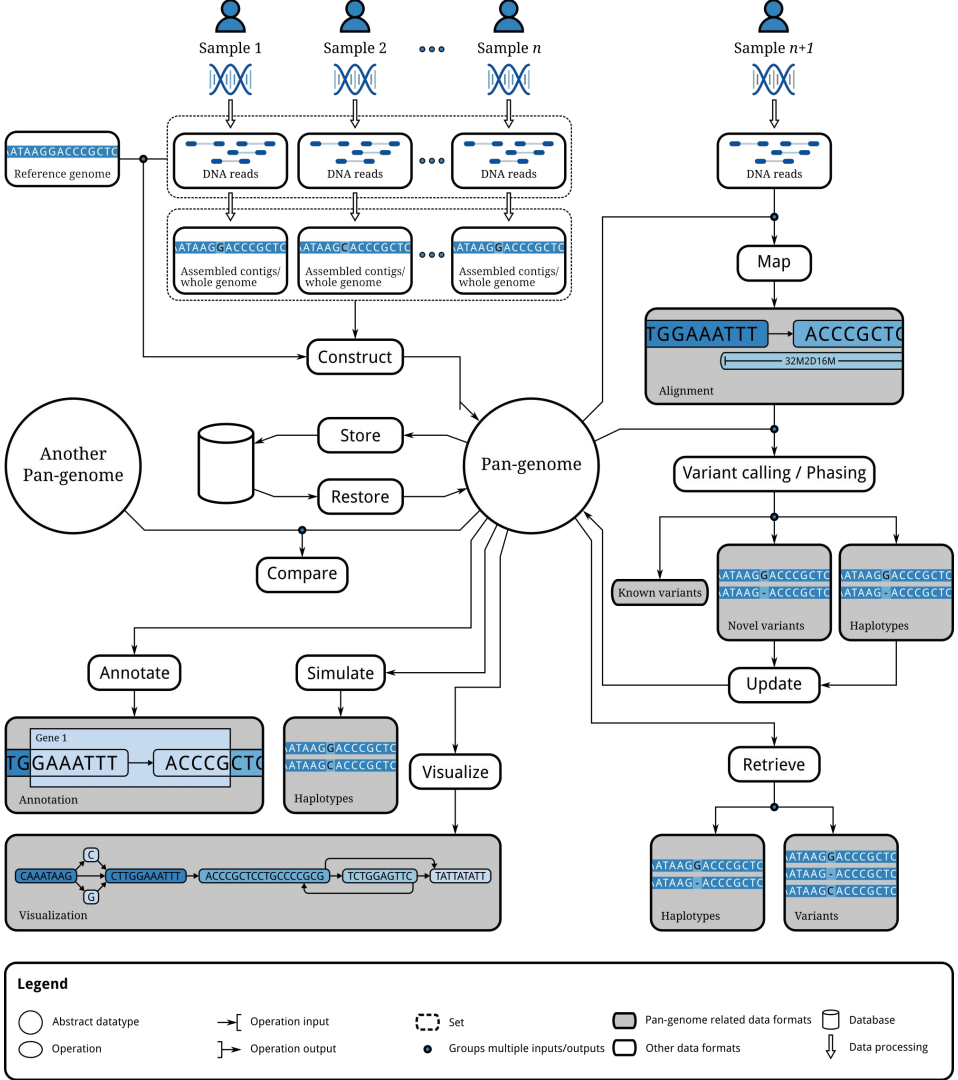
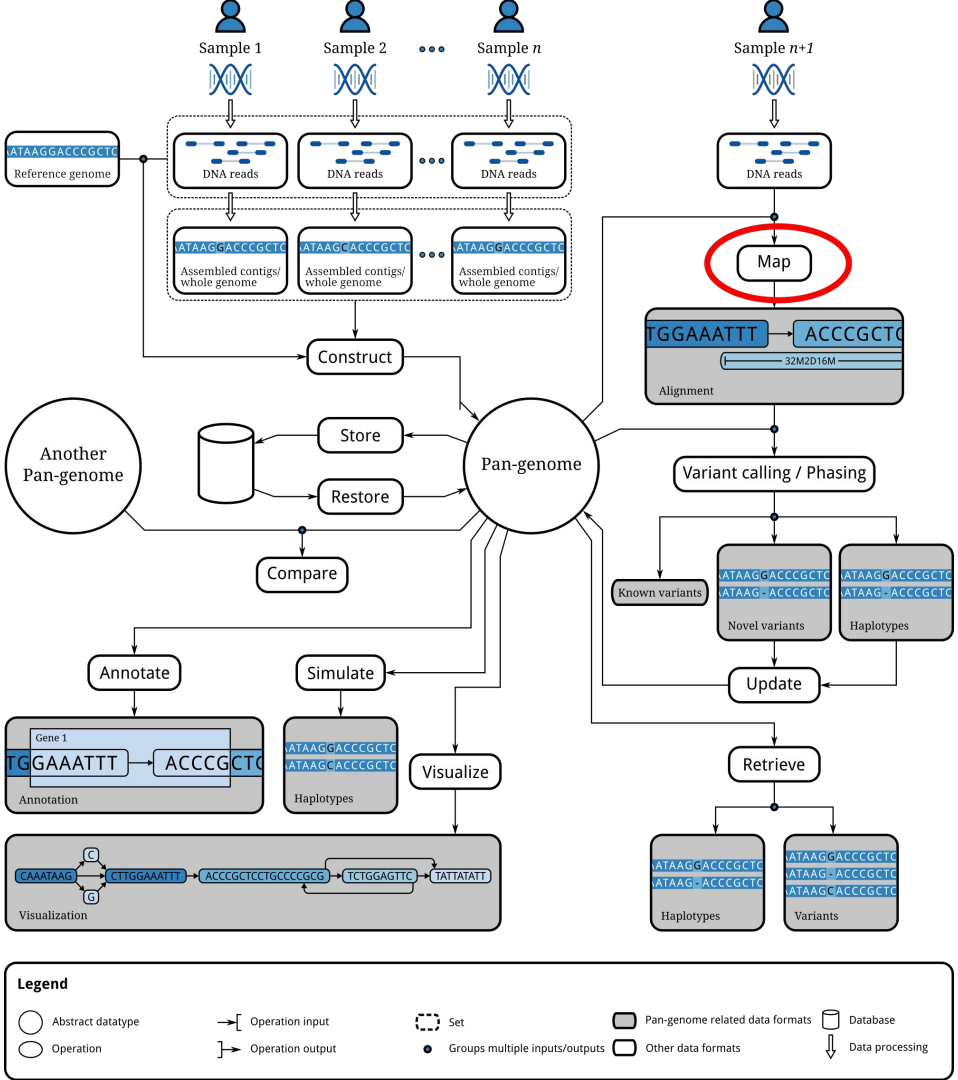
Computational Pangenomics #CPANG19

