Benchmark of *de novo* Short Read Assembly Strategies for Metagenomics

Ino de Bruijn

Content

- Introduction on Assembly
  - What is assembly?
  - Popular methods
- Benchmark Method
  - Which assemblers have been validated?
  - How have they been validated?
- Benchmark Results
  - What is the most awesome assembler?

Introduction on Assembly

- Different technologies to sequence your sample
  - Sanger (400-600nt, error rate 0.001-1.0%)
  - 454 (400nt, error rate 1-4%)
  - Illumina (100nt, error rate 0.1-1%)
  - PacBio (3k-5knt, error rate 13-20%)
  - Ion Torrent (200-400nt, error rate 0.5-2.5%)
- Assembly turns reads into contigs and/or scaffolds
- Algorithms fit sequencing technologies
- Three general approaches
  - Overlap-Layout-Concensus (Long reads Celera 2000, Newbler 2005)
  - Greedy (Short reads, SSAKE 2007)
  - de Bruijn Graph (Short reads, Velvet 2008)

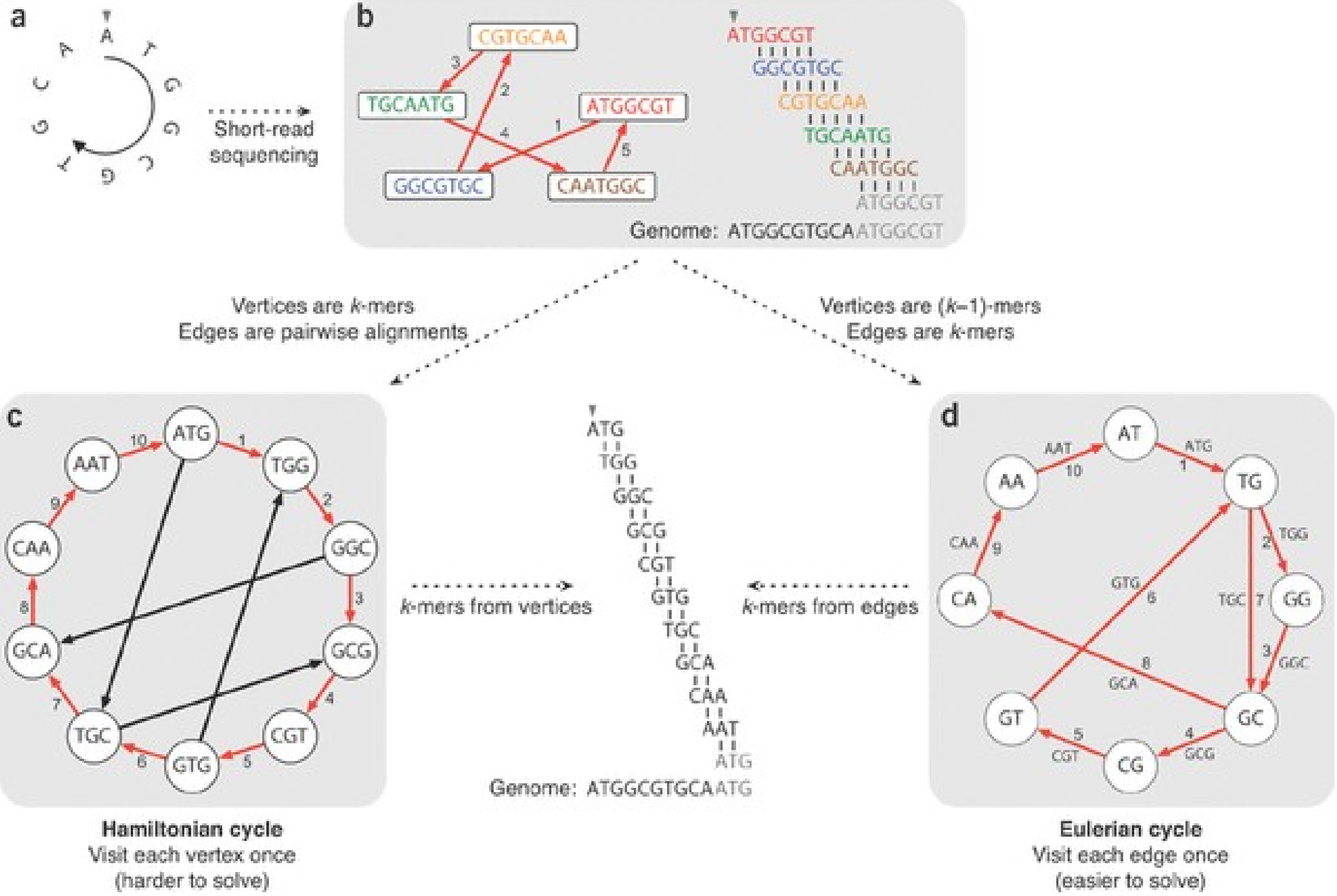
Introduction on Assembly

- Different technologies to sequence your sample
  - Sanger (400-600nt, error rate 0.001-1.0%)
  - 454 (400nt, error rate 1-4%)
  - Illumina (100nt, error rate 0.1-1%)
  - PacBio (3k-5knt, error rate 13-20%)
  - Ion Torrent (200-400nt, error rate 0.5-2.5%)
- Assembly turns reads into contigs and/or scaffolds
- Algorithms fit sequencing technologies
- Three general approaches
  - Overlap-Layout-Concensus (Long reads Celera 2000, Newbler 2005)
  - Greedy (Short reads, SSAKE 2007)
  - de Bruijn Graph (Short reads, Velvet 2008)

FYI, this man is not my grandpa

Introduction on Assembly

De Bruijn Graph



Introduction on Assembly

- Choosing the right K...
  - larger K more specific less coverage (span repeats, regions occurring twice, less connections in the graph)
  - Smaller K more sensitive more coverage (more connections in the graph)
  - Ideally combine both
- De Bruijn Graph potential information available
  - Overlap between kmers
  - Kmer coverage (how often does a kmer occur)
  - Read that created the kmer (choose between paths)
  - Insert size distribution between pairs (if paired reads were used)
- Programs differ in
  1) how the graph is stored
  2) how the graph is traversed

Introduction on Assembly

- Metagenomics uses DNA from environmental sample
    - Not all microbes can be cultured in the lab
    - Study microbial communities in their natural environment
- Metagenomic assembly more difficult than single genome assembly
    - Number of genomes unknown (maybe a rough idea)
    - Coverage of genomes differs (different abundances of genomes)
    - Closely related strains complicate the graph (in de Bruijn: anything that shares stretches of DNA longer than K)

Benchmark Method

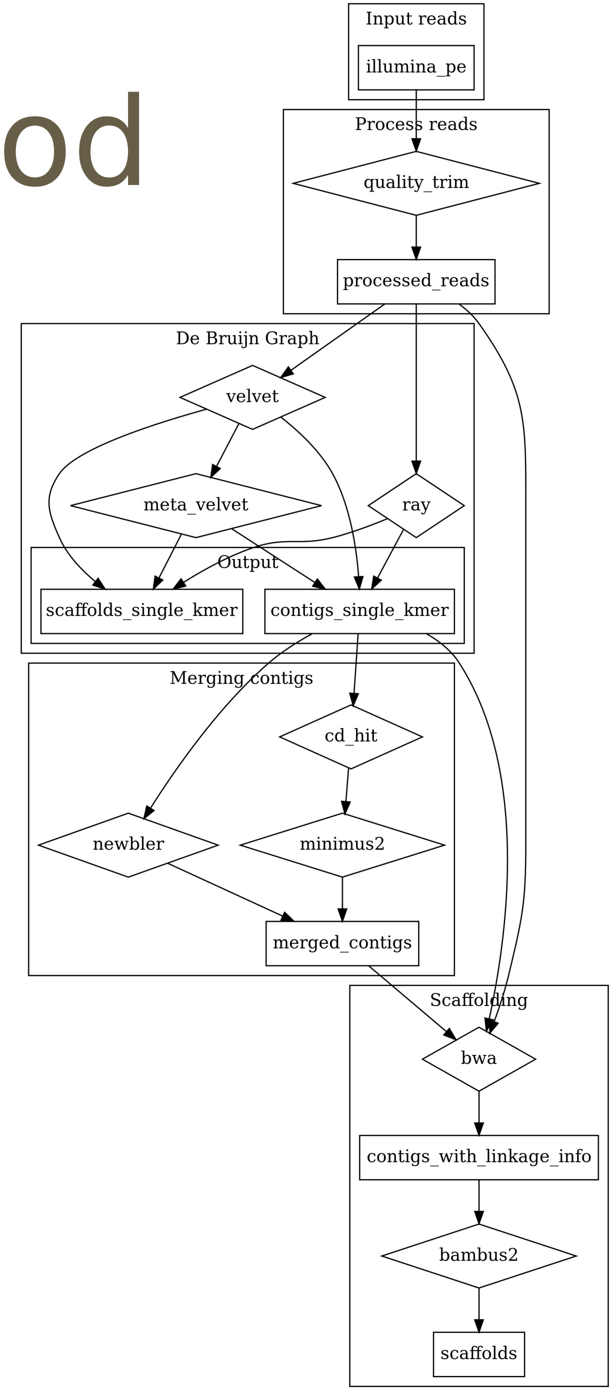
- Use an *in vitro* simulated metagenome with known species
  - 59 species, total size 195Mb
  - Two abundance distributions of genomes
    - Even, approximately the same genome copy numbers

