Pangenome graph augmentation from unassembled long reads

Luca Denti
Comenius University, Bratislava

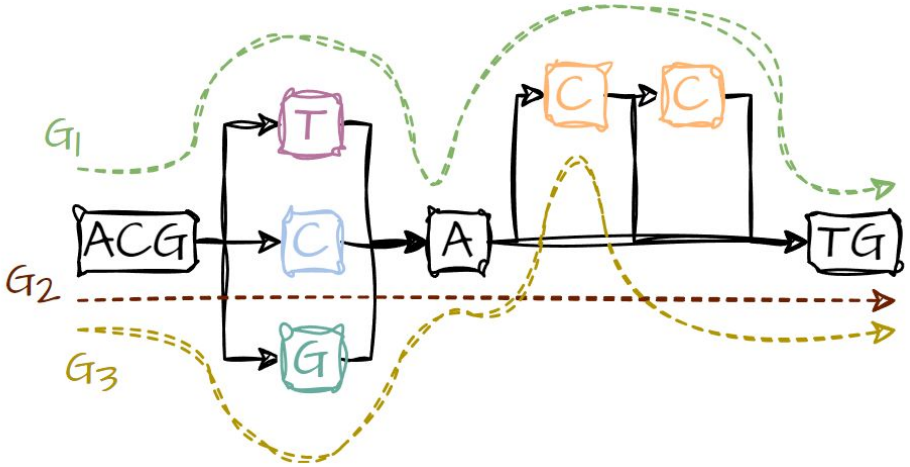
Pangenome Bio Hacking 2025 Workshop

Disclaimer: *work in progress*

What is a pangenome?

A pangenome is a collection of genomes

G_1 = ACGTACCTG
G_2 = ACGCATG
G_3 = ACGGACTG



genome == haplotype == assembly == path

*“most [graph] paths are unlikely recombinations of true haplotypes”**

** Sirén et al.* ***Personalized pangenome references.*** *Nature Methods (2024)*

Why pangenomes?

*"No single genome can represent the **diversity** in the human population. Using a single reference genome creates **reference biases**, adversely affecting variant discovery, gene–disease association studies and the accuracy of genetic analyses."*

https://humanpangenome.org/

A draft human pangenome reference

Wen-Wei Liao, Mobin Asri, Jana Ebler, Daniel Doerr, Marina Haukness, Glenn Hickey, Shuangjia Lu, Julian K. Lucas, Jean Monlong, Haley J. Abel, Silvia Buonaiuto, Xian H. Chang, Haoyu Cheng, Justin Chu, Vincenza Colonna, Jordan M. Eizenga, Xiaowen Feng, Christian Fischer, Robert S. Fulton, Shilpa Garg, Cristian Groza, Andrea Guarracino, William T. Harvey, Simon Heumos, ... Benedict Paten + Show authors

Nature **617**, 312–324 (2023) | Cite this article

Abstract

Here the Human Pangenome Reference Consortium presents a first draft of the human pangenome reference. The pangenome contains 47 phased, diploid assemblies from a cohort of genetically diverse individuals[1]. These assemblies cover more than 99% of the expected sequence in each genome and are more than 99% accurate at the structural and base pair levels. Based on alignments of the assemblies, we generate a draft pangenome that captures known variants and haplotypes and reveals new alleles at structurally complex loci. We also add 119 million base pairs of euchromatic polymorphic sequences and 1,115 gene duplications relative to the existing reference GRCh38. Roughly 90 million of the additional base pairs are derived from structural variation. Using our draft pangenome to analyse short-read data reduced small variant discovery errors by 34% and increased the number of structural variants detected per haplotype by 104% compared with GRCh38-based workflows, which enabled the typing of the vast majority of structural variant alleles per sample.

How to build a pangenome?

- `make_prg/Pandora` *[Colquhoun et al. 2021]*
- `panpa` *[Dabbaghie et al. 2023]*
- `pangeblocks` *[Avila et al. 2024]*

- `pggb` *[Garrison et al. 2024]*

- `minigraph` *[Li et al. 2020]*
- `minigraph-cactus` *[Hickey et al. 2024]*

How to build a pangenome?

MSA-based:

- `make_prg/Pandora` *[Colquhoun et al. 2021]*
- `panpa` *[Dabbaghie et al. 2023]*
- `pangeblocks` *[Avila et al. 2024]*

All-vs-all:

- `pggb` *[Garrison et al. 2024]*

Progressive:

- `minigraph` *[Li et al. 2020]*
- `minigraph-cactus` *[Hickey et al. 2024]*

How to build a pangenome?

MSA-based:

- `make_prg/Pandora` *[Colquhoun et al. 2021]*
- `panpa` *[Dabbaghie et al. 2023]*
- `pangeblocks` *[Avila et al. 2024]*

All-vs-all:

- `pggb` *[Garrison et al. 2024]*

Progressive:

- `minigraph*` *[Li et al. 2020]*
- `minigraph-cactus` *[Hickey et al. 2024]*

High-quality assemblies:

✗ *costly (multiple seq.tech.)*

✗ *computationally expensive*

** Minigraph can work with reads, but construction pipeline has been designed for assemblies.*

Progressive construction

A graph can be progressively constructed by iteratively aligning and inserting one genome at the time

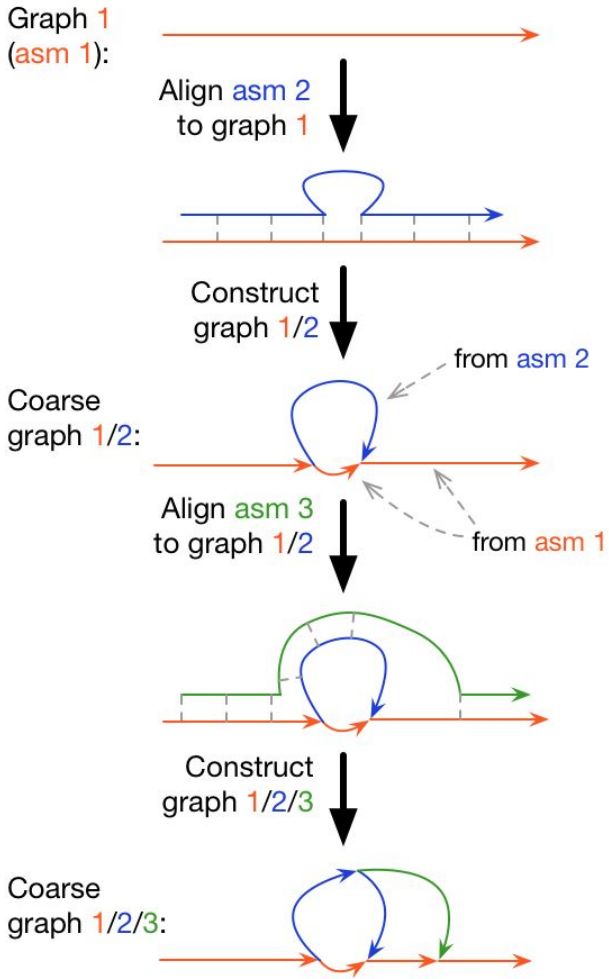
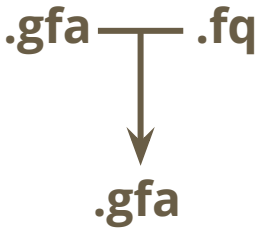


