Transcriptome Assembly

Presenter: *Nguyen Le Duc Minh*

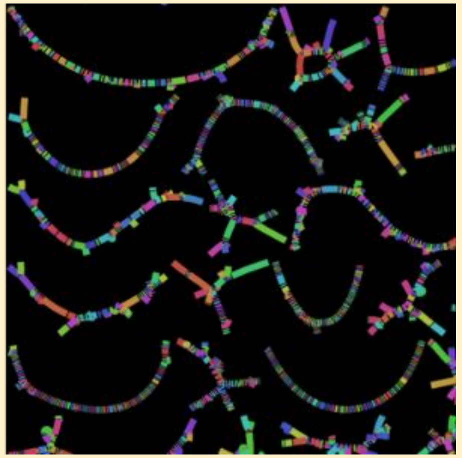
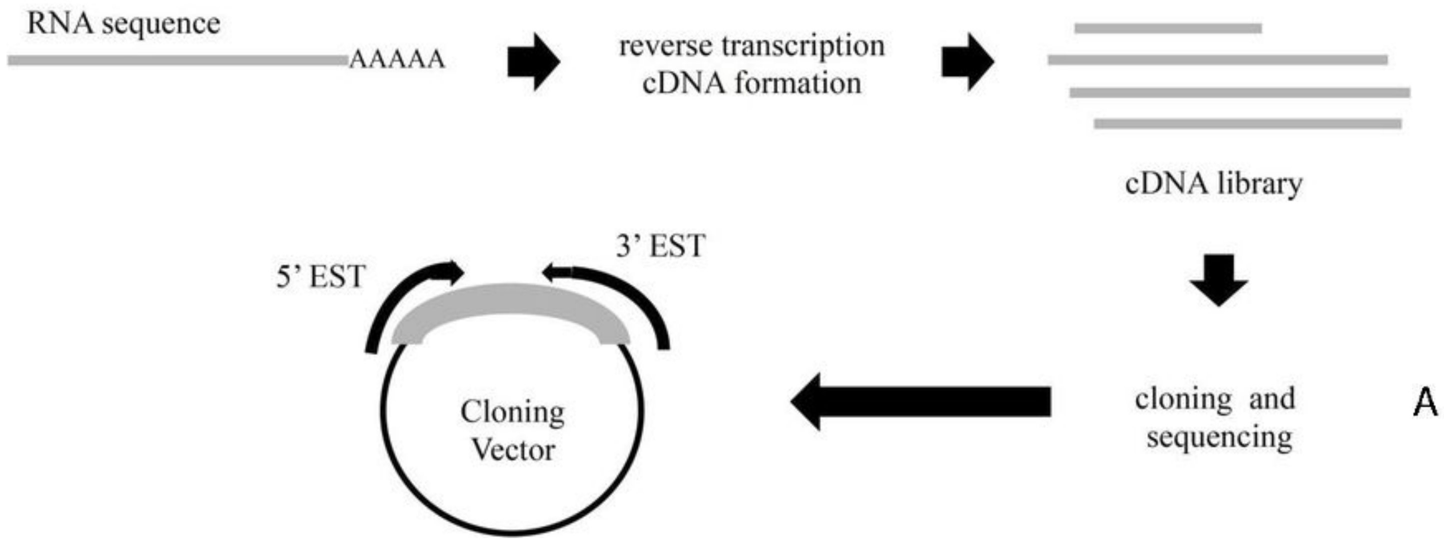


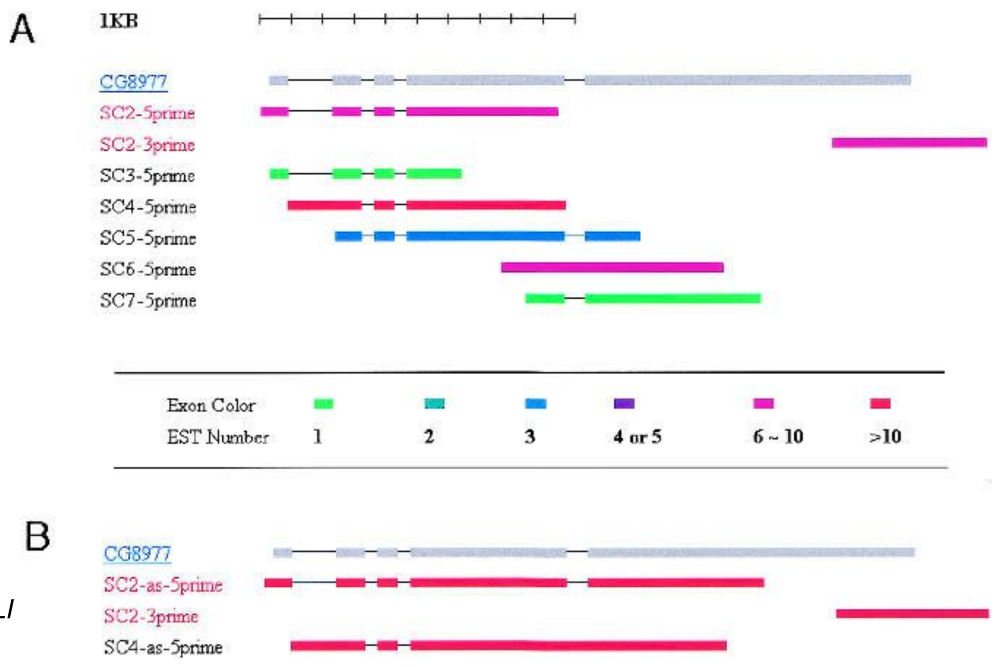
Table of Content

Transcriptome assembly fundamentals

Introduction to Transcriptome assembly

Roots of RNA-seq assembly
Expressed sequence tag (EST)





Processing of EST involved:

1. Clustering: Grouping similar EST reads together by calculating pairwise overlaps
2. Assembly: separately within each cluster

https://www.researchgate.net/publication/311343869_GENE_EXPRESSION_AN_OVERVIEW_OF_METHODS_AND_APPLICATIONS_FOR_CANCER_RESEARCH

Typically ESTs versus Present-day RNA-seq data

- Those 2 previous steps are still constituting the main steps of the transcript assembly process

1. Finding the reads which belong to the same locus
2. Constructing the graph representing the transcripts within each locus

- One big difference is that typically ESTs represented only fragments and partial transcripts. On the other hand, the nature of today's high-throughput data enables the representation of full length transcripts.

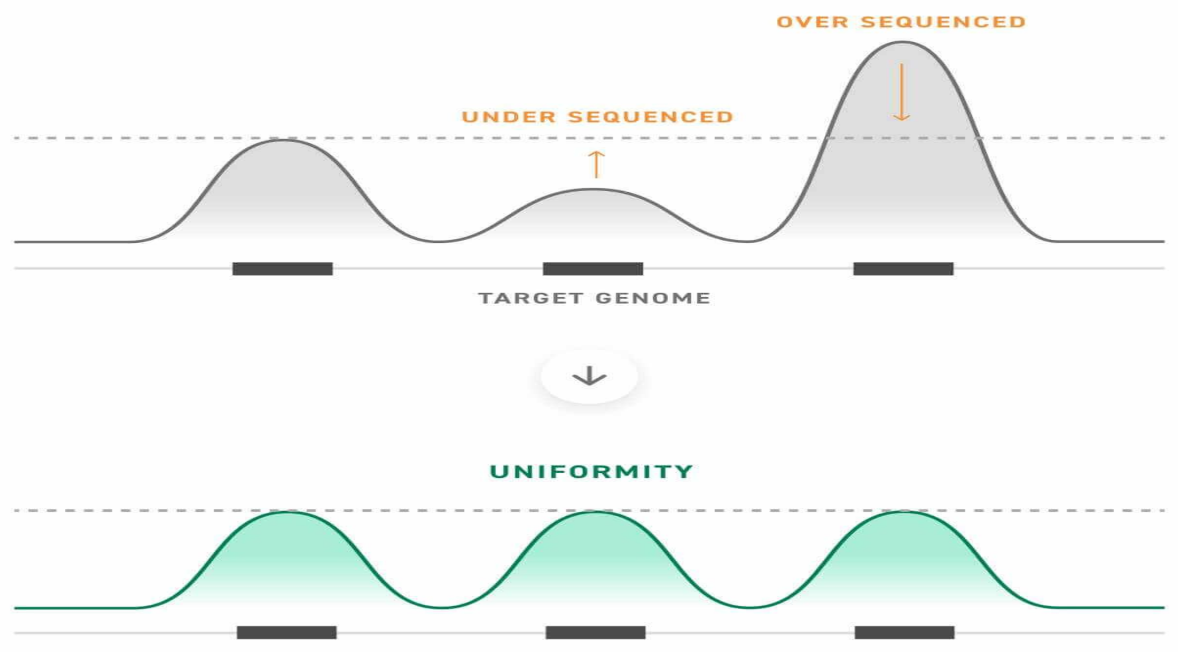
Transcriptome Assembly versus Genome Assembly

- With RNA-seq data, the abundance of gene expression can vary several magnitudes between genes.
- Furthermore, different isoforms of the same gene can be expressed at different levels. Highly different abundances between genes can cause challenges:

1. Uniformity of sequencing depth
2. Several isoforms from the same locus
3. Identify which isoforms are real

1. Uniformity of sequencing depth

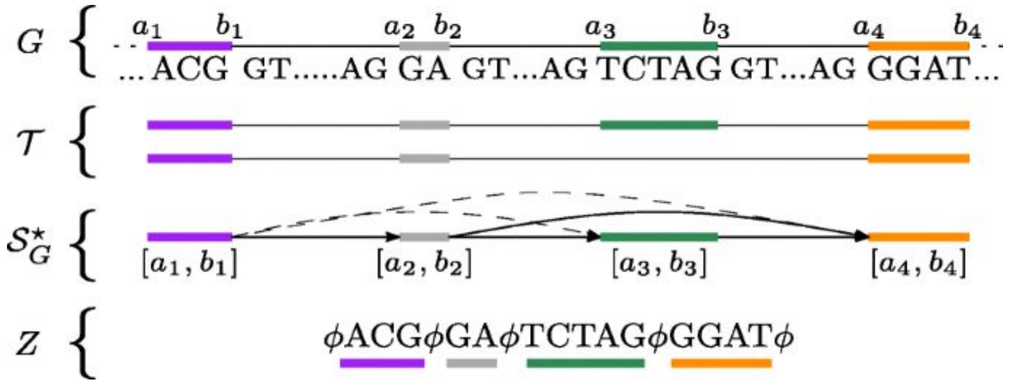
- More sequencing depth to represent *less abundant genes* and *rare events*.
- In order to balance abundance differences between the genes, there are wet laboratory procedures for *library normalization*.

