

# TRANSCRIPTOMICS

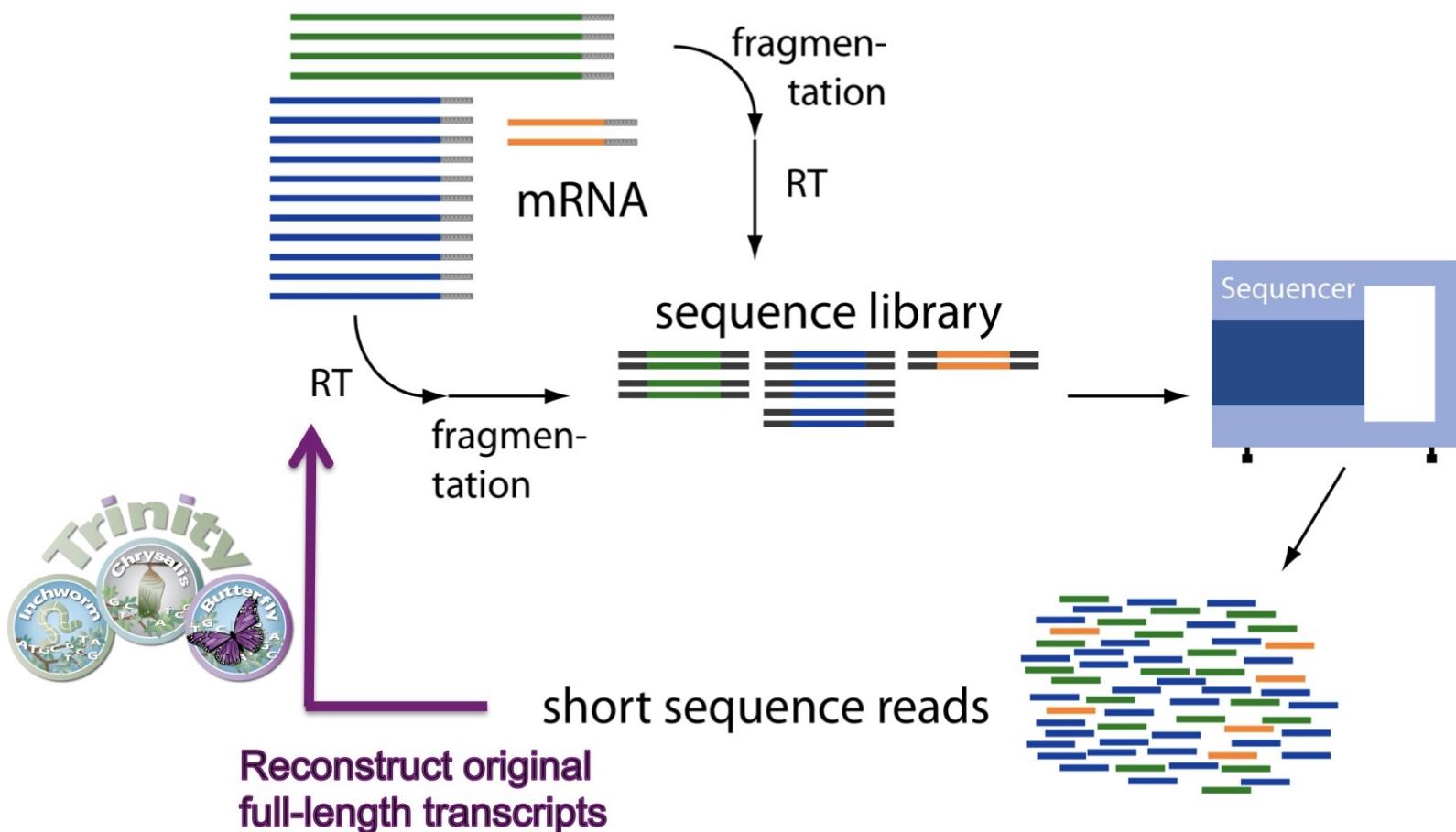
## Assembly with Trinity Part II

Day 03

<https://totorres.github.io/transcriptomics/>

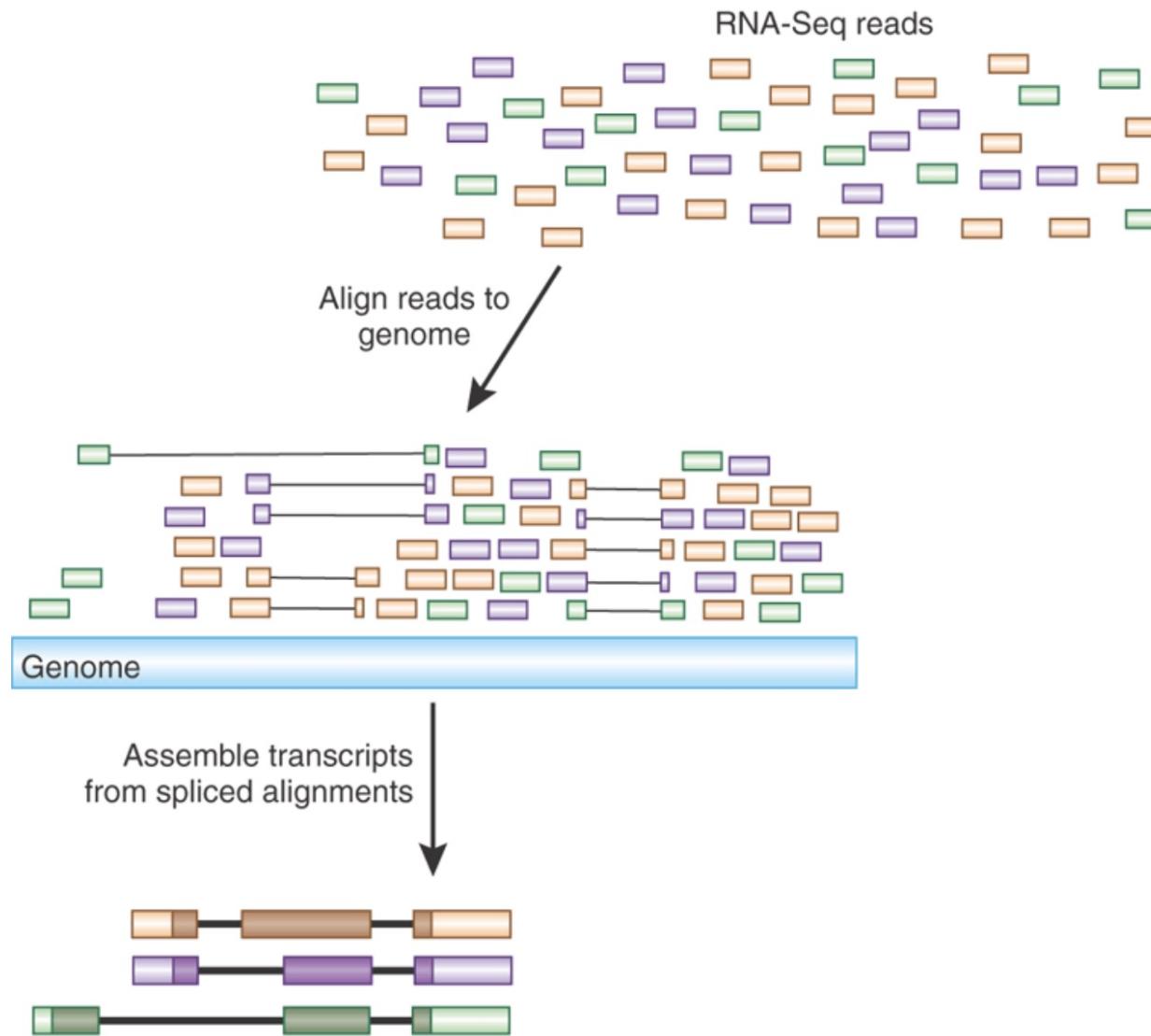
# Assembly of RNA-seq reads

## Overview



# Assembly of RNA-seq reads

## Reference genome



# Assembly by overlap

## Usual steps

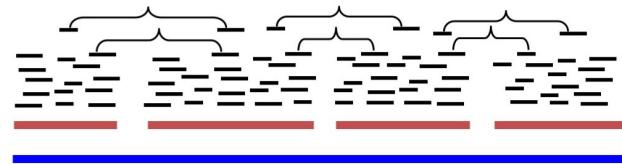
1. Finding overlaps between read pairs



2. Combining overlapping reads into contigs



3. Joining contigs to form scaffolds



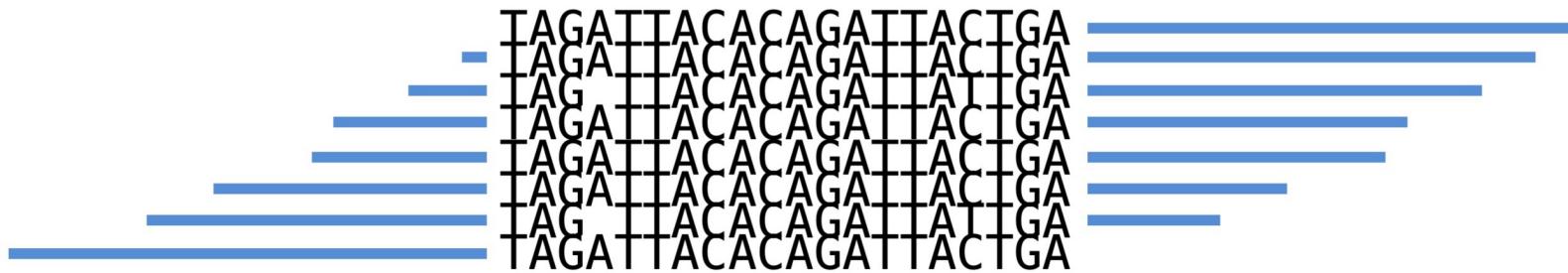
4. Determining the consensus sequence

..ACGATTACAATAGGTT..