Day 5 (September 13, 2018)

Erik Garrison and Mikko Rautiainen


Sample 1
Sample 2
Sample n
Sample n+1
Reference genome
DNA reads
DNA reads
DNA reads
DNA reads
Assembled contigs/ whole genome
Assembled contigs/ whole genome
Assembled contigs/ whole genome
Construct
Map
TGGAAATTT
ACCCGCTC
32M2D16M
Alignment
Another Pan-genome
Store
Restore
Pan-genome
Variant calling / Phasing
Compare
Known variants
Novel variants
Haplotypes
Update
Annotate
Simulate
Gene 1
Annotation
Haplotypes
Visualize
Retrieve
CAAATAAG
CTTGGAAATTT
ACCCGCTCCTGCCCCGCG
TCTGGAGTTC
TATTATATT
Visualization
Haplotypes
Variants
Legend
Abstract datatype
Operation
Operation input
Operation output
Set
Groups multiple inputs/outputs
Pan-genome related data formats
Other data formats
Database
Data processing


Sample 1
Sample 2
Sample n
Sample n+1
Reference genome
DNA reads
DNA reads
DNA reads
DNA reads
Assembled contigs/ whole genome
Assembled contigs/ whole genome
Assembled contigs/ whole genome
Map
Construct
TGGAAATTT
ACCCGCTC
32M2D16M
Alignment
Another Pan-genome
Store
Restore
Pan-genome
Variant calling / Phasing
Compare
Known variants
Novel variants
Haplotypes
Update
Annotate
Simulate
Gene 1
Annotation
Haplotypes
Visualize
Retrieve
CAAATAAG
CTTGGAAATTT
ACCCGCTCCTGCCCCGCG
TCTGGAGTTC
TATTATATT
Visualization
Haplotypes
Variants
Legend
Abstract datatype
Operation
Operation input
Operation output
Set
Groups multiple inputs/outputs
Pan-genome related data formats
Other data formats
Database
Data processing

Long reads

So far the workshop has focused on short reads

What about long reads?

