



RNA-Seq de novo assembly training Day 2



Session organisation: Day 2

Morning:

- Assembly quality assessment
 - * Assemblathon stats
 - * Read mapping stats
- Clustering
 - * CD-HIT
- Greedy assembly
 - * SSAKE/VCAKE
- Overlap Layout Consensus
 - * CAP3
 - * TGICL

Afternoon:

- de Bruijn graph
 - * Velvet/Oases
 - * Trinity
- Pretreatments

Objectives for this second day

Answer the following questions:

- Which assembler should I choose to process my data?
- Which procedure should I use for my assembler?
- Which computer do I need to run my assembly?

Genome vs transcriptome assembly

Differ mainly in:

- matrices coverage
 - * generally uniform vs highly variable
- combinations of sequences
 - * repeats and allelic variations vs alternative splicing

Two steps using genome assembler:

- use genome assembler to assemble transcriptome
- develop pipelines to postprocess the output of genome assemblers

Objectives of the assembly

An assembly is a sum-up of the matrices from which the reads have been produced:

- complete (all represented transcripts)
- compact (one contig for a transcript)
- independent of the expression level
- not affected by the random errors of the sequencing technology

Two possible level-analysis:

- transcripts
- genes

How do you assess the quality of an transcriptome assembly?

Assembly quality assessment

3 tracks

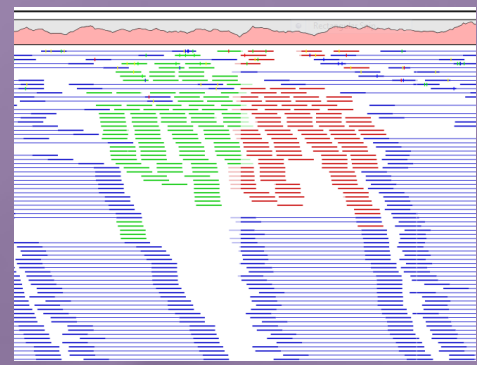
Assembly metrics



Shape of contig length histogram



Reads mapping back rate



Vocabulary

- **Contig**: a set of overlapping segments that together represent a consensus sequence
- **Scaffold**: a series of contigs that are in the right order but not necessarily connected in one continuous stretch of sequence
- **N50**: given a set of contigs of varying lengths, the N50 length is defined as the length N for which 50% of all bases in the contigs are in contigs of length $L < N$
 - contig size list $L = (2, 2, 2, 3, 3, 4, 8, 8)$
 - we have 50% of total length (16/32) above 4
 - N50** is equal to $4 + 8/2 = 6$
- **L50**: number of contigs that are greater than, or equal to, the N50 length

Contig metrics

The possible metrics derived from genome assembly:

- Idea of global size (# bases)
 - Idea of number of elements (# contigs/scaffolds)
 - Idea of compactness (N50)
- ↳ much more difficult to predict with transcriptome data

Assemblathon statistics

Script which calculates many of the basic contig and scaffold level statistics

- N50
- Longest/shortest contig/scaffold
- Median size of contigs/scaffolds
- Mean size of contigs/scaffolds
- Total size of contigs/scaffolds
- % N, A, T, G, C
- ...

Assemblathon 1: a competitive assessment of de novo short read assembly methods.
Earl D & al. Genome Res. 2011 Dec;21(12):2224-41

Assemblathon 2: evaluating de novo methods of genome assembly in three vertebrate species.
Bradnam KR & al. Gigascience. 2013 Jul 22;2(1):10

Assemblathon: command line

```
bash-4.1$ assemblathon_stats.pl
Usage: assemblathon_stats.pl <assembly_scaffolds_file>
options:
  -limit <int> limit analysis to first <int> sequences (useful for testing)
  -csv         produce a CSV output file of all results
  -graph       produce a CSV output file of NG(X) values (NG1 through to NG99), suitable for graphing
  -n <int>     specify how many consecutive N characters should be used to split scaffolds into contigs
  -genome_size <int> estimated or known genome size
```

-n distinguish scaffolds and contigs (default 25)
-csv output in csv format

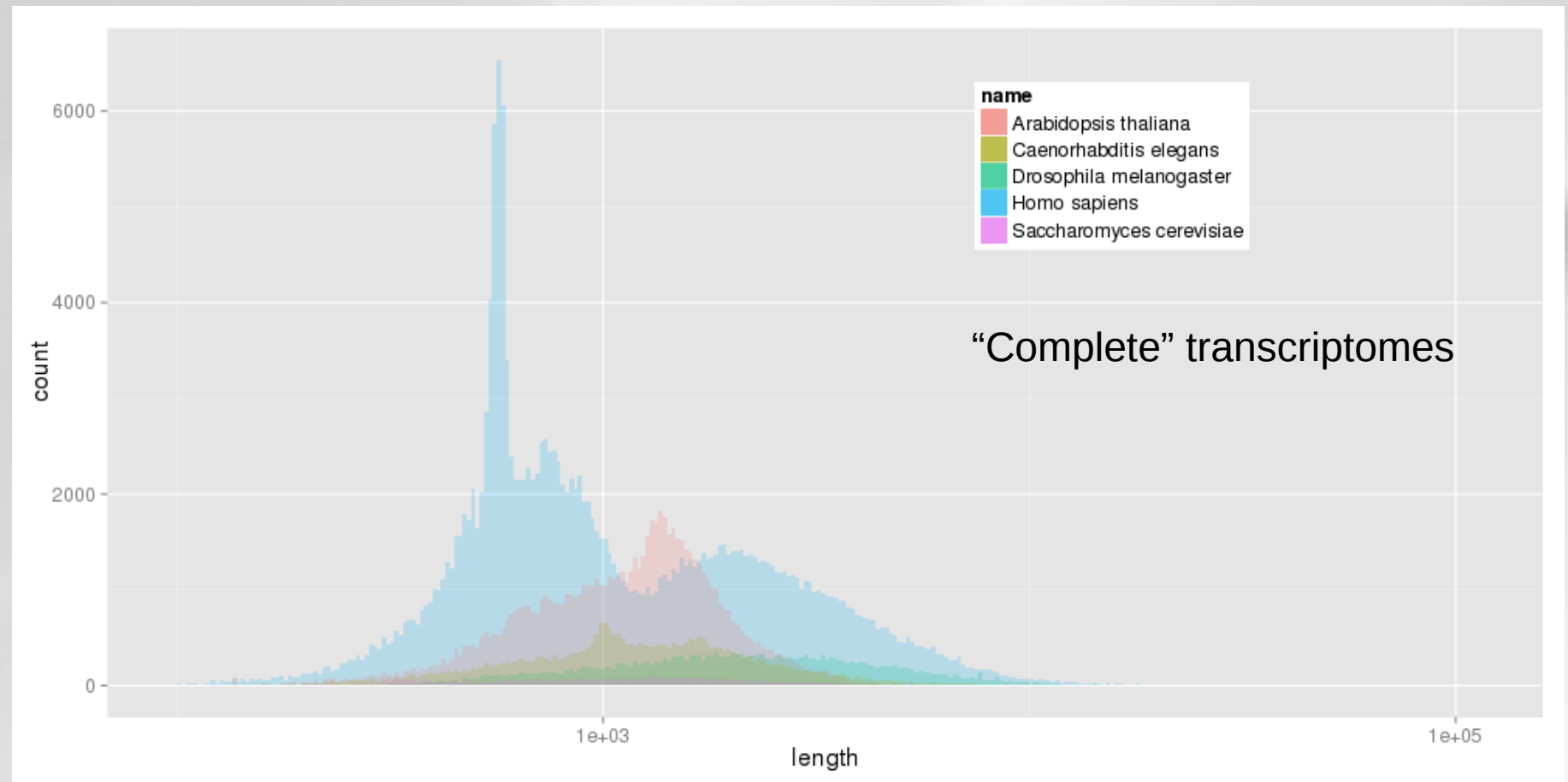
Assemblathon statistics

assemblathon_stats.pl contigs.fa

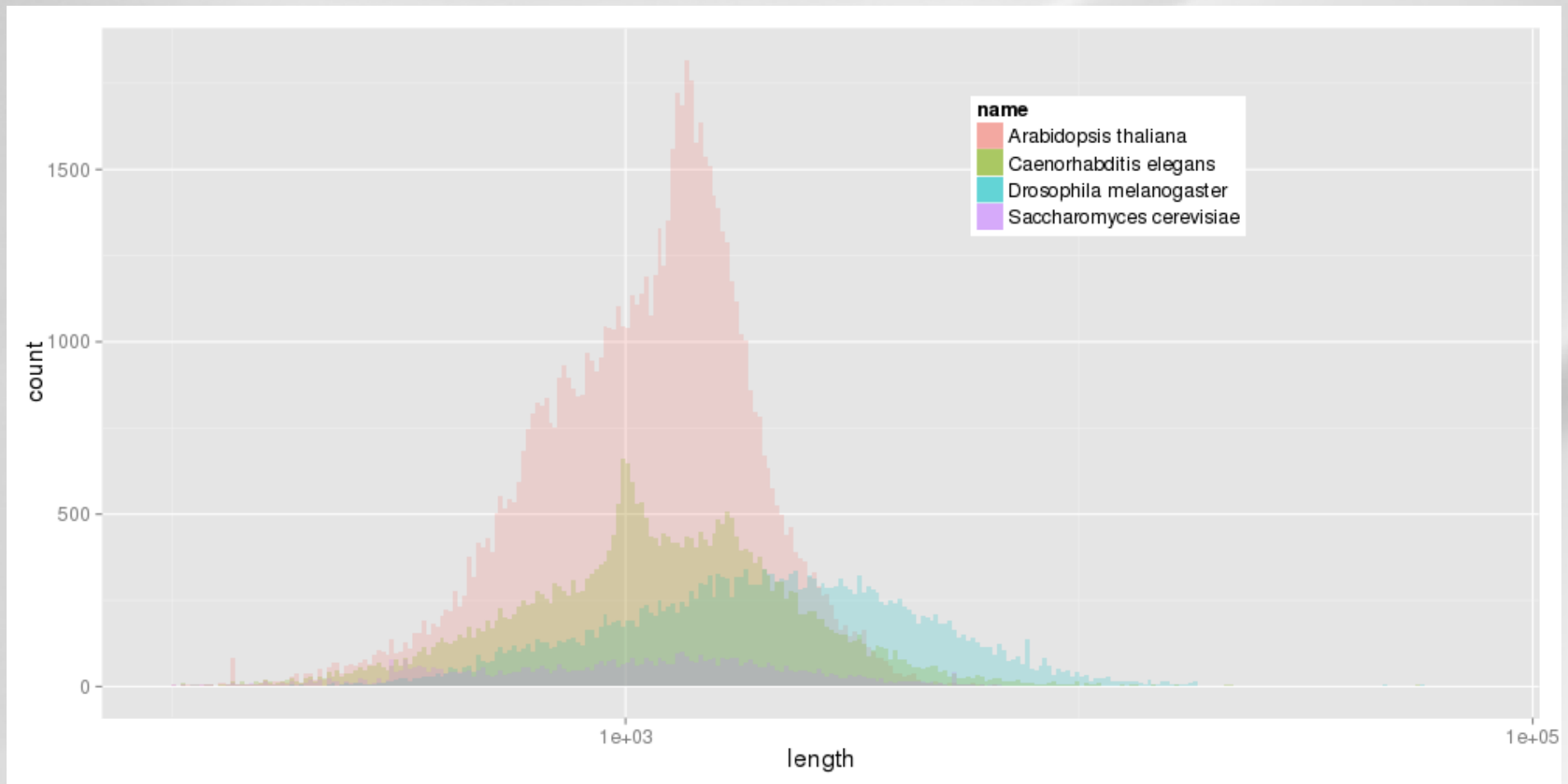
```
- Information for assembly 'F_Dr_1/merge/cap3/all_contigs_singlets.filtered.tfa' -
      Number of scaffolds          16164
    Total size of scaffolds      38963808
      Longest scaffold            17142
      Shortest scaffold           201
    Number of scaffolds > 1K nt   13358  82.6%
    Number of scaffolds > 10K nt    72   0.4%
    Number of scaffolds > 100K nt    0   0.0%
    Number of scaffolds > 1M nt     0   0.0%
    Number of scaffolds > 10M nt    0   0.0%
      Mean scaffold size           2411
      Median scaffold size          1922
      N50 scaffold length           3126
      L50 scaffold count            4028
      scaffold %A                   27.16
      scaffold %C                   22.96
      scaffold %G                   23.25
      scaffold %T                   26.62
      scaffold %N                    0.00
      scaffold %non-ACGTN           0.00
    Number of scaffold non-ACGTN nt  0
```

Transcript length histogram

Transcript lengths are not randomly distribute
We should get a known distribution shape



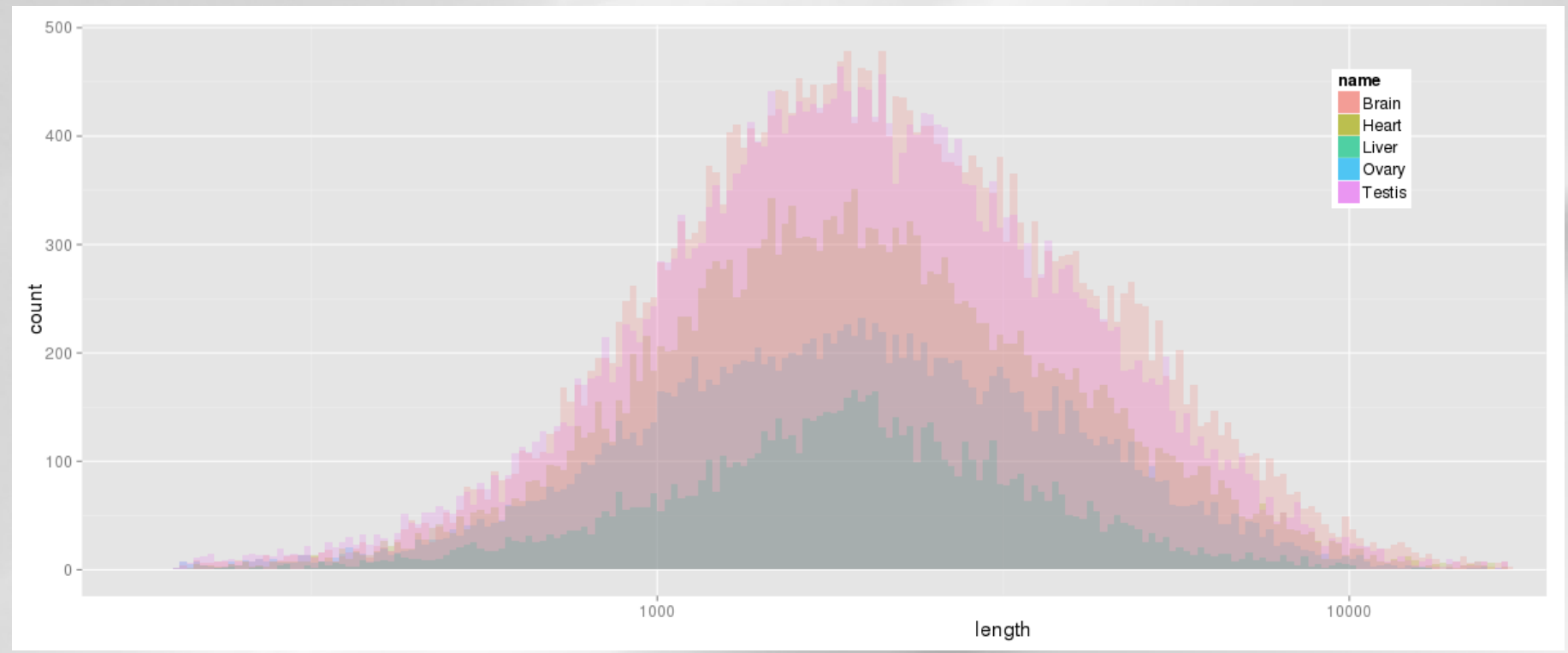
Transcript length histogram



Transcript length histogram

Zebrafish tissue specific assembled transcriptomes

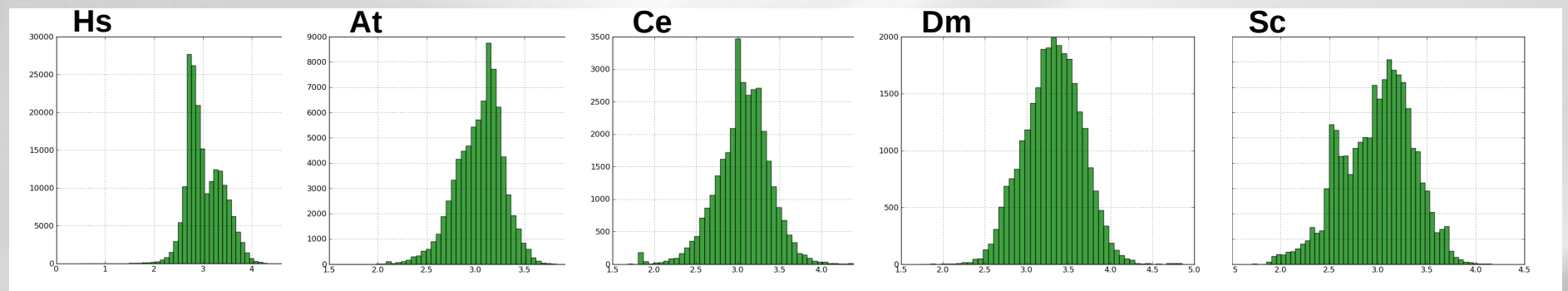
↳ not so different



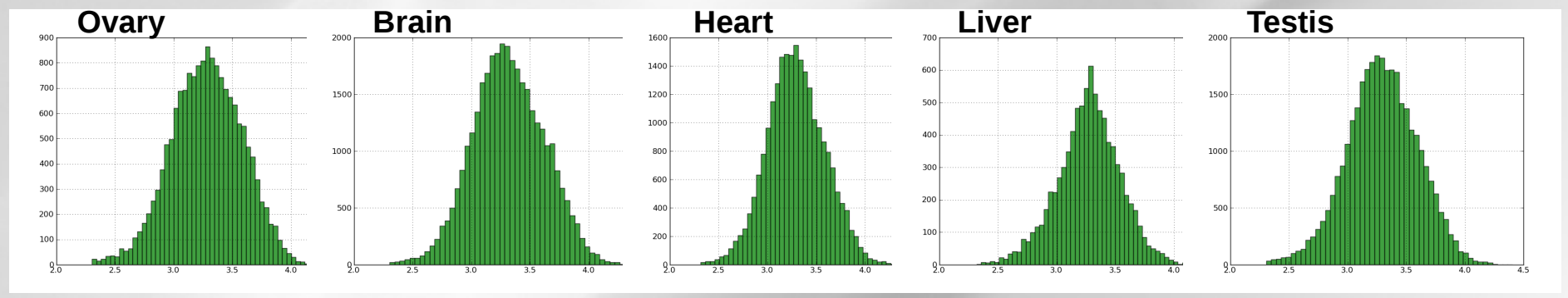
Length histogram

```
python /usr/local/bioinfo/Scripts/bin/length_histogram.py -l -i transcripts.fa
```