# Assembly by overlap

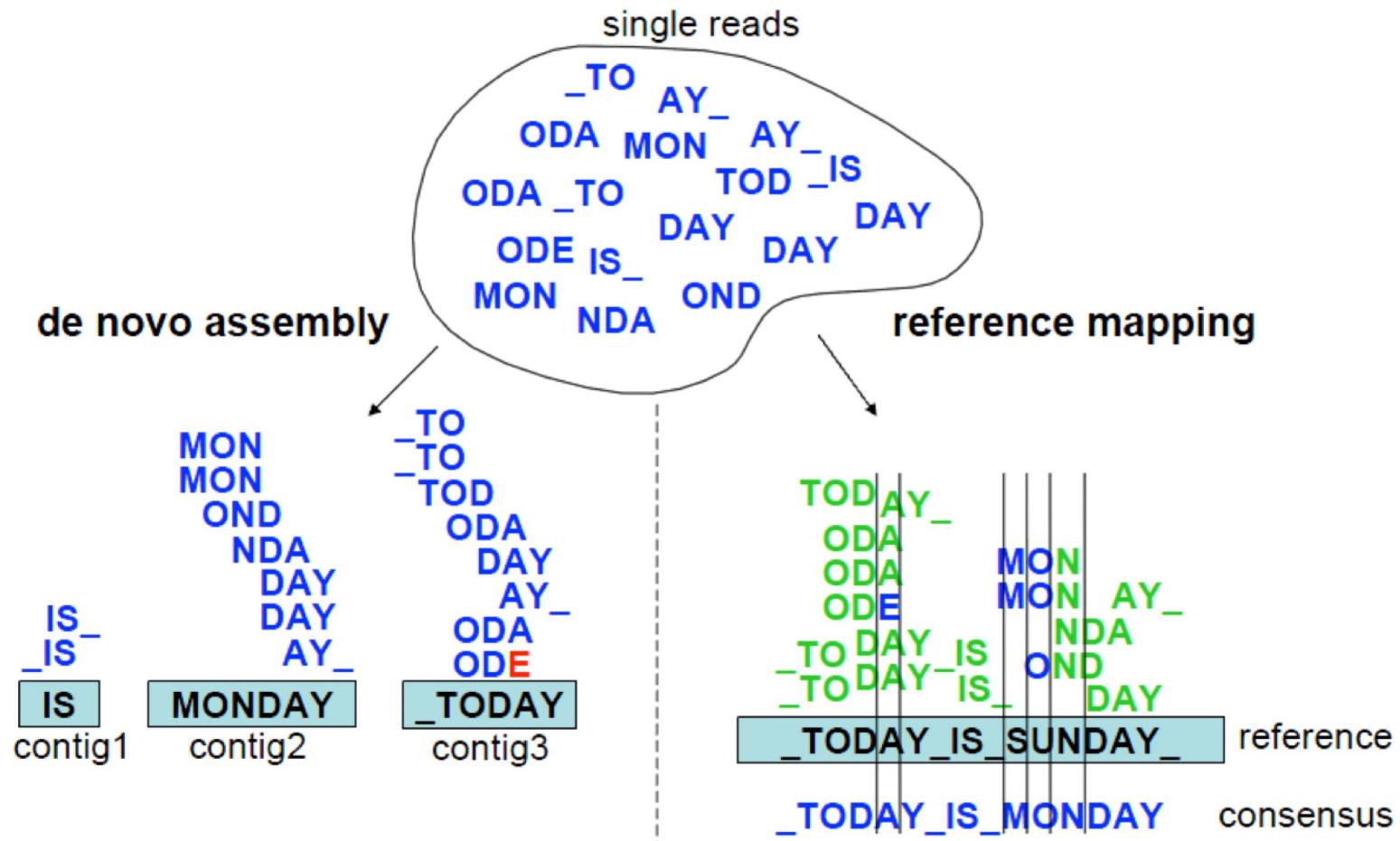
# Usual steps

# Creating multiple local alignments



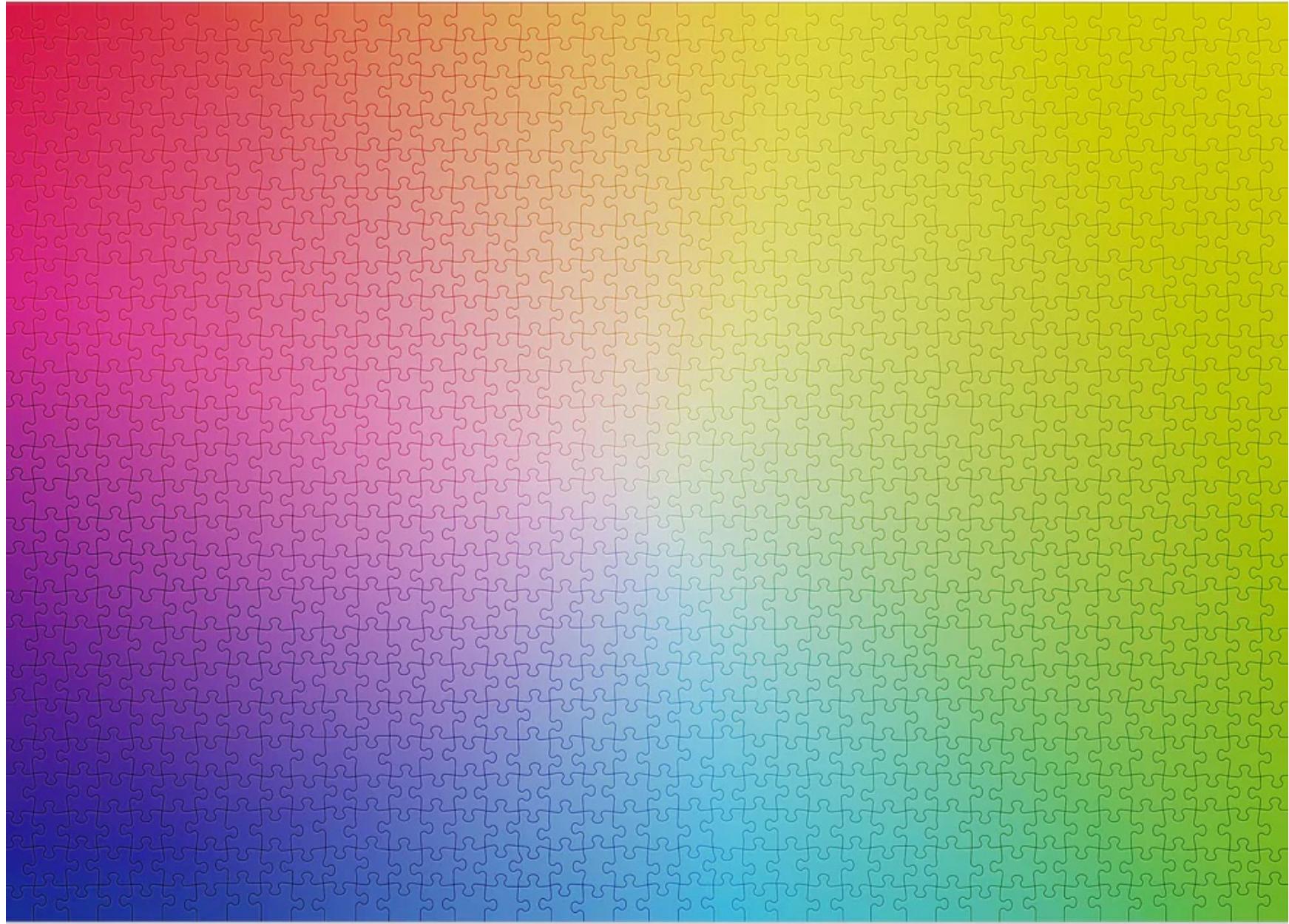
# Assembly by overlap

## Usual steps



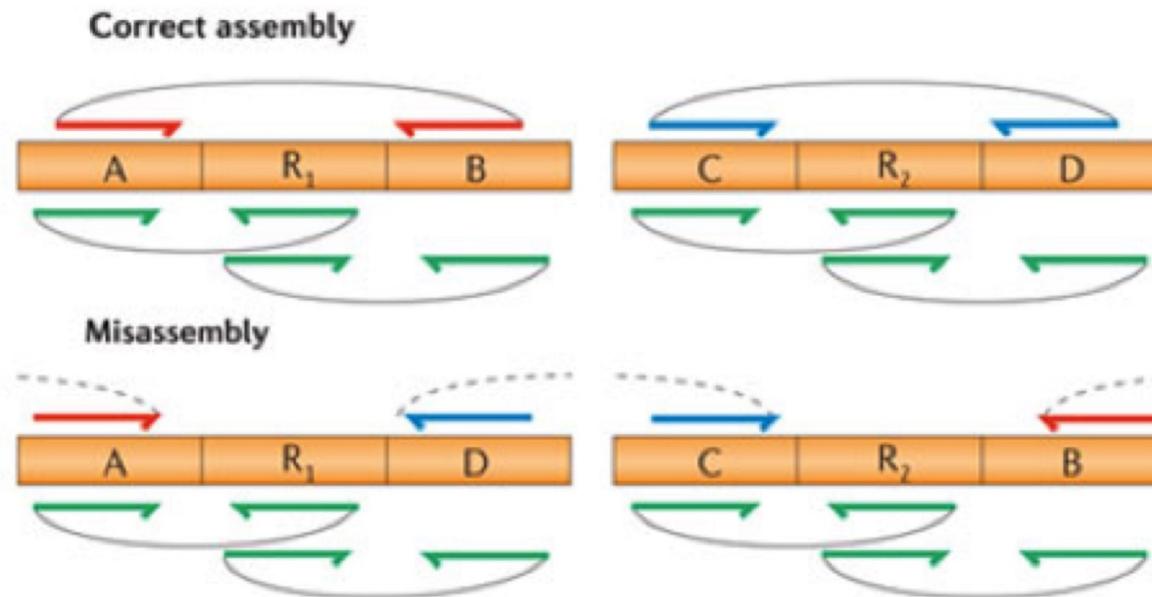
## Assembly by overlap

How do you assemble a puzzle with identical pieces?



