ATLAS to MetagenomeScope: Visualizing Metagenome Assembly Graphs

Todd J Treangen

Assistant Professor

Department of Computer Science – Rice University

Ready, set, download!

- **First**:
  - ./Chiron/bin/metacompass_interactive
  - python3 /opt/MetaCompass/go_metacompass.py

- **Second**:
  - cd /output
  - wget https://osf.io/xwk7m/download
  - wget https://osf.io/6dmh5/download

- **Third**:
  - wget http://tiny.cc/l65faz
  - wget http://tiny.cc/sa6faz

Docker details

- Where do I save my output?

/output

- How do I exit the container?

exit

- How do I access my output after exiting the container?

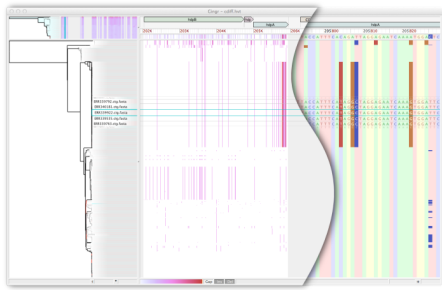
cd /home/stamps19/chiron/metacompass

Acknowledgements

- Mihai Pop (UMD): Metagenomics!
- Marcus Fedarko (UCSD): MetagenomeScope
- Victoria Cepeda (UMD): MetaCompass
- Jay Ghurye (Dovetail Genomics): MetaCarvel

Treangen Lab Members @Rice (Houston, TX)

Research background/interests



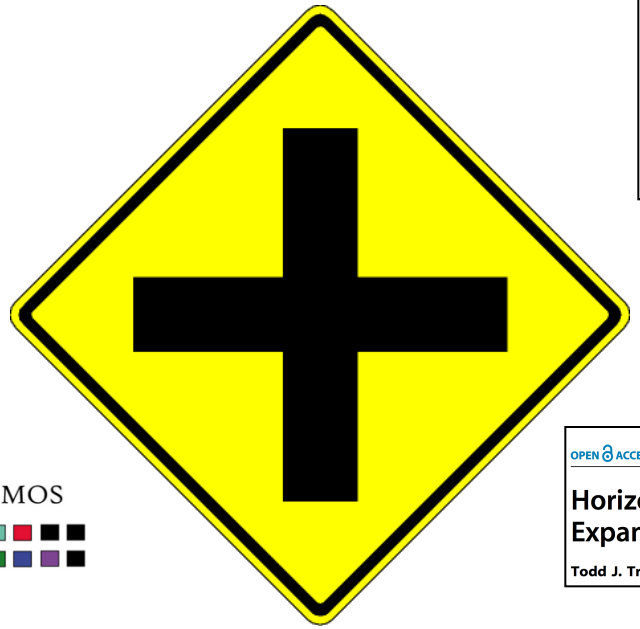
Pathogen Detection

RESEARCH ARTICLE

Identification and Genomic Analysis of a Novel Group C Orthobunyavirus Isolated from a Mosquito Captured near Iquitos, Peru

Todd J. Treangen[1]*, George Schoeler[2¤a], Adam M. Phillippy[1¤b], Nicholas H. Bergman[1], Michael J. Turell[3]

Software Engineering



Microbial Ecology & Evolution

:metAMOS:

a metagenomic assembly pipeline for AMOS

OPEN ACCESS Freely available online PLoS GENETICS

Horizontal Transfer, Not Duplication, Drives the Expansion of Protein Families in Prokaryotes

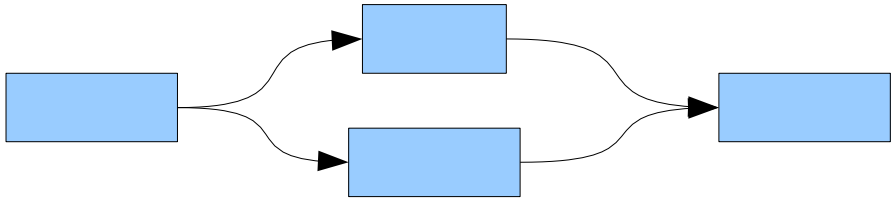
Todd J. Treangen[1,2,3¤]*, Eduardo P. C. Rocha[1,2,3]

Bioinformatics

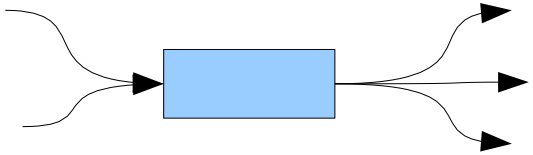
Goals for a metagenomic assembler

- Must work well for clonal data
  - handle repeats
  - handle errors
  - deal with low coverage regions

- Must deal with polymorphisms
  - distinguish between errors and polymorphisms



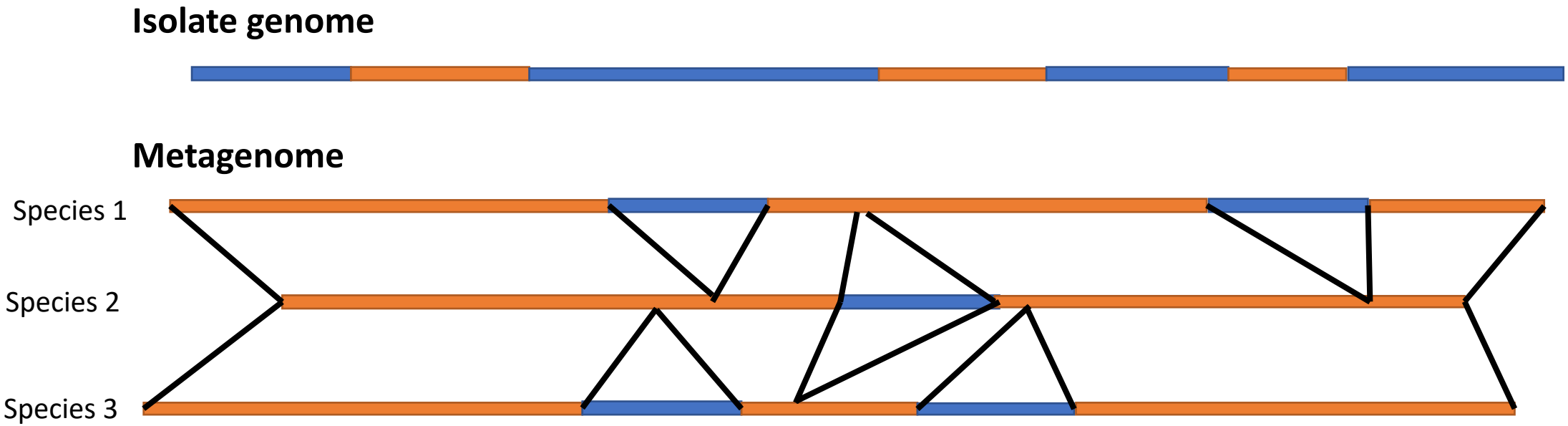
  - distinguish between repeats and polymorphisms



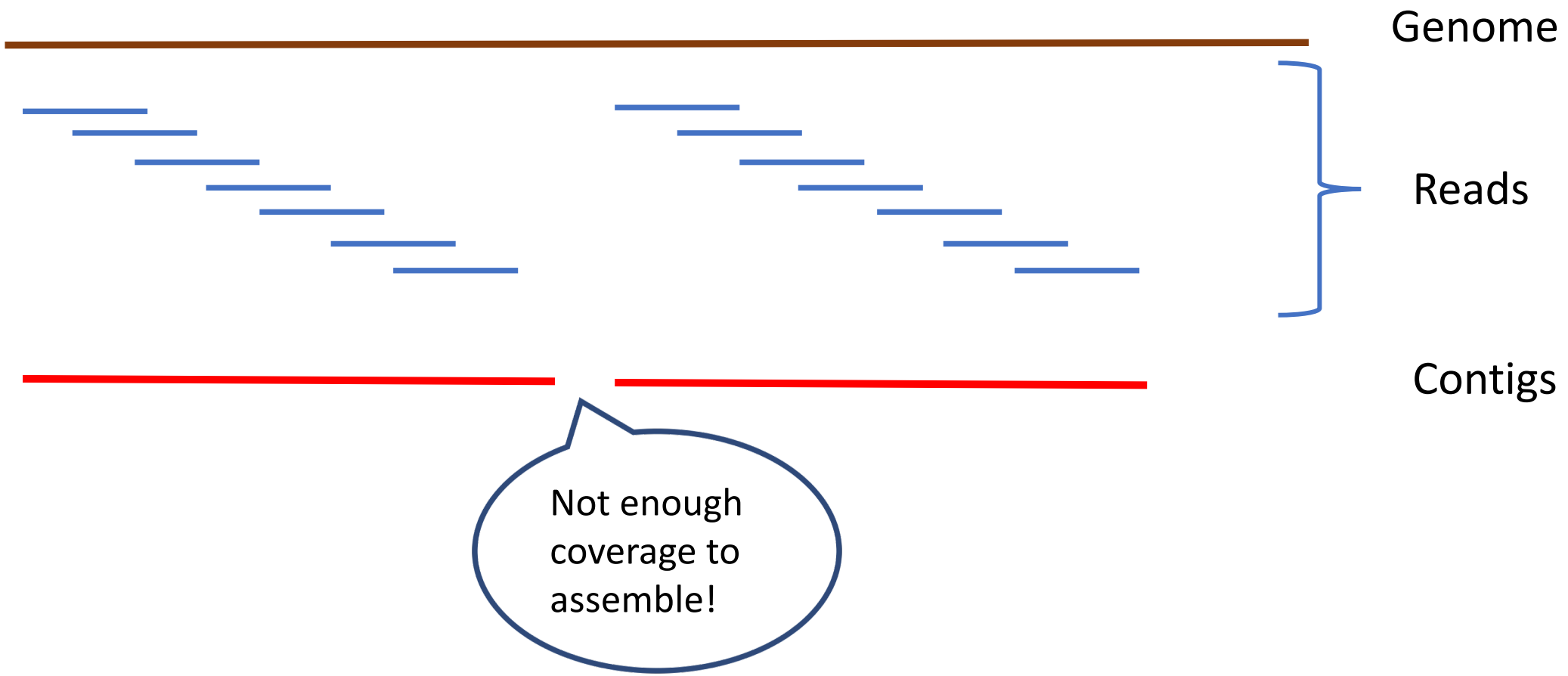
  - enable discovery of polymorphisms/variation

Repeats in metagenome

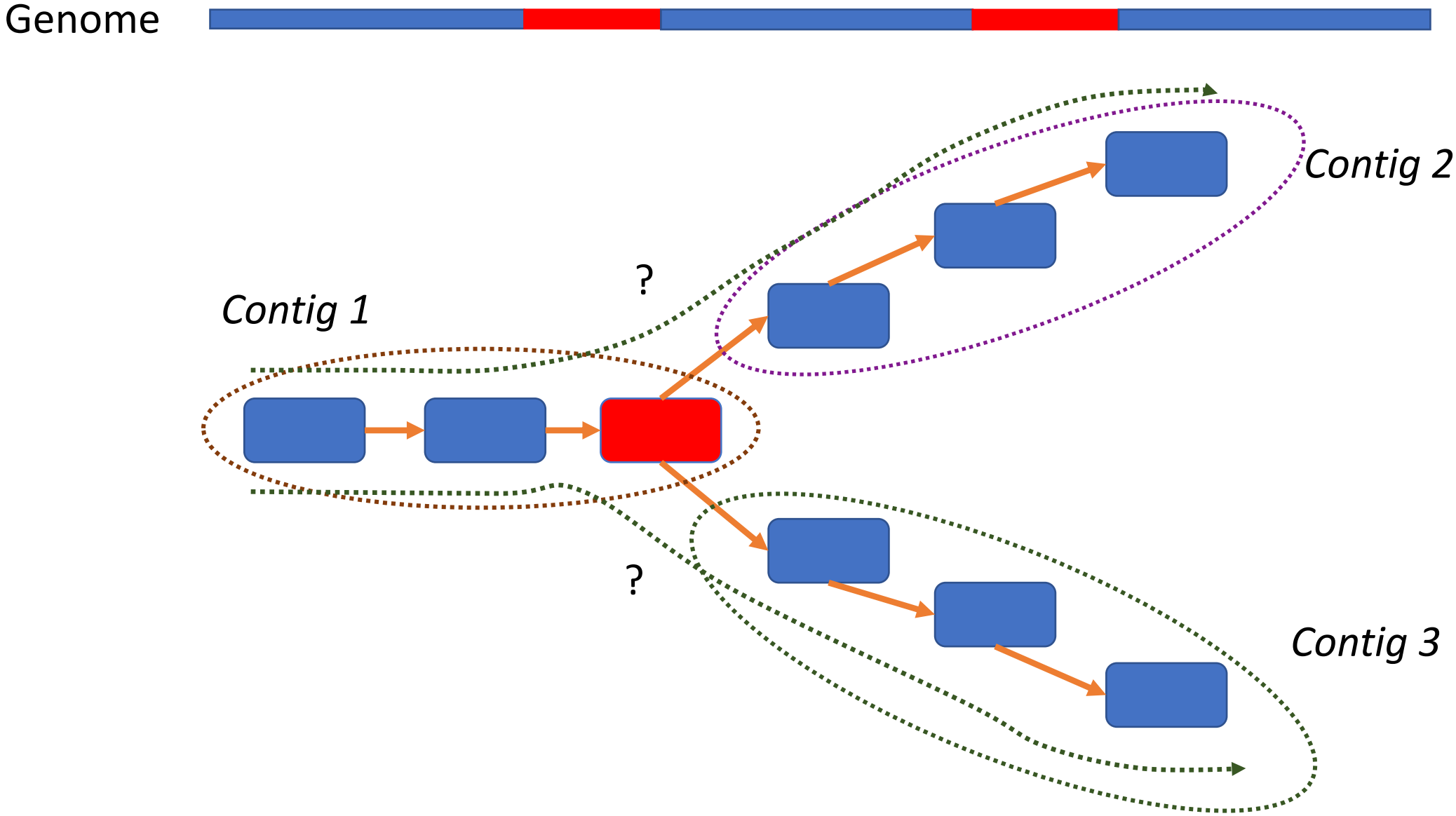
- High coverage regions -> abundant organisms
- Repeats are genome sized due to closely related strains in the sample



Why assemblers break contigs?

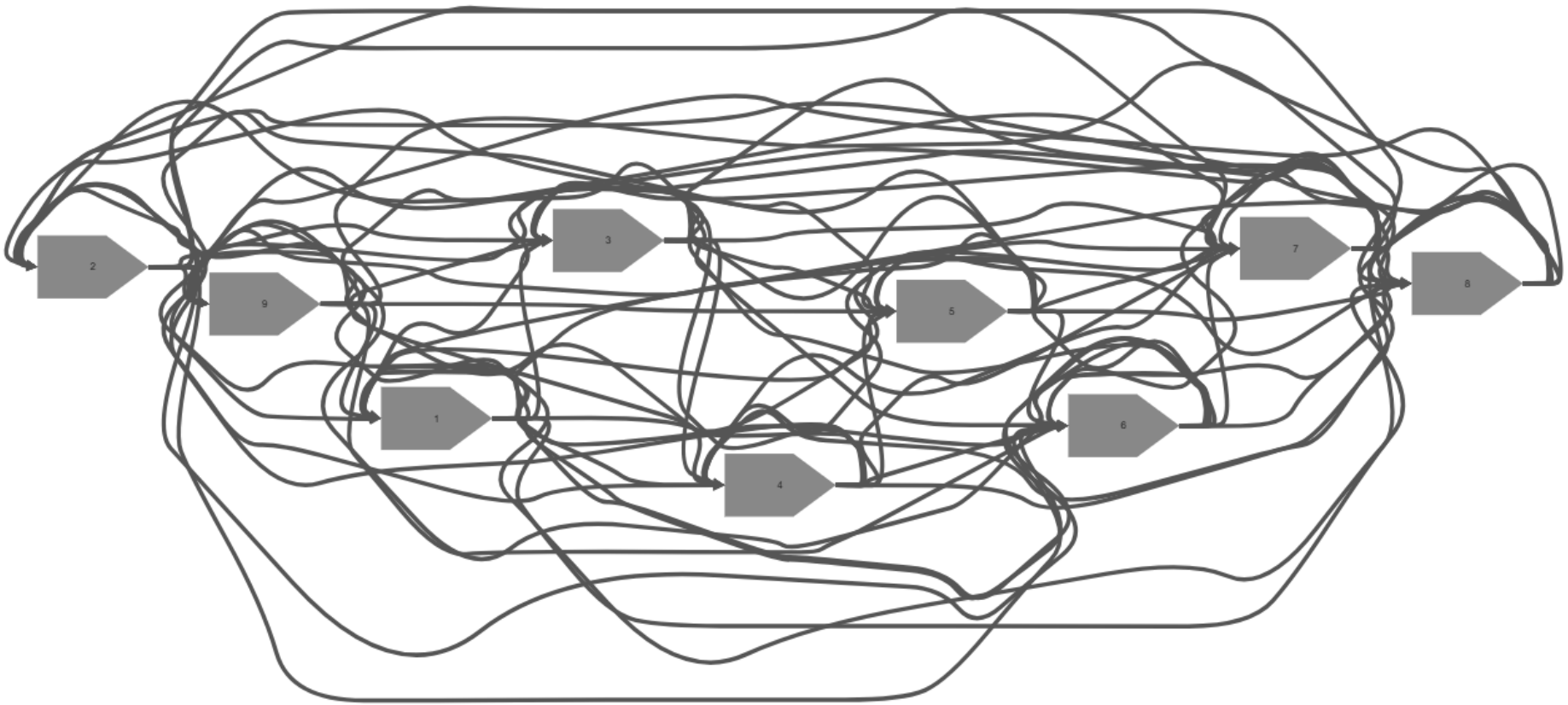


Why assemblers break contigs?



Example 1: AAAAAAAAAAAAAA (no unique 3-mer)

- 4bp reads:
  - (1) AAAA, (2) AAAA, (3) AAAA, (4) AAAA, (5) AAAA, (6) AAAA, (7) AAAA, (8) AAAA, (9) AAAA
- 3-mers:
  - AAA
- Overlaps:
  - (1)->(2),(1)->(3),(1)->(4),(1)->(5),(1)->(6),(1)->(7),(1)->8,(1)->(9)
  - (2)->(1),(2)->(3),(2)->(4),(2)->(5),(2)->(6),(2)->(7),(2)->8,(2)->(9)
  - ….
  - (9)->(1),(9)->(2),(9)->(3),(9)->(4),(9)->(5),(9)->(6),(9)->(7),(9)->(8)



Slide credits: MetagenomeScope

Example 2: AATCCGTTCGGA (no 3-mer repeats)

- 4bp reads:
  - (1) AATC, (2) ATCC, (3) TCCG, (4) CCGT, (5) CGTT, (6) GTTC, (7) TTCG, (8) TCGG, (9) CGGA
- 3-mers (10)
  - (i) AAT, (ii) ATC, (iii) TCC, (iv) CCG, (v) CGT, (vi) GTT, (vii) TTC, (viii) TCG, (ix) CGG, (x) GGA
- Overlaps
  - (1)->(2)
  - (2)->(3)
  - (3)->(4)
  - (4)->(5)
  - (5)->(6)
  - (6)->(7)
  - (7)->(8)
  - (8)->(9)

Slide credits: MetagenomeScope

```
AATCCGTTCGGA

AATC
 ATCC
  TCCG
   CCGT
    CGTT
     GTTC
      TTCG
       TCGG
        CGGA
```

Example 3: AATCCGTTCGGA (sequencing error)

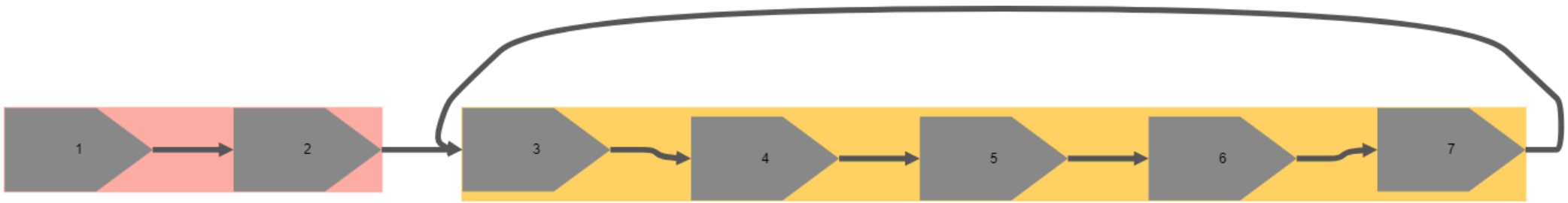
- 4bp reads:
  - (1) AAT**G**, (2) ATCC, (3) TCCG, (4) CCGA, (5) CGTT, (6) GTTC, (7) TTCG, (8) TCGG, (9) CGGA
- 3-mers (10)
  - (i) AAT, (ii) ATC, (iii) TCC, (iv) CCG, (v) ATG, (vi) CGT, (vii) GTT, (viii) TTC, (xi) TCG, (x) CGG, (xi) GGA
- Overlaps
  - ~~(1)->(2)~~
  - (2)->(3)
  - (3)->(4)
  - (4)->(5)
  - (5)->(6)
  - (6)->(7)
  - (7)->(8)
  - (8)->(9)

```
ATCCGTTCGGA


ATCC
 TCCG
  CCGT
   CGTT
    GTTC
     TTCG
      TCGG
       CGGA
```

Example 4: AATCCGTTCGGA (sequencing error)

- 4bp reads:
  - (1) AATC, (2) ATCC, (3) TCCG, (4) CCGT, (5) CGTT, (6) GTTC, (7) TTC**C**, (8) TCGG, (9) CGGA
- 3-mers (10)
  - (i) AAT, (ii) ATC, (iii) TCC (X2), (iv) CCG, (v) CGT, (vi) GTT, (vii) TTC, (viii) TCG, (ix) CGG, (x) GGA
- Overlaps
  - (1)->(2)
  - (2)->(3)
  - (3)->(4)
  - (4)->(5)
  - (5)->(6)
  - (6)->(7)
  - ~~(7)->(8)~~
  - (7)->(3)
  - (8)->(9)



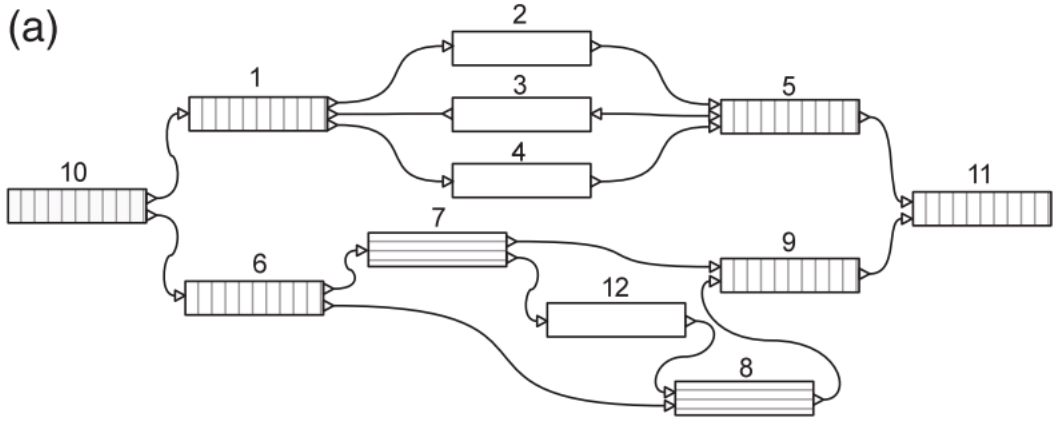
Slide credits: MetagenomeScope

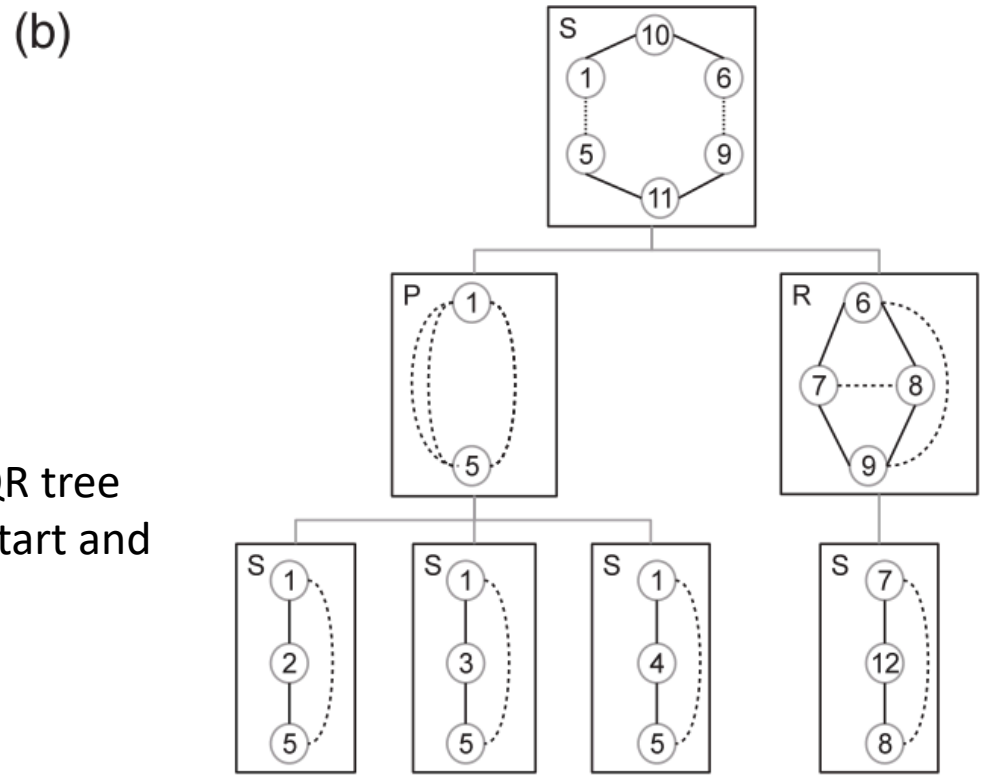
Example 5: AATCCGTTCGGA (coverage gap)

- 4bp reads:
  - (1) AATC, (2) ATCC, (3) TCCG, ~~(4) CCGT, (5) CGTT~~, (6) GTTC, (7) TTCG, (8) TCGG, (9) CGGA
- 3-mers (10)
  - (i) AAT, (ii) ATC, (iii) TCC, (iv) CCG, (v) TTC, (vi) TCG, (vii) CGG, (viii) GGA
- Overlaps
  - (1)->(2)
  - (2)->(3)
  - ~~(3)->(4)~~
  - ~~(4)->(5)~~
  - ~~(5)->(6)~~
  - (6)->(7)
  - (7)->(8)
  - (8)->(9)

Variant Detection Using Graph Topology

SPQR Tree





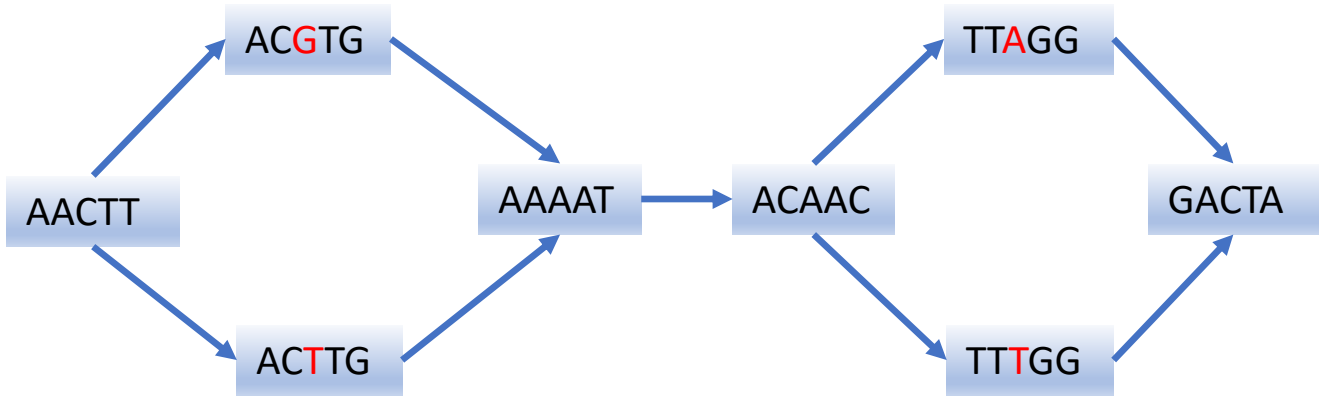
If two nodes lie in the same node of SPQR tree and share a virtual edge, then they are start and end of a bubble!

Figure credits: Nijkamp et al., 2013

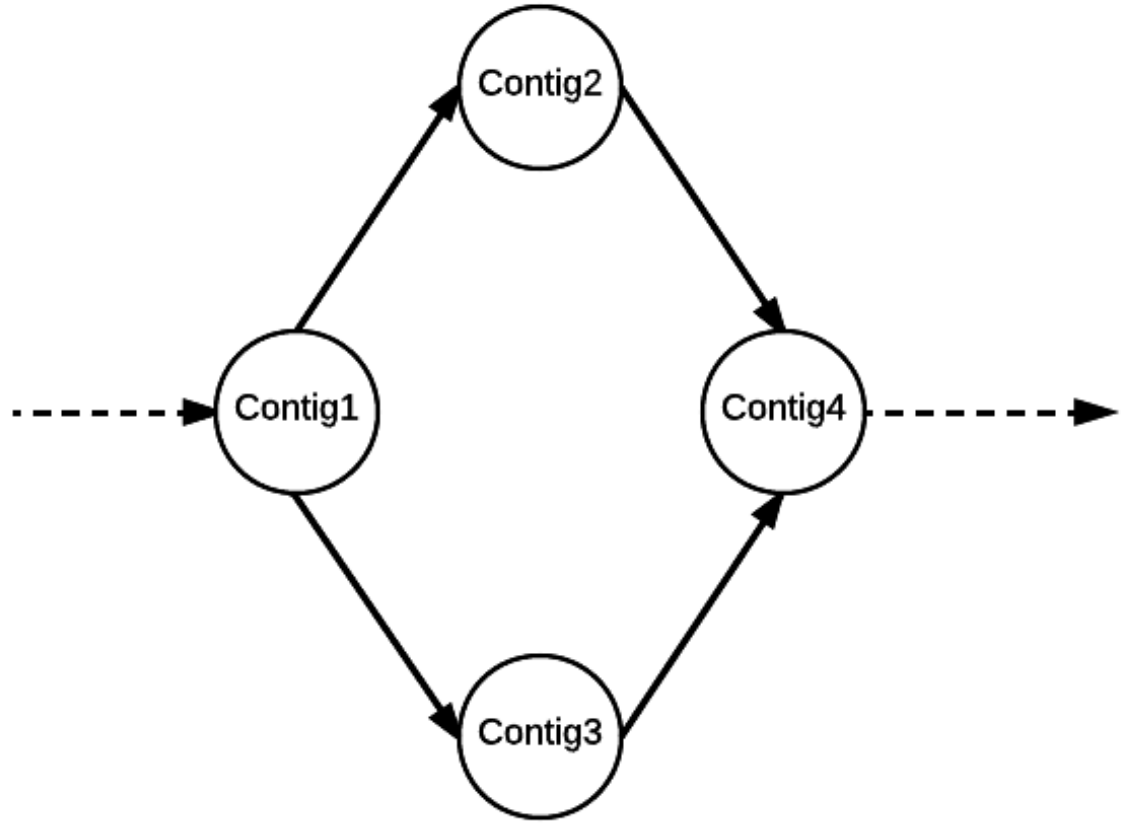
Variant detection using graph topology

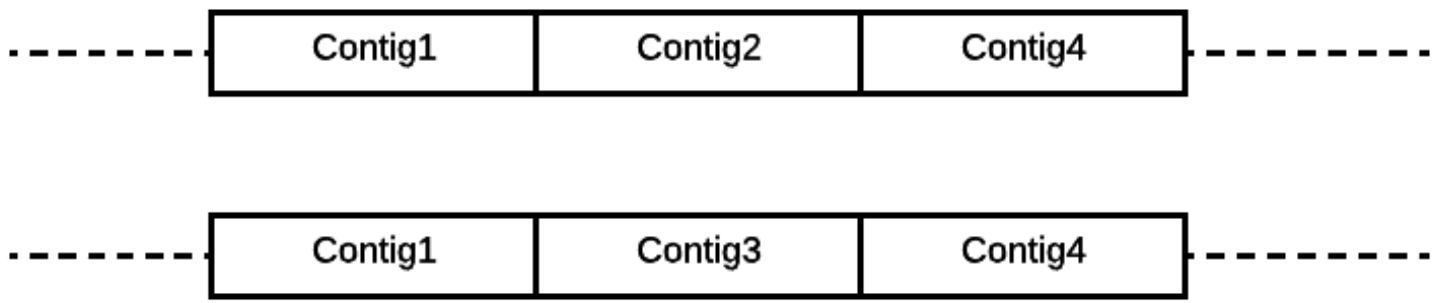
Species 1 …AACTTACGTGAAAATACAACTTAGGACTA…

Species 2 …AACTTACTTGAAAATACAACTTTGGACTA..