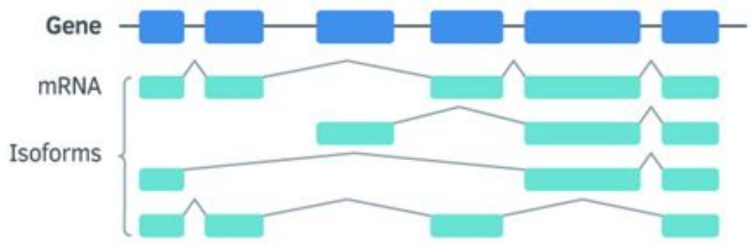


https://www.twistbioscience.com/blog/science/targeted-rna-seq-expands-toolkit-early-cancer-detection

2. Variety of isoforms from the same locus

- Same exon of a gene is present in different contexts with other exons depending on the transcript.
- In transcriptome assembly graph, nodes represent exons and arcs represent splicing events. Branches in node connections correspond to alternative splicing.
- One exon node is present only once in an isoform, but the same exon can be in multiple different isoforms.

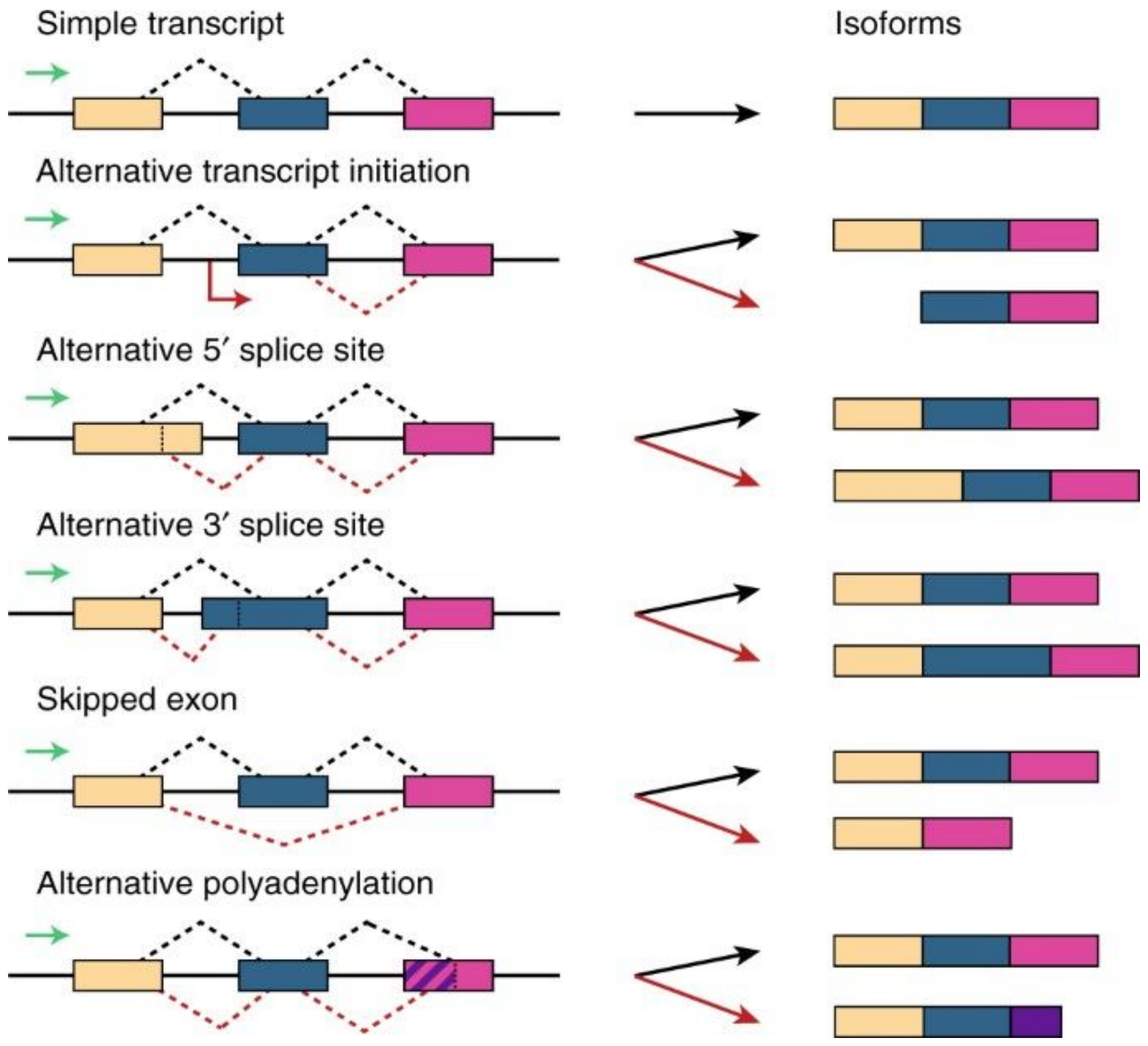

G
a_1 b_1 a_2 b_2 a_3 b_3 a_4 b_4
... ACG GT.....AG GA GT...AG TCTAG GT...AG GGAT...
$\mathcal{T}$
$S^{\star}_G$
$[a_1, b_1]$ $[a_2, b_2]$ $[a_3, b_3]$ $[a_4, b_4]$
Z
ϕACGϕGAϕTCTAGϕGGATϕ



https://bmcbioinformatics.biomedcentral.com/articles/10.1186/s12859-018-2436-3

https://www.dnalinkseqlab.com/rna-sequencing/

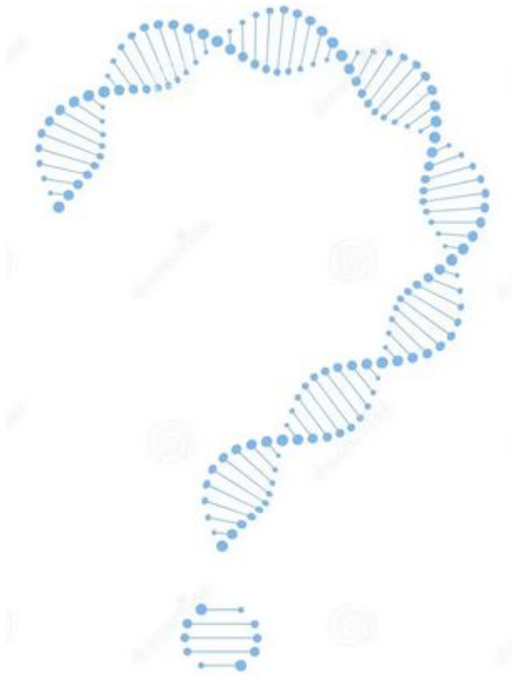
Figure illustrates different isoforms with the same exon


Simple transcript
Isoforms
Alternative transcript initiation
Alternative 5′ splice site
Alternative 3′ splice site
Skipped exon
Alternative polyadenylation

https://www.nature.com/articles/s41467-018-03402-w

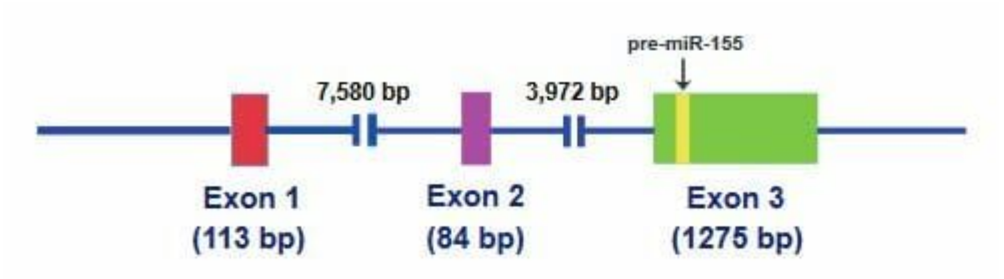
3. Which isoforms are real ?

- One of the challenges in the transcript assembly is to find which isoforms are real from all potential candidates.
- The problem comes from short sequence read and emerges when we try to build long sequences from short fragments.



Complexity of Transcript Reconstruction

Suppose that there are three exons in a gene, what are the number of possible isoforms ?



$$\sum_{k=1}^{N} \binom{N}{k} = 2^N - 1$$

N: number of exons

Impossible iteration

- The problem is to find which ones are true.
- The number of possible sets of isoforms is calculated as below, with each isoform is present or not present in the transcriptome:

$$2^{2^N - 1} - 1$$

- This formula shows that the sizes of possible isoform increase rapidly as the number of exon N grows.
- Therefore, it is not practical to enumerate all possible solutions to test which one matches best with the data.

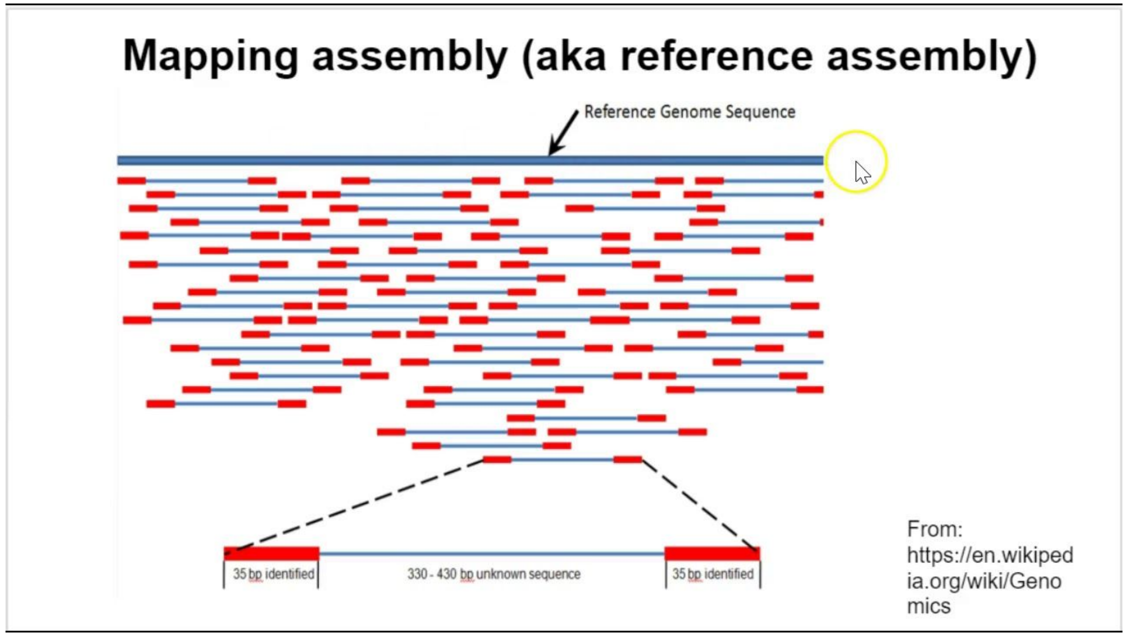
Assembly process (1)