Complete transcriptomes

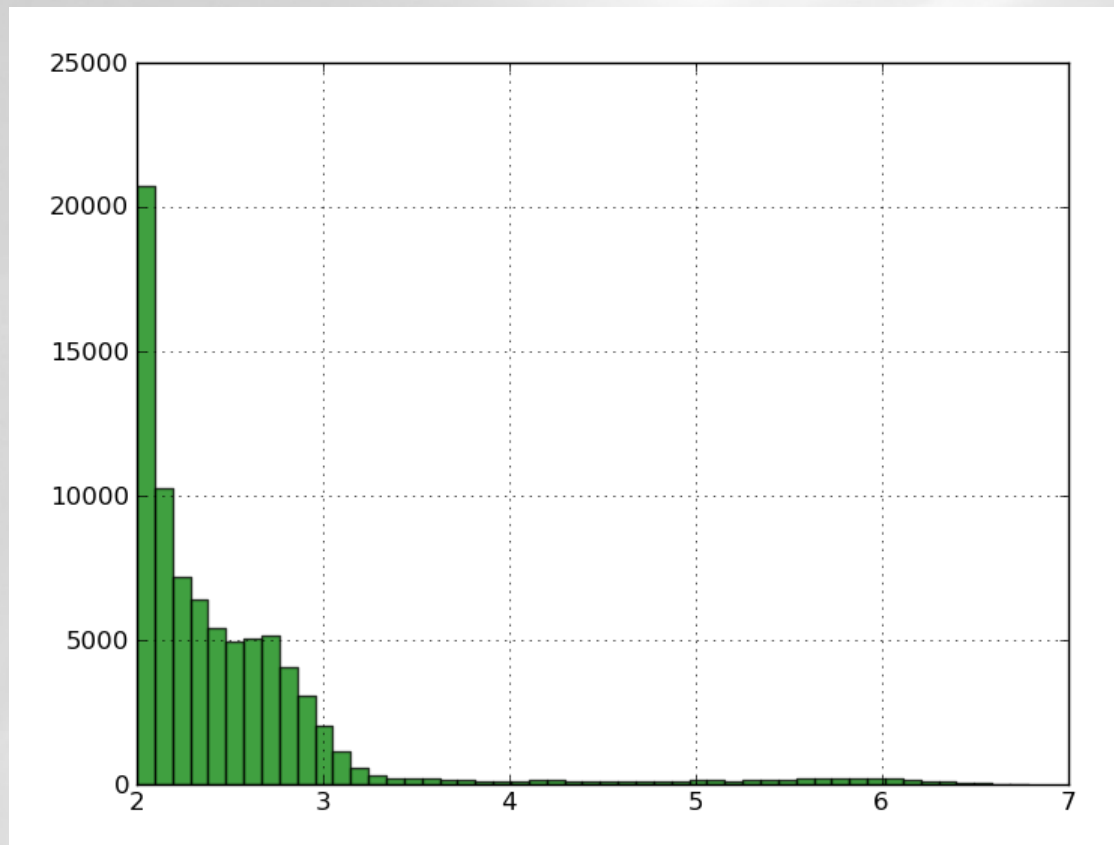


Tissue specific assembled transcriptomes



Genome histogram

Comparison with the panda genome assembly (v1, 2009)



81467 scaffolds

Total 2,3 Gb

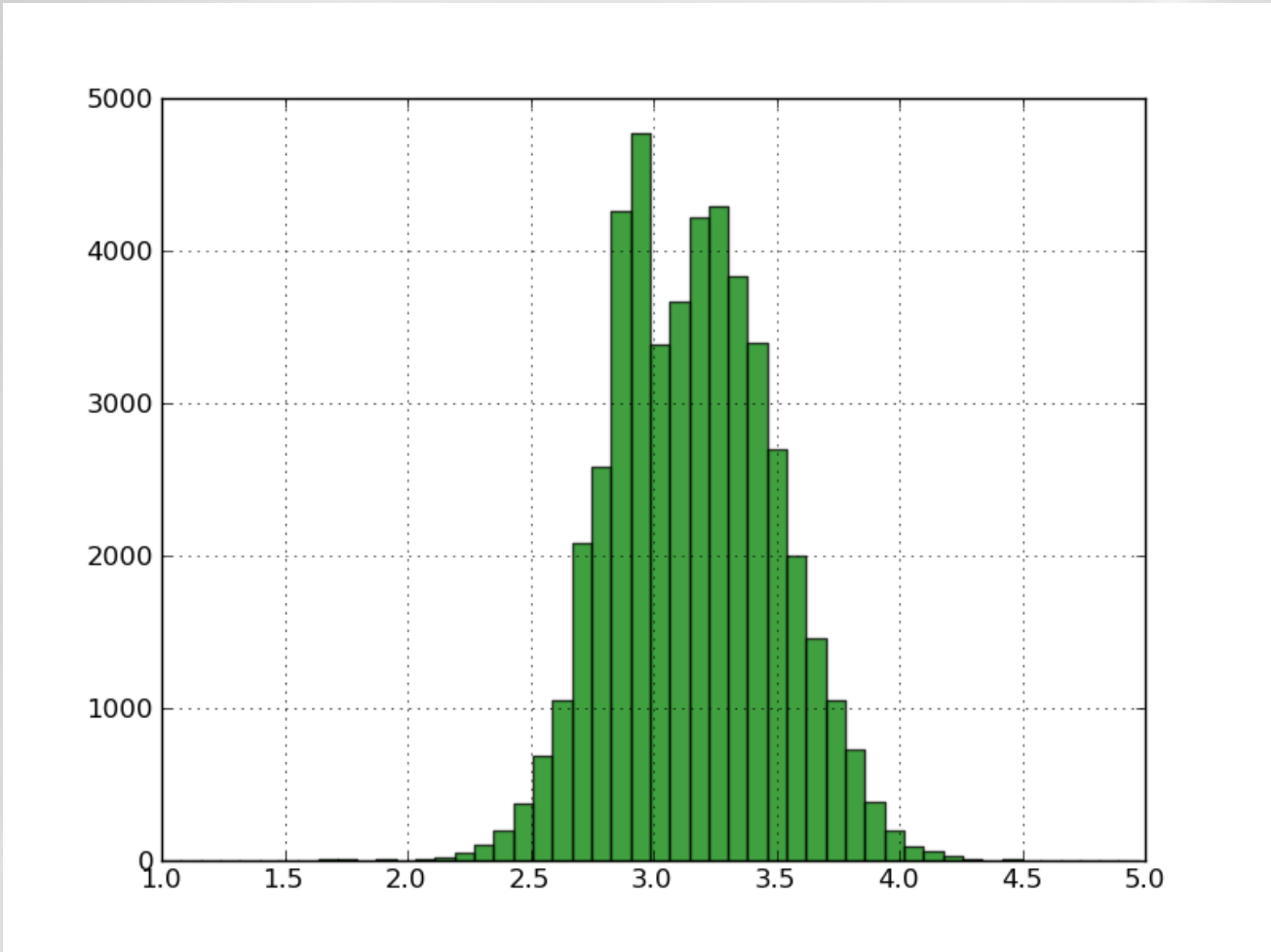
Longest 6Mb

Shortest 100 b

N50 1,3 Mb

L50 521

Zebrafish transcriptome



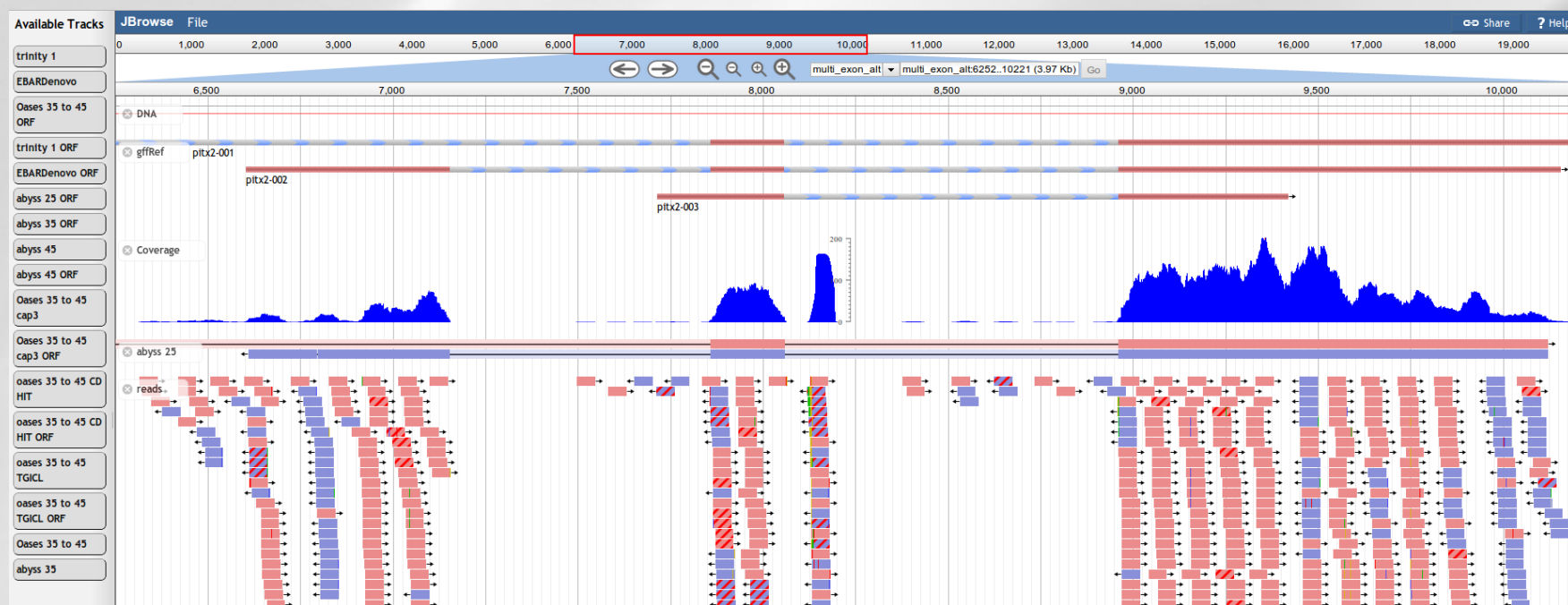
47880 contigs
Total 93 Mb
Longest 94 kb
Shortest 10 b
N50 2622
L50 10495

Realignment metrics

The assembly is a sum-up. The realignment rate gives how much of the initial information is inside the contigs.

Reads mapped back to transcripts (RMBT)

- align reads against assembly generated transcripts
- compute percentage of reads mapped

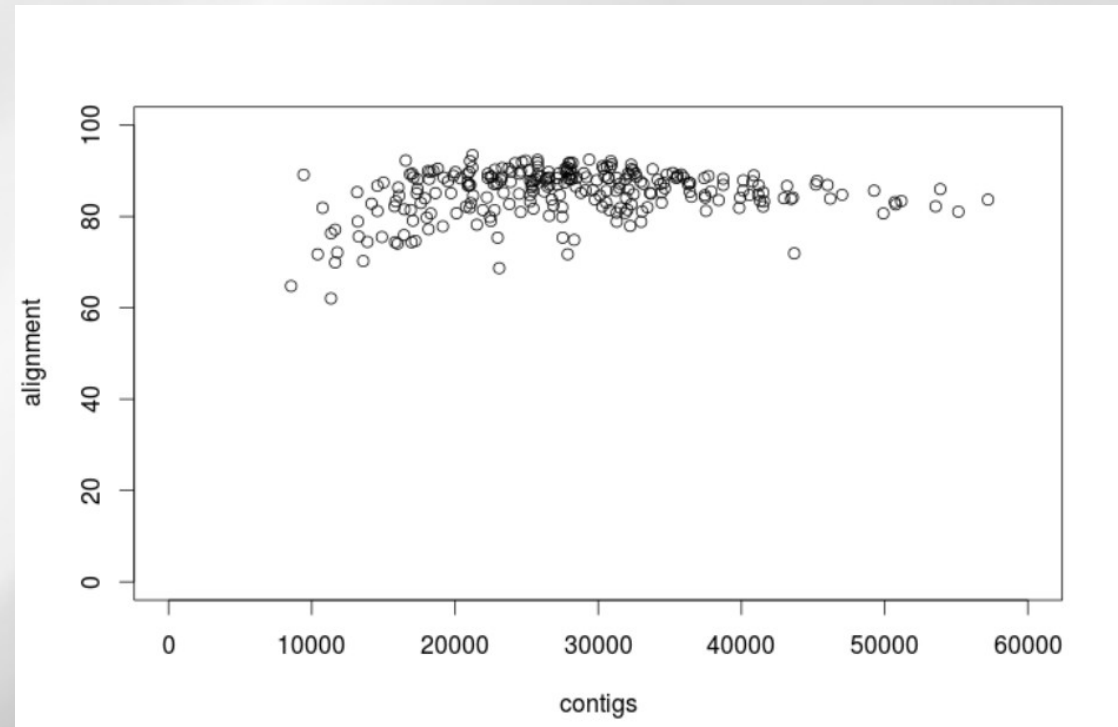


Realignment metrics

Factors affecting realignment rate:

- Presence of highly expressed genes
- Contamination by building blocks (adaptors)
- Reads quality

Should be higher or
around 80% of mapped
reads



RMBT with BWA

The realignment steps are:

- indexing the reference (bwa)
- aligning the reads, producing a sam file (bwa)
- compressing, sorting and indexing the sam file in a bam file (samtools)
- counting the aligned reads (samtools):
 - * global alignment rate
 - * num reads / contig

RMBT: command lines

Index the reference

```
bwa index -a [is|bwtsv] reference.fa
```

Align the reads

```
bwa aln -f R1.sai reference.fa R1.fastq.gz
```

```
bwa aln -f R2.sai reference.fa R2.fastq.gz
```

```
bwa sampe -f output.sam reference.fa R1.sai R2.sai R1.fastq.gz R2.fastq.gz
```

Compress, sort and index

```
samtools view -bS output.sam | samtools sort - output.sorted
```

```
samtools index output.sorted.bam
```

Count reads

```
samtools flagstat output.sorted.bam
```

```
samtools idxstats output.sorted.bam
```

Or use `/home/sigenae/bin/runSampe` 😊

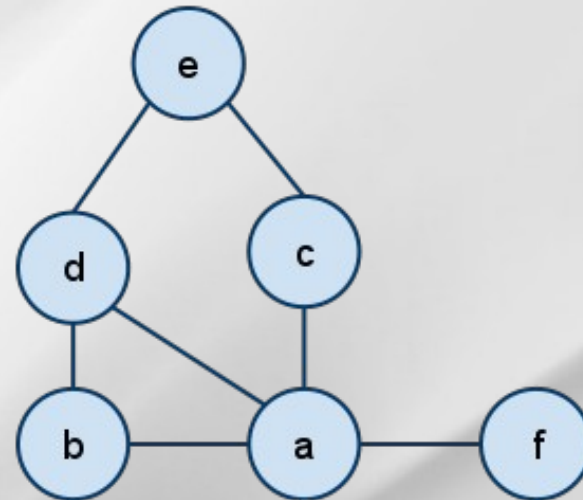


Exercice n°1

Lets assemble!

Assembly: vocabulary

- **Graph:** a data structure which consists of a finite set of ordered pairs, called edges or arcs, of certain entities called nodes or vertices
- a simple graph example:
 - * 6 nodes
 - * 7 edges



- **Path:** in a graph, a path is a sequence of edges which connect a sequence of vertices

Assembly: vocabulary

- **K-mer**: a k-mer is a sub-string of length k
- A string of length L has $(L-k+1)$ k-mers
- Example: a read with $L = 8$ has 5 k-mers when $k = 4$

```

- A G A T C C G T
- A G A T
-   G A T C
-     A T C C
-       T C C G
-         C C G T
    
```

Sequence clustering

Inclusion reduces the overall size of the dataset without removing any sequence information by only removing “redundant” (or highly similar) sequences.

Tools:

- CD-HIT
- BLASTClust
- UCLUST
- UICluster

CD-HIT is a widely used program for clustering and comparing protein or nucleotide sequences.

BIOINFORMATICS APPLICATIONS NOTE Vol. 22 no. 13 2006, pages 1658–1659
doi:10.1093/bioinformatics/btl158

Sequence analysis

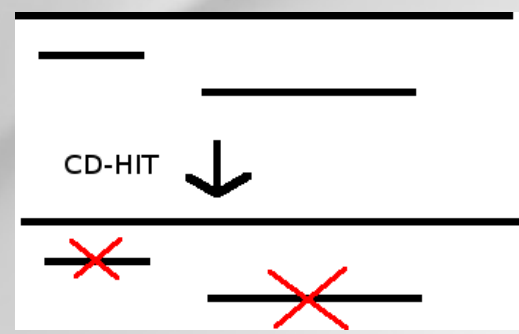
Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences

Weizhong Li* and Adam Godzik

Bumham Institute for Medical Research, La Jolla, CA 92037, USA

Received on March 23, 2006; revised and accepted on April 20, 2006
Advance Access publication May 26, 2006
Associate Editor: Golan Yona

Overview:
CD-HIT clusters included sequences.



CD-HIT uses greedy incremental clustering algorithm method:

- sequences are first sorted in order of decreasing length
- the longest one becomes the representative of the first cluster
- each remaining sequence is compared to the representatives of existing clusters
- if the similarity with any representative is above a given threshold → grouped into that cluster
- if not, a new cluster is defined

Cd-hit: command line

```
bash-4.1$ cd-hit-est
===== CD-HIT version 4.6 (built on May  2 2012) =====

Usage: cd-hit-est [Options]

Options

-i  input filename in fasta format, required
-o  output filename, required
-c  sequence identity threshold, default 0.9
    this is the default cd-hit's "global sequence identity" calculated as:
    number of identical amino acids in alignment
    divided by the full length of the shorter sequence
-G  use global sequence identity, default 1
    if set to 0, then use local sequence identity, calculated as :
    number of identical amino acids in alignment
    divided by the length of the alignment
    NOTE!!! don't use -G 0 unless you use alignment coverage controls
    see options -aL, -AL, -aS, -AS
-b  band_width of alignment, default 20
-M  memory limit (in MB) for the program, default 800; 0 for unlimited;
-T  number of threads, default 1; with 0, all CPUs will be used
-n  word_length, default 10, see user's guide for choosing it
-l  length of throw_away_sequences, default 10
-d  length of description in .clstr file, default 20
    if set to 0, it takes the fasta defline and stops at first space
-s  length difference cutoff, default 0.0
    if set to 0.9, the shorter sequences need to be
    at least 90% length of the representative of the cluster
```

```
cd-hit-est -i input.fa -o output.fa -M 0 -d 0 -c 0.98 -T 8 > output.log
```

Cd-hit: output

```
cd-hit-est -i transcripts.fa -o all_seq.fa -M 0 -d 0 -c 0.98 -T 8 > transcripts.fa.cd-hit.log
```

```
bash-4.1$ grep -c '^>' transcripts.fa all_seq.fa
transcripts.fa:177687
all_seq.fa:35450
```

```
bash-4.1$ sed -n '60,$p' all_seq.fa.clstr | head -20
>Cluster 1
0      16528nt, >k37_Locus_3131_Transcript_3... *
1      16528nt, >k31_Locus_3279_Transcript_3... at +/100.00%
2      16479nt, >k43_Locus_3102_Transcript_1... at -/100.00%
3      16528nt, >k49_Locus_3163_Transcript_1... at +/100.00%
4      16528nt, >k55_Locus_3311_Transcript_1... at -/100.00%
5      14091nt, >k61_Locus_3512_Transcript_1... at -/100.00%
6      2497nt, >k61_Locus_4997_Transcript_1... at -/100.00%
7      14091nt, >k65_Locus_3683_Transcript_1... at -/100.00%
8      2497nt, >k65_Locus_5316_Transcript_1... at -/100.00%
9      11230nt, >k69_Locus_3750_Transcript_1... at -/100.00%
10     2497nt, >k69_Locus_5474_Transcript_1... at -/100.00%
11     2927nt, >k69_Locus_5643_Transcript_1... at -/100.00%
>Cluster 2
0      15727nt, >k31_Locus_124_Transcript_1... *
>Cluster 3
0      15481nt, >k25_Locus_2766_Transcript_1... at -/99.97%
1      181nt, >k31_Locus_10917_Transcript_1... at -/100.00%
2      15675nt, >k37_Locus_3865_Transcript_3... *
3      15494nt, >k31_Locus_4017_Transcript_3... at +/99.94%
```

Assembly introduction

Three classes of methods:

- Greedy method
- Overlap Layout Consensus (OLC) method
- de Bruijn graph (DBG) method

Greedy method

Greedy method joins a read with another read that has the best overlap score until no more reads can be joined.

