de Bruijn graphs for sequencing data

Rayan Chikhi

CNRS
Bonsai team, CRIStAL/INRIA, Univ. Lille 1

SMPGD 2016

Motivation

- de Bruijn graphs are instrumental for reference-free sequencing data analysis:
  1. Genome assembly
  2. Transcriptome assembly
  3. Metagenomics assembly
  4. Ref-free variant detection (recent)
  5. Transcript quantification (recent)

Graphs

A graph is:

- a set of nodes, and
- a set of edges (directed or not)

k-MERS

k-mers are strings of length *k*

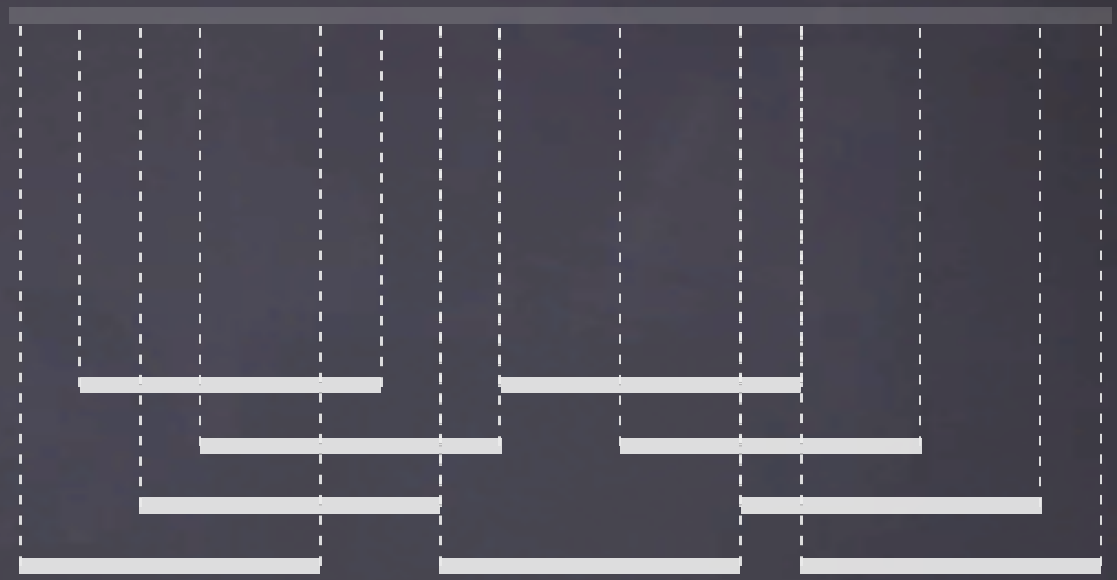
read **ACTGATGAC**

k-mers (k=3)

ACT
CTG
TGA
GAT
ATG
TGA
GAC

READS, ASSEMBLY

genome
not known



reads
overlapping substrings that cover the genome redundantly

assembly
what we think the genome is

Graphs for sequencing data

Graphs represent overlaps between sequences in reads.

Two families of graphs for sequencing data:

- de Bruijn graphs — generally for Illumina data
- string graphs — generally for Sanger/PacBio data

de Bruijn graphs

A **de Bruijn** graph for a fixed integer k:

1. **Nodes** = all *k-mers* (substrings of length k) in the reads.
2. There is an **edge** between x and y if the $(k-1)$-mer prefix of y matches exactly the $(k-1)$-mer suffix of x.

Example for $k = 3$ and a single read:

`ACTG`

de Bruijn graphs

Example for many reads and still $k = 3$.

```
ACTG
 CTGC
  TGCC
```

de Bruijn graphs: redundancy

What happens if we add redundancy?

```
ACTG
ACTG
 CTGC
 CTGC
 CTGC
  TGCC
  TGCC
```

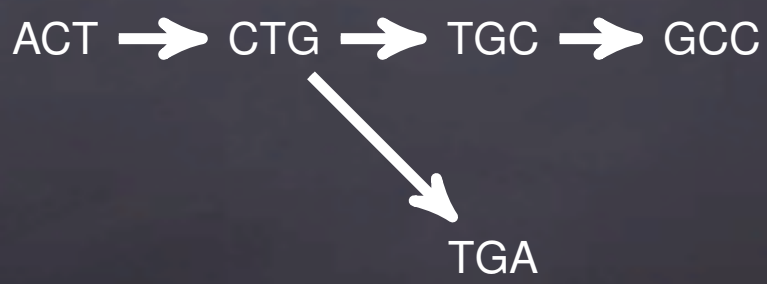
dBG, $k = 3$:

de Bruijn graphs: errors

How is a sequencing error (at the end of a read) impacting the de Bruijn graph?

```
ACTG
 CTGC
 CTGA
  TGCC
```

dBG, $k = 3$:

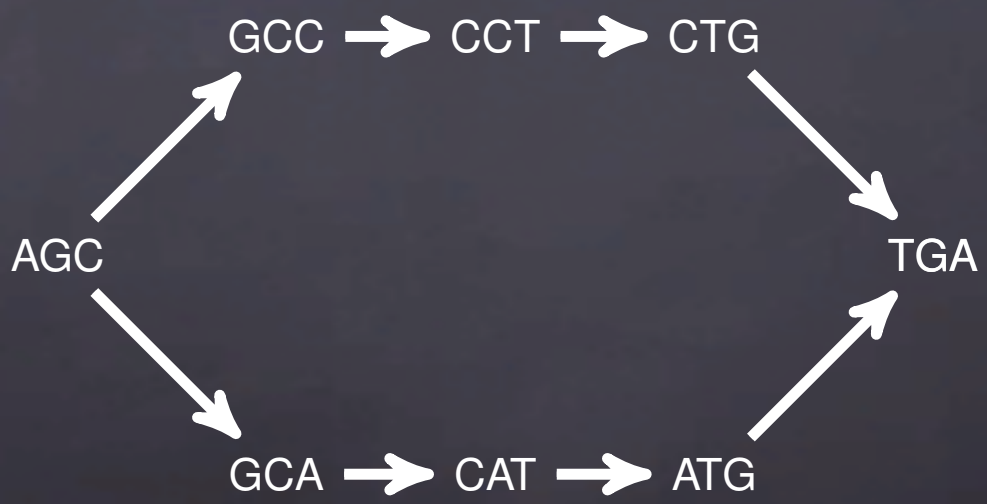


de Bruijn graphs: SNPs

What is the effect of a SNP (or a sequencing error inside a read) on the graph?

AGCCTGA
AGCATGA

dBG, $k = 3$:

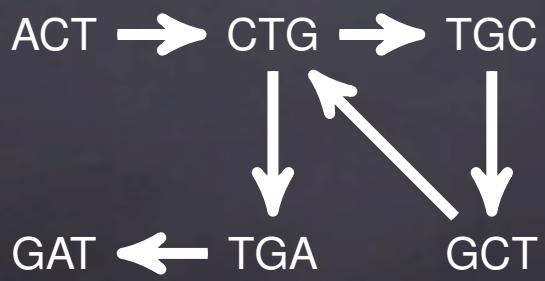


de Bruijn graphs: repeats

What is the effect of a small repeat on the graph?

```
ACTG
 CTGC
  TGCT
   GCTG
    CTGA
     TGAT
```

dBG, $k = 3$:



Comparison string graph / de Bruijn graph

On the same example, compare the de Bruijn graph with the string graph:

```
AGTGCT
 GTGCTA
   GCTAA
```

String graph, overlap threshold of 3:

de Bruijn graph, $k = 3$:

A short practical

Reads:

```
TACAGT
  CAGTC
   AGTCAG
     TCAGA
```

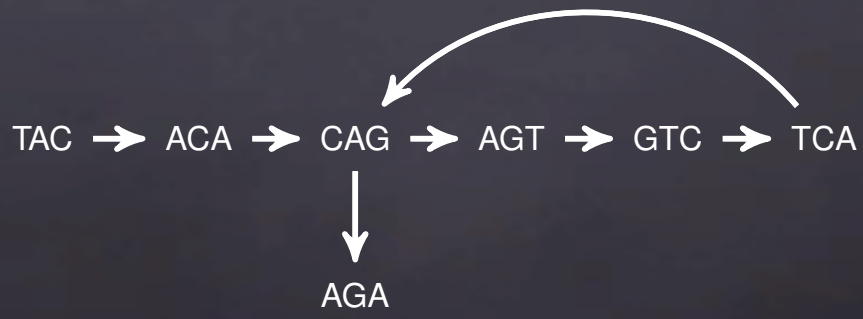
1. Enumerate all distinct k-mers in these reads, for $k = 3$.
2. Construct the de Bruijn graph for $k = 3$.
   (Reminder: nodes are distinct k-mers and edges are all exact $(k-1)$-overlaps)

Practical (solution)

Reads:

```
TACAGT
  CAGTC
   AGTCAG
     TCAGA
```

1. The distinct 3-mers are: TAC, ACA, CAG, AGT, GTC, TCA, AGA
2. Note that CAG appears at two places, but is always only a single node.
3. Construct the de Bruijn graph for $k = 3$.



4. Observe that the order and relative alignment of the reads were not necessary to construct the graph.

Short note on reverse complements

Because sequencing is generally not strand-specific:

> *We always consider that reads (and k-mers) are equal to their reverse complements.*

E.g:
AAA = TTT
ATG = CAT

The choice of k

Choice of k is critical:

- k-mers that contain a sequencing error are noise
- $k < \log_4(|\text{genome}|)$: nearly complete graph, uninformative
- small k: collapses repeats, more coverage of non-noisy k-mers
- large k: less repeat collapsing, less non-noisy k-mer cov.
- k too high: false negatives

Generally, $k \geq 20$.
(Compare 4^k to the genome size.)
Higher sequencing coverage means larger k values can be used.