- There are two approaches for transcript reconstruction:

1. Mapping-based assembly
2. De novo assembly

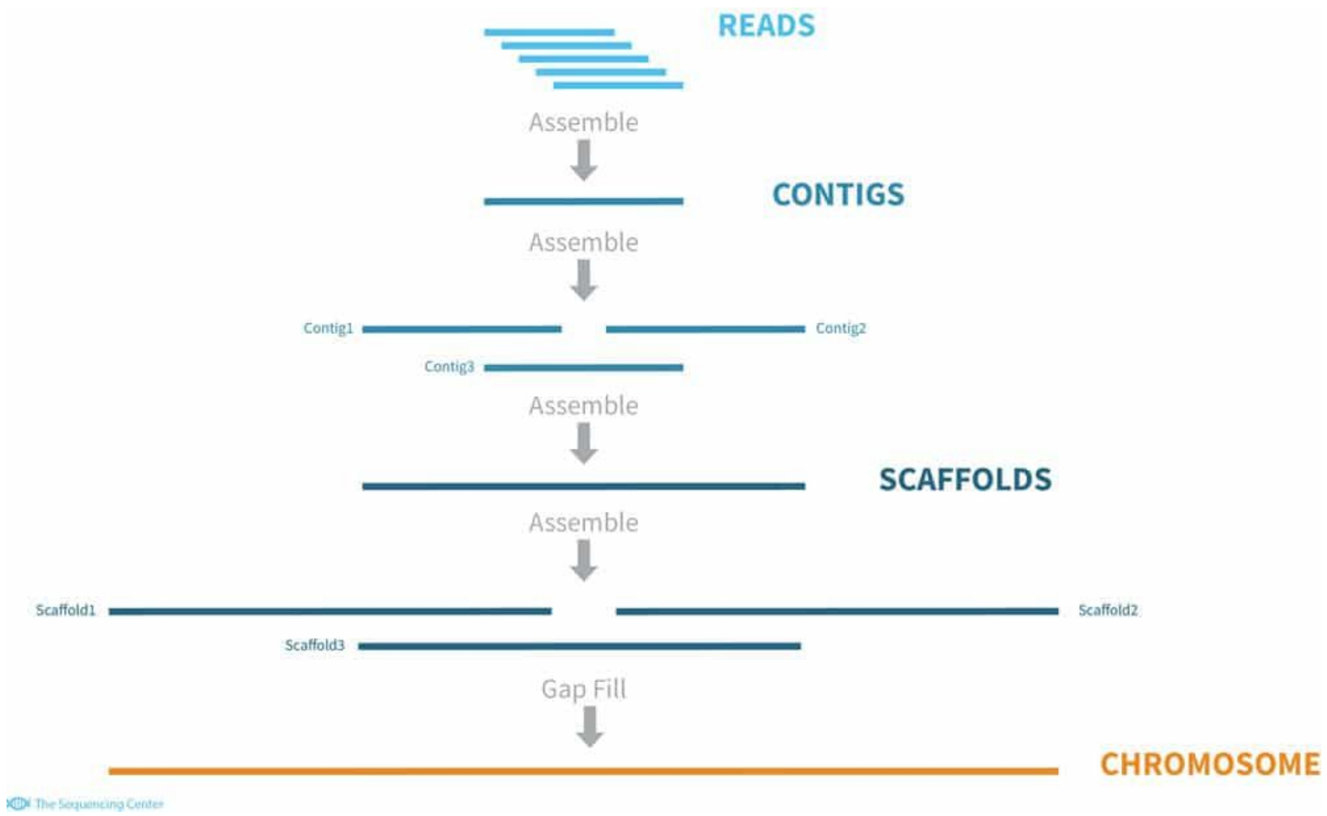
- Both involve constructing a graph for each locus based on RNA-seq reads.
- Both methods also include a problem of how to split the data so that a single graph represents only a single locus.

Comparison between 2 methods



Mapping assembly (aka reference assembly)

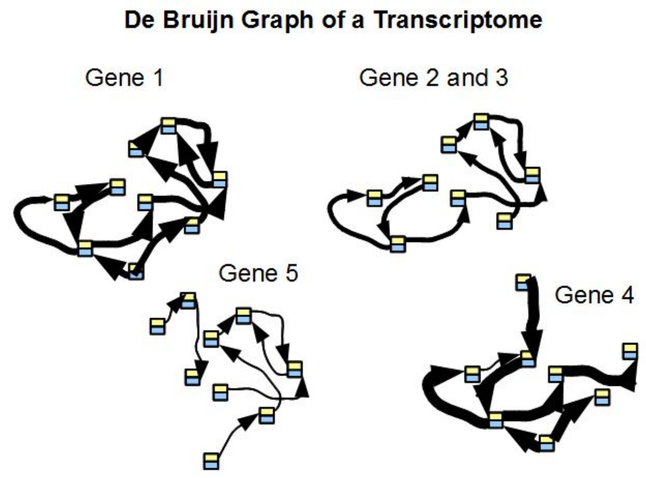
From: https://en.wikipedia.org/wiki/Genomics

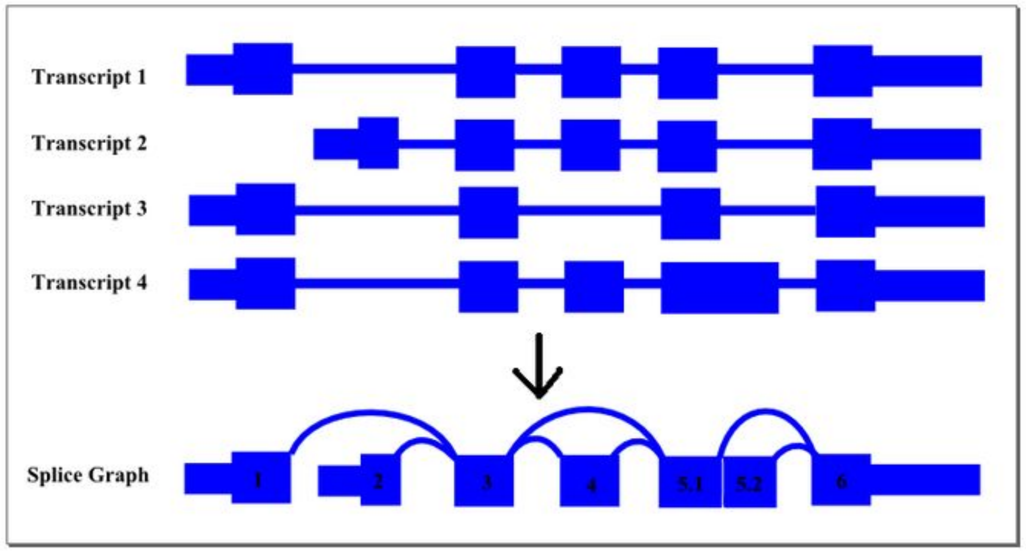


Assembly process (2)

- Mapped reads must first be segmented to represent gene loci.
- An exon graph called a *splicing graph* is then constructed for each locus and task of finding a set of paths is applied within each graph.
- Each path will represent an isoform

=> **Graph topology** which best corresponds to the data.

