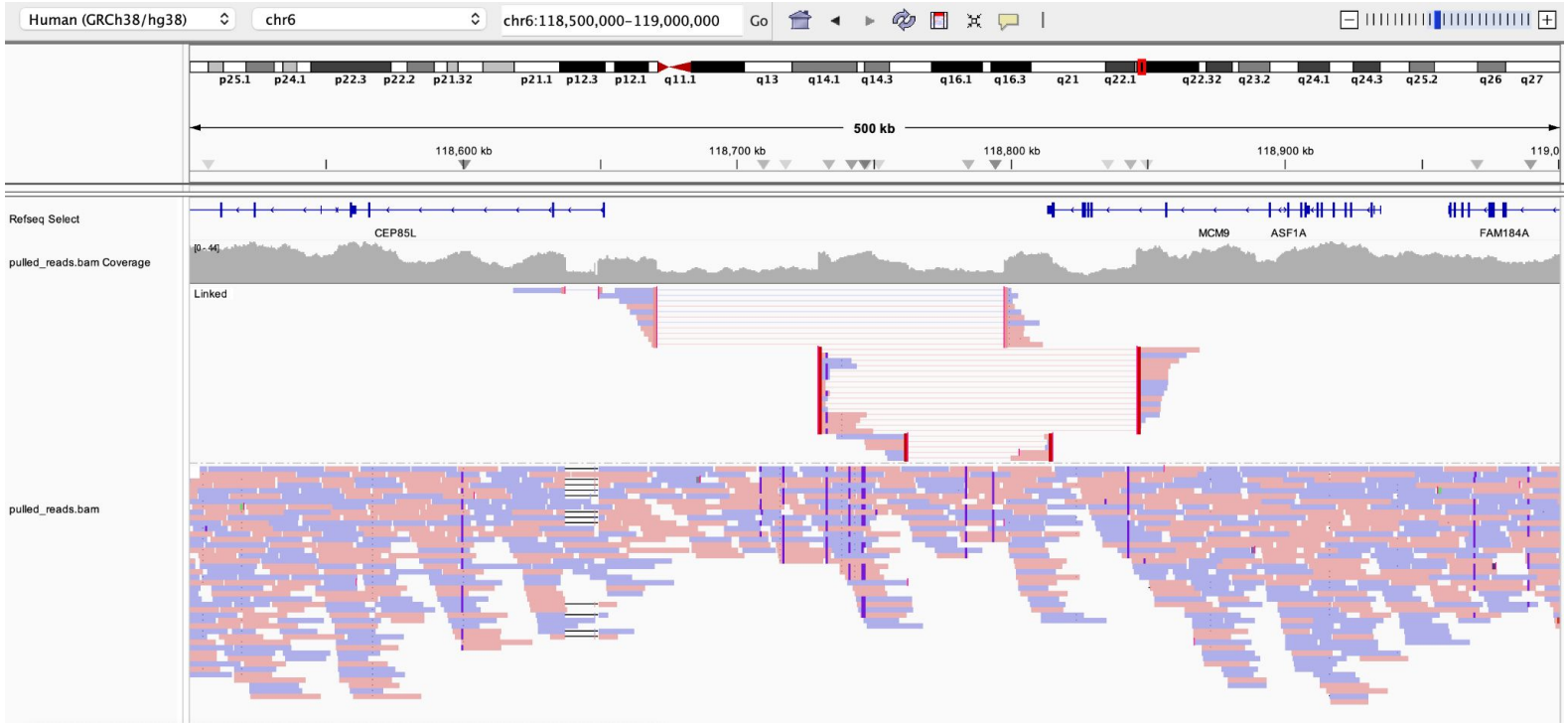
How to use long reads for de-novo assembly