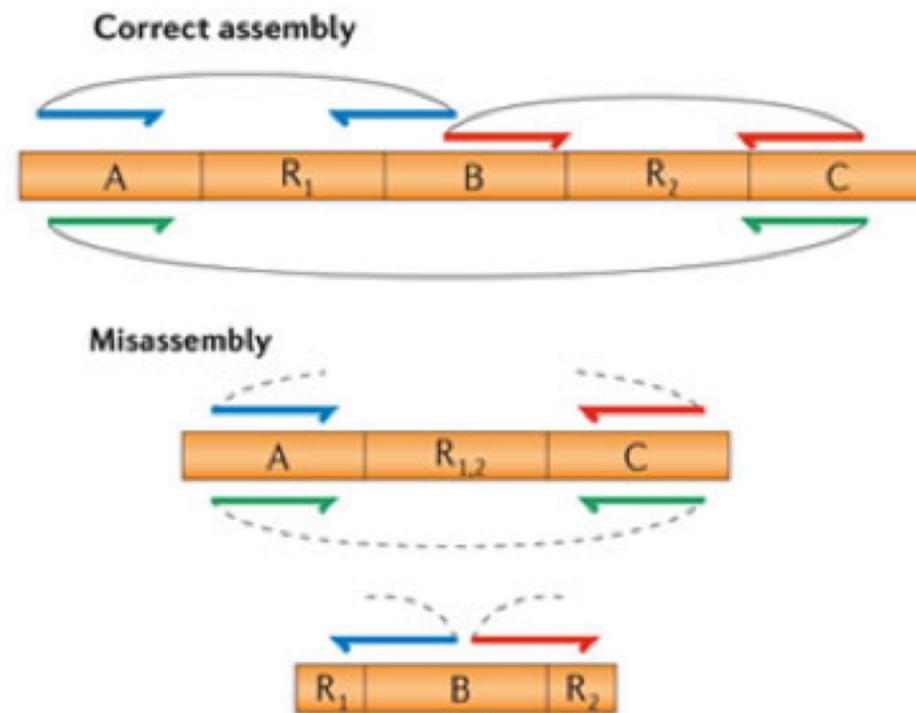
# Assembly by overlap

## Assembly challenges



# Assembly by overlap

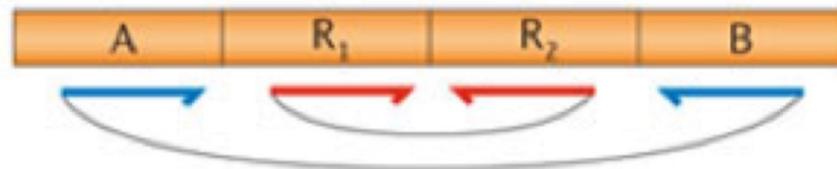
## Assembly challenges



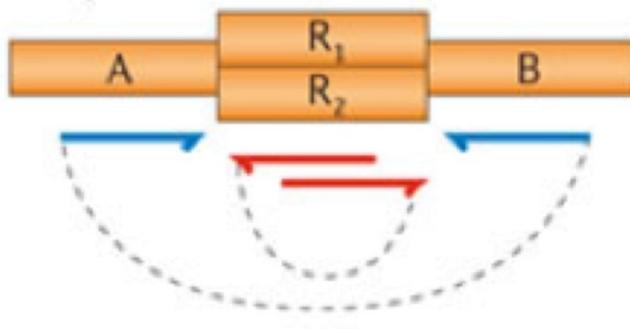
# Assembly by overlap

## Assembly challenges

Correct assembly

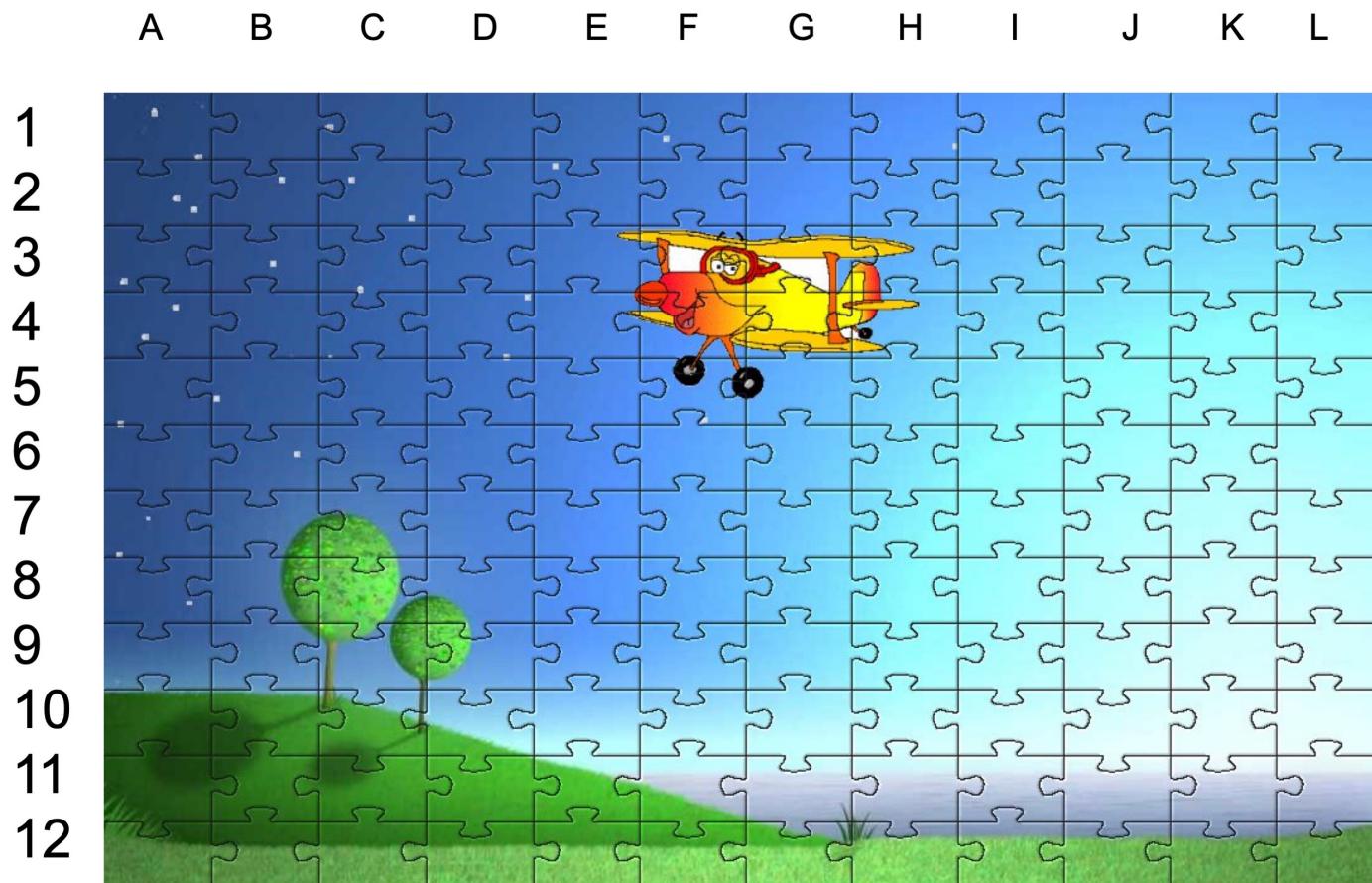


Misassembly



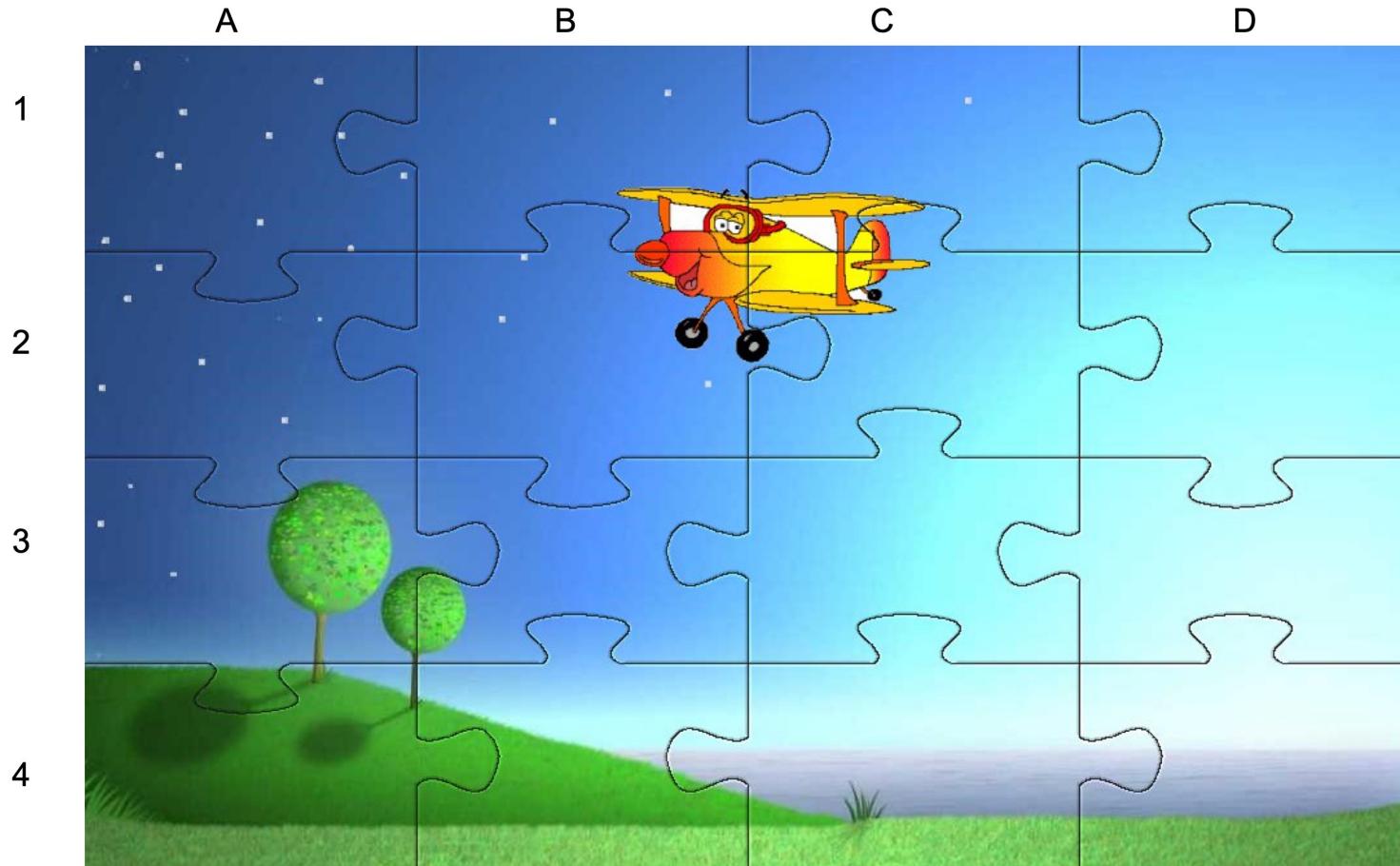
# Assembly by overlap

## Assembly challenges



# Assembly by overlap

## Assembly challenges



# Assembly by overlap

## Assembly challenges



# Assembly challenges

Millions of really short sequences



MiniSeq



MiSeq



NextSeq



NovaSeq

---

1-25 million reads  
150 - 250bp paired-end  
24 hr run time

0.4 - 1.1 billion reads  
150bp paired-end  
24 hr run time

0.6 - 10 billion reads  
50-250bp paired-end  
5 day run time

# Assembly challenges

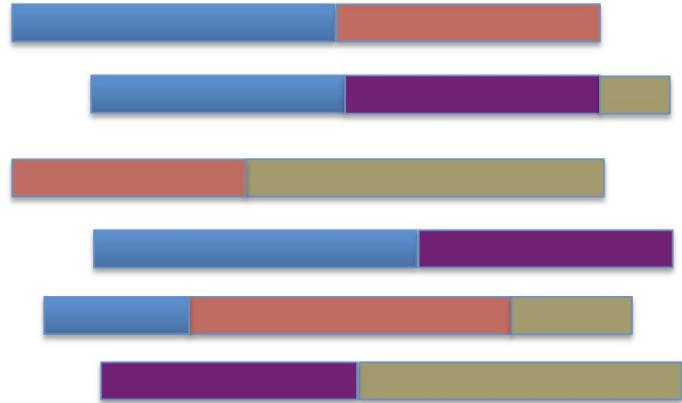
Millions of really short sequences



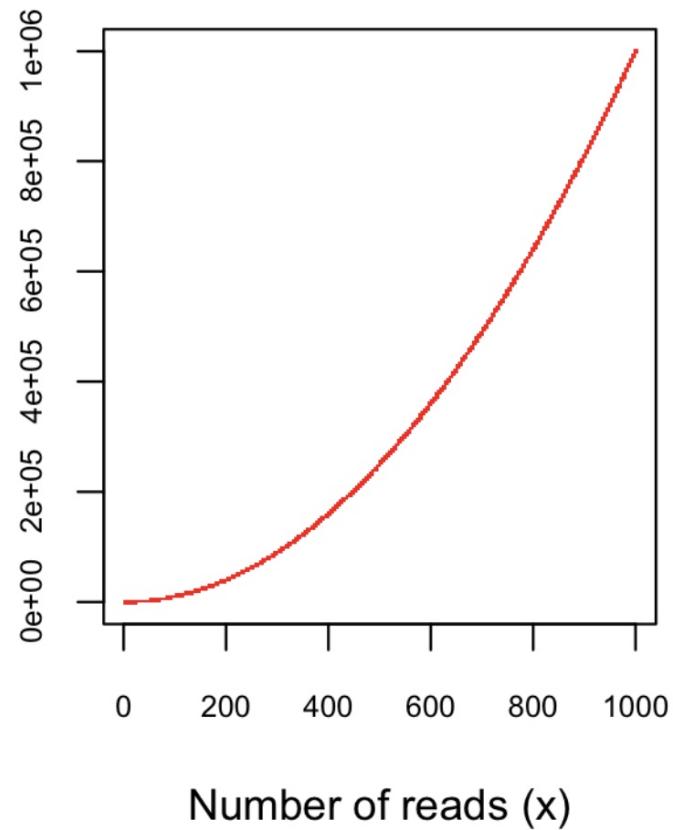
Node = read  
Edge = overlap

# Assembly challenges

Millions of really short sequences



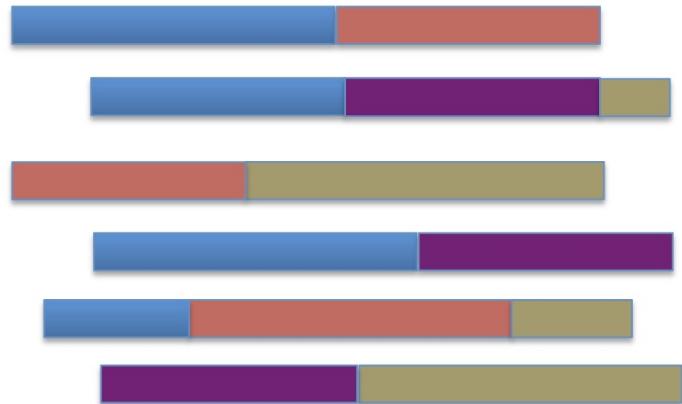
Number of alignments ( $\times 10^2$ )



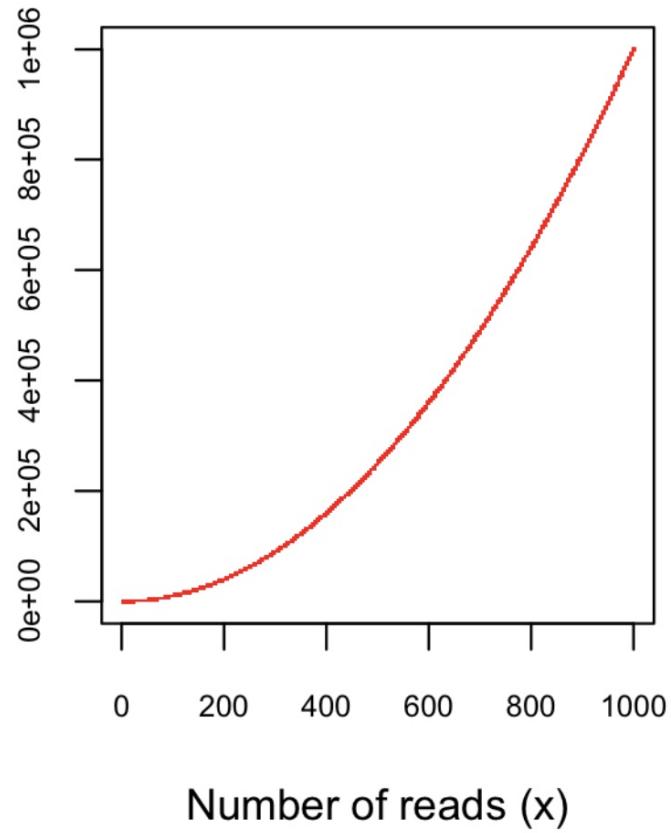
Finding pairwise overlaps between  $n$  reads involves  $\sim n^2$  comparisons

# Assembly challenges

Millions of really short sequences



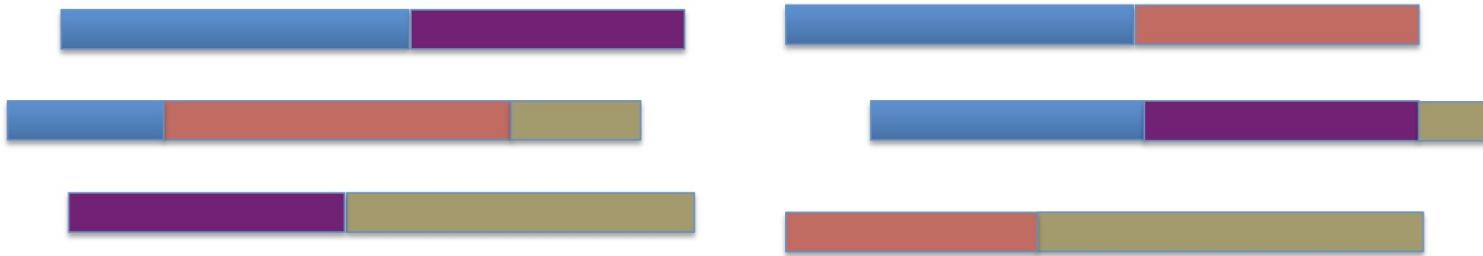
Number of alignments ( $x^2$ )



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

# Assembly challenges

Millions of really short sequences



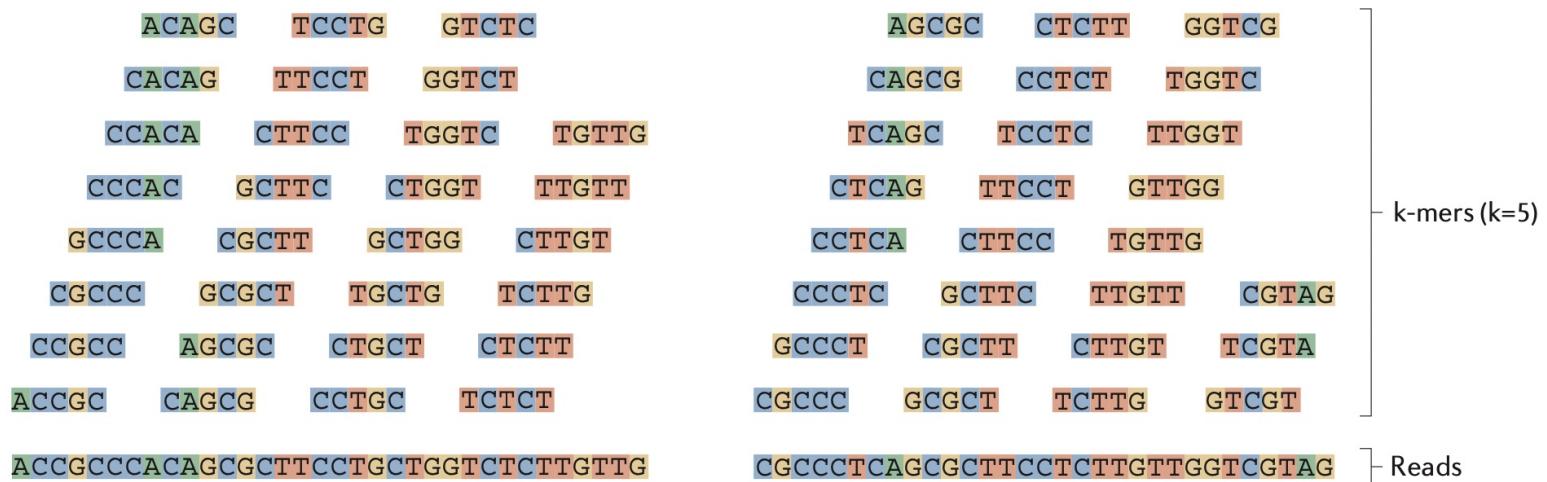
Want to avoid  $n^2$  read alignments to define overlaps