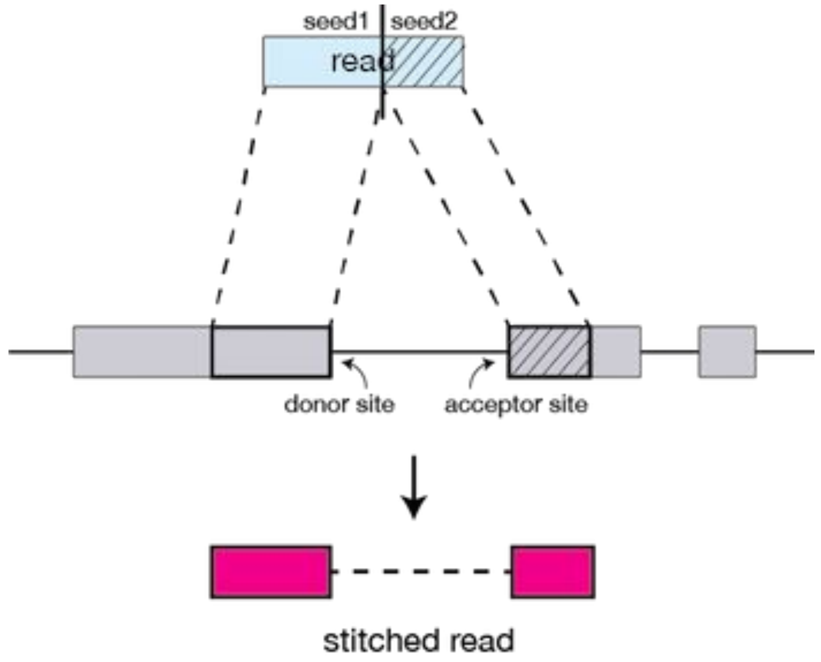




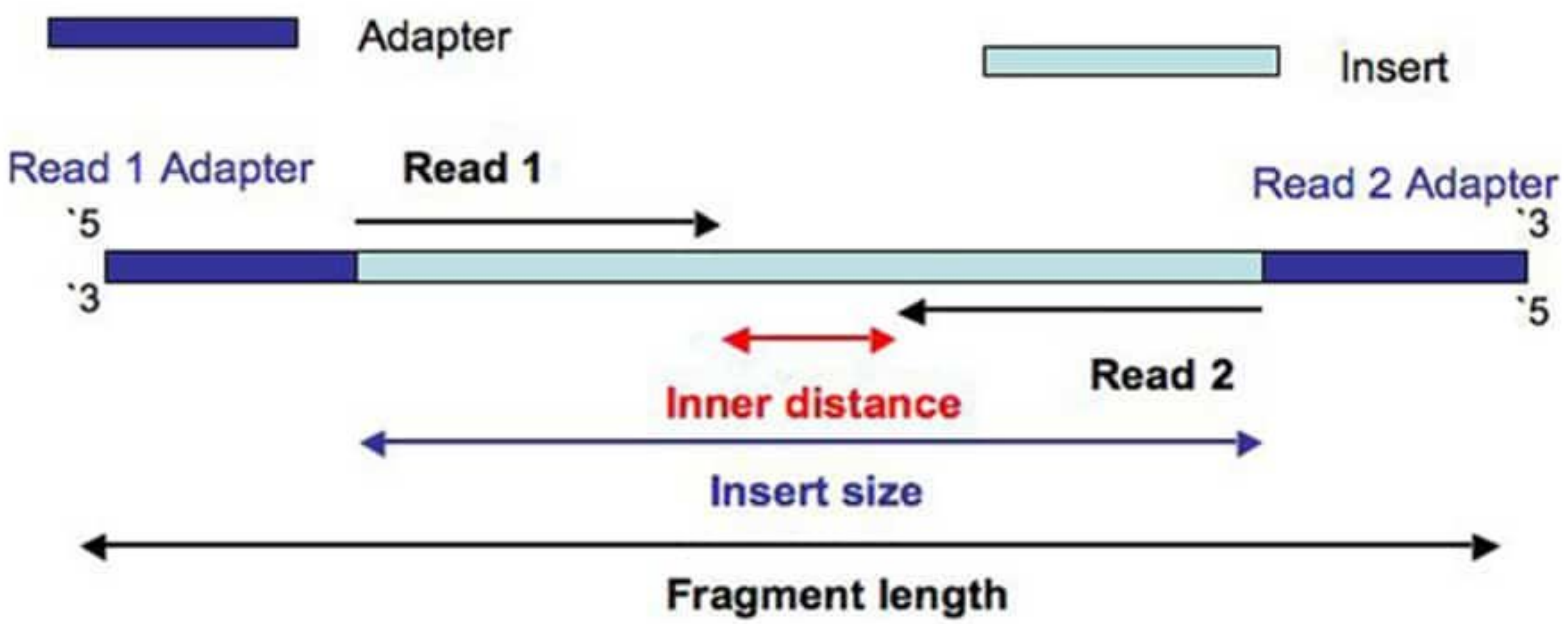
https://bioinformatics.mdanderson.org/public-software/spliceseq/methods/

Assembly process (3)

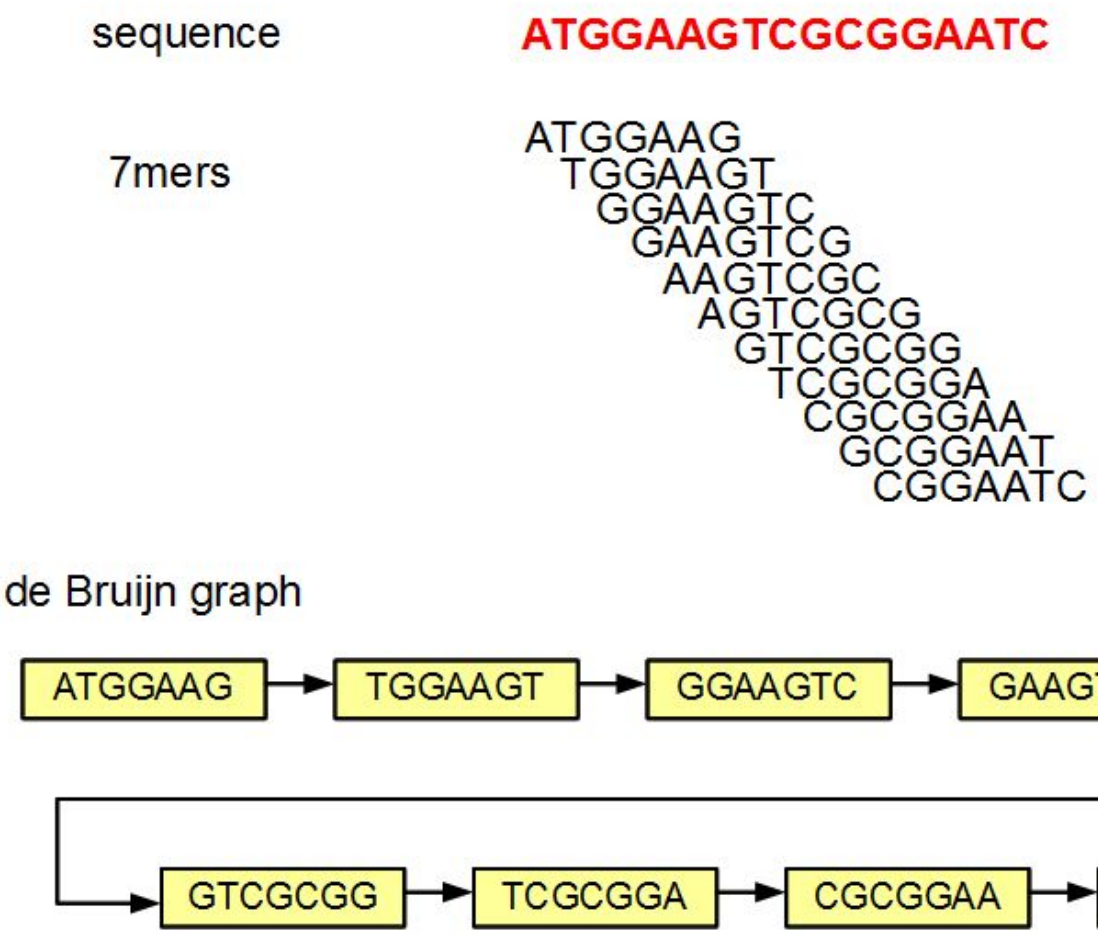
- **Split reads**: If the beginning of the read is mapped to one exon and the end of the read to another exon, this give support for these two exons to be adjacent in a transcript sequence.
- **Paired-end**: one end is mapped to one exon and another end is mapped to another exon.



Mapped read pairs

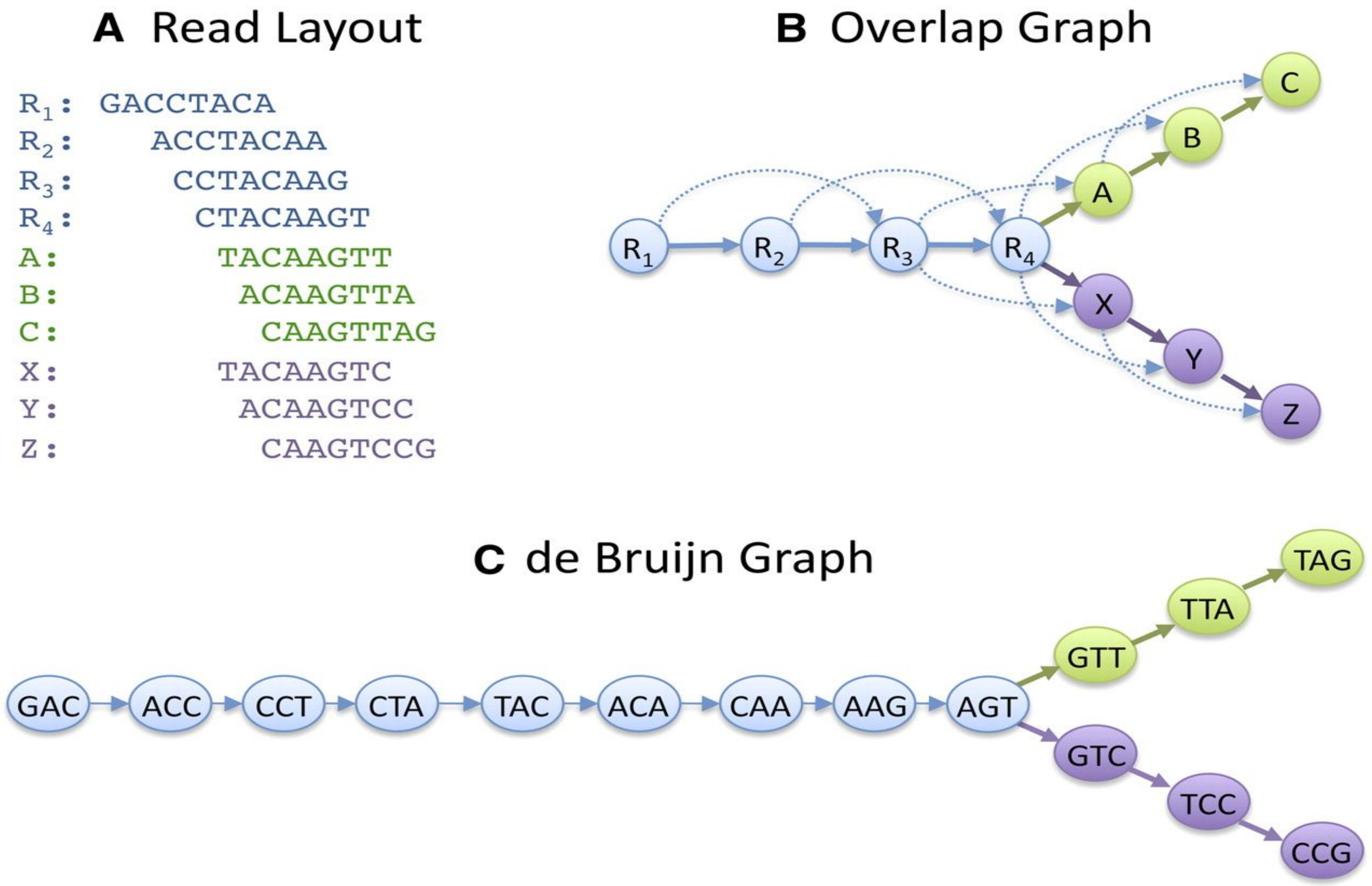


de Bruijn graph Overview



- Each node is associated with a (k-1)-mer. Two nodes A and B are connected if there is a k-mer whose *prefix* is the (k-1)-mer of node A and the *suffix* is the (k-1)-mer of node B.
- Each k-mer is represented in the graph only once as an edge connecting 2 nodes.
- Two sequence reads share an edge if they have common k-mer.

de Bruijn graph vs Overlap Graph



```
R1: GACCTACA
R2:    ACCTACAA
R3:     CCTACAAG
R4:      CTACAAGT
A:        TACAAGTT
B:         ACAAGTTA
C:          CAAGTTAG
X:        TACAAGTC
Y:         ACAAGTCC
Z:          CAAGTCCG
```

https://genome.cshlp.org/content/20/9/1165/F2.expansion.html

de Bruijn graph algorithms

- Construction of a de Bruijn graph is straightforward and much faster compared to calculating the overlaps between all read pairs.
- Extracting all k-mers from reads and connecting the nodes representing the (k-1)-mers.