Image from github.com/lh3/minigraph

Progressive construction from long reads

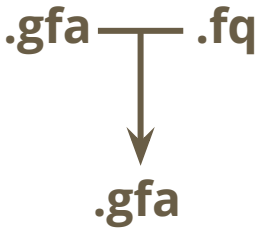
Provide an **assembly-free** and **alignment-free*** approach for the progressive construction of a pangenome



** Long read alignment to graph resulted unreliable from my tests on complex regions of the genome*

Progressive construction from long reads

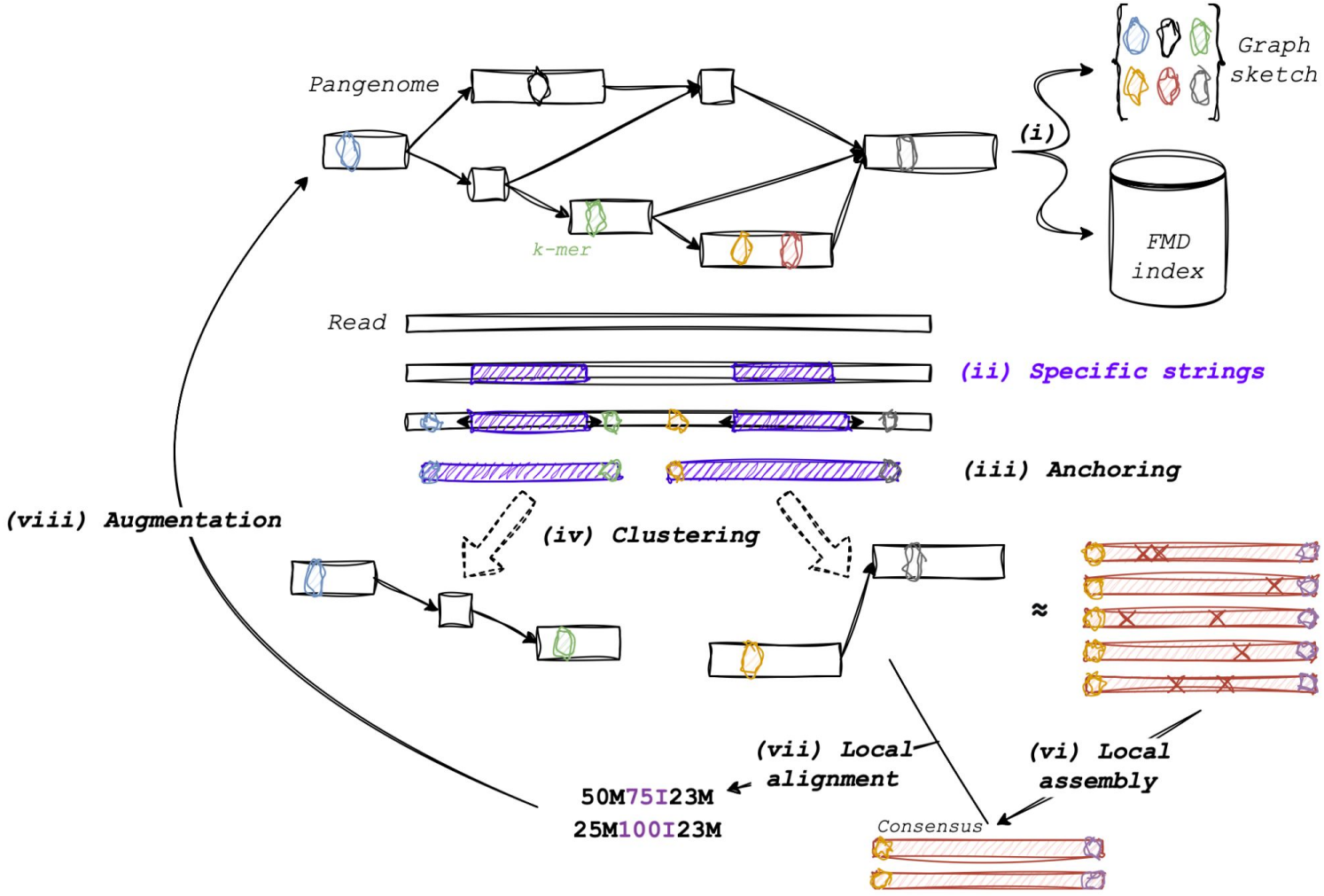
Provide an **assembly-free** and **alignment-free*** approach
for the progressive construction of a pangenome

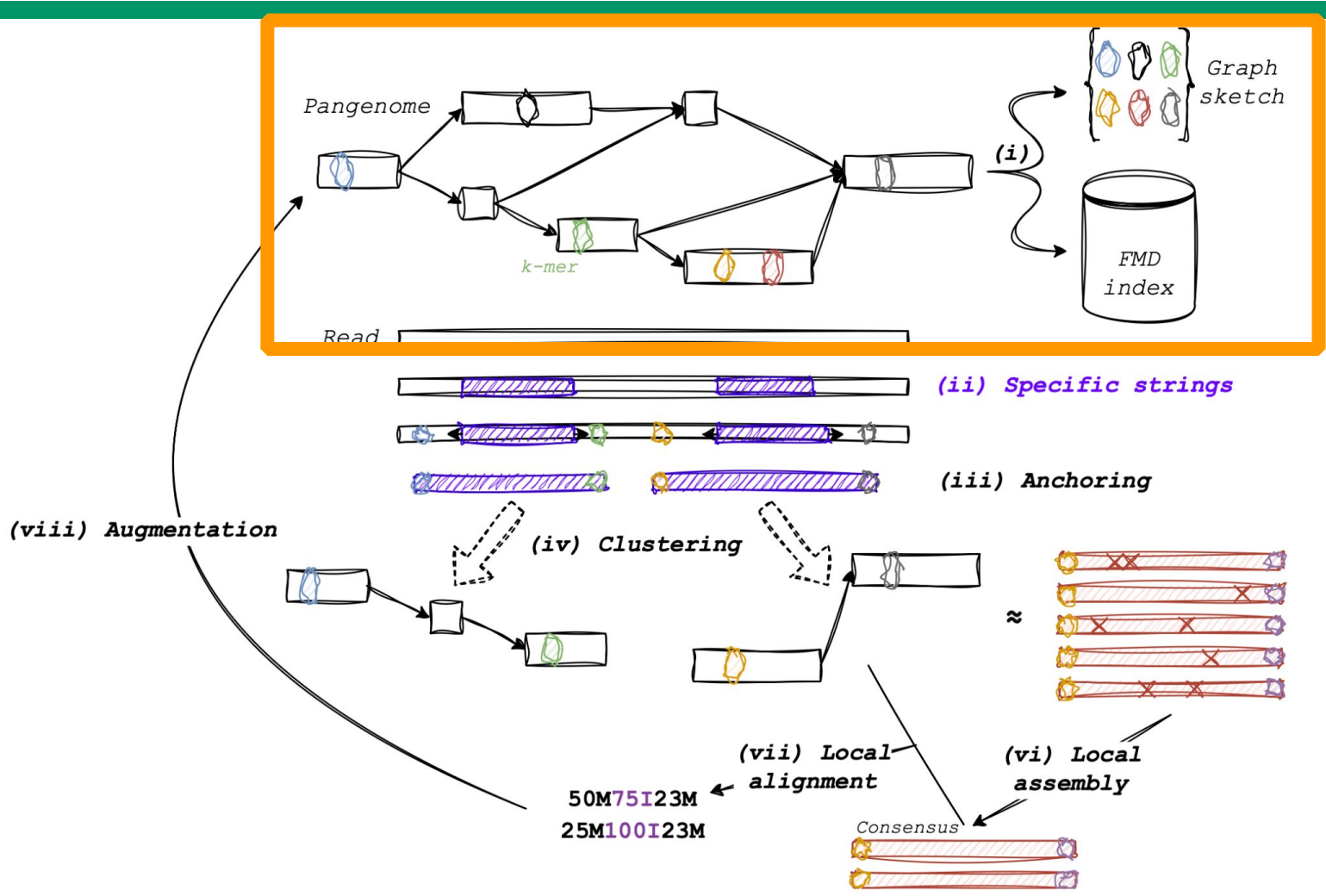


Several questions:

- how can we pinpoint differences with known genomes?
- how can we locate such differences in the graph?
- how can we summarize information coming from different reads?