    - Uneven, log-normal distribution of phyla similar to soil

    - Assemblies can be validated by aligning them against the reference genomes (we used nucmer for this)
- Use Illumina paired end reads since that is currently one of the most popular sequencing techniques for metagenomic samples

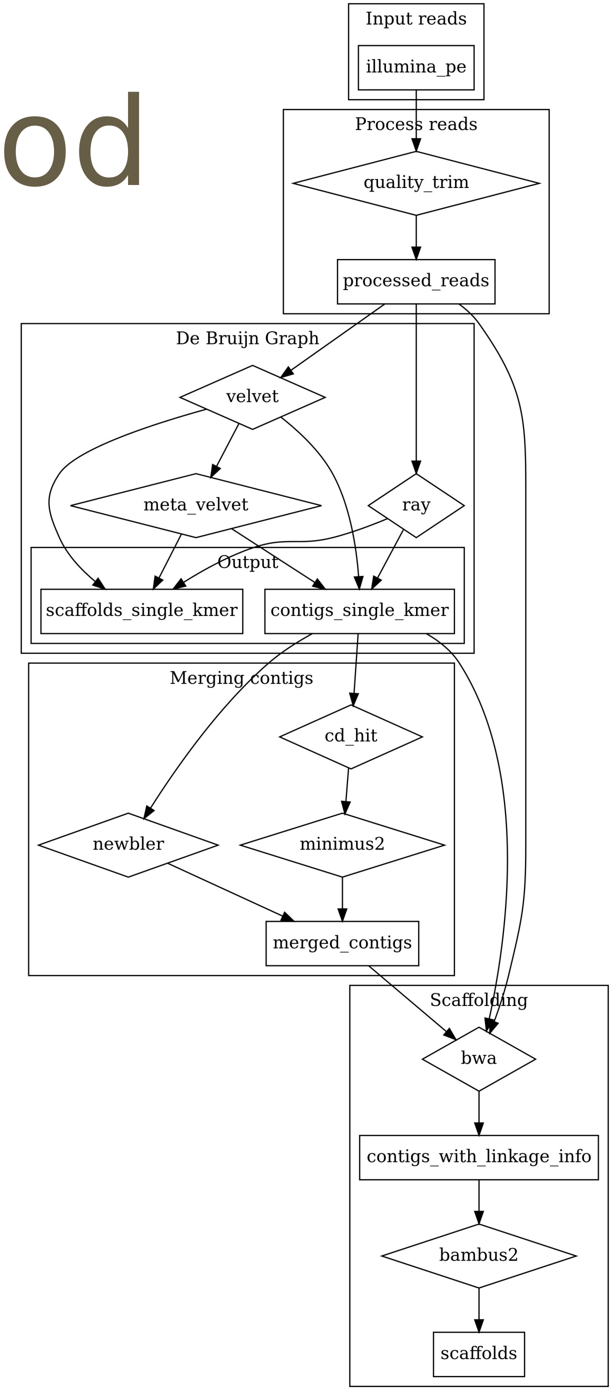
Benchmark Method

- Enormous amount of assembly strategies possible
  - Select a number of assembly strategies to test



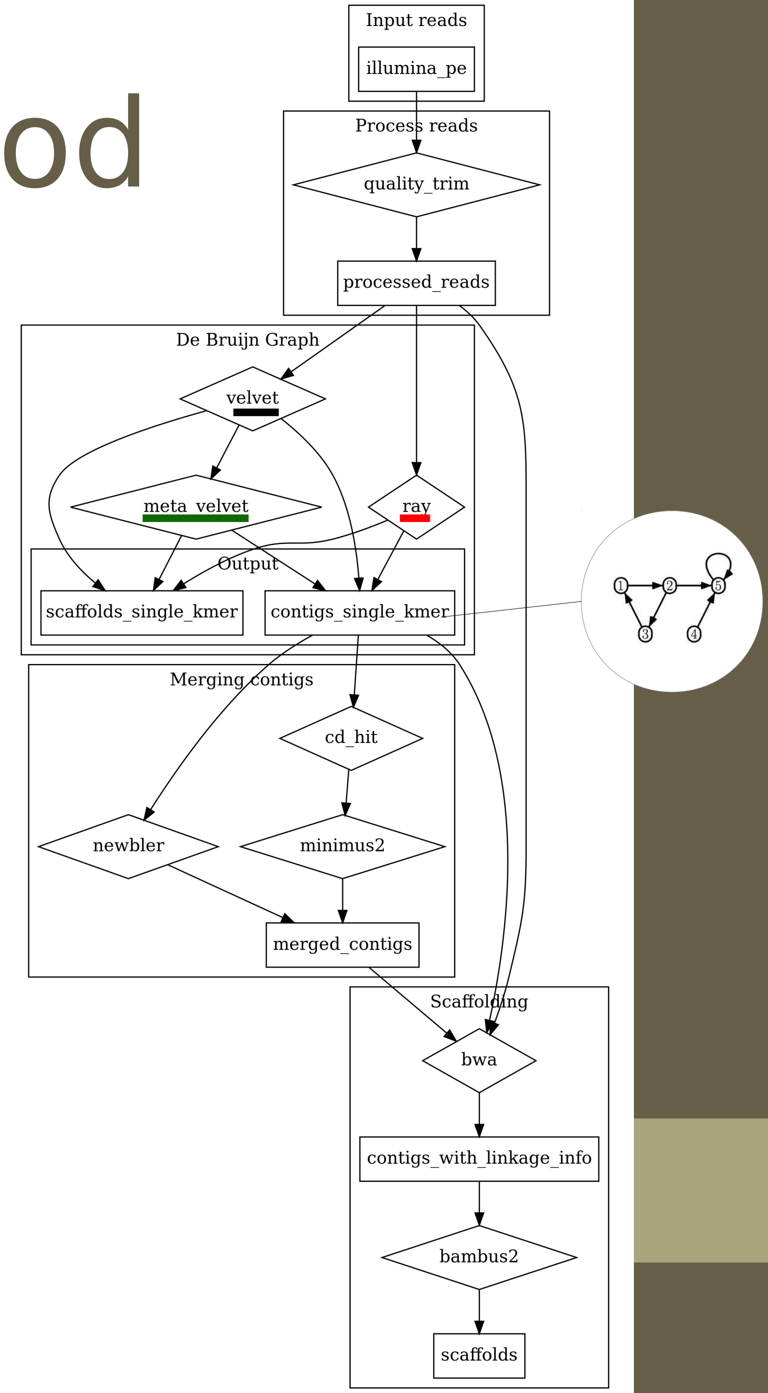
Benchmark Method

- Enormous amount of assembly strategies possible
  - Select a number of assembly strategies to test



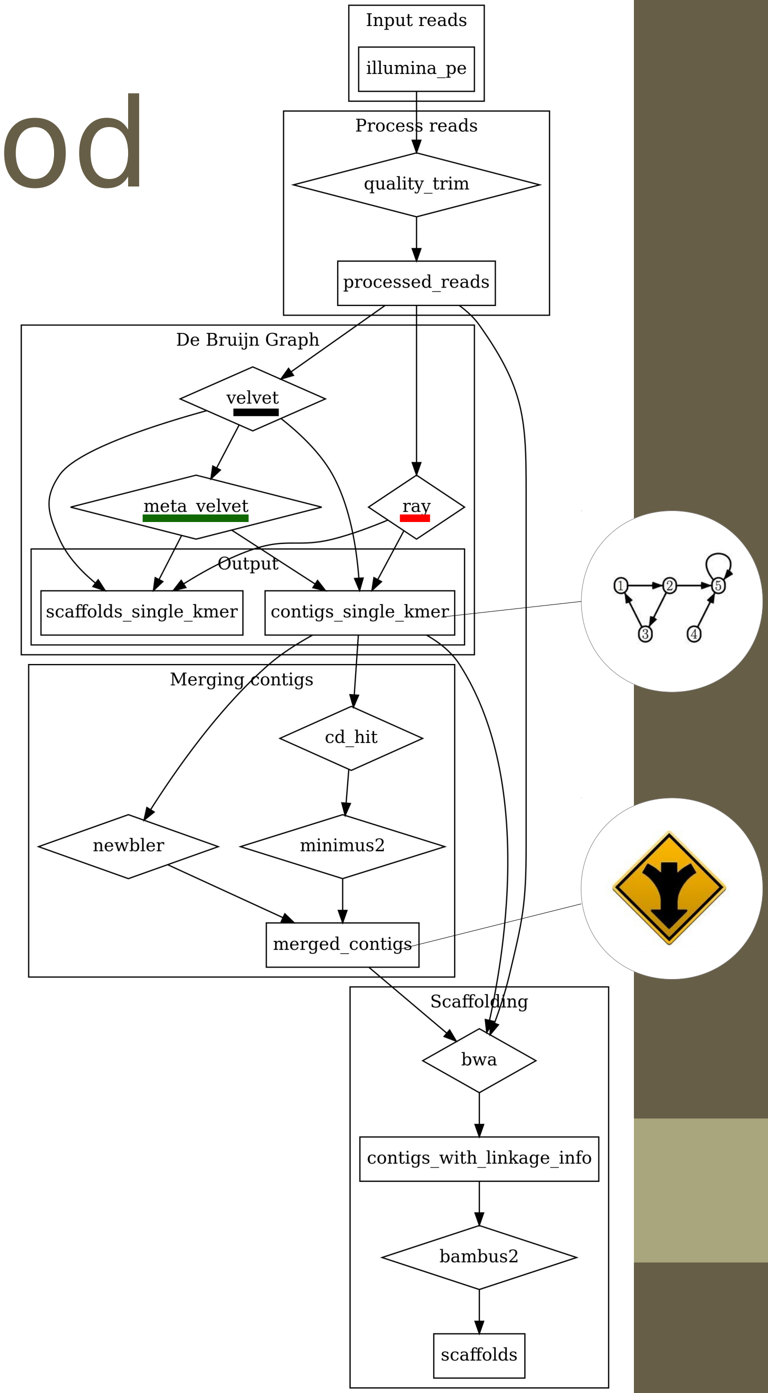
Benchmark Method

- Enormous amount of assembly strategies possible
  - Select a number of assembly strategies to test
- Contiging with
  - Velvet
  - Meta-Velvet
  - Ray