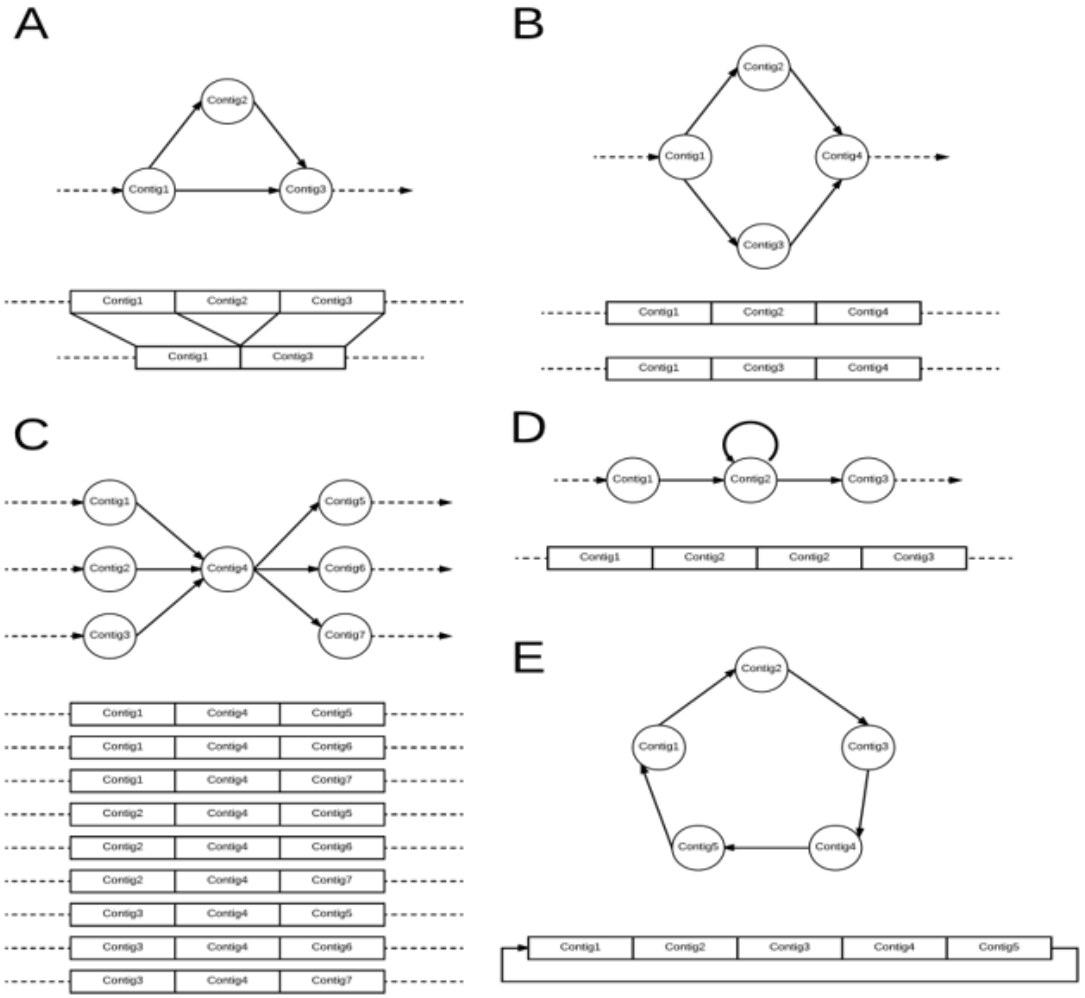


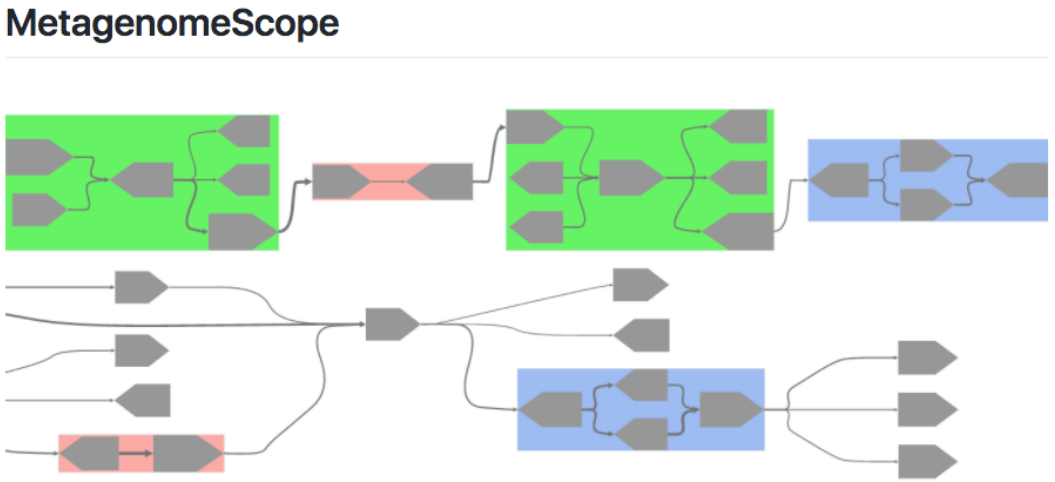
- Variations in the sequences cause bubble in the graph
- Naïve search for all bubble is exponential in the number of nodes
- Use a SPQR tree based approach to find out bubbles (Nijkamp et al. 2013)
- Runs in O(V+E) time (Gutwenger et al. 2001)


Contig2
Contig1
Contig4
Contig3


Contig1
Contig2
Contig4
Contig1
Contig3
Contig4

MetaCarvel and MetagenomeScope





Overview of the different types of variants detected by MetaCarvel.

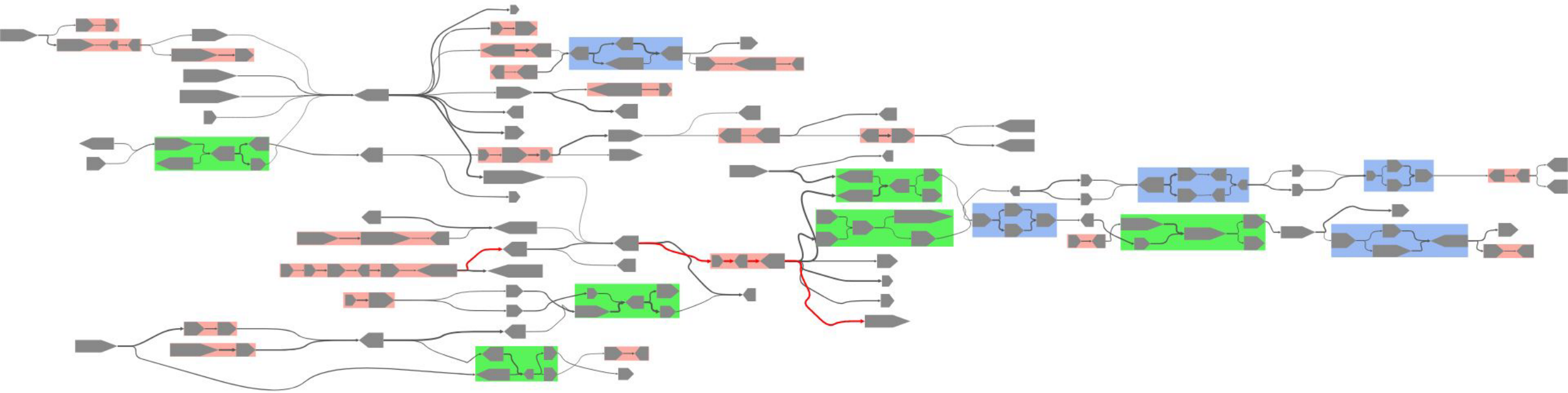
We focus on five types of variants:
A) Triangular bubbles,
B) Simple four bubbles
C) High centrality nodes
D) Simple Cycles, and
E) Multinode cycles

https://github.com/marbl/MetaCarvel

https://github.com/marbl/MetagenomeScope

Assembly Graphs

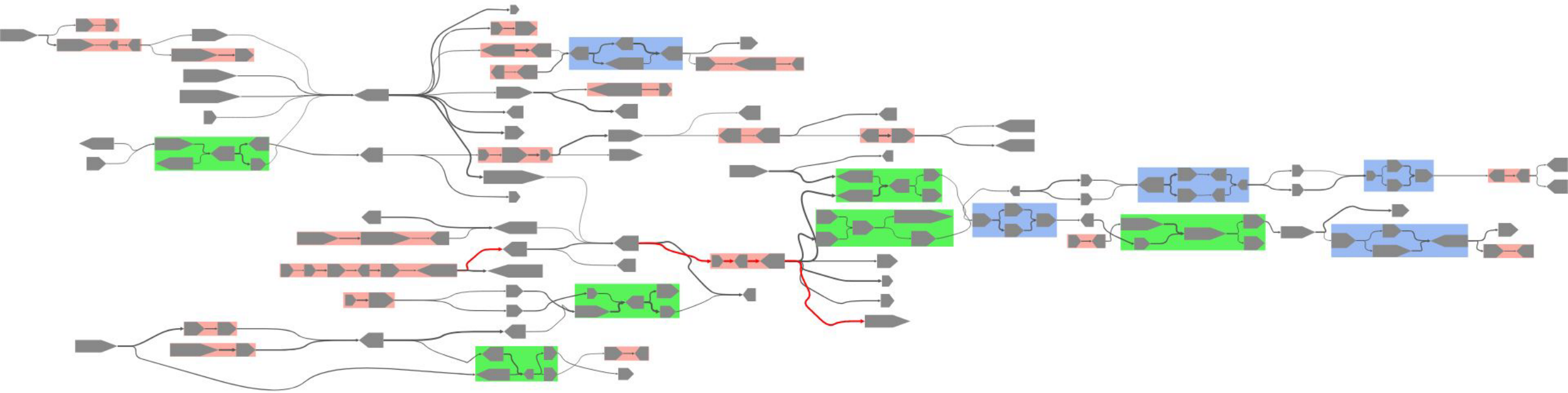
(Meta)genome assembly can be represented as a graph traversal



Assembly Graphs

(either a **de Bruijn graph** or an **overlap graph)**

(Meta)genome assembly can be represented as a graph traversal

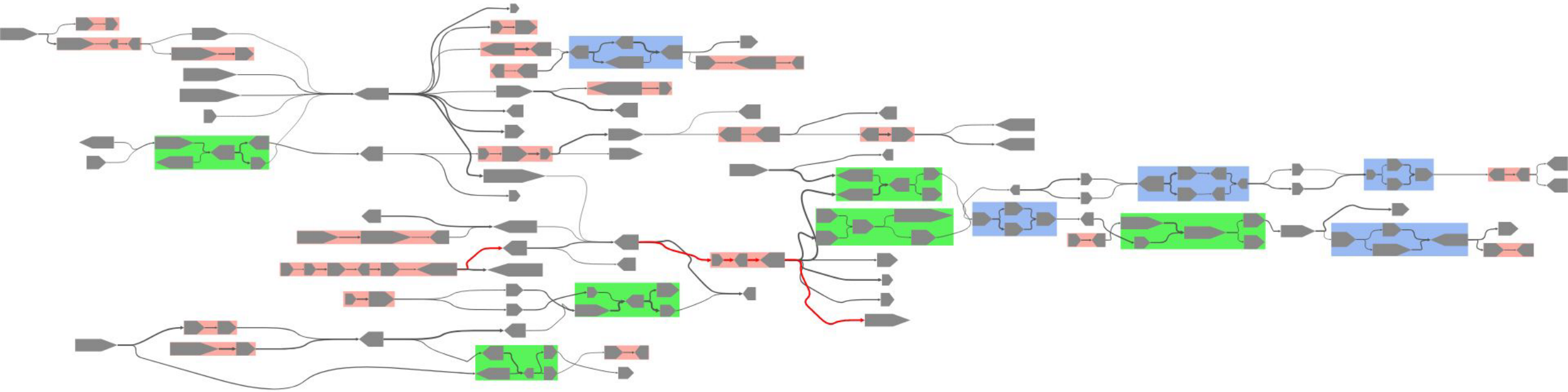


Assembly Graphs

(either a **de Bruijn graph** or an **overlap graph)**

(Meta)genome assembly can be represented as a graph traversal

In either case...



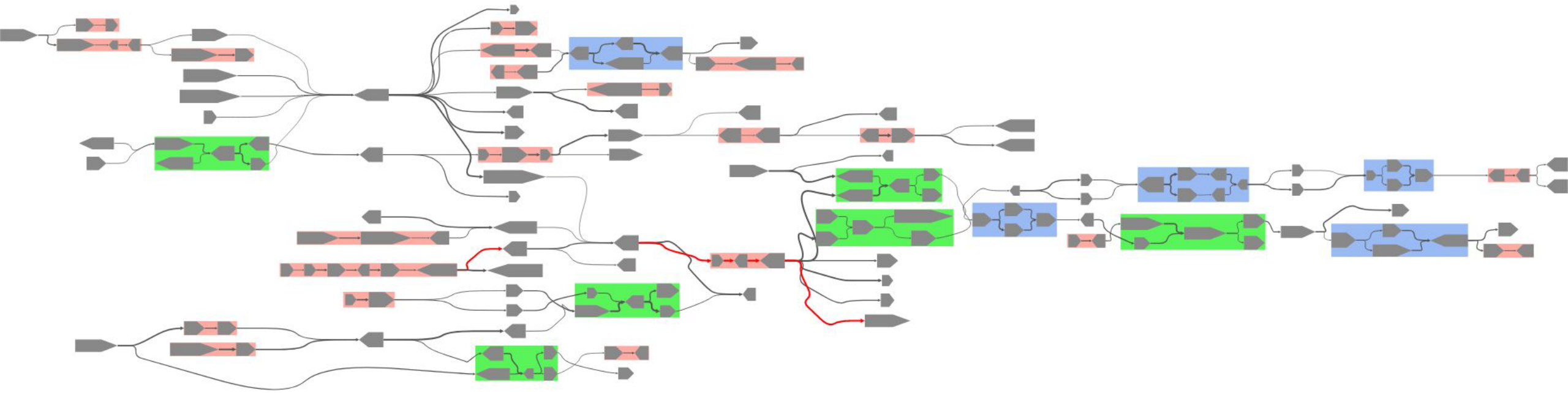
Assembly Graphs

(either a **de Bruijn graph** or an **overlap graph)**

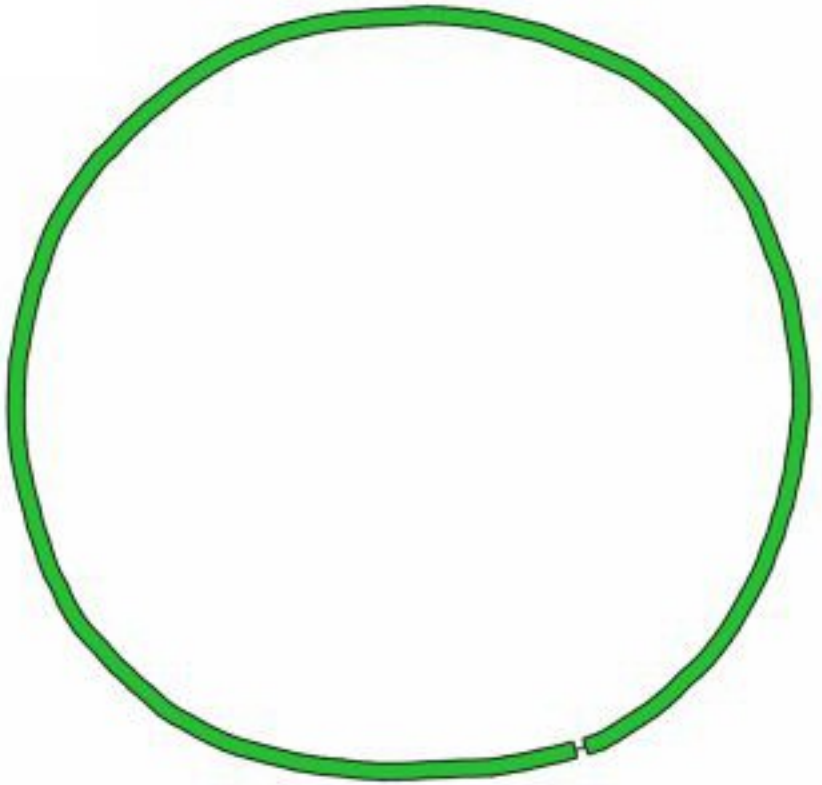
(Meta)genome assembly can be represented as a graph traversal

Nodes (*contigs*) - fragments of DNA obtained from assembly

Edges - overlaps between contigs' sequences



Assembly Graphs: The Good



Ideal genome assembly graph:

One node per sequence, maybe with an edge to itself

Ideal metagenome assembly graph:

c graphs, where each is an “ideal genome assembly graph”

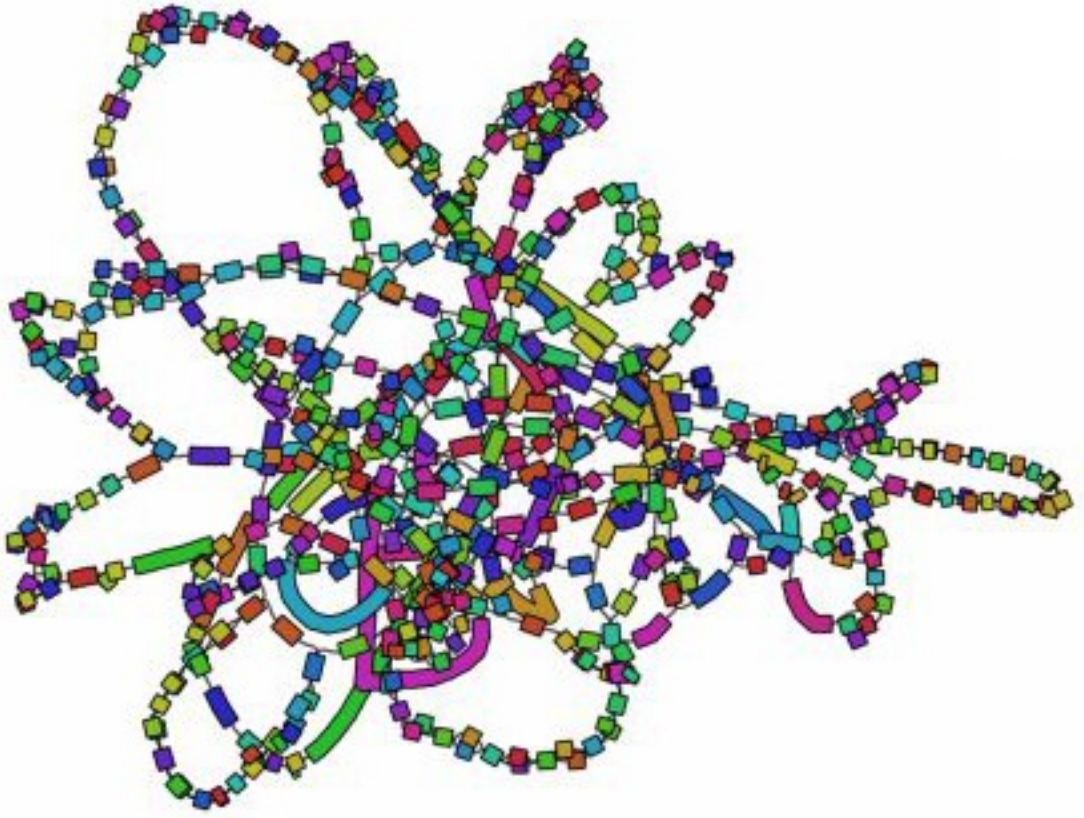
(assuming the input metagenome consists of *c* genomes)

These figures c/o the Bandage manuscript (Wick et al. 2015)

Assembly Graphs: The Bad /ugly

Actual genome assembly graph:

Lots of possible paths due to complexities like repeats, mutations, assembly errors, ...



Actual metagenome assembly graph:

More potential for these complexities to “compound” → even more fragmented

These figures c/o the Bandage manuscript (Wick et al. 2015)

Problem

How can we visualize metagenome assembly graphs in a way that highlights both

1. **small-scale details**, and
2. **large-scale structure**?

Your turn!

- https://gitlab.com/treangen/stamps2019/README.md#part1

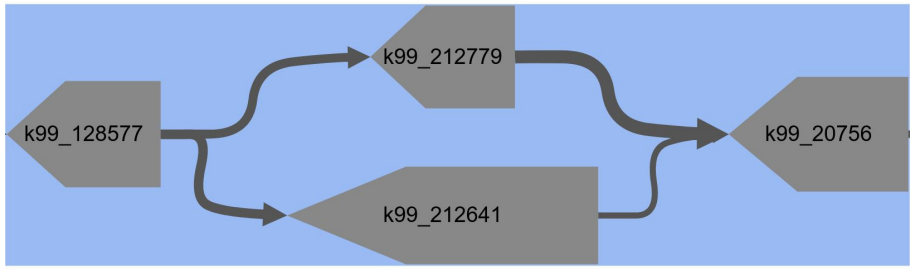
Exercise #1: Drawing Something

1. Open MetagenomeScope's viewer interface at mgsc.umiacs.io.
2. Pick a demo assembly graph using the "Demo .db" button.
3. Draw one of the graph's connected components[1]!

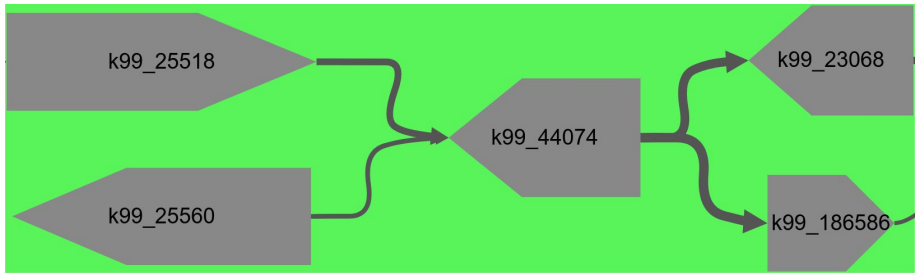
[1] "Weakly" connected components

Identifying Structural Patterns in Assembly Graphs

(See Miller, Koren, Sutton 2010)

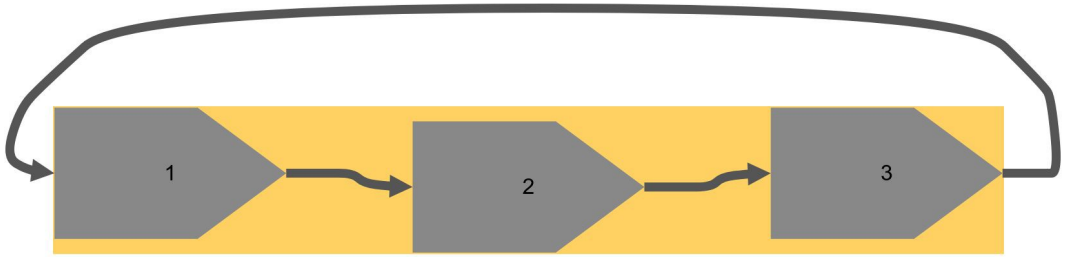


Bubble



Frayed Rope

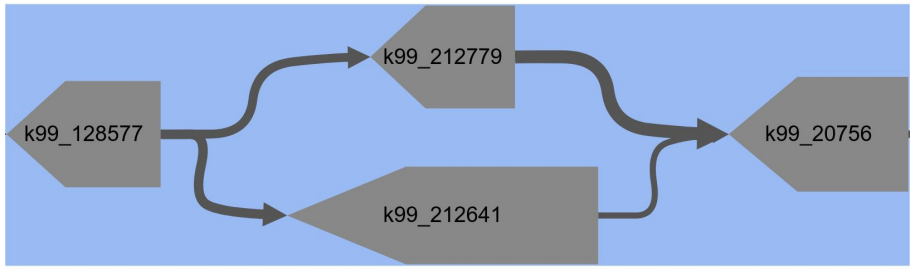
Chain



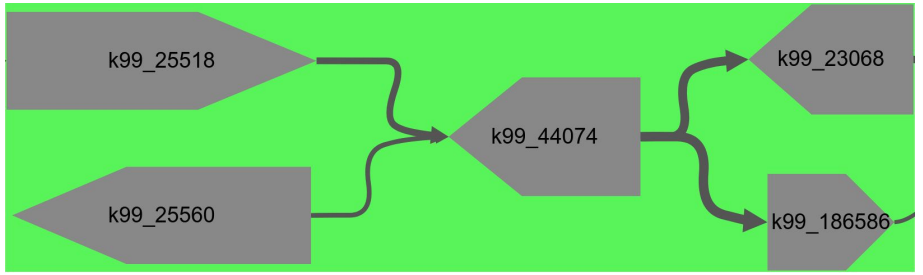
Cyclic Chain

Identifying Structural Patterns in Assembly Graphs

(See Miller, Koren, Sutton 2010)

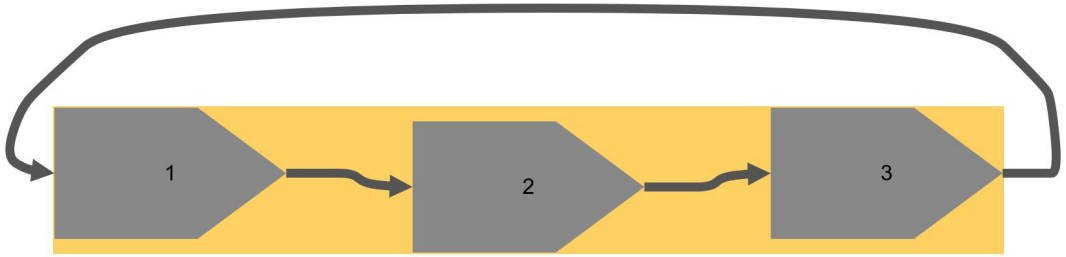


Bubble

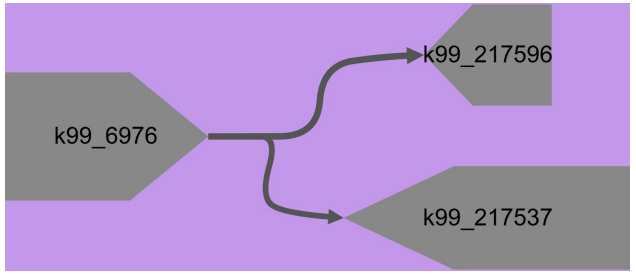


Frayed Rope

Chain



Cyclic Chain

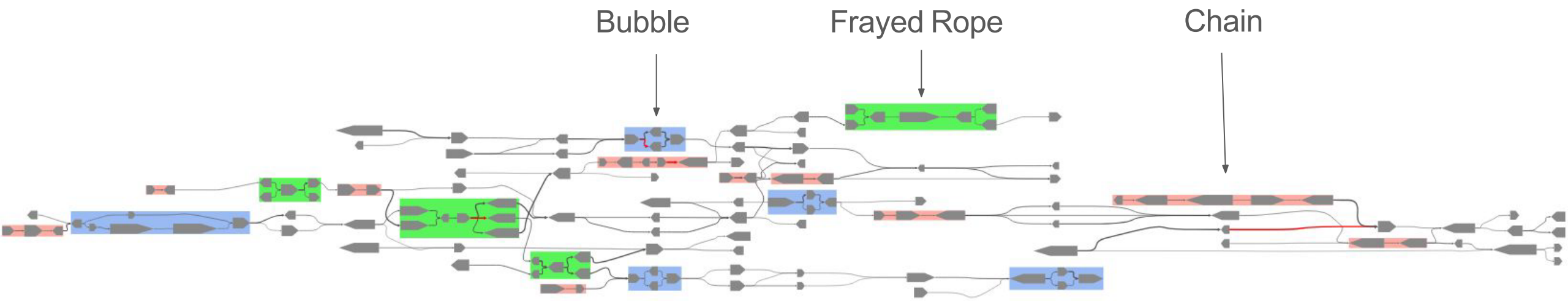


User-Defined Structures

Identifying Structural Patterns in Assembly Graphs

(See Miller, Koren, Sutton 2010)

Patterns can be collapsed/uncollapsed to decrease/increase graph complexity.

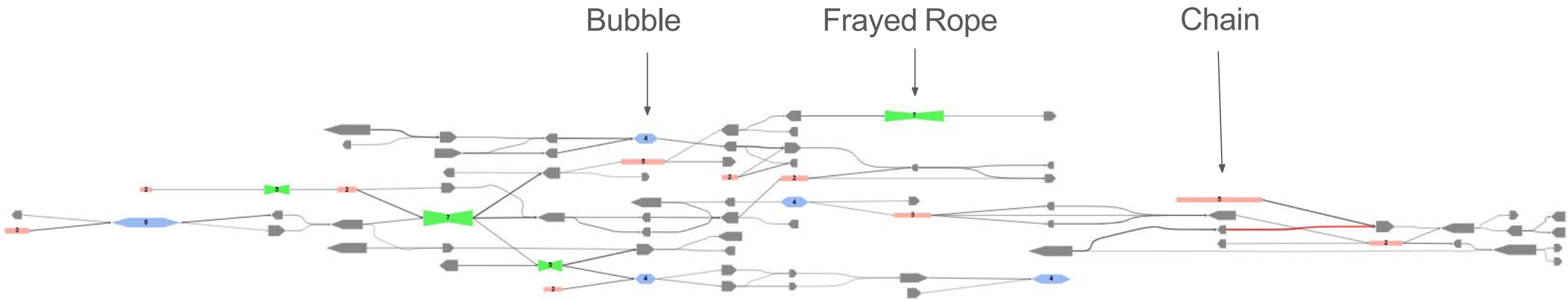


region of an assembly graph created from Human Microbiome Project (HMP) data, accession ID SRS049959

Identifying Structural Patterns in Assembly Graphs

(See Miller, Koren, Sutton 2010)

Patterns can be collapsed/uncollapsed to decrease/increase graph complexity.