Dr. Wolfram Höps
Radboudumc Nijmegen

25th May 2025

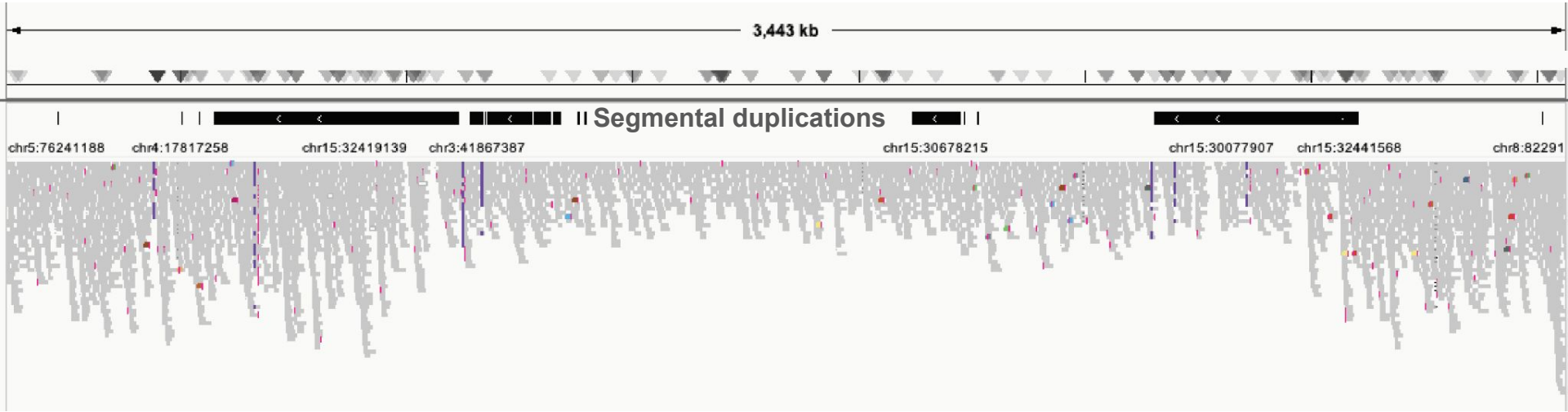
Complex SVs can be hard to interpret from aligned reads



Several Structural Variants are visible, but which genes are affected?

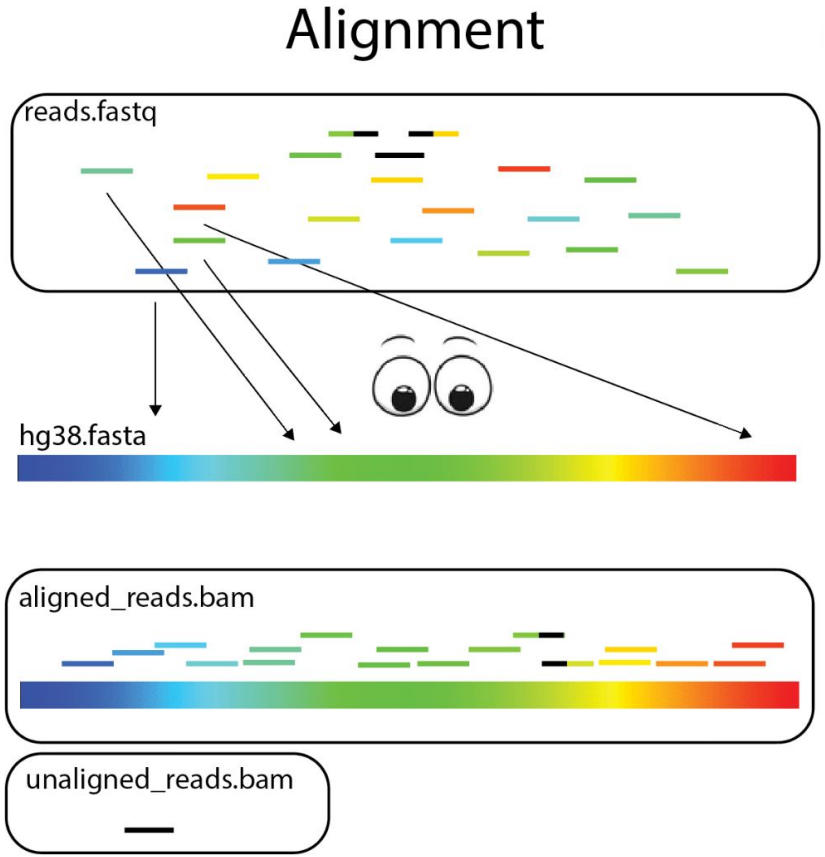
30-fold Hifi patient dataset: [Höps et al. 2025]