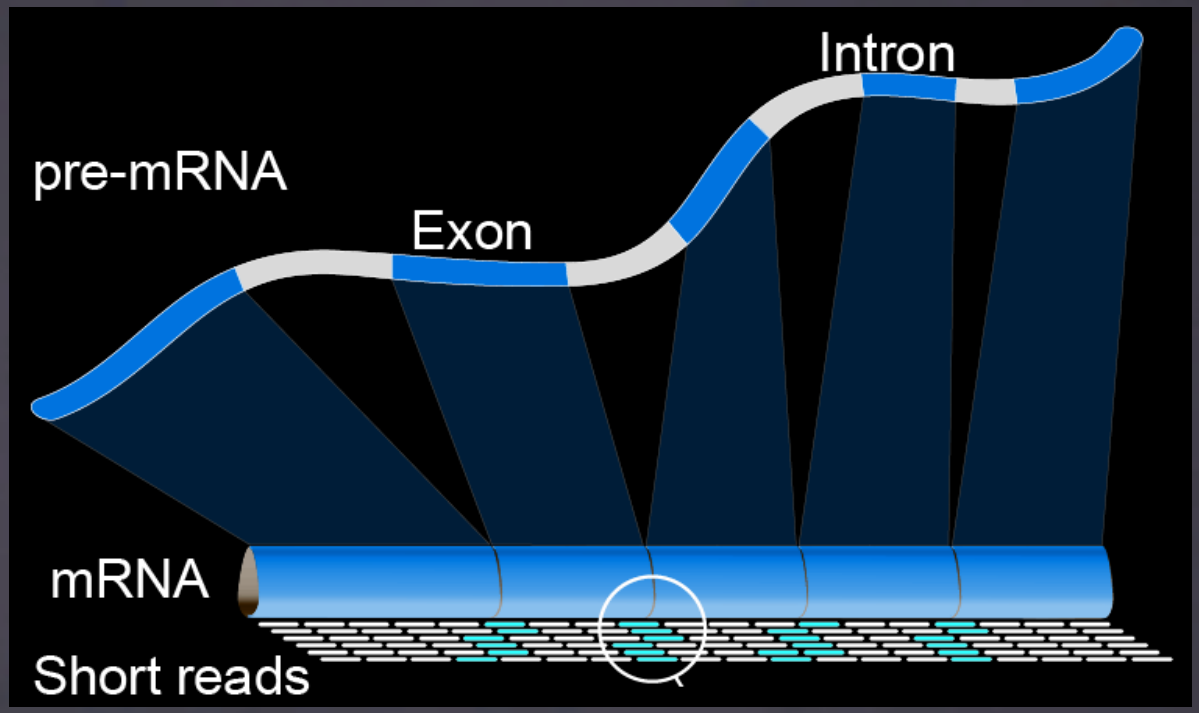
Highlight on 3 applications

1. DNA/RNA assembly
2. Transcript quantification
3. Variant detection

GENOME ASSEMBLY

Difficulties: repetitions, sequencing errors, heterozygosity

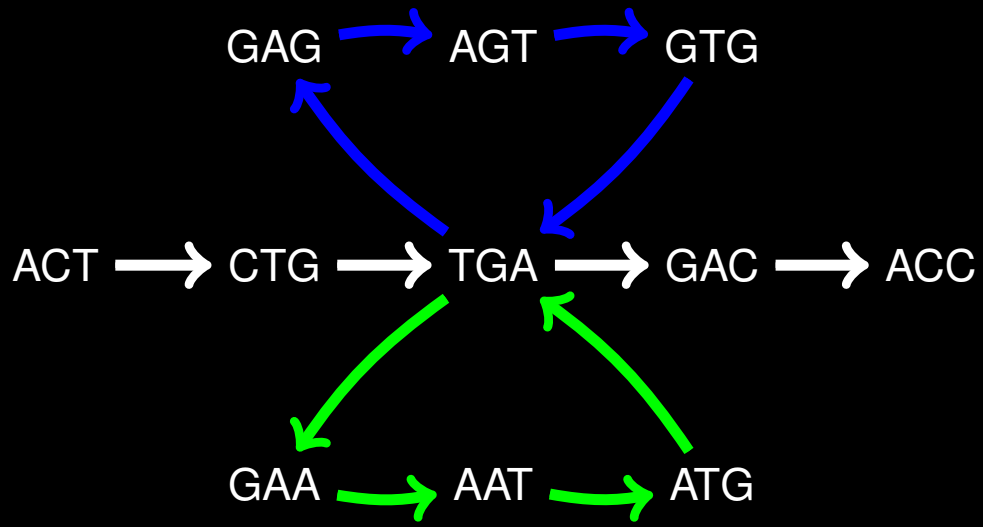
TRANSCRIPTOME ASSEMBLY



Goal: reconstruct mRNA sequences
Difficulties: (repetitions), various expression levels, alternative splicing

How does one assemble using a de Bruijn graph?

Return a **set of paths** covering the graph, such that *all possible assemblies* contain these paths.



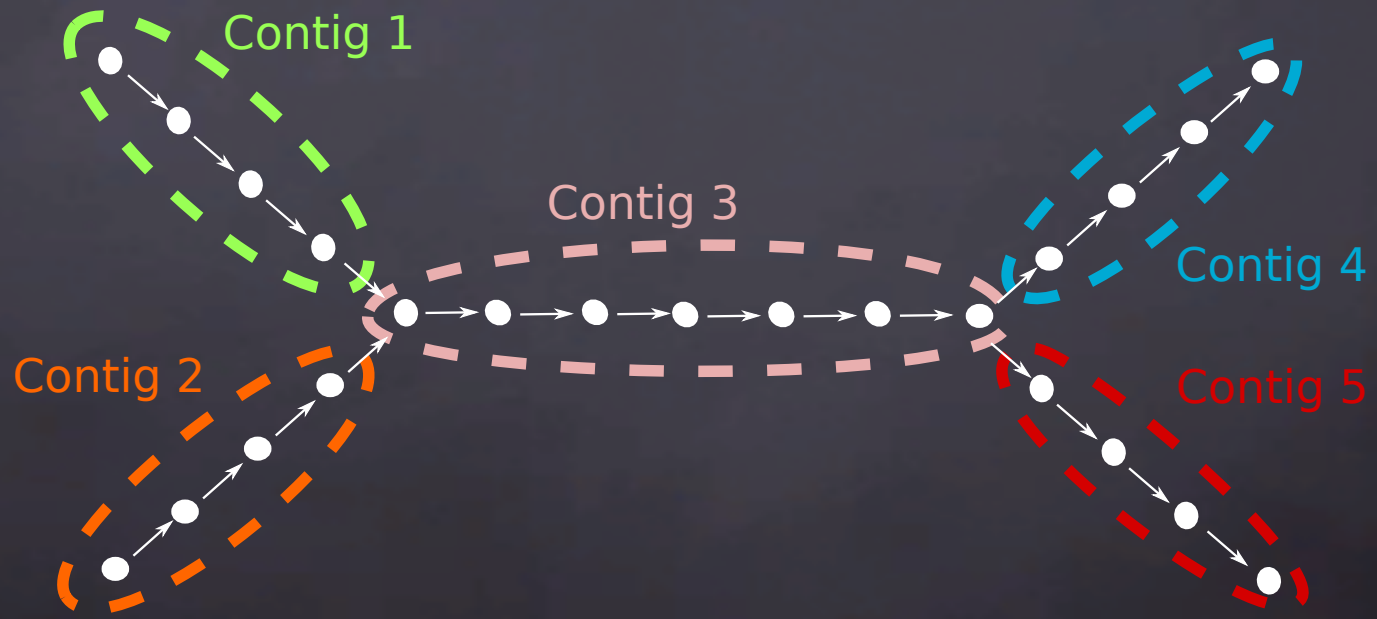
An assembly is the following set of paths:

$\{\texttt{ACTGA}, \texttt{GACC}, \texttt{GAGTG}, \texttt{GAATG}\}$

CONTIGS CONSTRUCTION

Contigs are *node-disjoint* **simple paths**.

simple path: a path that does not branch.
node-disjoint: two different paths cannot share a node.

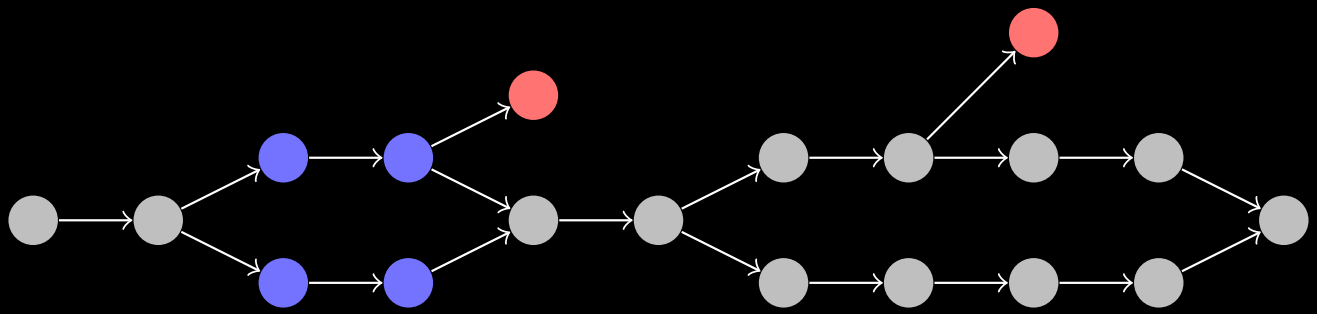


How an assembler works

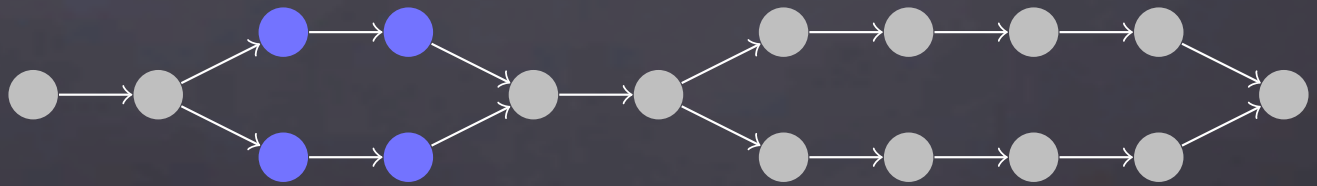
[SPAdes, Velvet, ABySS, SOAPdenovo, SGA, Megahit, Minia, .., HGAP, FALCON]

1) Maybe correct the reads. (SPAdes, HGAP, SGA, FALCON)
2) Construct a graph from the reads.