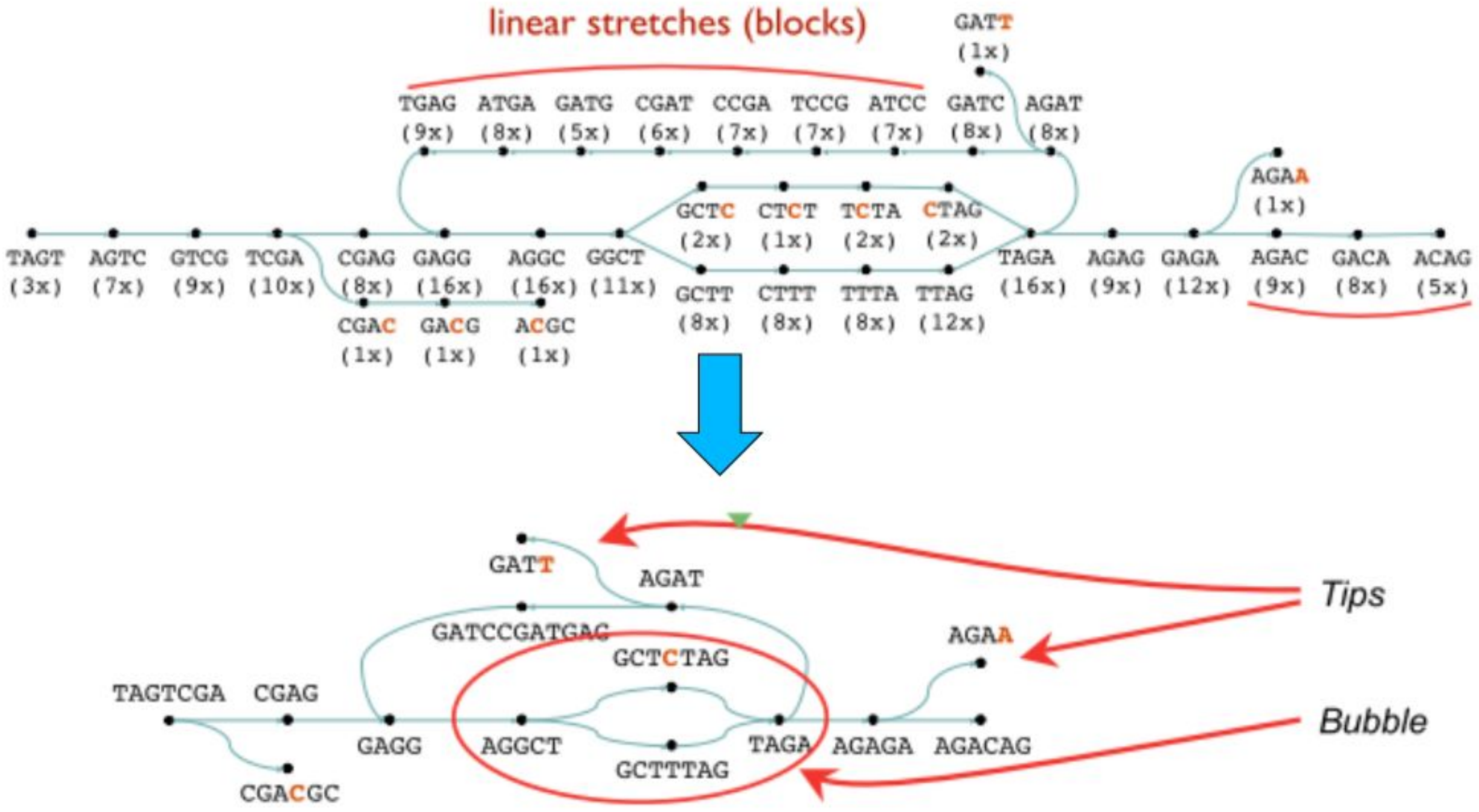
-> The challenge then becomes how to find the paths in a graph which represent true transcripts.

★ Sequencing errors result in:

  - Tips: dead ends in a graph
  - Bubbles: complicate the structure of the graph

*Some bubbles are due to alternative splicing, in the case of an exon in the middle of a gene model which is present in one isoform but skipped in another isoform.

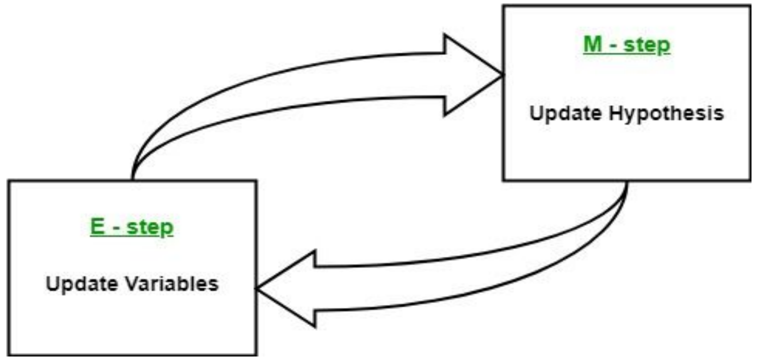
Tips and Bubbles in de Bruijn graph



http://profs.scienze.univr.it/~liptak/FundBA/slides/FragmentAssembly_6up.pdf

Abundance Information

- The abundance should be the same in all exons belonging to the same transcript.
- One transcript is one molecule, if there are no biases in library preparation and sequencing, sequence reads should cover and represent an entire transcript uniformly.
- If there are deviations, for example, some exons have larger sequence depth, it indicates that those exons are also present in other isoforms.
- In order to estimate their relative abundances, the Expectation Maximum (EM) algorithm has been used:
- (E): all reads are assigned proportionally to each isoform according to the isoform abundance.
- (M): relative abundances of isoforms are recalculated.



Data preprocessing

Trimming approach method

- Typically, base call quality diminishes toward the end of a read.
- This is characteristic to first and second generation sequencing technologies.
- The low-quality part of a read with more errors reduces the alignment score. By trimming the low-quality part of the read, the number of mappable reads can be increased.

De novo assembly preprocessing

- Pairwise read overlaps algorithm is beneficial to trim low-quality parts of reads.
- In de Bruijn graph-based methods, erroneous tails of reads result in tips and and dead ends in the graph. Furthermore, the low-quality end of read does not affect the k-mers in the beginning of the read -> Trimming of reads simplifies the graph and reduces number of dead ends.

Artifacts related to library construction

- Those artifacts include ***adaptor sequences*** which might be remaining in a portion of the sequence reads.
- Also, if ***polyA*** is included in sequencing, it should be trimmed off.
- User should know how the *sequencing library* was constructed and how the reads are oriented.
- *Strand specificity* of the library is needed to be taken into consideration because it gives an advantage to resolve overlapping genes which are in opposite strands.

Read error correction method (1)

- A completely different idea is to try to correct the errors in the reads without filtering and trimming.
- Its main applications of read correction is utilized for de novo assembly.
- Sequencing errors result in a number of incorrect k-mers and produce useless nodes :
    - slow down the computation
    - increase memory usage
- The error correction software purposely dedicated to RNA-seq data is SEECER

http://sb.cs.cmu.edu/seecer/

Read error correction method (2)

- However, not all variation in the data are random due to sequencing errors. There are also non-random variation due to differences between alleles in diploid and polyploid organisms.
- Read correction is based on the redundancy in the data (*sequencing depth*).
- The challenge comes when there is *no reference* available and there are *similar sequences* originating from different parts of genome due to *repeats* or *similar regions*.

Mapping-based assembly

Introduction

What is mapping-based assembly in RNA-seq ?

- Reconstruct full-length transcript sequences based on RNA-seq read mapping.
- Cufflinks and Scripture are two of many software packages that can be used for *ab initio* reconstruction of transcripts, that is, there is no need to have external gene models.
- Cufflinks reports the smallest possible set of isoforms which can explain the data.
- Output is given in BED or GTF format which contains the transcript coordinates in a reference sequence

Overview

- Mapping process can be done using Bowtie2 + Tophat2 or HISAT2 in order to index and generate raw bam file using fasta reference genome file.
- Input will be paired-end reads in fastq format.
- In this practical example, Cufflinks tool will be demonstrated but not obligated for mapping-based assembly in transcriptome.

HISAT2

- HISAT2 is a fast and sensitive alignment program for mapping next-generation sequencing reads (both DNA and RNA) to a population of human genomes as well as to a single reference genome.

1. Building an index.
2. Performing alignment with HISAT2.

http://daehwankimlab.github.io/hisat2/ *https://github.com/DaehwanKimLab/hisat2* cole-trapnell-lab/**cufflinks**

15 Contributors

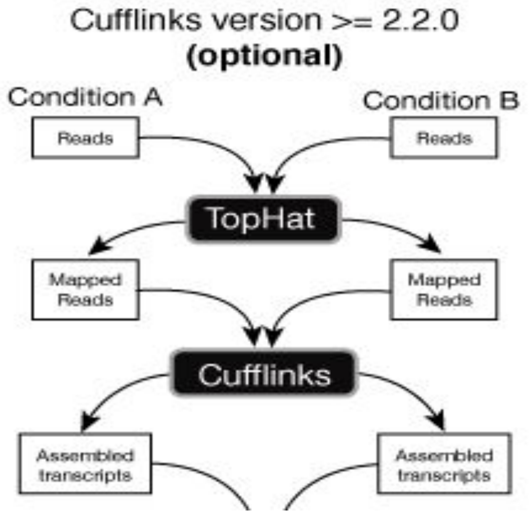
81 Issues

284 Stars

117 Forks

Cufflinks

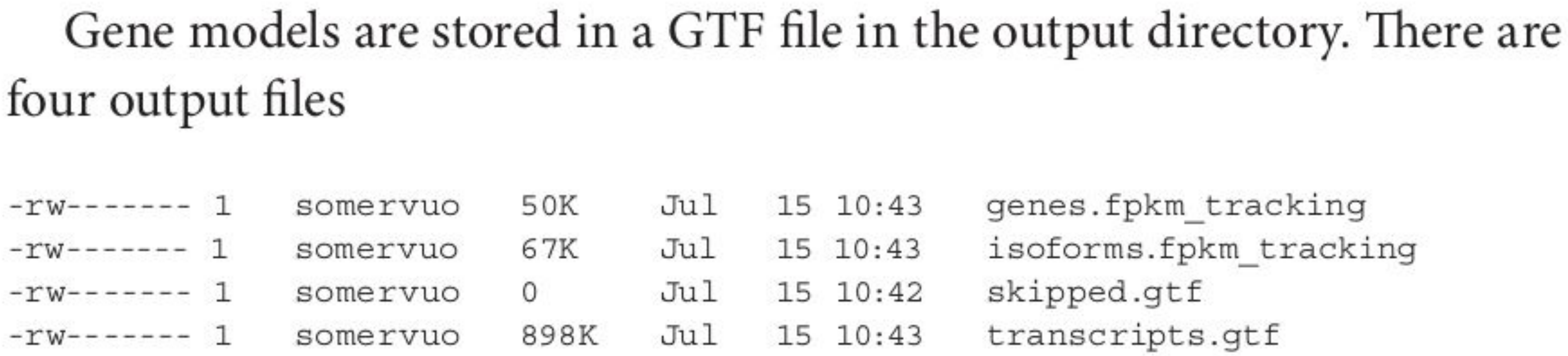
- Cufflinks is written in C++.
- This tool assembles transcripts and estimates their abundances in RNA-Seq samples.
- It accepts aligned RNA-Seq reads and assembles the alignments into a parsimonious set of transcripts.
- Cufflinks then estimates the relative abundances of these transcripts based on how many reads support taking into account biases in protocols



http://cole-trapnell-lab.github.io/cufflinks/ https://github.com/cole-trapnell-lab/cufflinks

Cufflinks output

- In order for Cufflinks to utilize paired-end information, both reads of the read pair should have the same identifier in the BAM file.
- Samtools is also required.
- If there is existing knowledge of gene models, it can be utilized by giving a GTF file as a guidance to Cufflinks using the argument “–g.”

