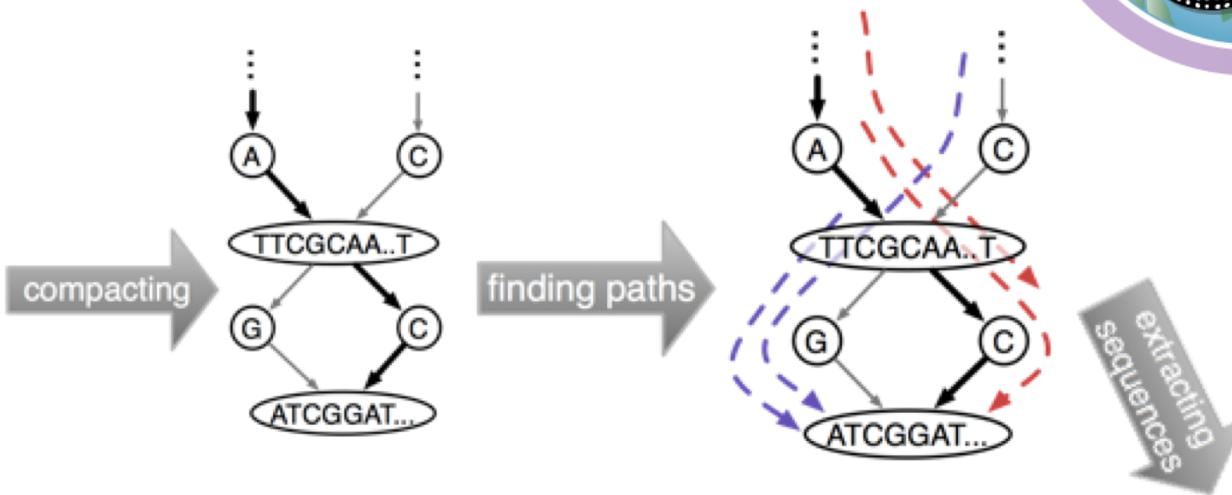


**Thousands of Chrysalis Clusters**



de Bruijn  
graph

# Butterfly



compact  
graph

compact  
graph with  
reads

..CTTCGCAA..TGATCGGAT...  
..ATTCGCAA..TCATCGGAT...  
sequences  
(isoforms and paralogs)

# Trinity output: A multi-fasta file

NATURE PROTOCOLS | PROTOCOL

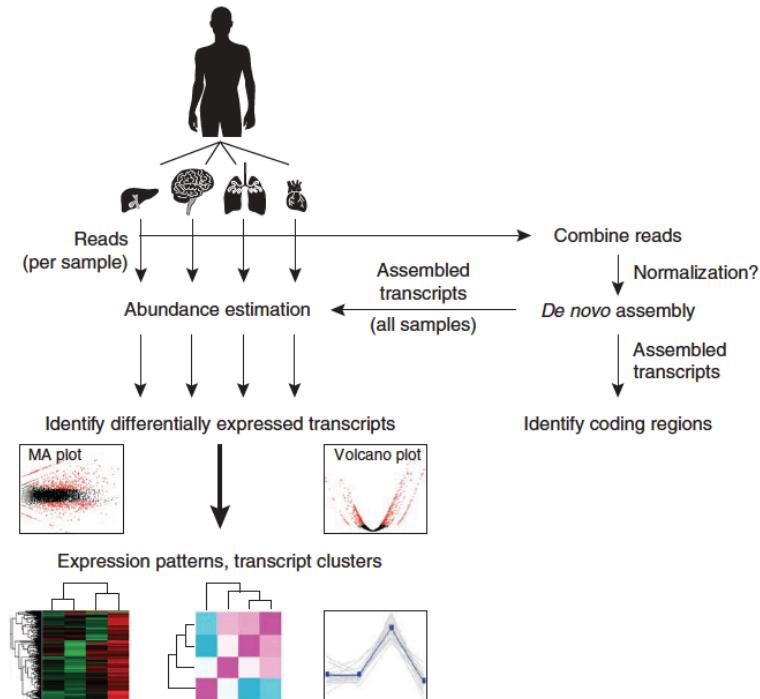
## *De novo* transcript sequence reconstruction from RNA-seq using the Trinity platform for reference generation and analysis