Benchmark Method

- Enormous amount of assembly strategies possible
    - Select a number of assembly strategies to test
- Contiging with
    - Velvet
    - Meta-Velvet
    - Ray
- Merging with
    - Newbler
    - Minimus2



Input reads

illumina_pe

Process reads

quality_trim

processed_reads

De Bruijn Graph

velvet

meta_velvet

ray

Output

scaffolds_single_kmer

contigs_single_kmer

Merging contigs

cd_hit

newbler

minimus2

merged_contigs

Scaffolding

bwa

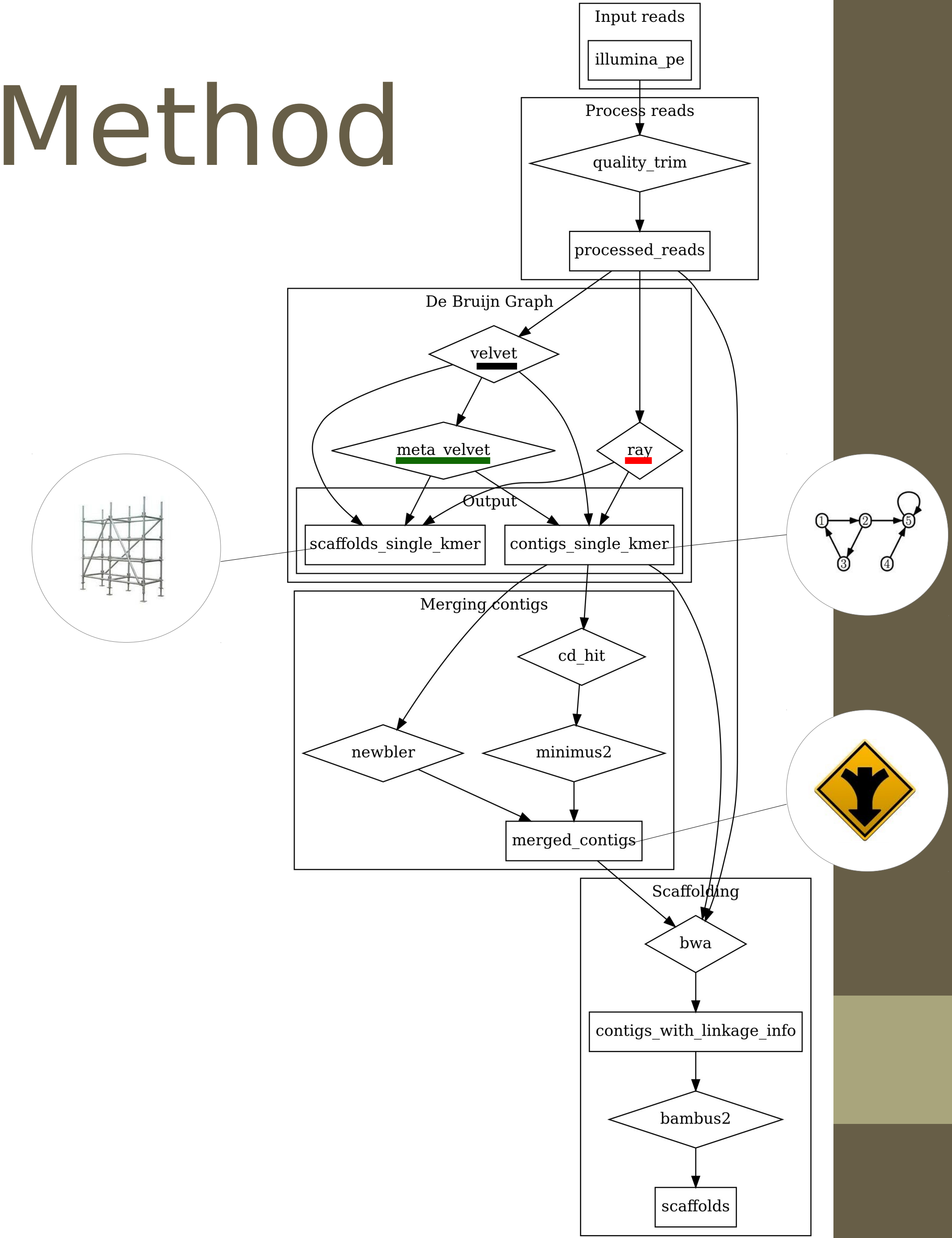
contigs_with_linkage_info

bambus2

scaffolds

Benchmark Method

- Enormous amount of assembly strategies possible
    - Select a number of assembly strategies to test
- Contiging with
    - Velvet
    - Meta-Velvet
    - Ray
- Merging with
    - Newbler
    - Minimus2
- Scaffolding with
    - Velvet
    - Meta-Velvet
    - Ray



Input reads

illumina_pe

Process reads

quality_trim

processed_reads

De Bruijn Graph

velvet

meta_velvet

ray

Output

scaffolds_single_kmer

contigs_single_kmer

Merging contigs

cd_hit

newbler

minimus2

merged_contigs

Scaffolding

bwa

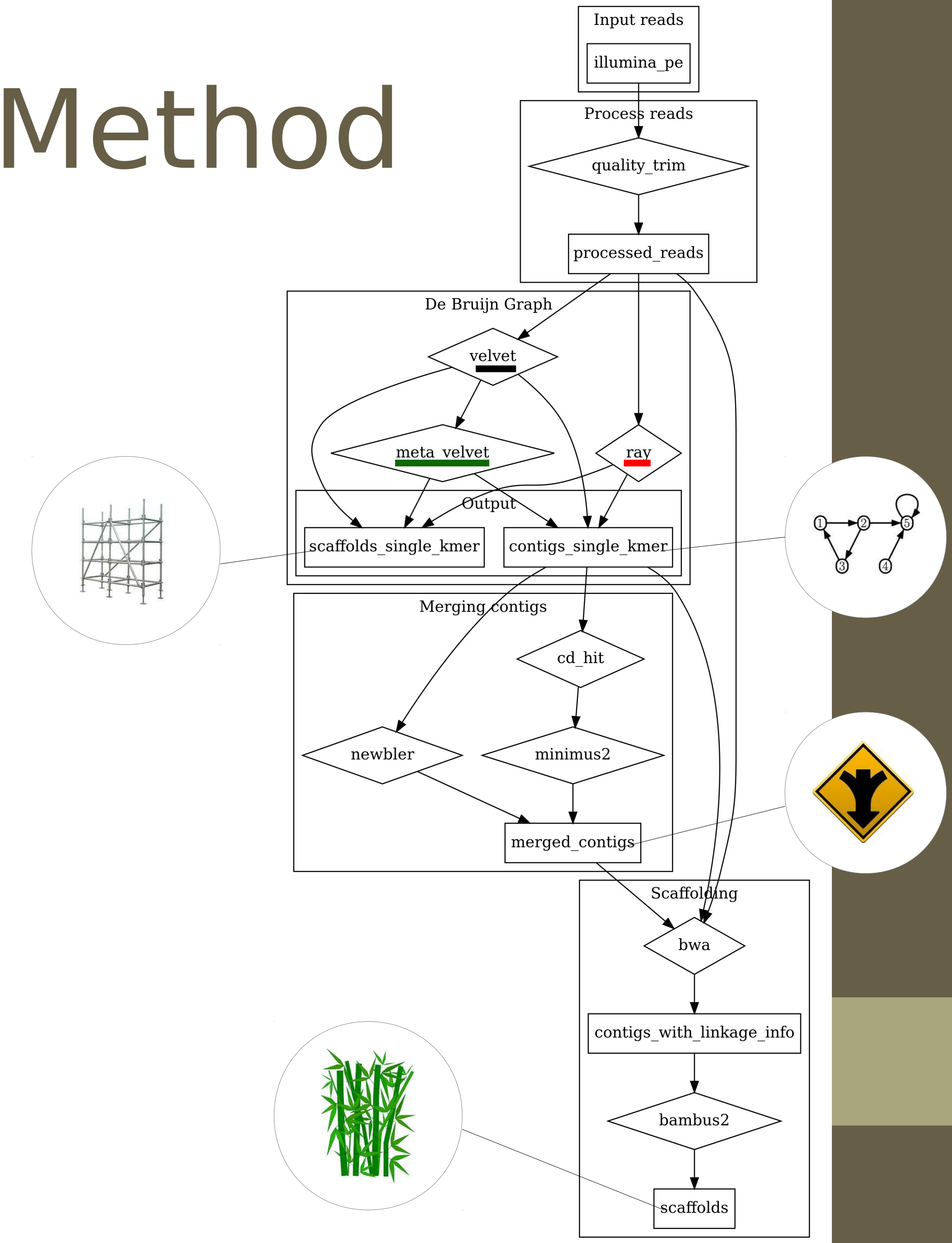
contigs_with_linkage_info

bambus2

scaffolds

Benchmark Method

- Enormous amount of assembly strategies possible
  - Select a number of assembly strategies to test
- Contiging with
  - Velvet
  - Meta-Velvet
  - Ray
- Merging with
  - Newbler
  - Minimus2
- Scaffolding with
  - Velvet
  - Meta-Velvet
  - Ray
  - Bambus2