** Long read alignment to graph resulted unreliable from my tests on complex regions of the genome*


Pangenome
k-mer
Graph sketch
(i)
FMD index
Read
(ii) Specific strings
(iii) Anchoring
(iv) Clustering
(viii) Augmentation
≈
(vii) Local alignment
(vi) Local assembly
50M75I23M
25M100I23M
Consensus


Pangenome
k-mer
Read
(i)
Graph sketch
FMD index
(ii) Specific strings
(iii) Anchoring
(iv) Clustering
(viii) Augmentation
≈
(vii) Local alignment
(vi) Local assembly
50M75I23M
25M100I23M
Consensus

Path indexing and graph sketching

We rely on both representations of a pangenome:

- **string-based**

- **graph-based**

Path indexing and graph sketching

We rely on both representations of a pangenome:

- **string-based**
  - *paths (i.e., haplotypes) are indexed using an FMD-index*
- **graph-based**

FM-Index

- **Self-index** for strings
  - data + index in a compressed structure
  - small size with full-text searching
  - based on BWT permutation
  - counts array + rank structure(s)

	SA	Rotations	BWT	$	a	b	n
0	7	$banana	a	0	1	0	0
1	6	a$banan	n	0	1	0	1
2	4	ana$ban	n	0	1	0	2
3	1	anana$b	b	0	1	0	2
4	0	banana$	$	1	1	1	2
5	5	na$bana	a	1	2	1	2
6	2	nana$ba	a	1	3	1	2
Counts				0	1	4	5

- Operations on pattern P in linear time *O(|P|)*:
  - **count(P)**: thanks to LF-mapping (aka **backward extension** in *O(1)*)
  - locate(P): with the addition of Suffix Array (very expensive, *sampling*)

FMD-Index

FM-Index of a *bidirectional collection* of DNA sequences:

$$R_1 \cdot \mathbf{RC}(R_1) \cdot R_2 \cdot \mathbf{RC}(R_2) \cdot \ldots \cdot R_n \cdot \mathbf{RC}(R_n)$$

RC(R) is reverse-and-complement R

Single index for both forward and reverse strands that allows in O(1):

➔ backward extensions

➔ **forward extensions**

*by backward extending pattern P with **A**, we also forward extend P with **T***

Heng Li. ***Exploring single-sample SNP and INDEL calling with whole-genome de novo assembly.*** *Bioinformatics (2012)*
Heng Li. ***Fast construction of FM-index for long sequence reads.*** *Bioinformatics (2014)*
Heng Li. ***BWT construction and search at the terabase scale.*** *Bioinformatics (2024)*

Path indexing and graph sketching

We rely on both representations of a pangenome:

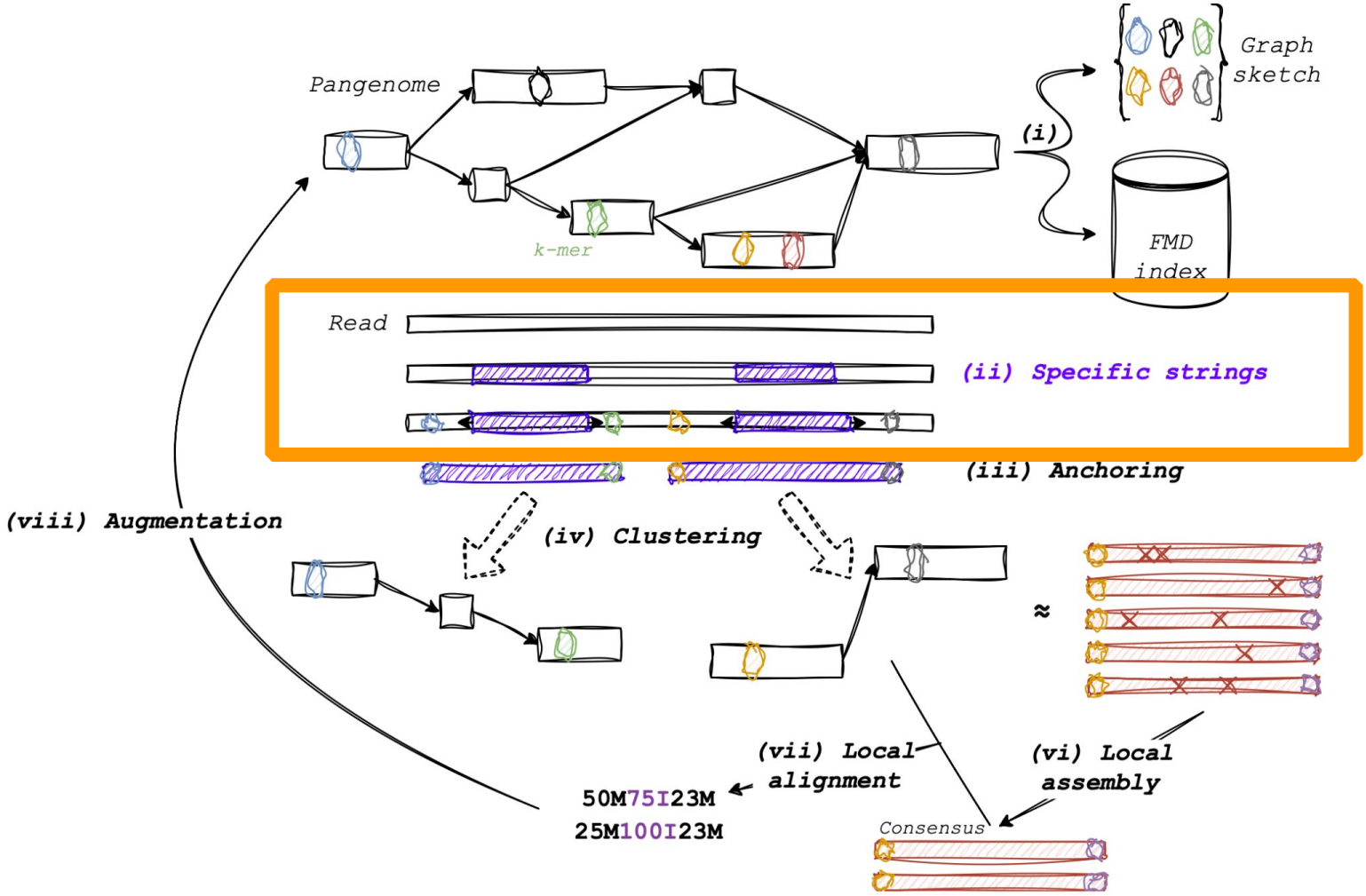
- **string-based**
  - *paths (i.e., haplotypes) are indexed using an FMD-index*
- **graph-based**

Path indexing and graph sketching

We rely on both representations of a pangenome:

- **string-based**
  - *paths (i.e., haplotypes) are indexed using an FMD-index*
- **graph-based**
  - *graph is sketched using **solid anchors** (k-mer → vertex)*