Complex loci can be hard to interpret from aligned reads



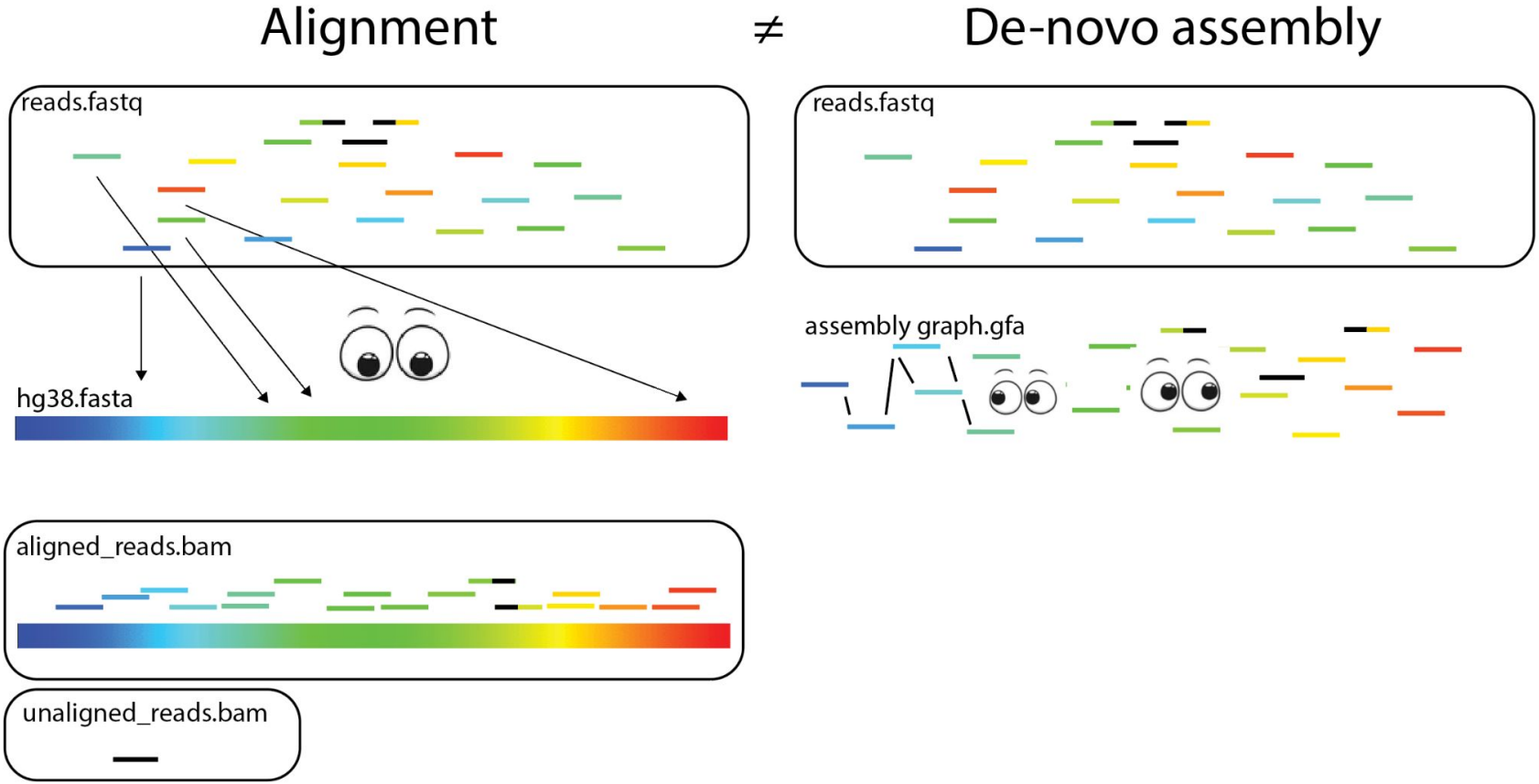
A multi-Mbp-sized deletion is visible, but where are the breakpoints?