Use a de Bruijn graph

# *De novo* transcriptome assembly

## Assembly strategy

Generate all substrings of length k from the reads

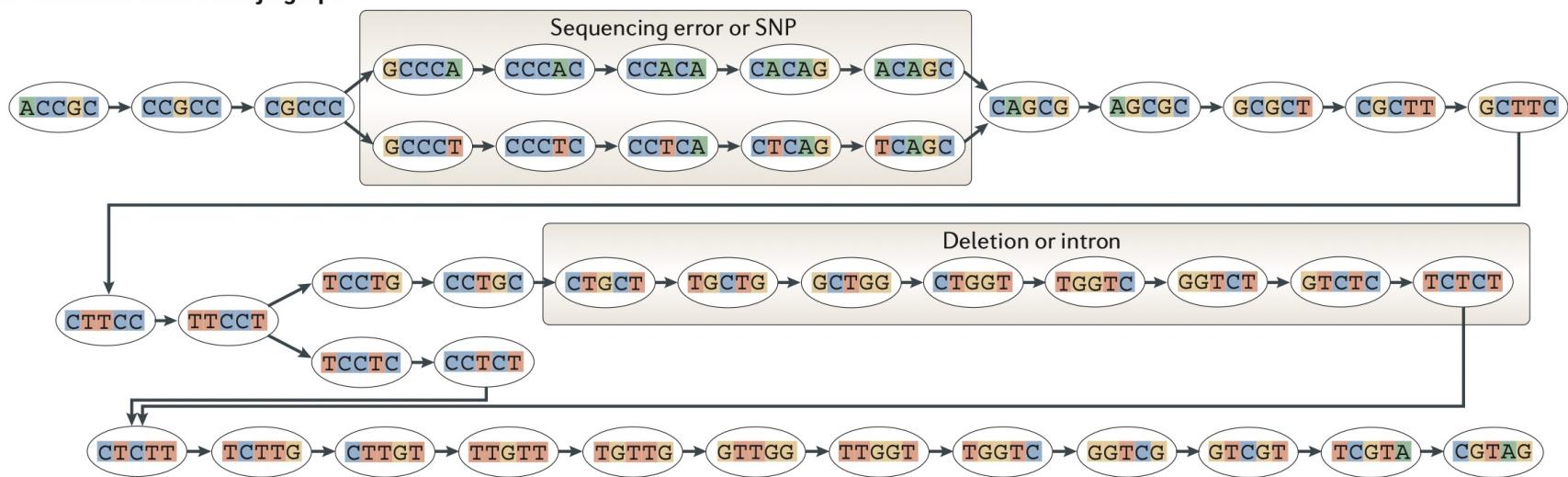


Martin, Wang (2011) Nat Rev Genet, 12, 671-682.

# *De novo* transcriptome assembly

## Assembly strategy

### b Generate the De Bruijn graph

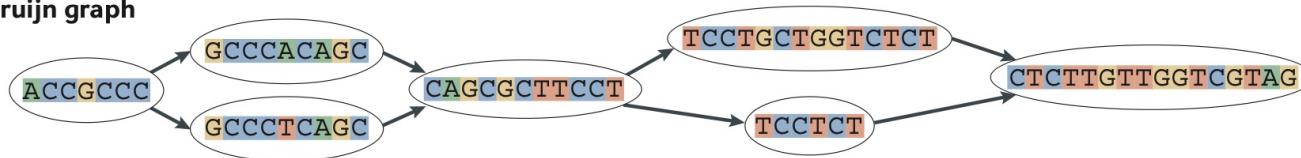


Martin, Wang (2011) Nat Rev Genet, 12, 671-682.

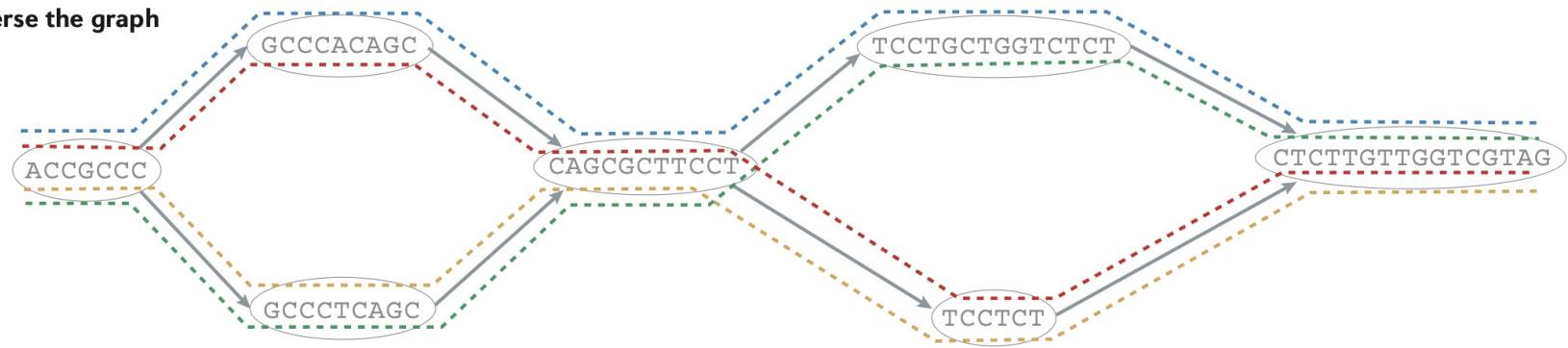
# *De novo* transcriptome assembly

## Assembly strategy

Collapse the De Bruijn graph



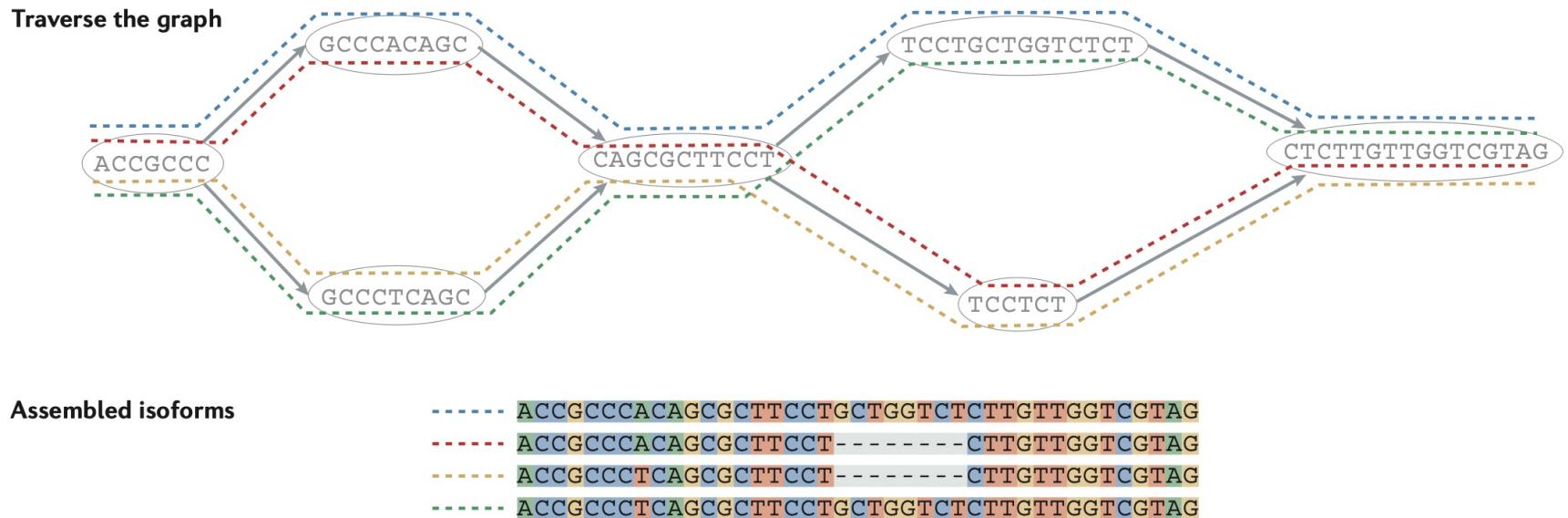
Traverse the graph



Martin, Wang (2011) Nat Rev Genet, 12, 671-682.

# *De novo* transcriptome assembly

## Assembly strategy



Martin, Wang (2011) Nat Rev Genet, 12, 671-682.

# RNA-Seq: Assembly

## Trinity

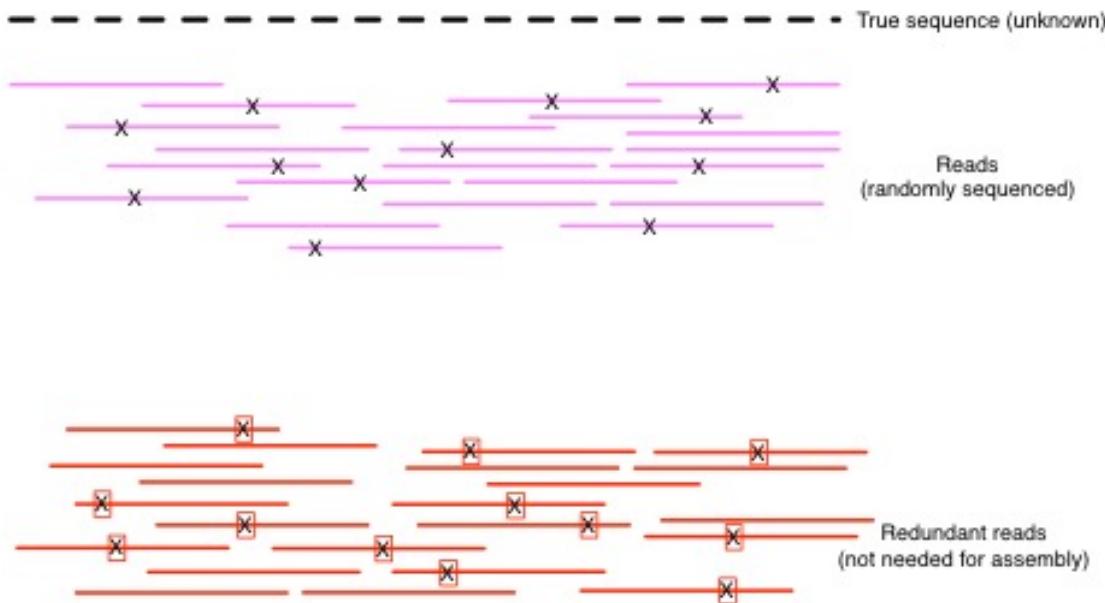
Uses of Trinity are:

- Inchworm assembles the RNA-seq data into the unique sequences of transcripts, often generating full-length transcripts for a dominant isoform.
- Chrysalis clusters the Inchworm contigs into clusters and constructs complete de Bruijn graphs for each cluster. Each cluster represents the full transcriptonal complexity for a given gene.
- Butterfly reports full-length transcripts for alternatively spliced isoforms, and teasing apart transcripts that corresponds to paralogous genes.