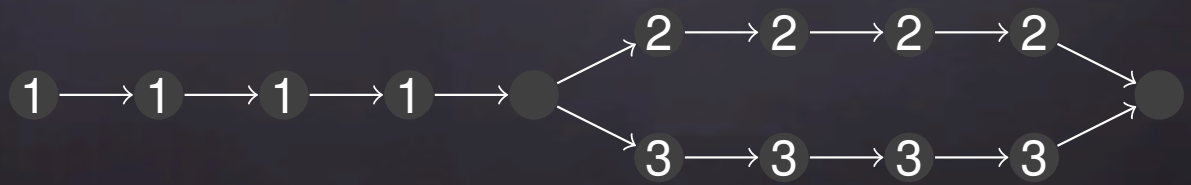
Assembly graph with variants & errors



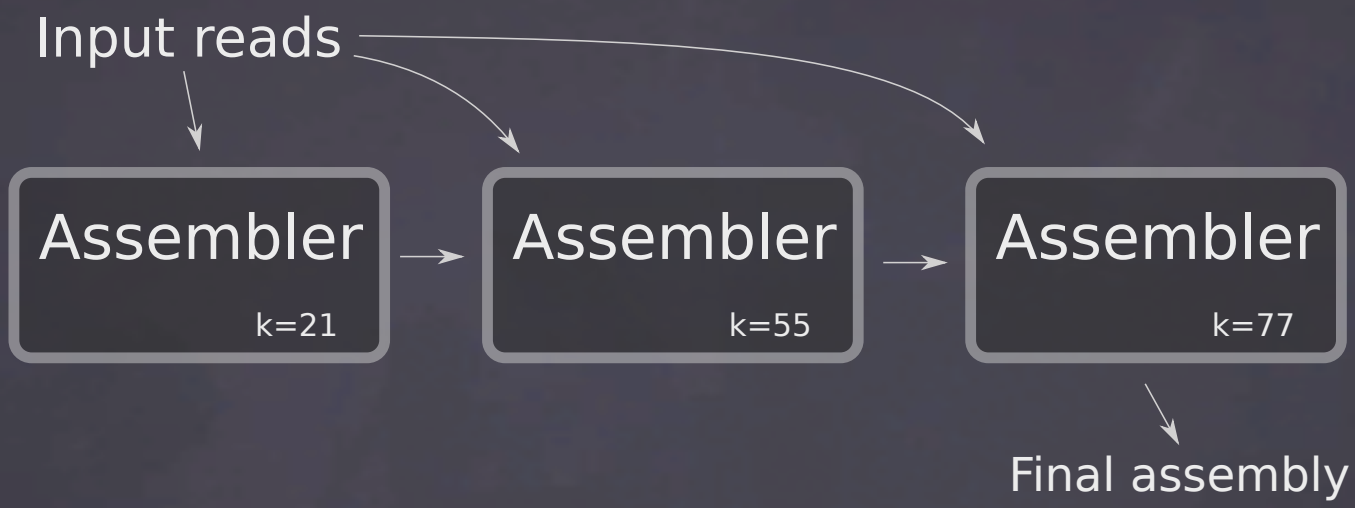
3) Likely sequencing errors are removed.



3) Known biological events are removed.
4) Finally, **simple paths** (i.e. contigs) are returned.