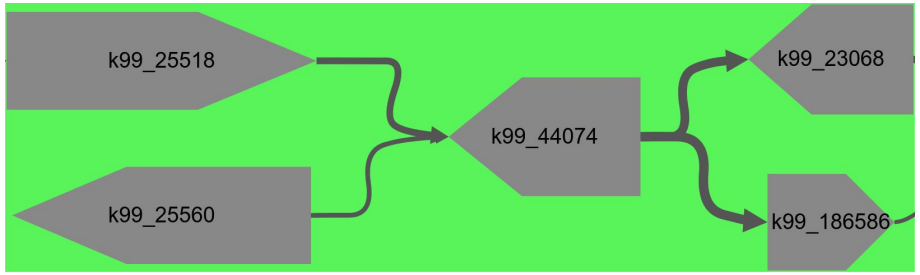


region of an assembly graph created from Human Microbiome Project (HMP) data, accession ID SRS049959

Exercise #2: Finding a frayed rope

https://gitlab.com/treangen/stamps2019

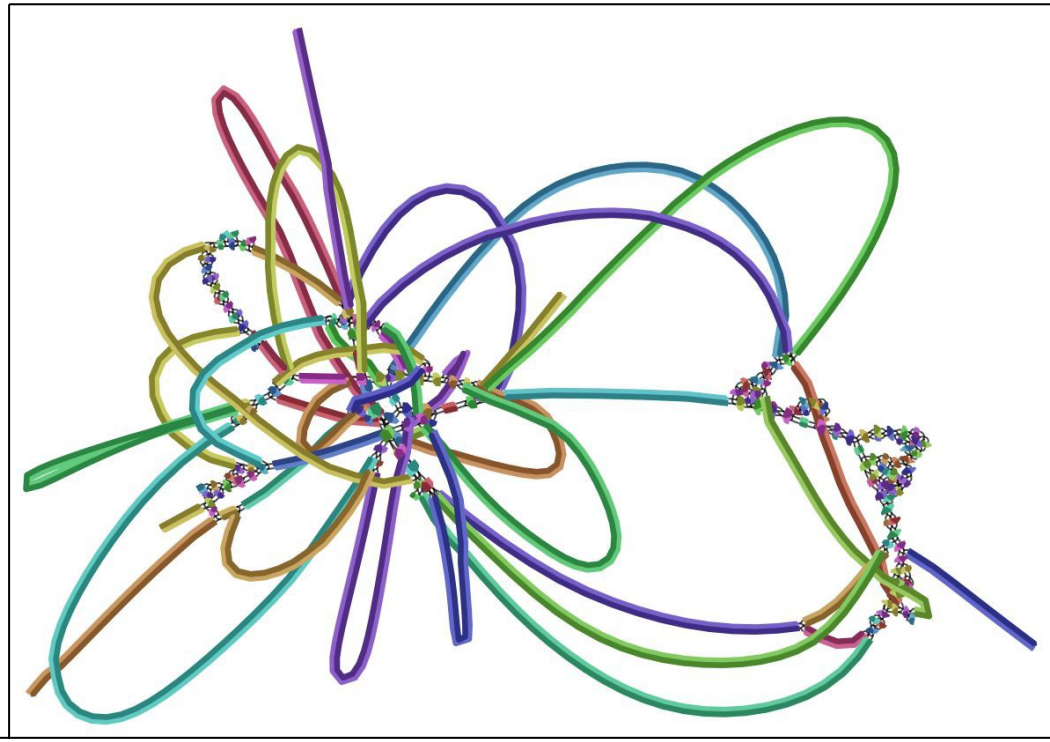
(this sort of thing)



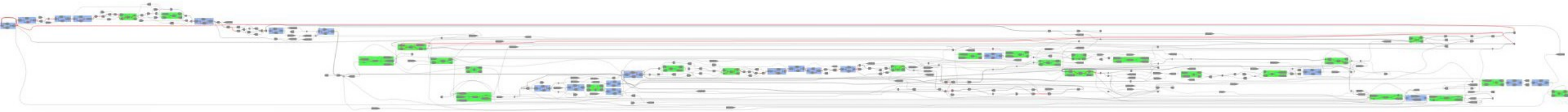
Hierarchical Graph Layout

Bandage, ABySS-Explorer, and Ray Cloud Browser all use **force-directed** layout algorithms

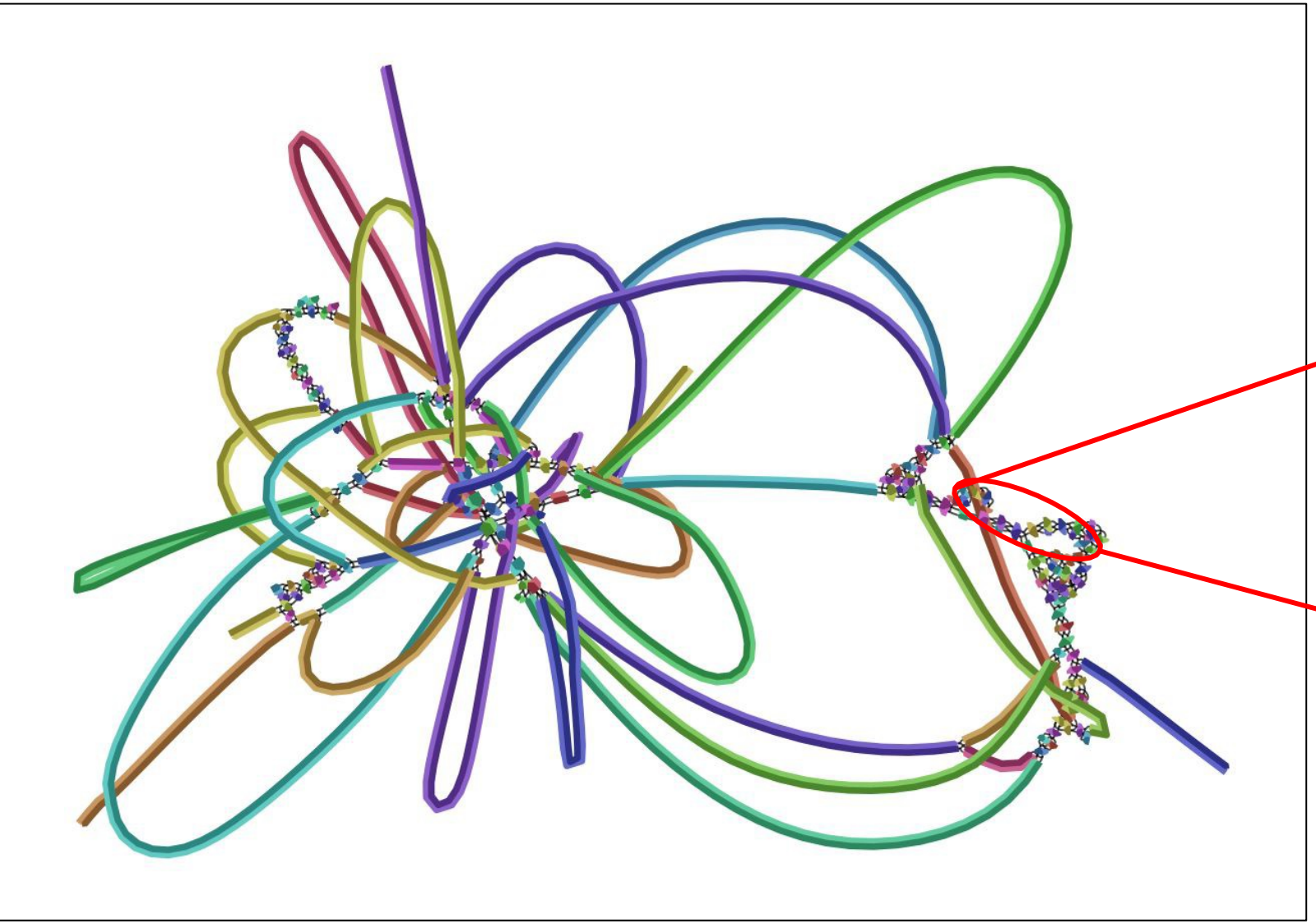
MetagenomeScope uses Graphviz' dot program to **hierarchically** lay out graphs

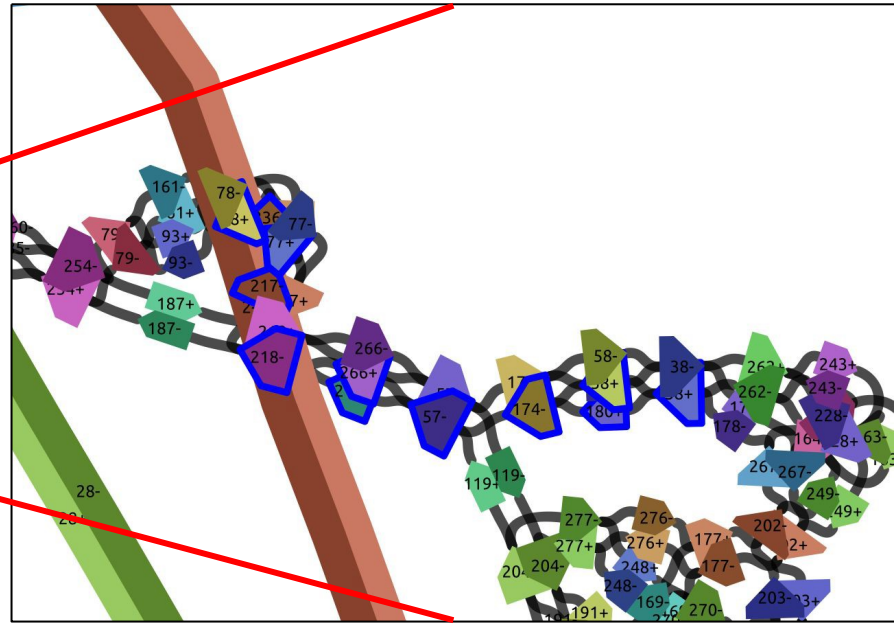


Largest connected component of an E. coli assembly graph produced with Velvet, visualized in Bandage (right) and MetagenomeScope (bottom)

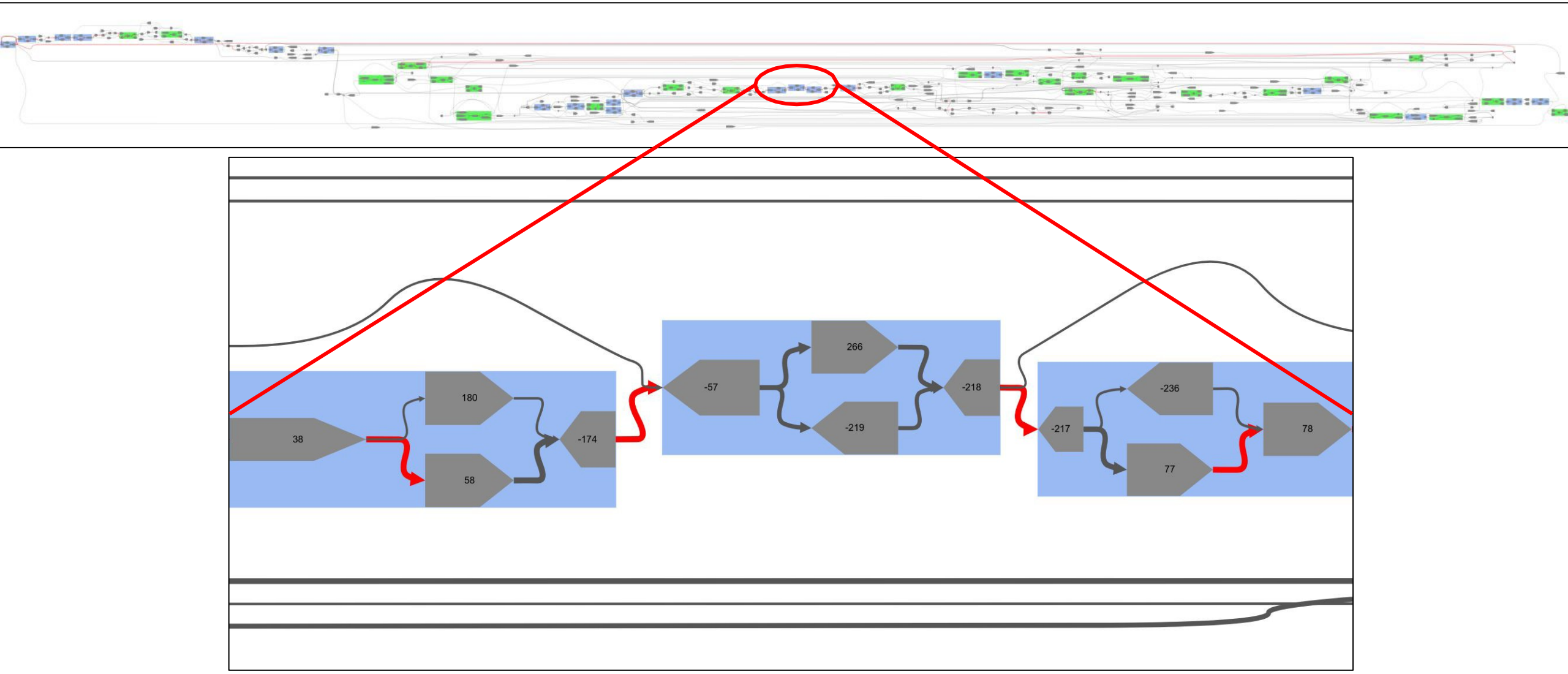


Reviewing that E. coli assembly graph (Bandage)



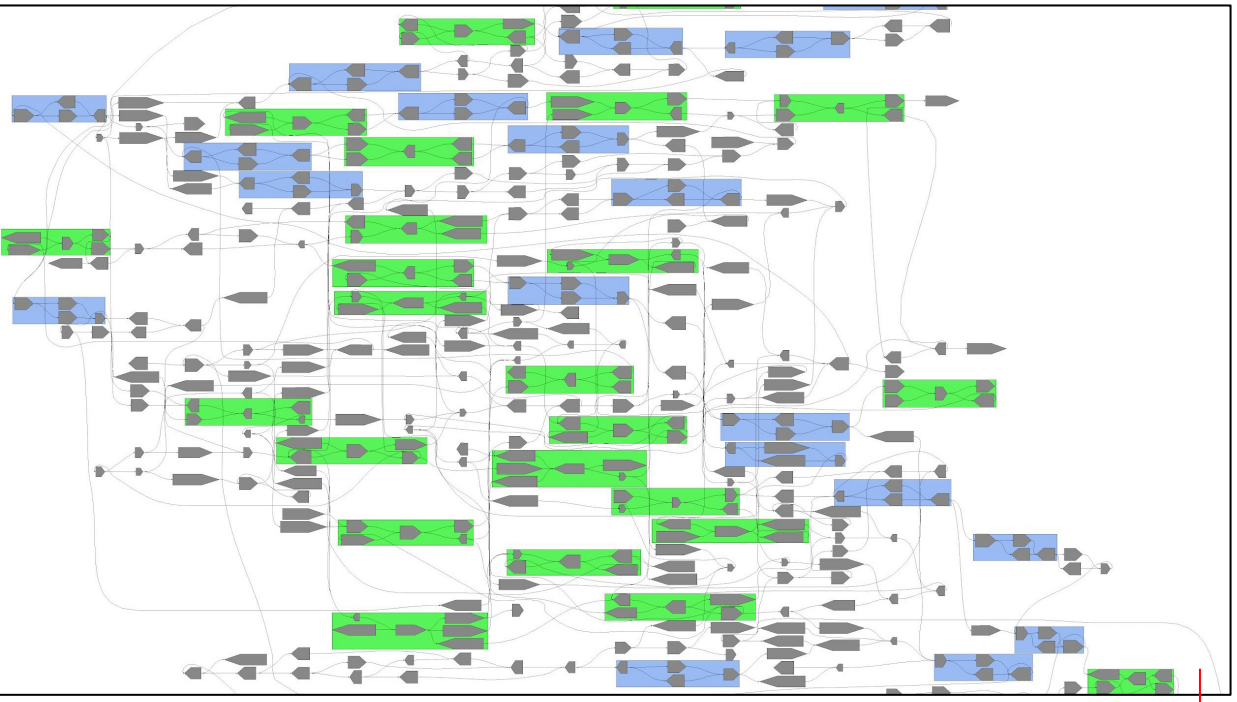


Reviewing that E. coli assembly graph (MetagenomeScope)



Lunch break

Structural Patterns and Layout Linearization



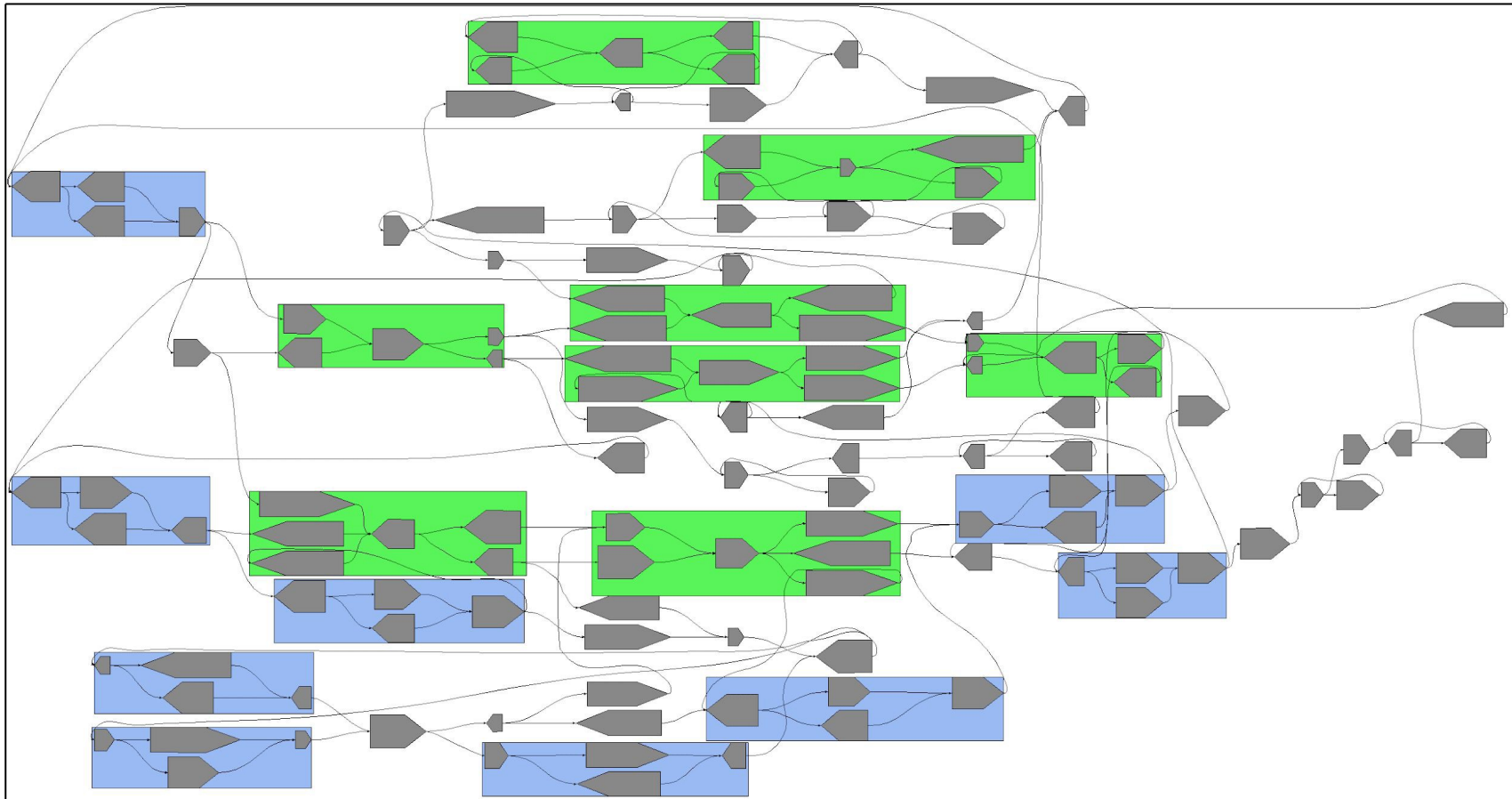
We modify our input to dot in order to linearize some graphs.

Structural Patterns and Layout Linearization

Initial Approach

Node groups (structural patterns) represented as “clusters” when laying out the graph.

Entire connected component is laid out at once.



Sample assembly graph provided with Bandage (Salmonella enterica)

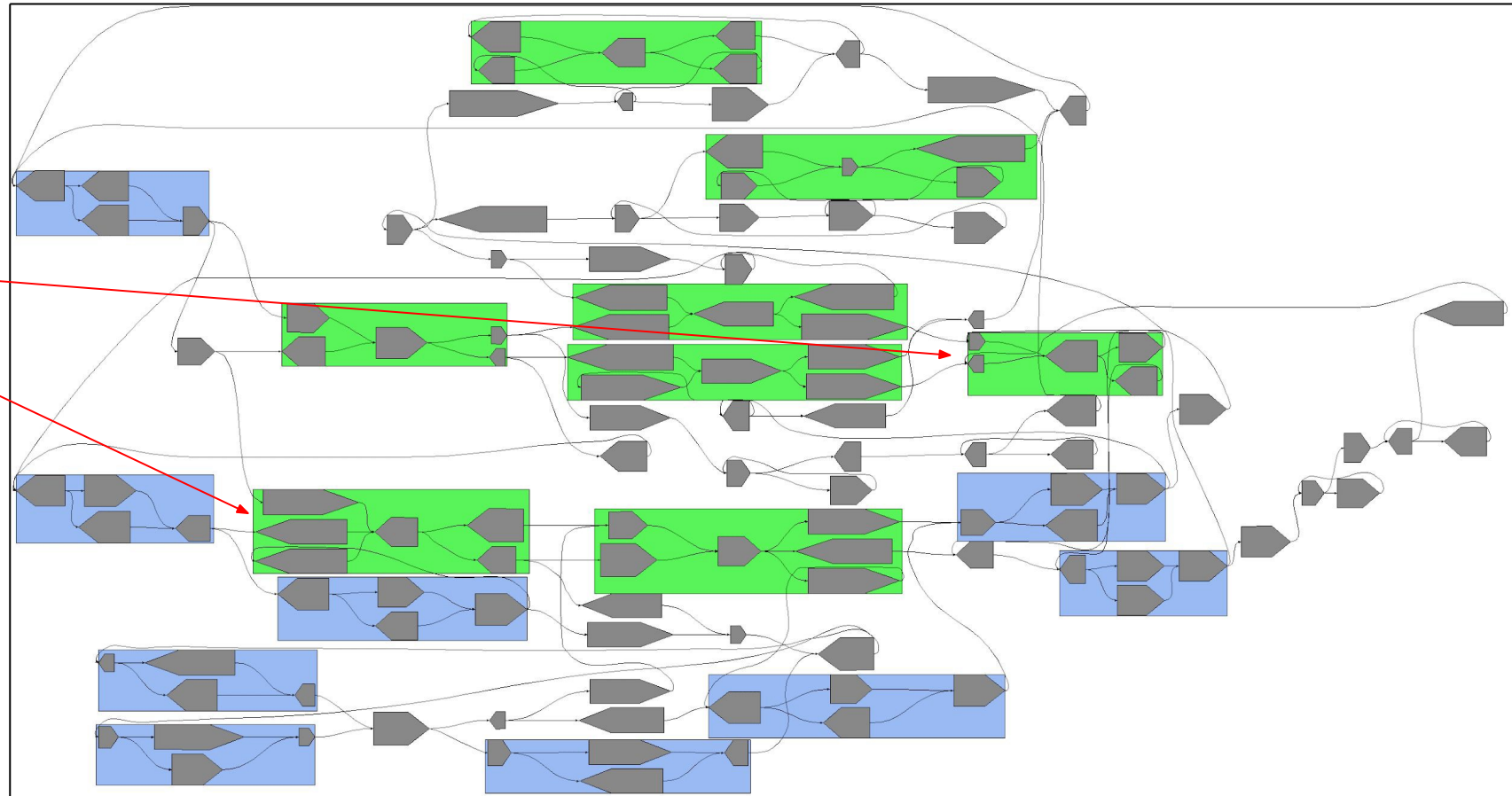
Structural Patterns and Layout Linearization

Initial Approach

Node groups (structural patterns) represented as “clusters” when laying out the graph.

Entire connected component is laid out at once.

Note that dot routes edges through clusters.



Sample assembly graph provided with Bandage (Salmonella enterica)

Structural Patterns and Layout Linearization

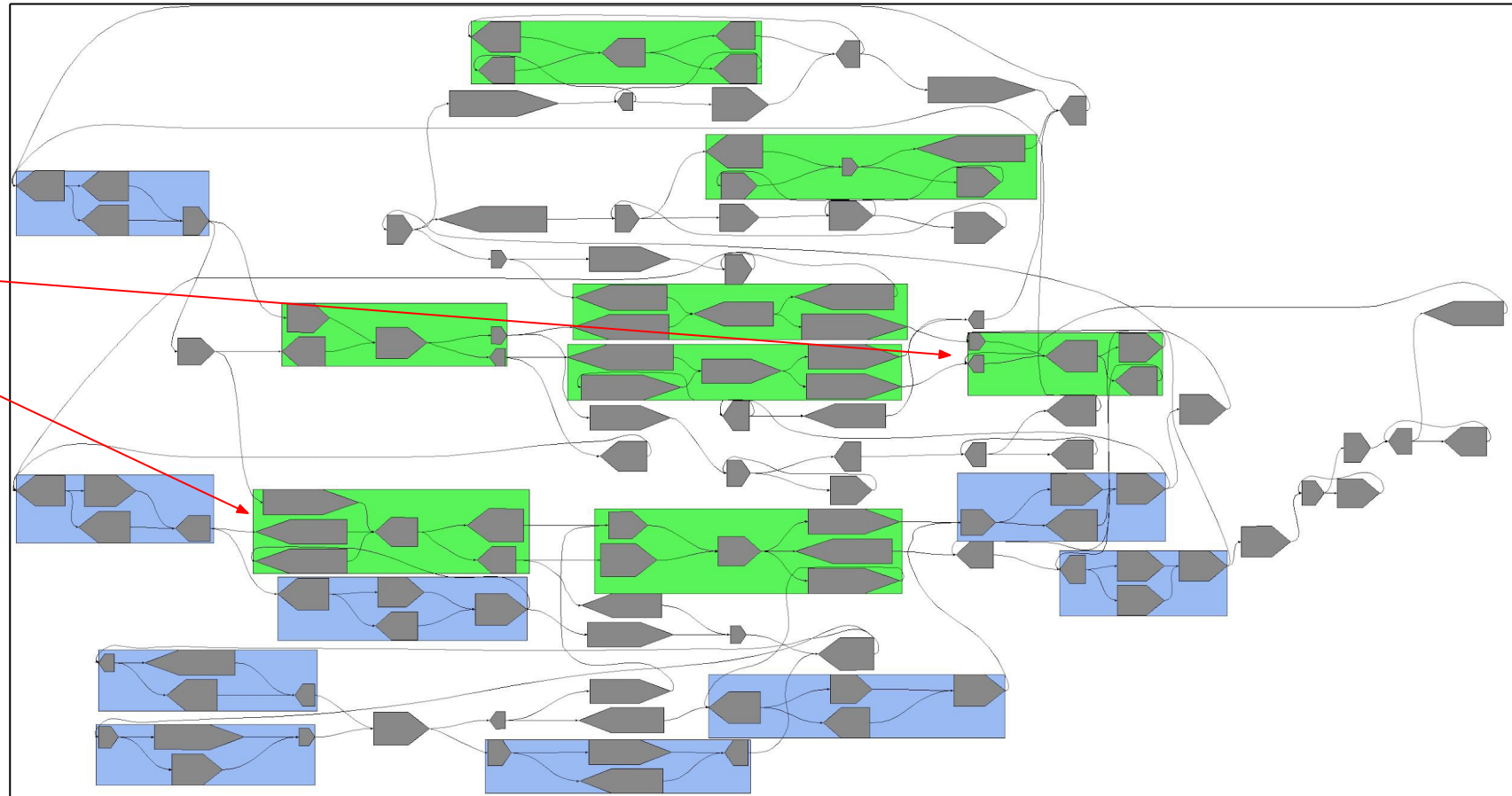
Initial Approach

Node groups (structural patterns) represented as “clusters” when laying out the graph.

Entire connected component is laid out at once.

Note that dot routes edges through clusters.

Can we avoid this?

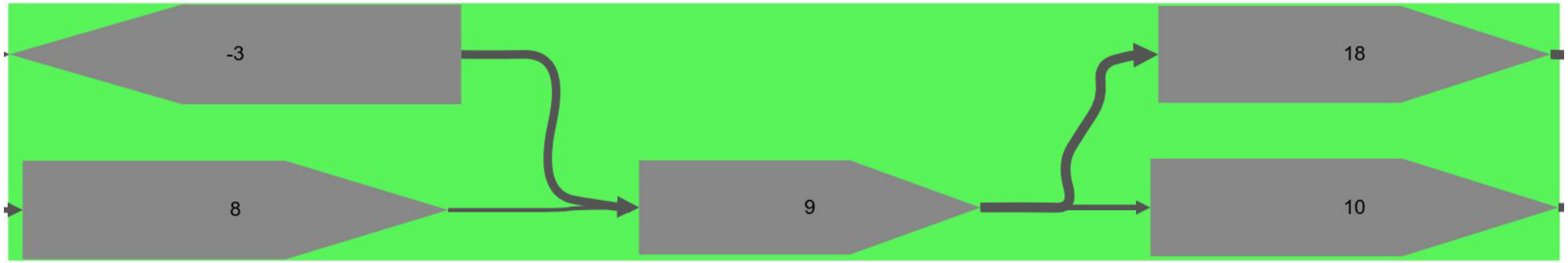


Sample assembly graph provided with Bandage (Salmonella enterica)

Structural Patterns and Layout Linearization

Revised Approach

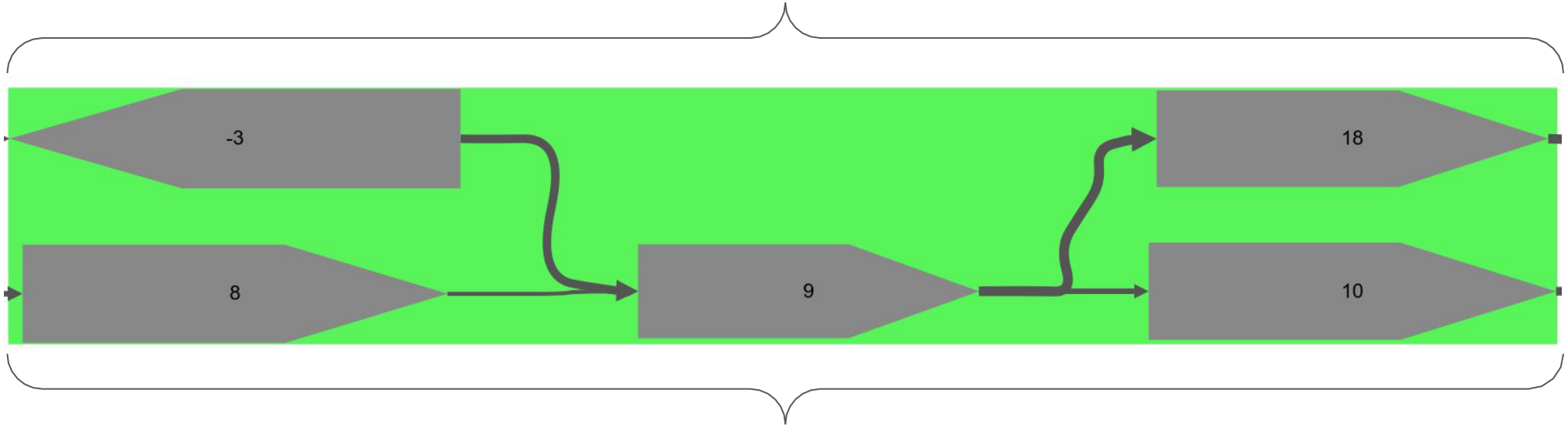
Node groups (structural patterns) laid out in isolation.
Their bounding box dimensions (and the relative positions of their child nodes and edges) are stored.



Structural Patterns and Layout Linearization

Revised Approach

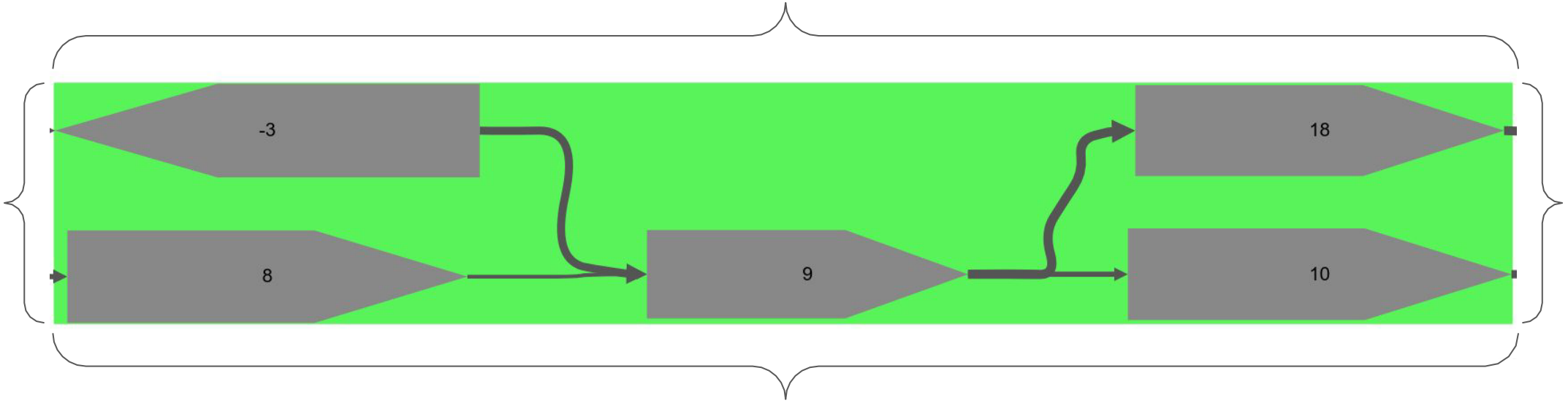
Node groups (structural patterns) laid out in isolation.
Their bounding box dimensions (and the relative positions of their child nodes and edges) are stored.



Structural Patterns and Layout Linearization

Revised Approach

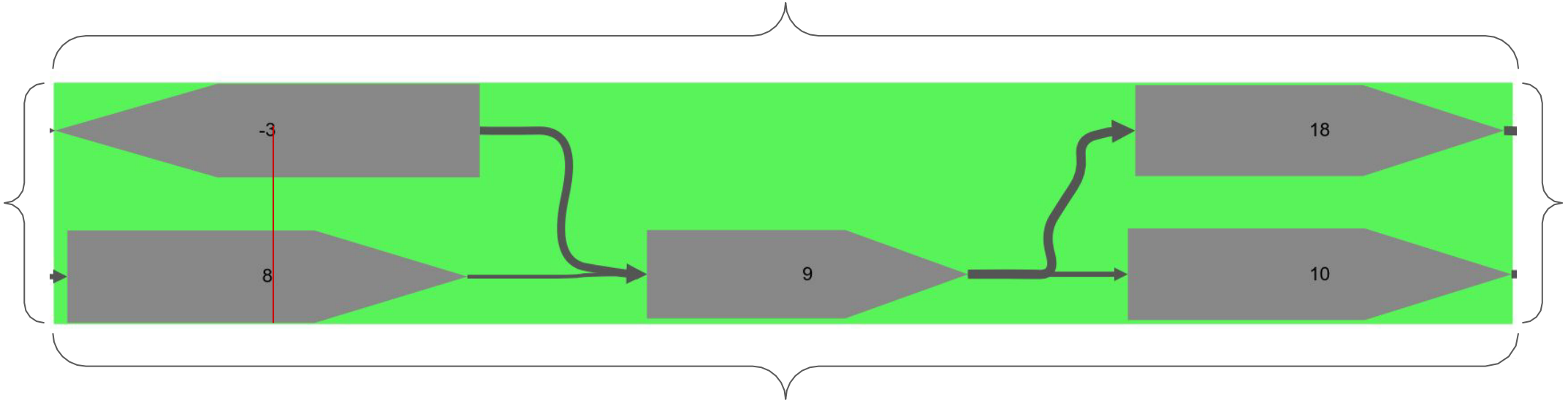
Node groups (structural patterns) laid out in isolation.
Their bounding box dimensions (and the relative positions of their child nodes and edges) are stored.



Structural Patterns and Layout Linearization

Revised Approach

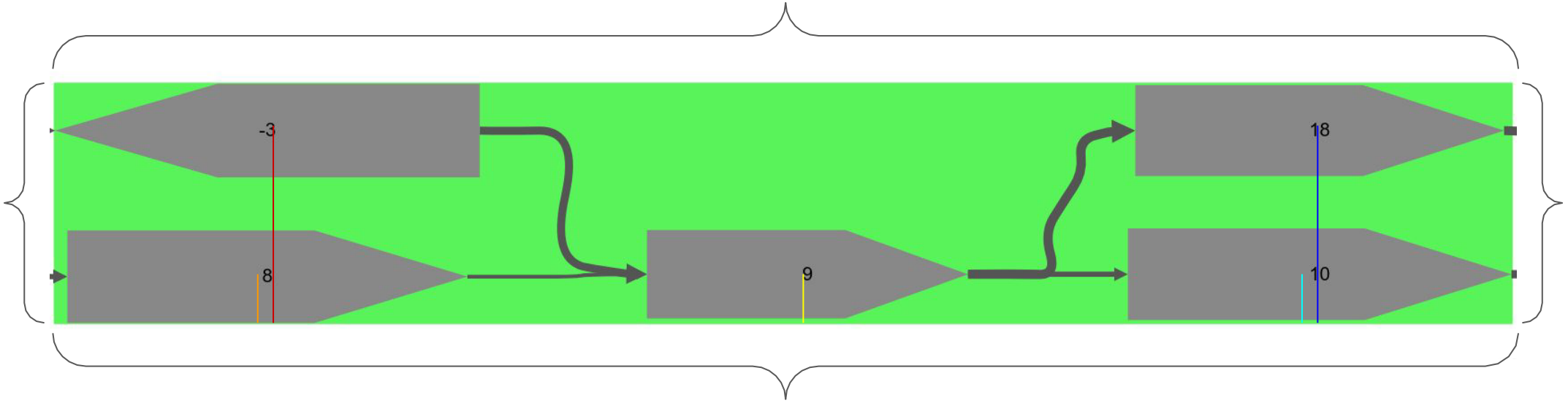
Node groups (structural patterns) laid out in isolation.
Their bounding box dimensions (and the relative positions of their child nodes and edges) are stored.