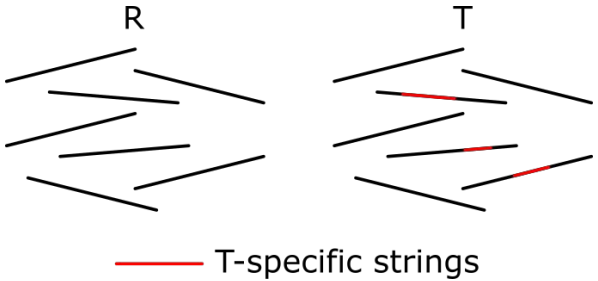
k-mer occurring in only one vertex of the graph
and less than #paths times in the FMD-index
(vertex space ≠ path space)


Pangenome
k-mer
Graph sketch
(i)
FMD index
Read
(ii) Specific strings
(iii) Anchoring
(iv) Clustering
(viii) Augmentation
≈
(vii) Local alignment
(vi) Local assembly
50M75I23M
25M100I23M
Consensus

Pinpoint differences with specific strings

Given two sets of strings *(R,T)*, a **specific string** is any substring:

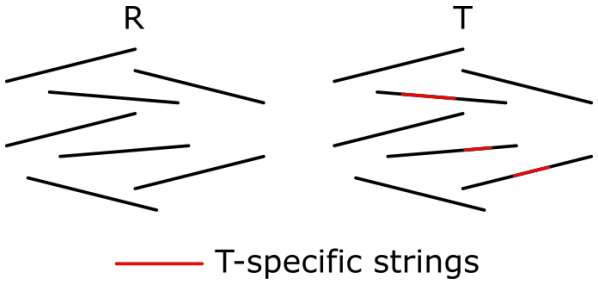
(i) occurring in T and not occurring in R *(substring)*

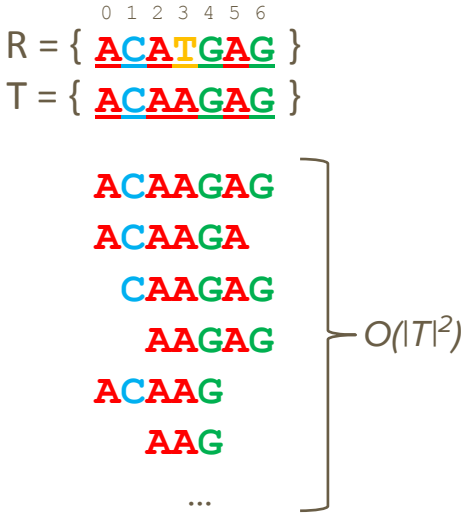


Pinpoint differences with specific strings

Given two sets of strings *(R,T)*, a **specific string** is any substring:

(i) occurring in T and not occurring in R *(substring)*



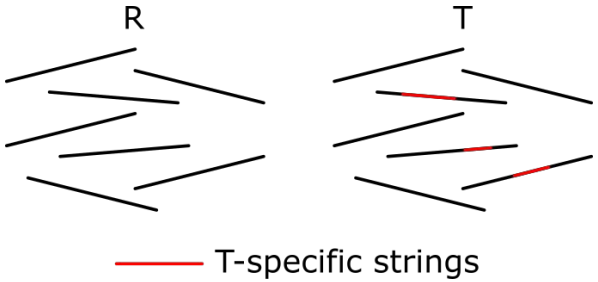


P. Khorsand, L. Denti, et al. ***Comparative genome analysis using sample-specific string detection in accurate long reads.*** *Bioinformatics Advances (2021)*

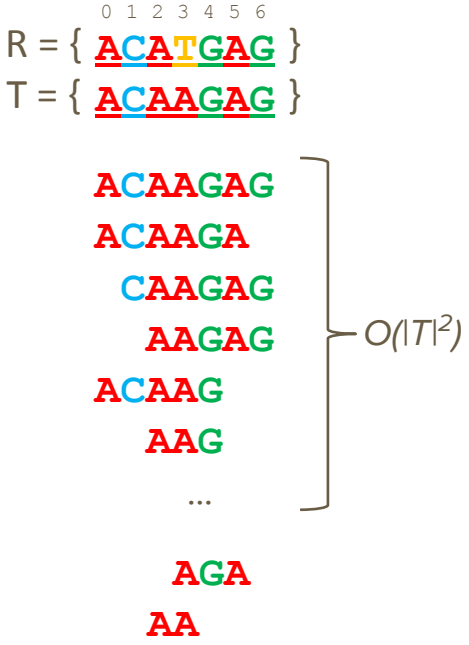
Pinpoint differences with specific strings

Given two sets of strings *(R,T)*, a **specific string** is any substring:

(i) occurring in T and not occurring in R *(substring)*



(ii) s.t. no other specific string is contained in it
(i.e., it is the shortest)

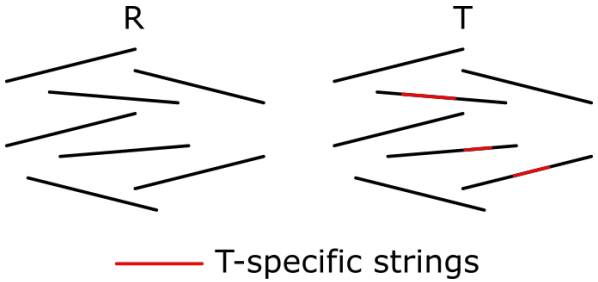


P. Khorsand, L. Denti, et al. ***Comparative genome analysis using sample-specific string detection in accurate long reads.*** *Bioinformatics Advances (2021)*

Pinpoint differences with specific strings

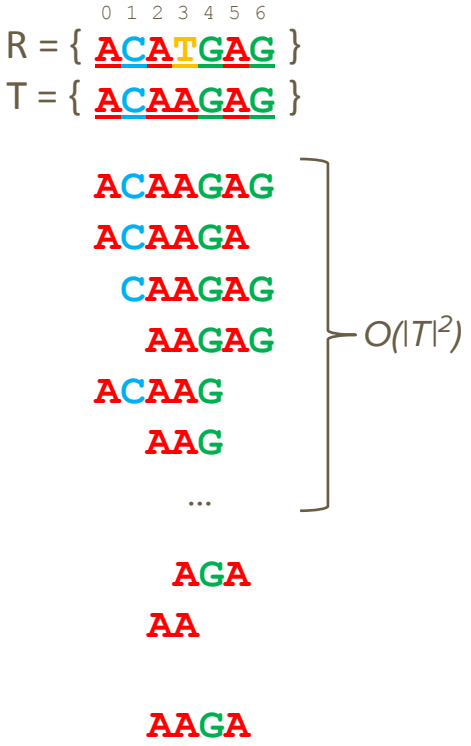
Given two sets of strings *(R,T)*, a **specific string** is any substring:

(i) occurring in T and not occurring in R *(substring)*



(ii) s.t. no other specific string is contained in it
(i.e., it is the shortest)

(super-)Specific: merge all specific strings overlapping on t∈T



P. Khorsand, L. Denti, et al. ***Comparative genome analysis using sample-specific string detection in accurate long reads.*** *Bioinformatics Advances (2021)*

Why specific strings?

Specific strings pinpoint any difference between two sets of strings

if we index a reference genome, they pinpoint (mostly) all variations

P. Khorsand, L. Denti, et al. ***Comparative genome analysis using sample-specific string detection in accurate long reads.*** *Bioinformatics Advances (2021)*

Why specific strings?