Brian J Haas, Alexie Papanicolaou, Moran Yassour, Manfred Grabherr, Philip D Blood, Joshua Bowden, Matthew Brian Couger, David Eccles, Bo Li, Matthias Lieber, Matthew D MacManes, Michael Ott, Joshua Orvis, Nathalie Pochet, Francesco Strozzi, Nathan Weeks, Rick Westerman, Thomas William, Colin N Dewey, Robert Henschel, Richard D LeDuc, Nir Friedman & Aviv Regev

[Affiliations](#) | [Contributions](#) | [Corresponding authors](#)

*Nature Protocols* 8, 1494–1512 (2013) | doi:10.1038/nprot.2013.084

Published online 11 July 2013



# RNA-Seq De novo Assembly Using Trinity

► Pages 27



## Quick Guide for the Impatient

Trinity assembles transcript sequences from Illumina RNA-Seq data.

Download Trinity [here](#).

Build Trinity by typing 'make' in the base installation directory.

Assemble RNA-Seq data like so:

```
Trinity --seqType fq --left reads_1.fq --right reads_2.fq --CPU 6 --max_memory 20G
```

Find assembled transcripts as: 'trinity\_out\_dir/Trinity.fasta'

Use the documentation links in the right-sidebar to navigate this documentation, and contact our [Google group for technical support](#).

- [Trinity Wiki Home](#)
- [Installing Trinity](#)
  - [Trinity Computing Requirements](#)
  - [Accessing Trinity on Publicly Available Compute Resources](#)
  - [Run Trinity using Docker](#)
- [Running Trinity](#)
  - [Genome Guided Trinity Transcriptome Assembly](#)
  - [Gene Structure Annotation of Genomes](#)
- [Trinity process and resource monitoring](#)
  - [Monitoring Progress During a Trinity Run](#)
  - [Examining Resource Usage at the End of a Trinity Run](#)
- [Output of Trinity Assembly](#)
- [Assembly Quality Assessment](#)
  - [Counting Full-length Transcripts](#)
  - [RNA-Seq Read Representation](#)
  - [Contig Nx and ExN50 stats](#)
  - [Examine strand-specificity of reads](#)
- [Downstream Analyses](#)

# ProgForBio Exercise 1: Build a program that counts k-mers

Generate all substrings of length k from the reads



Using a kmer size of 8 and reporting the top 10 kmers and their counts:

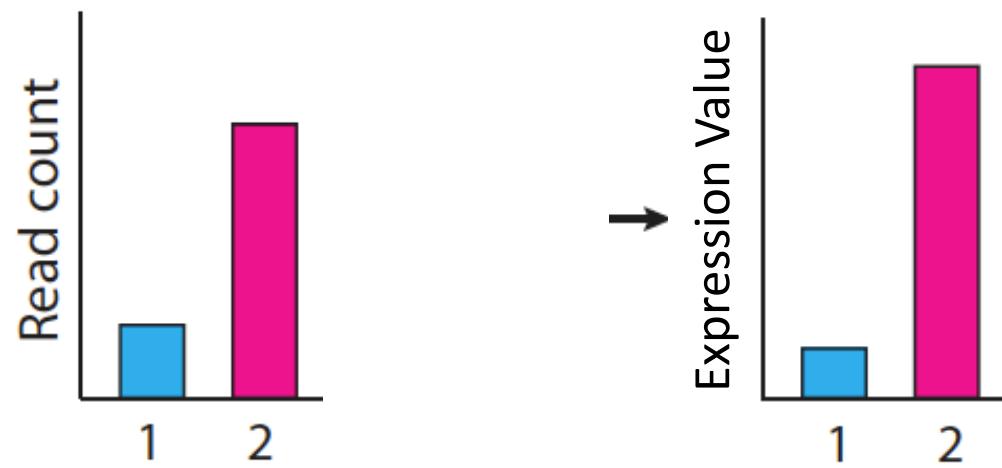
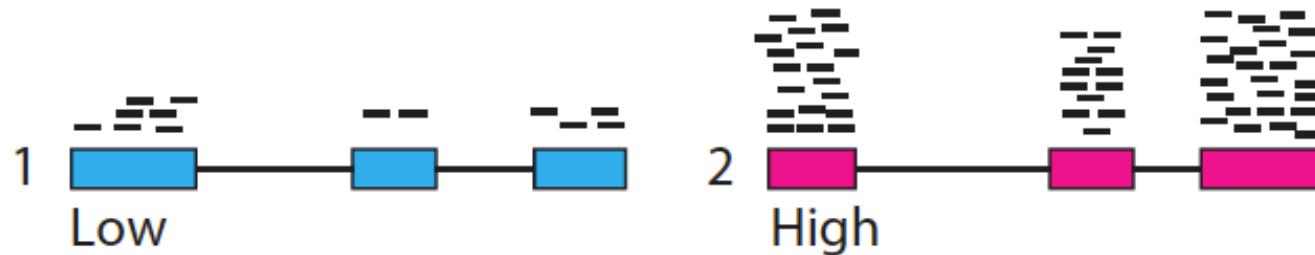
```
kmer_counter.py 8 reads.left.fq 10
```