Overview:

- calculate pairwise alignments of all reads
- score and sort alignments (length/matching)
- merge the two reads with the highest scoring overlap and add the resulting “contig” to the pool of sequences
- continue to extend a contig until no more quality overlaps exist

Greedy:

- optimize a local objective function (quality of the overlap)
- approach that may not lead to a globally optimal solution

SSAKE / VCAKE

VCAKE is a modification of simple k-mer extension (SSAKE) that overcomes error by using high depth coverage

BIOINFORMATICS APPLICATIONS NOTE Vol. 23 no. 4 2007, pages 500–501
doi:10.1093/bioinformatics/btl629

Genome analysis

Assembling millions of short DNA sequences using SSAKE

René L. Warren*, Granger G. Sutton¹, Steven J. M. Jones and Robert A. Holt
British Columbia Cancer Agency, Genome Sciences Centre, 675 West 10th Avenue, Vancouver, BC V5Z 1L3, Canada and ¹J. Craig Venter Institute, 9704 Medical Center Drive, Rockville, MD 20850, USA

Received on October 6, 2006; revised on November 15, 2006; accepted on December 5, 2006

Advance Access publication December 8, 2006

Associate Editor: Alex Bateman

BIOINFORMATICS APPLICATIONS NOTE Vol. 23 no. 21 2007, pages 2942–2944
doi:10.1093/bioinformatics/btm451

Genome analysis

Extending assembly of short DNA sequences to handle error

William R. Jeck^{1,*}, Josephine A. Reinhardt¹, David A. Baltrus¹, Matthew T. Hickenbotham², Vincent Magrini², Elaine R. Mardis², Jeffery L. Dangl^{1,3} and Corbin D. Jones^{1,3}

¹Department of Biology, University of Carolina—Chapel Hill, Chapel Hill, NC 27599, ²Department of Genetics, Washington University School of Medicine, St. Louis, MO 63108 and ³Carolina Center for Genome Sciences, University of Carolina—Chapel Hill, Chapel Hill, NC 27599, USA

Received on July 23, 2007; revised on August 21, 2007; accepted on August 24, 2007

Advance Access publication September 24, 2007

Associate Editor: Alex Bateman

Each possible 3' most k-mer from seed reads are used to browse a prefix tree organizing reads by their first eleven 5' end bases.

SSAKE / VCAKE

Hash table keyed by num occurrences

- 96 ATGTCTTCCTATCGTGTA
- 89 TGTAGCGCTATCGTCAAG
- 67 GTCATGTCGTATTTTGTA
- 42 CGATCGATGCTAGTATAT

Each 3' most k-mer

ATGTCTTCCTATCGTGTA

TGTCTTCCTATCGTGTA

GTCTTCCTATCGTGTA

TCTTCCTATCGTGTA

CTTCCTATCGTGTA

TTCCTATCGTGTA

TCCTATCGTGTA

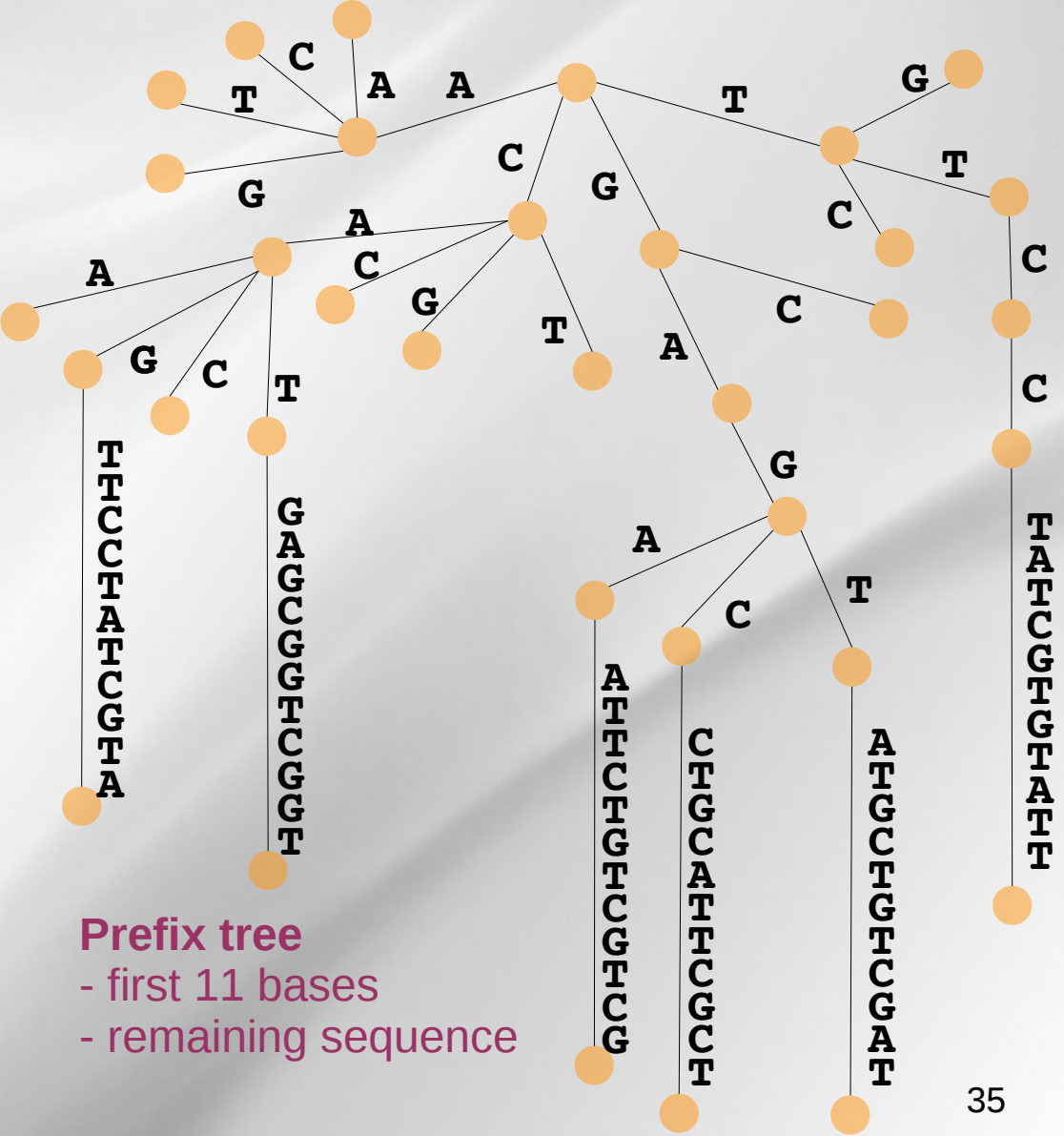
CCTATCGTGTA

CTATCGTGTA

TATCGTGTA

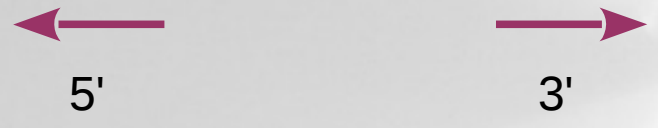
ATCGTGTA

Min length = 8



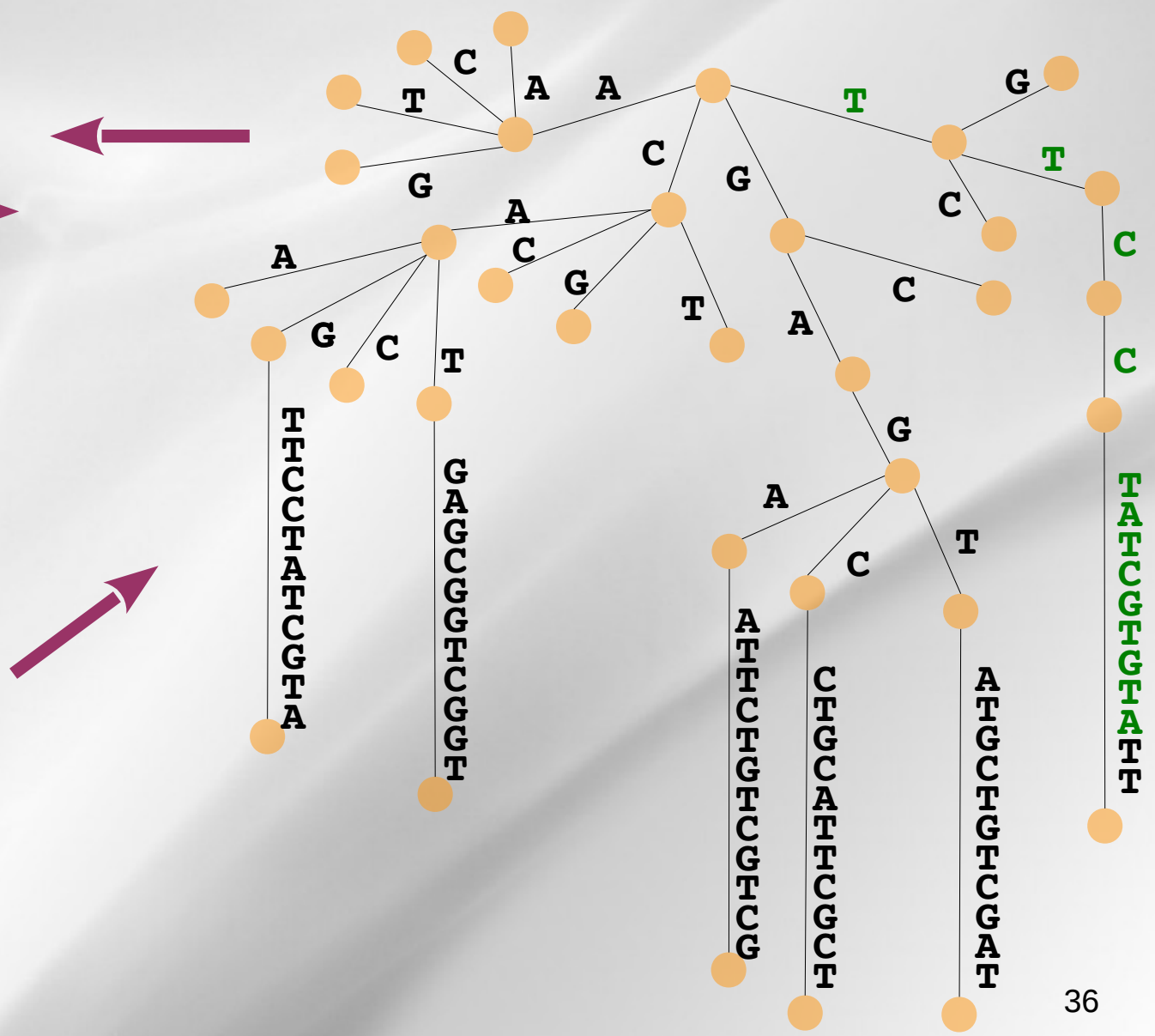
SSAKE / VCAKE

ATGCTTCCTATCGTGATT



- Each 3' most k-mer
- ATGCTTCCTATCGTGTA
- TGCTTCCTATCGTGTA
- GTCTTCCTATCGTGTA
- TCTTCCTATCGTGTA
- CTTCCTATCGTGTA
- TTCCTATCGTGTA
- TCCTATCGTGTA
- CCTATCGTGTA
- CTATCGTGTA
- TATCGTGTA
- ATCGTGTA

Min length = 8

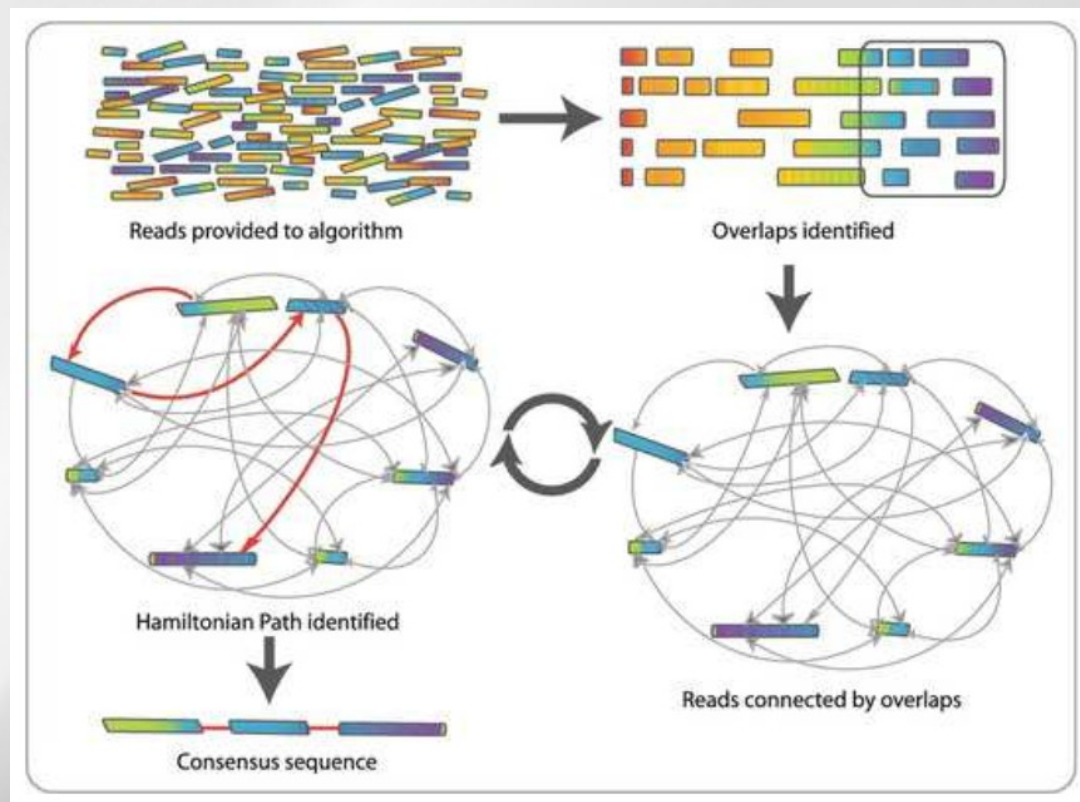


OLC method

OLC method generates a graph using reads and overlaps. The assembly process becomes synonymous with finding a pathway through the graph that visits every node at exactly once.

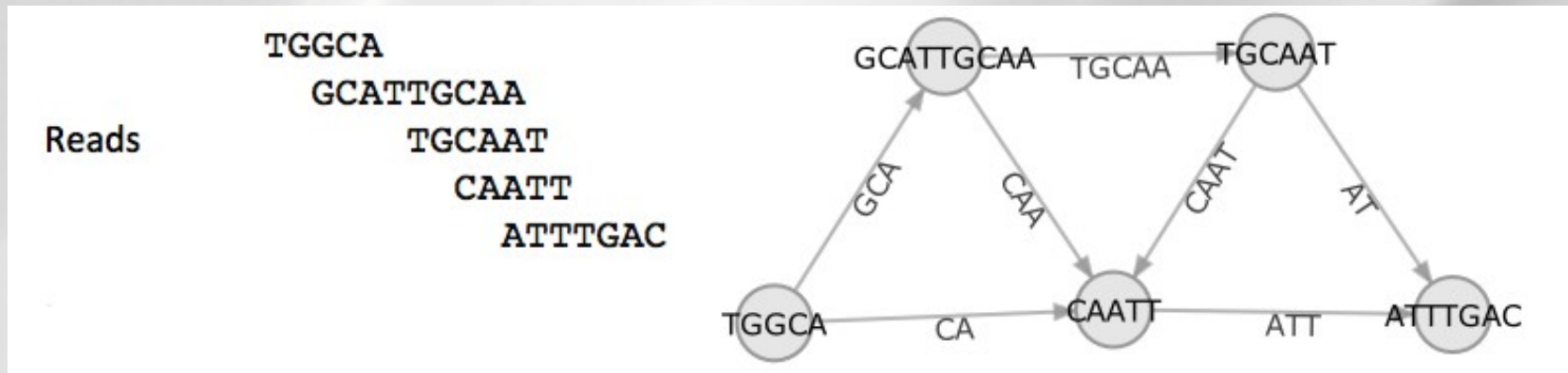
Tree steps:

- Overlap
- Layout
- Consensus



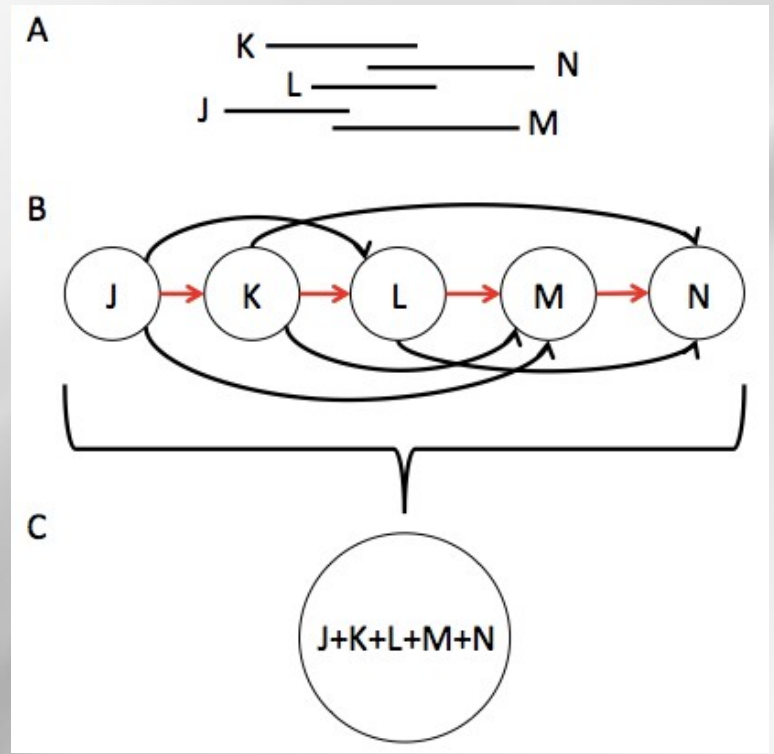
Overlap

- Each read is compared to every other reads in both the forward and reverse complement orientations
- Different OLC algorithms have different criteria for OLC-quality overlaps
- In the assembly graph, the nodes represent actual reads, the edges represent overlaps between these reads