Specific strings pinpoint any difference between two sets of strings

*if we index a reference **pan**genome, they pinpoint (mostly) all **unknown** variations*

Why specific strings?

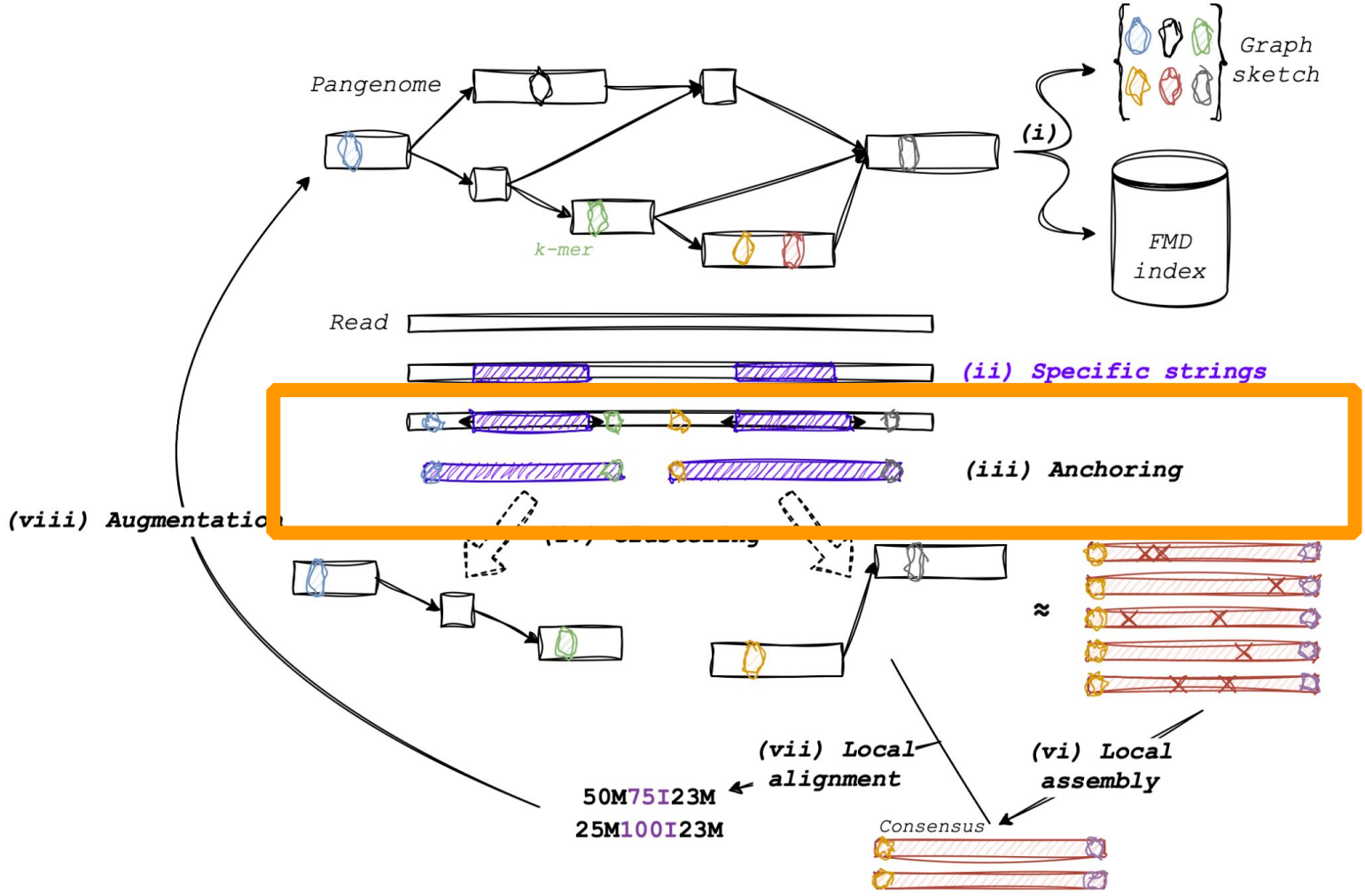
Specific strings pinpoint any difference between two sets of strings

*if we index a reference **pan**genome, they pinpoint (mostly) all **unknown** variations*

*Why not using specific k-mers?**

- *specific strings cover more variations (variable-length)*
- *(super-)specific strings are easier to compute*

P. Khorsand, L. Denti, et al. ***Comparative genome analysis using sample-specific string detection in accurate long reads.*** *Bioinformatics Advances (2021)*


Pangenome
k-mer
Graph sketch
(i)
FMD index
Read
(ii) Specific strings
(iii) Anchoring
(viii) Augmentatio
≈
(vii) Local alignment
50M75I23M
25M100I23M
(vi) Local assembly
Consensus

Anchoring specific strings to graph *(WIP)*

A specific string **S** is a substring of a read **R** (S = *R[p:p+l]*)

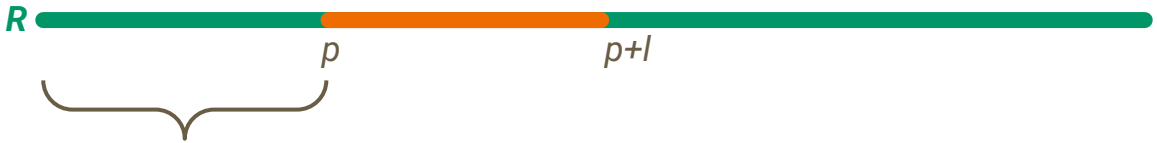
Anchoring specific strings to graph *(WIP)*

A specific string **S** is a substring of a read **R** (S = *R[p:p+l]*)

To anchor S to the graph, we search for solid anchors in its neighborhood:

Anchoring specific strings to graph *(WIP)*

A specific string **S** is a substring of a read **R** (S = *R[p:p+l]*)



To anchor S to the graph, we search for solid anchors in its neighborhood:

1. Select x(=20) solid anchors on the left of S *(i.e., in R[:p])*

Anchoring specific strings to graph *(WIP)*

A specific string **S** is a substring of a read **R** (S = *R[p:p+l]*)

To anchor S to the graph, we search for solid anchors in its neighborhood:

1. Select x(=20) solid anchors on the left of S *(i.e., in R[:p])*
2. Select x(=20) solid anchors on the right of S *(i.e., in R[p+l:])*

Anchoring specific strings to graph *(WIP)*

A specific string **S** is a substring of a read **R** (S = *R[p:p+l]*)

To anchor S to the graph, we search for solid anchors in its neighborhood:

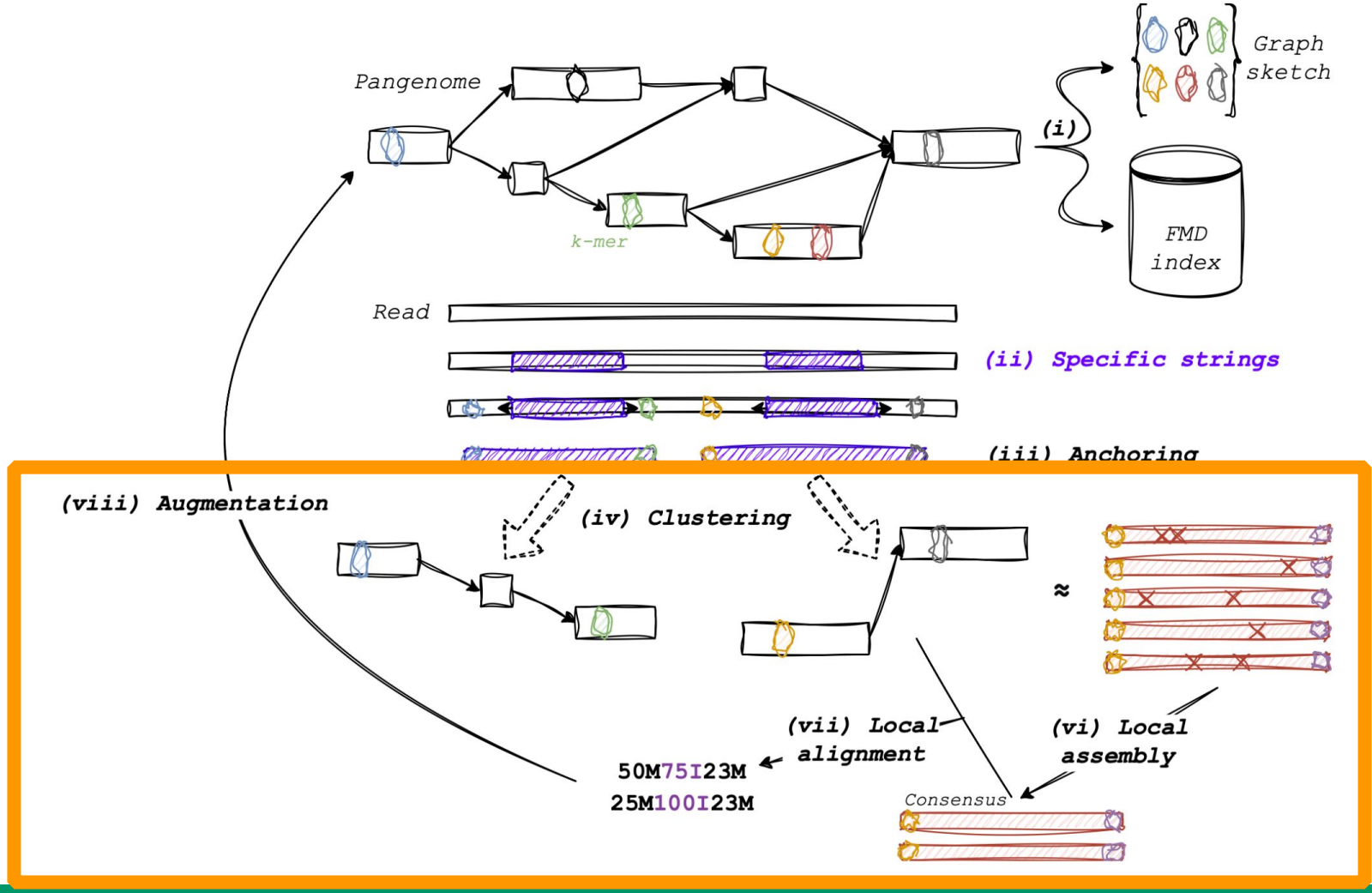
1. Select x(=20) solid anchors on the left of S *(i.e., in R[:p])*
2. Select x(=20) solid anchors on the right of S *(i.e., in R[p+l:])*
3. Pick the best pair *(closest on R and G, colinear on at least one haplotype)*

Anchoring specific strings to graph *(WIP)*

A specific string **S** is a substring of a read **R** (S = *R[p:p+l]*)

To anchor S to the graph, we search for solid anchors in its neighborhood:

1. Select x(=20) solid anchors on the left of S *(i.e., in R[:p])*
2. Select x(=20) solid anchors on the right of S *(i.e., in R[p+l:])*
3. Pick the best pair *(closest on R and G, co-linear on at least one haplotype)*
4. Extend the specific string to include the two solid anchors


Pangenome
k-mer
Graph sketch
(i)
FMD index
Read
(ii) Specific strings
(iii) Anchoring
(viii) Augmentation
(iv) Clustering
≈
(vii) Local alignment
(vi) Local assembly
50M75I23M
25M100I23M
Consensus

Specific strings clustering

Each anchored specific string is now a tuple
(R, p, l, A, B), with A and B solid anchors

Specific strings clustering

Each anchored specific string is now a tuple
(R, p, l, A, B), with A and B solid anchors

Cluster them based on where they have been anchored

Specific strings clustering

Each anchored specific string is now a tuple
(R, p, l, A, B), with A and B solid anchors

Cluster them based on where they have been anchored (**sweep-line**):

1. Sort all specific strings by their left anchor (A)

Specific strings clustering

Each anchored specific string is now a tuple
(R, p, l, A, B), with A and B solid anchors

Cluster them based on where they have been anchored (**sweep-line**):

1. Sort all specific strings by their left anchor (A)
2. Create a cluster with the first specific string

Specific strings clustering

Each anchored specific string is now a tuple
(R, p, l, A, B), with A and B solid anchors

Cluster them based on where they have been anchored (**sweep-line**):

1. Sort all specific strings by their left anchor (A)
2. Create a cluster with the first specific string
3. Iterate over remaining strings while keeping one open cluster:
    - if new string overlap with current cluster, add it
    - otherwise, close current cluster and create a new one

Cluster analysis

Each cluster represents **specific portions** of the haplotypes of the new individual

We just need to summarize the information contained in each cluster to understand how to augment the graph

Cluster analysis

Each cluster represents **specific portions** of the haplotypes of the new individual

We just need to summarize the information contained in each cluster to understand how to augment the graph

Some challenges:

- specific strings may come from different haplotypes *(due to ploidy)*
- they may overlap and do not share anchors *(due to ploidy and sequencing errors)*

Cluster analysis

Each cluster represents **specific portions** of the haplotypes of the new individual

We just need to summarize the information contained in each cluster to understand how to augment the graph

Some challenges:

- specific strings may come from different haplotypes *(due to ploidy)*
- ~~they may overlap and do not share anchors *(due to ploidy and sequencing errors)*~~

Cluster analysis

A cluster is a triple *(A, B, **S**)*, *with left-most anchor A, right-most anchor B and anchored specific strings **S***

Cluster analysis

A cluster is a triple *(A, B, **S**)*, *with left-most anchor A, right-most anchor B and anchored specific strings **S***

1. Cluster is split based on specific strings length $\frac{\min(|s_1|, |s_2|)}{\max(|s_1|, |s_2|)} \geq r(=0.97)$
    - *heterozygous variations of different length*

Cluster analysis

A cluster is a triple *(A, B, **S**)*, *with left-most anchor A, right-most anchor B and anchored specific strings **S***

1. Cluster is split based on specific strings length
    - *heterozygous variations of different length*
2. Create one or two consensus per subcluster via POA (abpoa)
    - heterozygous variations of same size *(e.g., multi-allelic SNPs)*

Cluster analysis

A cluster is a triple *(A, B, **S**)*, *with left-most anchor A, right-most anchor B and anchored specific strings **S***

1. Cluster is split based on specific strings length
   - *heterozygous variations of different length*
2. Create one or two consensus per subcluster via POA (`abpoa`)
   - heterozygous variations of same size *(e.g., multi-allelic SNPs)*
3. Each consensus is locally aligned back to the graph (`ksw2`)
   - ✗ Actually, back to each “unique” haplotype path passing through the two anchors

Cluster analysis

A cluster is a triple *(A, B, **S**)*, *with left-most anchor A, right-most anchor B and anchored specific strings **S***

1. Cluster is split based on specific strings length
   - *heterozygous variations of different length*
2. Create one or two consensus per subcluster via POA (`abpoa`)
   - heterozygous variations of same size *(e.g., multi-allelic SNPs)*
3. Each consensus is locally aligned back to the graph (`ksw2`)
   - ✗ Actually, back to each “unique” haplotype path passing through the two anchors
4. Alignments are used to augment the graph (`vg augment`)