# *De novo* transcriptome assembly

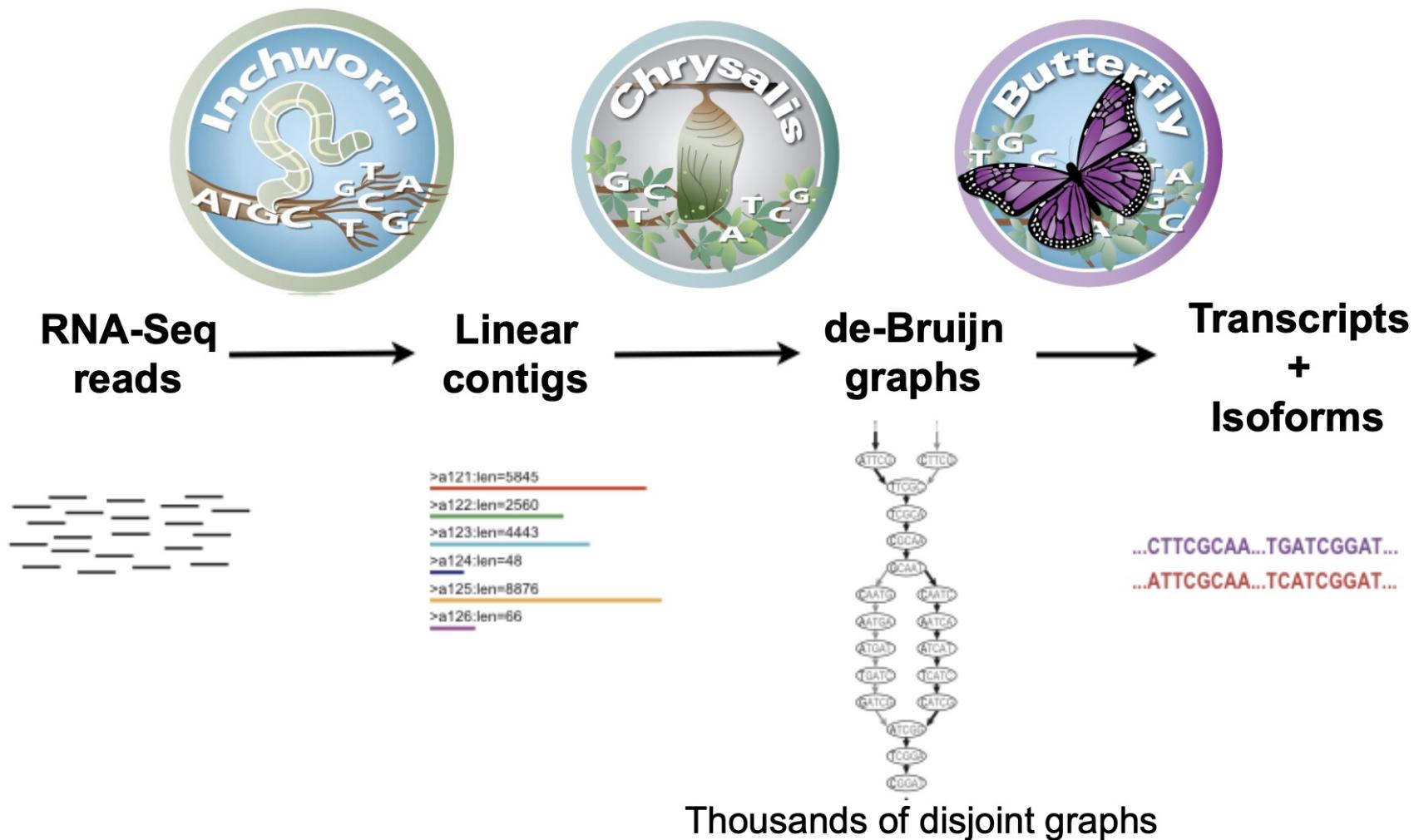
## Assembly strategy

### Digital normalization



# *De novo* transcriptome assembly

## Trinity



# Transcriptome assembly with Trinity

## Inchworm Algorithm

Read: AATGTGAAA**ACTGGATTACATGCTGGTATGTC...**

AATGTGA

ATGTGAA

Overlapping kmers of length (k)

TGTGAAA

...

### Kmer Catalog (hashtable)

Kmer	Count among all reads
AATGTGA	4
ATGTGAA	2
TGTGAAA	1
GATTACA	9

# Transcriptome assembly with Trinity

## Inchworm Algorithm

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

Read: AATGTGAAA ACTGGATTACATGCTGGTATGTC...

AATGTGA

ATGTGAA

Overlapping kmers of length (k)

TGTGAAA

...

## Kmer Catalog (hashtable)

Kmer	Count among all reads
AATGTGA	4
ATGTGAA	2
TGTGAAA	1
GATTACA	9

# Transcriptome assembly with Trinity

## Inchworm Algorithm

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

GATTACA  
9

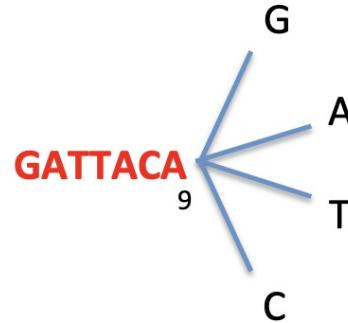
### Kmer Catalog (hashtable)

Kmer	Count among all reads
AATGTGA	4
ATGTGAA	2
TGTGAAA	1
GATTACA	9

# Transcriptome assembly with Trinity

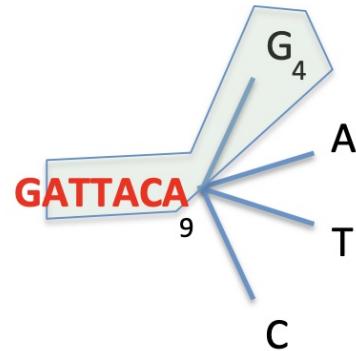
## Inchworm Algorithm

- 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.