Benchmark Method

- Enormous amount of assembly strategies possible
  - Select a number of assembly strategies to test
- Contiging with
  - Velvet
  - Meta-Velvet
  - Ray

= 3 strategies

- Merging with
  - Newbler
  - Minimus2

= 2 x 3 stragies = 6 strategies

- Scaffolding with
  - Velvet
  - Meta-Velvet
  - Ray
  - Bambus2

= 3 strategies

= (6 merge + 3 contig) strategies = 9 strategies

Total of 21 strategies

Benchmark Method

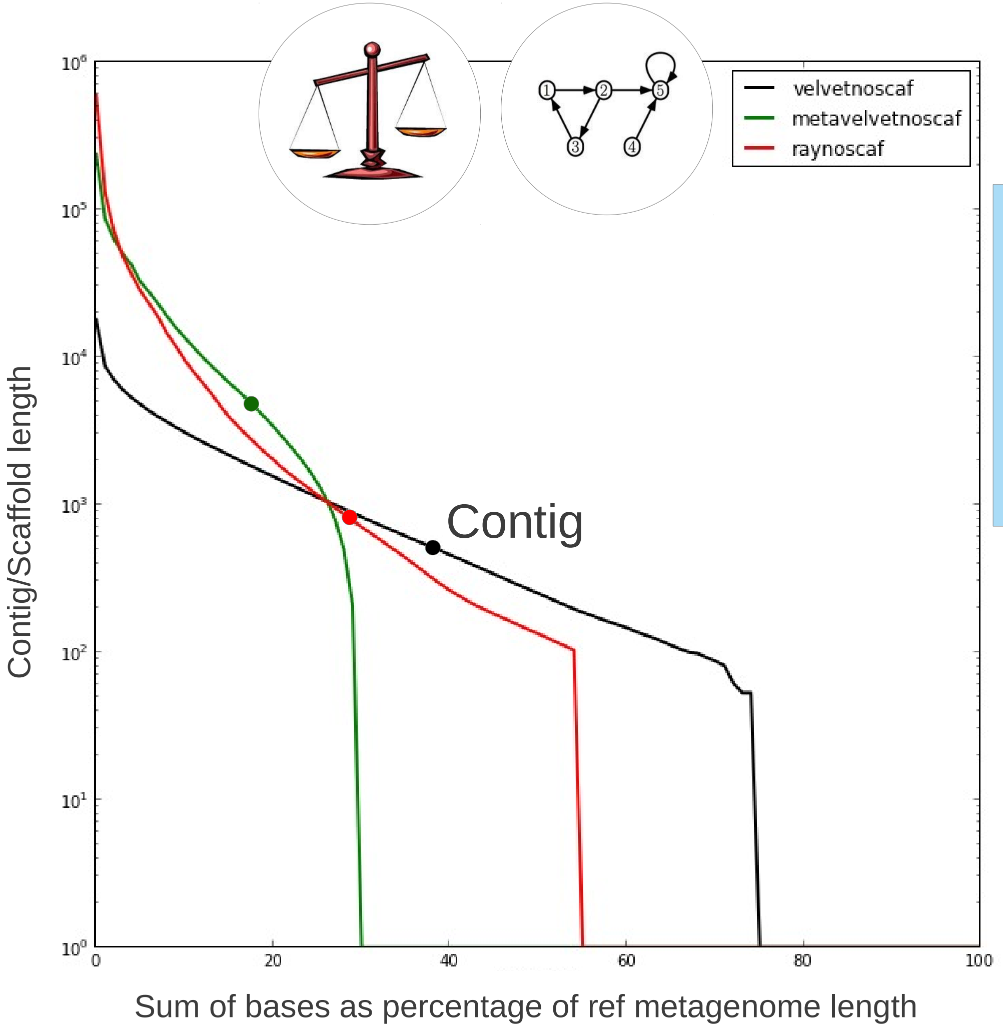
- How to compare all these different strategies?
- Validation of metagenomic assembly often focuses on one or more of the following points:
  1) contig/scaffold length distribution
  2) contig/scaffold coverage of the reference metagenome
  3) chimericity and erroneousness of the contigs/scaffolds
  4) functional annotation
  5) phylogenetic classification
- We focus on the first three since those also tend to improve 4 and 5 as shown by Mende et al (2012)
- Select winner in three categories

Benchmark Results

1) contig/scaffold length distribution
   - Popular statistic: N50 length (or L50)
     - Weighted median of contig lengths (contigs weighted by length)
     - 50% of all bases in the assembly are in contigs >= L50
     - Bigger is better
   - Problem when comparing between different assemblers: cut-off length matters and sum of bases between assemblies differs
     - Advantage for assemblers that only output long contigs