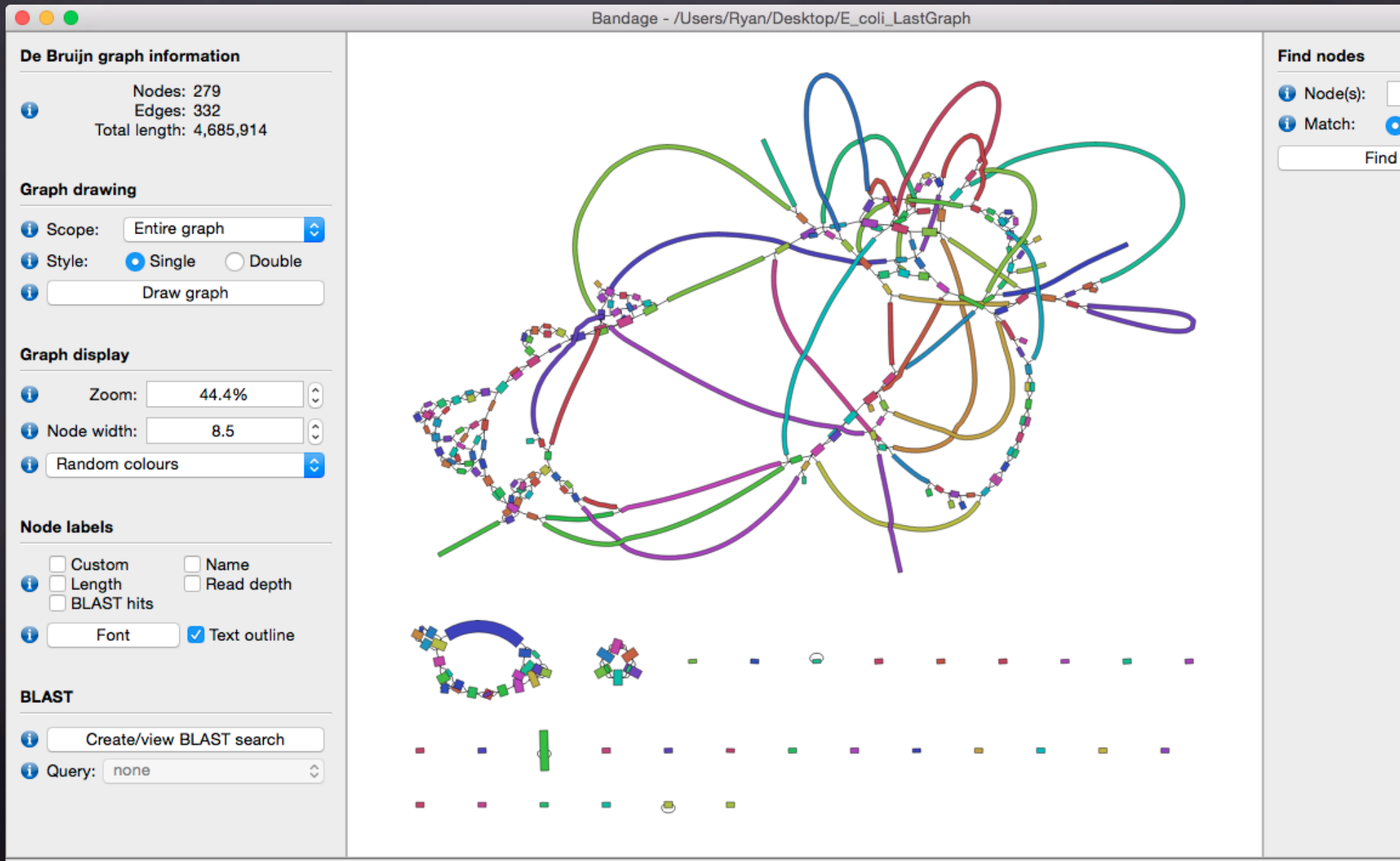
MULTI-K *de novo* ASSEMBLY



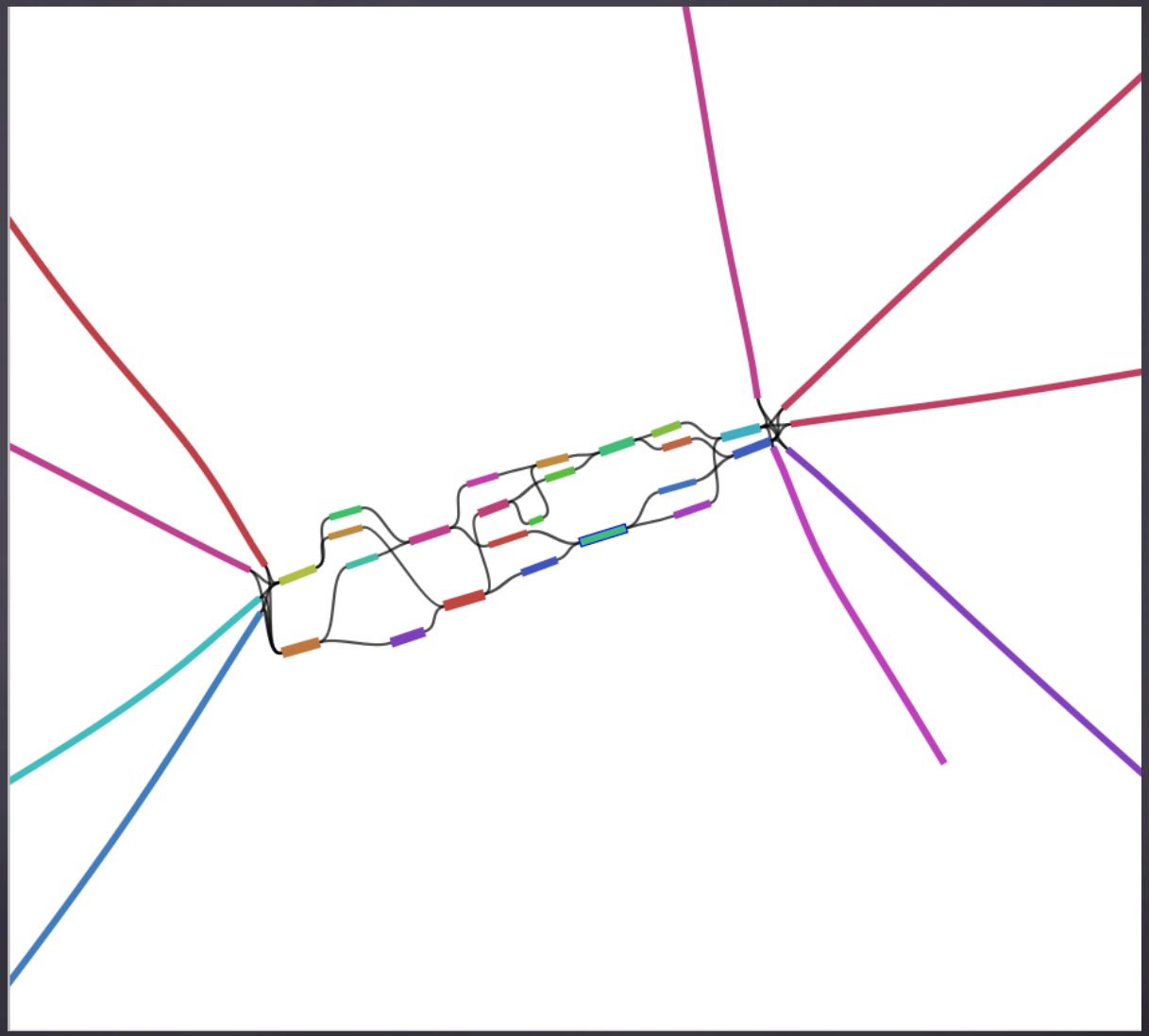
Principle:

- Assembler is a black box
- Input reads + previous assembly with shorter *k*

de Bruijn graph visualization: Bandage

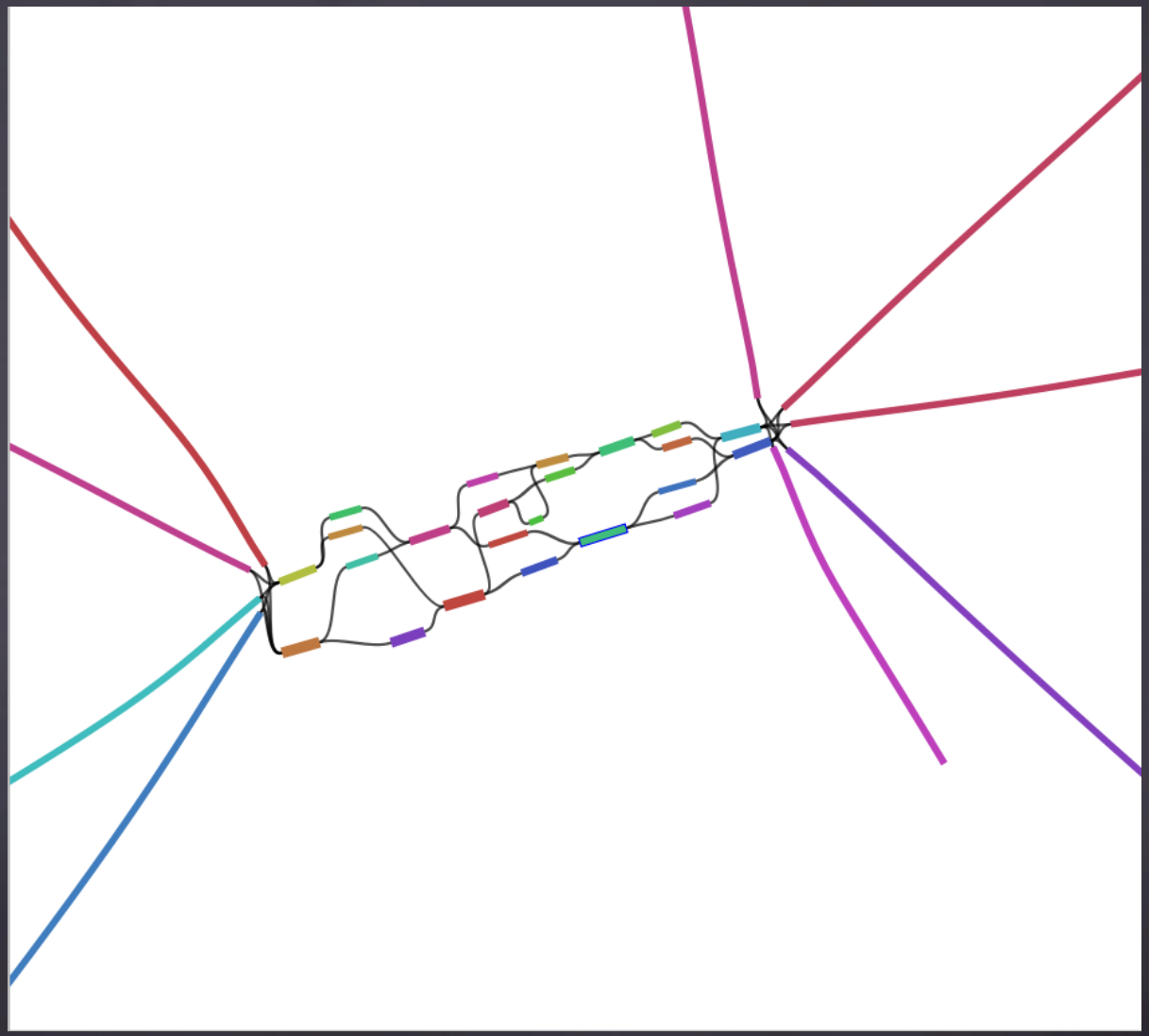


BANDAGE



E. coli SPAdes assembly (excerpt). Fig from Lex Nederbragt. What is this knot?

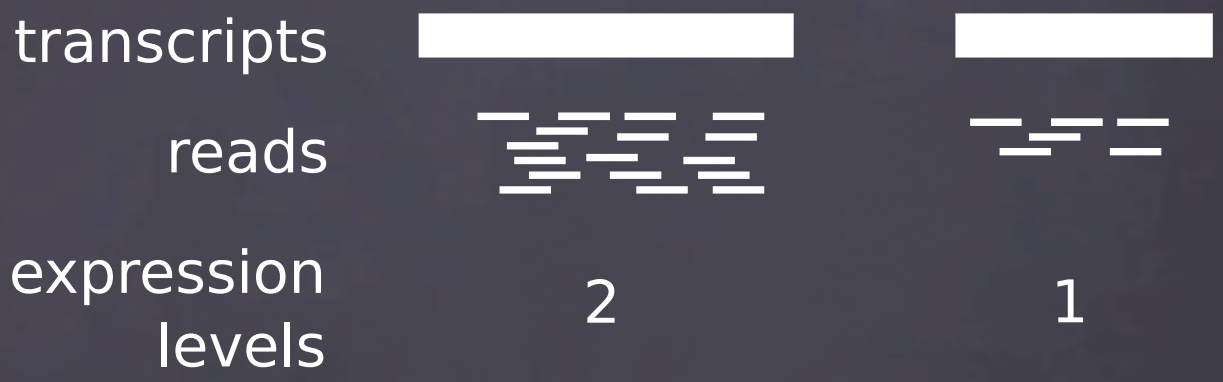
BANDAGE



E. coli SPAdes assembly (excerpt). Fig from Lex Nederbragt. What is this knot?
collapsed ribosomal genes (16S, 2S, ..)

RNA Quantification

Task: quantify abundance of transcripts in RNA-Seq data.



(Many possible units for expression: FPKM, RPKM, TPM)
But one basic task: assign reads to transcripts

RNA Quantification

k-mer based methods are emerging:

- Sailfish, Kallisto, Salmon, Graphalign

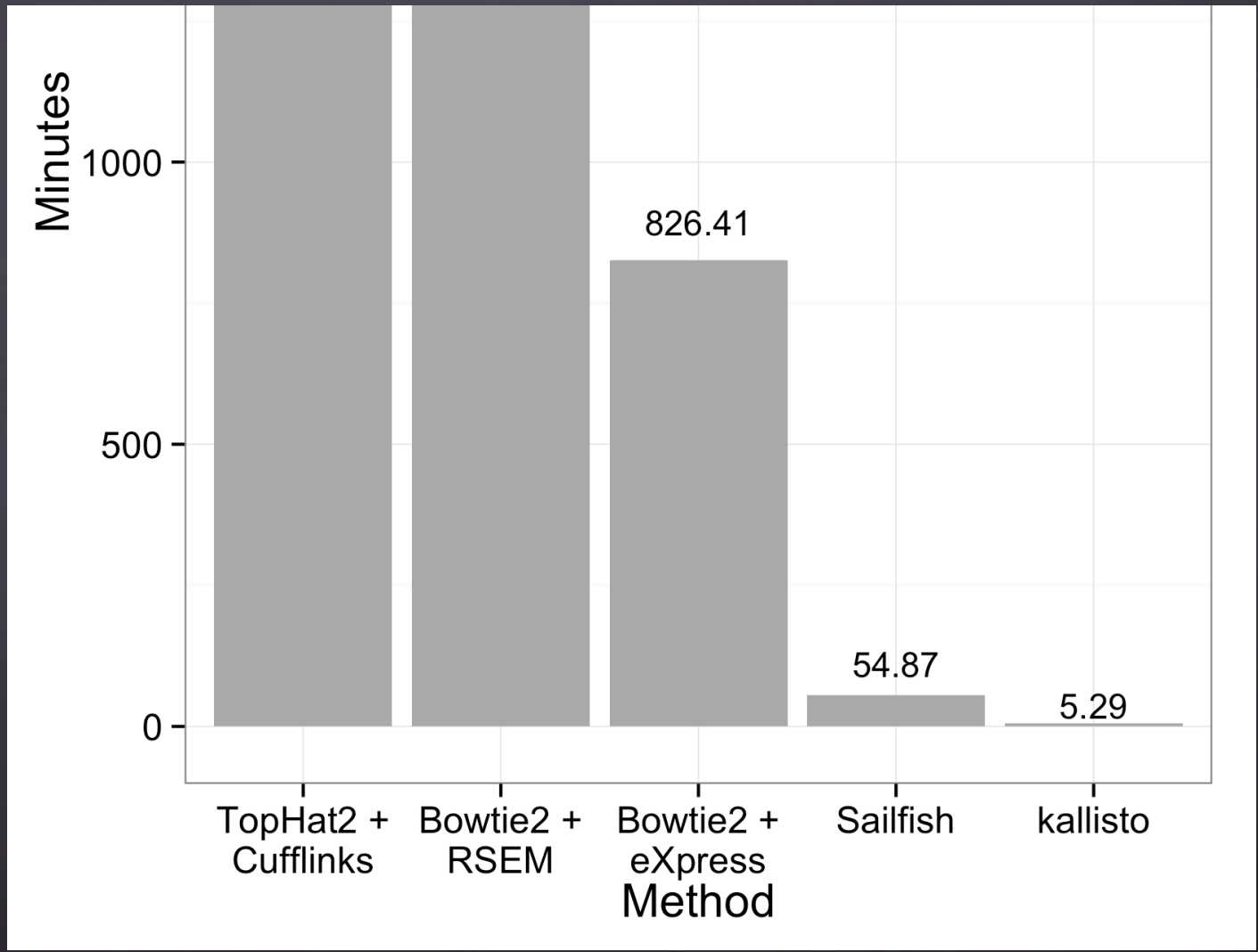


Fig: Kallisto website

KALLISTO

Index: [Bray 15 (arXiv)]

1. Construct ref. transcriptome de Bruijn graph
2. Color nodes with the transcript(s) it occurs in

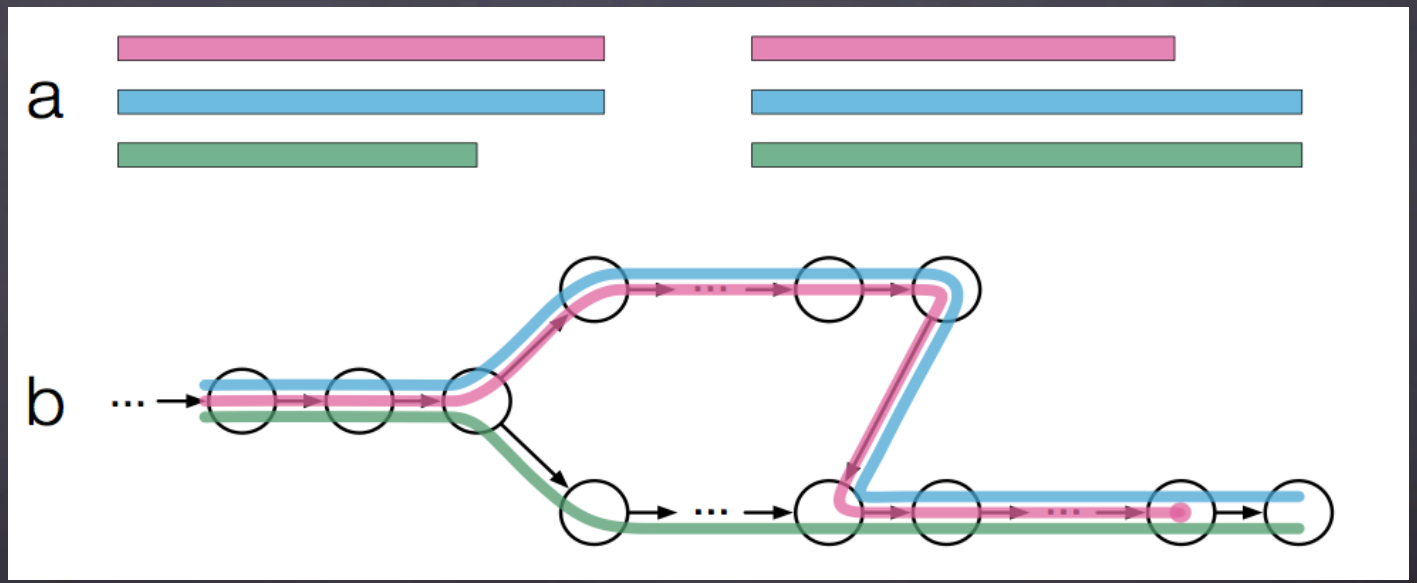


Fig: http://tinyheero.github.io/2015/09/02/pseudoalignments-kallisto.html

KALLISTO

Read pseudoalignment (1):

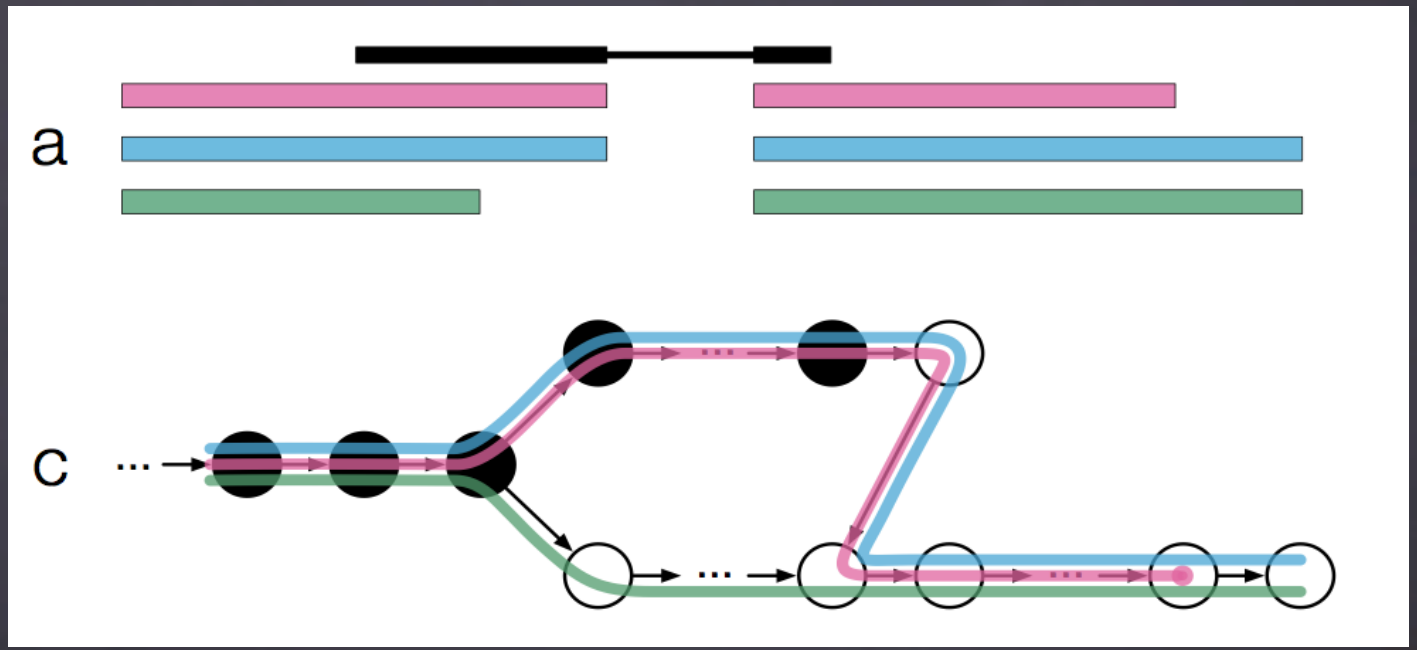


Fig: http://tinyheero.github.io/2015/09/02/pseudoalignments-kallisto.html

KALLISTO

Read pseudoalignment (2):