TTTTTTTT	1743
AAAAAAA	1204
CCAGCCTT	666
GAAGCTGG	627
CAGGCAGG	549
TTTGCTGT	536
GCCTGCTG	533
CTTGGTCT	525
AAGCTGGA	518
TTTTATTT	504

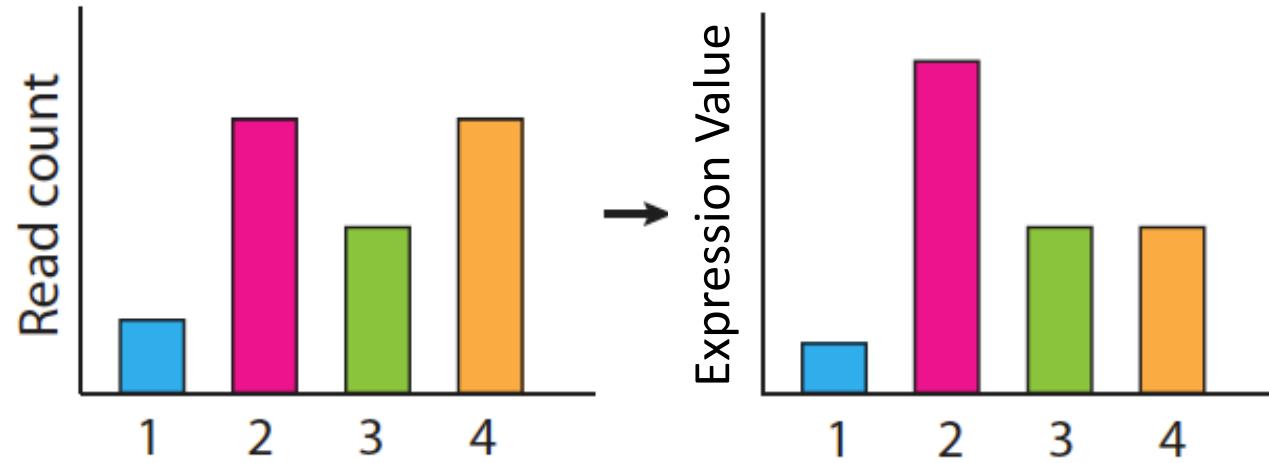
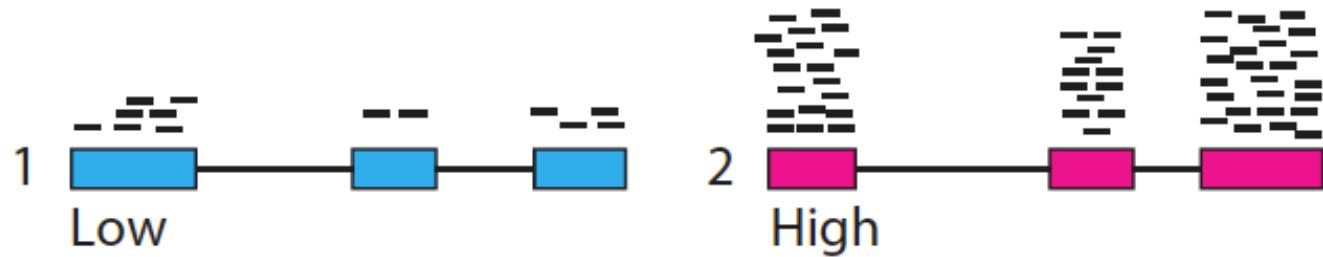
# Abundance Estimation