Structural Patterns and Layout Linearization

Revised Approach

Node groups (structural patterns) laid out in isolation.
Their bounding box dimensions (and the relative positions of their child nodes and edges) are stored.

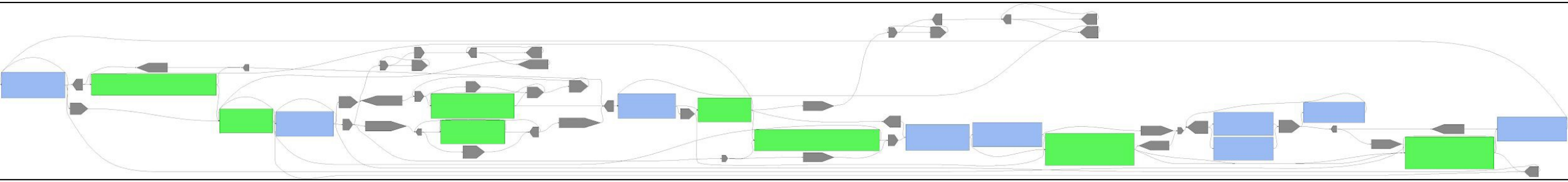


Structural Patterns and Layout Linearization

Revised Approach

Node groups (structural patterns) laid out in isolation.
Their bounding box dimensions (and the relative positions of their child nodes and edges) are stored.

Each node group is represented by a rectangular node of those dimensions during the component layout.



Sample assembly graph provided with Bandage (Salmonella enterica)

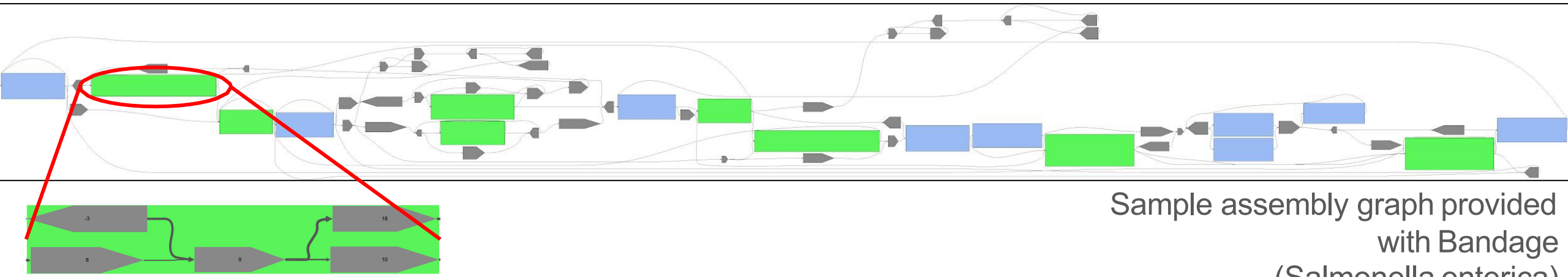
Structural Patterns and Layout Linearization

Revised Approach

Node groups (structural patterns) laid out in isolation.
Their bounding box dimensions (and the relative positions of their child nodes and edges) are stored.

Each node group is represented by a rectangular node of those dimensions during the component layout.

After the component layout process, “child” nodes and edges are backfilled into their parent node groups.



Sample assembly graph provided with Bandage (Salmonella enterica)

Structural Patterns and Layout Linearization

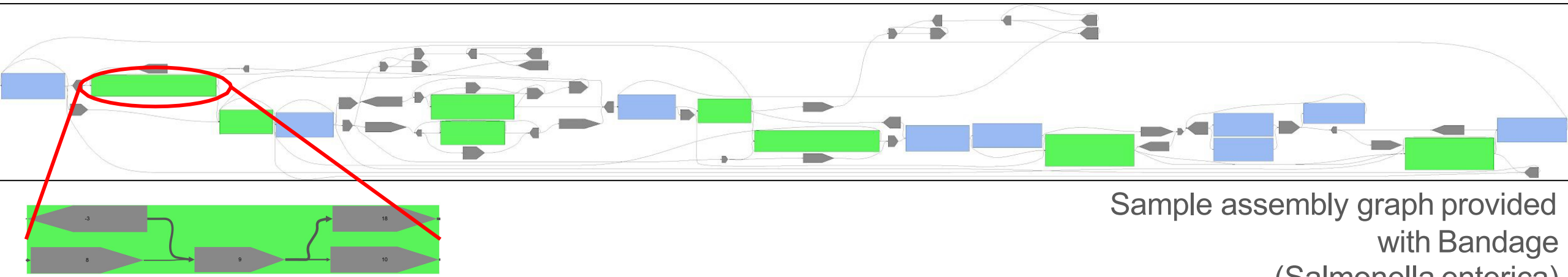
Revised Approach

Node groups (structural patterns) laid out in isolation.
Their bounding box dimensions (and the relative positions of their child nodes and edges) are stored.

Each node group is represented by a rectangular node of those dimensions during the component layout.

After the component layout process, “child” nodes and edges are backfilled into their parent node groups.

This prevents edges from being routed through node groups, linearizing the resulting layout.

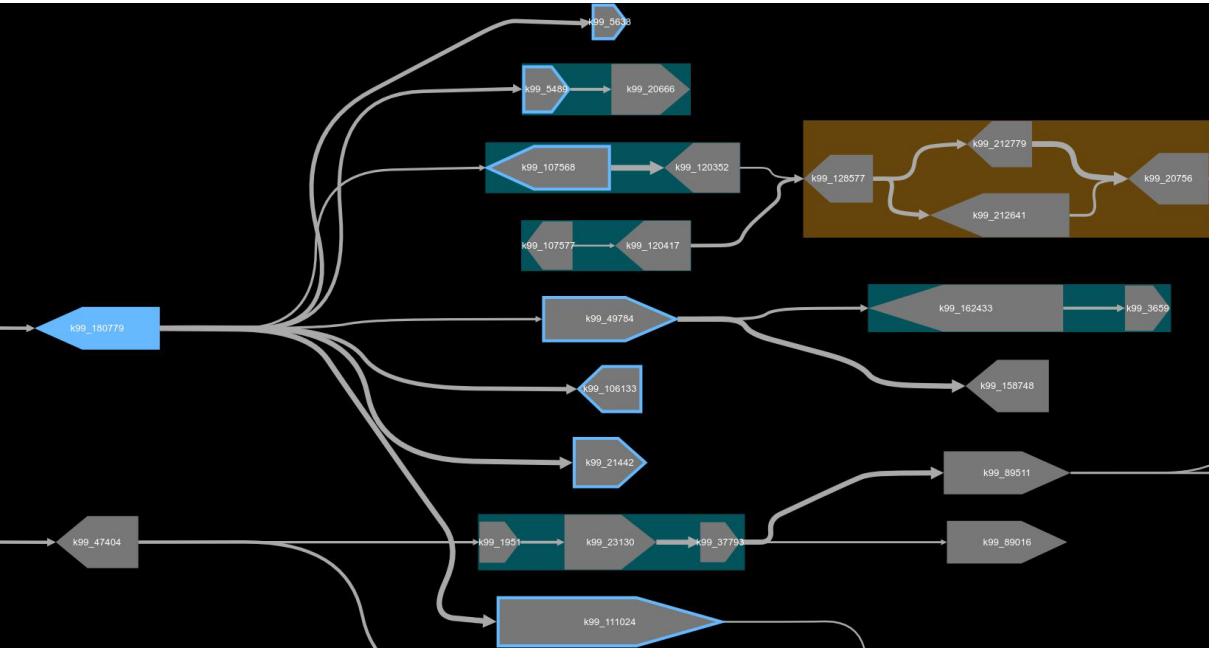


Sample assembly graph provided with Bandage (Salmonella enterica)

Path Construction and Visualization

MetagenomeScope includes tools that facilitate these tasks:

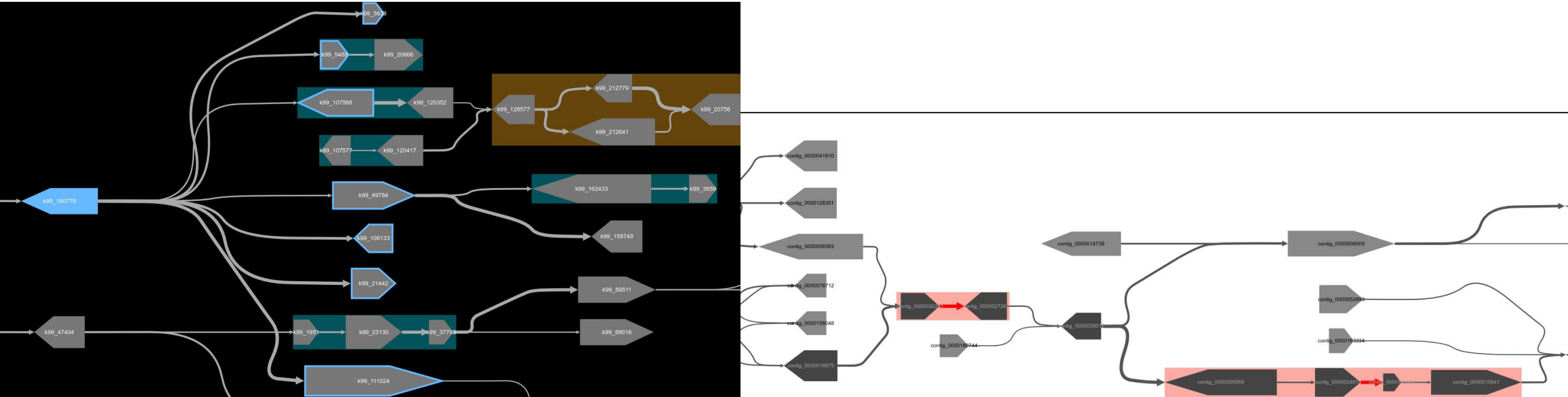
1. Select path(s) through an assembly graph



Path Construction and Visualization

MetagenomeScope includes tools that facilitate these tasks:

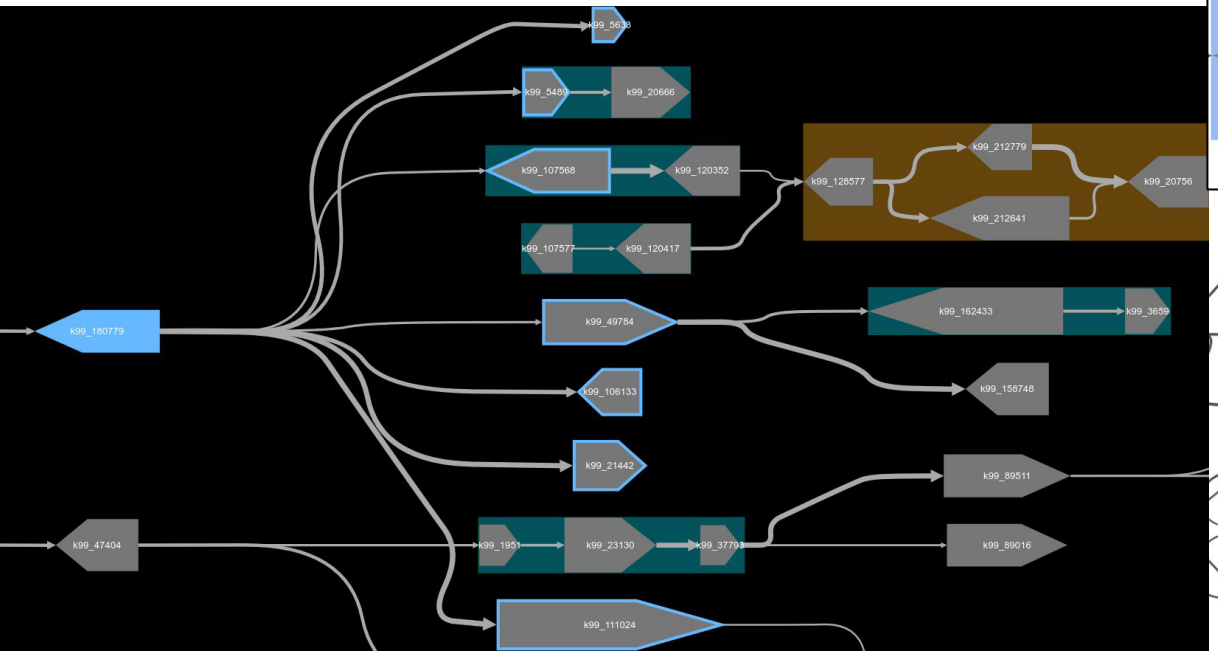
1. Select path(s) through an assembly graph
2. Visualize existing paths on top of an assembly graph

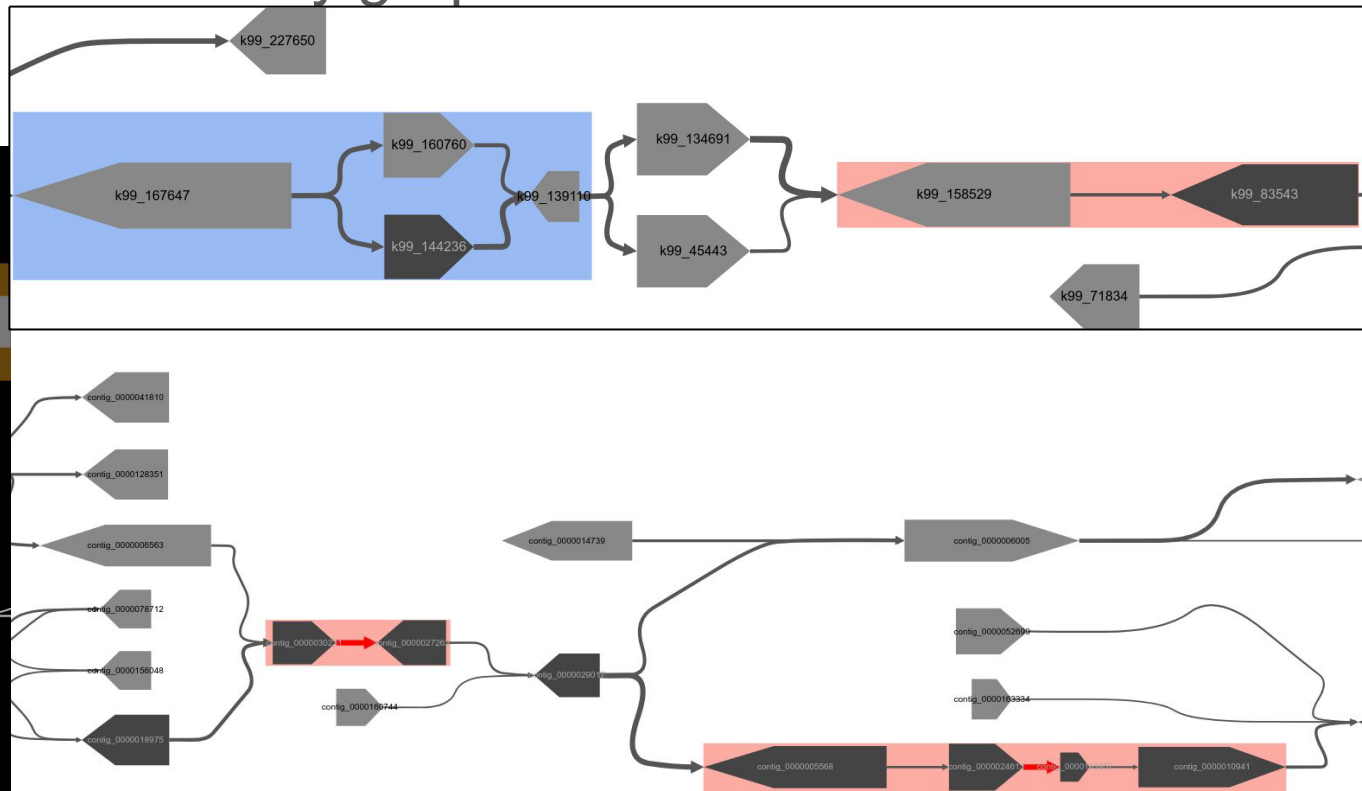


Path Construction and Visualization

MetagenomeScope includes tools that facilitate these tasks:

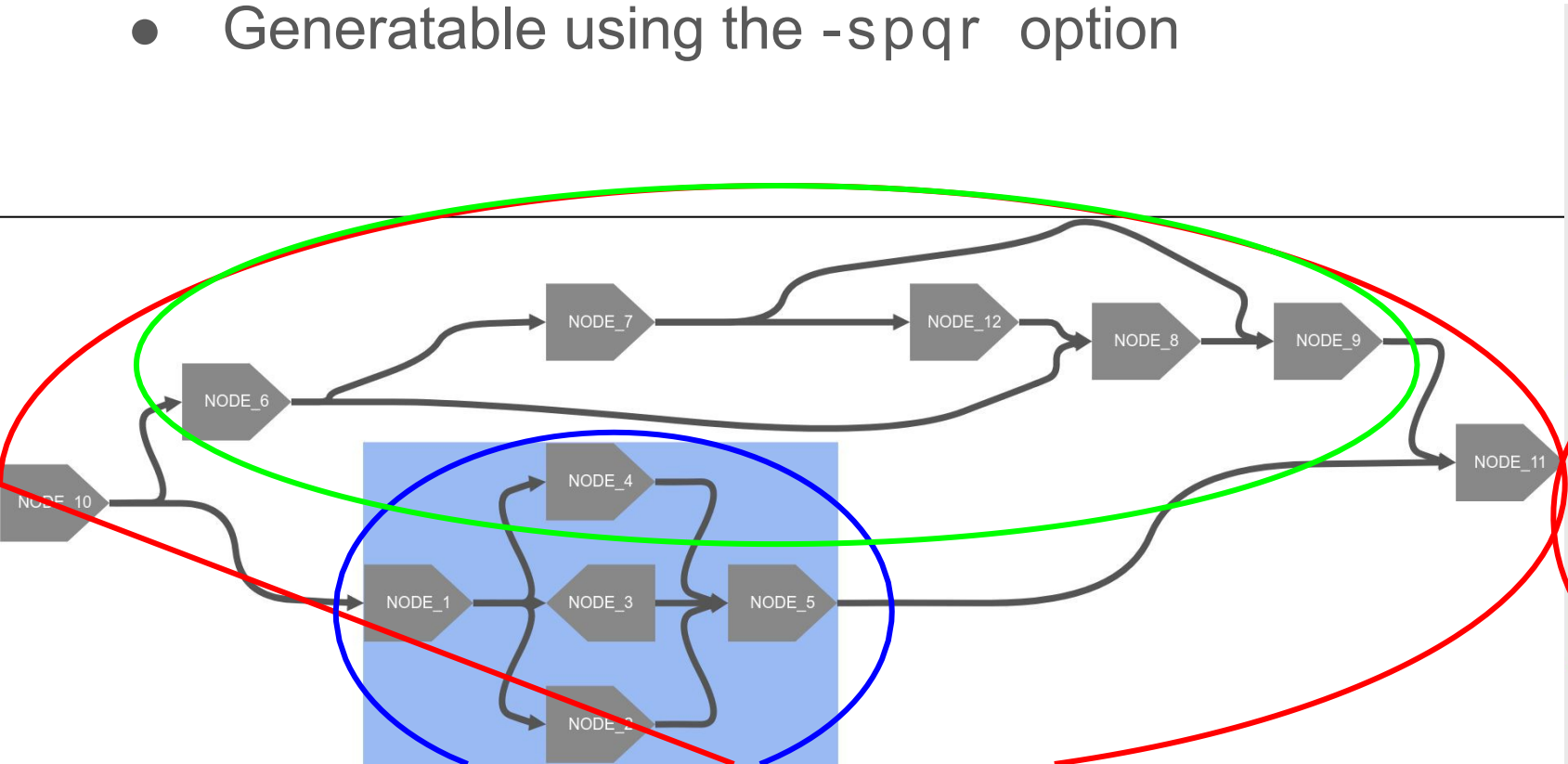
1. Select path(s) through an assembly graph
2. Visualize existing paths on top of an assembly graph
3. Identify errors in path generation

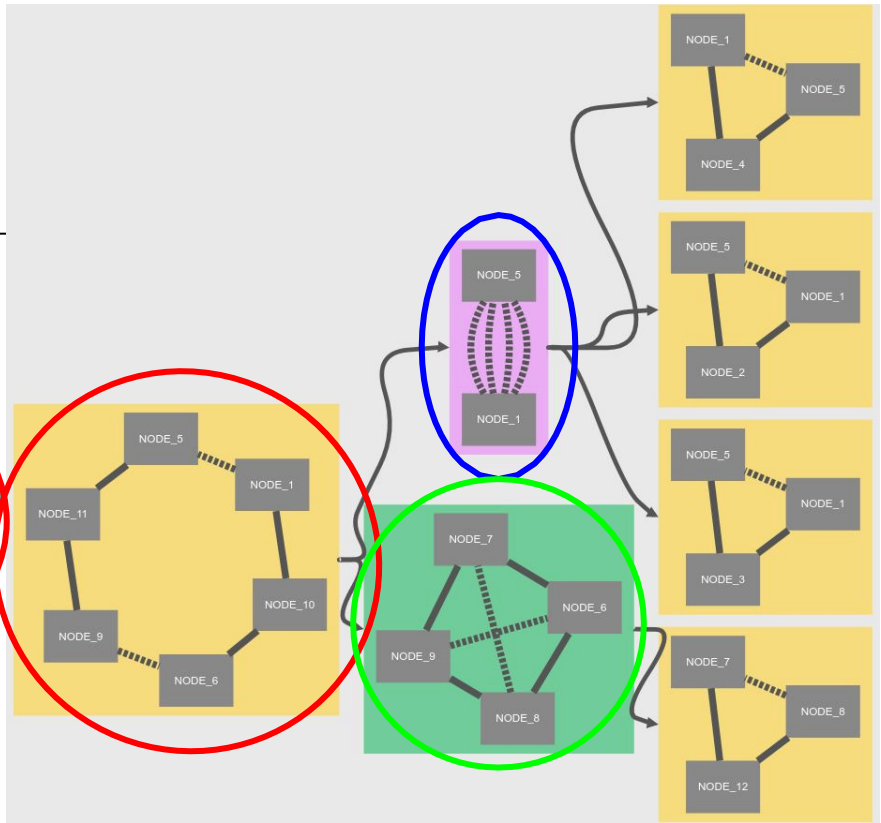




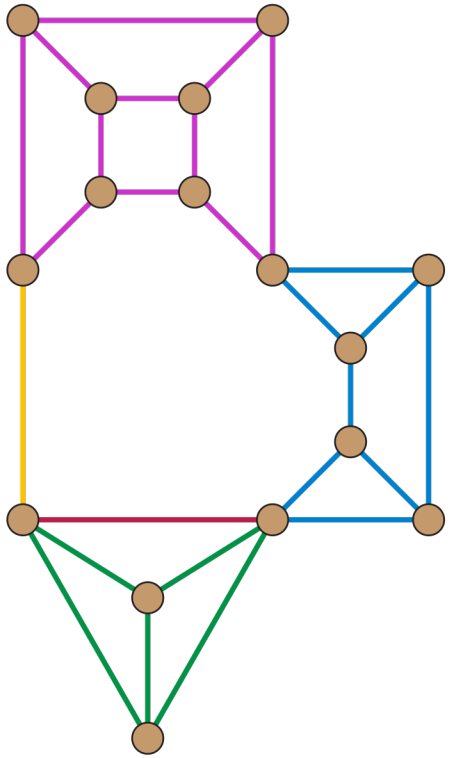
SPQR Tree Decomposition

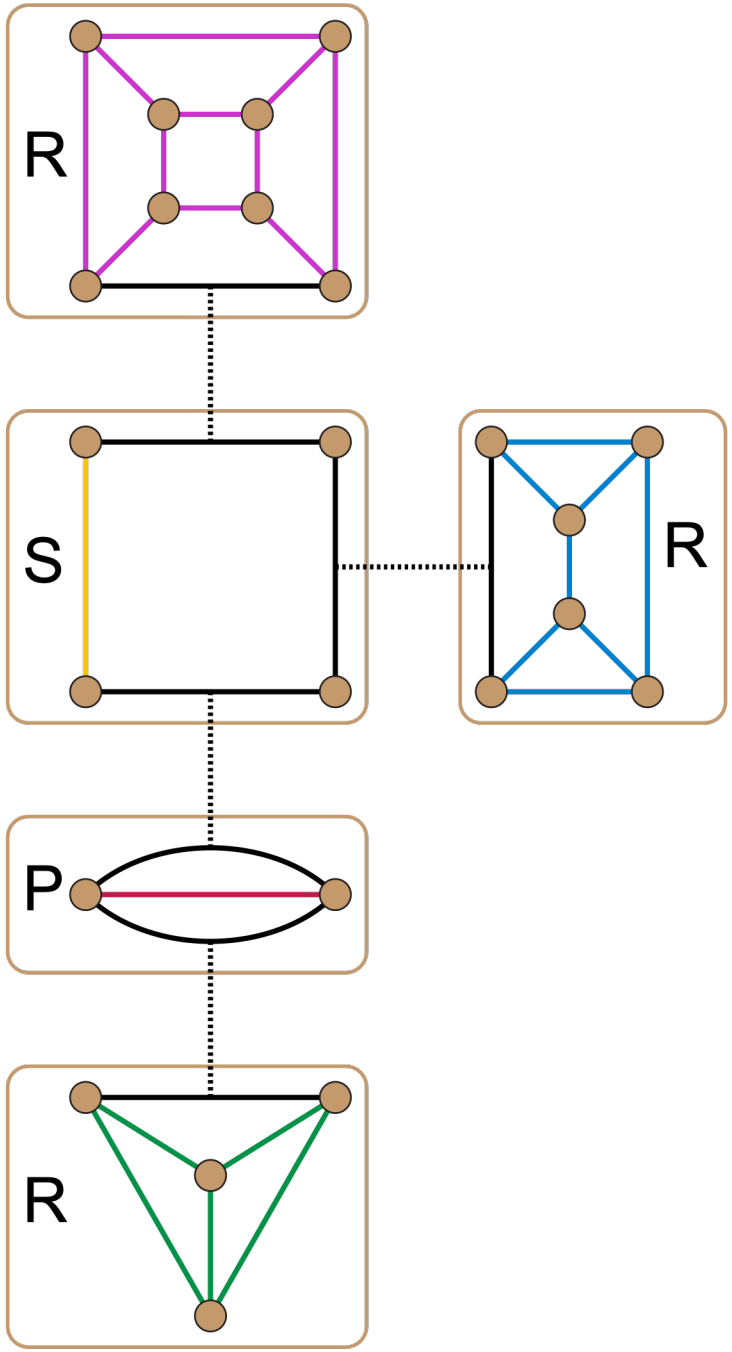
- Decomposition of biconnected graph into triconnected components
- Data structure originally used for testing graph planarity
- Link between triconnected components and bubbles
- Generatable using the `-spqr` option





- An SPQR tree takes the form of an unrooted tree in which for each node *x* there is associated an undirected graph or multigraph G_x.
- The node, and the graph associated with it, may have one of four types, given the initials SPQR

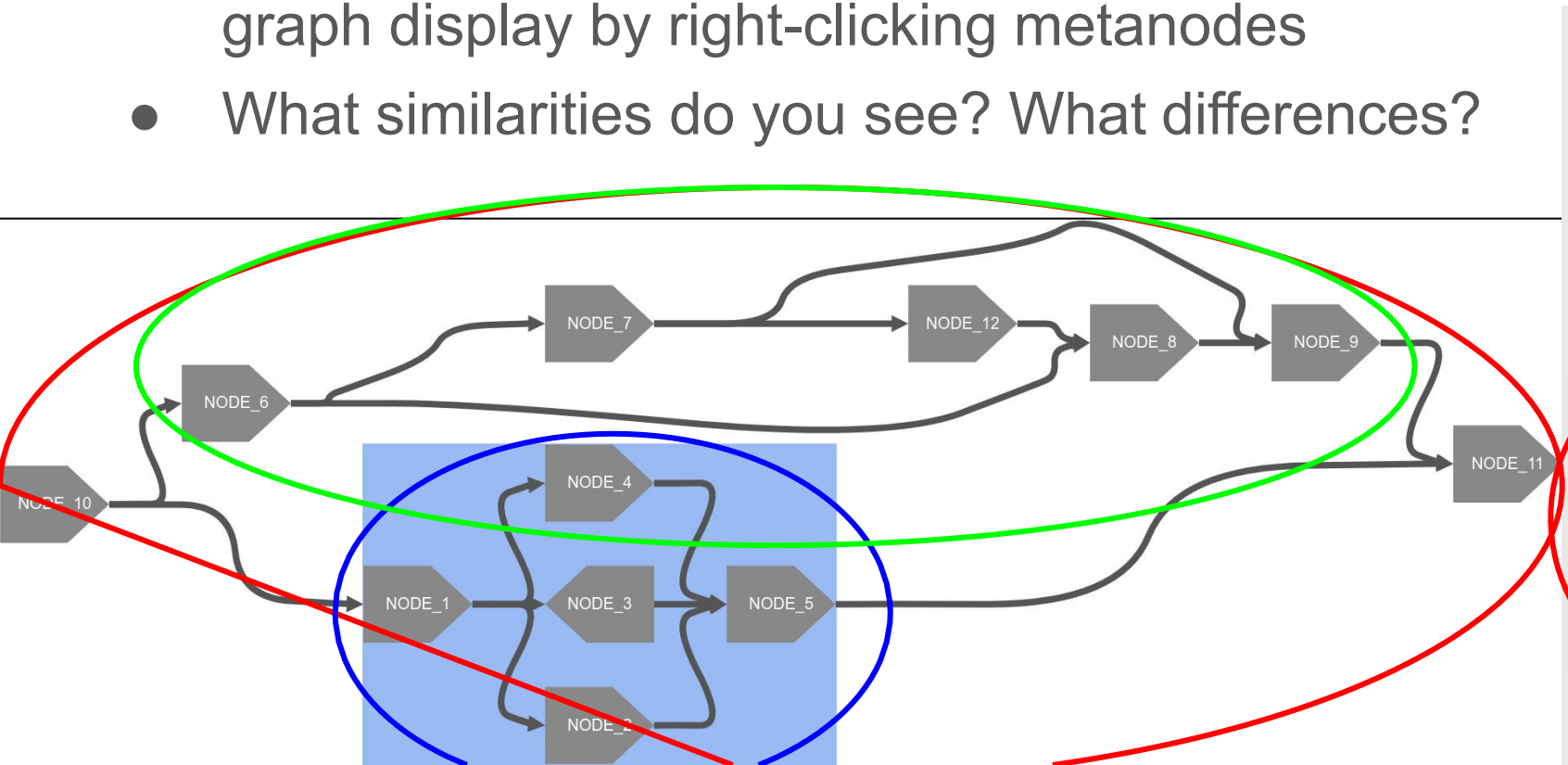


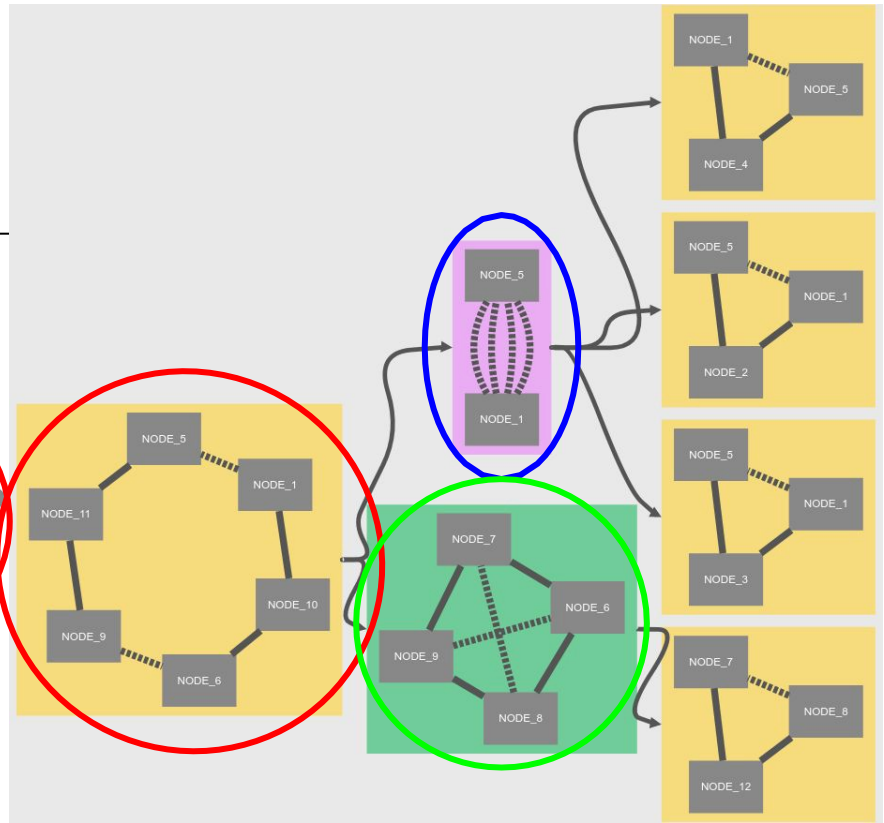


https://en.wikipedia.org/wiki/SPQR_tree

Exercise #3: Viewing an SPQR tree two ways

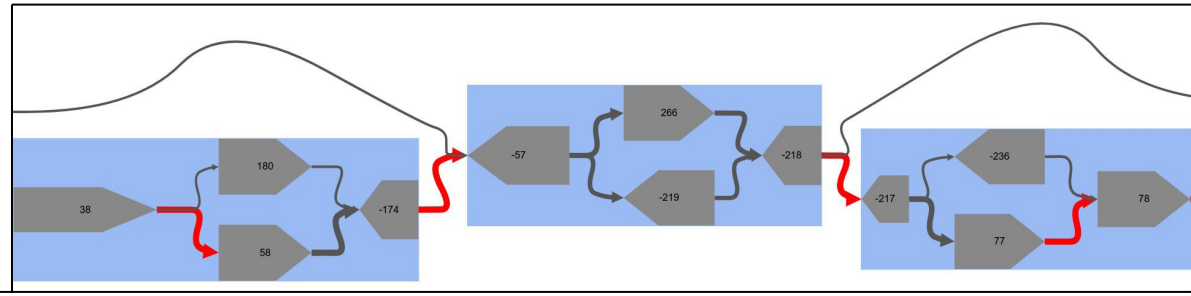
- Open the “MaryGold Paper Figure 2 graph” demo graph in MetagenomeScope’s viewer interface
- Draw the “explicit” and “implicit” SPQR tree graphs, and fully expand the graph display by right-clicking metanodes
- What similarities do you see? What differences?