Experimental evaluation

Experiments on real graphs (from VCF) and simulated read samples to evaluate:

1. Reliability of sketching scheme
    - *can we use our anchors?*

2. Accuracy of graph augmentation
    - *can we trust our results?*

WIP: *preliminary results*

Exp1 - Reliability of sketching scheme

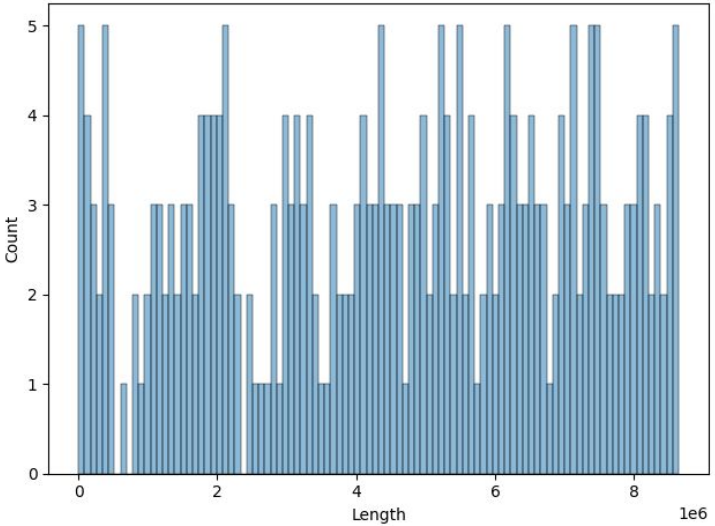
1. Build graphs with increasing number **n** of individuals *(HPRC VCF on T2T)*
2. Sketch graphs
3. Check distance between consecutive solid anchors on T2T genome

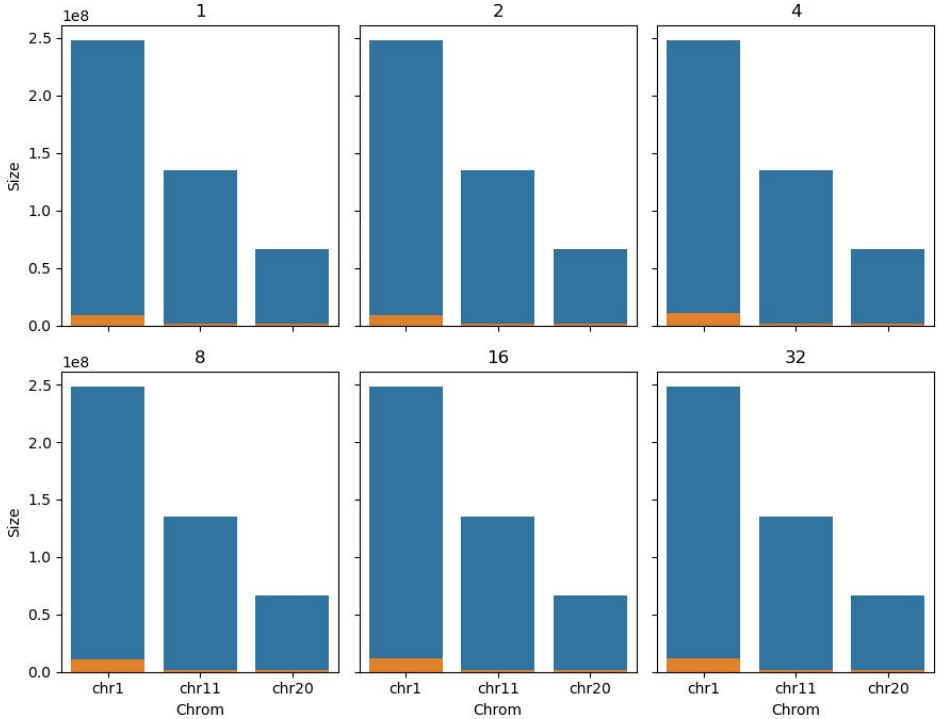
(we miss all regions longer than read length)

Exp1 - Results (chr1+11+20, d=15000)

99.996% of anchors are very close *(≤100bp apart)* even with 32 samples

285 missed regions on 32 samples graph overlapping known repeats (250) and segdups (35)





Exp2 - Accuracy of graph augmentation

1. Build graphs with increasing number **n** of individuals *(HPRC VCF on T2T)*
2. Simulate diploid reads from individual **x** not in graph *(leave-1-out)*
3. Augment the graph with corrected reads *(hifiasm ec)*
4. Align reads to **graph** with minigraph

Exp2 - Accuracy of graph augmentation

1. Build graphs with increasing number **n** of individuals *(HPRC VCF on T2T)*
2. Simulate diploid reads from individual **x** not in graph
3. Augment the graph with corrected reads *(hifiasm ec)*
4. Align reads to **graph** with minigraph

- *Original graph with n+x* *(FULL)*
- *Original graph with n* *(1OUT)*
- *Augmented n+x graph* *(FULL-AUG)*
- *Augmented n graph* *(1OUT-AUG)*

Exp2 - Accuracy of graph augmentation

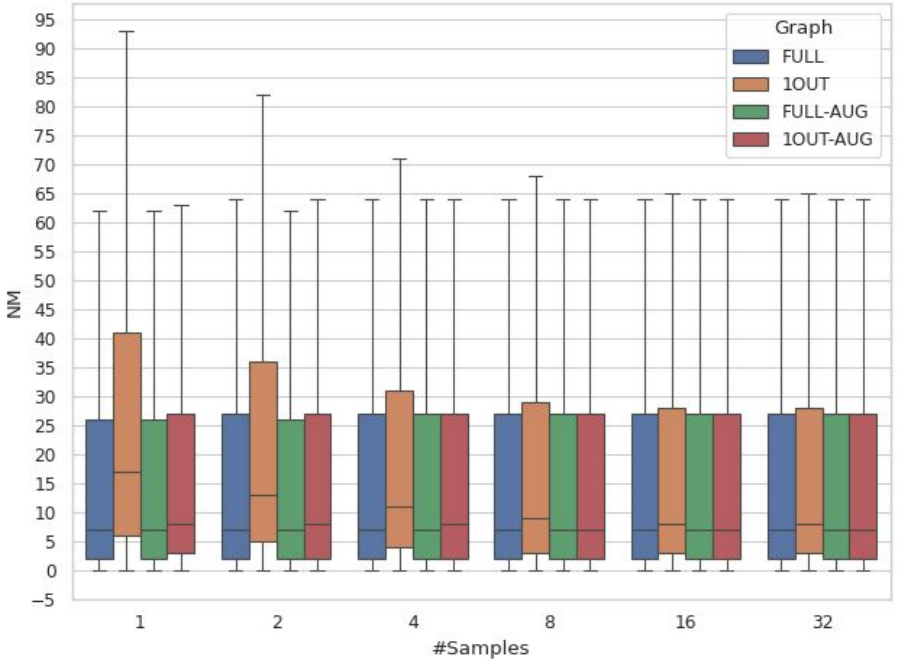
1. Build graphs with increasing number **n** of individuals *(HPRC VCF on T2T)*
2. Simulate diploid reads from individual **x** not in graph
3. Augment the graph with corrected reads *(hifiasm ec)*
4. Align reads to **graph** with minigraph