(Aka. Computing Expression Values)

# Calculating expression of genes and transcripts



# Calculating expression of genes and transcripts



# Normalized Expression Values

- Transcript-mapped read counts are normalized for both length of the transcript and total depth of sequencing.
- Reported as: Number of RNA-Seq **F**ragments  
**P**er **K**ilobase of transcript  
per total **M**illion fragments mapped

**FPKM**

RPKM (reads per kb per M) used with Single-end RNA-Seq reads  
FPKM used with Paired-end RNA-Seq reads.

# Transcripts per Million (TPM)

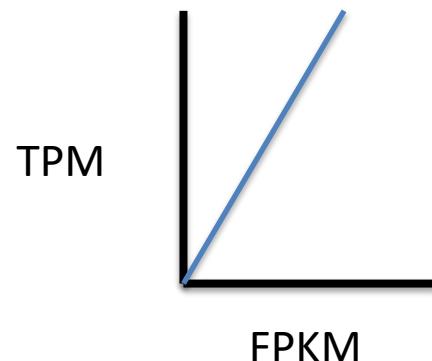
$$TPM_i = \frac{FPKM_i}{\sum_j FPKM} * 1e6$$

Preferred metric for measuring expression

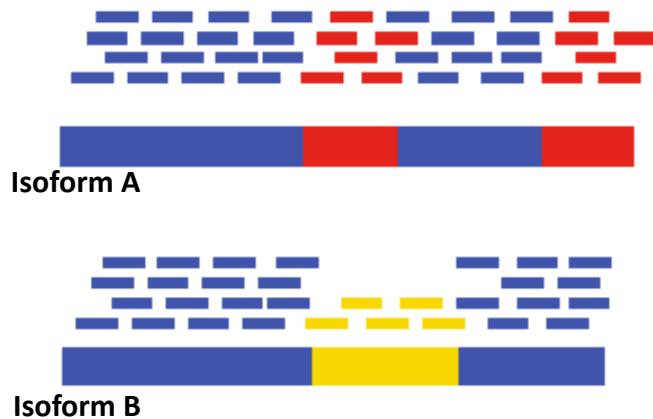
- Better reflects transcript concentration in the sample.
- Nicely sums to 1 million

Linear relationship between TPM and FPKM values.

Both are valid metrics, but best to be consistent.