Layout: simplify graph

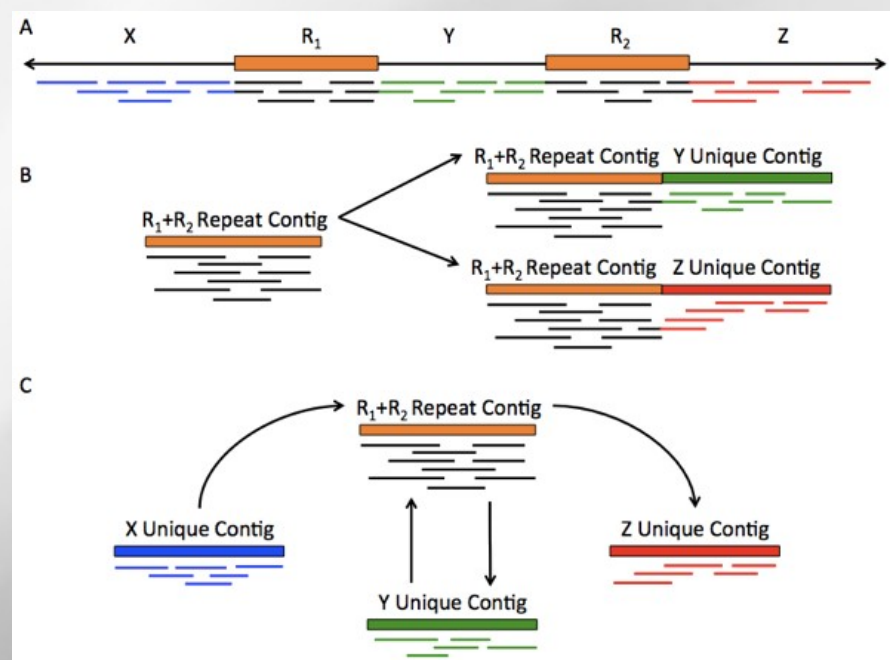
- In order to decrease the size of the graph, the OLC assembly graph is simplified in the layout stage
- There is one path that visits every read, highlighted in red
- Nodes are compressed into contigs until a fork is reached



Layout: resolve repeats

Forks typically signify the boundary between repeats and unrepeatd segments

- a fork is formed because the reads that link the R1+R2 contig to Y and Z do not overlap on the suffix end
- both repeat sections R1 and R2 are compressed into a repeat contig. X, Y and Z are compressed into unique contigs

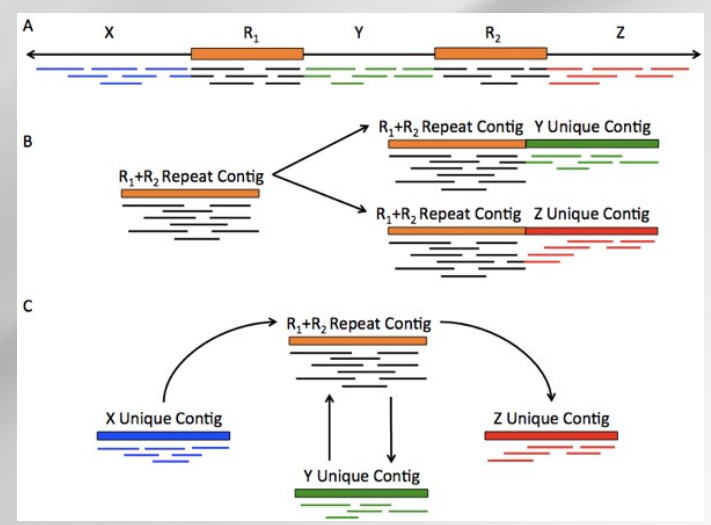


Consensus


- After contig generation, consensus sequences are derived
- Starting from the left most read of each contig, the OLC algorithm computes the consensus of all of the reads composing each contig

Greedy vs OLC

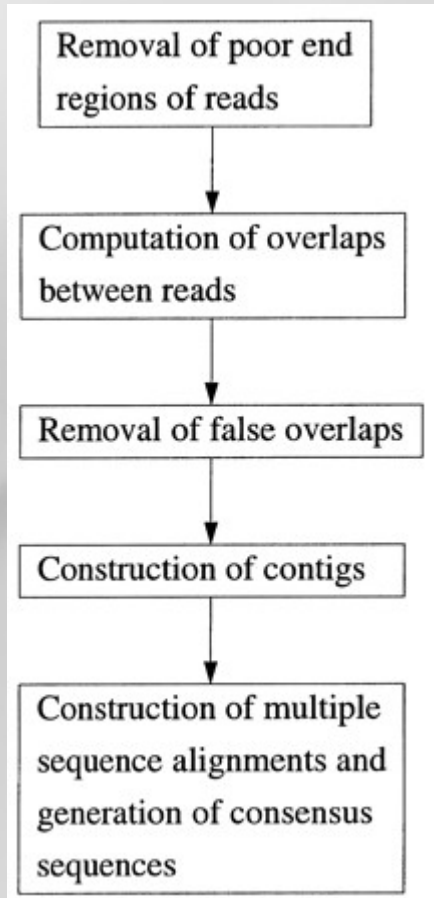
- Both begin with overlap generation
- Steps of OLC enable a global analysis of the assembly problem
- Local analysis of greedy algorithms is a limit
- For this repeat example:
 - * the proper reconstruction $X-R_1+R_2-Y-R_1+R_2-Z$ is easily inferred using the OLC method
 - * a greedy extension would produce
 - fragmented assembly ($X-R_1+R_2$; Y ; Z)
 - misassembly ($X-R_1+R_2-Z$; Y)



CAP3 (CONTIG ASSEMBLY PROGRAM Version 3) is a sequence assembly program for small-scale assembly with or without quality



CAP3: A DNA Sequence Assembly Program
Xiaoqiu Huang and Anup Madan
Genome Res. 1999 9: 868-877
Access the most recent version at doi:[10.1101/gr.9.9.868](https://doi.org/10.1101/gr.9.9.868)



TGICL is a pipeline for analysis of large Expressed Sequence Tags (EST) and mRNA databases

- MegaBLAST
- Clustering
- Large clusters splitting
- CAP3

BIOINFORMATICS APPLICATIONS NOTE Vol. 19 no. 5 2003, pages 651–652
DOI: 10.1093/bioinformatics/btg034



TIGR Gene Indices clustering tools (TGICL): a software system for fast clustering of large EST datasets

Geo Pertea^{1,*}, Xiaoqiu Huang², Feng Liang^{1,3},
Valentin Antonescu¹, Razvan Sultana¹, Svetlana Karamycheva¹,
Yuandan Lee¹, Joseph White¹, Foo Cheung¹, Babak Parvizi¹,
Jennifer Tsai¹ and John Quackenbush^{1,*}

¹The Institute for Genomic Research, Rockville, MD 20850 USA, ²Department of Computer Science, Iowa State University, Ames, IA 50011 USA and ³Current address Invitrogen Corporation, Carlsbad, CA 92008 USA

Received on July 10, 2002; revised on August 12, 2002; accepted on August 16, 2002

TGICL benefit:

- mgbblast (MegaBLAST with new filtering options)
→ compressed sorted file of decreasing pairwise alignment score
- clustering (tclust, sclust, nrcl) → generally a very large connected component
- splitting → partitioning attempt based on full-length transcripts

TGICL: command line

TGICL(1) User Contributed Perl Documentation TGICL(1)

TGICL

The Gene Index Clustering Tool (TGICL) is a software package that tries to efficiently cluster and create assemblies (contigs) from a set of input DNA sequences given in a fasta file. The "clustering" phase is intended to partition the input data set into smaller groups of sequences (clusters) that due to stringent similarity have a greater probability to originate from the same longer sequence. However, the clustering phase does not perform any multiple alignment but only fast pairwise alignments (using megablast), which are then filtered and used to build subsets of sequences by a transitive closure approach. In the assembly phase each such cluster is sent to the assembly program (cap3) which attempts the multiple alignment of the sequences in the cluster and creates one or more contigs (consensus sequences). Both clustering and assembly phases can be executed in parallel on multiple CPU machines or in a PVM environment.

VERSION

Version 2.1

SYNOPSIS

```
tgicl -F <fasta_db> [-q <qualdb>] [-d <refDb>] [-c {<num_CPUs>|<PVM_nodefile>}]
  [-m <user>] [-O 'cap3_options'] [-l <min_overlap>] [-v <max_overhang>]
  [-p <pid>] [-n <slicesize>] [-s <maxsize>] [-a <cluster_file>] [-M] [-K]
  [-L] [-X] [-I] [-C] [-G] [-R] [-W <pairwise_script.psx>] [-A <asm_program.psx>]
  [-b user:pass:driver:server:schema] [-P <param_file>] [-u <seq_list>] [-f <prefix_filter>] [-D]
```

To start clustering and assembling all the sequences from fasta file myseq.fasta using 2 CPUs :

```
tgicl -F myseq.fasta -c 2
```

To start the same process loading the results into a MySQL database and generating some static HTML file reports.

```
tgicl -F myseq.fasta -c 2 -b tgiuser:tgipass:mysql:dbserver.my.domain:tgicldb -R
```

In the previous example the database username is tgiuser, the password is tgipass, the requested database is of course mysql, the database is located on server dbserver.my.domain and the db is tgicldb.

```
tgicl -F input.fa -c 2 -l 60 -p 96
```

TGICL: output

```
tgicl -F all_seq.fa -c 2 -l 60 -p 96
```

```
bash-4.1$ ll
total 13392
-rw-rw-r-- 1 sigenae sigenae 10312464 30 oct. 16:00 all_seq.fa
-rw-rw-r-- 1 sigenae sigenae 297172 30 oct. 16:00 all_seq.fa.cidx
-rw-rw-r-- 1 sigenae sigenae 4003 30 oct. 16:01 all_seq.fa_cl_clusters
-rw-rw-r-- 1 sigenae sigenae 1926 30 oct. 16:01 all_seq.fa_cl_tabhits_001.Z
-rw-rw-r-- 1 sigenae sigenae 228976 30 oct. 16:00 all_seq.fa.nhr
-rw-rw-r-- 1 sigenae sigenae 60064 30 oct. 16:00 all_seq.fa.nin
-rw-rw-r-- 1 sigenae sigenae 2502693 30 oct. 16:00 all_seq.fa.nsq
-rw-rw-r-- 1 sigenae sigenae 136851 30 oct. 16:01 all_seq.fa.singletons
drwxr-x--- 2 sigenae sigenae 16384 30 oct. 16:01 asm_1
drwxr-x--- 2 sigenae sigenae 16384 30 oct. 16:01 asm_2
-rw-rw-r-- 1 sigenae sigenae 2598 30 oct. 16:01 err_tgicl_all_seq.fa.log
-rw-rw-r-- 1 sigenae sigenae 163 30 oct. 16:00 formatdb.log
-rw-rw-r-- 1 sigenae sigenae 0 30 oct. 16:01 masked.lst
-rw-rw-r-- 1 sigenae sigenae 2238 30 oct. 16:01 tgicl_all_seq.fa.log
-rw-rw-r-- 1 sigenae sigenae 3357 30 oct. 16:00 tgicl.cfg
```

```
bash-4.1$ ll asm_1
total 736
-rw-rw-r-- 1 sigenae sigenae 297943 30 oct. 16:01 ACE
-rw-rw-r-- 1 sigenae sigenae 335080 30 oct. 16:01 align
-rw-rw-r-- 1 sigenae sigenae 53919 30 oct. 16:01 contigs
-rw-rw-r-- 1 sigenae sigenae 0 30 oct. 16:01 err_log
-rw-rw-r-- 1 sigenae sigenae 1902 30 oct. 16:01 log_std
-rw-rw-r-- 1 sigenae sigenae 781 30 oct. 16:01 singlets
```


TGICL: ACE format

```

AS 31 115

CO CL1Contig1 286 2 3 U
GAAGGAGGAGAGCGAAGACCTGAGTGAAGATGAGGAGAAAACATCATGTCAAAACTGAAGG
AGAAACTCAGTCAGAGATTGATCATTATCTTTCTCTACAAACAAACACAAACAACACTCA
CAACCTGACGGTGAAGAAAAGAGGAAAAGTGAAGAACTGTGTAAAGACAAGAAAAGTCTATCA
AAGCGTTCAAACAGAAAGTGAAGTCAAATACTTGTCTTTTGTGTGGAAAGACTTTTATAAA
GCCATCGTATTTAAAACGACACCAGAGGACTCACACTGGAGCGAAA

BQ
10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 0 15 15 15 15 15 15 15 15 1
15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15 15
10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10

AF k25_Locus_18748_Transcript_2 U -61
AF k25_Locus_14452_Transcript_9 U 1
BS 1 19 k25_Locus_14452_Transcript_9
BS 20 20 k25_Locus_18748_Transcript_2
BS 21 286 k25_Locus_14452_Transcript_9

RD k25_Locus_18748_Transcript_2 154 0 0
AATCAGAGACTGAACATTTATCTTTTTCTACAATCAAACACACATAACAATTACAATGTGA
TCATGAAGGAGGAGAGTGAAG
CTGAGTGAAGATGAGGAGAAAACATCATGTCAAAACTGAAGGAGAAAACACTCACTCAGAGATT
GATCATTATCTT

QA 82 154 82 154
DS

RD k25_Locus_14452_Transcript_9 286 0 0
GAAGGAGGAGAGCGAAGACCTGAGTGAAGATGAGGAGAAAACATCATGTCAAAACTGAAGG
AGAAACTCAGTCAGAGATTGATCATTATCTTTCTCTACAAACAAACACAAACAACACTCA
CAACCTGACGGTGAAGAAAAGAGGAAAAGTGAAGAACTGTGTAAAGACAAGAAAAGTCTATCA
AAGCGTTCAAACAGAAAGTGAAGTCAAATACTTGTCTTTTGTGTGGAAAGACTTTTATAAA
GCCATCGTATTTAAAACGACACCAGAGGACTCACACTGGAGCGAAA

QA 1 286 1 286
DS
    
```

TGICL: Align format

```

***** CL1Contig13 *****
k55_Locus_14801_Tran+ CGTTCATTTCAGCGTCGAGTCTTCAGAGCTGAGCTCTGCCGCCATCAGTGAAAAGCCAAAAGA
consensus           CGTTCATTTCAGCGTCGAGTCTTCAGAGCTGAGCTCTGCCGCCATCAGTGAAAAGCCAAAAGA

k55_Locus_14801_Tran+ CAGAGATTATTGATGGACAGTGAAGAAGATGAGTGTCCAGAACCCCTGCAAAATCAAACAG
k37_Locus_13099_Tran+ TTATTGATGGACAGTGAAGAAGATGAGTGTCCAGAACCCCTGCAAAATCAAACAG
k69_Locus_1514_Tran+  ATTGAAGGACAGTGAAGAAGATGAGTGTCCAGAACCCCTGCAAAATCAAACAG
consensus           CAGAGATTATTGATGGACAGTGAAGAAGATGAGTGTCCAGAACCCCTGCAAAATCAAACAG

k55_Locus_14801_Tran+ GAAGACACTGAAGAACTAATAGATGTGAAGGAGGAGAGTGAAGAACTGAGTGAAGATGAG
k37_Locus_13099_Tran+ GAAGACACTGAAGAACTAATAGATGTGAAG
k69_Locus_1514_Tran+  GAAGACACTGAAGAACTAATAGATGTGAAGGAGGAGAGTGAAGAACTGAGTGAAGATGAG
consensus           GAAGACACTGAAGAACTAATAGATGTGAAGGAGGAGAGTGAAGAACTGAGTGAAGATGAG

k55_Locus_14801_Tran+ GAGAAACATCAGGTCA
k69_Locus_1514_Tran+ GAGAAACATCAGGTCAAAAAGTGAAGAAGAACTCAATCAGAGACTGAACATTTATCTTTT
k37_Locus_16908_Tran+ GTCAAAAGTGGAGAAGAACTCAATCAGAGACTGAACATTTATCTTTT
consensus           GAGAAACATCAGGTCAAAAAGTGAAGAAGAACTCAATCAGAGACTGAACATTTATCTTTT

k69_Locus_1514_Tran+ CTACAATCAAACACACATAACAATTACAATGTGATCATGAAGGAGGAGAGTGAAGAAGTGA
k37_Locus_16908_Tran+ CTACAATCAAACACACATAACAATTACAATGTGATCATGAAGGAGGAGAGTGAAGAAGTGA
k55_Locus_11728_Tran+  GTGAAGGAGGAGAGTGAAGAAGTGA
consensus           CTACAATCAAACACACATAACAATTACAATGTGATCATGAAGGAGGAGAGTGAAGAAGTGA

k69_Locus_1514_Tran+ AATGAAGATGAGGAGAAACATCAGGTCAAAAAGTGAAGAAGAACTCAATCAGAGACTGAA
k37_Locus_16908_Tran+ AATGAAGATGAGGAGAAACATCAGGTCAAAAAGTGAAGA
k55_Locus_11728_Tran+  AGTGAAGATGAGGAGAAACATCAGGTCAAAAAGTGGAGAAGAACTCAATCAGAGACTGAA
consensus           AATGAAGATGAGGAGAAACATCAGGTCAAAAAGTGAAGAAGAACTCAATCAGAGACTGAA

k69_Locus_1514_Tran+ CATTTATCTTCTCTACAAAACAAAACAAAACAACTCACAACCTGACGGTGAAGAAAAGAG
k55_Locus_11728_Tran+ CATTTATCTTCTCTACAAAACAAAACAAAACAACTCACAACCTGACGGTGAAGAAAAGAG
consensus           CATTTATCTTCTCTACAAAACAAAACAAAACAACTCACAACCTGACGGTGAAGAAAAGAG

k69_Locus_1514_Tran+ GAAAGTGAAGAAGTGTGTAAGACAAGAAAAGTCTATCAAAGCGTTCAAACAGAACTGAAG
k55_Locus_11728_Tran+ GAAAGTGAAGAAGTGTGTAAGACAAGAAAAGTCTATCAAAGCGTTCAAACAGAACTGAAG
consensus           GAAAGTGAAGAAGTGTGTAAGACAAGAAAAGTCTATCAAAGCGTTCAAACAGAACTGAAG

k69_Locus_1514_Tran+ TCAAATACTTGCTCTTTGTGTGAAAAGACTTTTATAAAGCCATCGTATTTAAAACGACAC
k55_Locus_11728_Tran+ TCAAATACTTGCTCTTTGTGTGAAAAGACTTTTATAAAGCCATCGTATTTAAAACGACAC
consensus           TCAAATACTTGCTCTTTGTGTGAAAAGACTTTTATAAAGCCATCGTATTTAAAACGACAC
    
```

Other OLC assemblers

BIOINFORMATICS ORIGINAL PAPER Vol. 24 no. 24 2008, pages 2818–2824
doi:10.1093/bioinformatics/btn548

Genome analysis

Aggressive assembly of pyrosequencing reads with mates

Jason R. Miller^{1,*}, Arthur L. Delcher², Sergey Koren¹, Eli Venter¹, Brian P. Walenz¹, Anushka Brownley¹, Justin Johnson¹, Kelvin Li¹, Clark Mobarry³ and Granger Sutton¹

¹The J. Craig Venter Institute, 9712 Medical Center Drive, Rockville MD 20850, ²Center for Bioinformatics & Computational Biology, University of Maryland, College Park, MD 20742 and ³White Oak Technologies Inc, 1300 Spring St., Ste 320, Silver Spring, MD 20910, USA

Received on June 20, 2008; revised on October 17, 2008; accepted on October 20, 2008
Advance Access publication October 24, 2008
Associate Editor: Dmitrij Frishman

CABOG
Celera Assembler with
the Best Overlap Graph

Newbler

nature Vol 437|15 September 2005|doi:10.1038/nature03959

ARTICLES

Genome sequencing in microfabricated high-density picolitre reactors

Marcel Margulies^{1*}, Michael Egholm^{1*}, William E. Altman¹, Said Attiya¹, Joel S. Bader¹, Lisa A. Bemben¹, Jan Berka¹, Michael S. Braverman¹, Yi-Ju Chen¹, Zhoutao Chen¹, Scott B. Dewell¹, Lei Du¹, Joseph M. Fierro¹, Xavier V. Gomes¹, Brian C. Godwin¹, Wen He¹, Scott Helgesen¹, Chun He Ho¹, Gerard P. Irzyk¹,



Using the miraEST Assembler for Reliable and Automated mRNA Transcript Assembly and SNP Detection in Sequenced ESTs

Bastien Chevreux, Thomas Pfisterer, Bernd Drescher, et al.