# Transcriptome assembly with Trinity

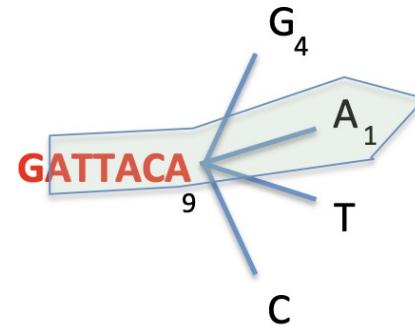
## Inchworm Algorithm



Brian Haas

# Transcriptome assembly with Trinity

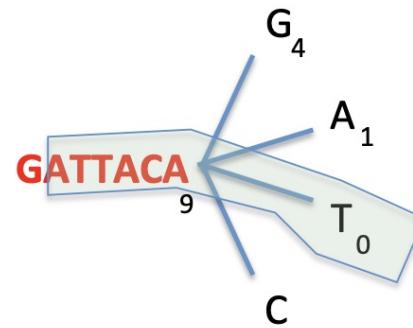
## Inchworm Algorithm



Brian Haas

# Transcriptome assembly with Trinity

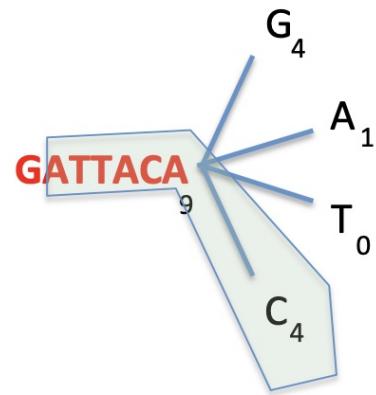
## Inchworm Algorithm



Brian Haas

# Transcriptome assembly with Trinity

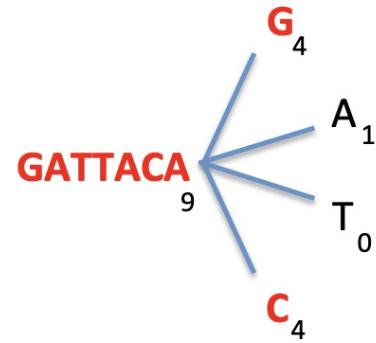
## Inchworm Algorithm



Brian Haas

# Transcriptome assembly with Trinity

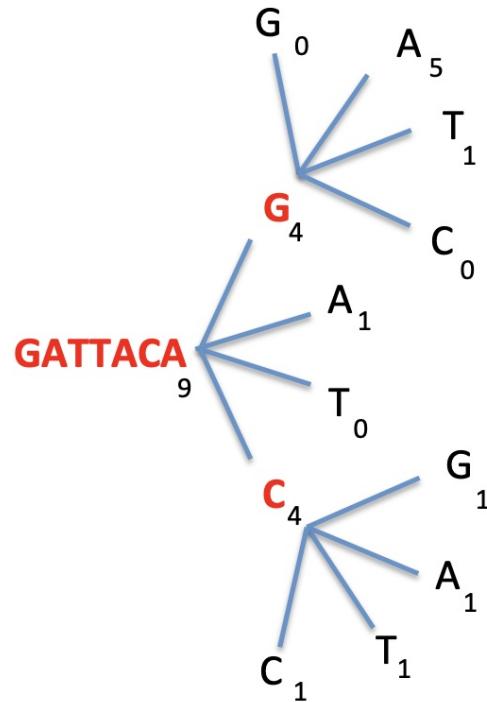
## Inchworm Algorithm



Brian Haas

# Transcriptome assembly with Trinity

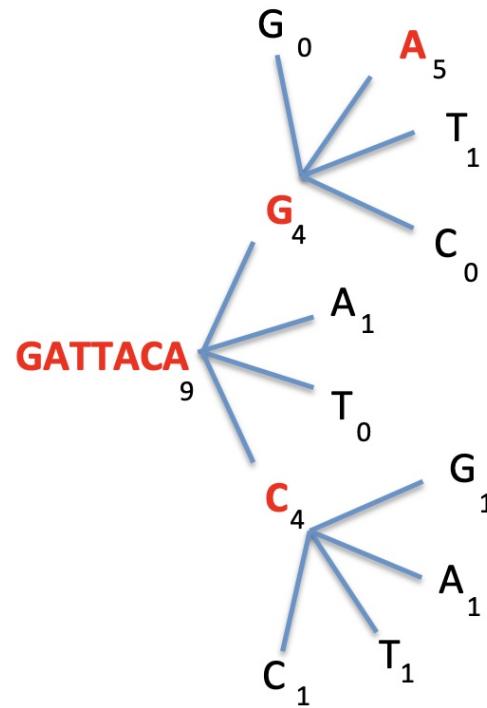
## Inchworm Algorithm



Brian Haas

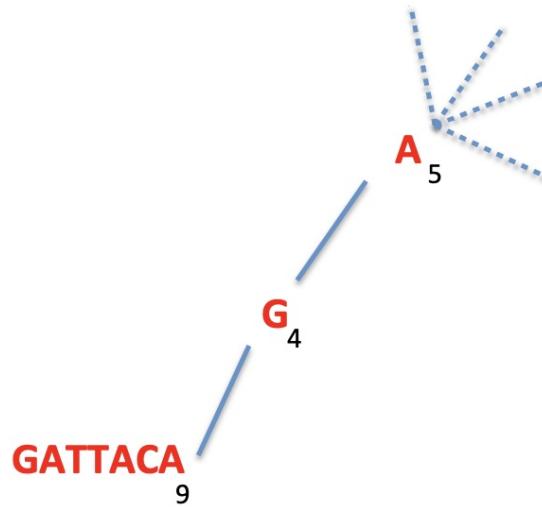
# Transcriptome assembly with Trinity

## Inchworm Algorithm



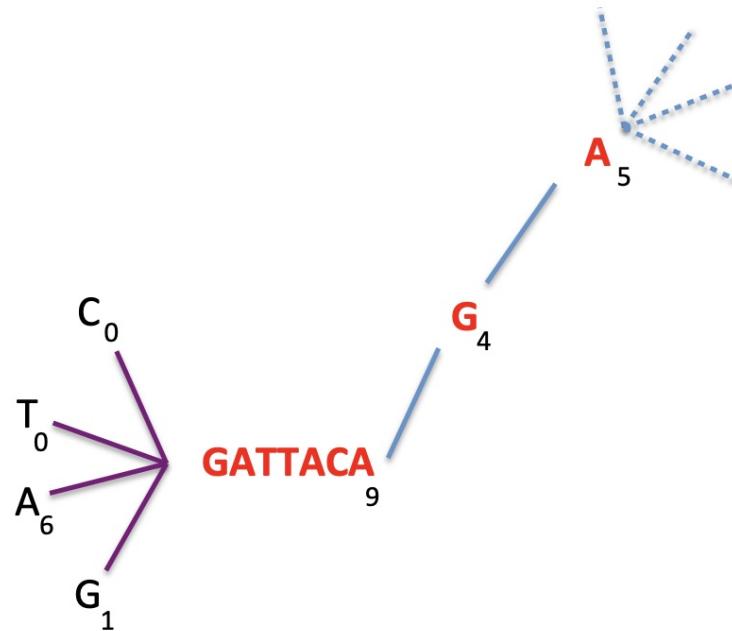
# Transcriptome assembly with Trinity

## Inchworm Algorithm



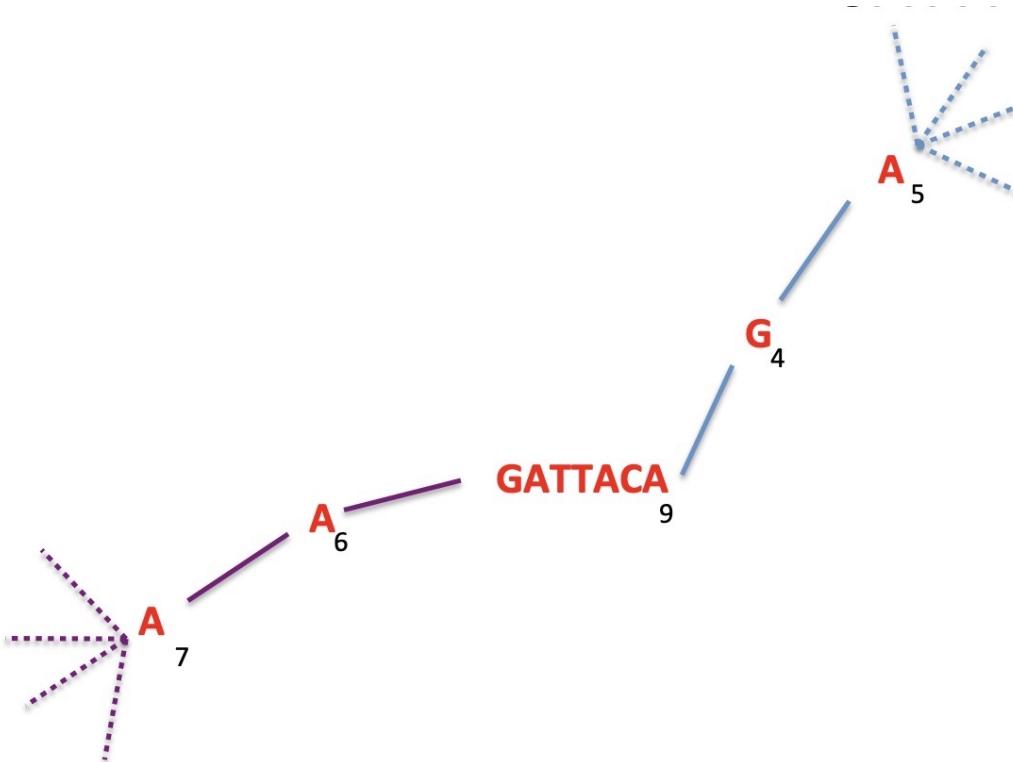
# Transcriptome assembly with Trinity

## Inchworm Algorithm



# Transcriptome assembly with Trinity

## Inchworm Algorithm

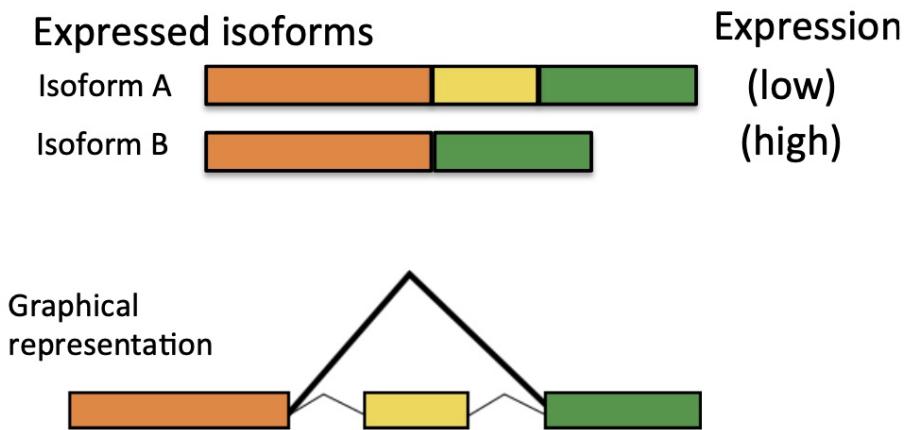


Report contig: ....**AAGATTACAGA**....

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

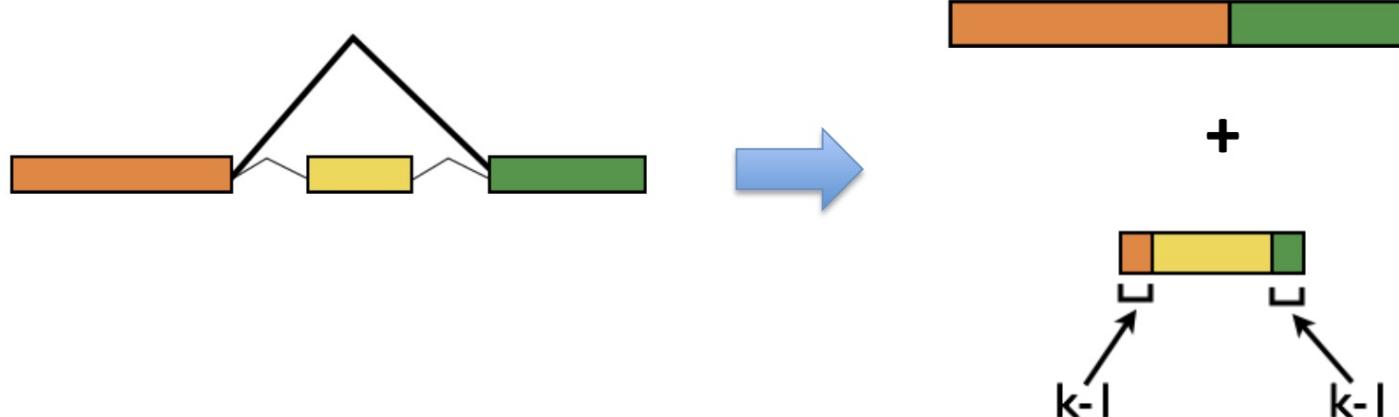
# Transcriptome assembly with Trinity

## Inchworm Algorithm



# Transcriptome assembly with Trinity

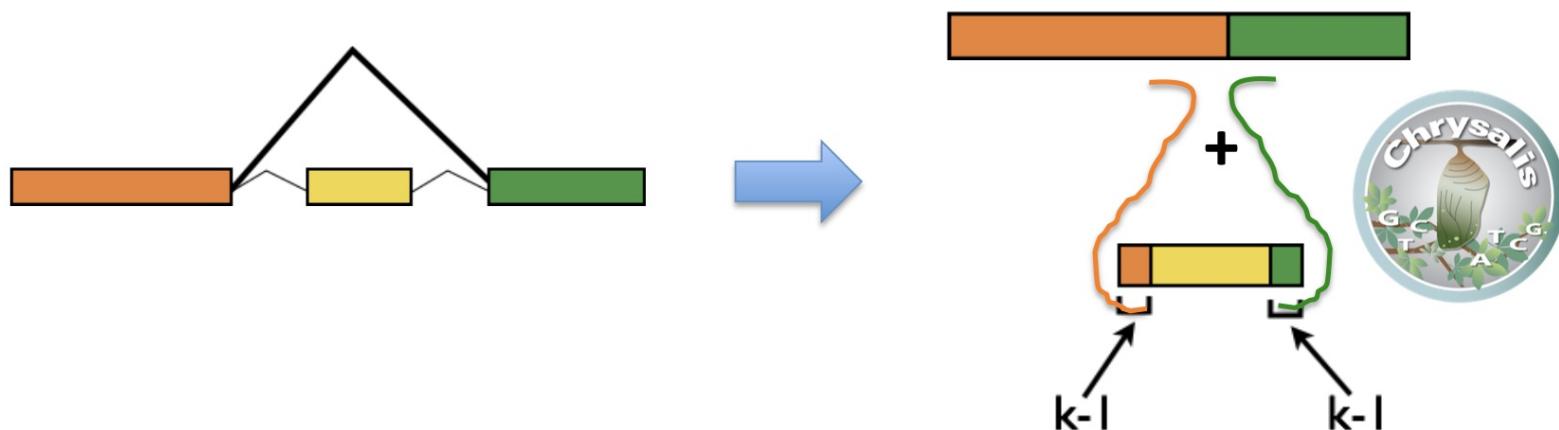
## Inchworm Algorithm



# Transcriptome assembly with Trinity

## Chrysalys Algorithm

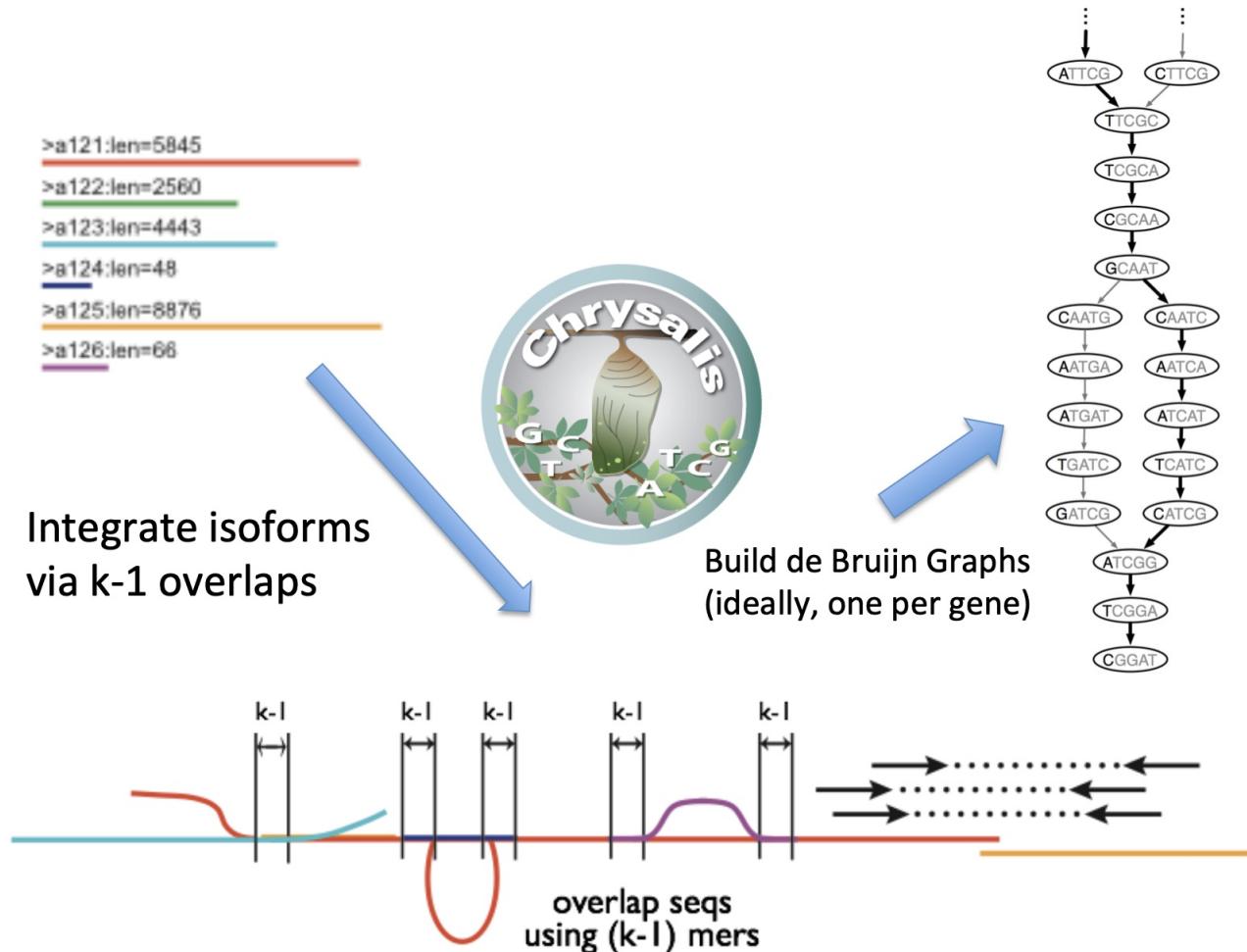
Chrysalys Re-groups Related Inchworm Contigs



Chrysalis uses  $(k-1)$  overlaps and read support to link related Inchworm contigs