Benchmark Results

1) contig/scaffold length distribution



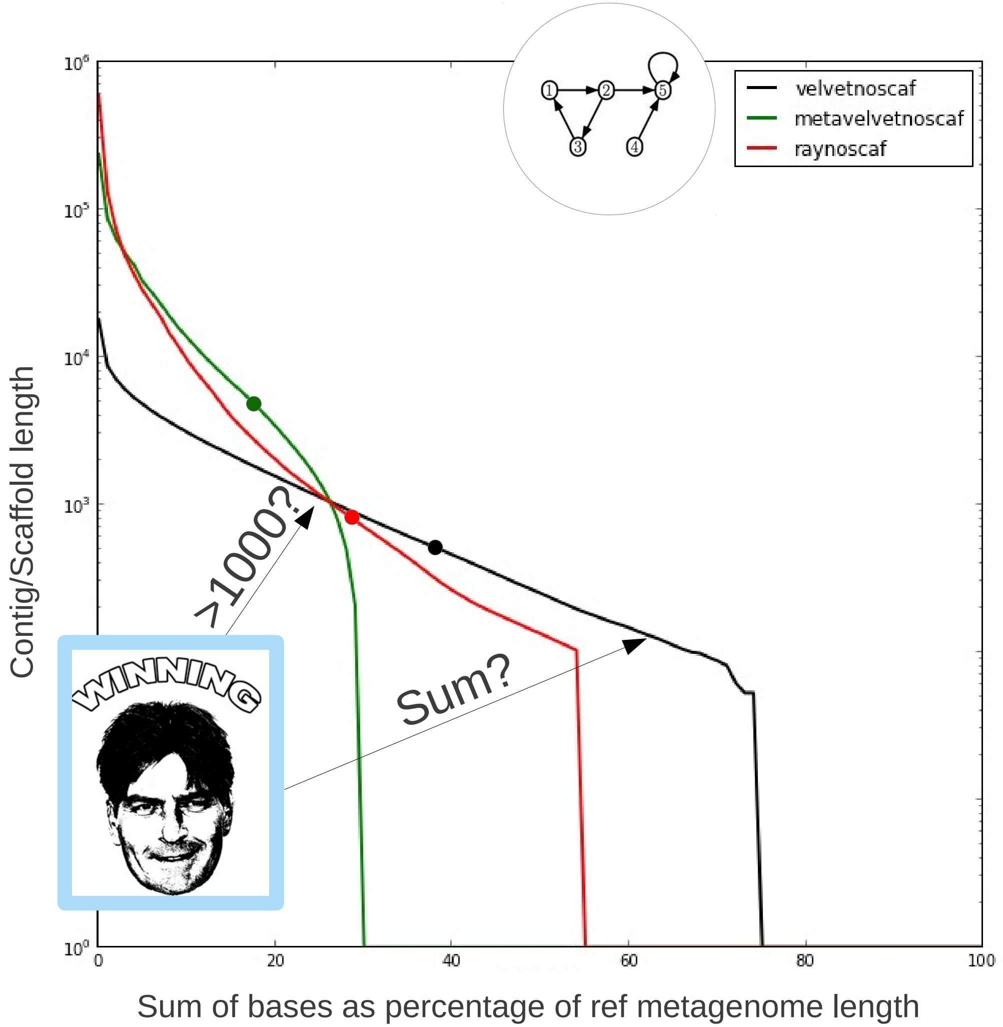
Contig

Shows best single kmer strategy per assembler based on L50

Dots are L50 values at cut-off 100

Benchmark Results

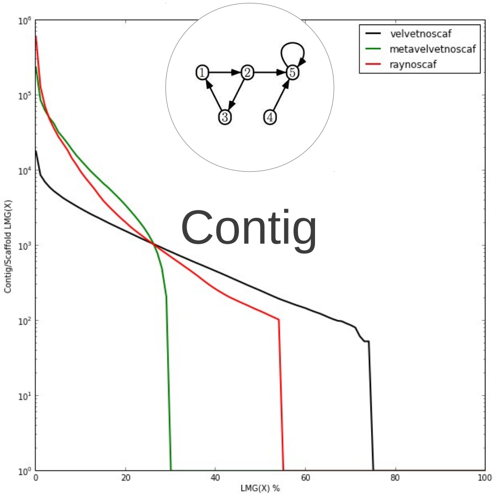
1) contig/scaffold length distribution

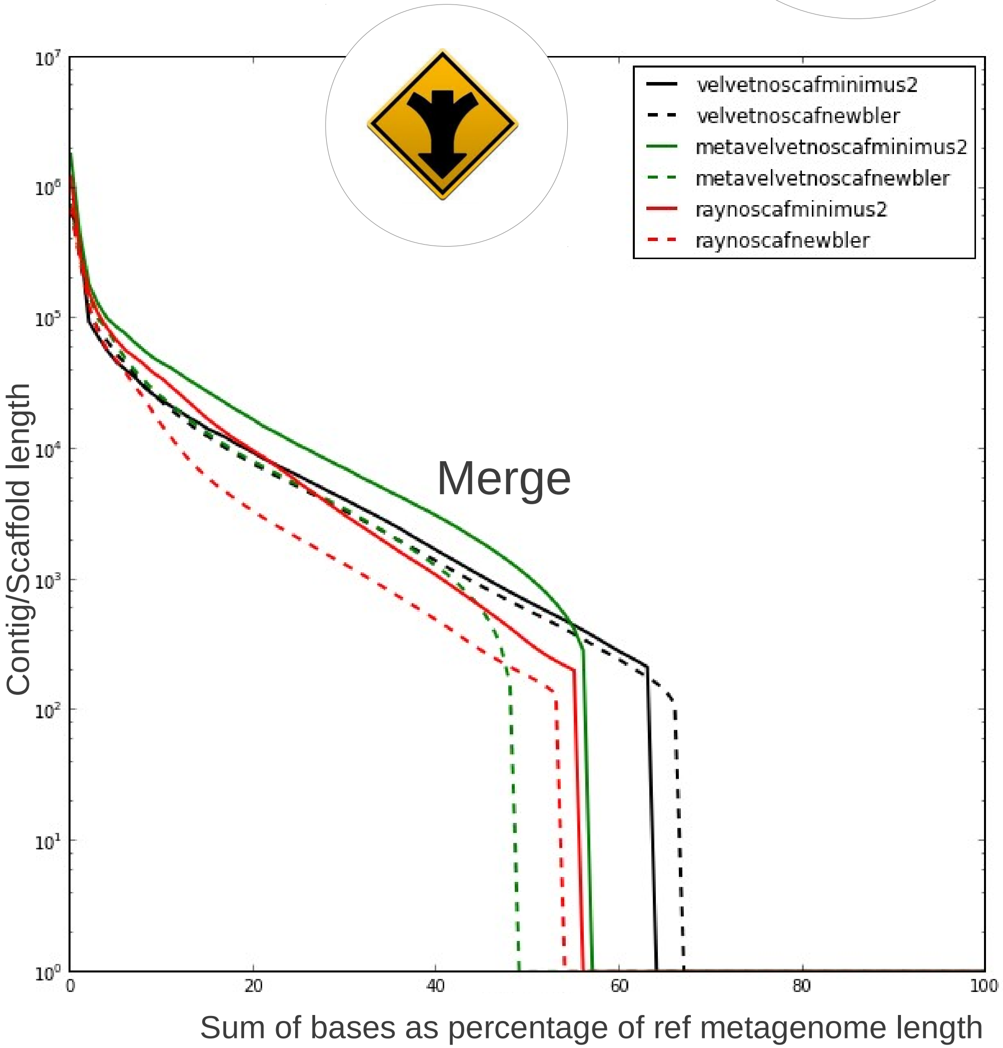


Difficult to say which is winning based on length distribution alone.

Benchmark Results

1) contig/scaffold length distribution



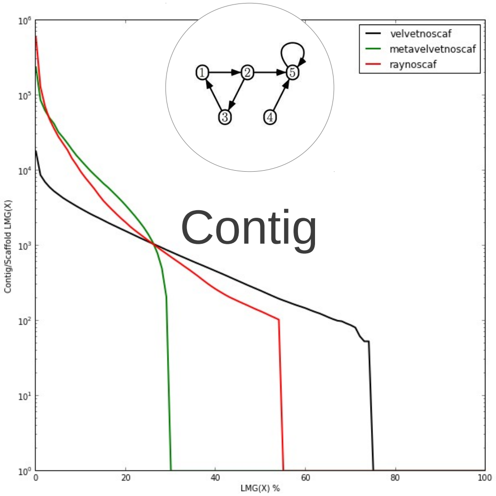


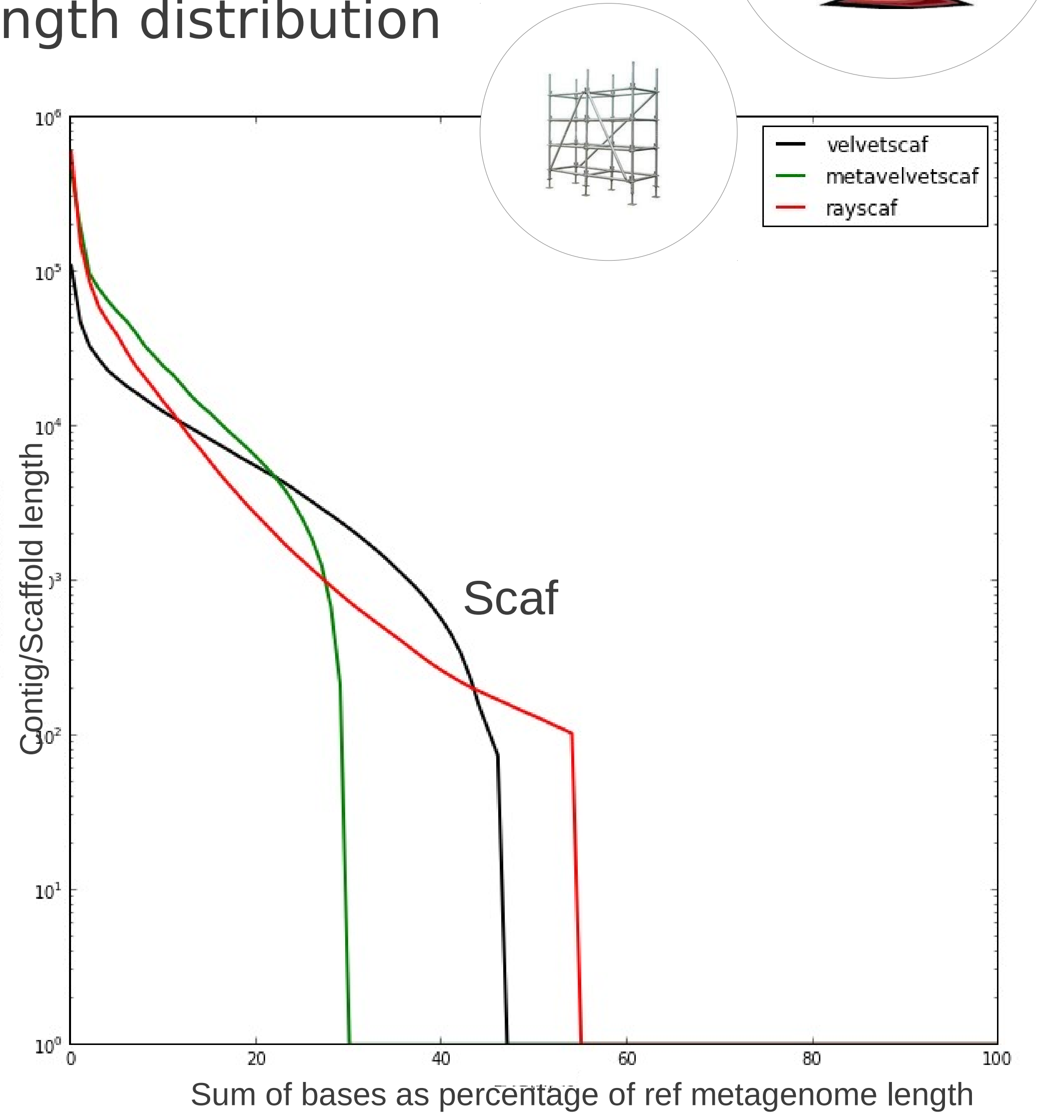
Merging increases lengths and makes length distributions more similar.

L50 before → after
V 665 → 3487 / 2553
M-V 6892 → 43685 / 45085
R 919 → 4076 / 1731