Read alignment != de-novo assembly



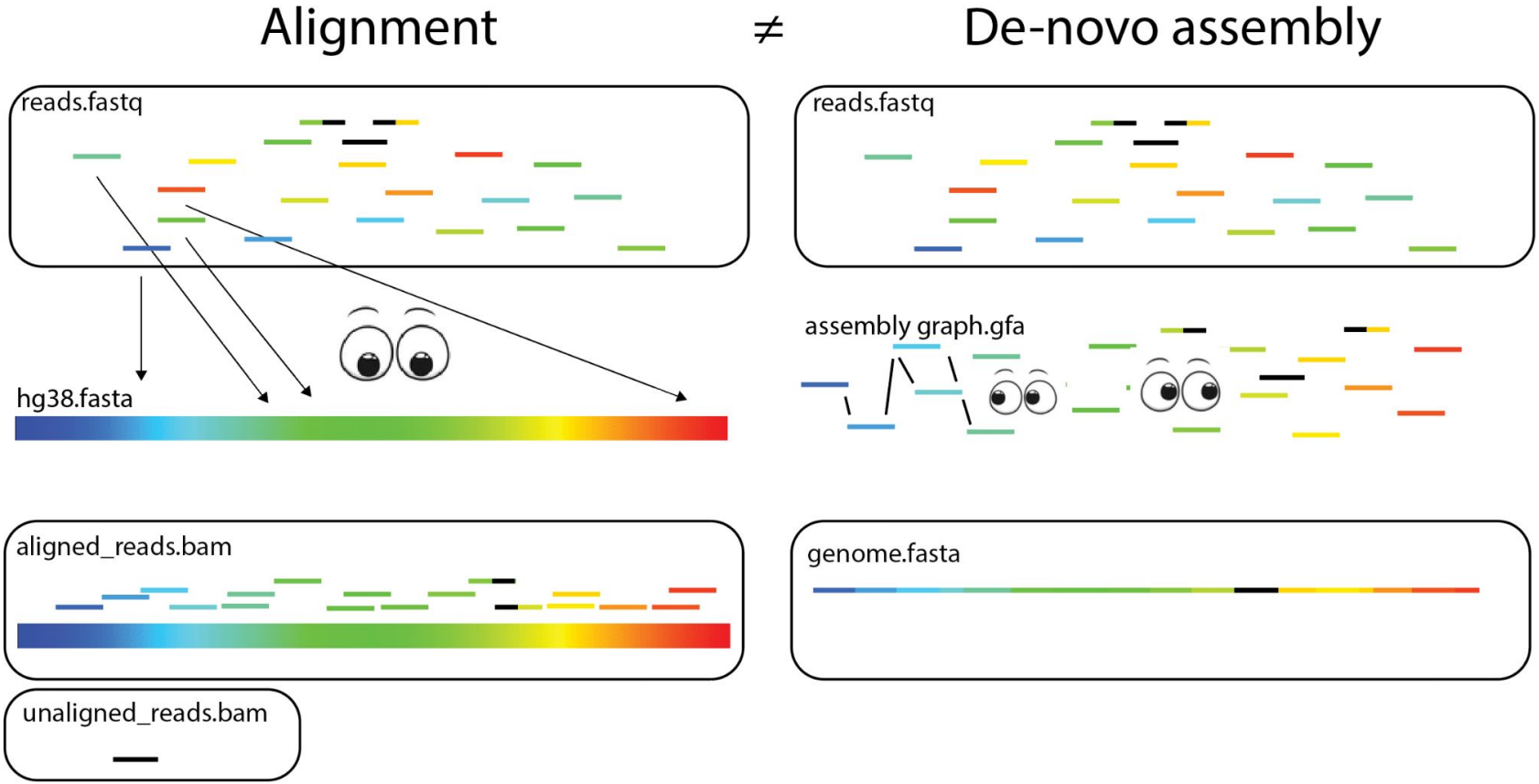
Own figure

Read alignment != de-novo assembly



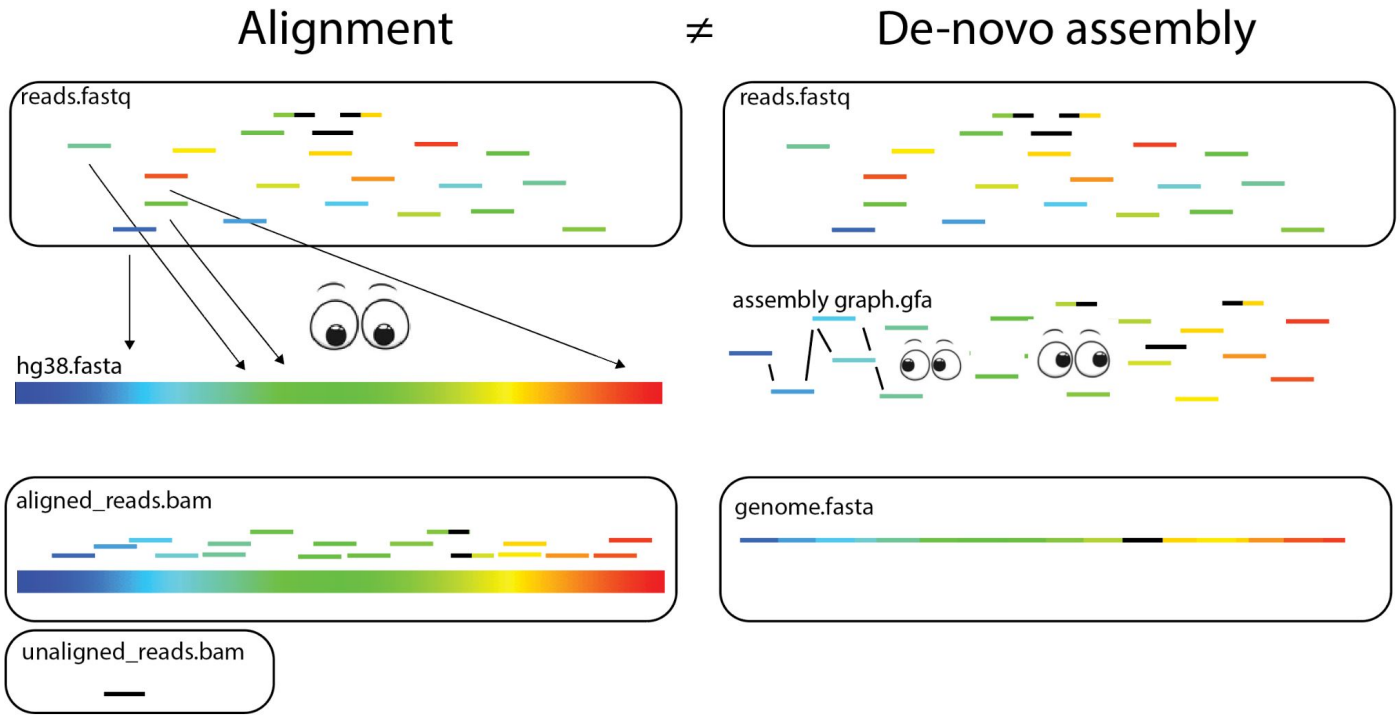
Own figure