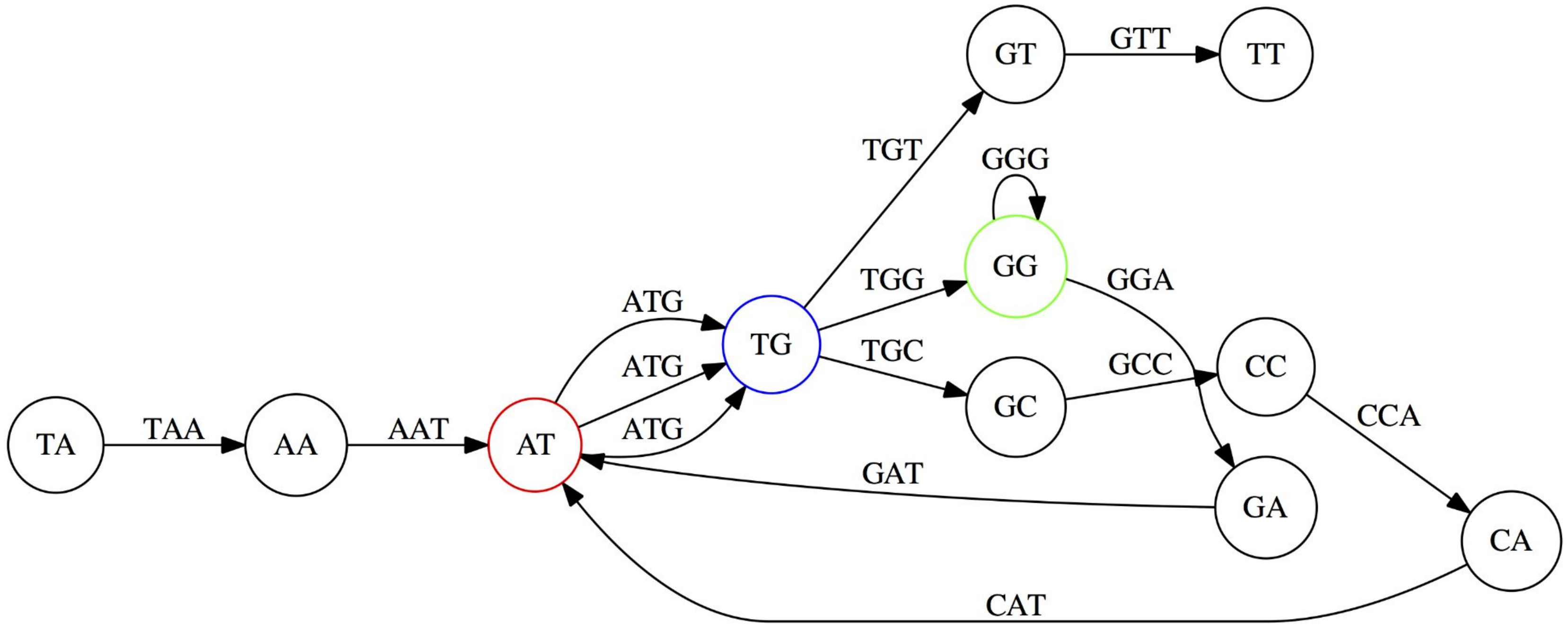


Gene models are stored in a GTF file in the output directory. There are four output files

```
-rw------- 1   somervuo   50K    Jul   15 10:43   genes.fpkm_tracking
-rw------- 1   somervuo   67K    Jul   15 10:43   isoforms.fpkm_tracking
-rw------- 1   somervuo   0      Jul   15 10:42   skipped.gtf
-rw------- 1   somervuo   898K   Jul   15 10:43   transcripts.gtf
```

De novo assembly

Introduction



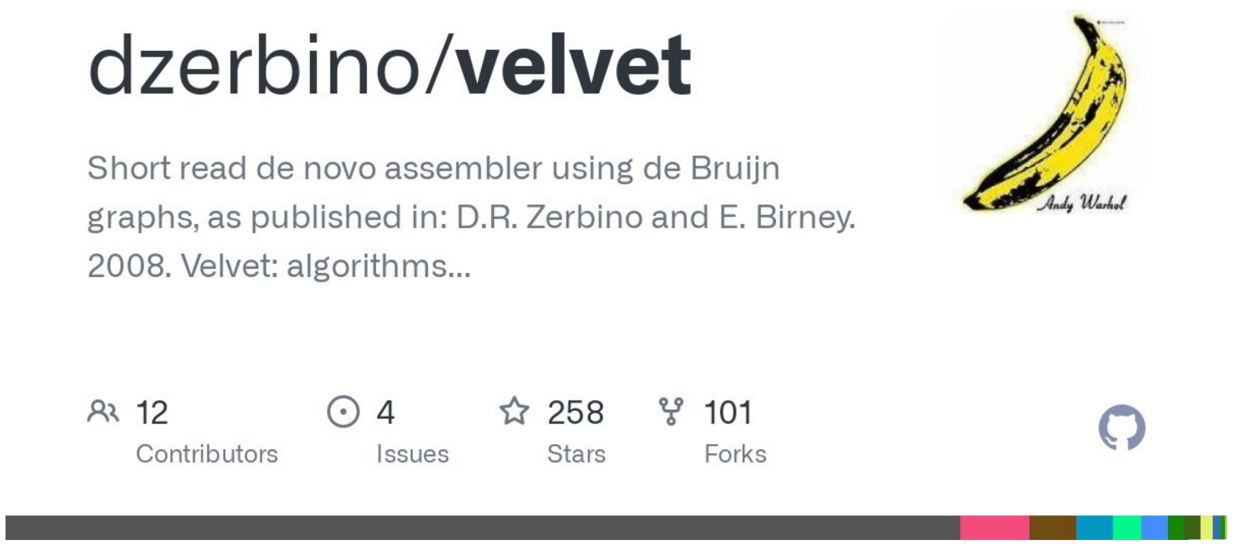
What is de novo assembly in transcriptomic ?

- Reconstruct full length transcript sequences *de novo,* without the help of reference genome.
- In this section, two software packages are introduced in order to perform de novo assembly practice based on your interest.
  1. Velvet + Oases
  2. Trinity
- Both tools utilized de Bruijn graphs.

Overview

- **Velvet** is a genome assembler which produces one assembly graph that is used by second program **Oases** to find paths which represent isoforms.
- The other assembly program is **Trinity** which consists of three modules.
   - First, the RNA-seq reads are initially assembled and clustered, each cluster representing a locus in a genome.
   - A de Bruijn graph is constructed for each cluster.
   - Linear transcript sequences are extracted so that there can be several isoforms from the same locus.

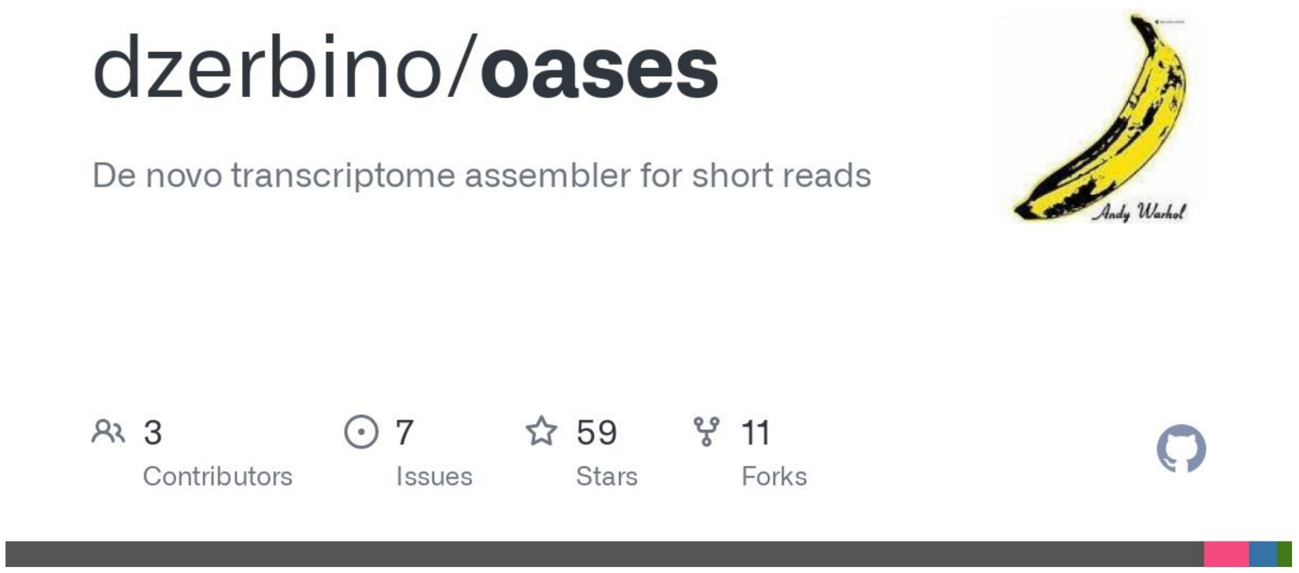
https://github.com/dzerbino/velvet
https://github.com/dzerbino/oases


dzerbino/**velvet**

Short read de novo assembler using de Bruijn graphs, as published in: D.R. Zerbino and E. Birney. 2008. Velvet: algorithms...