Genome Res. 2004 14: 1147-1159
Access the most recent version at doi:10.1101/gr.1917404

MIRA

But we have billions of reads!

De Bruijn graph

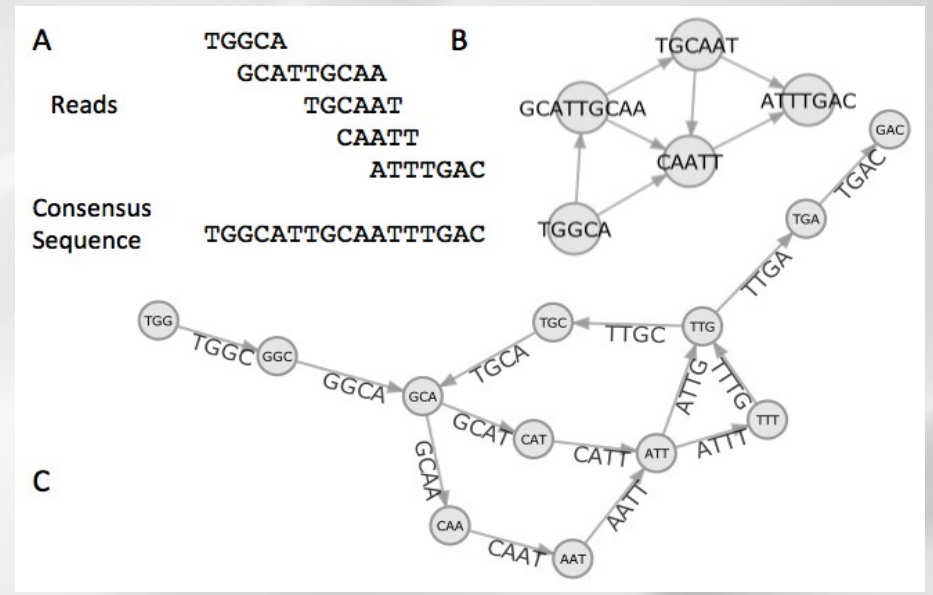
The de Bruijn graph (k-mer graph) approach is more appropriate for the large volumes of reads associated with short-read sequencing:

- avoids the computationally expensive all-against-all pairwise read comparisons
- avoids loading all the replicate sequences associated with high-coverage sequencing

De Bruijn graph

The DBG:

- directed graph
- edges are unique k-mers
- nodes are overlaps of length k-1
- an edge connects two nodes if the suffix of the source node shares an exact match of length k-2 with the prefix of the destination node
- the assembly algorithm becomes finding a path in the graph that visits every edge at least once



De Bruijn graph

Overview of the assembly strategy

a Generate all substrings of length k from the reads

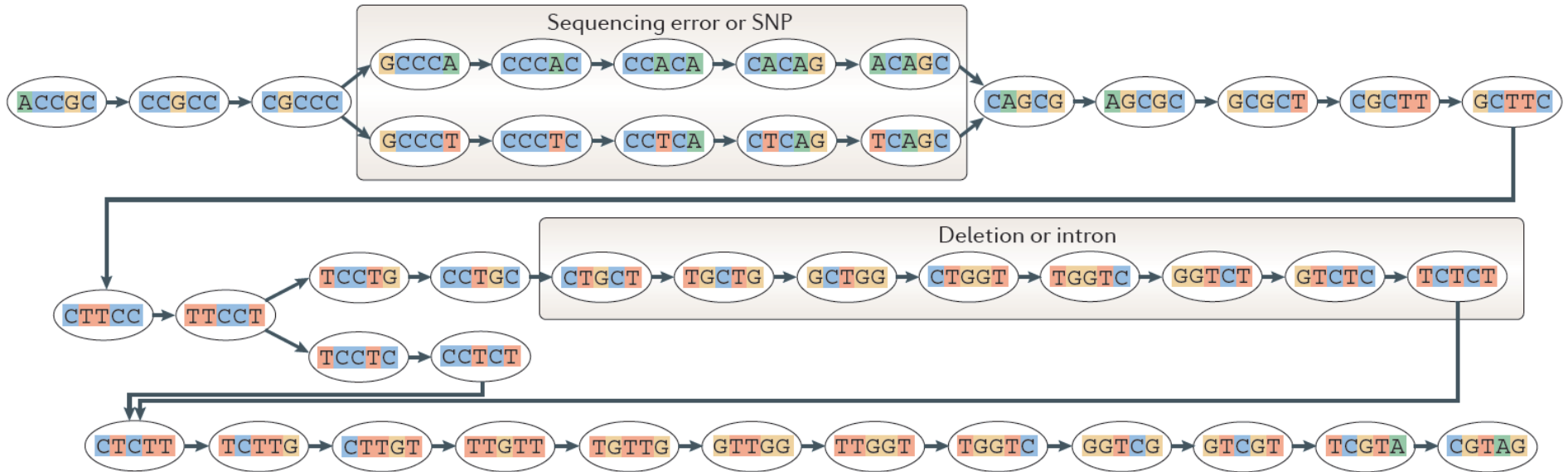


Martin JA, Wang Z. Next-generation transcriptome assembly. Nat Rev Genet. 2011

a Generate all substrings of length k from the reads



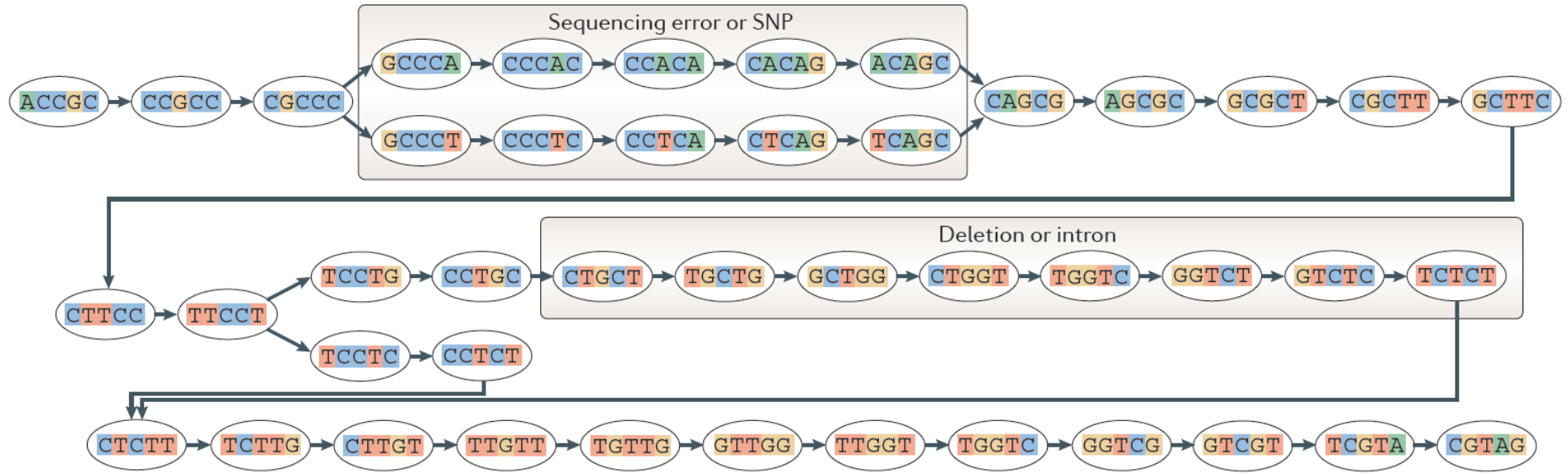
b Generate the De Bruijn graph



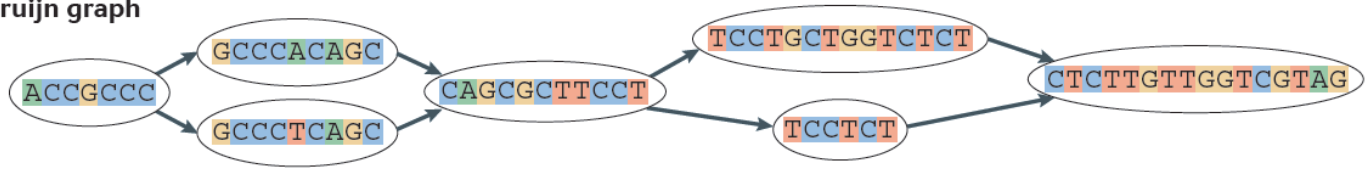
De Bruijn graph

Overview of the assembly strategy

b Generate the De Bruijn graph



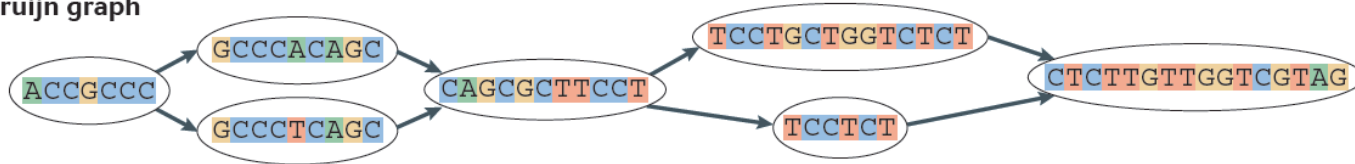
c Collapse the De Bruijn graph



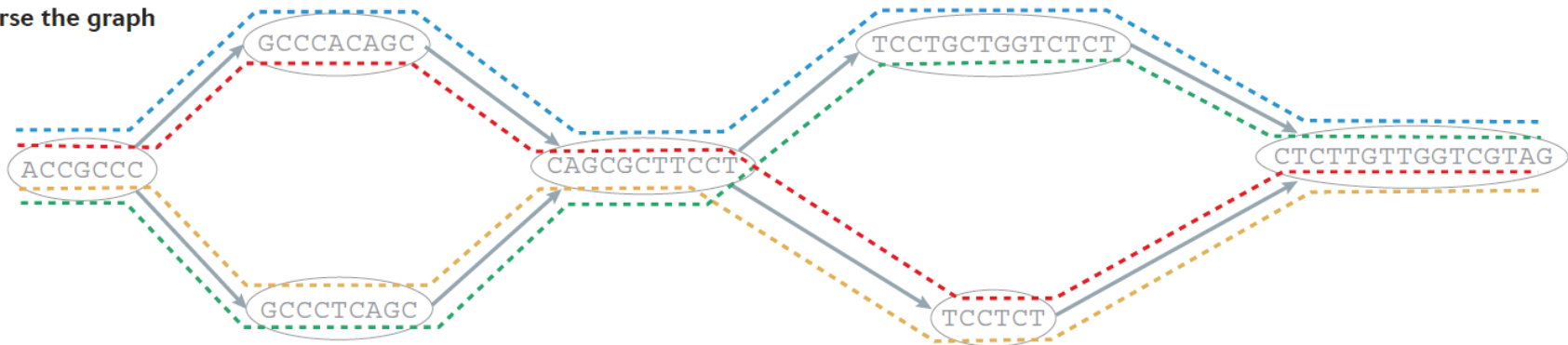
De Bruijn graph

Overview of the assembly strategy

c Collapse the De Bruijn graph

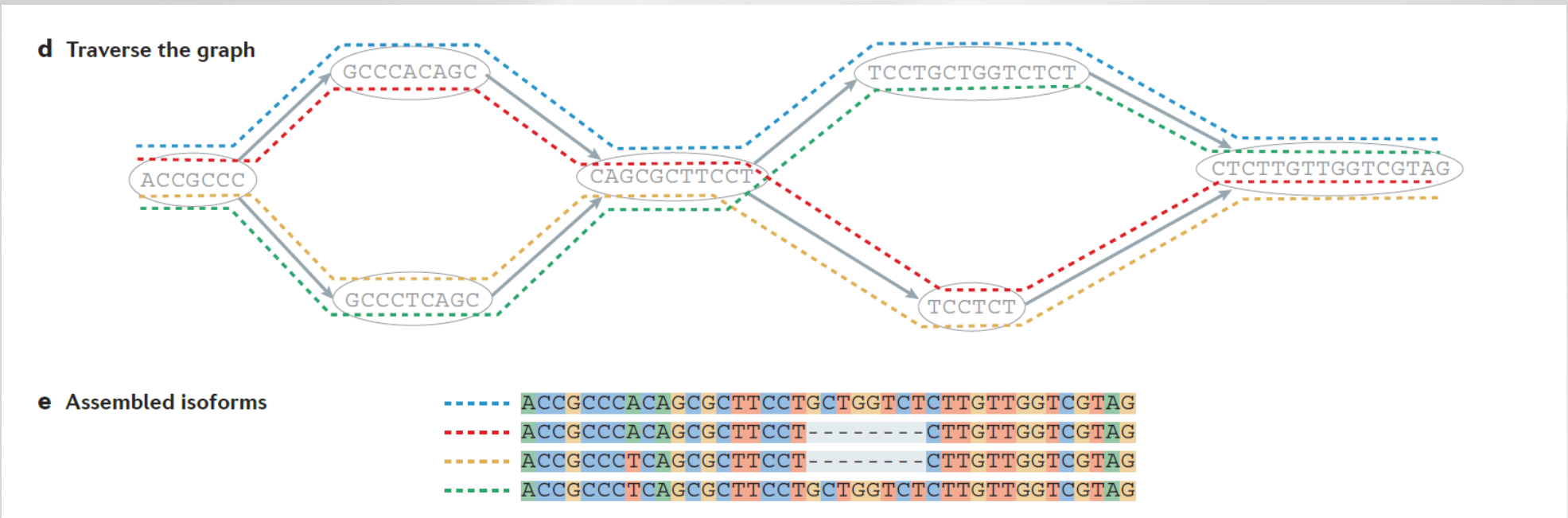


d Traverse the graph



De Bruijn graph

Overview of the assembly strategy



Martin JA, Wang Z. Next-generation transcriptome assembly. Nat Rev Genet. 2011

DBG vs OLC

DBG

- ⊕ Overlap computation is a very time and computationally intensive step ⊗
- ⊕ More efficient ways to find Eulerian paths than Hamiltonian paths ⊗
 - ⊗ Overlaps can vary in length ⊕
 - ⊗ Very sensitive to repeats ⊕
 - ⊗ Very sensitive to sequencing errors ⊕

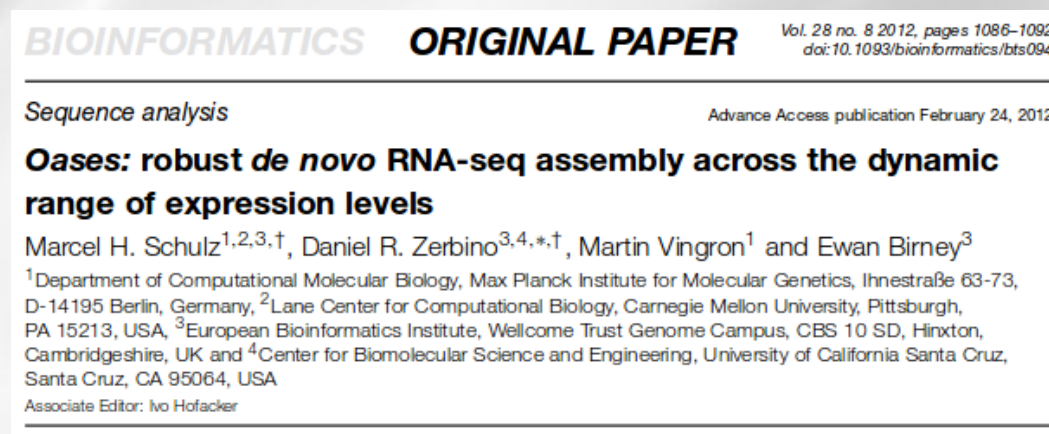
OLC

Velvet / Oases

Velvet: sequence assembler for very short reads (2008)



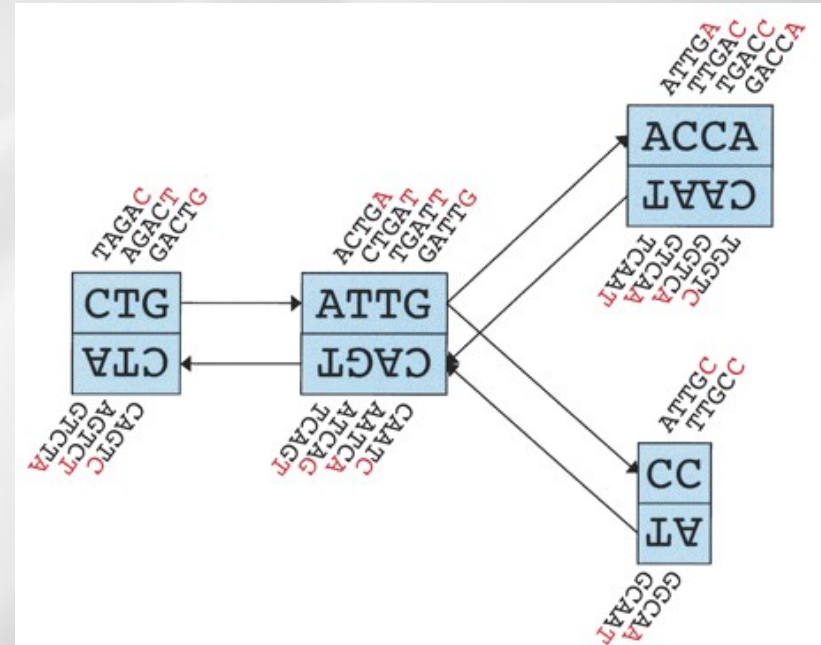
Oases: de novo transcriptome assembler (2012)



Velvet - Graph structure

Graph structure:

- each node represents a series of overlapping k-mers
- sequence of the final nucleotides is called the sequence of the node
- each node is attached to a twin node (rev. comp k-mers → k must be odd)
- nodes are connected by a directed arc. The last k-mer of an arc's source node shares an overlap of length k-1 with the first of the destination node
- reads are mapped as “paths” traversing the graph



Velvet - Graph construction

Graph construction:

- reads are first hashed according to predefined k-mer length
- build an hash table storing for each k-mer, the ID of the first read encountered containing this k-mer and the position
- build another table storing for each read which of its original k-mers are overlapped by subsequent reads
- ordered set of original k-mers of a read is cut each time an overlap with another read begins or ends
- for each uninterrupted sequence of original k-mers, a node is created