Read alignment != de-novo assembly



Own figure

Read alignment != de-novo assembly

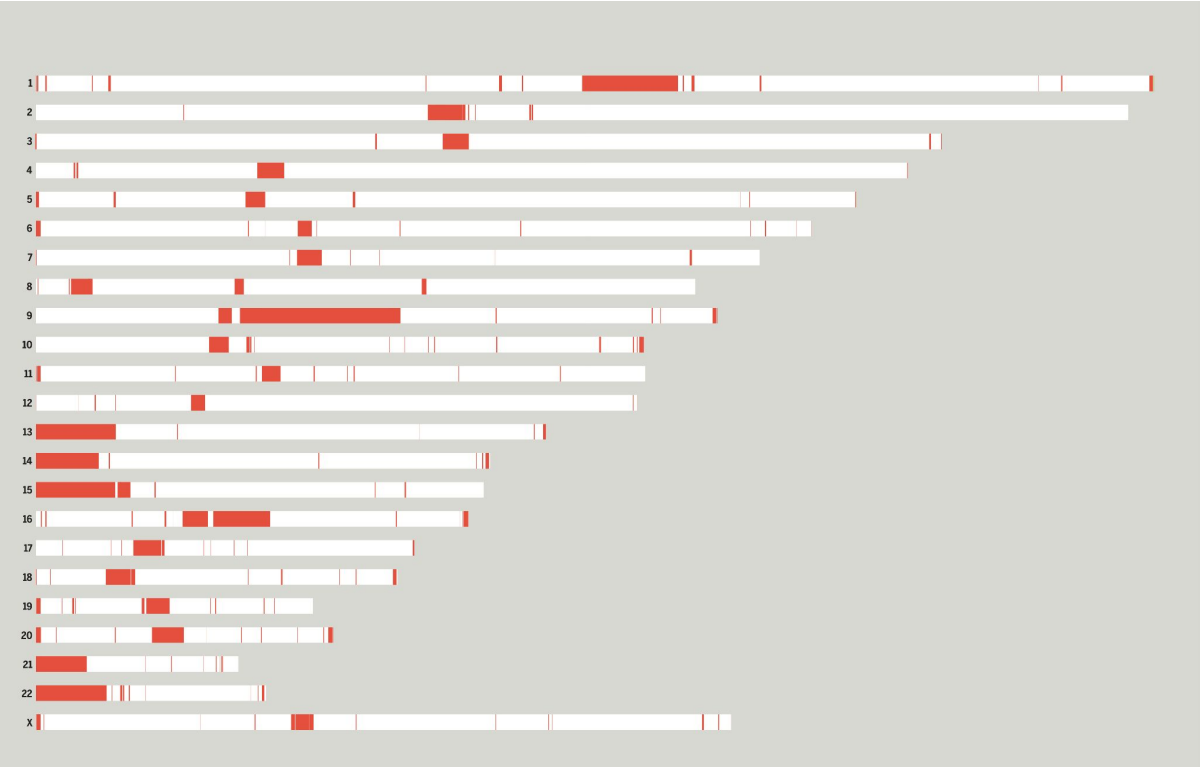


Requires Reference Genome
Variant calling in non-complex regions
Computationally light
Automated, easy to use

No Reference required (though sometimes useful)
Long, complex variants in hard-to-map regions
Computationally heavy
Harder to use, less automated