12 Contributors · 4 Issues · 258 Stars · 101 Forks

https://github.com/dzerbino/velvet

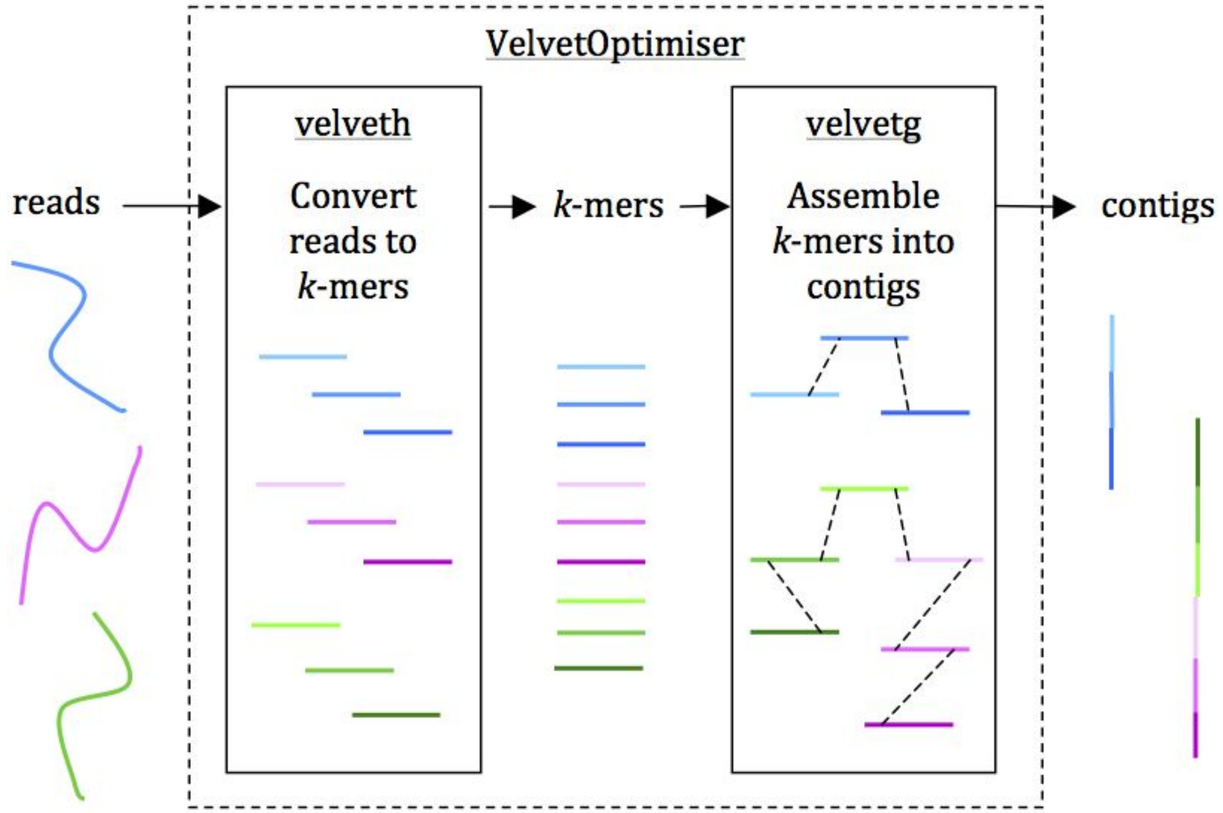

dzerbino/**oases**

De novo transcriptome assembler for short reads

3 Contributors · 7 Issues · 59 Stars · 11 Forks

https://github.com/dzerbino/oases

Velvet algorithm

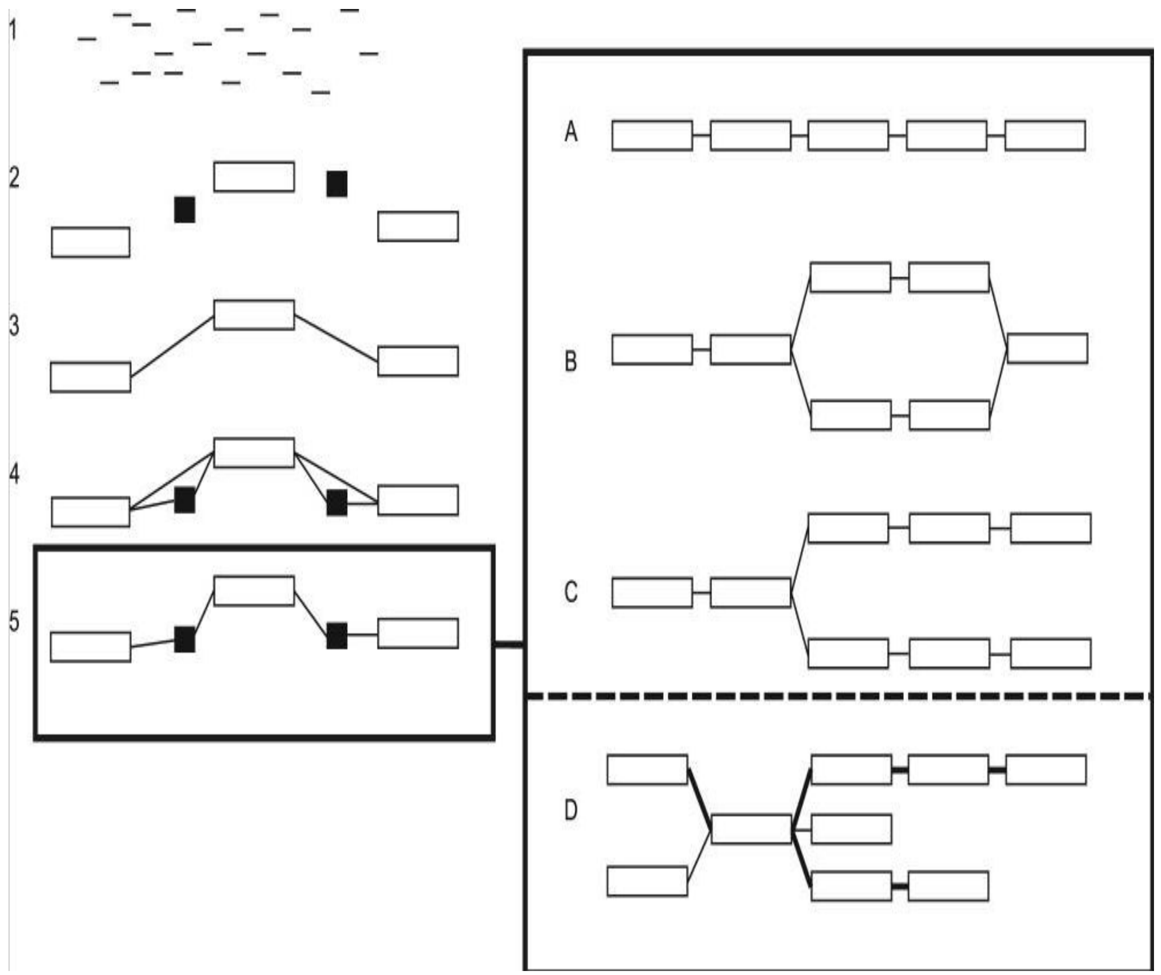


Velvet consists of 2 programs:

- **velveth**: calculates k-mers of data
- **velvetg**: finds and extracts contigs

  in a de Bruijn graph.

Oases



(1): Individual reads are sequenced from an RNA sample

(2): Contigs are built from those reads, some of them are labeled as long (clear), others short (dark)

(3): Long contigs, connected by single reads or read-pairs are grouped into connected components called loci

(4): Short contigs are attached to the loci

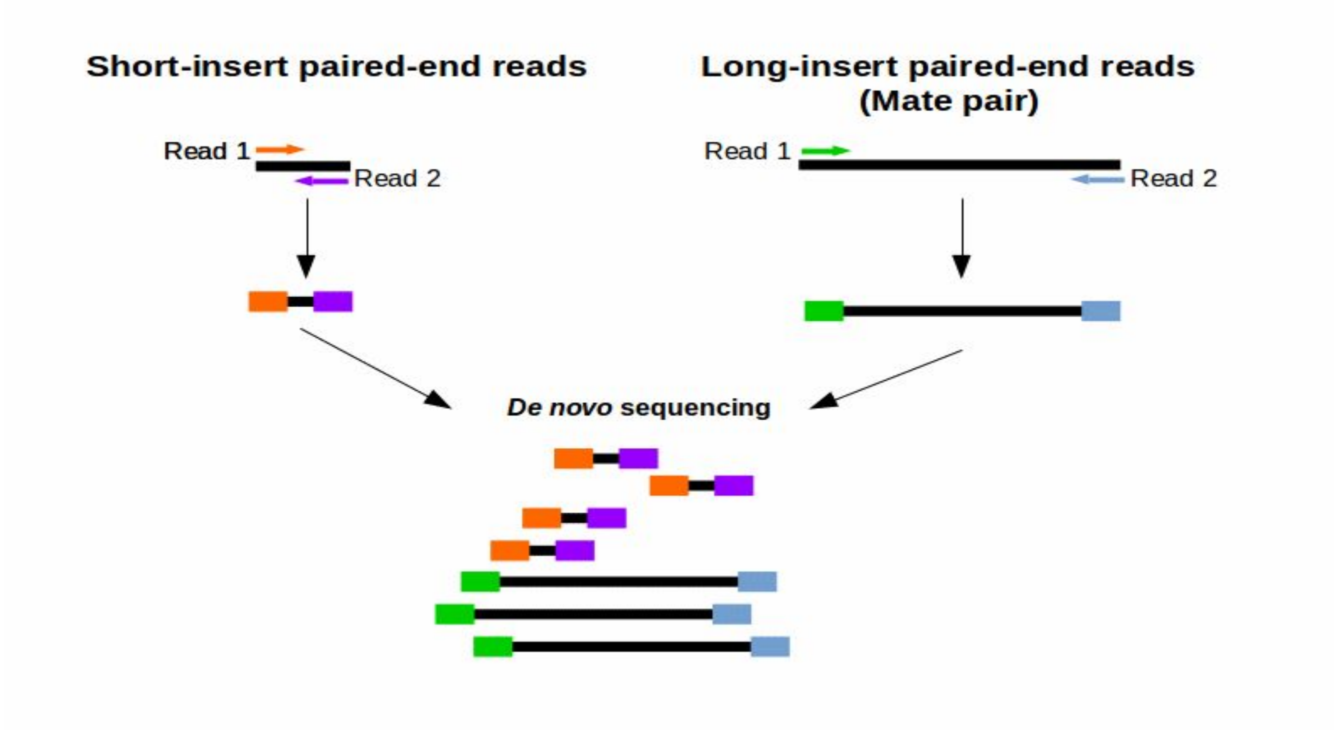
(5): The loci are transitively reduced. Tranfrags are then extracted from the loci.

- The loci are divided into four categories:
  - (A) chains
  - (B) bubbles
  - (C) forks
  - (D) complex (i.e. all the loci which did not fit into the previous categories).