Long reads

So far the workshop has focused on short reads

What about long reads?

High error rates (10% - 20%)

Longer length (10kbp - 100kbp)

Long read alignment

vg

SPAligner

minigraph (proof of concept)

GraphAligner

Long read alignment

vg

SPAligner

minigraph (proof of concept)

GraphAligner <- Focus on this today

GraphAligner

Reads:

- Long read aligner
- Not for short reads
- Handles high error rates
- File formats: fasta, fastq (also gzip-compressed)

GraphAligner

Graphs:

- Arbitrary graph topologies
- Variation graphs
- de Bruijn graphs
- File formats: vg and gfa

GraphAligner

Output:

.gam: interoperable with vg

.json: text equivalent to .gam

Corrected reads: replace the read with the path of the alignment

Installation

Bioconda

conda install -c bioconda graphaligner

Running

GraphAligner -g graph.gfa -f reads.fa -a aln.gam -t 8

Some theory

Seed-and-extend aligner

Seed hits: matches between the read and the graph

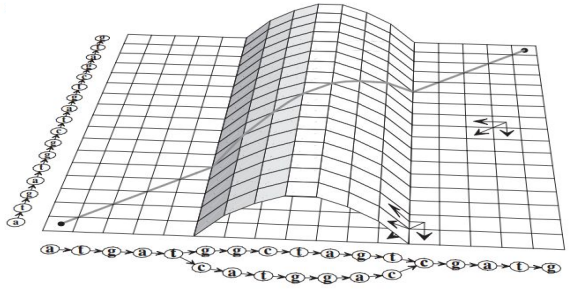
Finding seed hits is necessary for getting proper alignments

Different seeding modes (a few slides later)

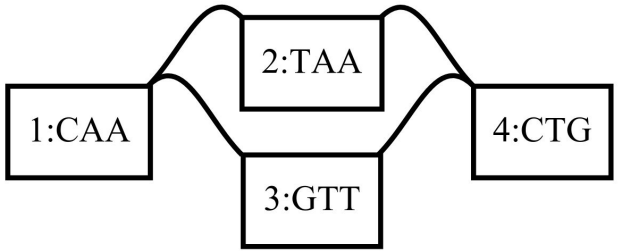
Local alignment to the graph

query:
CAAATTCT

	T	A	A
C	0	0	0
A	0	2	2
A	3	2	4
A	4	5	4
T	6	3	3
T	4	4	1
C	2	2	2
T	2	0	0



	C	A	A
C	**2**	0	0
A	0	**4**	2
A	0	2	**6**
A	0	2	4
T	0	0	2
T	0	0	1
C	2	0	0
T	0	0	0



	C	T	G
C	2	0	0
A	0	0	0
A	1	0	0
A	2	0	0
T	2	4	1
T	5	4	3
C	**10**	7	6
T	7	**12**	9

	G	T	T
C	0	0	0
A	0	0	0
A	3	2	1
A	**4**	1	0
T	2	**6**	3
T	0	4	**8**
C	0	2	5
T	0	2	4

1. fill the score matrixes
2. find the maximum score
3. trace back for alignment

scores:
match = 2
mismatch = 2
gap_open = 3
gap_extension = 1

Some theory

Extension by banded dynamic programming

Banding: heuristic method to limit runtime, trade off between runtime and correctness

4	3	2	3	4		4	3	4	
	4	3	2	3	4		4	4	
		4	3	3	4	5			
		5	4	3	4	4	5		
			5	4	3	4	4	5	
				5	4	4	5	5	6