Velvet – Correcting graph

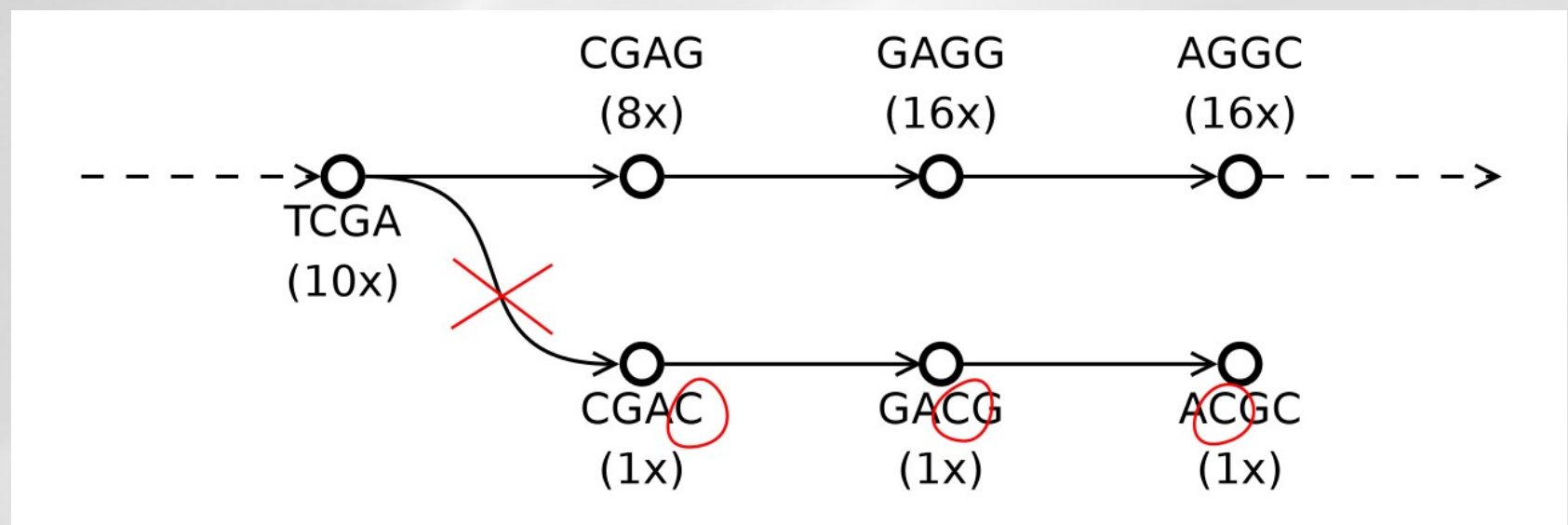
Next steps

- graph simplification → chains of blocks or linear connected subgraph are merged (no loss of information)
- error removal → three major problems:
 - * “false positive graphs” due to errors in reads
 - * “gap problems” due to non-uniform or low coverage → short “dead-end” paths (← larger k)
 - * “branching problems”: k -mers connected to multiple k -mers due to repeat regions or erroneous reads introduce branches in the graph (← smaller k)

Velvet - Removing errors

Removing tips

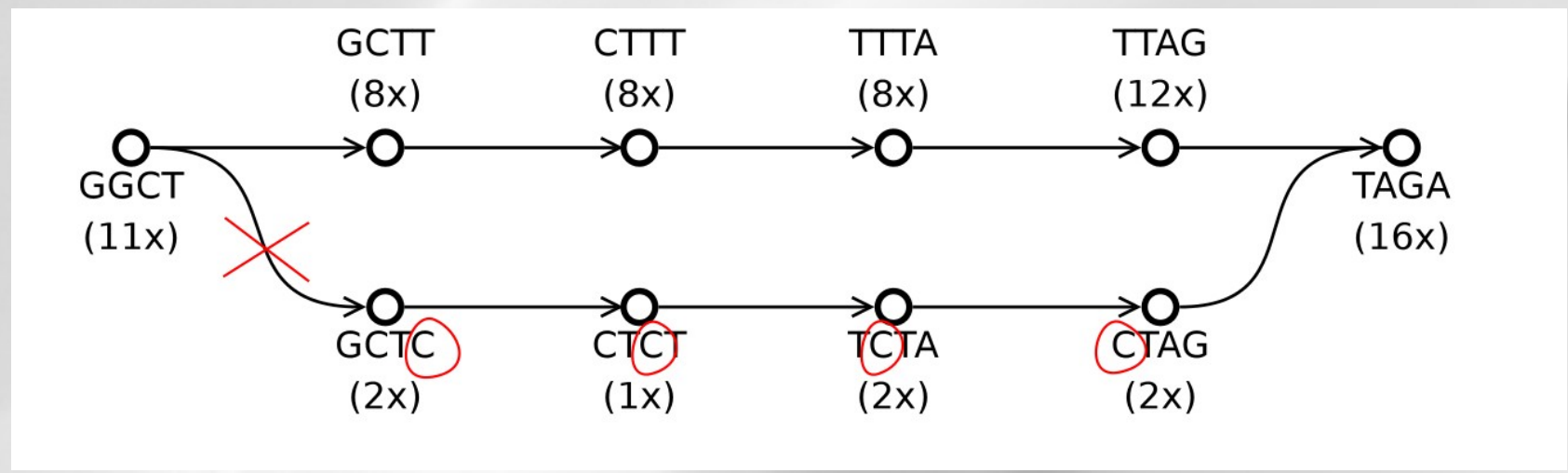
Tip is a chain of nodes that is disconnected on one end. Tips to remove are recognized on the base of two criteria: length and minority count.



Velvet - Removing errors

Removing bubbles (Tour Bus algorithm)

Two paths that start and end at the same nodes and contain similar sequences defined a “bubble”. If judged “similar” enough, the paths are merged. The path that reaches the end node first* is used as the consensus path.



* according to the notion of distance, that considers the path-length and the related multiplicity
 → gives priority to higher coverage paths

Velvet - Removing errors

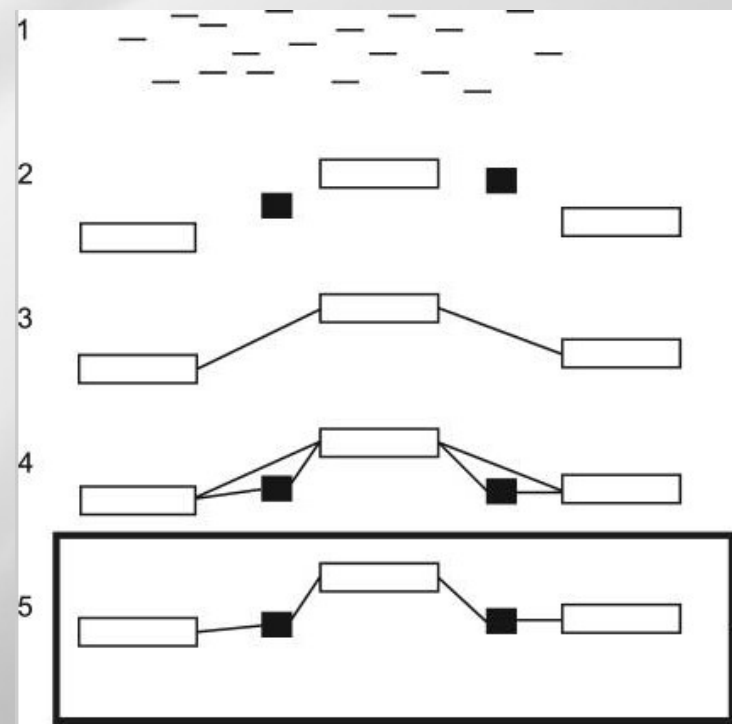
Removing erroneous connections

These unwanted connections do not create any recognizable loop or structure. Remove them with a basic coverage cutoff.

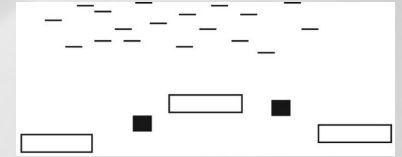
Genome assembly process in Velvet is stopped here → fork due to transcriptome assembly singularities

Overview

- individual reads are sequenced from an RNA sample
- contigs are built from those reads, some of them are labeled as long (clear), others short (dark)
- long contigs, connected by single reads or read-pairs are grouped into connected components called loci
- short contigs are attached to the loci
- the loci are transitively reduced. Transfrags are then extracted from the loci



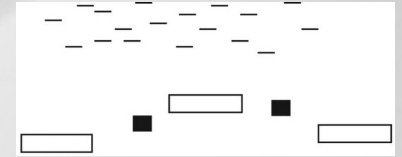
Oases - Contig construction



Contig construction (Velvet preprocess)

- the Oases pipeline receives as input a preliminary assembly produced by Velvet
- initial stages (hashing and graph construction) can be used indifferently on transcriptomic data
- contig correction performed with a modified version of the Tour Bus algorithm (fitted for coverage disparity and complexity of graphs)

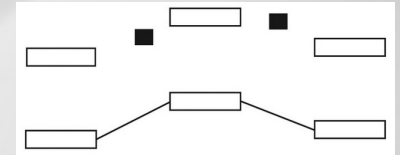
Oases - Contig correction



Contig correction

- local edge removal → for each node, an outgoing edge is removed if its coverage represents less than 10% of the sum of coverages of all outgoing edges)
- static coverage cutoff → contigs with less than the cutoff are removed from the assembly (3x by default)
- contigs longer than a given threshold (by default $> 50+k-1$) are labeled as long contig and treated as if unique
- other contigs are labeled as short

Oases - contigs to scaffolds



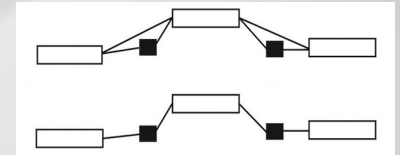
Scaffold construction

- connexion between contigs can be supported by both spanning single reads (direct) or paired-end reads (indirect)
- the total number of spanning reads and pair-reads confirming a connection between 2 contigs is called its support

Scaffold filtering

- based on static (very low) and dynamic (vs local) coverage thresholds
- connections with low support are removed
- short contigs can only be joined by direct connections

Oases - Locus resolution



Locus construction

- contigs are organized into clusters called loci
- two steps:
 - * long contigs are first clustered into connected components
 - * to each locus are added short nodes connected to one of the long nodes in the cluster

Transitive reduction of the loci

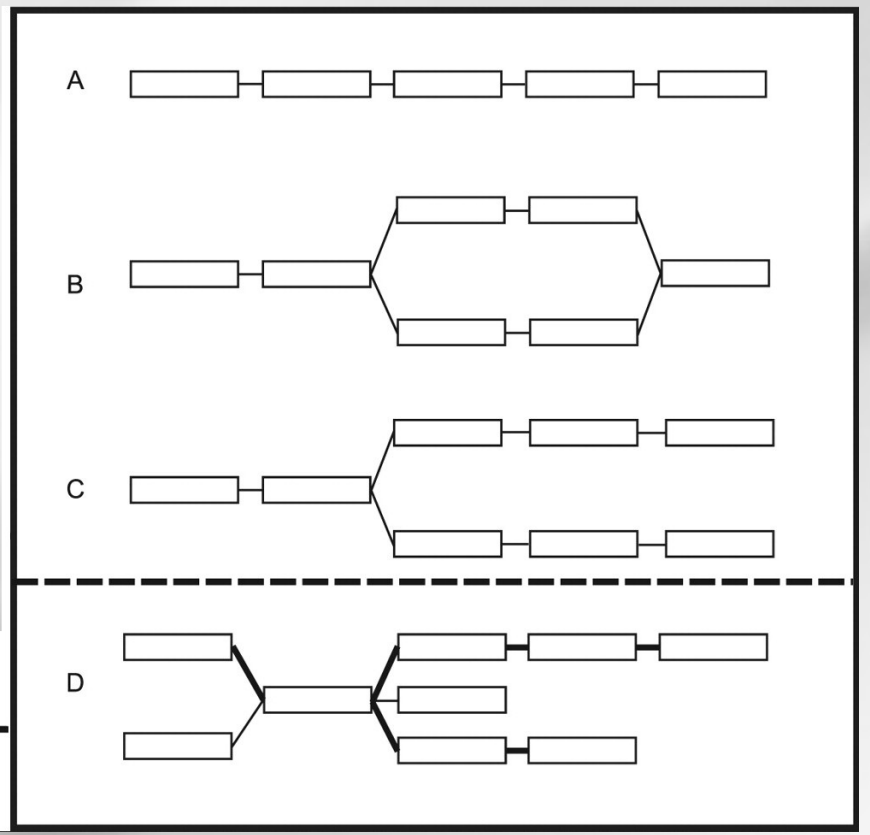
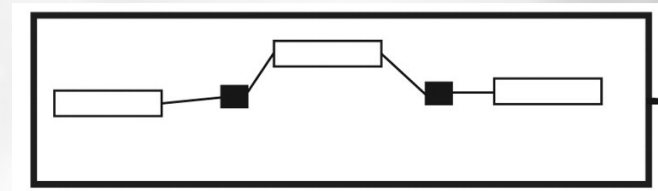
- remove redundant long distance connexions
- example: two contigs which are not consecutive in a locus are frequently connected by a paired-end read

Oases - Transcripts extraction

Extracting transcript assemblies

Loci are divided into four categories

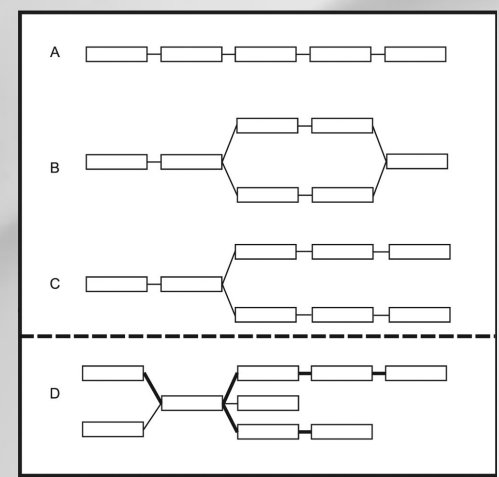
- chains
- bubbles
- forks
- complex



Oases - Transcripts extraction

Extracting transcript assemblies

- trivial locus topologies (chains, forks and bubbles) are straightforward to resolve (if isolated from any other branching point)
- complex loci are processed using an additional heuristic method* which produces a parsimonious set of putative highly expressed transcripts



* Lee C. Generating consensus sequences from partial order multiple sequence alignment graphs. Bioinformatics. 2003

Which k-mer size to choose?

K-mer size is the parameter the most influent upon assembly results.

It's a trade-off between specificity and sensitivity.

Longer k-mers bring you more specificity:

- inherently rarer
- give graphs with lower coverage over all
- the longer the k-mer the more likely it is that it includes an error

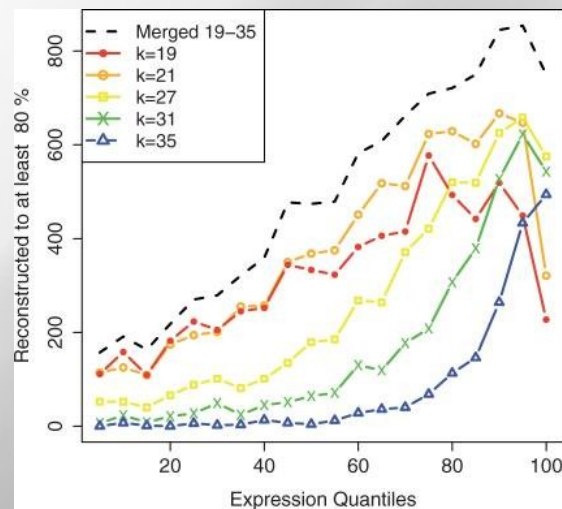
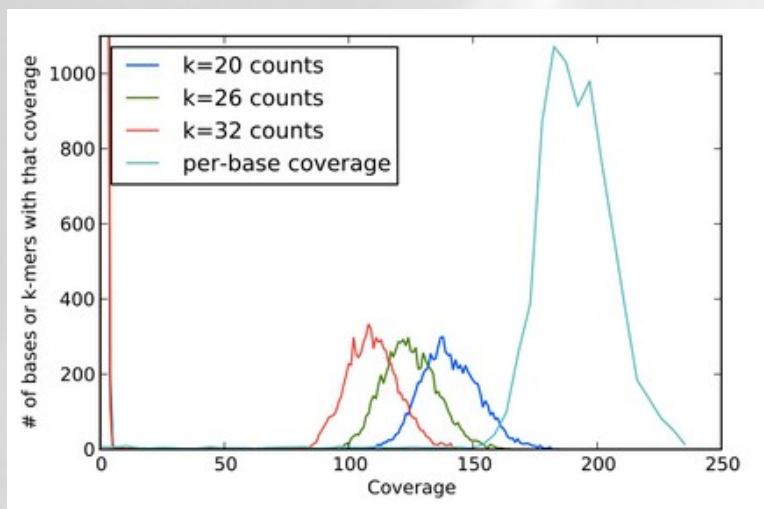
↳ larger k values bias your results towards more abundant isoforms

Which k-mer size to choose?

Shorter k-mers bring you more sensitivity:

- high coverage graph
- at the cost of a more complex graph
- due to more spurious overlaps

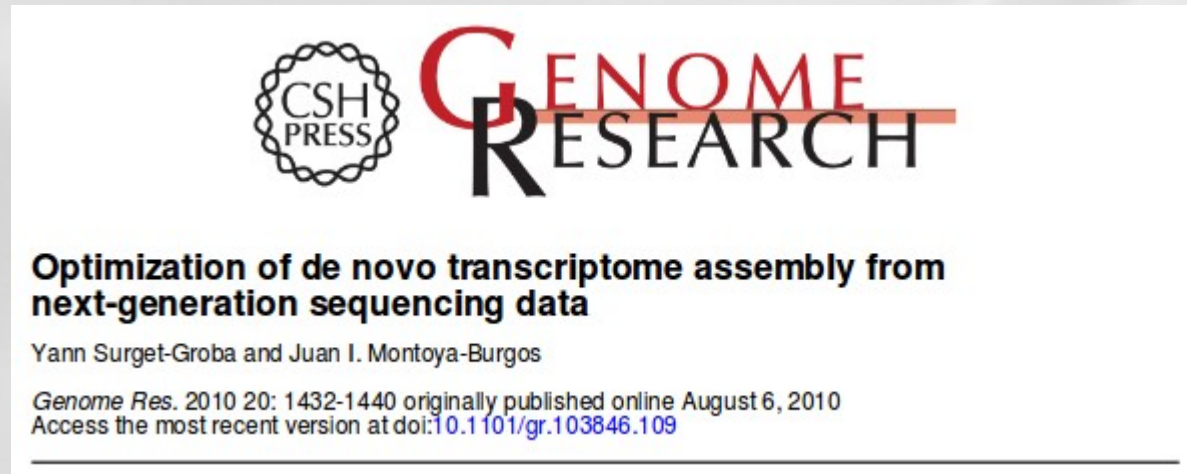
↳ smaller k values are susceptible to assembled low-abundance isoforms



H	M	k	transfrags (≥100bps)	95% aligned
✓		19	67319	81
✓		25	53504	85
✓		29	50936	90
✓		35	34012	90
	✓	21	57095	86
	✓	25	56473	88
	✓	29	59503	88
	✓	35	61939	90

Schulz MH & al. Oases: robust de novo RNA-seq assembly across the dynamic range of expression levels. Bioinformatics. 2012

The multiple k-mer strategy



Overview

- independent assemblies which vary by k-mer length
- assemblies are then merged into a final assembly

Velvet / Oases: command line

```

bash-4.1$ velveth
velveth - simple hashing program
Version 1.2.07

Copyright 2007, 2008 Daniel Zerbino (zerbino@ebi.ac.uk)
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

Compilation settings:
CATEGORIES = 2
MAXKMERLENGTH = 31

Usage:
./velveth directory hash_length {[-file_format][-read_type][-separate|-interleaved] filename1 [filename2 ...]} {...} [options]

    directory      : directory name for output files
    hash_length    : EITHER an odd integer (if even, it will be decremented) <= 31 (if above, will be reduced)
                   : OR: m,M,s where m and M are odd integers (if not, they will be decremented) with m < M <= 31 (if above, will be reduced)
                   :           and s is a step (even number). Velvet will then hash from k=m to k=M with a step of s
    filename       : path to sequence file or - for standard input

Options:
-strand_specific  : for strand specific transcriptome sequencing data (default: off)
-reuse_Sequences : reuse Sequences file (or link) already in directory (no need to provide original filenames in this case (default: off)
-noHash          : simply prepare Sequences file, do not hash reads or prepare Roadmaps file (default: off)
-create_binary   : create binary CnyUnifiedSeq file (default: off)