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3324515/

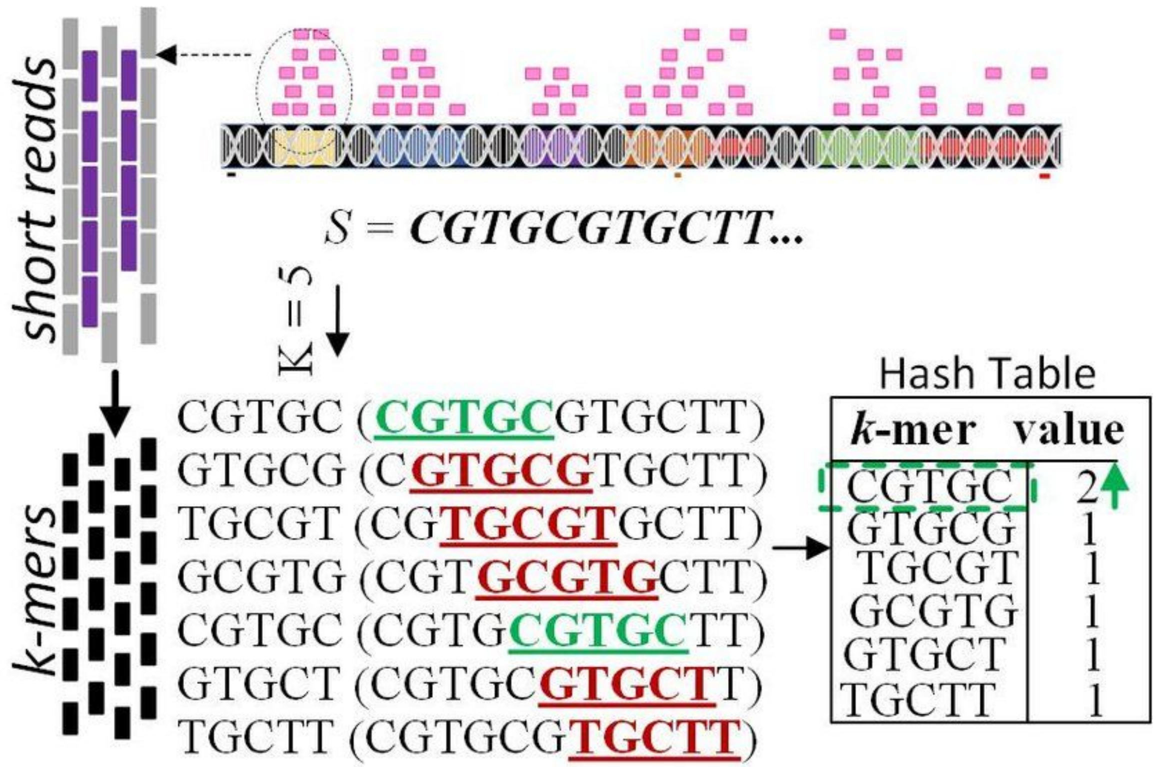
Interleave paired-end reads

- Requirement for paired-end reads to be used in velvet is that they must be **interleaved**.
- Both reads of a read pair are located adjacently in the same file.

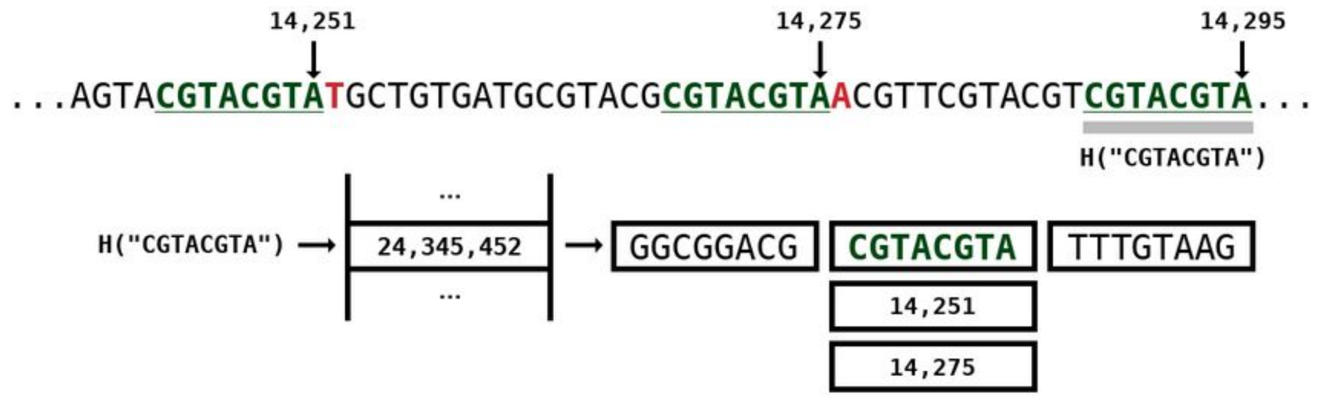

Short-insert paired-end reads
Long-insert paired-end reads (Mate pair)
Read 1
Read 2
Read 1
Read 2
De novo sequencing

https://www.ecseq.com/support/ngs/what-is-mate-pair-sequencing-useful-for

Hash table



k-mer	value
CGTGC	2
GTGCG	1
TGCGT	1
GCGTG	1
GTGCT	1
TGCTT	1



https://www.researchgate.net/figure/The-hash-table-generation-out-of-k-mers_fig2_343689618

Velvet process

- *Velveth* defines k-mer length, set up the output directory and specify the data format.
- Next, graph traversal and contig extraction is done in the second step using *velvet*g. The insert size can be manually chosen as it is the fragment length that includes the read lengths.

Oases process

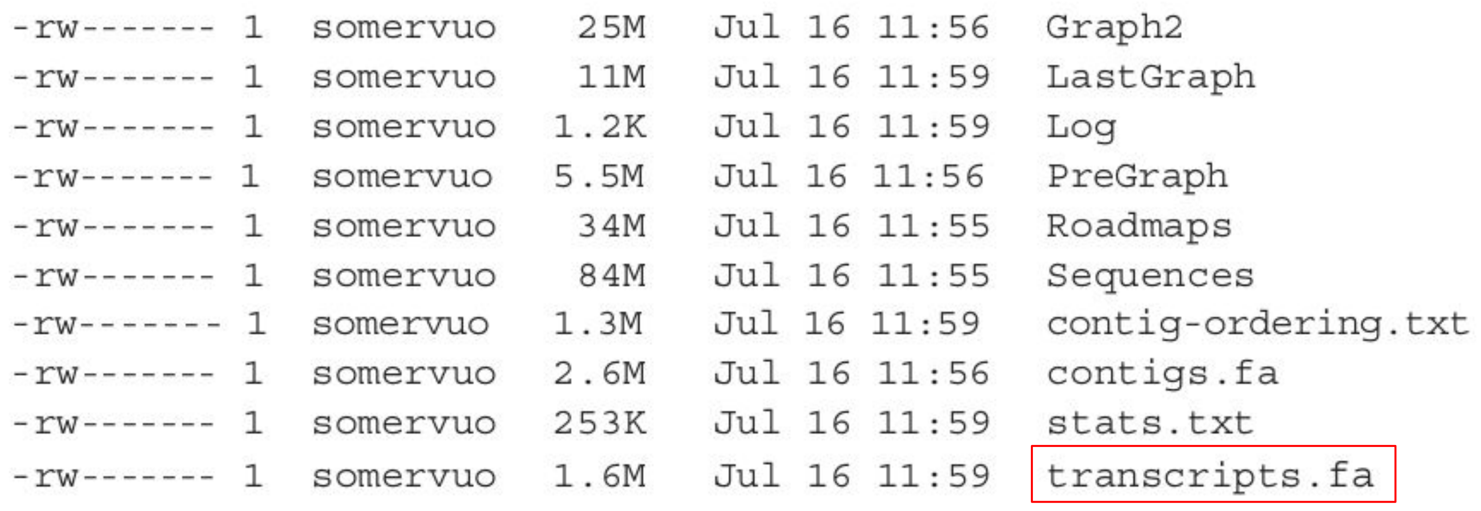
- Oases is applied to the resulting de Bruijn graph.
- Oases will accept the velvet output as its input.
- In the case of paired-end reads, insert size must also be defined.
- Both insert lengths and minimum transcript lengths are able to be defined by user.

Output

- FASTA entry name example in “transcript.fa” file:

“*Locus_10_Transcript_1/3_Confidence_0.571_Length_3815*”

- There are 3 transcripts from locus 10.
- Confidence value is between 0 and 1.
- Length of the transcript.



```
-rw------- 1  somervuo   25M  Jul 16 11:56  Graph2
-rw------- 1  somervuo   11M  Jul 16 11:59  LastGraph
-rw------- 1  somervuo  1.2K  Jul 16 11:59  Log
-rw------- 1  somervuo  5.5M  Jul 16 11:56  PreGraph
-rw------- 1  somervuo   34M  Jul 16 11:55  Roadmaps
-rw------- 1  somervuo   84M  Jul 16 11:55  Sequences
-rw------- 1  somervuo  1.3M  Jul 16 11:59  contig-ordering.txt
-rw------- 1  somervuo  2.6M  Jul 16 11:56  contigs.fa
-rw------- 1  somervuo  253K  Jul 16 11:59  stats.txt
-rw------- 1  somervuo  1.6M  Jul 16 11:59  transcripts.fa
```

Assembly with different k-mers combined

- Oases version 2.0 is possible to run several assemblies with variety of different k-mers and merge the assemblies.
- k-mers must be set in odd number from -m (minimum) to -M (maximum).
- This will output for each k-mer and one directory contains the results of a merged assembly.

Ex:

```
python oases_pipeline.py -m 19 -M 29 -o odir -d “ -fastq -shortPaired chr18_12.fq” -p “ -ins_length
200 –min_trans_lgth 200”
```

Comparison

- ➔ In order to compare the longest transcript against the known transcripts, it can be mapped to the appropriate human genome using the UCSC genome browser BLAT software.

Summary

- Beside the data and preprocessing, output of an assembly depends on the parameter setting particular to each software.
- However, the plain number and length of transcripts reveal nothing about the accuracy and errors of assembly.
- Measuring the quality of an assembly is not a straightforward task.
- It is difficult because there are no references or previously known gene models.
- Both mapping and de novo-based assembly methods are able to reconstruct full-length transcripts from short sequence reads, but it still depend on the data quality and coverage.

Take home messages

- It is good to keep in mind that the quality of assembly consists of the combination of *data* and *computational methods*.
- Since sequencing technology only converts the content of an RNA-seq library into a digital form, *library preparation* is a key element in obtaining good quality data.
- Garbage in–garbage out applies to both sequencing and assembly. *Quality control* of data should be done before any assembly.

Thank You