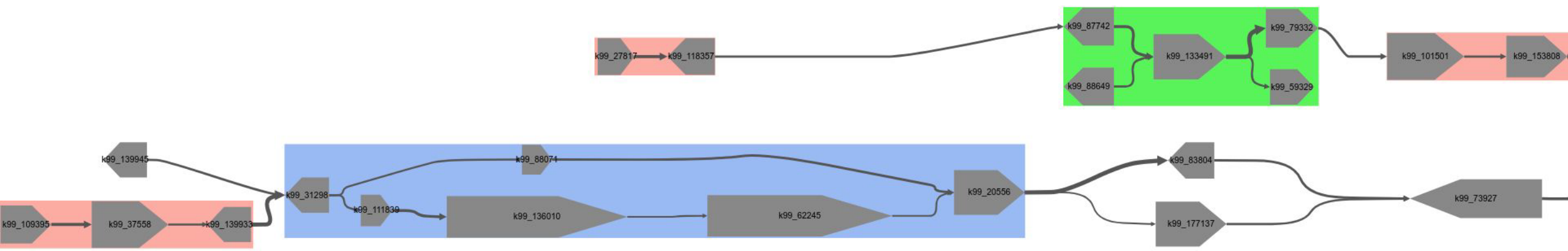




How MetagenomeScope Scales Nodes and Edges

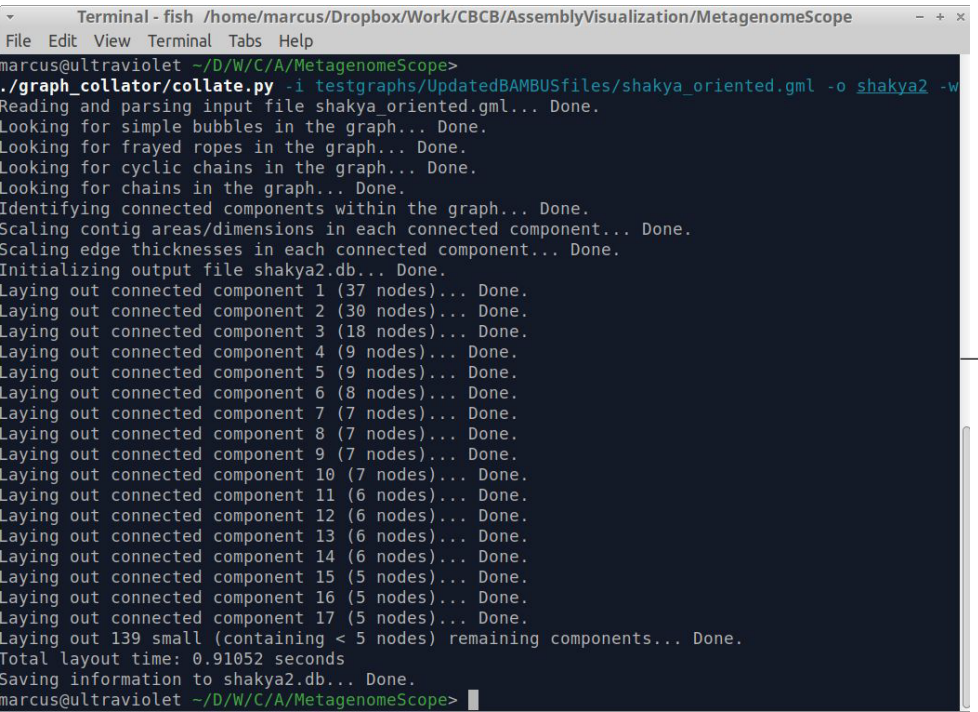
- Scaling nodes' area
  - Logarithmic → relative scaling of node area, based on length
  - Width/height proportions are altered based on percentiles: makes larger nodes look "longer"
- Scaling edges' thickness
  - Using Tukey fences to identify outlier edge weights
  - Flagging outliers in the display

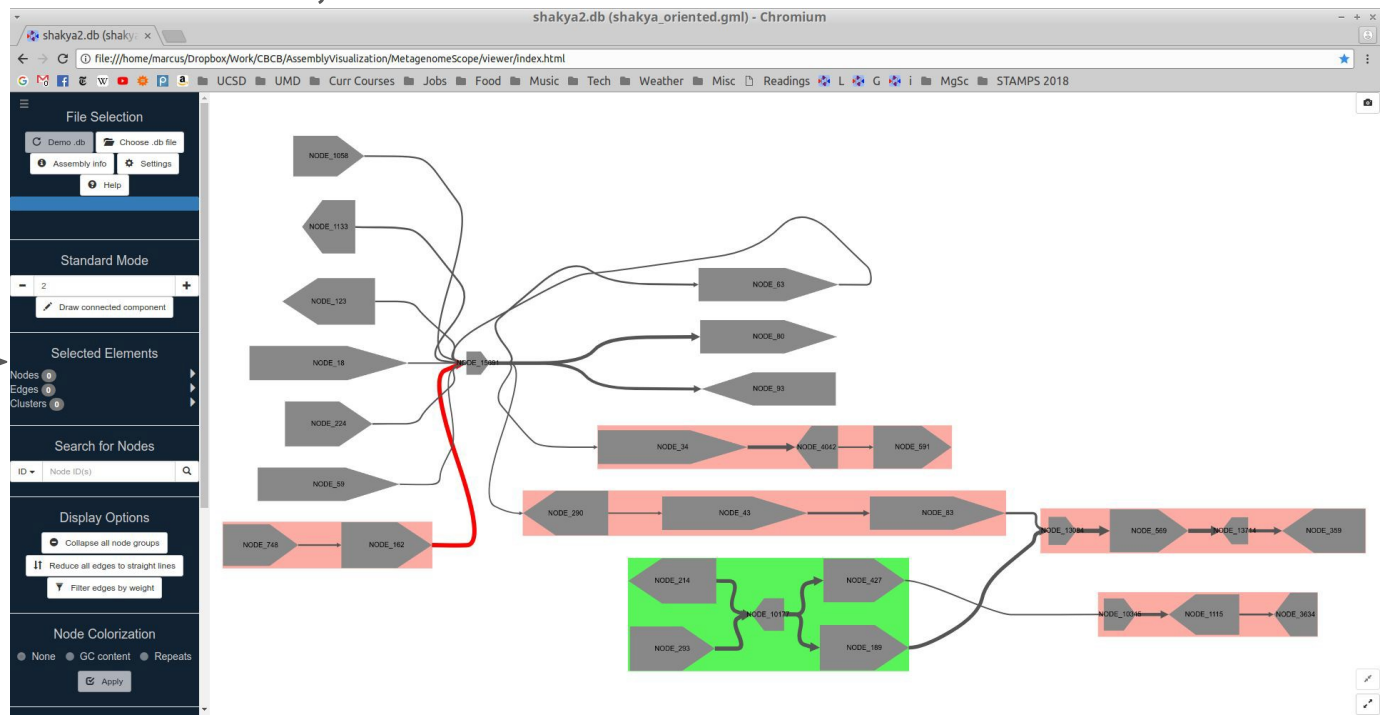




Implementation

- Command-line preprocessing script and web viewer interface
  - Script produces a SQLite3 database file, which is parsed on the client side using `sql.js`
  - Graph visualization is done on the client side (“serverlessly”) using Cytoscape.js
- Various tradeoffs in this approach (database filesize, layout time, …)
- Source code on GitHub under the GNU GPL, version 3





File Formats Accepted

- Velvet (.LastGraph)
- General (.gfa)
- MetaCarvel (.gml)
- SPAdes/MEGAHIT (.fastg) (sort of)

Exercise #4: Find an interesting pattern in the HMP (SRS049959) demo graph

- Human stool sample
- Sample processed by ATLAS in previous tutorial

Summary

- MetagenomeScope: new tool for visualizing/interacting with assembly graphs
- Uses a hierarchical layout algorithm
- Identifies structural patterns in the graph
  - Can be collapsed/uncollapsed
  - Influences the hierarchical layout process
- Supports various novel features that augment exploratory analysis of assembly graphs

Reference-guided metagenomic assembly and strain-level analyses

What is reference-guided assembly?

- An assembler with a *strong assumption* that genomes in your metagenome look a lot like those that are in a reference database.

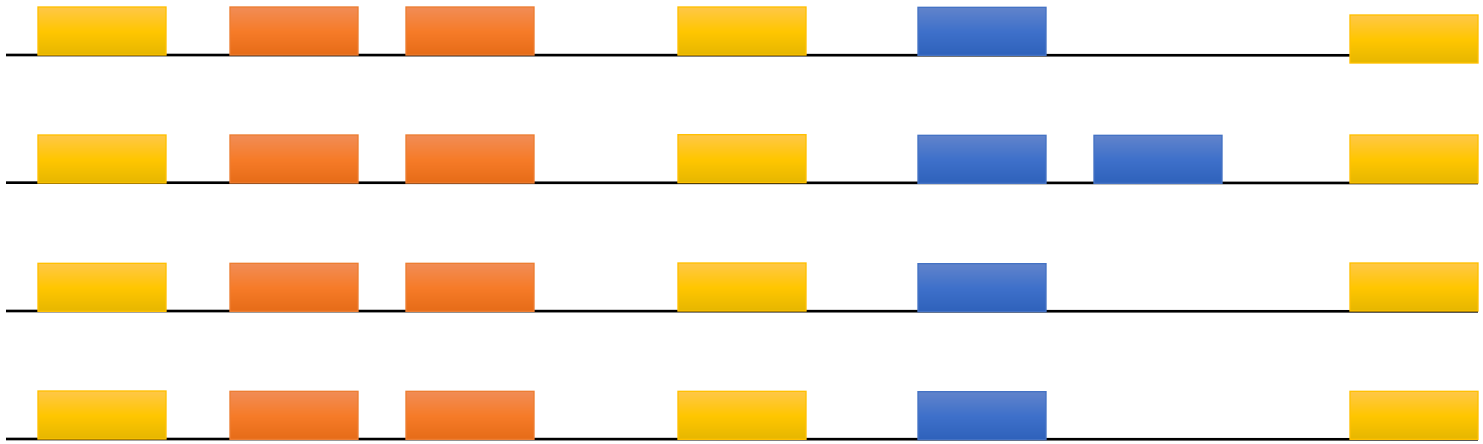
- If this is a reasonable assumption, **proceed with caution:**
  - Hint: rearrangements, horizonal gene transfer, and duplications are common!
- If this is not a reasonable assumption (viral genomes, soil samples), **think de novo assembly:**
  - Megahit
  - MetaSpades

Microbial genomes evolve over time

- *The presence of two or more homologous sequences within a single genome might reflect the acquisition of DNA sequence from a foreign source rather than the duplication of a resident gene.*

- Thus, since we do not know the origin a priori, we refer to these potential paralogs or xenologs as **synologs** (Lerat et al 2005).

Ubiquitous sequence fragments

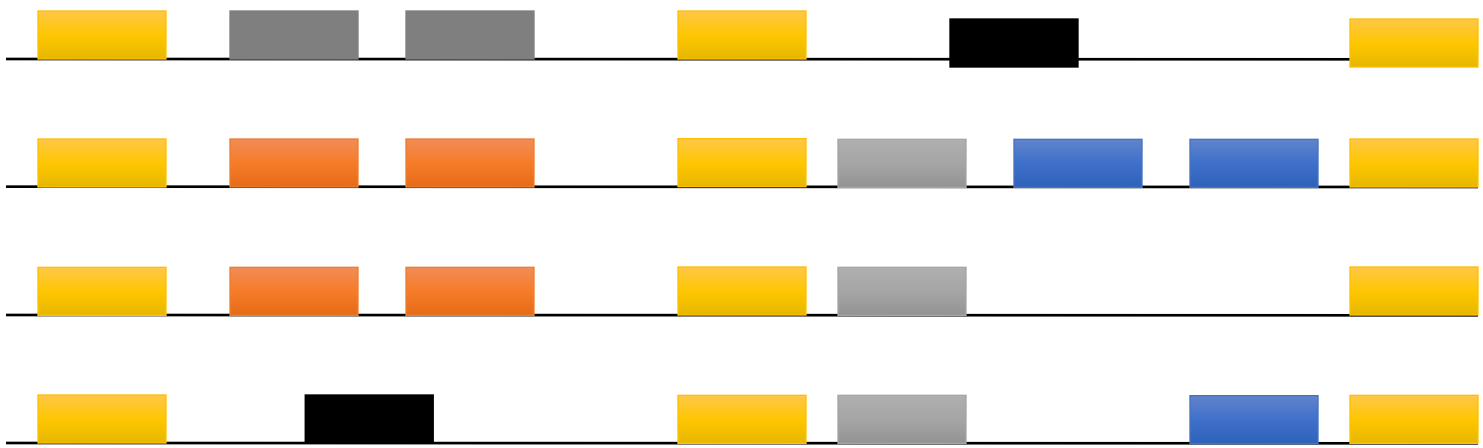


= ubiquitous with synologs (constant)

= core genome

= ubiquitous with synologs (variable)

Non-Ubiquitous sequence fragments



= core genome

= genome specific

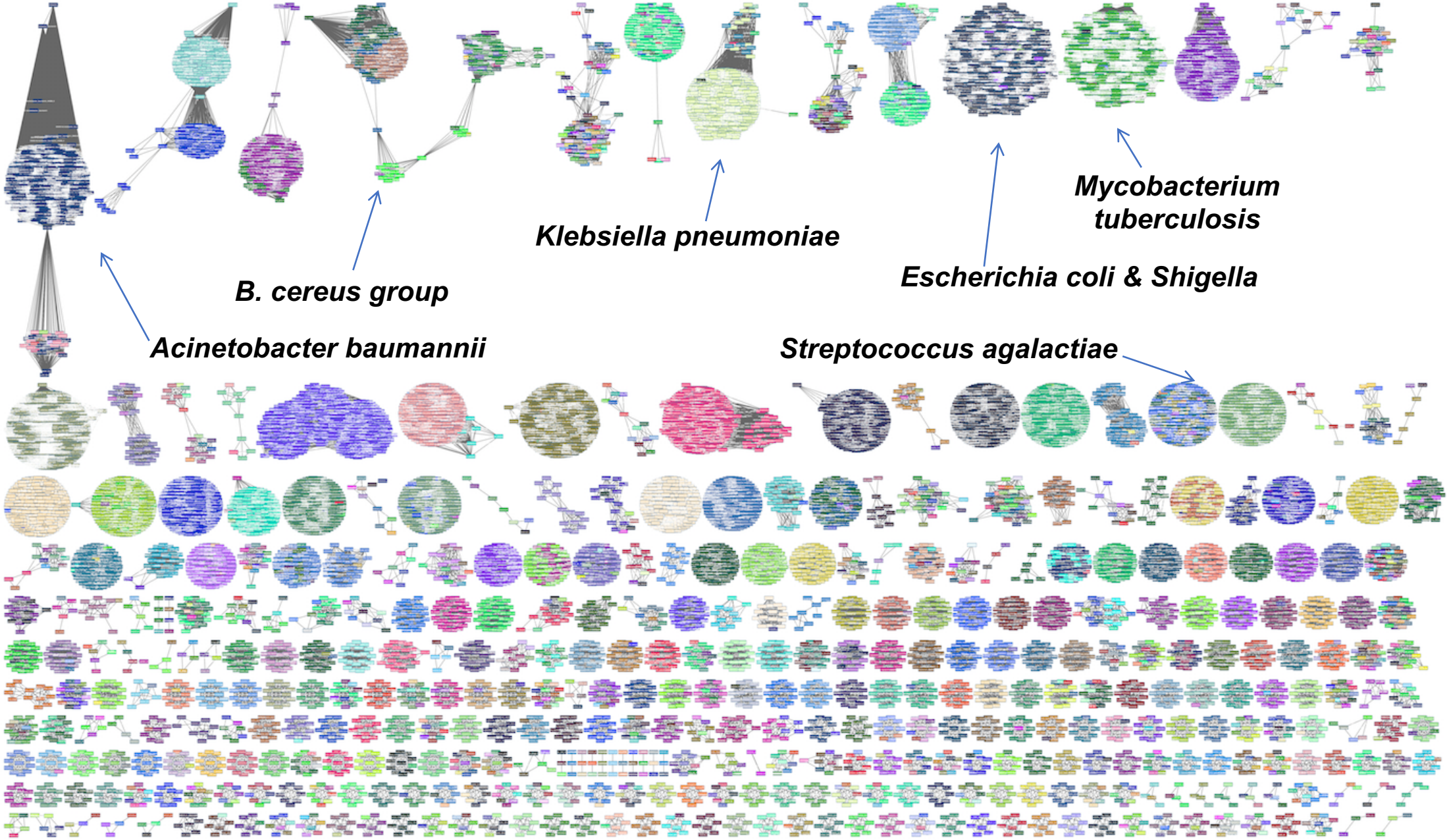
= singletons

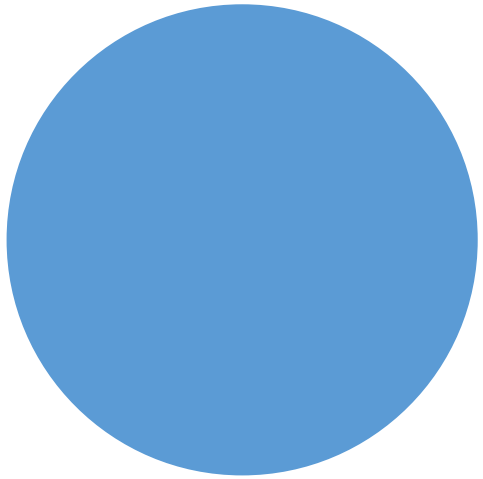
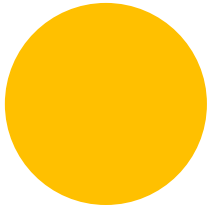
= non-ubiquitous with synologs (constant)

= non-ubiquitous with synologs (variable)

= non-ubiquitous without synologs

Powdered water -- to drink, just add water.


Mycobacterium tuberculosis
Klebsiella pneumoniae
Escherichia coli & Shigella
B. cereus group
Acinetobacter baumannii
Streptococcus agalactiae

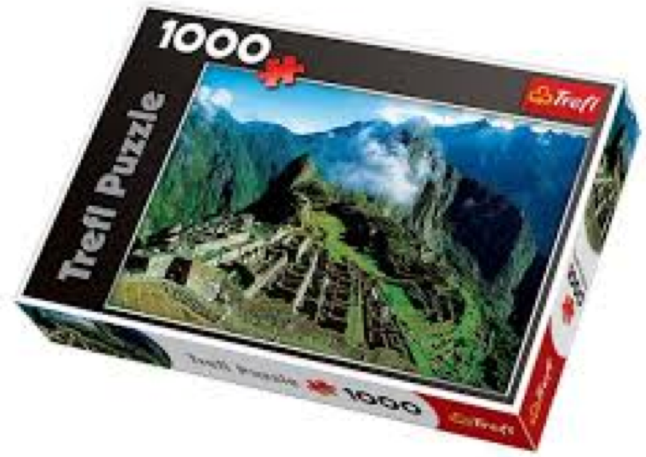



Reference-guided metagenomic assembly

Ready, set, go!

Reference-guided genome assembly

- Reconstructing the original DNA sequence aligning reads to a genome.
- Intuitively like a puzzle
- But we have the box!



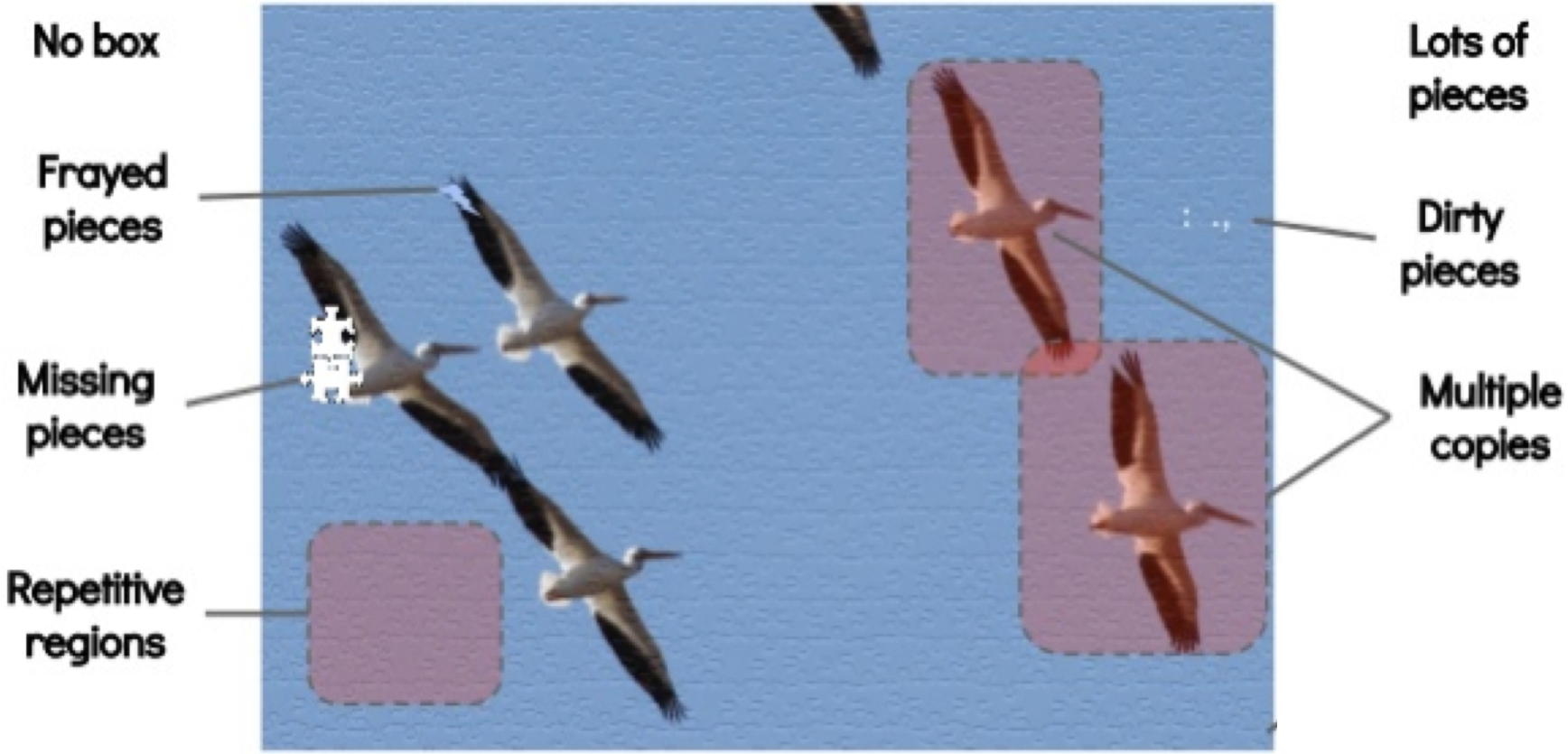
Reference-guided metagenome assembly

- Reconstructing original DNA sequences aligning reads to a set of genomes.
- Intuitively like multiple puzzles
- But we need to find the boxes!

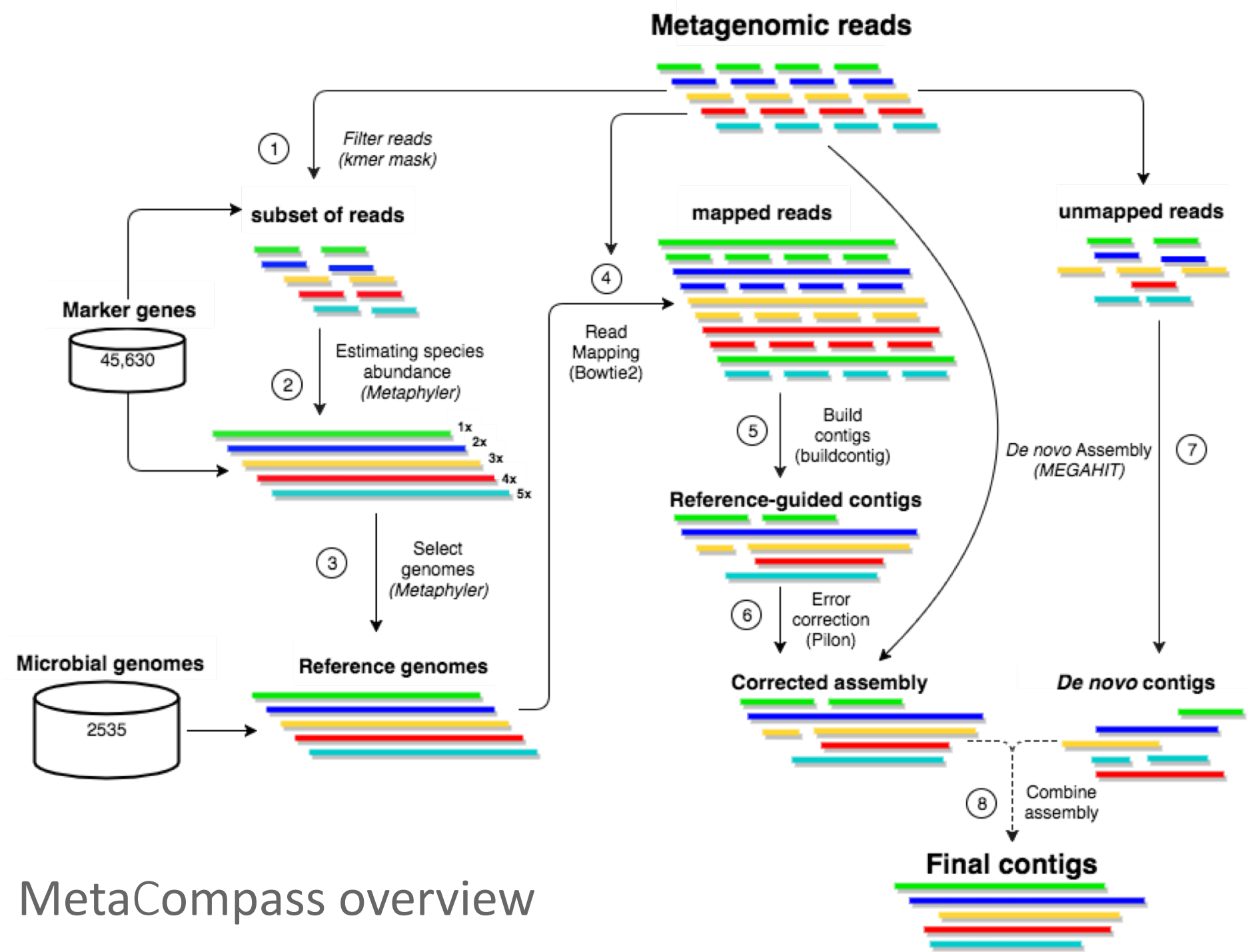
Reference-guided metagenome assembly

- Step 1: Find the puzzle boxes (reference selection)
- Step 2: Bin pieces into the right boxes (read mapping)
- Step 3: Solve each puzzle (assembly)

What makes a puzzle hard?