```

```

bash-4.1$ velvetg
velvetg - de Bruijn graph construction, error removal and repeat resolution
Version 1.2.07
Compilation settings:
CATEGORIES = 2
MAXKMERLENGTH = 31

Usage:
./velvetg directory [options]

    directory      : working directory name

Standard options:
-cov_cutoff <floating-point|auto>      : removal of low coverage nodes AFTER tour bus or allow the system to infer it
    (default: no removal)
-ins_length <integer>                  : expected distance between two paired end reads (default: no read pairing)
-read_trkg <yes|no>                    : tracking of short read positions in assembly (default: no tracking)
-min_contig_lgth <integer>             : minimum contig length exported to contigs.fa file (default: hash length * 2)

```

Velvet / Oases: command line

```

bash-4.1$ oases
oases - De novo transcriptome assembler for the Velvet package
Version 0.2.06

Copyright 2009,2010 Daniel Zerbino (dzerbino@soe.ucsc.edu)
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

Compilation settings:
CATEGORIES = 2
MAXKMERLENGTH = 64

Usage:
./oases directory [options]

    directory                : working directory name

Standard options:
-ins_length2 <integer>      : expected distance between two paired-end reads in the second short-read dataset (default: no read pairing)
-ins_length_long <integer>  : expected distance between two long paired-end reads (default: no read pairing)
-ins_length*_sd <integer>   : est. standard deviation of respective dataset (default: 10% of corresponding length)
                             [replace '*' by nothing, '2' or '_long' as necessary]
-unused_reads <yes|no>      : export unused reads in UnusedReads.fa file (default: no)
-amos_file <yes|no>         : export assembly to AMOS file (default: no export)
-alignments <yes|no>        : export a summary of contig alignment to the reference sequences (default: no)
--help                      : this help message

```

Velvet / Oases: command line

```
mkdir oasesOutDir
```

```
velveth_70_LONG oasesOutDir 27 -shortPaired -fastq.gz \  
-separate R1.fastq.gz R2.fastq.gz -noHash
```

```
---
```

```
mkdir oasesOutDir/k61
```

```
ln -s ../Sequences oasesOutDir/k61/Sequences
```

```
velveth_70_LONG oasesOutDir/k61 61 -reuse_Sequences
```

```
velvetg_70_LONG oasesOutDir/k61 -read_trkg yes \  
-min_contig_lgth 200
```

```
oases_70_LONG oasesOutDir/k61
```

```
---
```

```
Redo for all chosen k-mers
```


Velvet / Oases: command line

```
mkdir oasesOutDir/merge
```

```
velveth_70_LONG oasesOutDir/merge 27 -long \  
oasesOutDir/k*/transcripts.fa
```

```
velvetg_70_LONG oasesOutDir/merge -read_trkg yes \  
-conserveLong yes
```

```
oases_70_LONG oasesOutDir/merge -merge yes
```

Velvet / Oases: output

```
bash-4.1$ ls -lth --color k69
total 81M
-rw-rw-r-- 1 sigenae sigenae 300K  4 déc.  11:31 contig-ordering.txt
-rw-rw-r-- 1 sigenae sigenae 301K  4 déc.  11:31 transcripts.fa
-rw-rw-r-- 1 sigenae sigenae 9,7M  4 déc.  11:31 LastGraph
-rw-rw-r-- 1 sigenae sigenae  42K  4 déc.  11:31 stats.txt
-rw-rw-r-- 1 sigenae sigenae 1,3K  4 déc.  11:31 Log
-rw-rw-r-- 1 sigenae sigenae 272K  4 déc.  11:31 contigs.fa
-rw-rw-r-- 1 sigenae sigenae 9,7M  4 déc.  11:31 Graph2
-rw-rw-r-- 1 sigenae sigenae 584K  4 déc.  11:31 PreGraph
-rw-rw-r-- 1 sigenae sigenae  60M  4 déc.  11:31 Roadmaps
lrwxrwxrwx 1 sigenae sigenae   12  4 déc.  11:31 Sequences -> ../Sequences
```

```
bash-4.1$ ../count.sh
grep -c '^>' Sequences
1700490
grep -c ^NODE PreGraph
4177
grep -c ^NODE Graph2
719
grep -c '^>' contigs.fa
609
grep -c ^NODE LastGraph
646
grep -c '^>' transcripts.fa
586
```



Exercise n°2

Trinity: a novel method for the efficient and robust de novo reconstruction of transcriptomes from RNA-seq data

NATURE BIOTECHNOLOGY | RESEARCH | ARTICLE



日本語要約

Full-length transcriptome assembly from RNA-Seq data without a reference genome

Manfred G Grabherr, Brian J Haas, Moran Yassour, Joshua Z Levin, Dawn A Thompson, Ido Amit, Xian Adiconis, Lin Fan, Raktima Raychowdhury, Qiandong Zeng, Zehua Chen, Evan Mauceli, Nir Hacohen, Andreas Gnirke, Nicholas Rhind, Federica di Palma, Bruce W Birren, Chad Nusbaum, Kerstin Lindblad-Toh, Nir Friedman & Aviv Regev

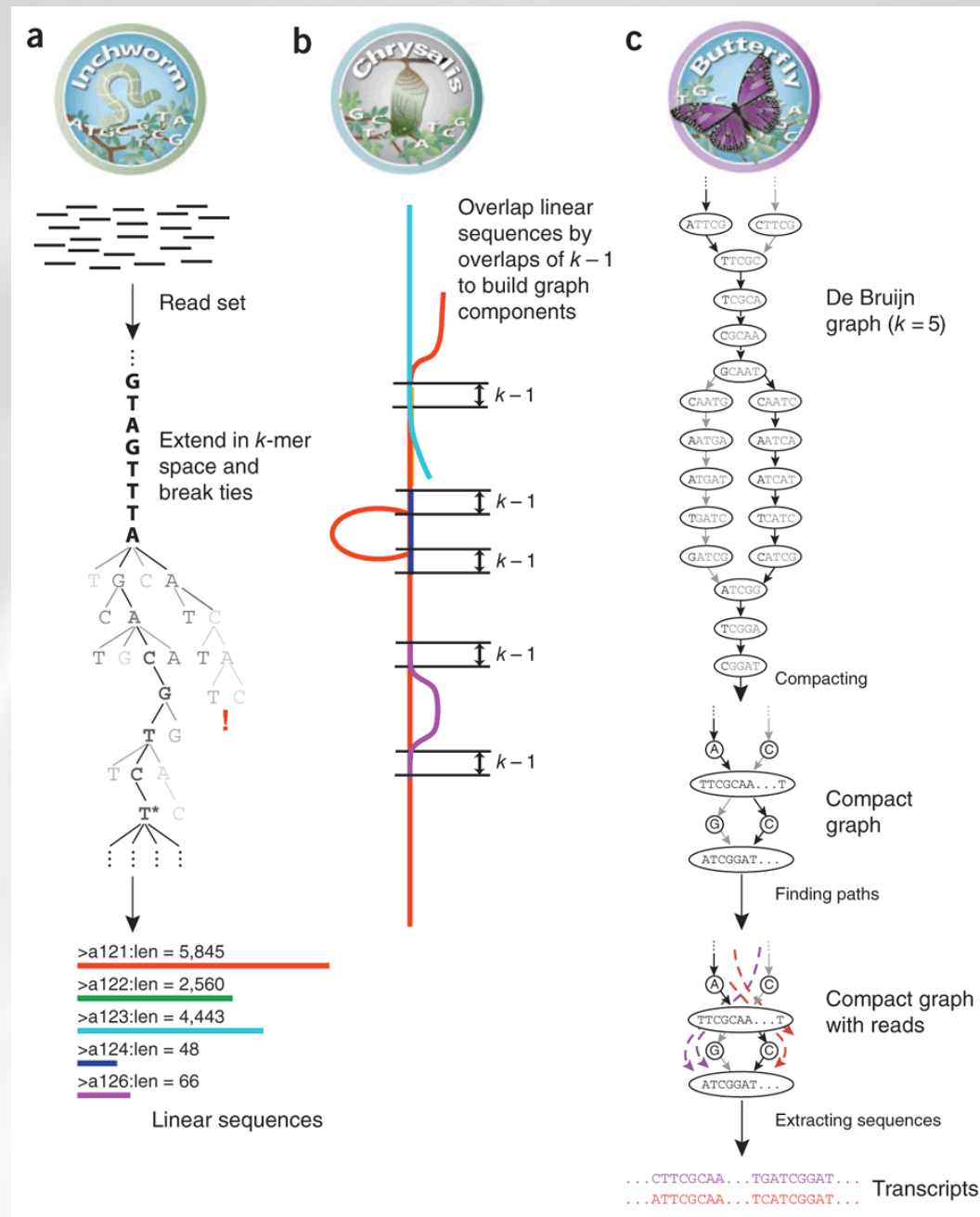
[Affiliations](#) | [Contributions](#) | [Corresponding authors](#)

Nature Biotechnology **29**, 644–652 (2011) | doi:10.1038/nbt.1883

Received 03 December 2010 | Accepted 28 April 2011 | Published online 15 May 2011

Pipeline:

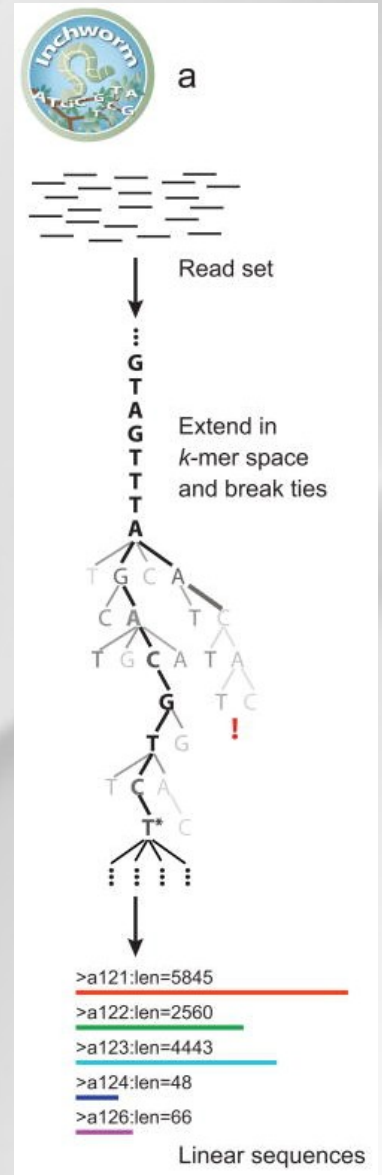
- Inchworm
- Chrysalis
- Butterfly



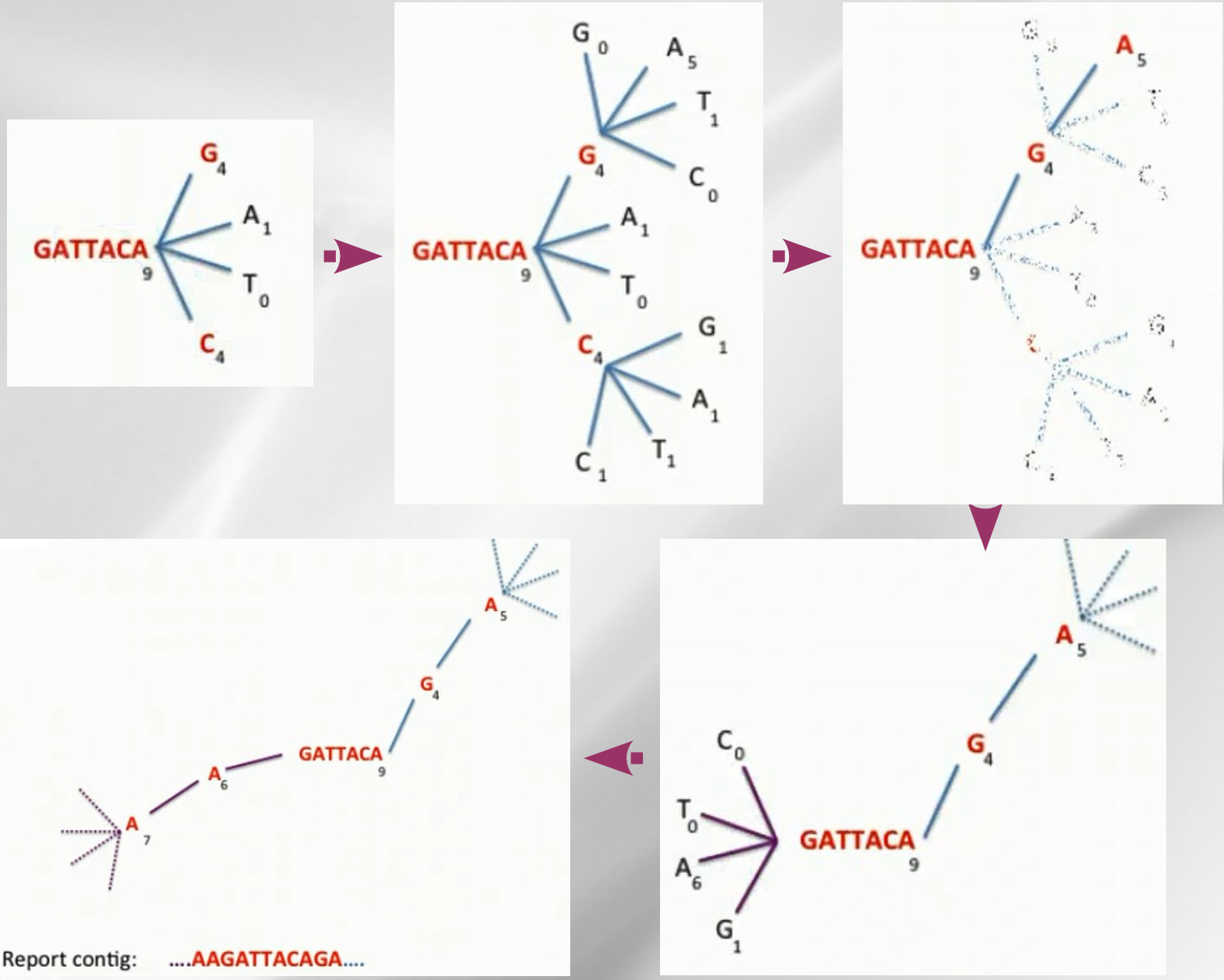
Trinity - Inchworm

Inchworm

1. build a k-mer dictionary from reads set (count occurrences)
2. removes likely error-containing k-mer based on occurrence
3. selects the most frequent k-mer to seed contig assembly
4. extends the seed in each direction by finding highest occurring k-mer with a k-1 overlap
5. extends until it cannot be extended and report contig ; removes assembled k-mers
6. repeats steps 3-5, starting with the most abundant k-mer until the entire dictionary has been depleted



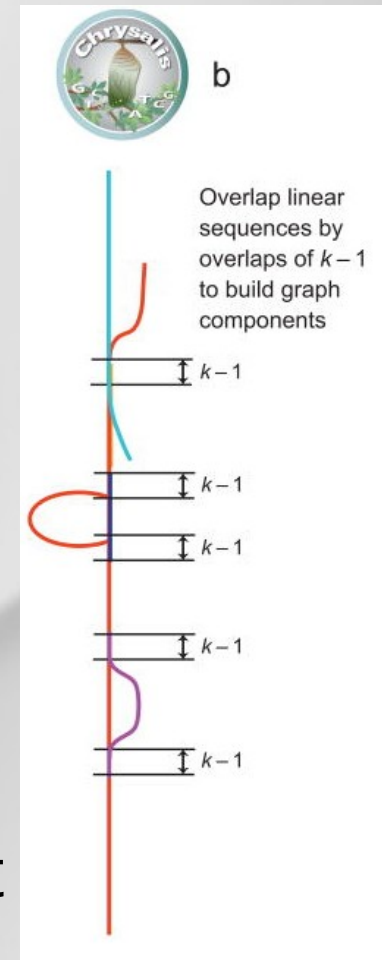
Trinity - Inchworm



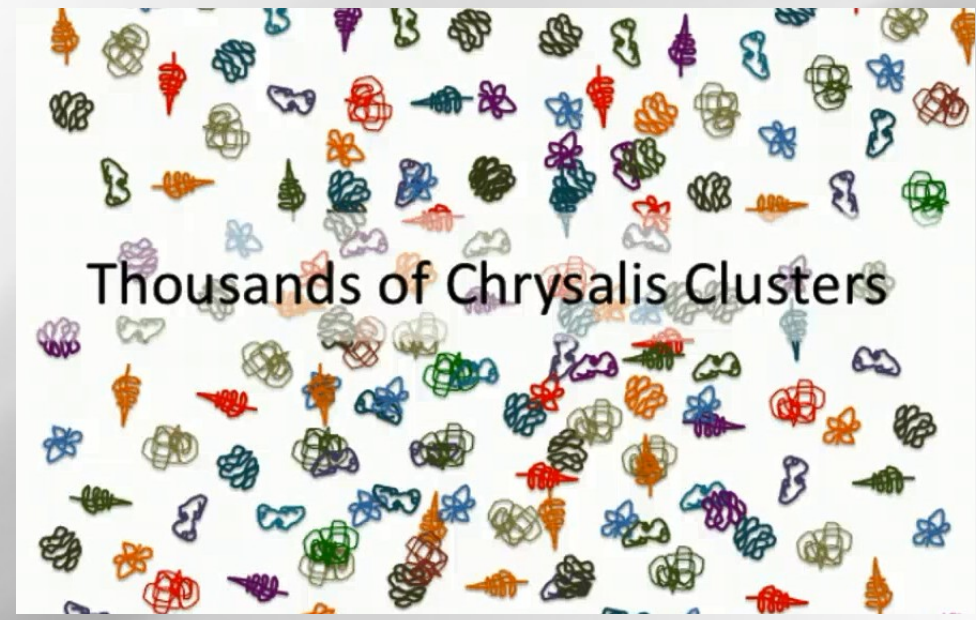
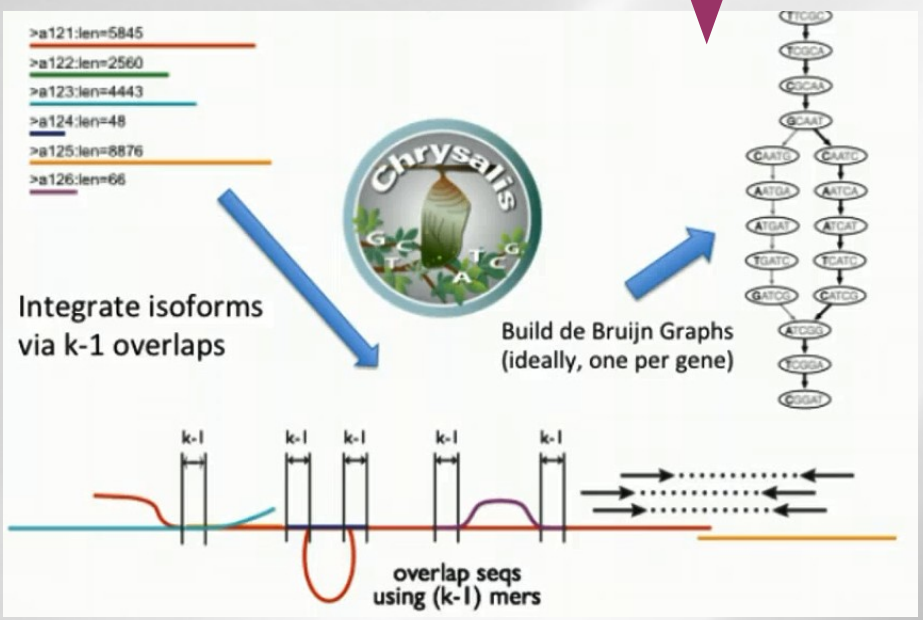
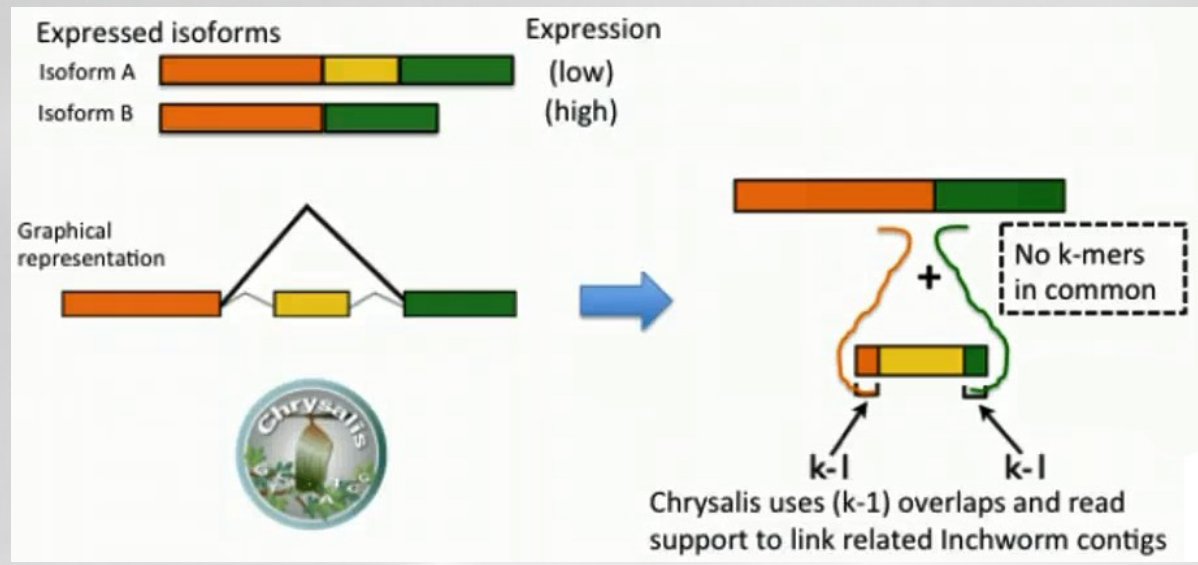
Trinity - Chrysalis