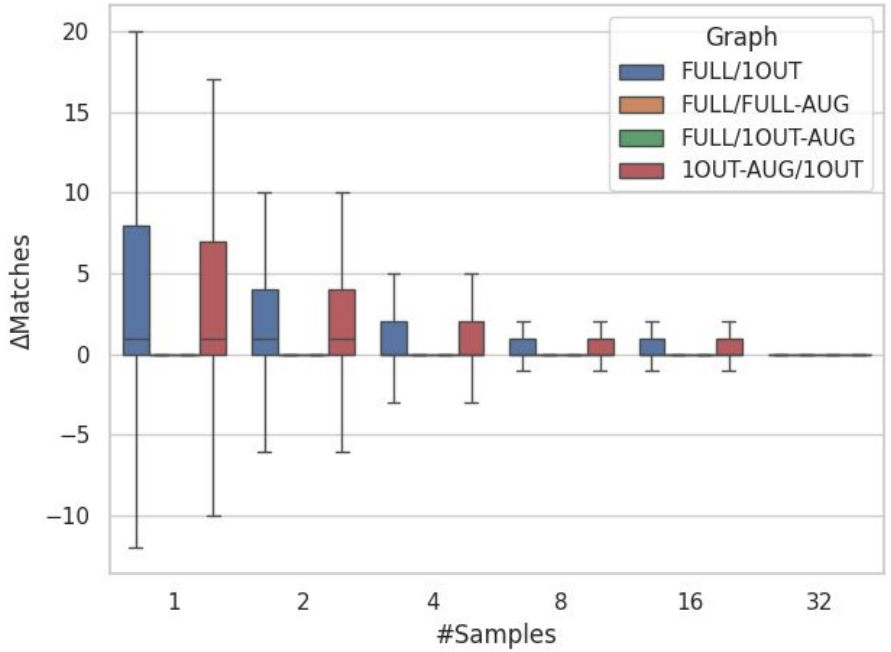
- *Original graph with n+x* *(FULL)*
- *Original graph with n* *(1OUT)*
- *Augmented n+x graph* *(FULL-AUG)*
- *Augmented n graph* *(1OUT-AUG)*

Metrics:

- “recall”, alignments accuracy
- “precision”, percentage of novel vertices/edges used by the alignments

Exp2 - Results (chr11)

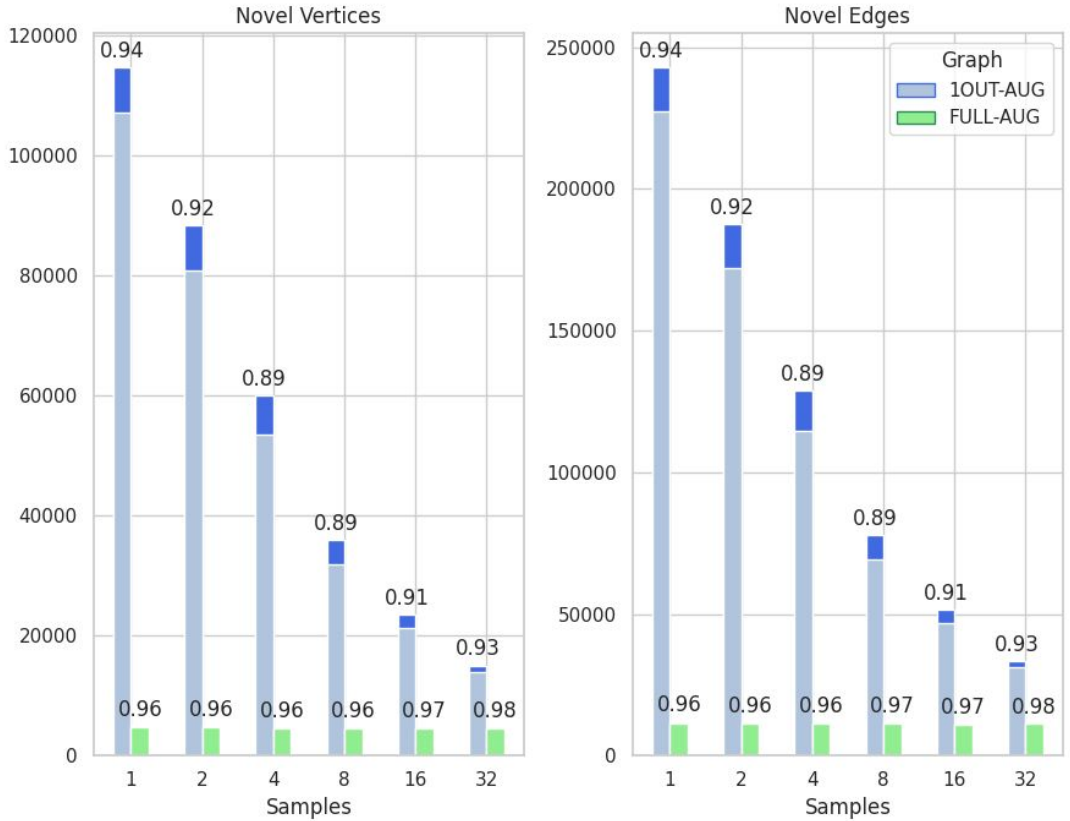


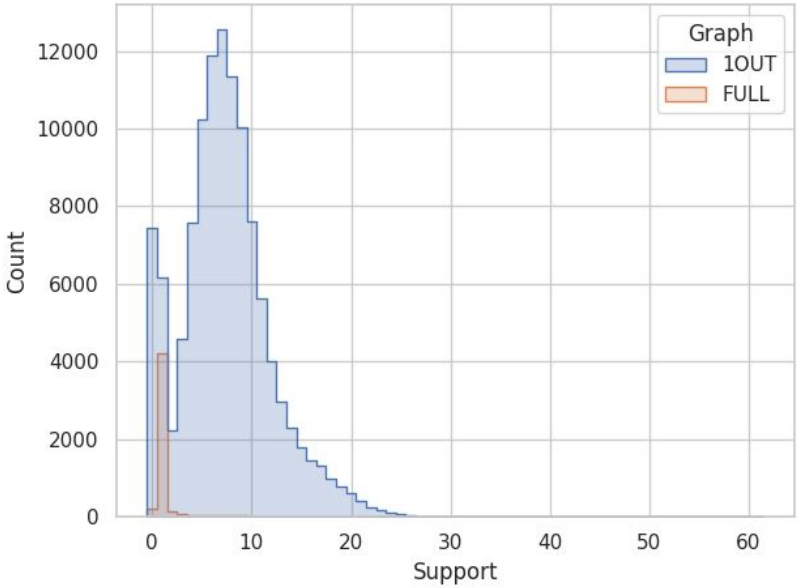


➢ *More individuals ➡ less novel variations ➡ more similar graphs*

➢ *SNPs dominates our analysis*

Exp2 - Results (chr11)





Conclusion

Results are promising: pangenome graphs can be augmented using long reads

Work-in-progress:

- improve “precision”
- local augmentation that does not include full haplotype paths (realignment)
- more general graphs, e.g., cycles *(DAG is assumed in second stage)*
- general code improvements (vg library)
- scalability issue
  - *RAM usage due to sketch*
  - *running times due to ksw2 against each “unique” path*
- HPRC graph and real samples

Thank you!

Questions?

This work has received funding from the European Union's Horizon programme under the Horizon Europe grant agreement (ASVA-CGR No.101180581)

RAM / Times

FMD-indexing: 82GB

Sketching: 18GB

Augmentation: 37GB

Sketching of full HPRC graph: 5 hours (16 threads)

Augmentation of single chr11: 7.5 hours (1 thread)