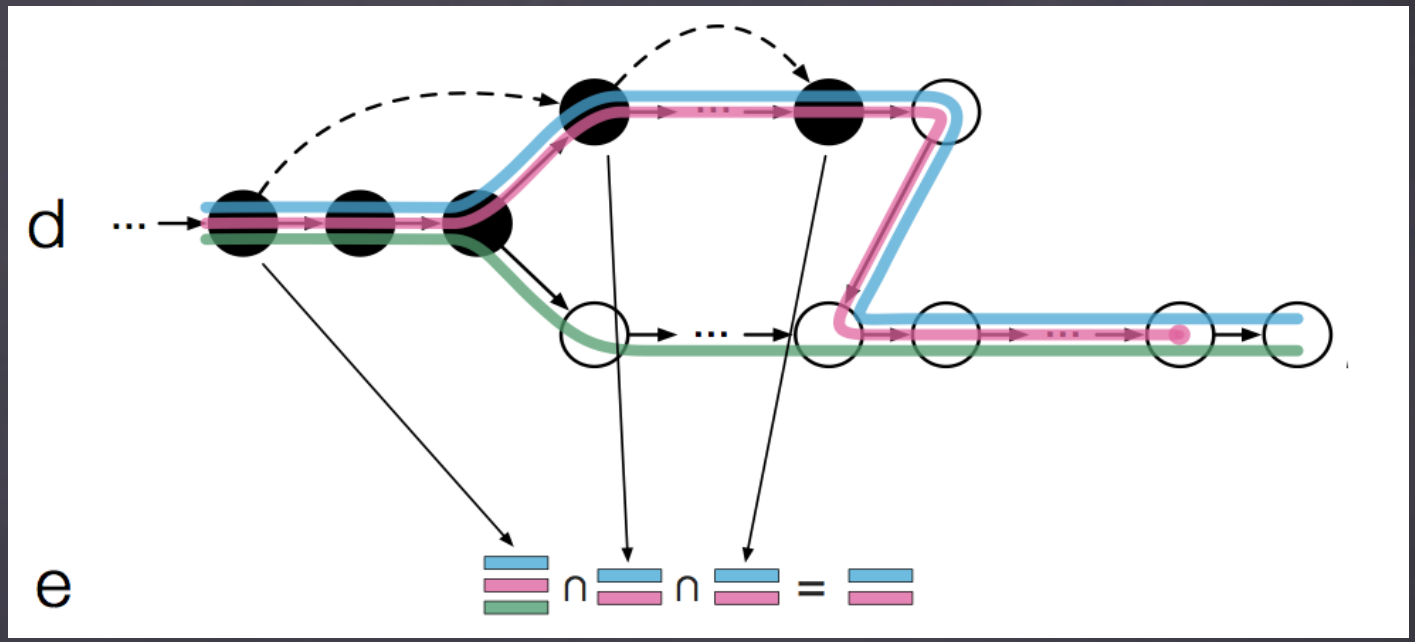
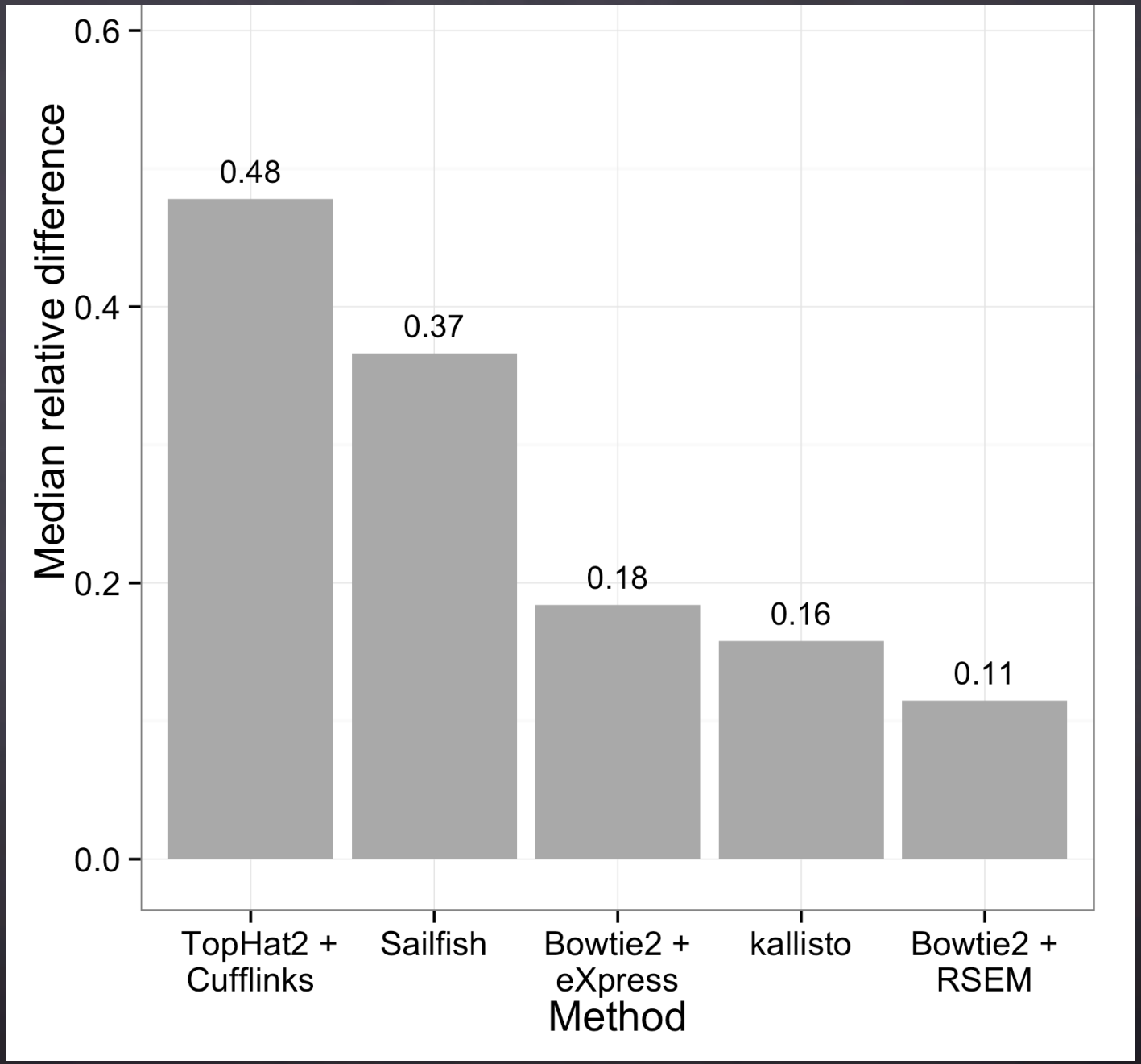


Fig: http://tinyheero.github.io/2015/09/02/pseudoalignments-kallisto.html

Result of pseudoalignment of read is a set of transcripts (no coordinates)

KALLISTO QUANTIF. PERFORMANCE

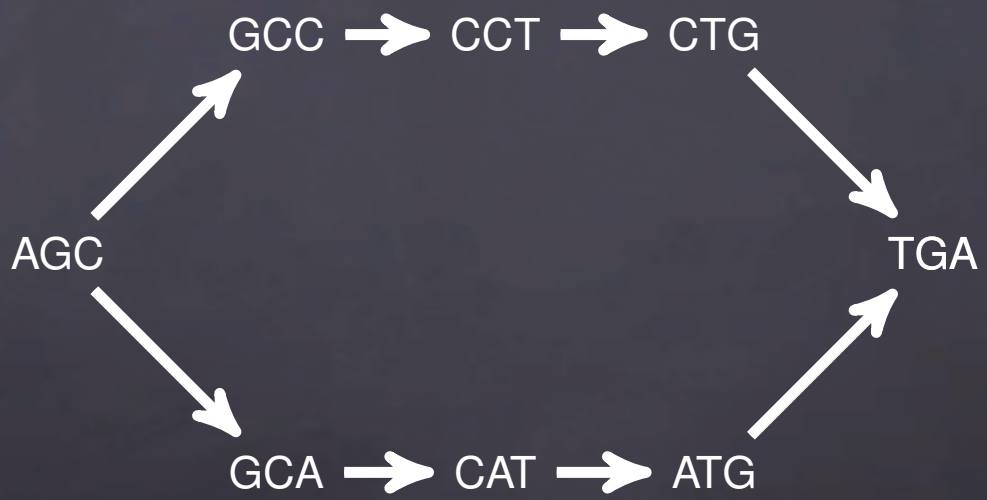


Reference-free variant detection

Core idea: Variants appear as special structures in the dBG.

AGCCTGA
AGCATGA

dBG, $k = 3$:

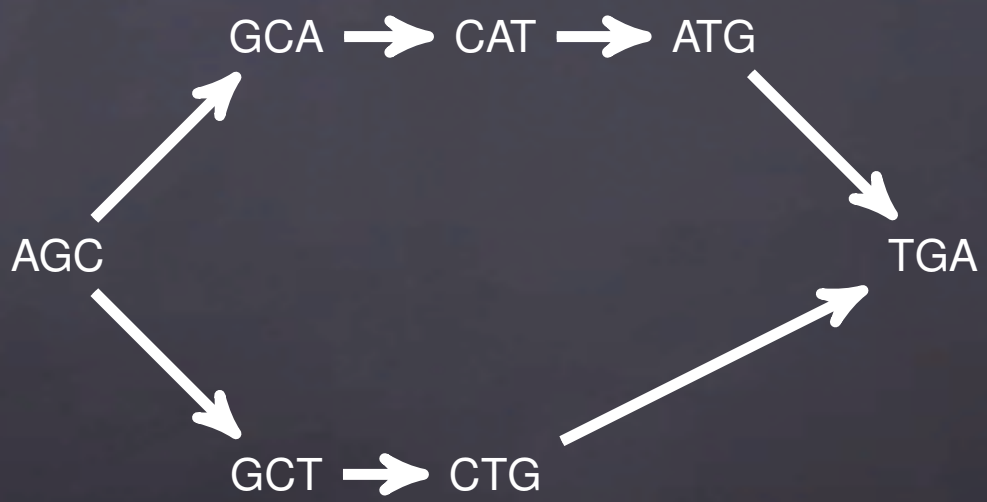


Reference-free variant detection

Small indels:

AGCATGA
AGCTGA

dBG, $k = 3$:



NOT SO SIMPLE IN PRACTICE

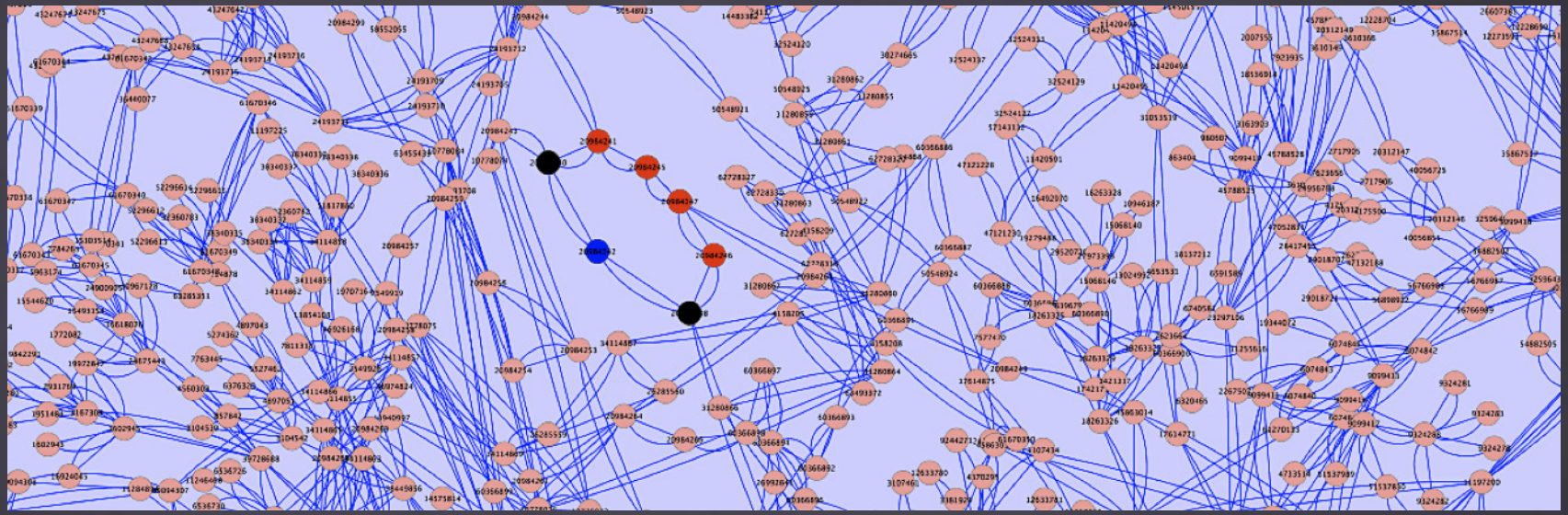


Fig: [Sacomoto et al 2014]

- Bubble structure detection (combinator.)
- Bubble classification: repeat vs. het (stat. criteria)

Reference-free variant detection

Principle:

- *(No reference genome needed)*
- Construct de Bruijn graph of reads
- Detect variant structures

As opposed to reference-based (classical):

- Map reads to reference
- Call variants from pileup (GATK, Freebayes, . . .)

Reference-free variant detection

Software:

- Cortex [Iqbal '12]
- Bubbleparse [Legett '13]
- DiscoSNP++ [Uricaru '14]

Use *colored* de Bruijn graphs.
Given n sequencing datasets,

- Construct de Bruijn graph of union of datasets.
- Nodes are annotated with n coverage values

k-MER ABUNDANCE HISTOGRAM

1) Example reads dataset:

```
ACTCA
GTCA
```

2) 3-mers:

```
ACT
CTC
TCA
GTC
TCA
```

3) Abundance of each distinct 3-mer:

```
ACT: 1
CTC: 1
TCA: 2
GTC: 1
```

4) 3-mer abundance:

x	y
1	3
2	1
3	0
4	0

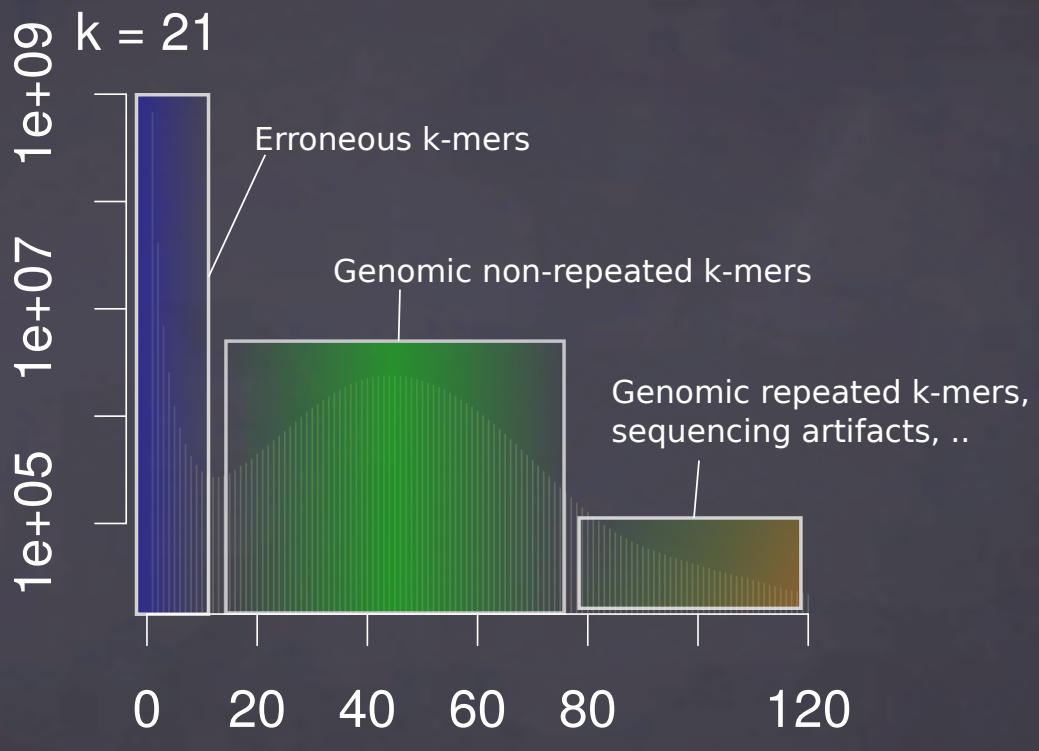
Methods: k-mer counting, e.g. DSK, KMC 2, Jellyfish, ...

k-MER HISTOGRAM STATISTICS

- Quake corrector, SPAdes assembler [Kelley '10, Bankevich '12]
  - Node coverage cut-off (seq. errors)
- SGA PreQC [Simpson '13 (arXiv)]
  - Genome size, graph branch classification, & more
- **KmerGenie** [Chikhi '13]
  - Assembly size, optimal *k* parameter

Dissection of a k-mer histogram

Chr 14 ($\approx$ 88 Mbp) GAGE dataset; histogram $k = 21$



Genomic area

$\approx$

number of distinct k-mers covering the genome

$\approx$

size of the assembly

$\rightarrow$ *How to determine exactly this area?*

Histogram model

We use Quake's model: [Kelley '10]

Erroneous k-mers Pareto distribution with shape α:

$$pdf = \frac{\alpha}{x^{\alpha+1}}$$

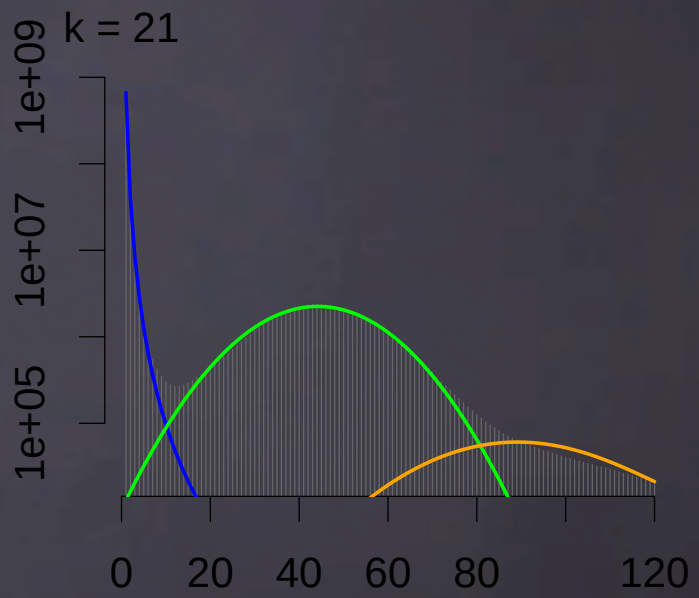
Genomic k-mers Mixture of n Gaussians, weighted by a Zeta distribution of shape s:

$$w_1 X_1 + w_2 X_2 + \ldots + w_n X_n$$

$$X_j \sim \mathcal{N}(j\mu_1, (j\sigma_1)^2)$$

$$P(w_j = k) = k^{-s}/\zeta(s)$$

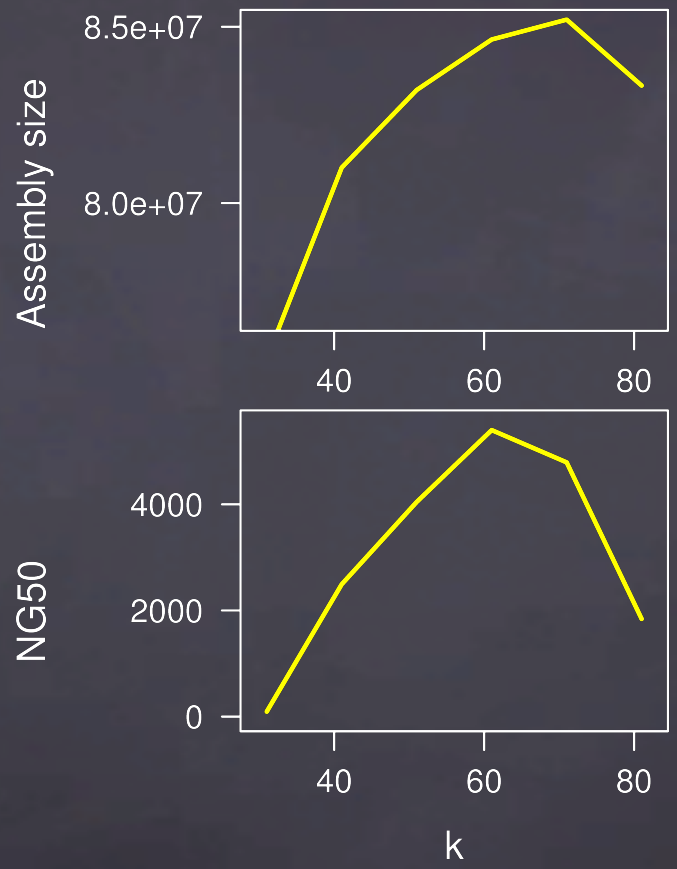
Full model Mixture weighted by $(p_e, 1 - p_e)$.