Chrysalis

- recursively groups inchworm contigs into connected components
 - * perfect overlaps of $k-1$ bases
 - * minimal number of reads span the junction across both contigs
- builds a de Bruijn graph per component
 - * nodes are word size of $k-1$
 - * edges are word size of k
 - * weights each edge with the number of $(k-1)$ -mers in the original reads that support it
- assigns each read to the component with which it shares the largest number of k -mers



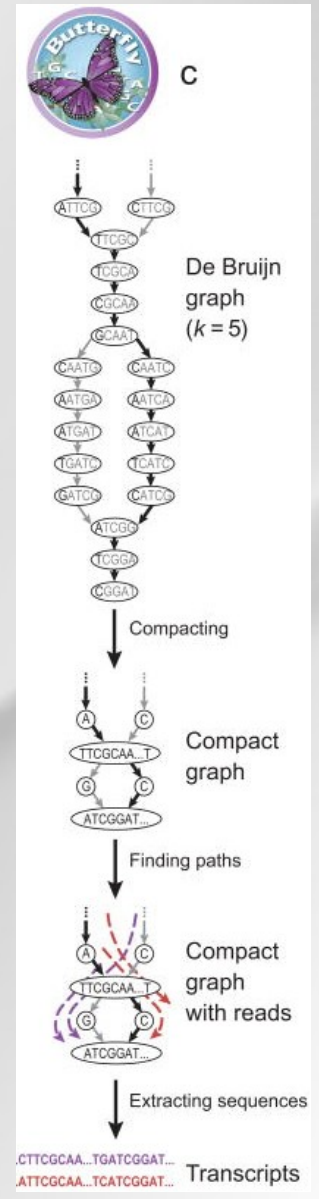
Trinity Chrysalis



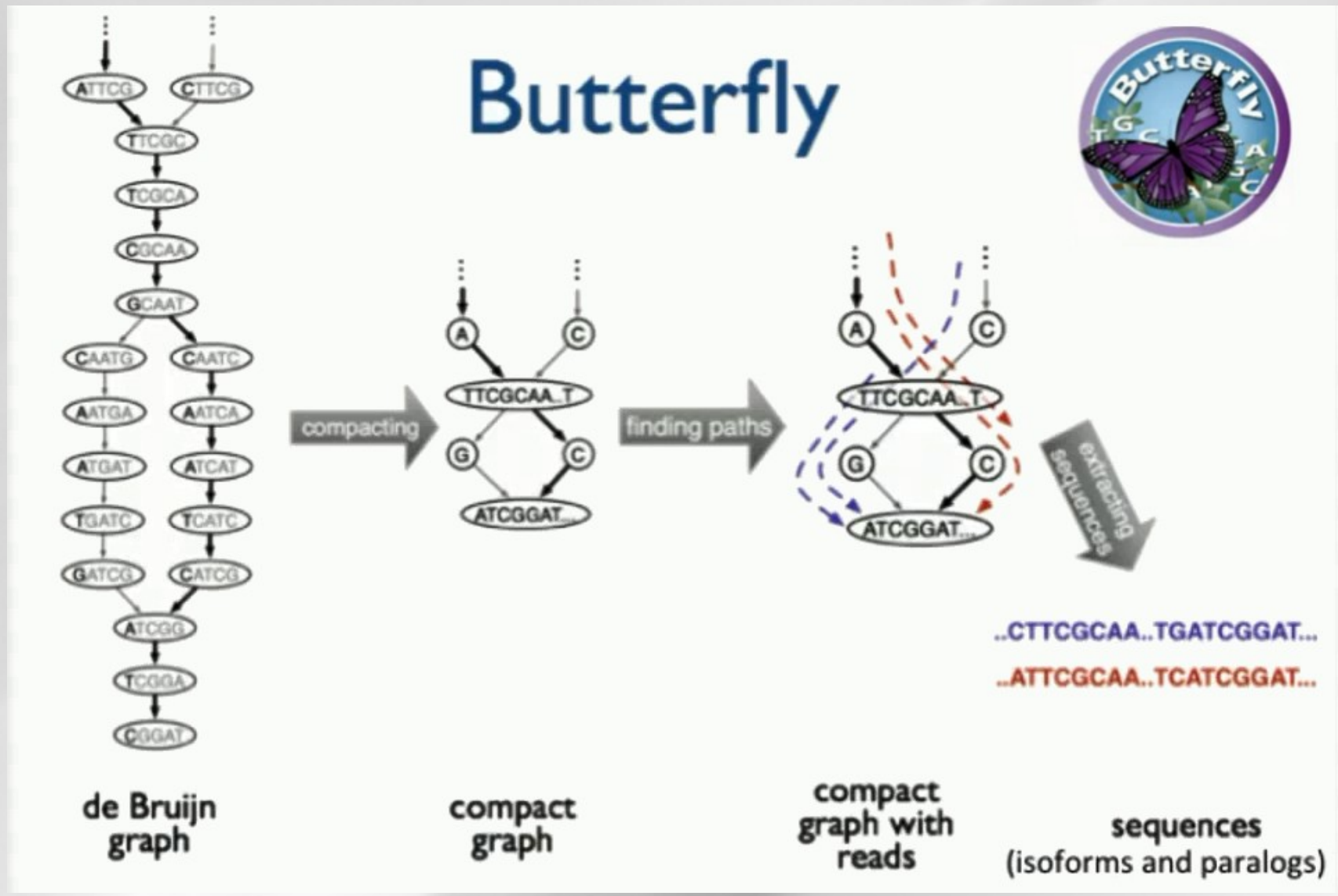
Trinity - Butterfly

Butterfly reconstructs distinct transcripts for splice isoforms and paralogous genes

- Graph simplification
 - * merge consecutive nodes in linear paths
 - * prune edges that represent minor deviations (supported by comparatively few reads)
- Plausible path scoring
 - * identifies those paths that are supported by actual reads and read pairs
 - * a dynamic programming procedure resolves ambiguities and reduce the combinatorial number of paths
 - * enumerate linear sequences



Trinity - Butterfly



Trinity: command line

```
#####
#
#
#
#
#
#
#
#
#####
#
# Required:
#
# --seqType <string>      :type of reads: ( cfa, cfq, fa, or fq )
#
# --JM <string>           :(Jellyfish Memory) number of GB of system memory to use for
#                          k-mer counting by jellyfish (eg. 10G) *include the 'G' char
#
# If paired reads:
#   --left <string>       :left reads, one or more (separated by space)
#   --right <string>      :right reads, one or more (separated by space)
#
# Or, if unpaired reads:
#   --single <string>     :single reads, one or more (note, if single file contains pairs, can use flag: --run_as_paired )
#
#####
## Misc: #####
#
# --SS_lib_type <string>  :Strand-specific RNA-Seq read orientation.
#                          if paired: RF or FR,
#                          if single: F or R. (dUTP method = RF)
#                          See web documentation.
#
# --output <string>      :name of directory for output (will be
#                          created if it doesn't already exist)
#                          default( "/work/sigenae/temp/RNASeq_de_novo_2014/trinity_out_dir" )
#
# --CPU <int>             :number of CPUs to use, default: 2
#
# --min_contig_length <int> :minimum assembled contig length to report
#                          (def=200)
#
```

Trinity.pl --output trinityOutDir --seqType fq --JM 64G --left R1.fastq.gz --right R2.fastq.gz --CPU 4
 Mem usage 1G RAM per 1M ~76 base Illumina paired reads

Trinity: output

```
bash-4.1$ ls -lth --color
total 913M
 617 4 déc. 14:19 Trinity.timing
3,8M 4 déc. 14:19 Trinity.fasta
 16K 4 déc. 14:07 chrysalis
291K 4 déc. 14:02 iworm_bundle_file_listing.txt
  0 4 déc. 14:01 iworm_scaffolds.txt.finished
458K 4 déc. 14:01 iworm_scaffolds.txt
 39M 4 déc. 14:01 scaffolding_entries.sam
  0 4 déc. 14:00 bowtie.nameSorted.sam.finished
360M 4 déc. 14:00 bowtie.nameSorted.sam
  0 4 déc. 14:00 bowtie.out.finished
  0 4 déc. 13:59 target.fa.finished
8,3M 4 déc. 13:59 target.rev.1.ebwt
728K 4 déc. 13:59 target.rev.2.ebwt
8,3M 4 déc. 13:59 target.1.ebwt
728K 4 déc. 13:59 target.2.ebwt
570K 4 déc. 13:59 target.3.ebwt
1,5M 4 déc. 13:59 target.4.ebwt
 99 4 déc. 13:59 target.fa -> inchworm.K25.L25.DS.fa
  0 4 déc. 13:59 inchworm.K25.L25.DS.fa.finished
7,5M 4 déc. 13:59 inchworm.K25.L25.DS.fa
  9 4 déc. 13:58 inchworm.kmer_count
  0 4 déc. 13:58 jellyfish.1.finished
317M 4 déc. 13:58 jellyfish.kmers.fa
  8 4 déc. 13:54 both.fa.read_count
160M 4 déc. 13:54 both.fa
```

```
bash-4.1$ ls -lth --color chrysalis/
total 22M
801K 4 déc. 14:19 butterfly_commands.adj.completed
128K 4 déc. 14:19 RawComps.0
256K 4 déc. 14:19 RawComps.1
 16K 4 déc. 14:19 RawComps.3
 64K 4 déc. 14:19 RawComps.2
 16K 4 déc. 14:18 RawComps.5
 16K 4 déc. 14:18 RawComps.7
 16K 4 déc. 14:18 RawComps.4
 16K 4 déc. 14:17 RawComps.9
 16K 4 déc. 14:17 RawComps.6
 16K 4 déc. 14:17 RawComps.8
  0 4 déc. 14:07 quantifyGraph_commands.run.finished
1,1M 4 déc. 14:07 quantifyGraph_commands.completed
801K 4 déc. 14:06 butterfly_commands.adj
  0 4 déc. 14:06 chrysalis.finished
1,1M 4 déc. 14:05 quantifyGraph_commands
559K 4 déc. 14:05 butterfly_commands
259K 4 déc. 14:05 component_file_listing.txt
  0 4 déc. 14:05 readsToTranscripts.finished
  7 4 déc. 14:05 readcounts.out
3,3M 4 déc. 14:02 bundled.fasta
  0 4 déc. 14:02 GraphFromIwormFasta.finished
14M 4 déc. 14:02 GraphFromIwormFasta.out
```

```
bash-4.1$ ./count.sh
grep -c '^>' both.fa
1700490
grep -c '^>' jellyfish.kmers.fa
11380211
grep -c '^>' inchworm.K25.L25.DS.fa
64776
cat chrysalis/component_file_listing.txt | wc -l
2479
cat chrysalis/quantifyGraph_commands | wc -l
2479
cat chrysalis/butterfly_commands | wc -l
2479
grep -c '^>' Trinity.fasta
4365
```



Exercise n°3

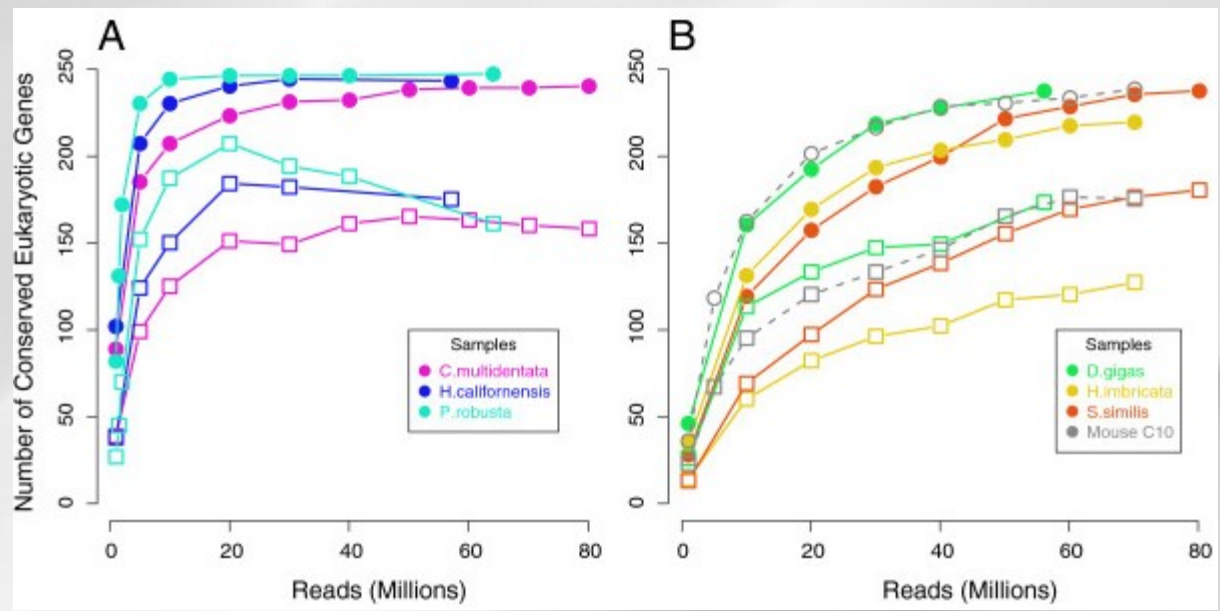
What to do before assembling?

The aim is to simplify graphs:

- Cleaning
- Sampling
- Unicity
- Normalization

Sampling

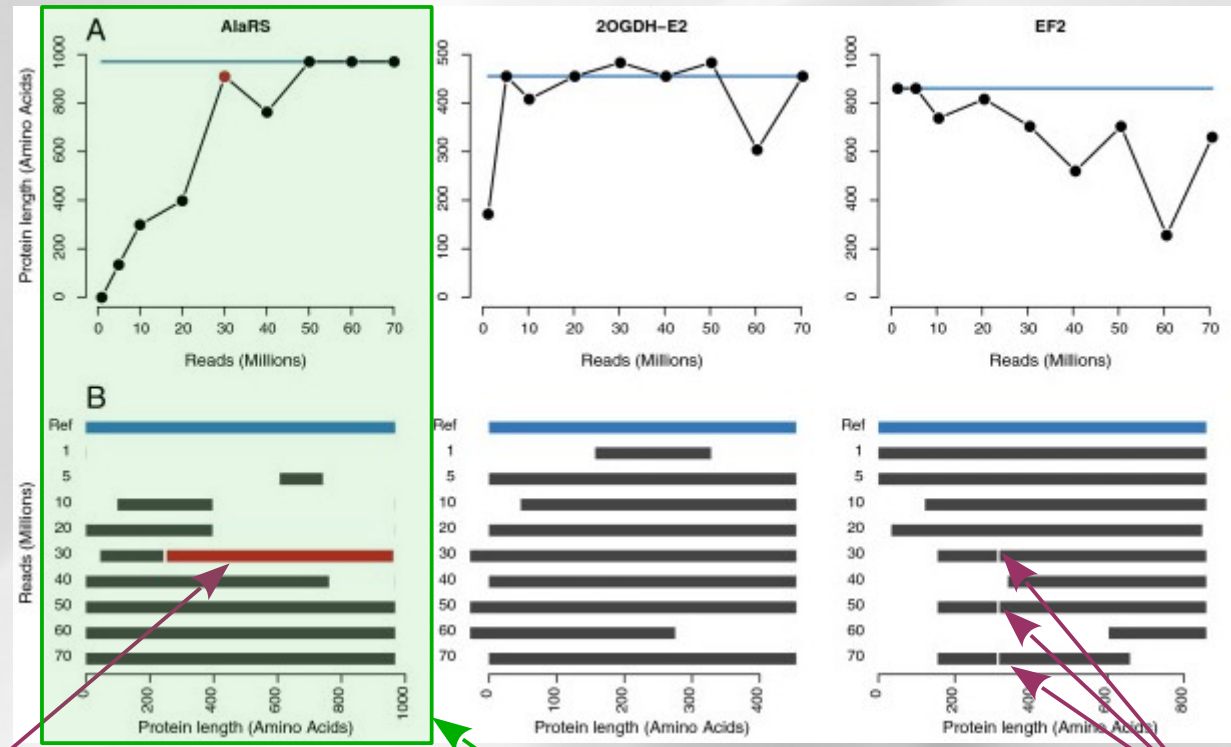
A comparison across non-model animals suggests an optimal sequencing depth for *de novo* transcriptome assembly
 ↪ 20 (tissue) to 30 (whole-animal) millions reads



● contigs with reliable hit against KOGs

□ contigs which the translated protein was within the expected size range of the conserved gene

Misasassembly examples



Chimera

Majority trend

Short gaps

Normalization

TRINITY_RNASEQ_ROOT/util/insilico_read_normalization.pl

- Build a catalog of k-mers and compute abundance
- Compute the k-mer abundance profile for each read
 - * median k-mer abundance (C)
 - * standard deviation for the k-mer coverage
- Retain reads with probability $\min(1, T/C)$ [Perl: $\text{rand}(1) \leq T/C$]
 - * captures all reads falling below the targeted cov. level (T)
 - * down-samples reads occurring at higher coverage than T
- Discard reads with aberrant k-mer abundance profile (std-dev k-mer cov > median k-mer abundance)

Normalization

Normalization effects (our experience):

- drastically decrease #reads or #pairs (-50 to -90%) ⊕
- significantly decrease #contigs (-10 to 15%) ⊕
- slightly decrease #rebuilt proteins (-3%) ⊗
- null or positive effect on remapping rate (0 to 10%) ⊕

See you tomorrow!