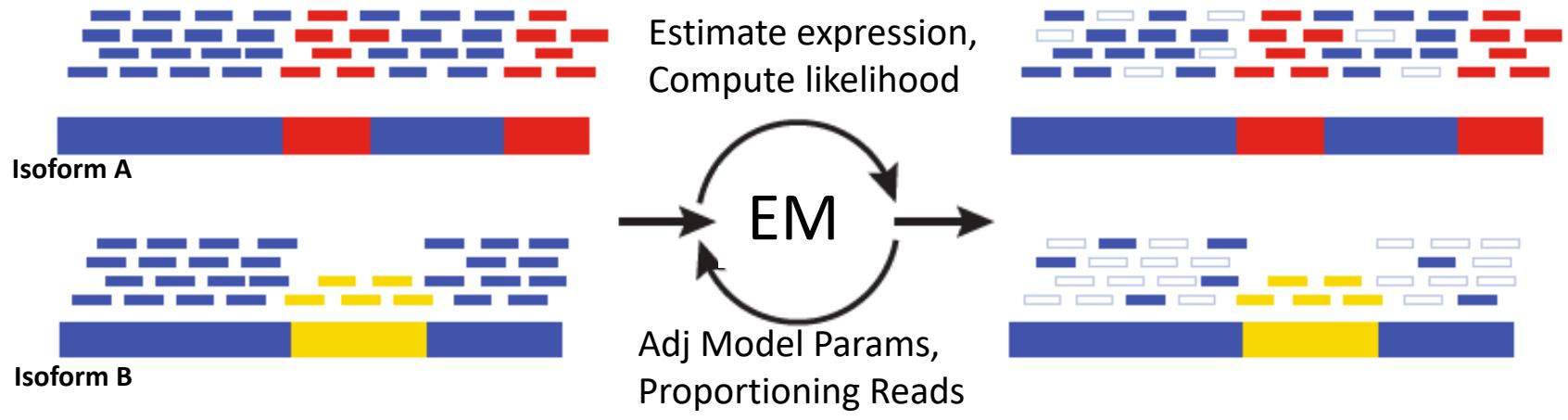
# Multiply-mapped Reads Confound Abundance Estimation