Some theory

Extension by banded dynamic programming

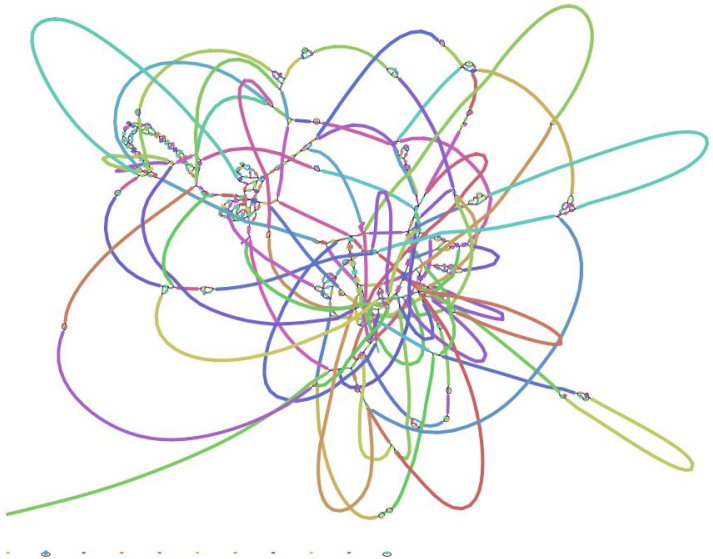
Banding: heuristic method to limit runtime, trade off between runtime and correctness

Band size: maximum allowed mismatches

- Relevant for the error rate of the reads

Tangle effort: maximum exploration in complex areas

- Relevant to the complexity of the graph



Some theory

GraphAligner estimates if the alignment is correct or wrong based on the alignment scores

Extension stops when the alignment is predicted to be wrong

Common reasons for stopping

- High error rate areas in the read
- Gaps in the graph (de novo assemblies, de Bruijn graphs)
- Tangles in the graph (de novo assemblies, de Bruijn graphs)

Output statistics

After running GraphAligner, it will output something like this:

Input reads: 15932 (19401507bp)
Seeds found: 946105
Seeds extended: 17312
Reads with a seed: 14538 (18563349bp)
Reads with an alignment: 14538
Alignments: 14685 (18539766bp)
End-to-end alignments: 14317 (18274862bp)

Parameters

Seeding modes

Minimizers

--seeds-minimizer-count 5

--seeds-minimizer-length 15

--seeds-minimizer-windowsize 40

--seeds-minimizer-chunksize 100

Parameters

Seeding modes

Maximal exact matches

--seeds-mem-count 20

--seeds-mxm-length 20

Parameters

Seeding modes

Maximal unique matches

--seeds-mum-count 20

--seeds-mxm-length 20

Parameters

Seeding modes

--try-all-seeds

Parameters

Extension

- Bandwidth: -b 5
- Tangle effort: -C 10000

Parameters

Miscellaneous

--all-alignments

--seeds-first-full-rows

Debugging alignments

Low number of reads with a seed

- Try different seeding
- Lower the match size
- Switch to MEM seeding
- For small graphs (bacterial): "--seeds-first-full-rows"

Debugging alignments

Low number of alignments (but reads have seeds)

- Try higher tangle effort
- For small graphs (bacterial): try "--seeds-first-full-rows"

Debugging alignments

Lower alignment identities than expected

Try higher tangle effort

For simple graphs (eg, variation graphs, bacterial de Bruijn graphs) try unlimited tangle effort (-1)

Try higher bandwidth

Debugging alignments

Make sure that the graph is also fine

Questions

How confident are you at aligning long reads to genome graphs?

How confident are you at debugging problems with long read alignment?

Error correction

Hybrid error correction pipeline

Input:

- Accurate short reads (Illumina)
- Long reads (PacBio, ONT)

Output:

- Accurate long reads

Error correction

Idea:

Build an assembly from short reads

Align the long reads to the assembly

Extract the aligned sequence from the assembly as the corrected read

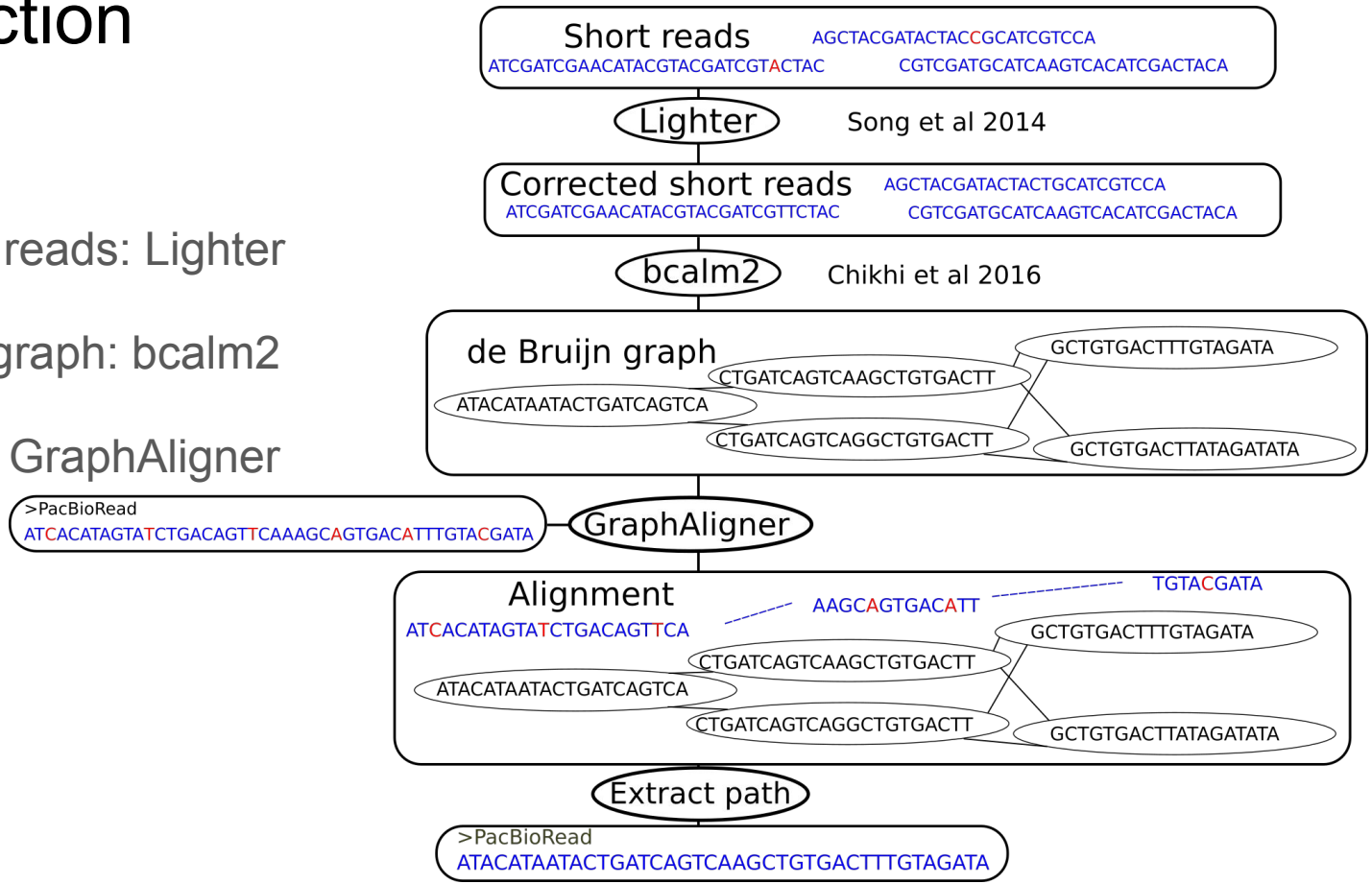
Error correction

Pipeline:

Self-correct short reads: Lighter

Build a de Bruijn graph: bcalm2

Align and extract: GraphAligner



Installation

Install snakemake, GraphAligner, bcalm from bioconda

Get the snakefile & config from https://github.com/maickrau/GraphAligner/tree/master/Snakemakes/ErrorCorrect

Running

Set the parameters in config.yaml

- Genome size
- Short read coverage
- Input read names
- Assembly parameters (next slide)

Snakemake --cores 8 all

Parameters

SmallK: Error correction k (lighter)

BigK: Graph k (bcalm)

Higher means more accurate but also more fragmented correction

Abundance: k-mer abundance (bcalm)