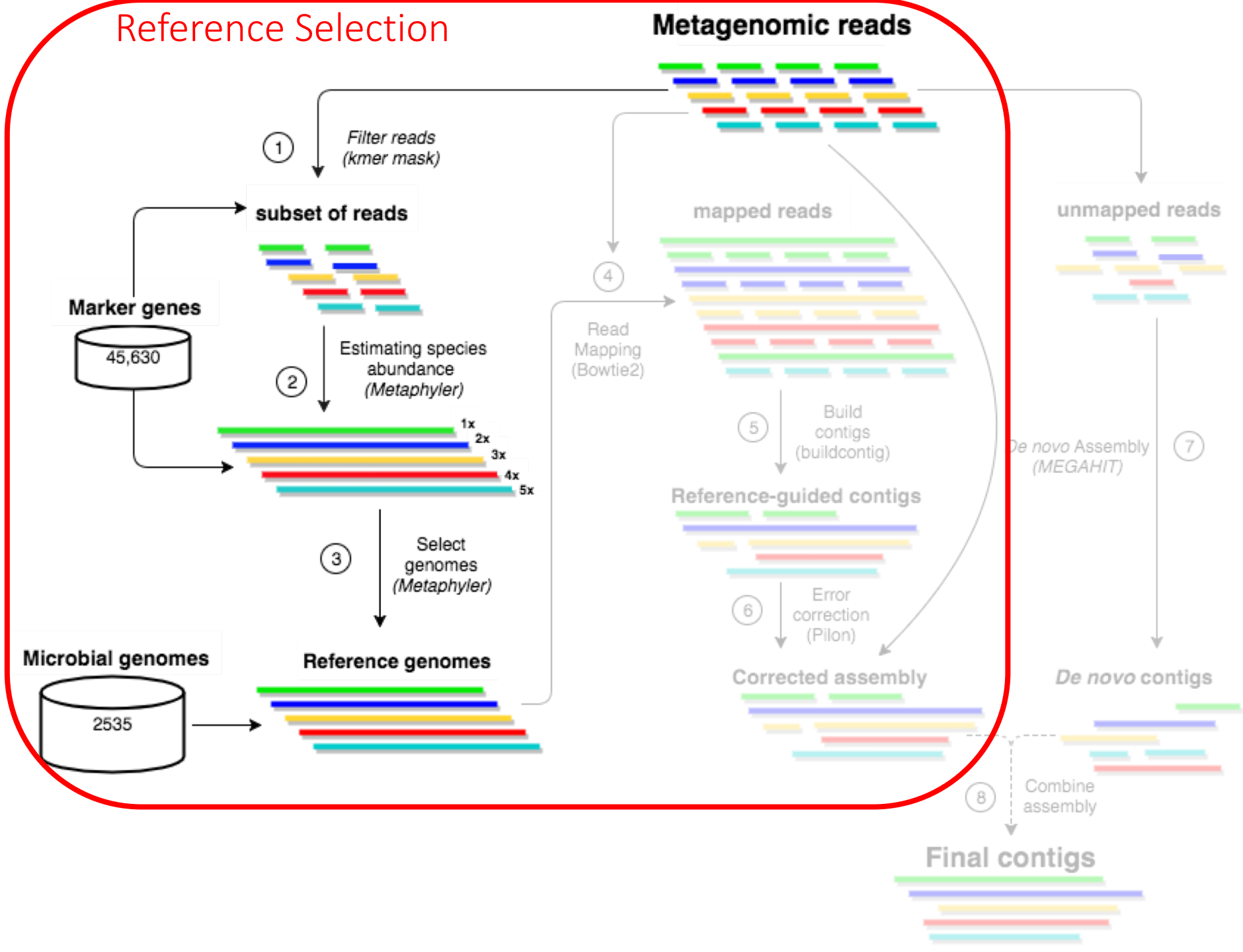


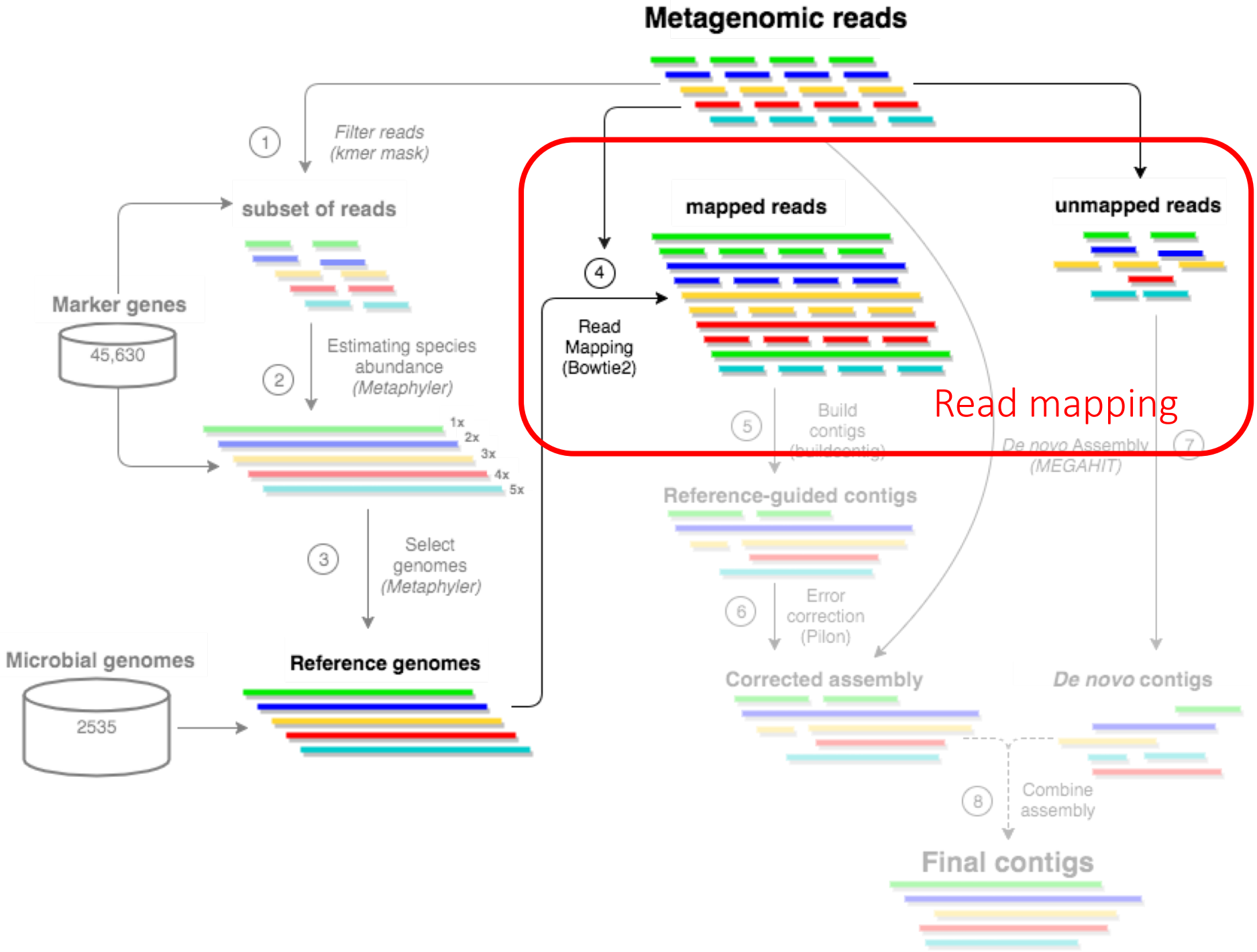
Slide credit → Torsten Seemann

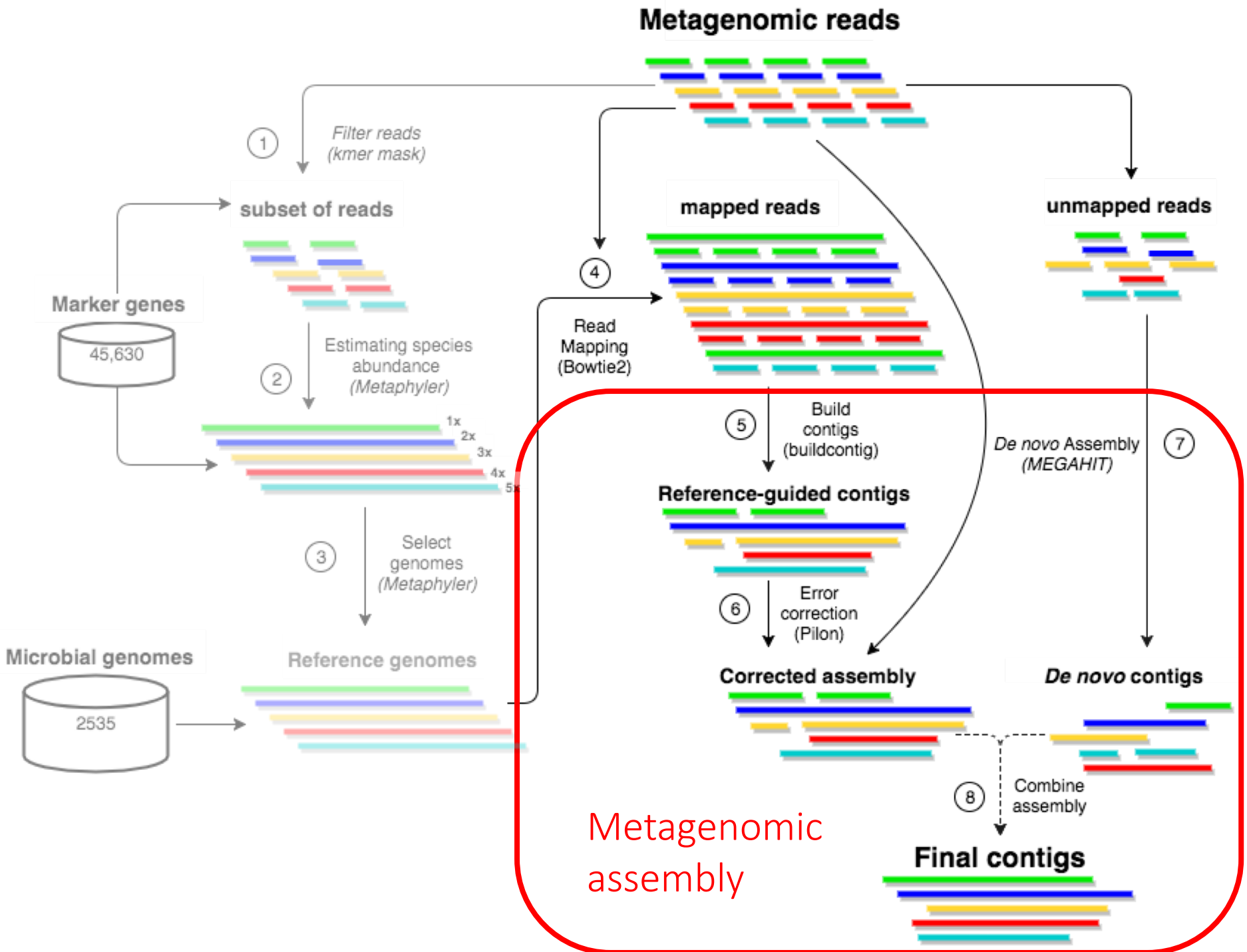


MetaCompass overview

Reference Selection




Metagenomic reads
1
Filter reads
(kmer mask)
subset of reads
Marker genes
45,630
2
Estimating species
abundance
(Metaphyler)
1x
2x
3x
4x
5x
3
Select
genomes
(Metaphyler)
Microbial genomes
2535
Reference genomes
4
Read
Mapping
(Bowtie2)
mapped reads
unmapped reads
Read mapping
5
Build
contigs
(buildcontig)
Reference-guided contigs
6
Error
correction
(Pilon)
Corrected assembly
De novo Assembly
(MEGAHIT)
7
De novo contigs
8
Combine
assembly
Final contigs


Metagenomic reads
1
Filter reads
(kmer mask)
subset of reads
Marker genes
45,630
2
Estimating species
abundance
(Metaphyler)
1x
2x
3x
4x
5x
3
Select
genomes
(Metaphyler)
Microbial genomes
2535
Reference genomes
4
Read
Mapping
(Bowtie2)
mapped reads
unmapped reads
5
Build
contigs
(buildcontig)
Reference-guided contigs
6
Error
correction
(Pilon)
Corrected assembly
De novo Assembly
(MEGAHIT)
7
De novo contigs
8
Combine
assembly
Final contigs
Metagenomic
assembly

Choose your own adventure: how shall we identify the reference genomes?

1. Universal marker gene- based approaches (MetaPhlan, MetaPhyler, etc)

2. MinHash based approaches (Mash, SourMash, etc)

3. Kmer + LCA based approaches (Clark, Kraken, etc)

- http://tiny.cc/stamps19

STEP 1: Reference selection

Liu *et al. BMC Genomics* 2011, **12**(Suppl 2):S4
http://www.biomedcentral.com/1471-2164/12/S2/S4

PROCEEDINGS Open Access

Accurate and fast estimation of taxonomic profiles from metagenomic shotgun sequences

Bo Liu[1,2], Theodore Gibbons[1,3], Mohammad Ghodsi[1,2], Todd Treangen[1], Mihai Pop[1,2,3*]

From IEEE International Conference on Bioinformatics and Biomedicine 2010
Hong Kong, P. R. China. 18-21 December 2010

STEP 1: Reference selection

Ondov *et al. Genome Biology* (2016) 17:132
DOI 10.1186/s13059-016-0997-x

Genome Biology

SOFTWARE Open Access

Mash: fast genome and metagenome distance estimation using MinHash

Brian D. Ondov[1], Todd J. Treangen[1], Páll Melsted[2], Adam B. Mallonee[1], Nicholas H. Bergman[1], Sergey Koren[3] and Adam M. Phillippy[3*]

Abstract

Mash extends the MinHash dimensionality-reduction technique to include a pairwise mutation distance and *P* value significance test, enabling the efficient clustering and search of massive sequence collections. Mash reduces large sequences and sequence sets to small, representative sketches, from which global mutation distances can be rapidly estimated. We demonstrate several use cases, including the clustering of all 54,118 NCBI RefSeq genomes in 33 CPU h; real-time database search using assembled or unassembled Illumina, Pacific Biosciences, and Oxford Nanopore data; and the scalable clustering of hundreds of metagenomic samples by composition. Mash is freely released under a BSD license (https://github.com/marbl/mash).

Keywords: Comparative genomics, Genomic distance, Alignment, Sequencing, Nanopore, Metagenomics

STEP 1: Reference selection

Wood and Salzberg *Genome Biology* 2014, **15**:R46
http://genomebiology.com/2014/15/3/R46

Genome Biology

METHOD Open Access

Kraken: ultrafast metagenomic sequence classification using exact alignments

Derrick E Wood[1,2*] and Steven L Salzberg[2,3]

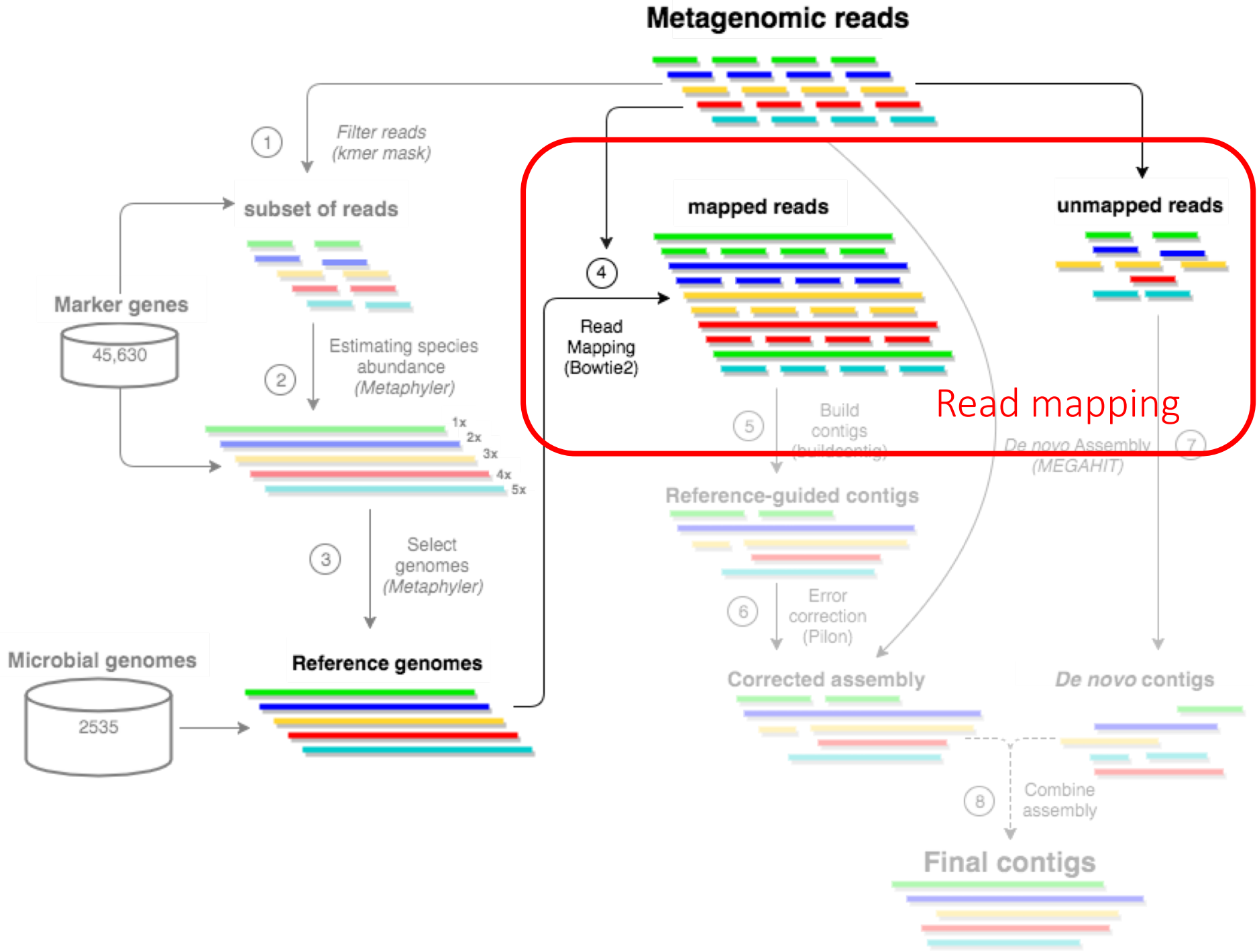
Abstract

Kraken is an ultrafast and highly accurate program for assigning taxonomic labels to metagenomic DNA sequences. Previous programs designed for this task have been relatively slow and computationally expensive, forcing researchers to use faster abundance estimation programs, which only classify small subsets of metagenomic data. Using exact alignment of *k*-mers, Kraken achieves classification accuracy comparable to the fastest BLAST program. In its fastest mode, Kraken classifies 100 base pair reads at a rate of over 4.1 million reads per minute, 909 times faster than Megablast and 11 times faster than the abundance estimation program MetaPhlAn. Kraken is available at http://ccb.jhu.edu/software/kraken/.

Keywords: metagenomics, sequence classification, sequence alignment, next-generation sequencing, microbiome

Survey results:

- https://ql.tc/axwUgB


Metagenomic reads
1
Filter reads
(kmer mask)
subset of reads
Marker genes
45,630
2
Estimating species
abundance
(Metaphyler)
1x
2x
3x
4x
5x
3
Select
genomes
(Metaphyler)
Microbial genomes
2535
Reference genomes
4
Read
Mapping
(Bowtie2)
mapped reads
unmapped reads
Read mapping
5
Build
contigs
(buildcontig)
Reference-guided contigs
6
Error
correction
(Pilon)
Corrected assembly
De novo Assembly
(MEGAHIT)
7
De novo contigs
8
Combine
assembly
Final contigs

STEP 2: Read mapping

Software | Open Access

Ultrafast and memory-efficient alignment of short DNA sequences to the human genome

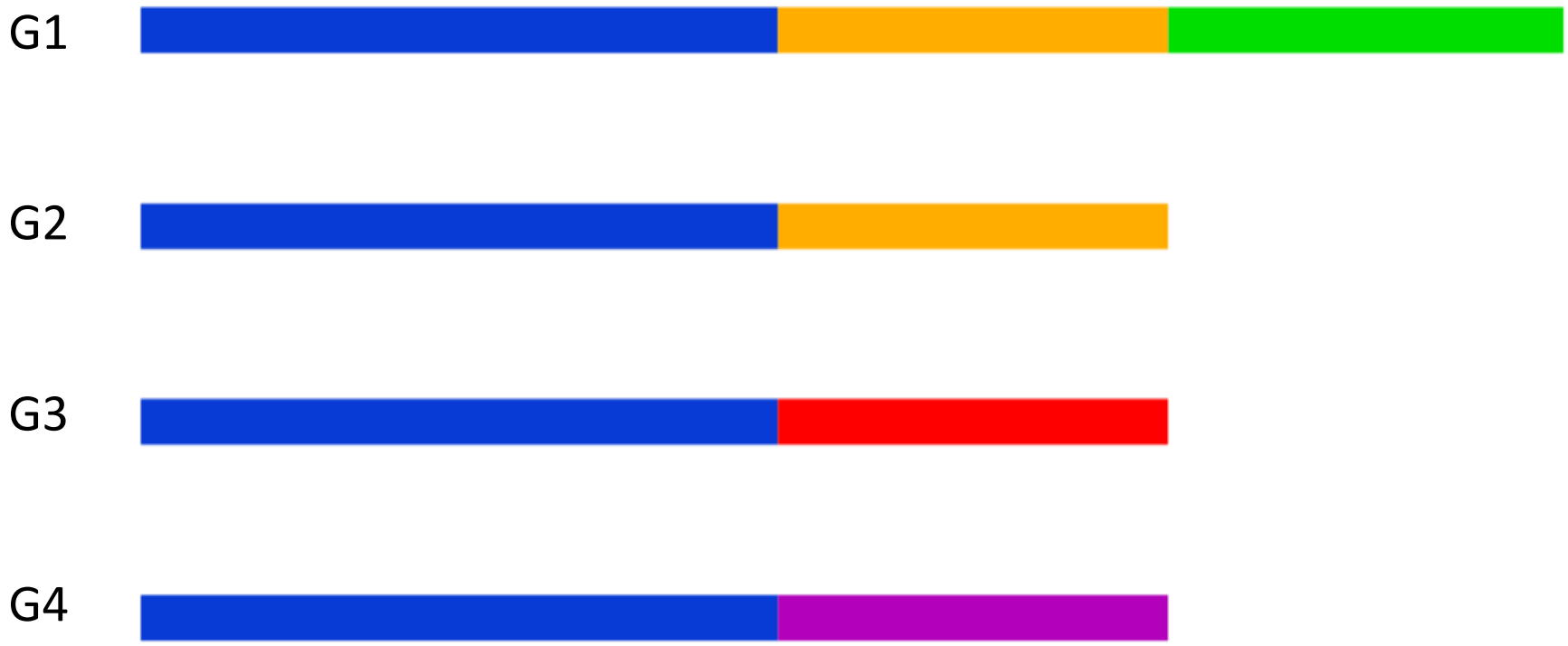
Ben Langmead ✉, Cole Trapnell, Mihai Pop and Steven L Salzberg

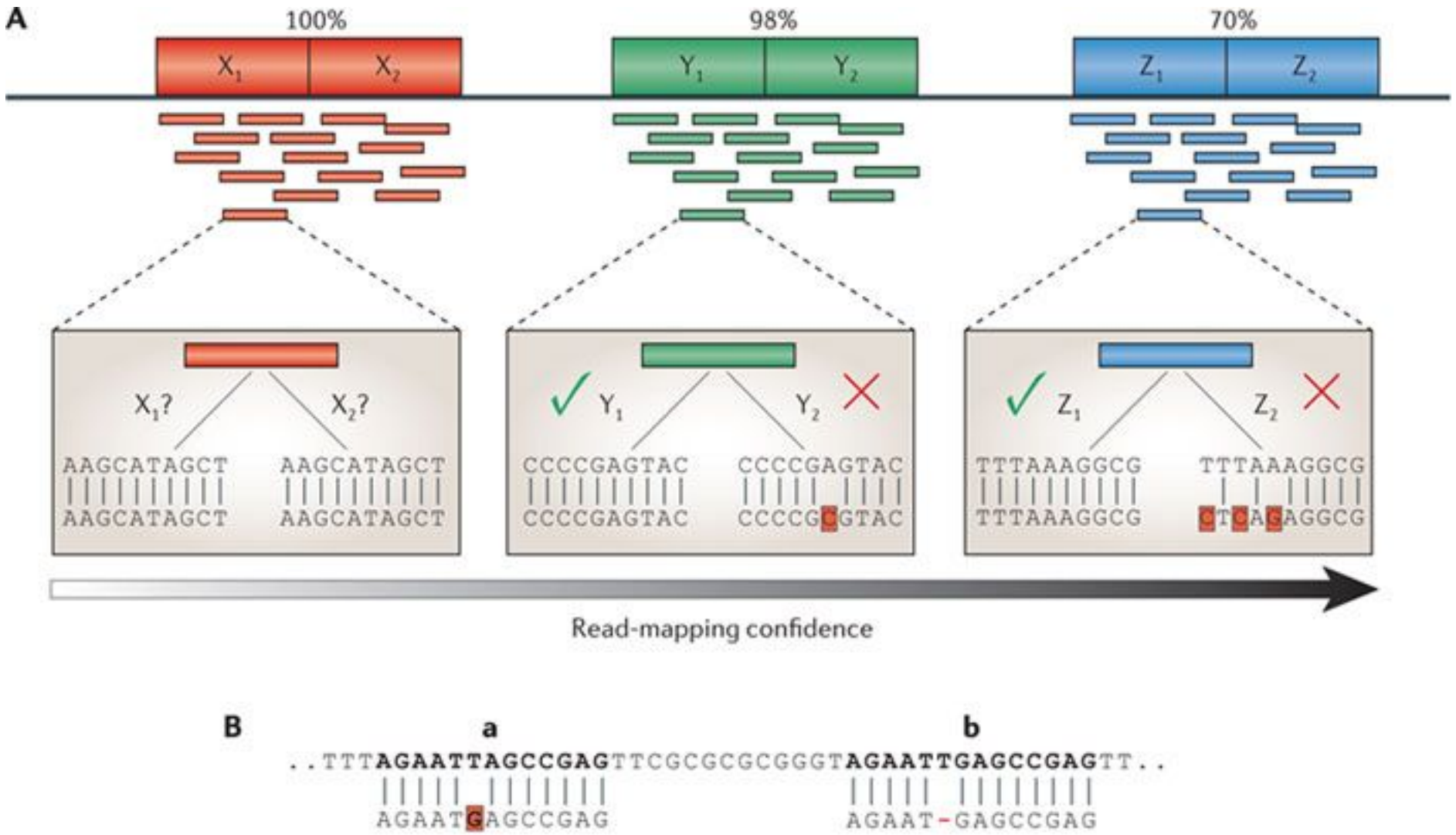
Genome Biology 2009 **10**:R25

https://doi.org/10.1186/gb-2009-10-3-r25

Received: 21 October 2008 | **Accepted:** 4 March 2009 | **Published:** 4 March 2009

STEP 2: Read mapping





Treangen and Salzberg, Nature Reviews Genetics, 2015

Nature Reviews | Genetics

Choose your own adventure: how shall we map reads to the recruited genomes?

1. All → map all reads to every equally good mapping location

2. Random → randomly assign reads amongst equally good mapping location

3. Depth → genome with highest depth of coverage takes all of the reads that map to it

4. Breadth → genome with highest breadth of coverage takes all the reads that map to it

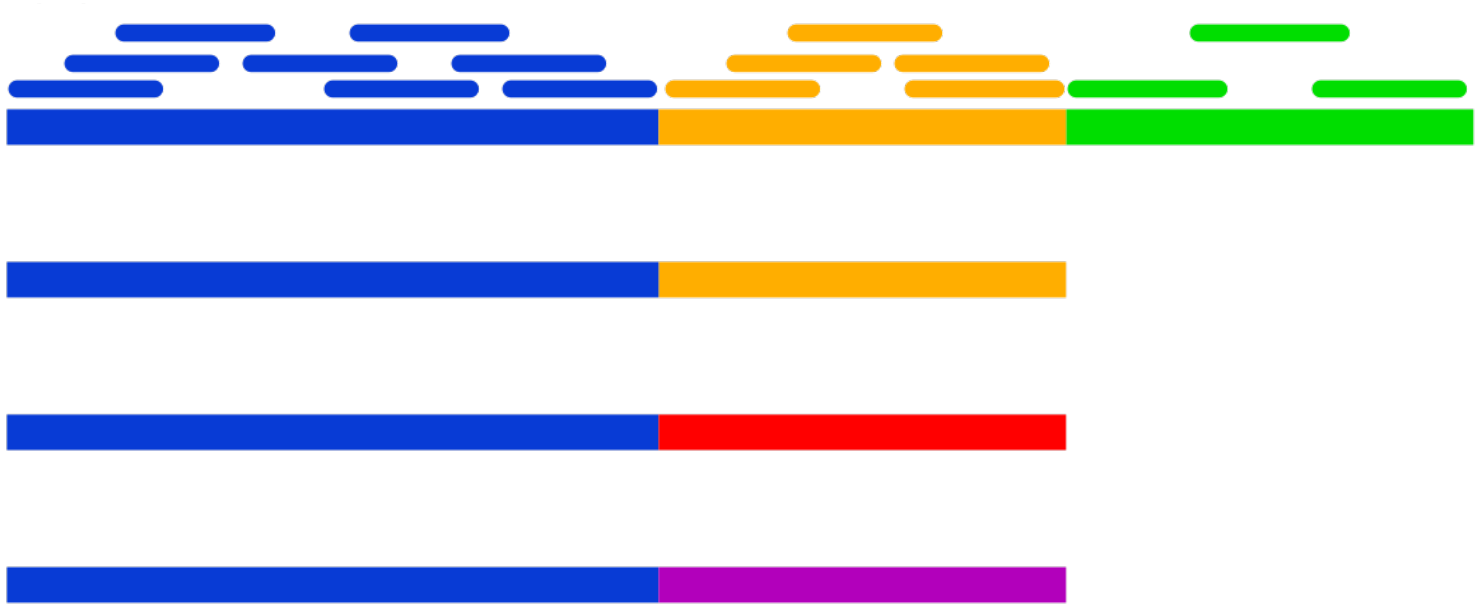
- http://tiny.cc/stamps19

READ MAPPING: How to place reads?

Random assignment "coin flip"

Rank by depth of coverage

Rank by minimum set cover



Survey results:

- https://ql.tc/axwUgB

Minimum set cover?

- Given a set of elements U { 1 , 2 , . . . , n } and a collection S of m sets whose union equals the universe, the set cover problem is to identify the smallest sub-collection of S whose union equals the universe.

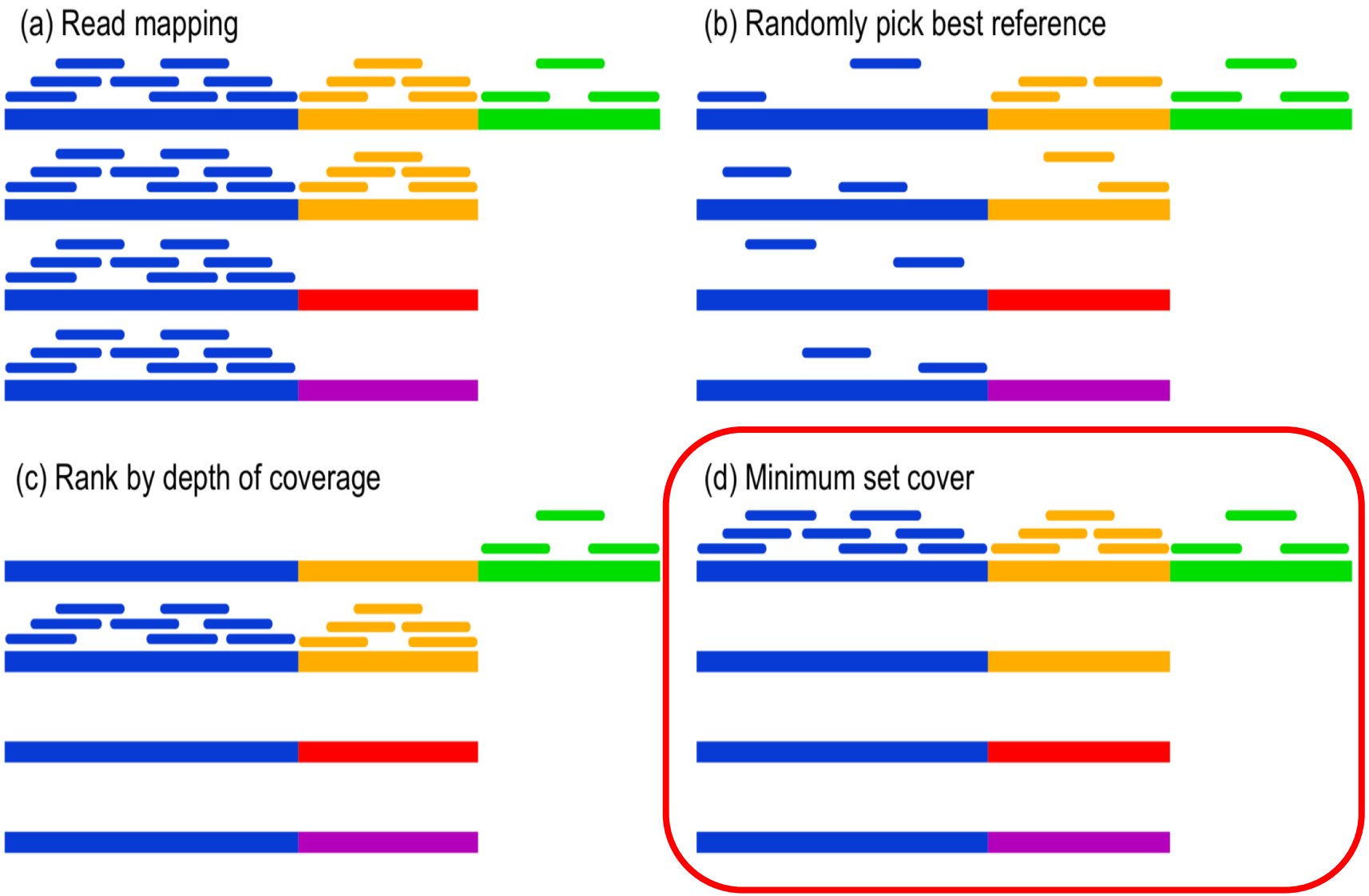
- For example, consider the universe U = { 1 , 2 , 3 , 4 , 5 } and the collection of sets S = { { 1 , 2 , 3 } , { 2 , 4 } , { 3 , 4 } , { 4 , 5 } } the union of S is U.

- The minimum set cover is the smallest number of m sets that cover U:

{ { 1 , 2 , 3 } , { 4 , 5 } }

- For reference selection, it's the smallest number of genomes that cover all of the input reads.