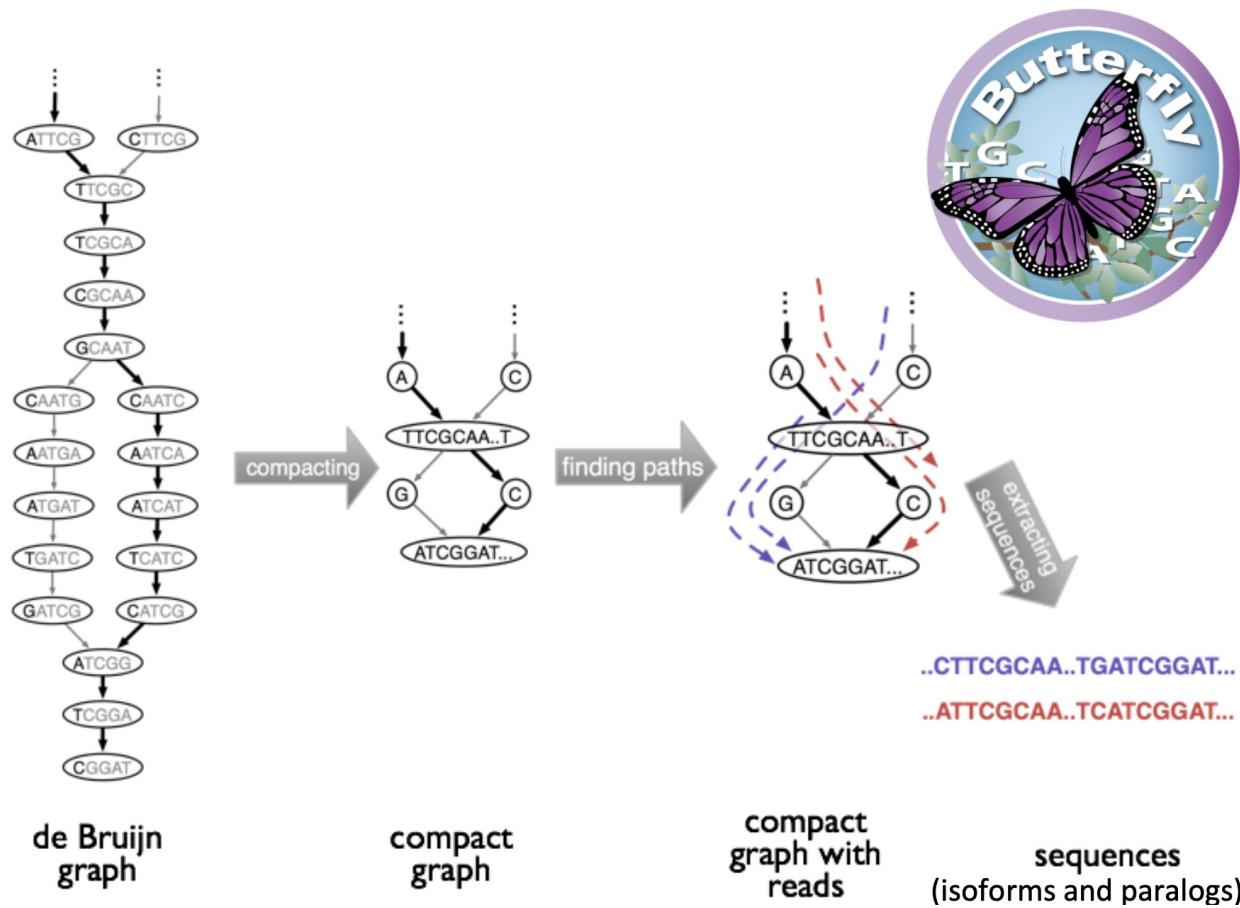
# Transcriptome assembly with Trinity

## Chrysalys Algorithm



# Transcriptome assembly with Trinity

## Butterfly Algorithm



Brian Haas

# Transcriptome assembly with Trinity

## Butterfly Algorithm

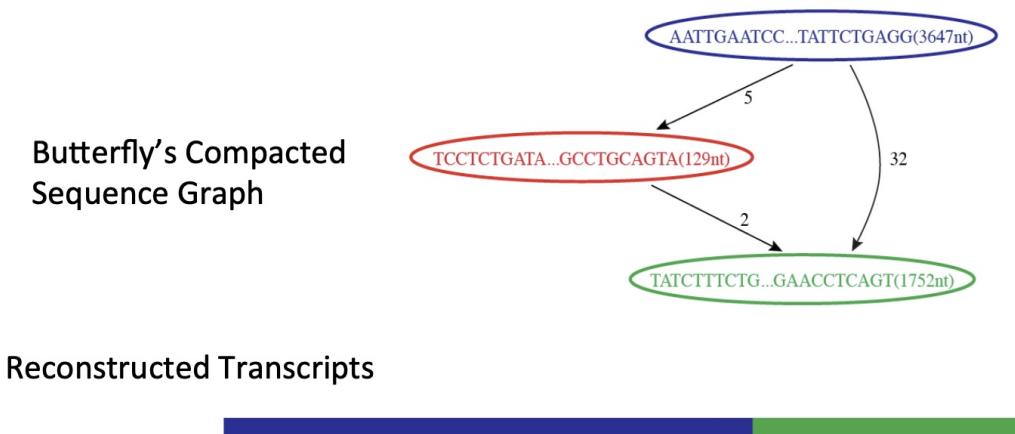
Butterfly Example 1:  
Reconstruction of Alternatively Spliced Transcripts



# Transcriptome assembly with Trinity

## Butterfly Algorithm

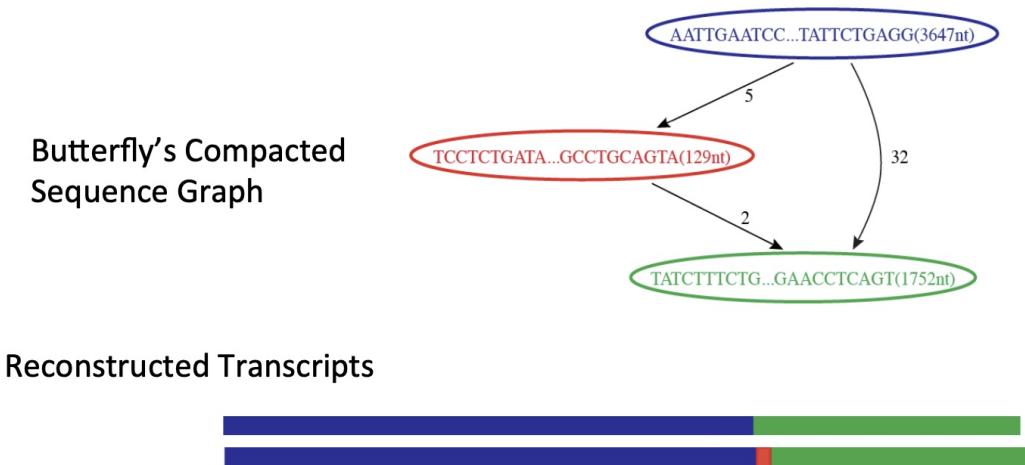
### Reconstruction of Alternatively Spliced Transcripts



# Transcriptome assembly with Trinity

## Butterfly Algorithm

### Reconstruction of Alternatively Spliced Transcripts

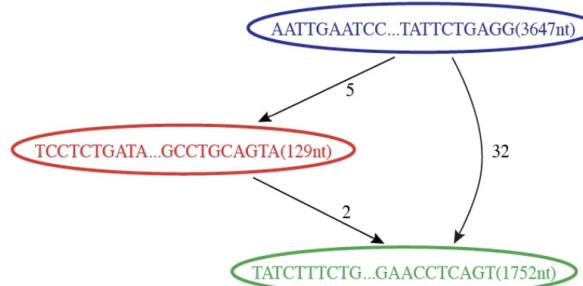


# Transcriptome assembly with Trinity

## Butterfly Algorithm

### Reconstruction of Alternatively Spliced Transcripts

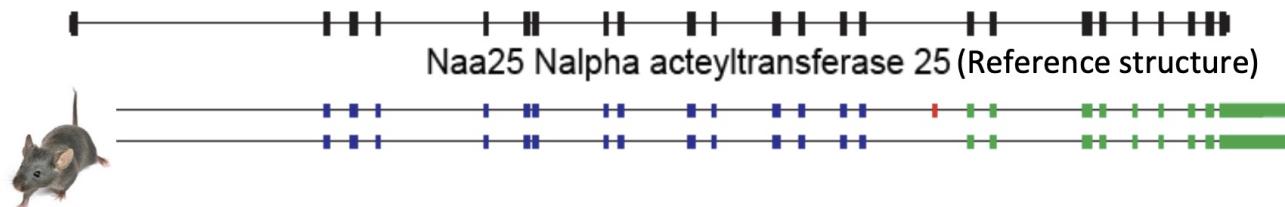
Butterfly's Compacted Sequence Graph



Reconstructed Transcripts



Aligned to Mouse Genome



# RNA-Seq: Assembly

## Trinity

### Assembly results

Dois arquivos:

- \${SPECIES}.Trinity.fasta
- \${SPECIES}.fasta.gene\_trans\_map

Exemplo:

- Chom.Trinity.fasta
- Chom.fasta.gene\_trans\_map

# RNA-Seq: Assembly

## Trinity results

1. Check files: \*.Trinity.fasta :

```
# head ${SPECIES}.Trinity.fasta

head ChomF_1.Trinity.fasta
>TRINITY_DN1870_c0_g1_i1 len=208 path=[0:0-207]
ACTGAATCTATCATGAAACGTACCGCTTGGTAGCCAACACCTCTAACATGCCTGTC
GCTCGTGAAGCTTCAATTACACTGGTATTACCTTATCTGAATACTTCCGTGATATG
TACAACGTATCTATGATGGCTGATTCTACCTCTCGTTGGGCTGAAGCTTCGTGAA
TCTGGTCGTTGGCTGAAATGCCTGCCG
>TRINITY_DN1811_c0_g1_i1 len=738 path=[0:0-737]
CAGGATGTCTATAAAATTGGTGGTATTGGTACAGTACCCGTGGGTGTTGAAACTC
ATTTTGAAACCTGGTATGGTGGTTAATTTCCTGTCAATTGGTAAGTC
TCGGTGGAAATGCATCATGAAGCCTGTCGGAAGCAATGCCCGGTGATAATGTTGGT
AATGTTAAAAACGTTCGGTTAAGGAATTGCGTCGCGGTTATGTTGCCGGTGATAACC
```

# RNA-Seq: Assembly

## Trinity results

### The `.Trinity.fasta` file

```
TRINITY_DN1870_c0_g1_i1 len=208 path=[0:0-207]
```

→ TRINITY\_DN1870\_c0\_g1\_i1:

- DN1870: unique identifier for the de novo assembly cluster to which the contig belongs.
- c0: specific component within the cluster, connected sequences that likely derive from the same gene family.
- g1: gene-like group within the component. Sequences within the same group (g1) are interpreted as splice variants or isoforms of the same gene.
- i1: Represents the specific isoform number (e.g., i1 is isoform 1).

# RNA-Seq: Assembly

## Trinity results

### The `.Trinity.fasta` file

```
TRINITY_DN1870_c0_g1_i1 len=208 path=[0:0-207]
```

→ len=208:

- Length of the contig in base pairs (bp). In this example, the contig is 208 bp long.

→ path=[0:0-207]:

- Assembly path used to reconstruct the contig.
  - 0: Represents the node index in the assembly graph.
  - 0-207: Specifies the position range in the node that was used to assemble this contig.

# RNA-Seq: Assembly

## Trinity results

2. Check files: \*.Trinity.fasta.gene\_trans\_map :

```
# head ${SPECIES}.Trinity.fasta.gene_trans_map

head ChomF_1.Trinity.fasta.gene_trans_map
TRINITY_DN1870_c0_g1      TRINITY_DN1870_c0_g1_i1
TRINITY_DN1811_c0_g1      TRINITY_DN1811_c0_g1_i1
TRINITY_DN1862_c0_g1      TRINITY_DN1862_c0_g1_i1
TRINITY_DN1808_c0_g1      TRINITY_DN1808_c0_g1_i1
TRINITY_DN1869_c0_g1      TRINITY_DN1869_c0_g1_i1
TRINITY_DN1803_c0_g1      TRINITY_DN1803_c0_g1_i1
TRINITY_DN1835_c0_g1      TRINITY_DN1835_c0_g1_i1
TRINITY_DN1892_c0_g1      TRINITY_DN1892_c0_g1_i1
TRINITY_DN1854_c0_g1      TRINITY_DN1854_c0_g1_i1
TRINITY_DN1880_c0_g1      TRINITY_DN1880_c0_g1_i1
```

# RNA-Seq: Assembly

## Trinity results

### The `.gene_trans_map` file

Mapping between genes and transcripts identified during the assembly. Links individual transcripts to their respective parent genes.



Software tools like Trinotate, RSEM, and Salmon use this file to relate the expression of isoforms and genes.

# RNA-Seq: Assembly

## Trinity results

2. How many contigs were assembled?

# RNA-Seq: Assembly

## Trinity results

2. How many contigs were assembled?

```
#grep -c '^>' ${SPECIES}.Trinity.fasta
```

```
grep -c '^>' Chom.Trinity.fasta
```

1733

# RNA-Seq: Assembly

## Trinity results

```
TrinityStats.pl Trinity.fasta

#####
## Counts of transcripts, etc.
#####
Total trinity 'genes': 1636
Total trinity transcripts: 1733
Percent GC: 38.72
```

# RNA-Seq: Assembly

## Trinity results

```
#####
Stats based on ALL transcript contigs:
#####
```

Contig N10: 1955

Contig N20: 1400

Contig N30: 988

Contig N40: 778

Contig N50: 620

Median contig length: 351

Average contig: 519.83

Total assembled bases: 900862

# RNA-Seq: Assembly

## Trinity results

```
#####
## Stats based on ONLY LONGEST ISOFORM per 'GENE':
#####
```

Contig N10: 1625

Contig N20: 1195

Contig N30: 869

Contig N40: 684

Contig N50: 552

Median contig length: 338

Average contig: 480.36

Total assembled bases: 785867

# RNA-Seq: Assembly

Trinity

Assembly results

Exemplo:

- ChomF.Trinity.fasta
- ChomF.fasta.gene\_trans\_map

Quality assessments ?

# RNA-Seq: Assembly

## Quality assessment: BUSCO



# BUSCO

from QC to gene prediction and phylogenomics

We are pleased to announce the release of new BUSCO datasets! Based on OrthoDBv12 (<https://orthodb.org>), the new datasets represent a significant increase in coverage over all domains. The new odb12 dataset release contains 36 datasets for archaea, up from 16, and 334 datasets for bacteria, up from 83. The eukaryota dataset release is being finalised and will be released in the coming weeks.

BUSCO v5.8.2 is the current stable version!  
[Gitlab](#), a [Conda package](#) and [Docker container](#) are also available.

Based on evolutionarily-informed expectations of gene content of near-universal single-copy orthologs, the BUSCO metric is complementary to technical metrics like N50.

BUSCO was selected as one of the [SIB Remarkable Outputs of 2021](#)!

<https://busco.ezlab.org>

# RNA-Seq: Assembly

## Quality assessment: BUSCO



# BUSCO

from QC to gene prediction and phylogenomics

We are pleased to announce the release of new BUSCO datasets! Based on OrthoDBv12 (<https://orthodb.org>), the new datasets represent a significant increase in coverage over all domains. The new odb12 dataset release contains 36 datasets for archaea, up from 16, and 334 datasets for bacteria, up from 83. The eukaryota dataset release is being finalised and will be released in the coming weeks.

BUSCO v5.8.2 is the current stable version!  
[Gitlab](#), a [Conda package](#) and [Docker container](#) are also available.