Benchmark Results

1) contig/scaffold length distribution





Scaffolding most notable with Velvet and Meta-Velvet

L50	before	→	after
V	665	→	4453
M-V	6892	→	12604
R	919	→	1002

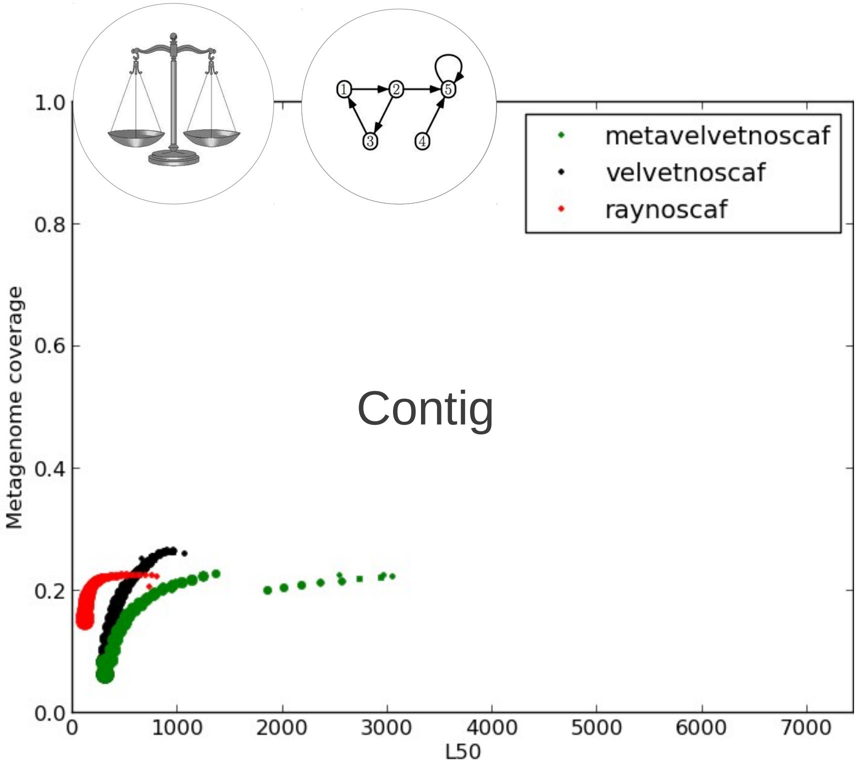
Benchmark Results

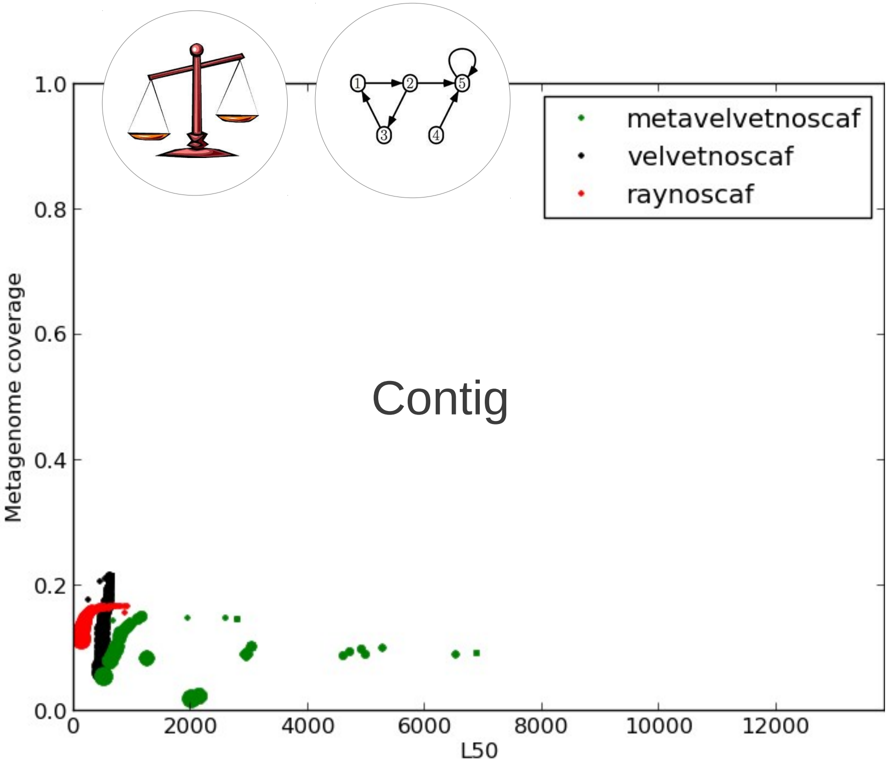
2) contig/scaffold coverage of the reference metagenome

- So the length distributions don't say a lot, how much of the reference metagenome is actually covered?

Benchmark Results

2) contig/scaffold coverage of the reference metagenome





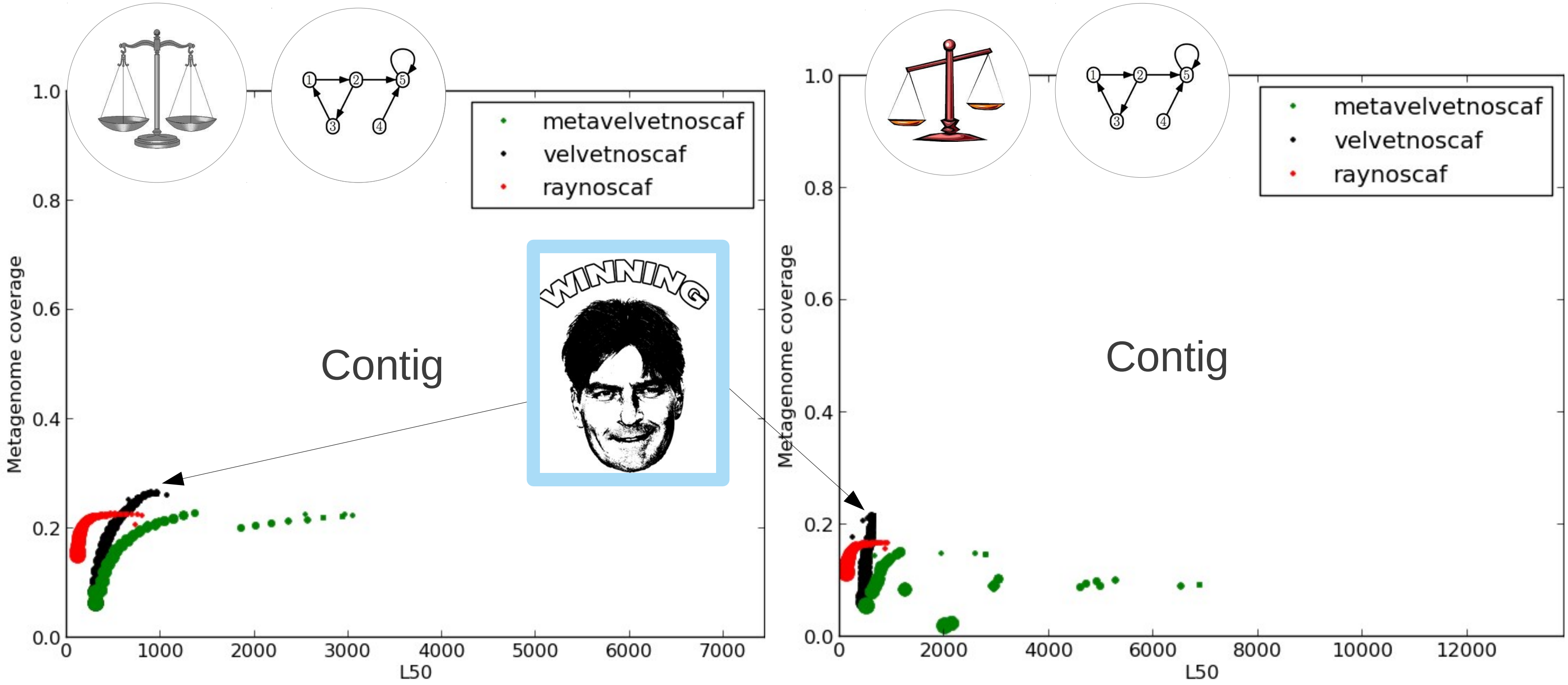
Metagenome coverage:
Number of non-overlapping bases covered by the assembly expressed as ratio of entire metagenome

Each dot is an assembly

Point size indicates kmer size (21-79)

Benchmark Results

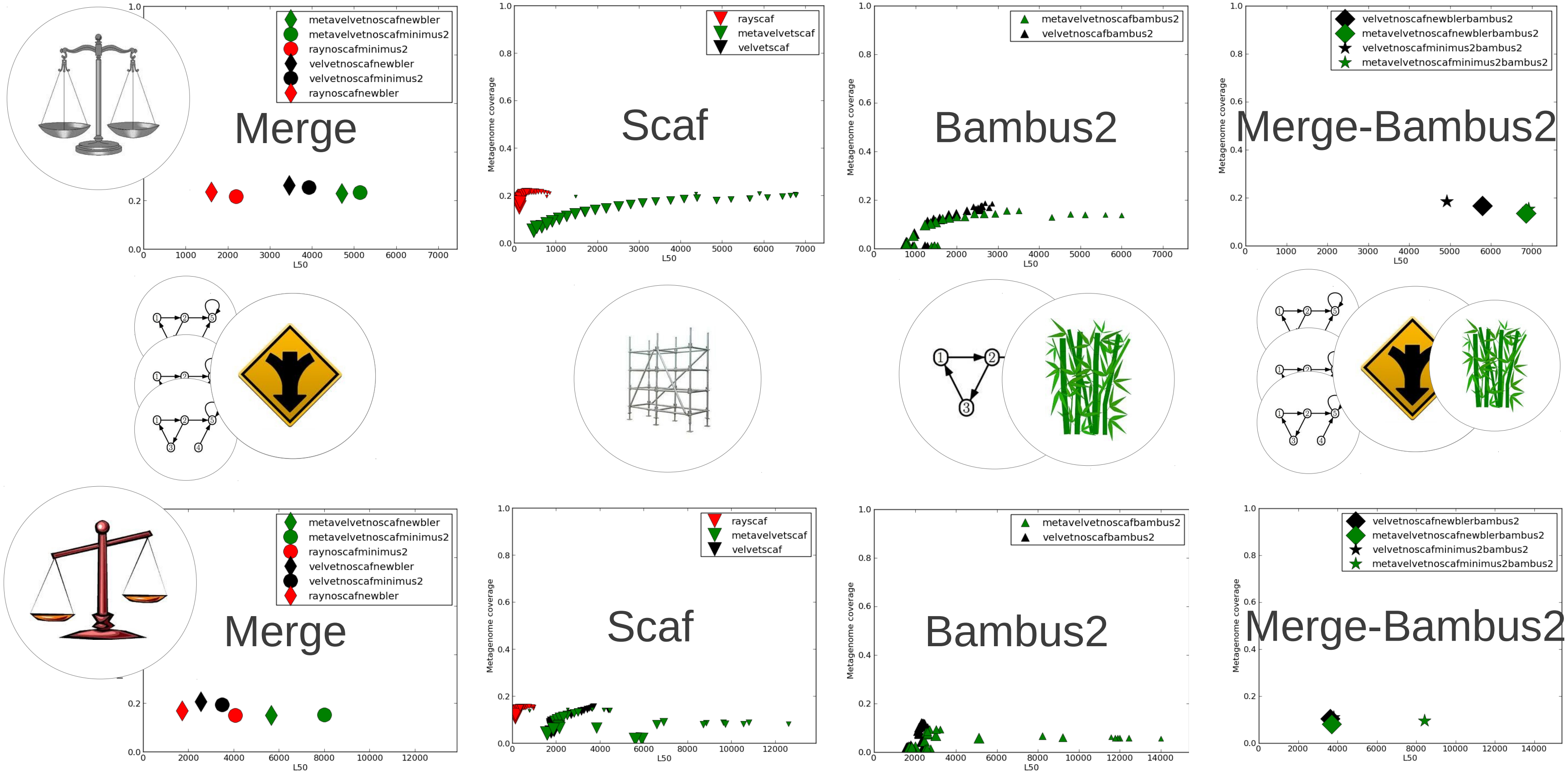
2) contig/scaffold coverage of the reference metagenome