Read mapping selection: Minimum set cover



STEP 3: Building the contigs

```
                                        C
Reads      TGCACGGATG   TGCATGCACG
       TTAATGCACG   TG-ATGCATG
TGGATTAATG   TGGATG-ATG  C
TGGATTCATGCATGGATGCATGCATGCACG   Reference
TGGATTAATGCACGGATG-ATGCACTGCACG  Contig
```

Min. depth of coverage:2
Min. length:10

STEP 4: de novo assembly

Assembly unmapped reads to reference

De novo assembly using MEGAHIT

Evaluation Datasets

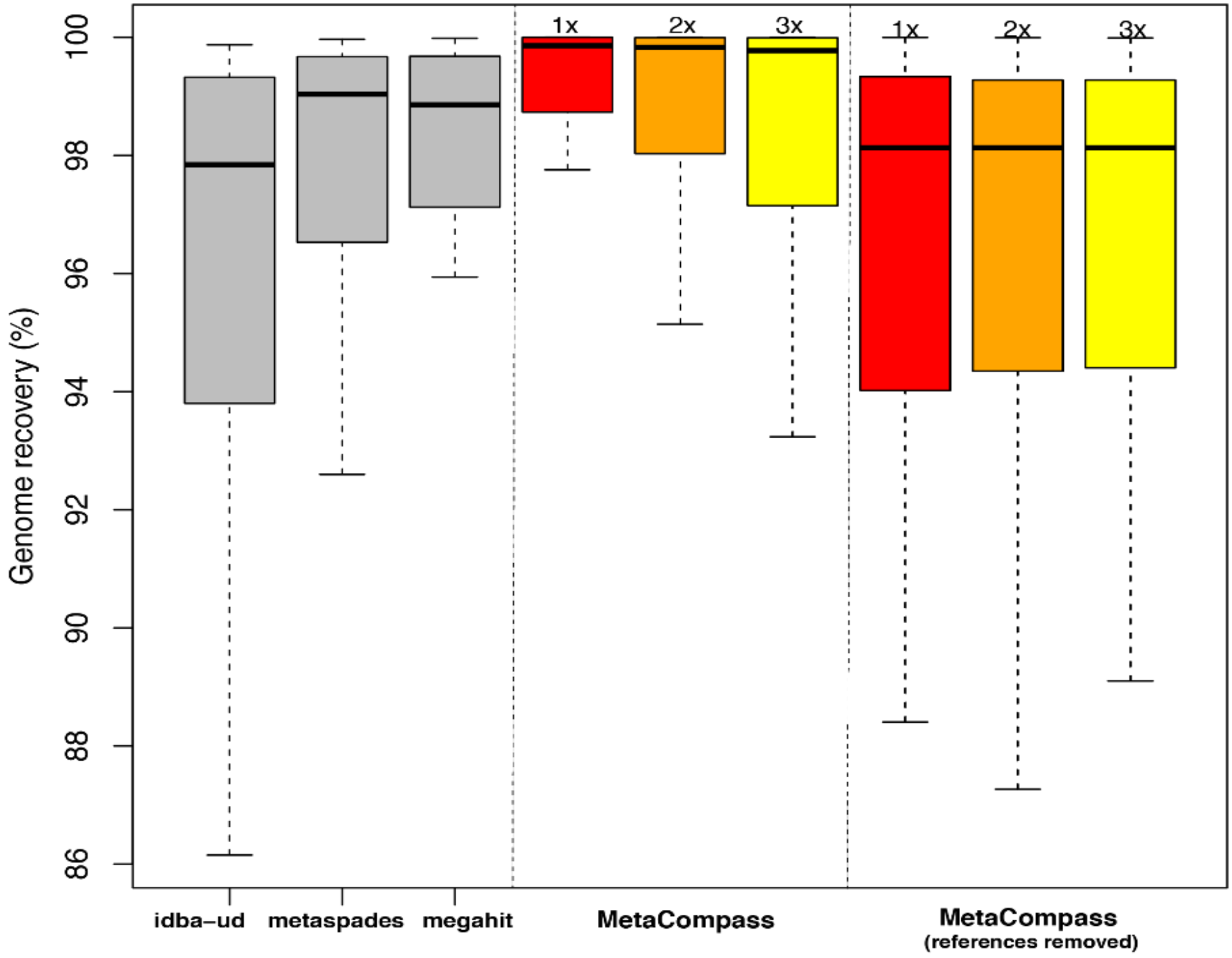
- Dataset 1: Synthetic dataset, Shakya et. al.
- Dataset 2: Down-sampled Dataset 1(low coverage)
- Dataset 3: 2,077 samples from HMP2

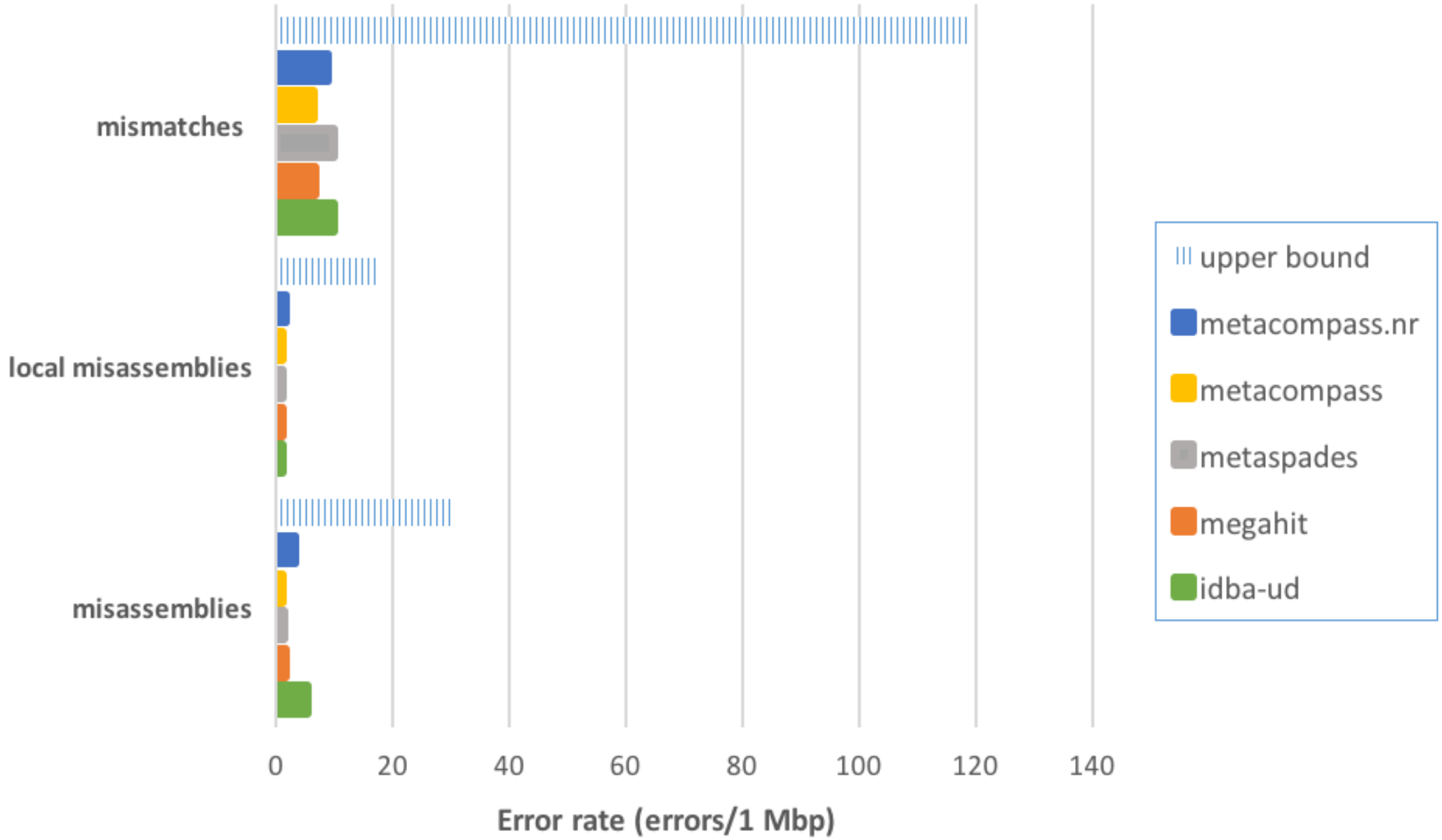
Results - Dataset 1

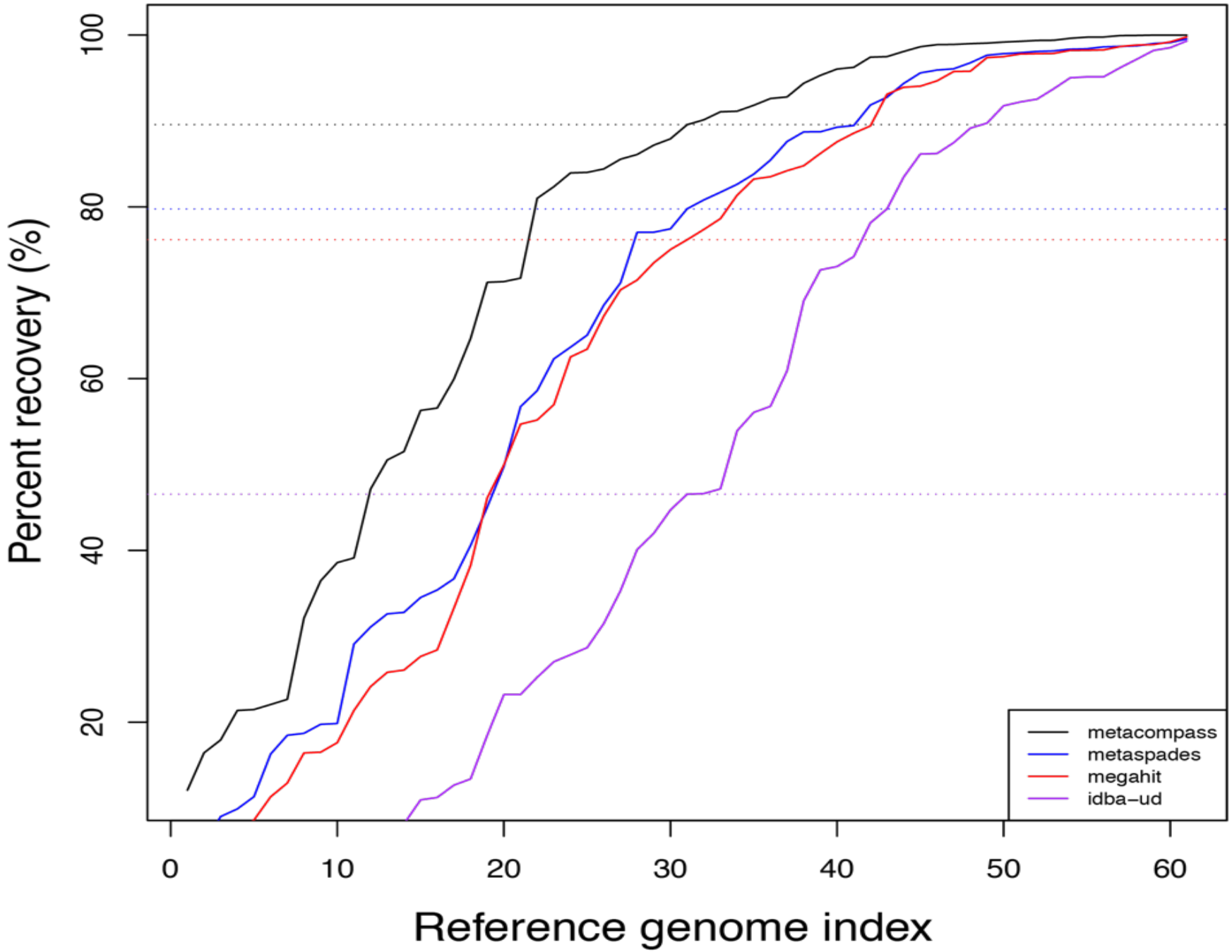
- Mixture of 64 bacterial and archaeal species (Shakya et al., 2013)
- 109 million reads with mean insert size 206 bp and 100 bp read length
- Easier to evaluate assembly since the truth is known

Results – Assembly **Statistics**

Method	No. Contigs	Longest Contig (bp)	Median genome recovery	Mismatches (Per 1Mbp)	Misassemblies (Per 1Mbp)
MetaCompass	**18,766**	**7,057,109**	**100%**	**61.9**	**1.9**
IDBA-UD	22,355	991,792	98%	98.6	6.3
MEGAHIT	35,351	1,151,857	99%	66.5	2.5
metaSPAdes	21,424	1,438,235	99%	97.1	2.3


Genome recovery (%)
100
98
96
94
92
90
88
86
1x
2x
3x
1x
2x
3x
idba−ud
metaspades
megahit
MetaCompass
MetaCompass
(references removed)


mismatches
local misassemblies
misassemblies
upper bound
metacompass.nr
metacompass
metaspades
megahit
idba-ud
0
20
40
60
80
100
120
140
Error rate (errors/1 Mbp)


Percent recovery (%)
100
80
60
40
20
0
10
20
30
40
50
60
Reference genome index
metacompass
metaspades
megahit
idba−ud

MetaCompass tutorial

-- I have a set of metagenomic reads and want to perform reference-guided assembly.

```
python go_metacompass.py -P [read1.fq,read2.fq] -l [max read length]-o [output_folder] -t [ncpu]
```

-- I know the reference genomes and have a set of metagenomic reads

```
python go_metacompass.py -r [references.fasta] -P [read1.fq,read2.fq] -o [output_folder] -t [ncpu]
```

Output directory:

```
 Assembled contigs:
            metacompass_output/metacompass.final.ctf.fa

  Selected Reference genomes:
            metacompass_output/metacompass.recruited.fa
```

Docker details

- Where do I save my output?

/output

- How do I exit the container?

exit

- How do I access my output after exiting the container?

cd /home/stamps19/chiron/metacompass

Warming up: *Carsonella ruddii*

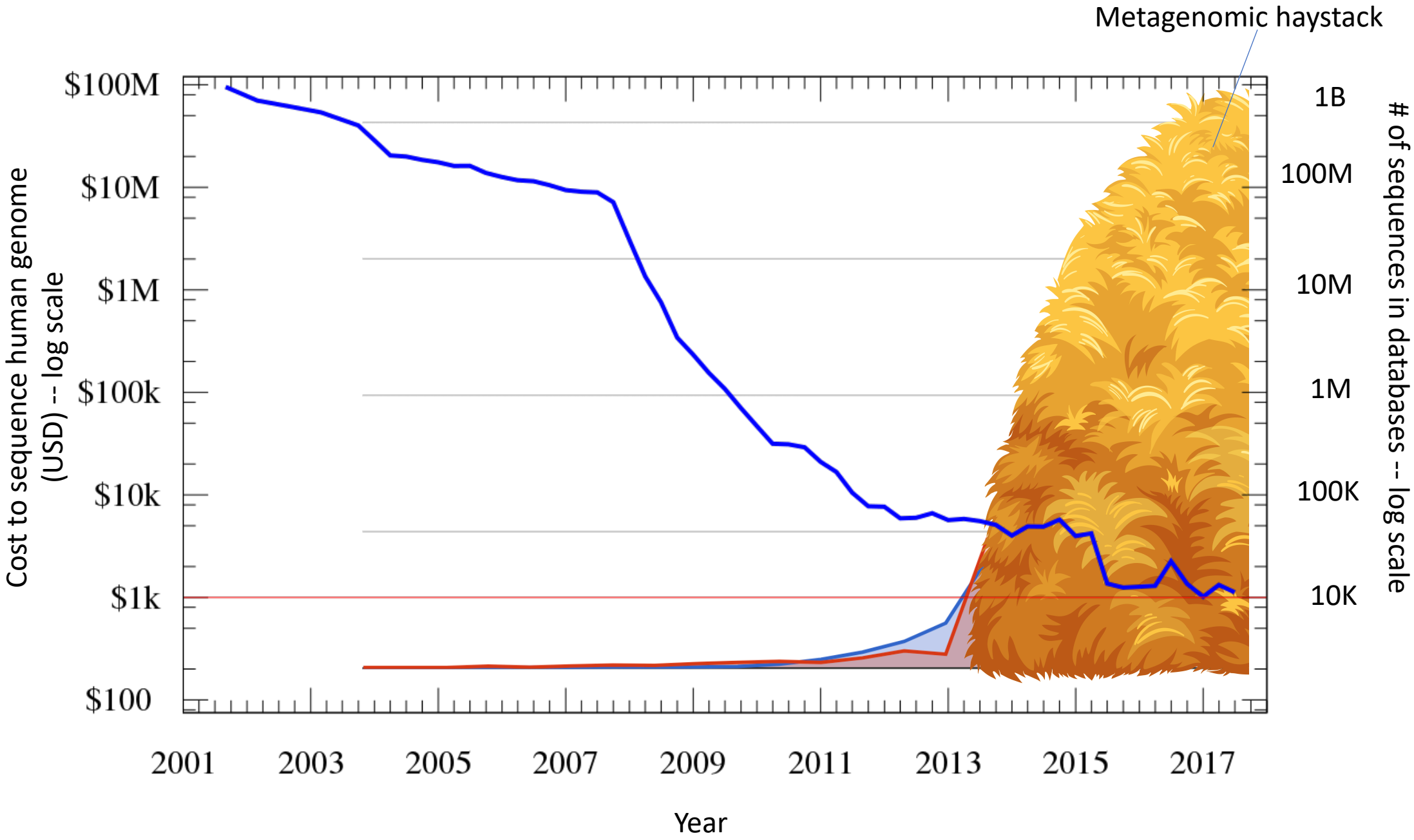
- Obligate endosymbiotic Gamma protobacteria
- Present in all species of phloem sap-feeding insects known as psyllids

Your turn!

- https://gitlab.com/treangen/stamps2019/README.md#part2

Part II

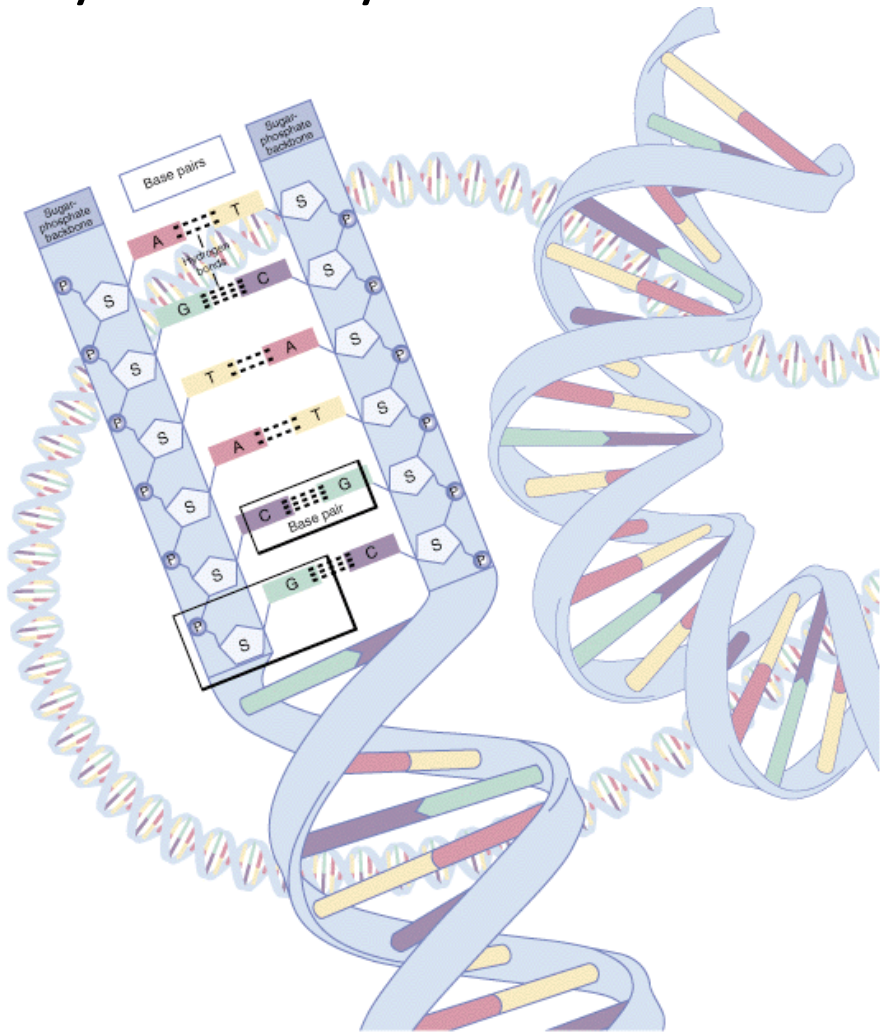
Strain-level analyses



https://en.wikipedia.org/wiki/DNA_sequencing

Documents of evolutionary history

- The **genome** of an organism contains its hereditary information and is encoded in its DNA.
- My research has focused on tracking evolution, one nucleotide at a time.
- How?



Zuckerandl and Pauling, “Documents of Evolutionary History” 1965.

How to detect strains from metagenomes ?

- **Known microbe/virus/pathogen in compex sample**
  - Gene expression array cards (TLDA)
  - PCR-based, targeted
  - Computational approaches
    - Signature based
    - **Marker based (MetaCompass)**
    - **K-mer based (more about this tomorrow!)**
    - Phylogenetic placement

- **Interesting genome discovery/unknown (pathogen) in sample**
  - DB might be helpful, likely not
  - Approaches:
    - Profile HMM for distant homology detection
    - **Assembly -> Annotation -> Systems based approach**

Should strain detection methods operate on:

1. Bags of kmers?
2. Bags of genes?
3. Bags of replicons?
4. Signature sequences?

Choose your own adventure: strain detection methods should operate on:

1. Bags of kmers?
2. Bags of genes?
3. Bags of replicons?
4. Signature sequences?

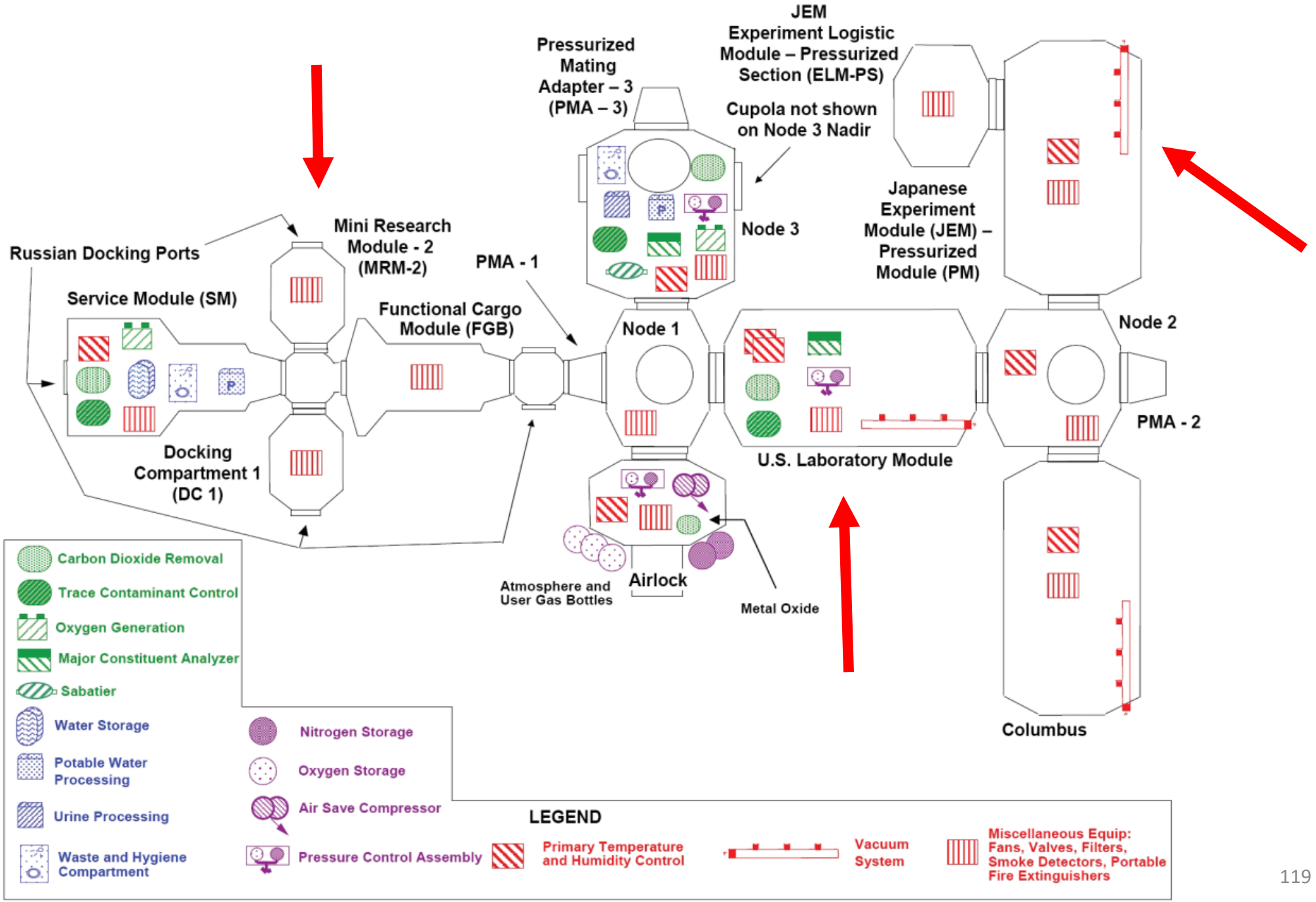
- http://tiny.cc/stamps19

Survey results:

- https://ql.tc/axwUgB

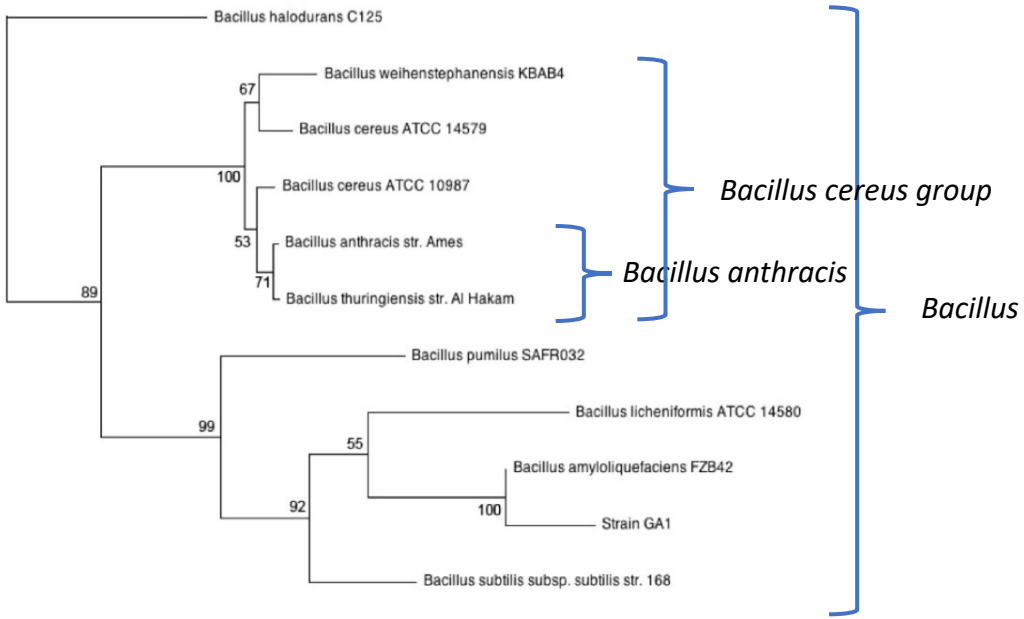
Investigation of *Anthrax* on the ISS

- Air filters from the International Space Station sent to NBACC for analysis
- Initial analysis suggested likely *Anthrax* presence
- Computing a multiple sequence alignment (MSA) allowed us to detect a single-nucleotide difference and determine Anthrax was not present on the International Space Station


JEM
Experiment Logistic
Module – Pressurized
Section (ELM-PS)
Pressurized
Mating
Adapter – 3
(PMA – 3)
Cupola not shown
on Node 3 Nadir
Node 3
Japanese
Experiment
Module (JEM) –
Pressurized
Module (PM)
Russian Docking Ports
Mini Research
Module - 2
(MRM-2)
PMA - 1
Service Module (SM)
Functional Cargo
Module (FGB)
Node 1
Node 2
PMA - 2
U.S. Laboratory Module
Docking
Compartment 1
(DC 1)
Atmosphere and
User Gas Bottles
Airlock
Metal Oxide
Columbus
Carbon Dioxide Removal
Trace Contaminant Control
Oxygen Generation
Major Constituent Analyzer
Sabatier
Water Storage
Potable Water
Processing
Urine Processing
Waste and Hygiene
Compartment
Nitrogen Storage
Oxygen Storage
Air Save Compressor
Pressure Control Assembly
LEGEND
Primary Temperature
and Humidity Control
Vacuum
System
Miscellaneous Equip:
Fans, Valves, Filters,
Smoke Detectors, Portable
Fire Extinguishers

Environmental surveillance on the ISS

- Microbial environment on the ISS is monitored continually
  - Routine testing performed in space
  - Sample sets sent back to Earth for more detailed analyses
- Samples from ISS air filters in American, Japanese, and Russian modules produce bacterial growth consistent with *Bacillus spp.*



- **Initial testing**
  - **Morphology/phenotypic testing indicates *Bacillus* isolates are part of the *B. cereus* group**
  - **16S rRNA and *gyrB* sequencing consistent with *B. anthracis*, though PCR tests for virulence markers are negative**
  - **MLST and WGS/Avg Nucleotide Identity analyses also point to *B. anthracis***

B. anthracis on the ISS?

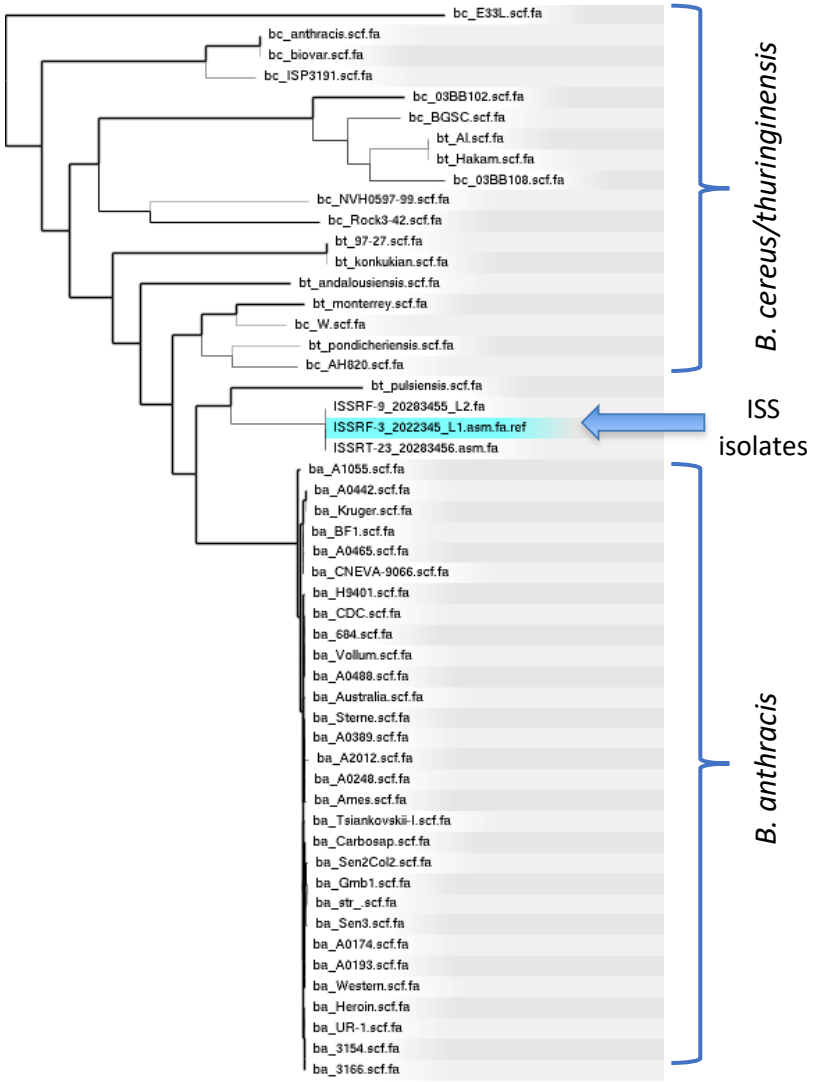
Bacillus anthracis + plcR biology

- Can a single SNP matter?
  - YES!

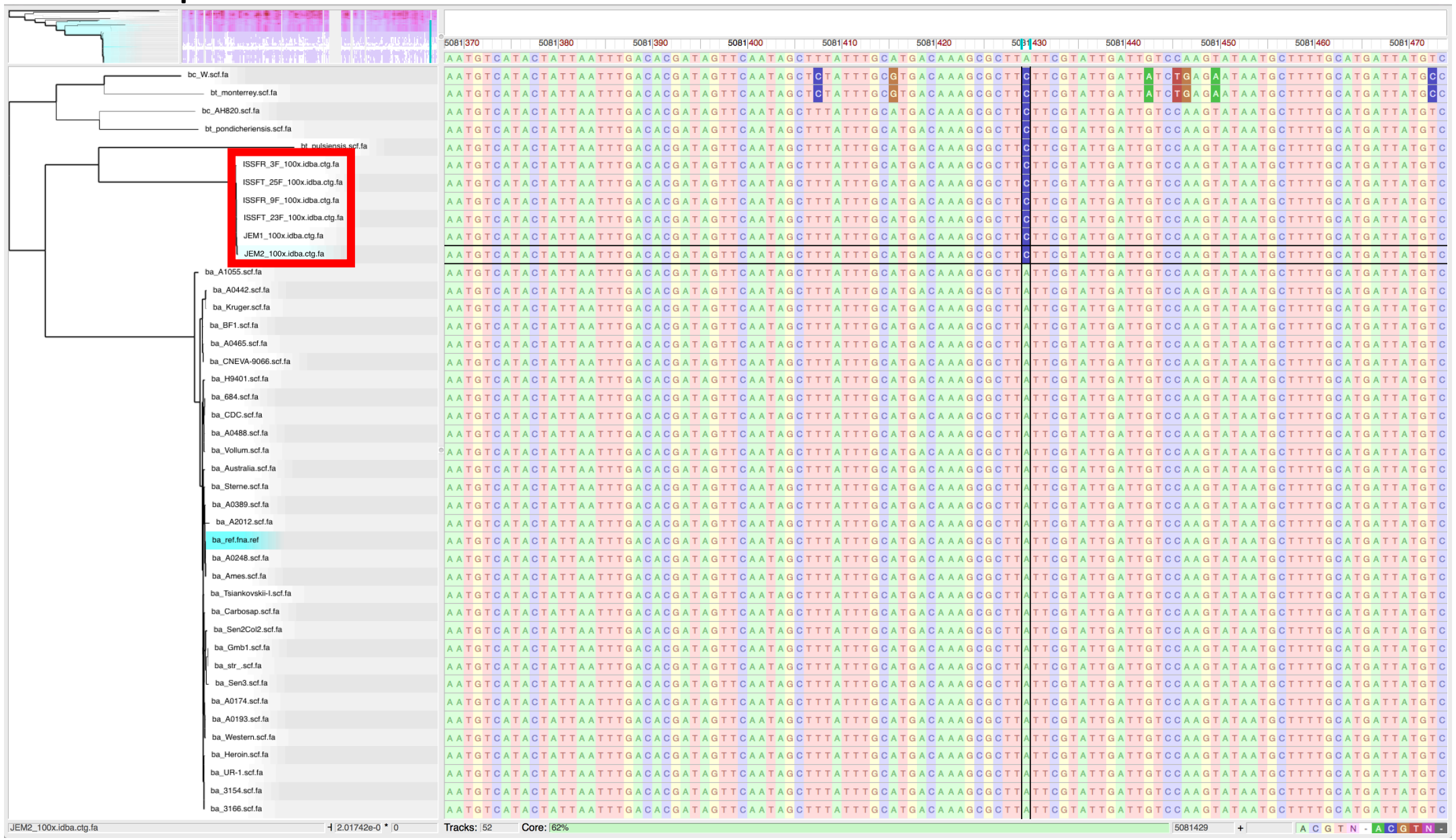
- plcR expression interferes with sporulation in B. anthracis.
  - Sporulation is a critical component of the ecology of B. anthracis
  - plcR nonsense mutation present in all known B. anthracis genomes

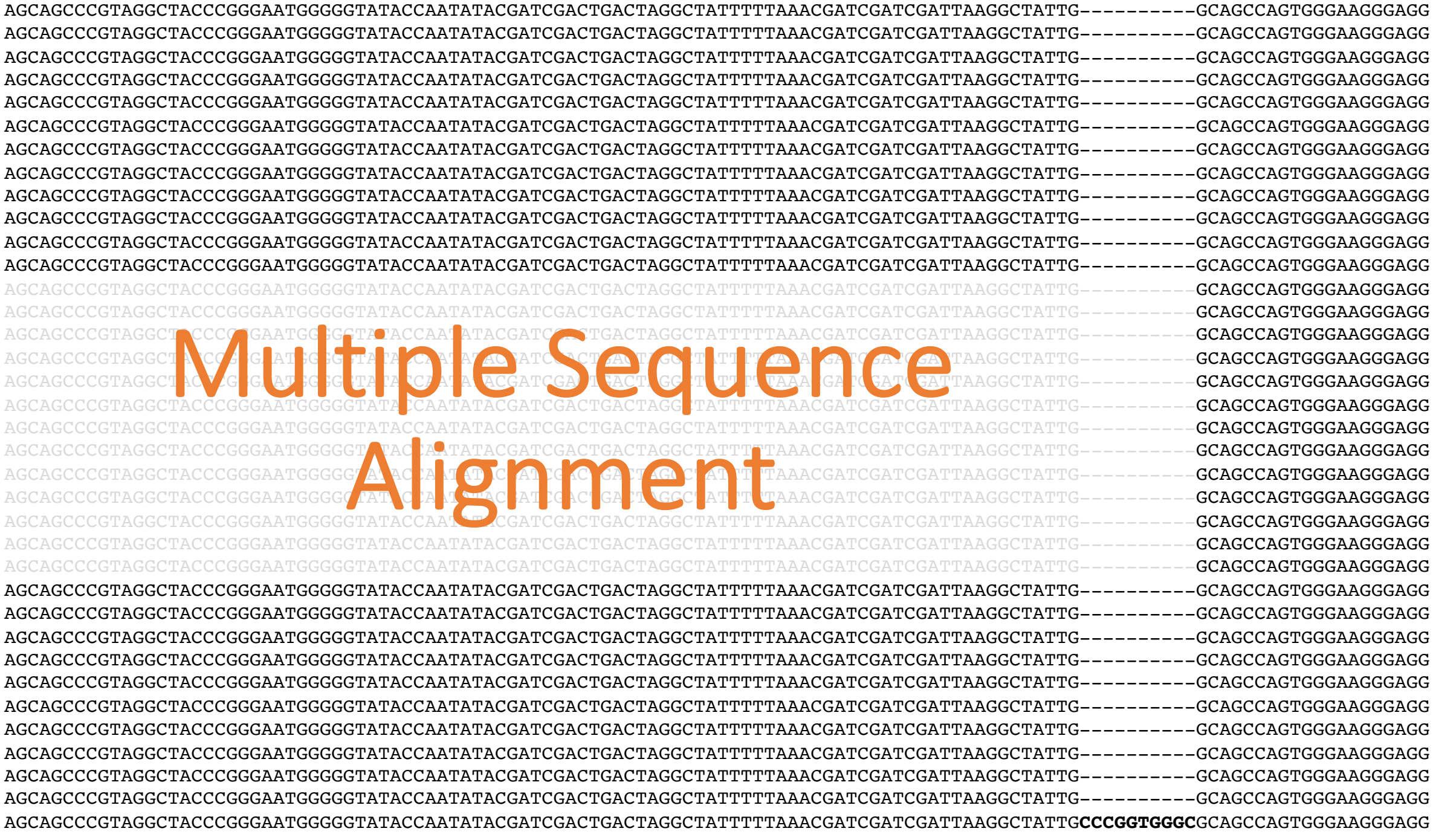
Genomic analysis

- Grow US, Japanese, and Russian isolates and extract genomic DNA
- WGS on multiple platforms
  - Illumina
  - Pac Bio (Oxford Nanopore)
- All isolates are very similar
  - <20 SNP's separate the most distant pair
- Closest to *B. anthracis* clade, but still ~50,000 SNP's away from nearest *B. anthracis* (not a member)
- No *B. anthracis* virulence genes
- New species
  - *Bacillus ???*



plcR nonsense mutation




Multiple Sequence
Alignment

Multiple Sequence Alignment (MSA): *another grand challenge*[1]

```
S1 = AGGCTATCACCTGACCTCCA          S1 = -AGGCTATCACCTGACCTCCA
S2 = TAGCTATCACGACCGC              S2 = TAG-CTATCAC--GACCGC--
S3 = TAGCTGACCGC                   S3 = TAG-CT-------GACCGC--
  ...                              ...
Sm = TCACGACCGACA          ⟶      Sm = -------TCAC--GACCGACA
```

Novel techniques needed for scalability and accuracy

NP-hard problems and large datasets
Current methods do not provide good accuracy
Few methods can analyze even moderately large datasets

Many important applications besides phylogenetic estimation

[1] Frontiers in Massive Data Analysis, National Academies Press, 2013

Given: Two strings

$$a = a_1 a_2 a_3 a_4 \ldots a_m$$
$$b = b_1 b_2 b_3 b_4 \ldots b_n$$

where a_i, b_i are letters from some alphabet like {A,C,G,T}.