Blue = multiply-mapped reads

Red, Yellow = uniquely-mapped reads

# Multiply-mapped Reads Confound Abundance Estimation

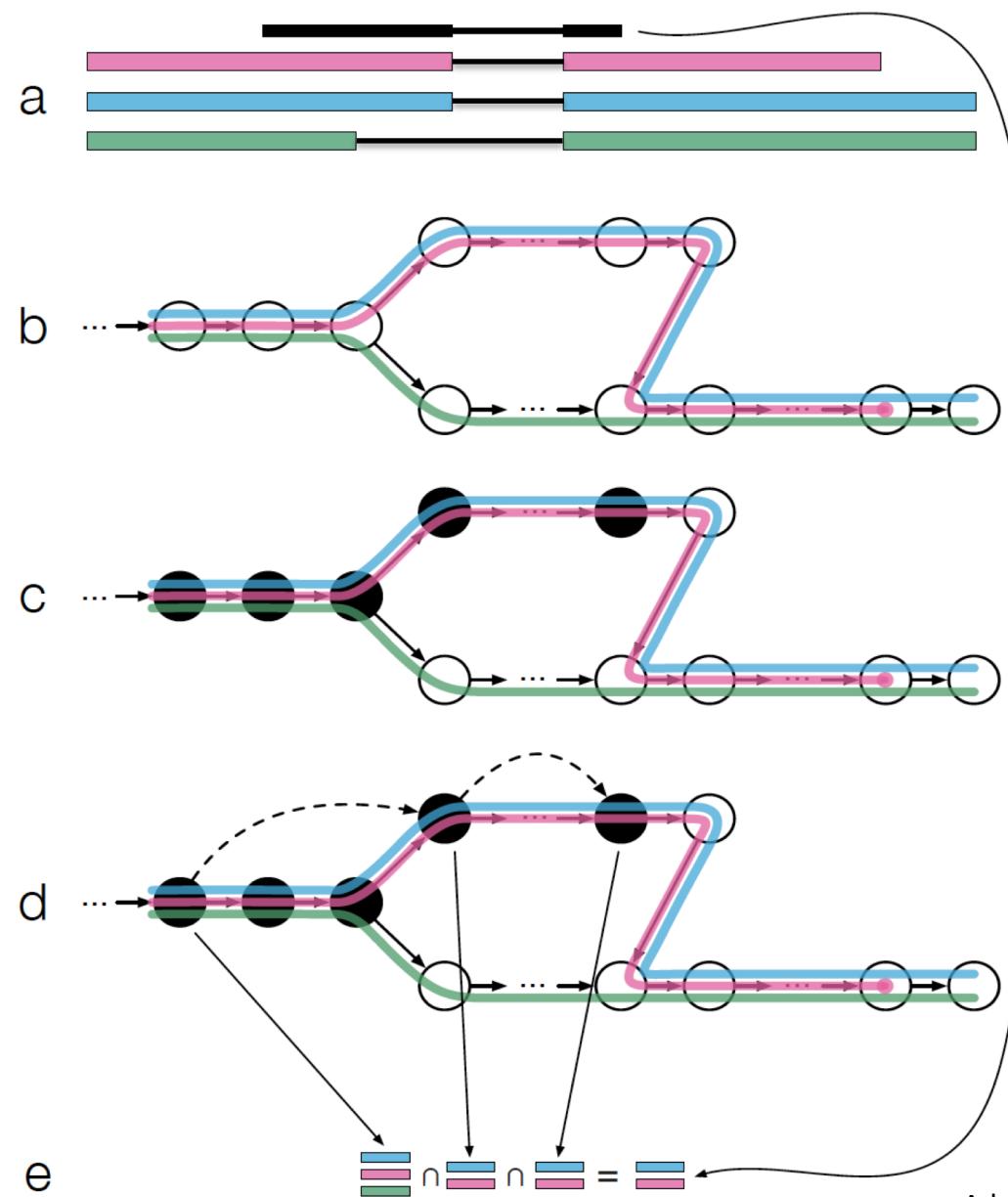


Use Expectation Maximization (EM) to find the most likely assignment of reads to transcripts.

Performed by:

- Cufflinks, String Tie (Tuxedo)
- RSEM, eXpress (genome-free)
- Kallisto, Salmon (alignment-free)

# Fast Abundance Estimation Using Pseudo-alignments and Equivalence Classes (Kallisto software, Bray et al., NBT 2016)



Adapted from Fig 1 from Bray et al.

# ProgForBio Exercise 2: Build a utility that measures expression by counting aligned reads

## Estimating Gene Expression Levels

---

Write a python program that reads in the 'bowtie2.bam' file and generates a table containing the number of reads mapped to each gene.

For example:

```
gene_read_counter.py bowtie2.bam
```

would return:

```
CG14995 1663
S-Lap3 1608
Eno 1423
sqd 877
AdipoR 801
Est-6 789
...
```

[https://github.com/trinityrnaseq/CSHLProgForBiol2018/tree/master/Exercise\\_2-aligned\\_reads\\_to\\_expression](https://github.com/trinityrnaseq/CSHLProgForBiol2018/tree/master/Exercise_2-aligned_reads_to_expression)

# Summary of Key Points

- RNA-Seq is a versatile method for transcriptome analysis enabling quantification and novel transcript discovery.
- Expression quantification is based on sampling and counting reads derived from transcripts
- Trinity assembly and supported downstream computational analysis tools facilitate transcriptome studies.

# RNA-Seq De novo Assembly Using Trinity



Visit website for more info:

<http://trinityrnaseq.github.io>

## Quick Guide for the Impatient

Trinity assembles transcript sequences from Illumina RNA-Seq data.

Download Trinity [here](#).

Build Trinity by typing 'make' in the base installation directory.

Assemble RNA-Seq data like so:

```
Trinity --seqType fq --left reads_1.fq --right reads_2.fq --CPU 6 --max_memory 20G
```

Find assembled transcripts as: 'trinity\_out\_dir/Trinity.fasta'

Use the documentation links in the right-sidebar to navigate this documentation, and contact our [Google group for technical support](#).

- [Trinity Wiki Home](#)
- [Installing Trinity](#)
  - [Trinity Computing Requirements](#)
  - [Accessing Trinity on Publicly Available Compute Resources](#)
  - [Run Trinity using Docker](#)
- [Running Trinity](#)
  - [Genome Guided Trinity Transcriptome Assembly](#)
  - [Gene Structure Annotation of Genomes](#)
- [Trinity process and resource monitoring](#)
  - [Monitoring Progress During a Trinity Run](#)
  - [Examining Resource Usage at the End of a Trinity Run](#)
- [Output of Trinity Assembly](#)
- [Assembly Quality Assessment](#)
  - [Counting Full-length Transcripts](#)
  - [RNA-Seq Read Representation](#)
  - [Contig Nx and ExN50 stats](#)
  - [Examine strand-specificity of reads](#)
- [Downstream Analyses](#)

► Pages 27

Let's go write some code! ☺