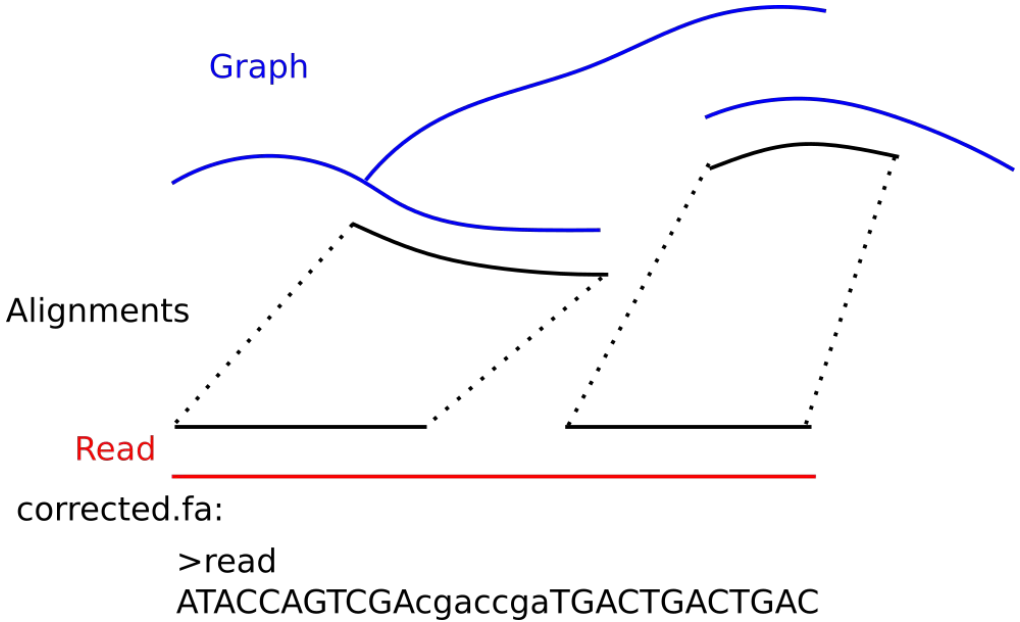
Higher means more accurate but also more fragmented correction

Output

Corrected reads will be in the output folder

corrected.fa

- Corrected where possible
- Uncorrected areas left in
- Uppercase is corrected
- Lowercase is uncorrected

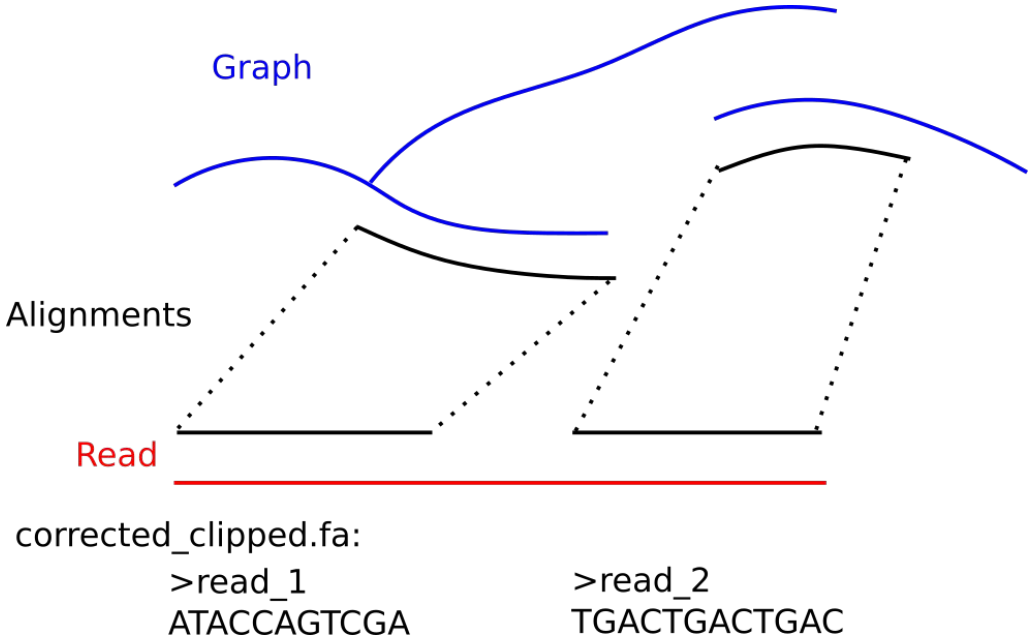


Output

Corrected reads will be in the output folder

corrected_clipped.fa

- Only corrected sequences
- Uncorrected areas removed
- The read is split across all gaps



Caveats

Works well for "normal" genomic data

Requires short read coverage

Does not work for RNA-seq data

May or may not work for metagenomics data

Results between runs might be slightly different

Questions

How confident are you at running error correction on hybrid sequencing data?