Velvet does a good job covering the metagenome.

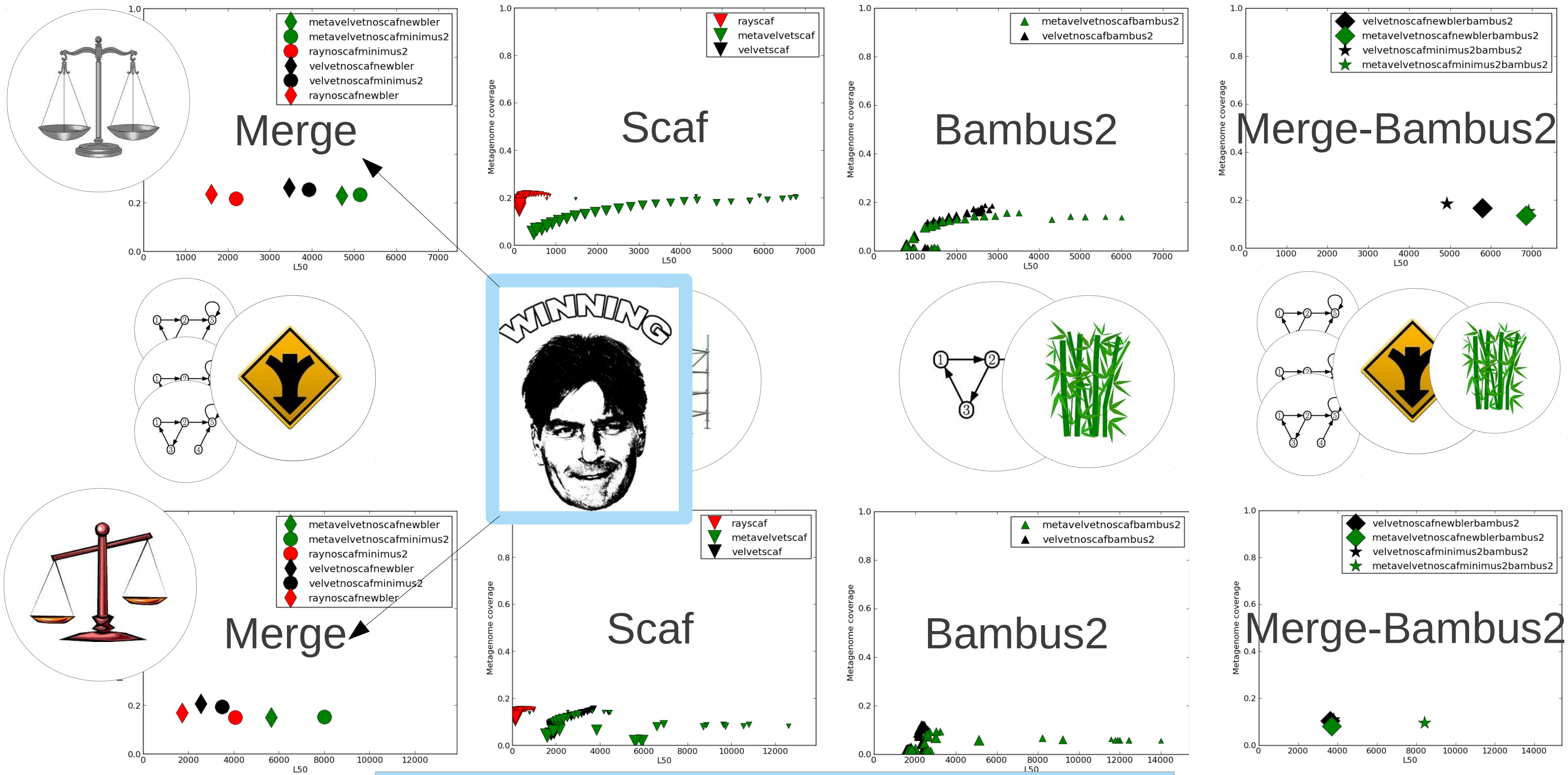
Benchmark Results

2) contig/scaffold coverage of the reference metagenome



Benchmark Results

2) contig/scaffold coverage of the reference metagenome



Merging increases length while keeping coverage high. Still original Velvet contigs win overal

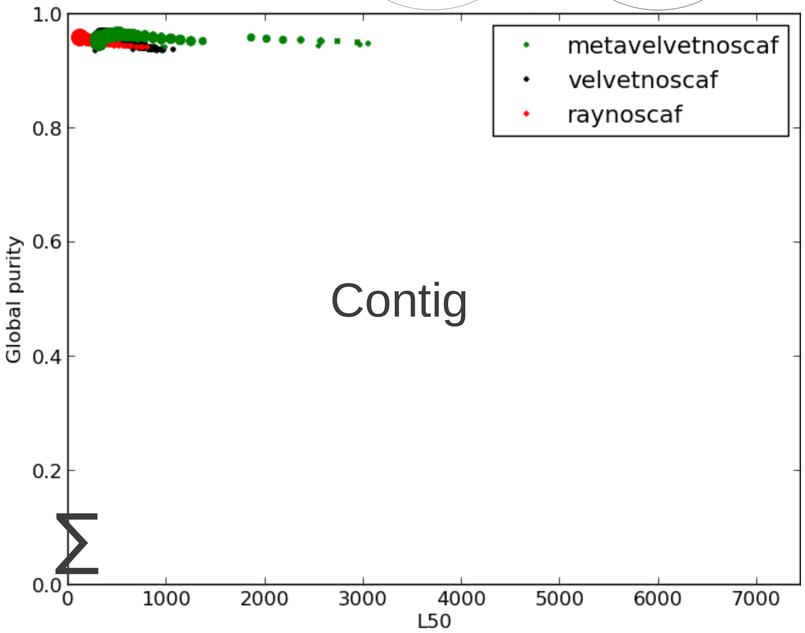
Benchmark Results

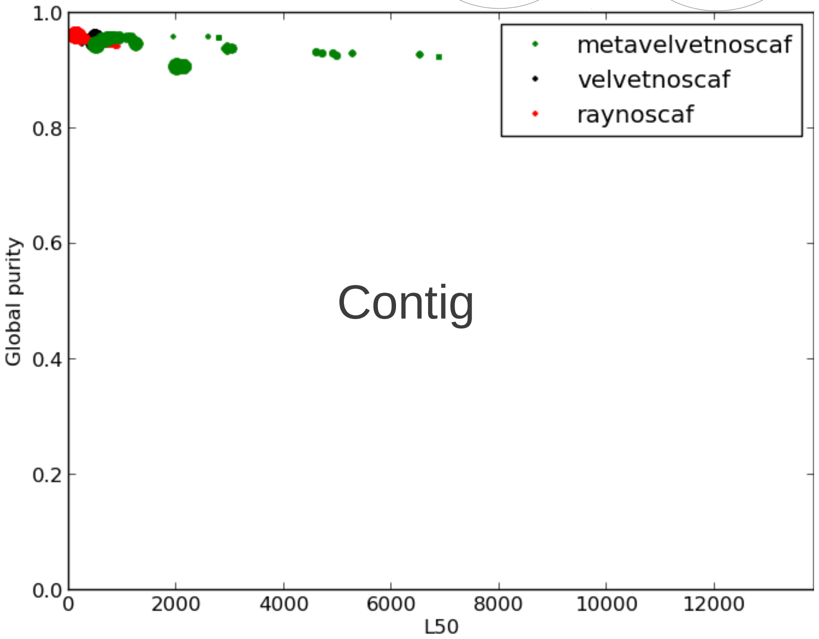
3) chimericity and erroneousness of the contigs/scaffolds

- If lengths increase, but coverage doesn't. Do we have duplicate contigs or erroneous contigs?