Numerical optimization (R) is used to fit the model to actual histograms.

Application: finding suitable k value

Genome assembly is **not robust** with respect to k.

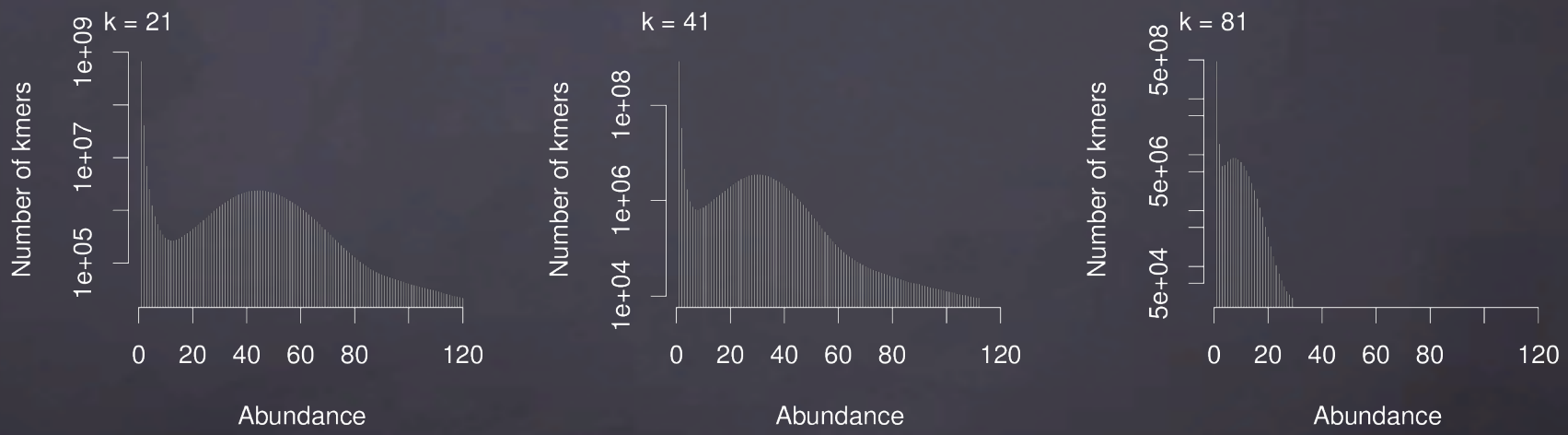


Total length and contiguity (NG50) of chr. 14 (88 Mbp) assemblies

FINDING OPTIMAL k

- Genome is sufficiently covered by k-mers $\Longrightarrow$ good k value
- Requires to know the number of genomic k-mers
- Can be estimated with a k-mer histogram and the Quake model

To find the optimal k, one can compare histograms for different values of k.



Chr 14 ($\approx$ 88 Mbp) GAGE dataset; histograms for three values of k

$\rightarrow$ Issue: computing a single histogram (using k-mer counting) is time and memory expensive

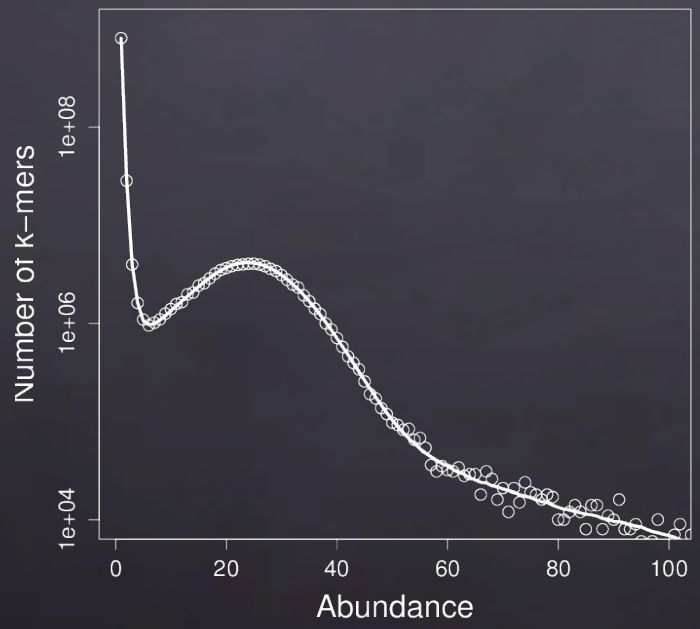
SAMPLING HISTOGRAMS

Organism	CPU time per *k* value
	DSK
S. aureus	2min
chr14	48min
B. impatiens	7.5hour

SAMPLING HISTOGRAMS

Organism	CPU time per k value		Memory usage of Sampling method (GB)
	DSK	Sampling method	
S. aureus	2min	11sec	0.1
chr14	48min	7min	0.1
B. impatiens	7.5hour	1.2hour	0.4

An efficient histogram sampling technique:
Use hashing to sample 1 distinct k-mer out of r (the same k-mer seen in two different reads will be either consistently sampled, either consistently ignored)



- continuous line = exact histogram
- dots = sampled histogram
- sampling errors are visible for low number of k-mers (log scale)
- (Chr 14 ($\approx$ 88 Mbp) $k = 41$)

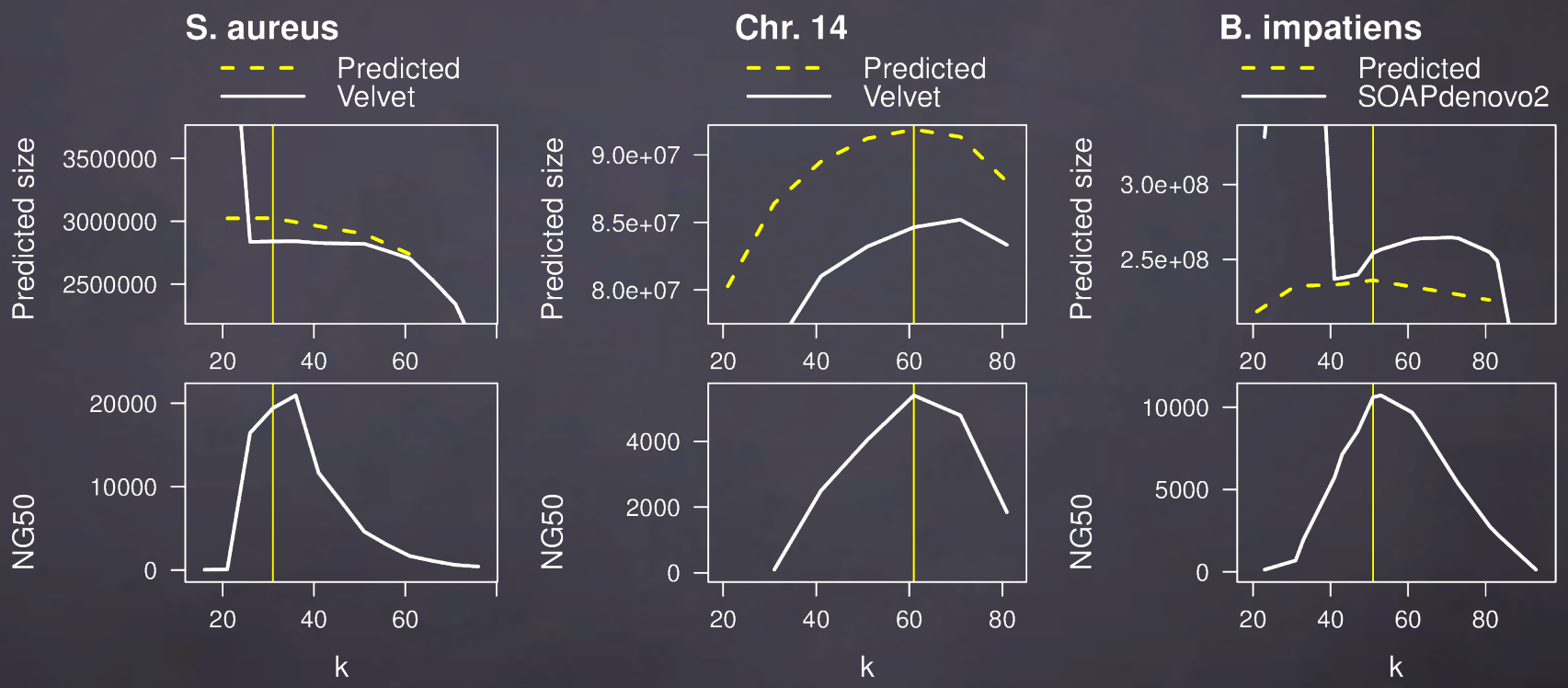
KmerGenie

`KmerGenie` software (`http://kmergenie.bx.psu.edu`)
Joint work with P. Medvedev (Penn State)

- Assembly size prediction
- optimal k prediction
- k-mer histogram sampling

KmerGenie Results: accuracy

Predicted best *k* and predicted assembly size vs actual assembly size and NG50 for 3 organisms (GAGE benchmark).



vertical lines corresponds to predicted best *k*

Open questions on *k*-mer histograms analysis

- Robustness of model
  - low-coverage and very-high coverage
  - polyploidy
  - metaDNA/RNA
- *k* choices in multi-*k* frameworks

Conclusion

de Bruijn graphs

- Tool for reference-free analysis of sequencing data
- Besides assembly, new applications emerge (quantification, variants)
- Information from k-mer histograms

- Practical aspects (mem. usage)
- Software for large de Bruijn graphs: BCALM (`github.com/GATB/bcalm`), GATB library (`www.gatb.fr`)