Compute how similar the two strings are.

What do we mean by "similar"?

Edit distance between strings a and b = the smallest number of the following operations that are needed to transform a into b:

- Replace a character
- Delete a character
- Insert a character

Multiple Sequence Alignment

Input: Sequences S_1 , S_2 , … S_k;

Let $\boldsymbol{M}$ be a **MSA** between these sequences.

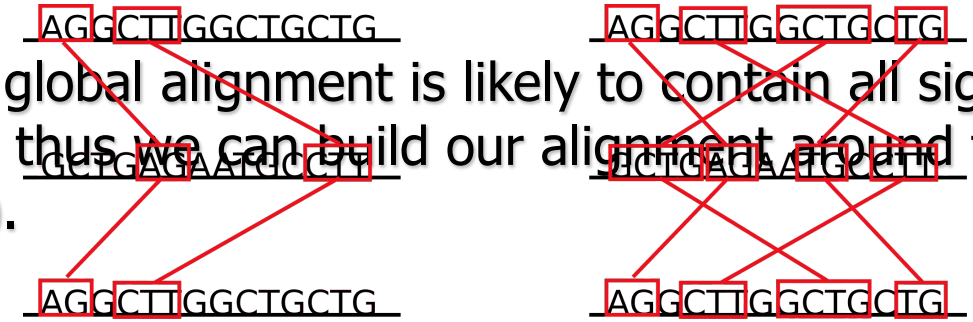
Let $\boldsymbol{d_M(Si\ \ Sj)}$ = be the edit distance of the alignment between $S_i^{'}$ and S_j.

D$(\boldsymbol{M}) = \sum_{ij} \boldsymbol{d_M(Si, Sj)}$ = Sum of all pairwise edit distances be the edit distance of the alignment between S_i and S_j.

- **Two important parameters: the number k of sequences, the length n of the longest sequence**.

Maximal Unique Matches (MUMs)

- Using unique matches as anchors, we can reduce the computational cost by partitioning large DNA sequences into tractable parts.
- The idea is that any optimal global alignment is likely to contain all significant common unique regions and thus we can build our alignment around these anchors (Delcher et al 1999).



- Anchor-based sequence alignment has been widely recognized as a reasonable heuristic for efficient multiple global alignment, and is featured in tools such as MUMmer and MAUVE.

How to detect unique matches in multiple DNA sequences?

- Several approaches exist, but the most efficient have been based around using suffix trees & suffix arrays.

- We use a compressed suffix structure, a specialization of a suffix tree, with increased efficiency and reduced space requirements with respect to the number of genomes being compared.

Suffix tree

Given string ababaabs#:

Suffixes:

1: ababaabs#

2: babaabs#

3: abaabs#

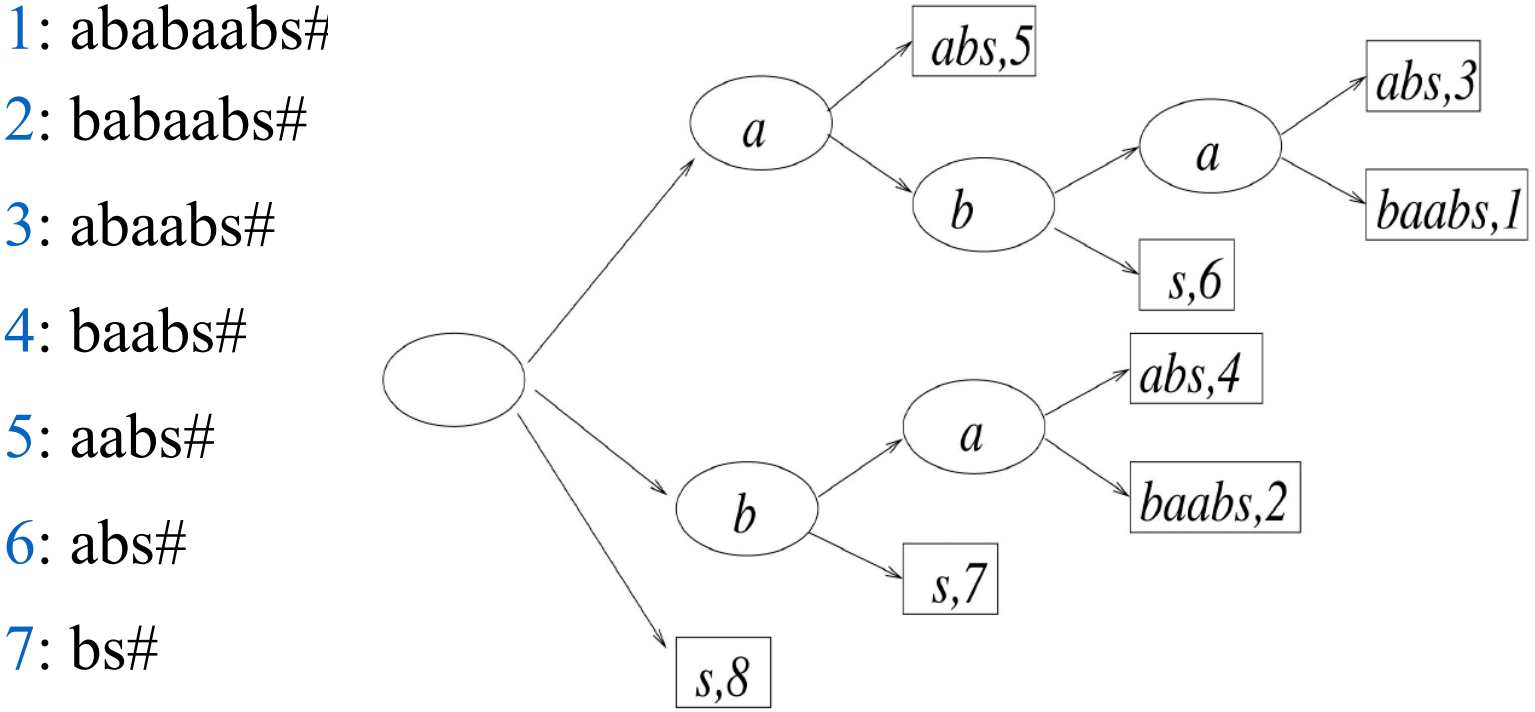
4: baabs#

5: aabs#

6: abs#

7: bs#

8: s#

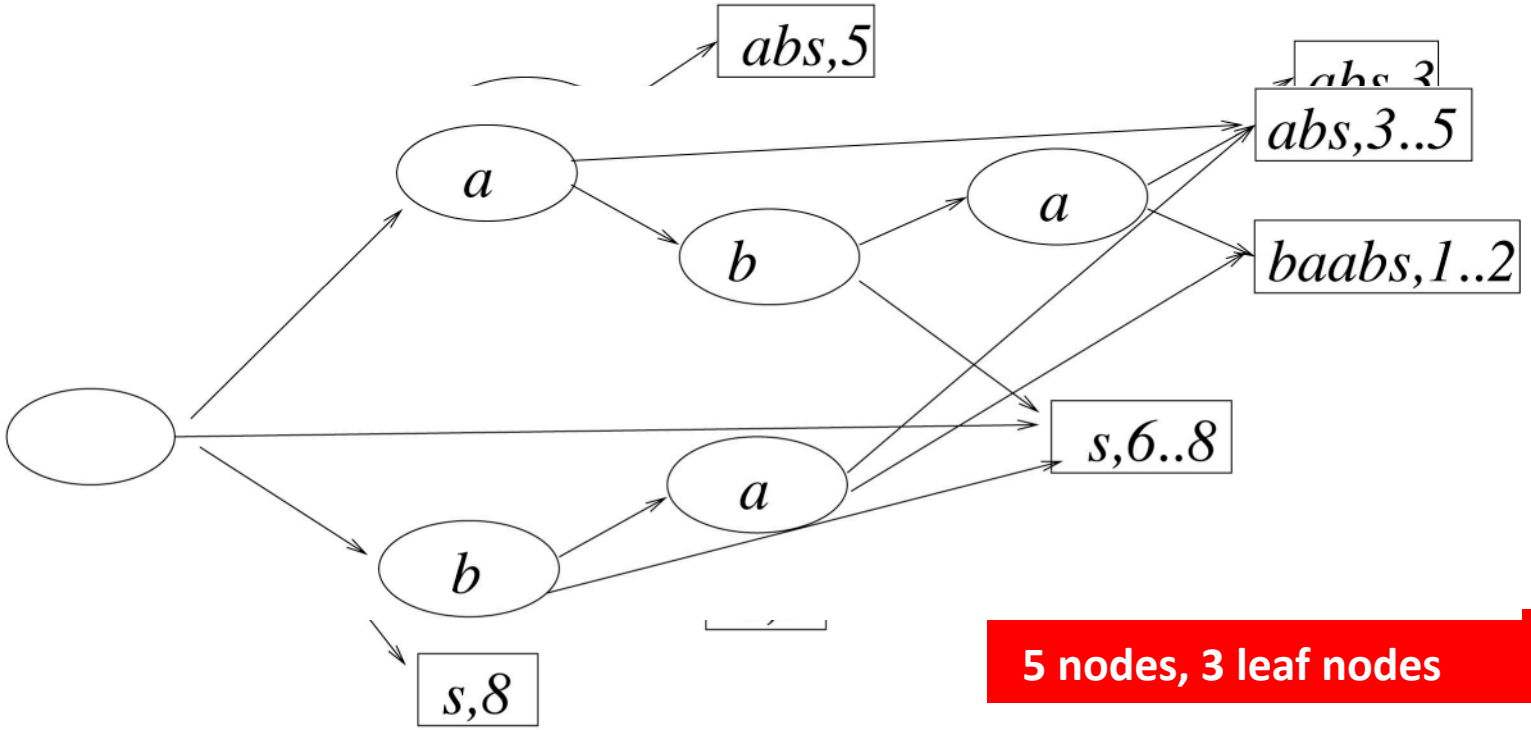


How can we generate a genome comparison framework efficiently?

1) Compress the suffix tree data structure.

2) Stream the remaining $k - 1$ sequences through the suffix structure for linear time complexity O($|S_1|+ \ldots +| S_k|$)

Compressing the suffix tree



Suffix tree of the string *abababaabs*.

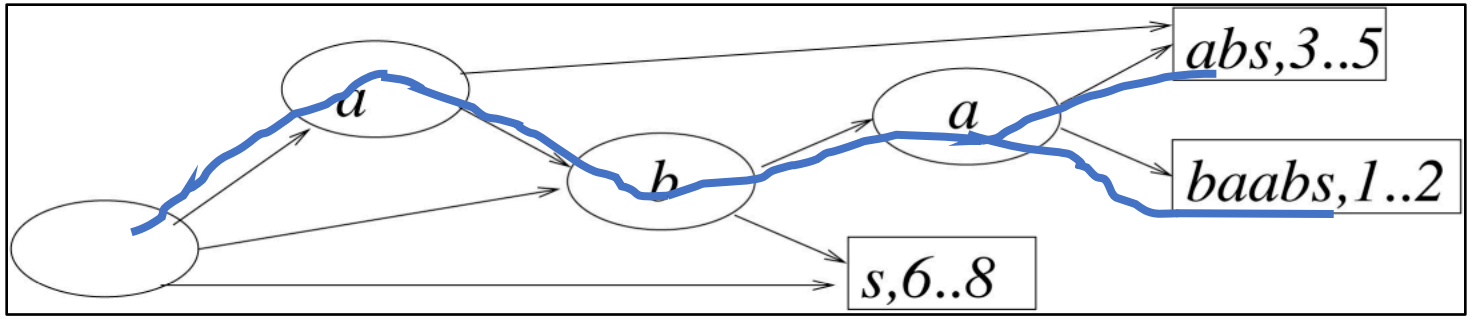
Treangen, Todd J., and Xavier Messeguer. "M-GCAT: interactively and efficiently constructing large-scale multiple genome comparison frameworks in closely related species." BMC bioinformatics 7.1 (2006): 433.APA

How can we generate a genome comparison framework efficiently?

1) Compress the suffix tree data structure (McCreight 1976, Ukonnen 1995).

2) Stream the remaining $k-1$ sequences through the suffix structure for linear time complexity O($|S_1|+ \ldots +|S_k|$)

MUM search algorithm



Unique Matches

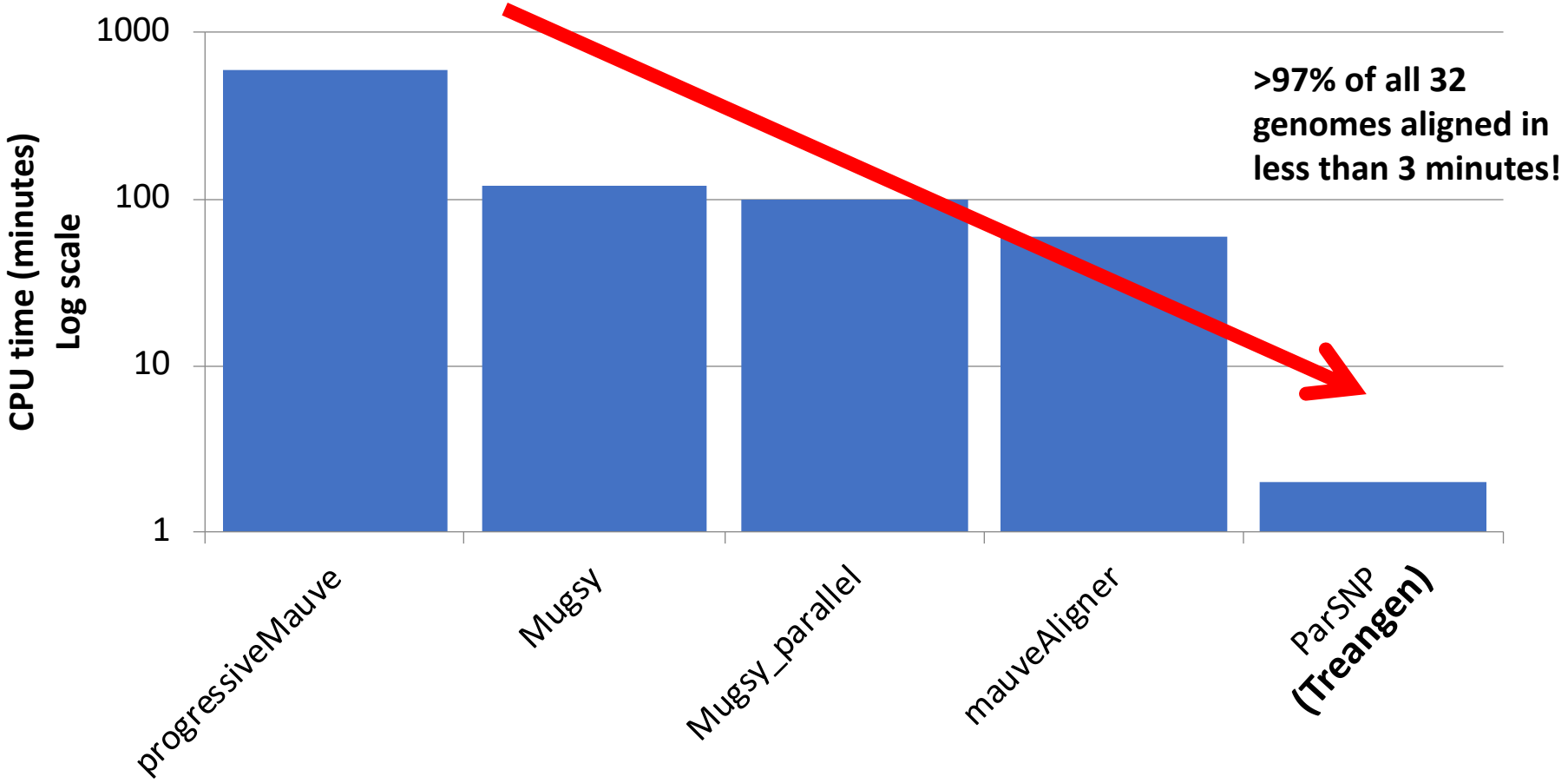
abaa
ababaabs

abaasbbababaabs

Treangen, Todd J., and Xavier Messeguer. "M-GCAT: interactively and efficiently constructing large-scale multiple genome comparison frameworks in closely related species." BMC bioinformatics 7.1 (2006): 433.APA

Run time performance

(32 simulated E. coli W3110 genomes)



Treangen, Ondov, Koren, & Phillippy, A. M. (2014). Rapid and interactive microbial genome alignment of thousands of genomes. *Genome Biology.*

Sustainable and organic strain-level harvesting with ParSNP

Motivation

- Hundreds to thousands of closely-related strains now available from public genome DBs and Metagenomes
  - Only to increase with 100K Foodborne Pathogen Genome Project (http://100kgenome.vetmed.ucdavis.edu), etc

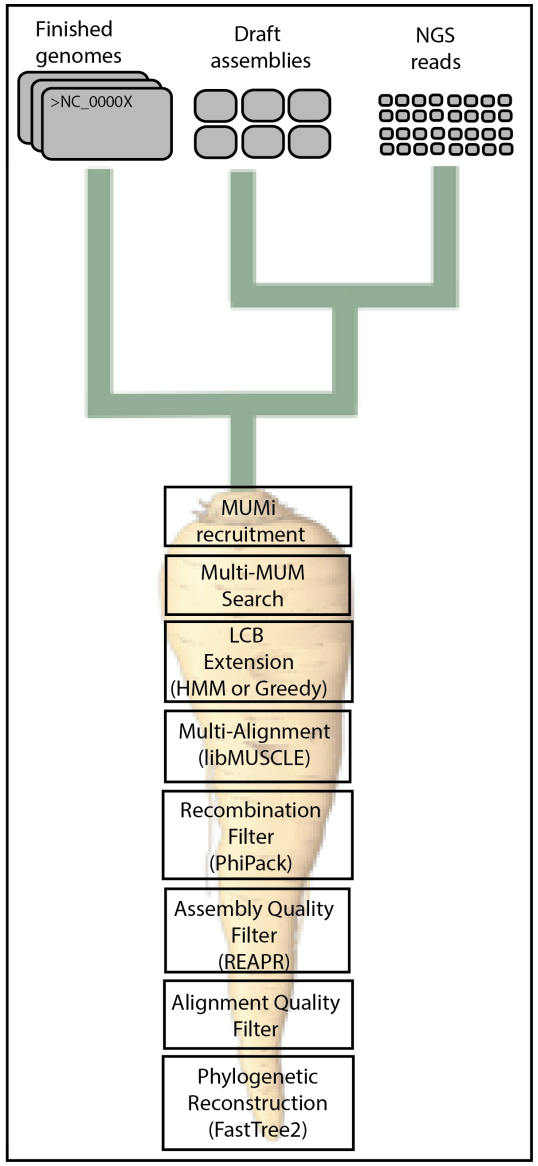
- Large-scale whole genome SNP typing and phylogenetic reconstruction is becoming a common task

ParSNP

1. NGS reads, draft assemblies, and/or finished genomes as input

1. Near-neighbor genomes are recruited

2. **Efficient Multi-MUM search to identify locally collinear blocks**



4. Multi-Alignment of LCBs with Muscle

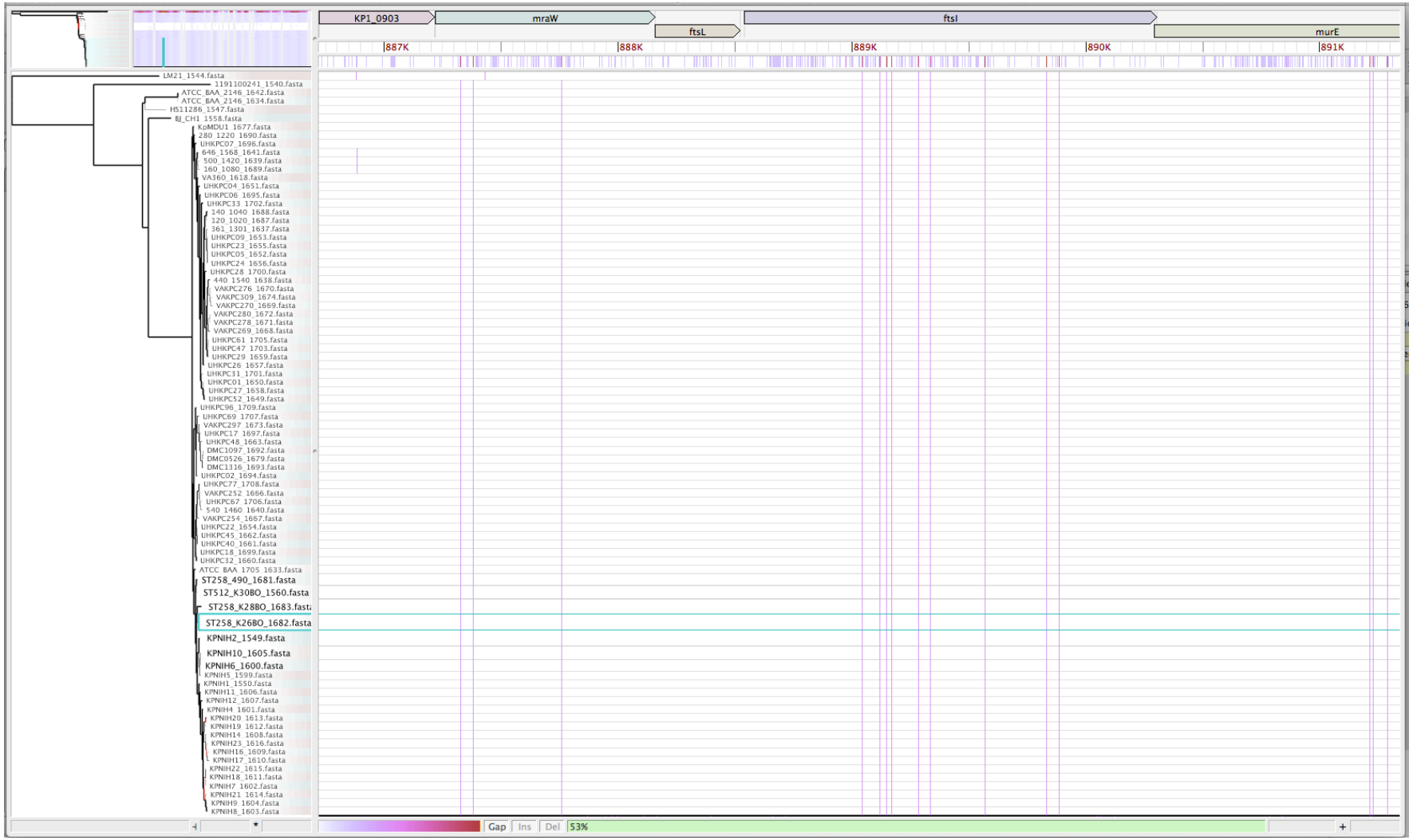
5. HMM-based LCB extension

6. Apply quality filters

7. Phylogenetic reconstruction with FastTree2

...with a touch of Gingr

100+ *Klebsiella pneumoniae genomes* (Primary developer: Brian Ondov)



Conclusions

① ParSNP can align hundreds closely-related microbial strains in *minutes*

- 200 *K. pneumoniae* genomes in 20 minutes on 8 cores

② Gingr provides an interactive interface for the simultaneous visualization of:

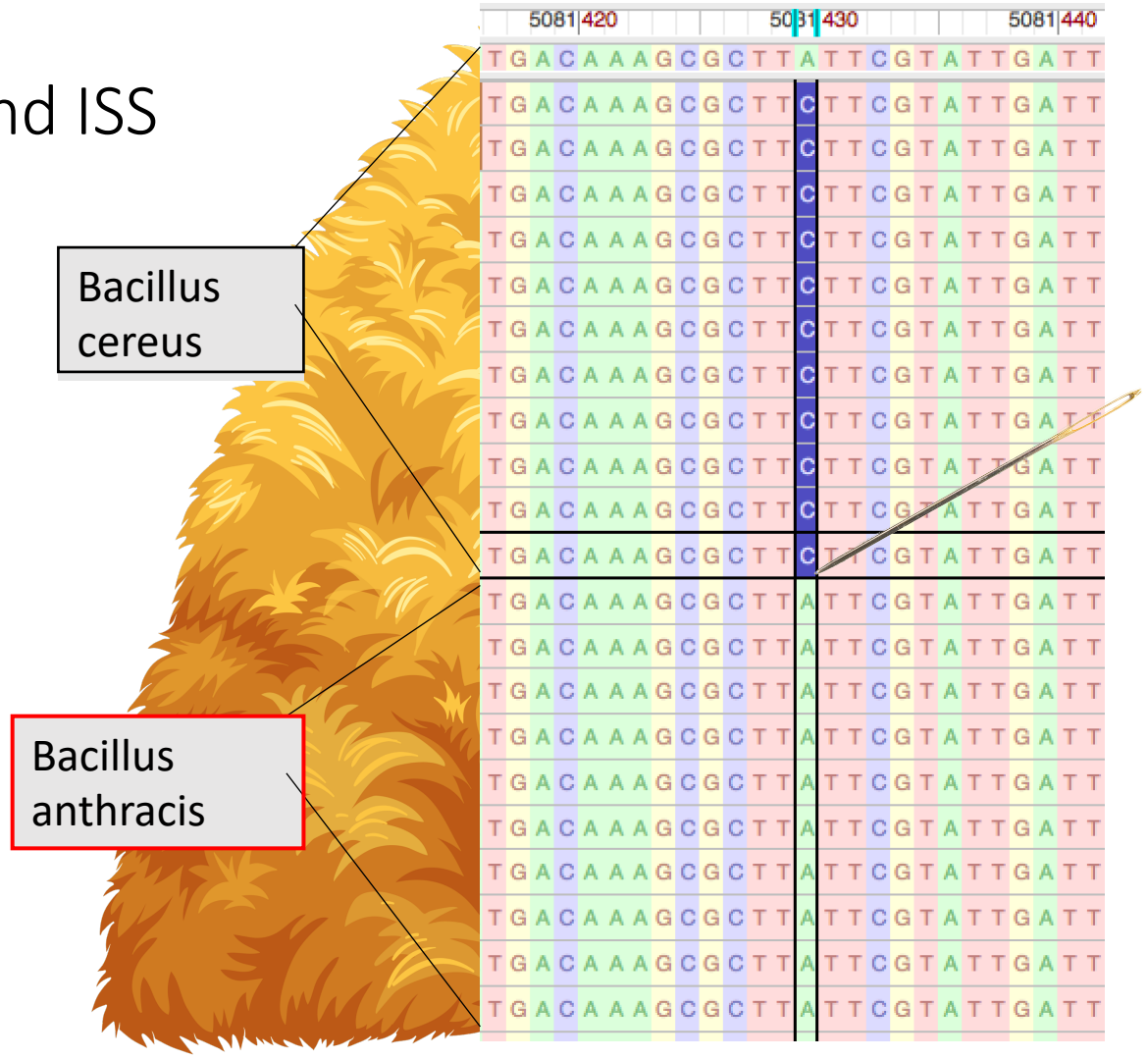
- *core genomes*
- *SNPs*
- *clade phylogeny*

③ Assembly + multiple alignment as a sustainable path to core genome SNP typing:

- 825 genomes 1500 GB (reads) 3.3 GB (asms) 0.13 GB (aln)

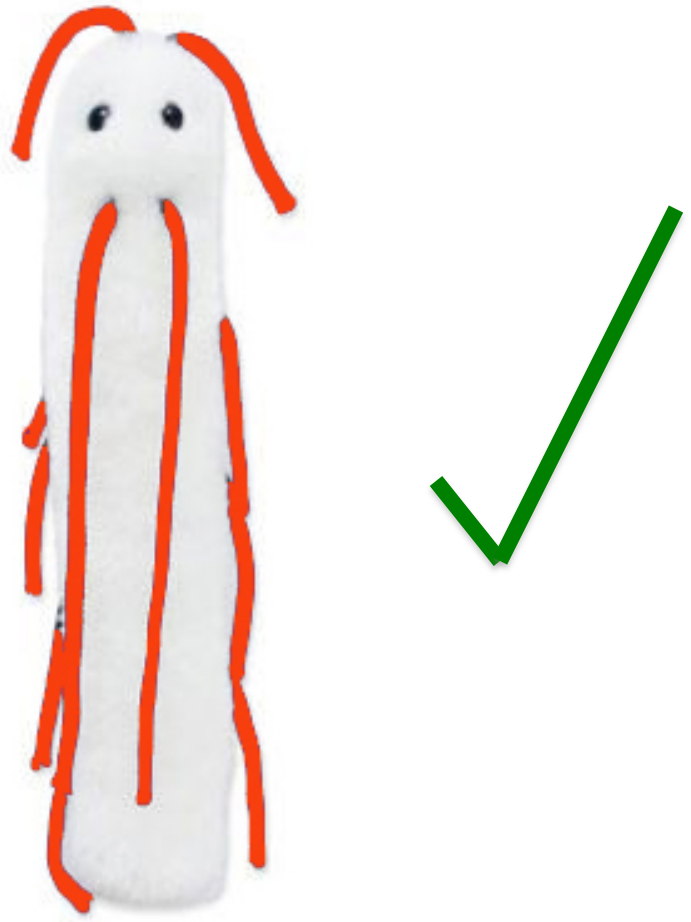
MSA and ISS

- Confirmation of single nucleotide difference via MSA
- 1 single nucleotide difference (needle) out of 5,000,000 nucleotides (haystack)



Conclusions

B. anthracis



***Novel B. cereus group* strain**

Your turn!

- https://gitlab.com/treangen/stamps2019/README.md#part3

Final exercise: AliView

- https://gitlab.com/treangen/stamps2019/README.md#part4