Benchmark Results

3) chimericity and erroneousness of the contigs/scaffolds





Global purity:

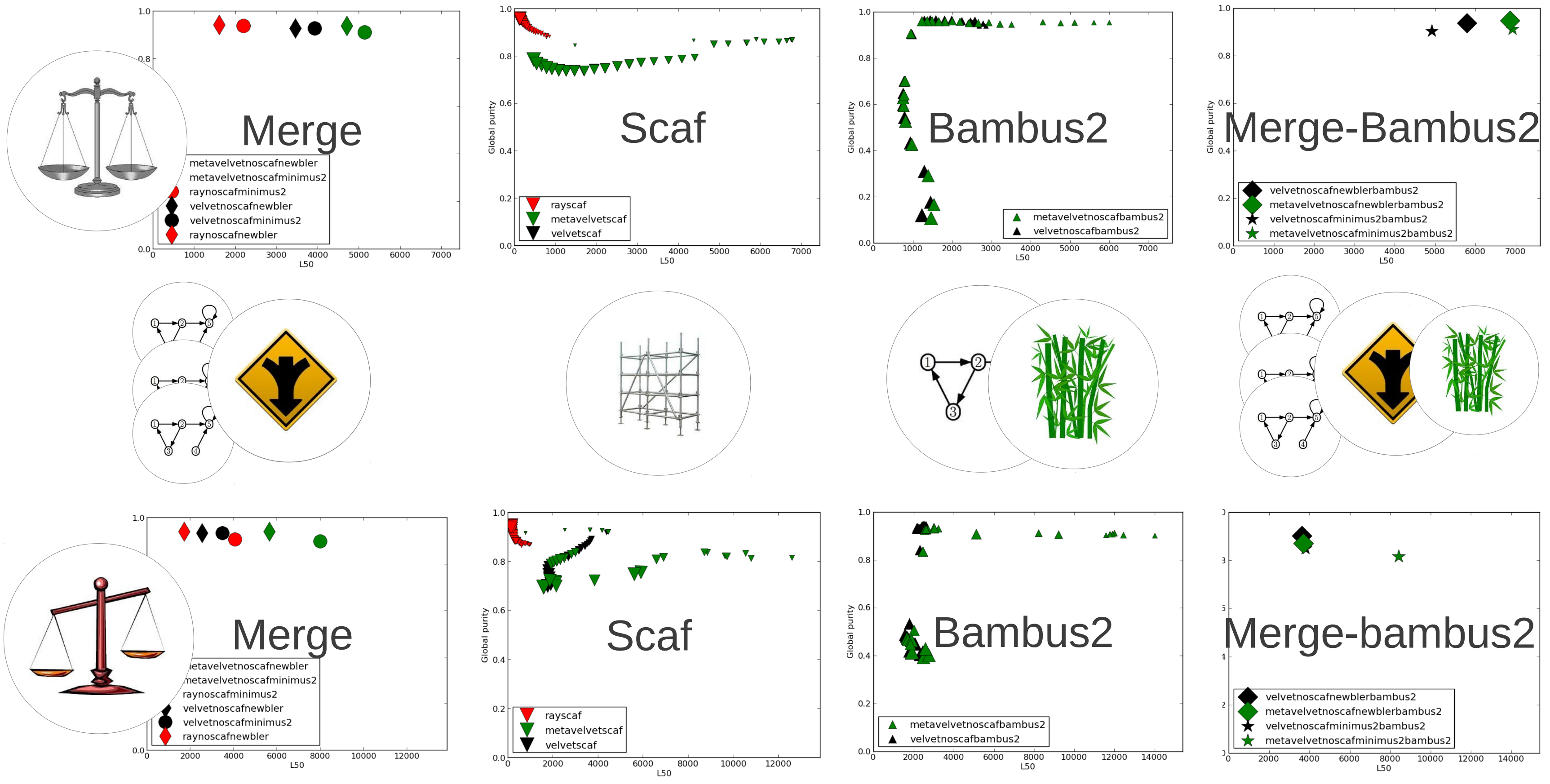
$$\frac{\sum_{\text{contigs}} \text{\# bases in best alignment contig}}{\text{total nr of bases assembly}}$$

Each dot is an assembly

Point size indicates kmer size (21-79)

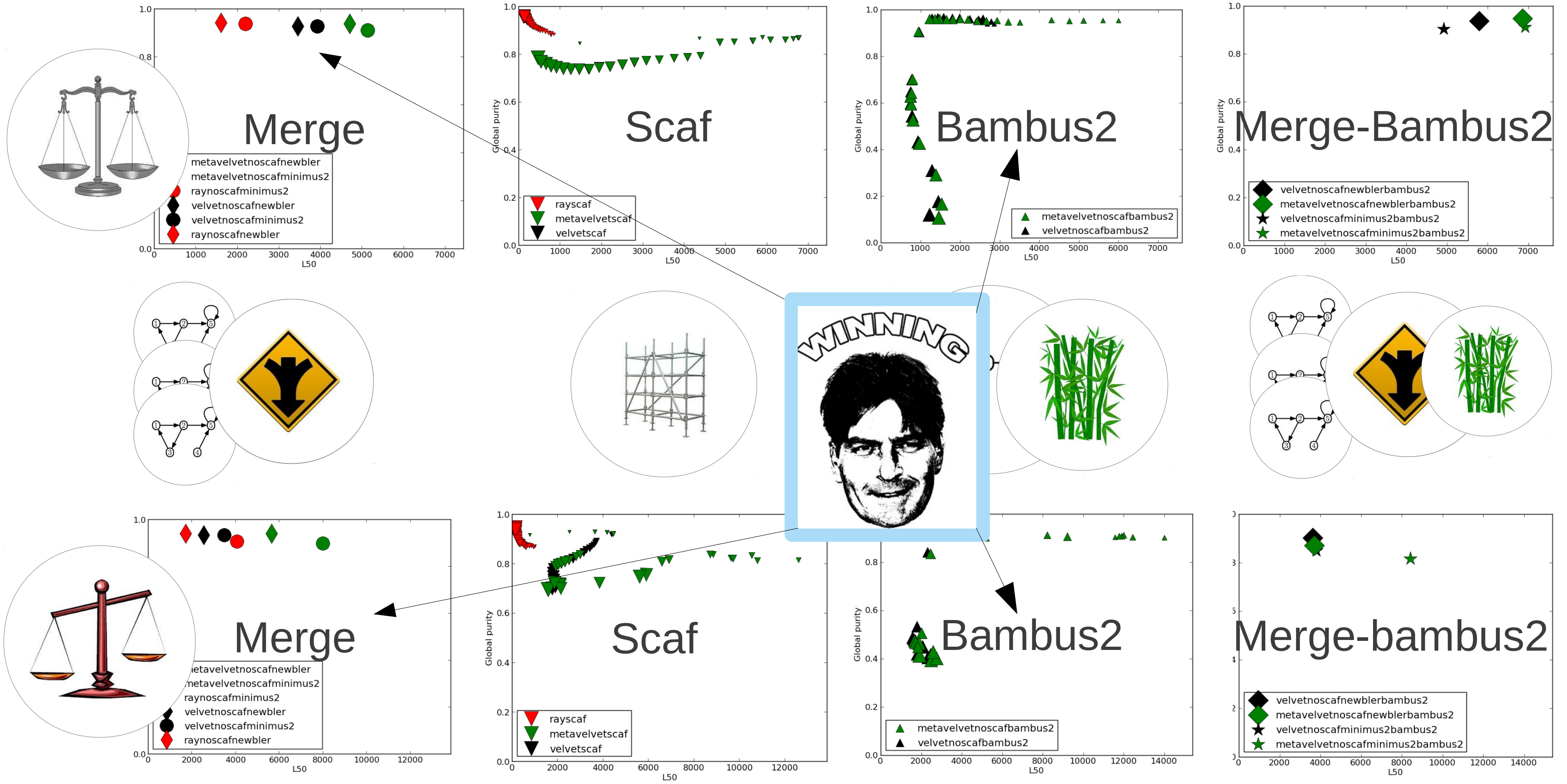
Benchmark Results

3) chimericity and erroneousness of the contigs/scaffolds



Benchmark Results

3) chimericity and erroneousness of the contigs/scaffolds



Ergo, Merging and Bambus2 both perform well purity-wise

Conclusions

- Difficult to name one true winner. Depends on what the post-processing will be like
  - Velvet provides good coverage of the metagenome at cost of length
  - Meta-Velvet improves length at a substantial cost of purity and coverage of the meta-genome
  - Ray seems middle ground between Velvet and Meta-Velvet
  - Merging improves lengths while keeping purity and coverage high
  - Internal scaffolding improves lengths at a cost of purity
  - Bambus2 scaffolding improves lengths but mostly by throwing away all short contigs (??)

Conclusions

- Recommendations
  - If interested in functional annotation: Ray or Velvet. Velvet preferred if annotation works on really short contigs 100-1000 (more coverage of metagenome).
  - If only interested in long contigs and less so in their purity, perhaps choose Meta-Velvet.
  - Merging seems to work quite well although you lose some metagenome coverage
  - Might want to avoid scaffolding since it doesn't work that well
    - bambus2 in particular loses a lot of metagenome coverage
    - Internal scaffolding works better than bambus2, but merging alone does a better job

Future work

- Try different mock communities, preferably higher covered samples
- See how results hold with functional annotation
- Compare reference-less validation against this validation
  - Compare for instance with Feature Response Curve (Vezzi, 2012)
- Make the validation pipeline available as a web service
  - Give reference and assemblies, watch the performance of your assembly
  - Make your own assembly of existing reference and reads and compare against existing ones

Questions?