Based on evolutionarily-informed expectations of gene content of near-universal single-copy orthologs, the BUSCO metric is complementary to technical metrics like N50.

BUSCO was selected as one of the [SIB Remarkable Outputs of 2021](#)!

<https://busco.ezlab.org>

# RNA-Seq: Assembly

## Quality assessment: BUSCO

### Benchmarking Universal Single-Copy Orthologue (BUSCO)

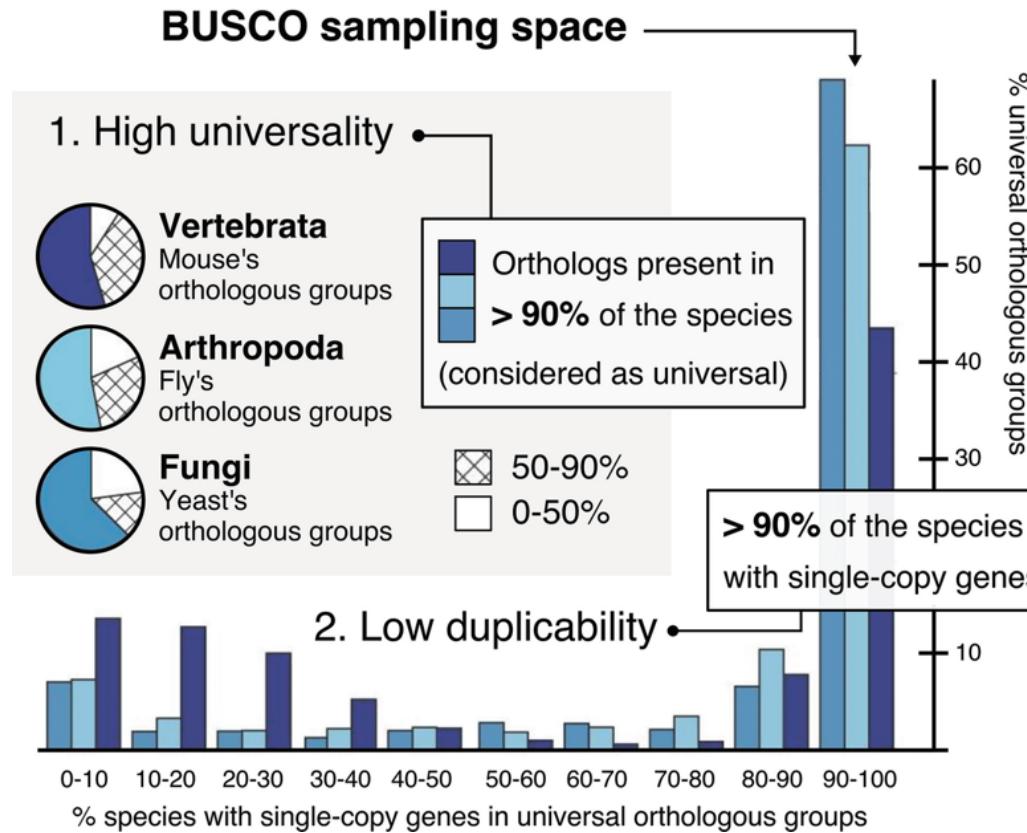
Provides evolutionarily sound measures of completeness and redundancy in terms of expected gene content

This is achieved by considering the presence of a predefined and expected set of single-copy marker genes as a proxy for genome-wide completeness.

<https://busco.ezlab.org>

# RNA-Seq: Assembly

## Quality assessment: BUSCO



# RNA-Seq: Assembly

## Quality assessment: BUSCO

-  BUSCO uses AUGUSTUS to predict gene models (determine what genes are likely present in your transcriptome).
-  HMMER is then used to detect orthologs in those predicted genes (checking if the genes are similar to the set of conserved, universal orthologs).

# RNA-Seq: Assembly

## BUSCO

### Running BUSCO

1. Activate environment

```
conda activate annotation
```

2. Call BUSCO to read help

```
busco --help
```

# RNA-Seq: Assembly

## BUSCO

### Running BUSCO

#### 2. Call BUSCO to read help

```
usage: busco -i [SEQUENCE_FILE] -l [LINEAGE] \
             -o [OUTPUT_NAME] -m [MODE] [OTHER OPTIONS]

-i SEQUENCE_FILE, --in SEQUENCE_FILE
-o OUTPUT, --out OUTPUT
-m MODE, --mode MODE  Specify which BUSCO analysis mode
                      There are three valid modes:
                      - geno or genome (DNA)
                      - tran or transcriptome (DNA)
                      - prot or proteins (protein)
-l LINEAGE, --lineage_dataset LINEAGE
                      Specify the name of the BUSCO
                      lineage to be used
```

# RNA-Seq: Assembly

## BUSCO

### Running BUSCO

#### 3. See available databases

```
busco --list-datasets
```

- eukaryota\_odb10
  - metazoa\_odb10
    - arthropoda\_odb10
      - arachnida\_odb10
      - insecta\_odb10
        - endopterygota\_odb10
          - diptera\_odb10
          - hymenoptera\_odb10
          - lepidoptera\_odb10
        - hemiptera\_odb10

# RNA-Seq: Assembly

## BUSCO

### Running BUSCO

#### 3. See available databases

```
busco --list-datasets

- eukaryota_odb10..... 255 BUSCOs
  - metazoa_odb10..... 954 BUSCOs
    - arthropoda_odb10..... 1013 BUSCOs
      - arachnida_odb10
      - insecta_odb10..... 1367 BUSCOs
        - endopterygota_odb10..... 2124 BUSCOs
          - diptera_odb10..... 3285 BUSCOs
          - hymenoptera_odb10
          - lepidoptera_odb10
        - hemiptera_odb10
```

# RNA-Seq: Assembly

## BUSCO

### Running BUSCO

#### 4. Run BUSCO

```
busco -i Trinity.fasta -o busco_diptera -l diptera_odb10  
-m transcriptome -c 8
```

- ➔ -i: Transcriptome file
- ➔ -o: Output folder
- ➔ -l: dataset
- ➔ -m: analysis mode
- ➔ -c: number of threads

# RNA-Seq: Assembly

## BUSCO

### BUSCO results

```
|Results from dataset diptera_odb10
|C:3.4%[S:3.2%,D:0.3%],F:0.8%,M:95.8%,n:3285
|113    Complete BUSCOs (C)
|104    Complete and single-copy BUSCOs (S)
|9      Complete and duplicated BUSCOs (D)
|25     Fragmented BUSCOs (F)
|3147   Missing BUSCOs (M)
|3285   Total BUSCO groups searched
```

# RNA-Seq: Assembly

## BUSCO

### Plotting BUSCO results

5. Where is `generate_plot.py` ?

```
which generate_plot.py
```

6. Test `generate_plot.py`

```
/path/to/script/generate_plot.py -h
```

7. Create a folder for summary files

```
mkdir BUSCO_summaries
```

# RNA-Seq: Assembly

## BUSCO

### Plotting BUSCO results

8. Move summary files to new folder

```
cp short_summary.txt BUSCO_summaries/
```

9. Run `generate_plot.py`

```
generate_plot.py -wd BUSCO_summaries/
```

10. If it doesn't work, run

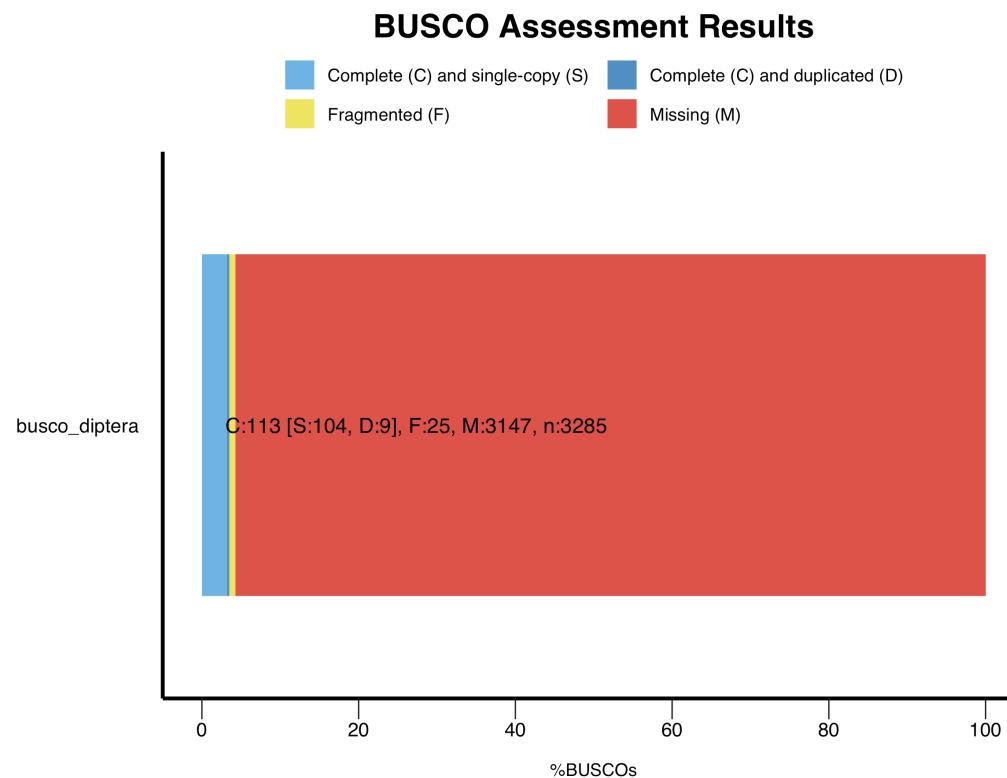
```
generate_plot.py -wd BUSCO_summaries/ --no_r
```

11. Open the R script in Rstudio and run it

# RNA-Seq: Assembly

## BUSCO

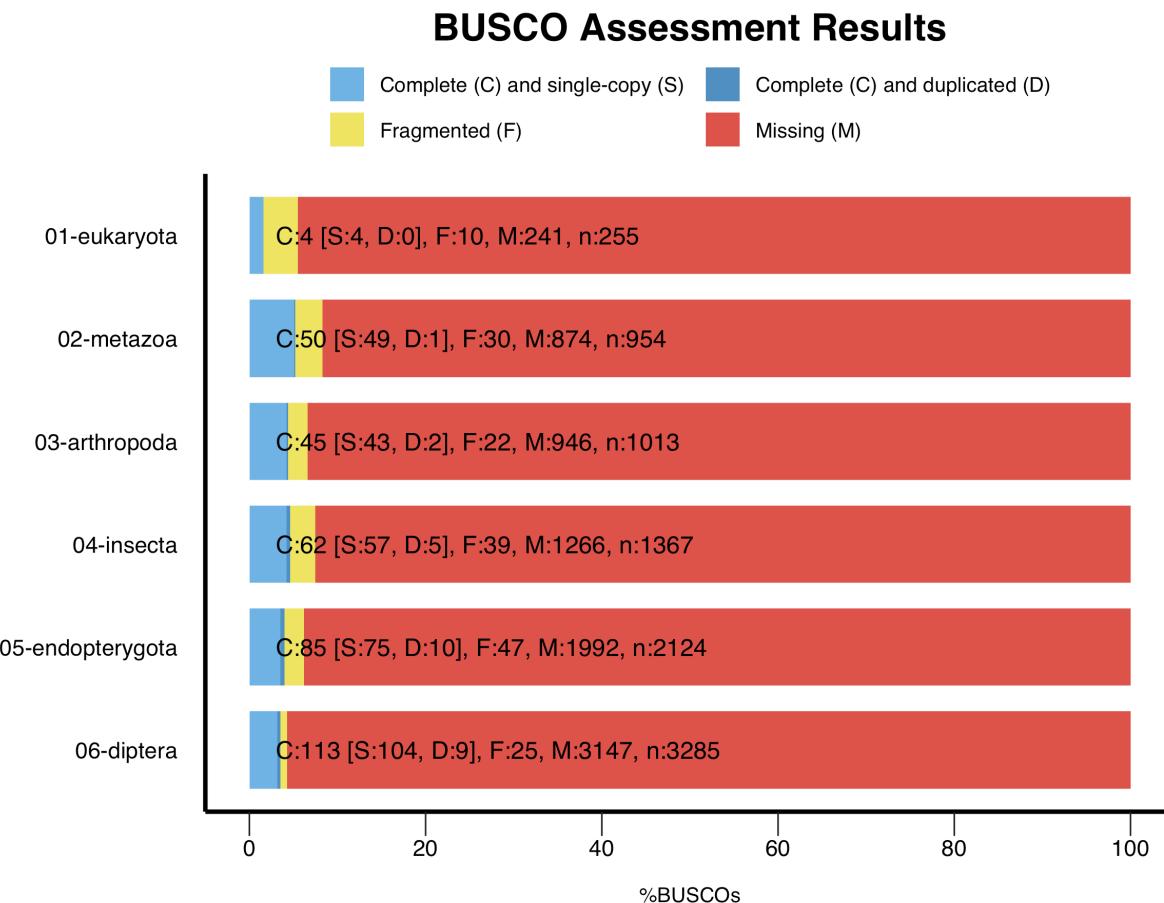
### Plotting BUSCO results



# RNA-Seq: Assembly

## BUSCO

### Plotting BUSCO results



# RNA-Seq: Assembly

## BUSCO

### Plotting BUSCO results