Own figure

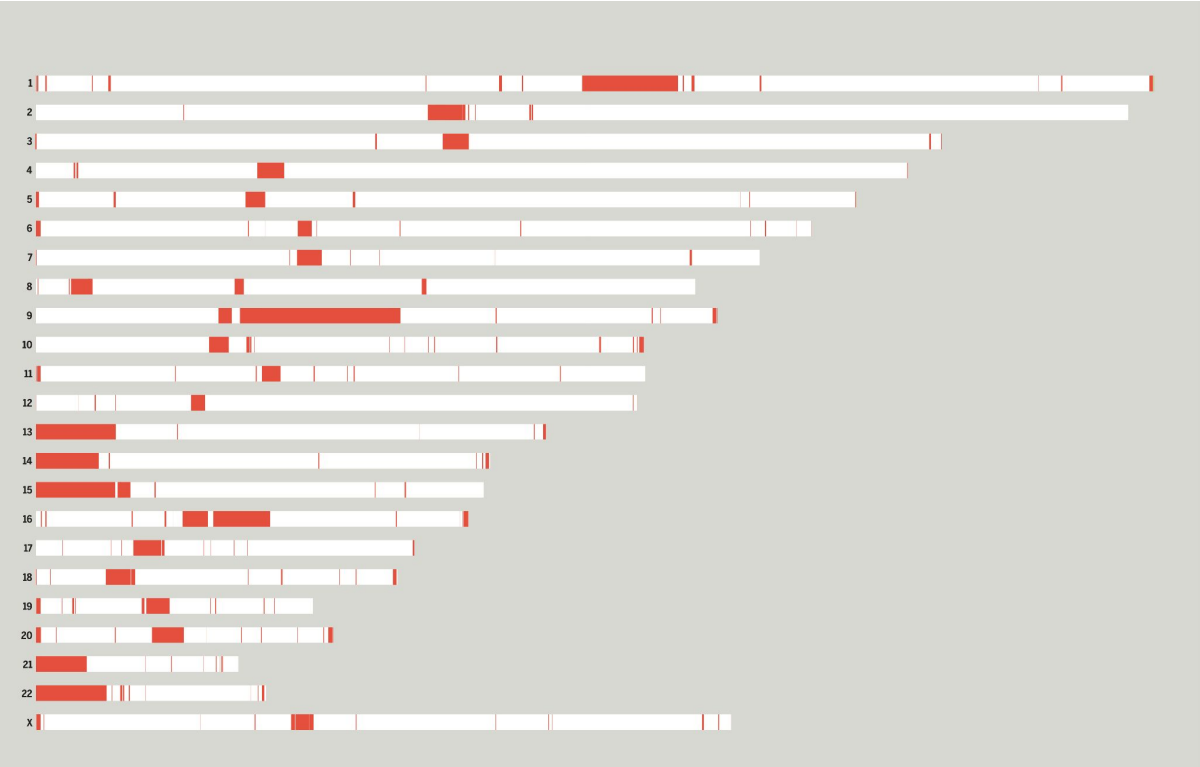
The telomere-to-telomere reference **assembly**



red:
regions
previously
(hg38)
unresolved

T2T consortium, 2021

The telomere-to-telomere reference **assembly**



red:
regions
previously
(hg38)
unresolved

More recently: >200 samples assembled near-T2T by the Human Pangenome Reference Consortium

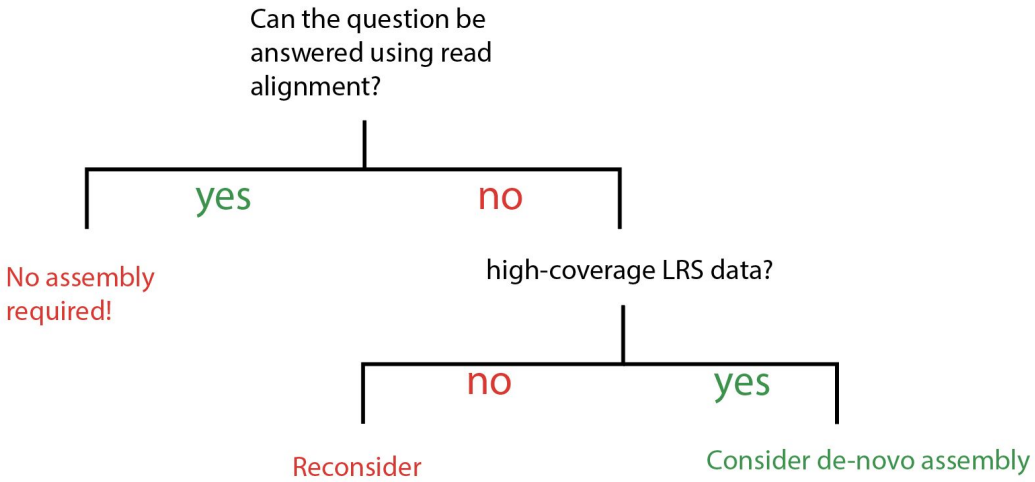
T2T consortium, 2021

The assembly ‘hype’ is technology-driven

Larger pieces make for an easier puzzle

Own figure

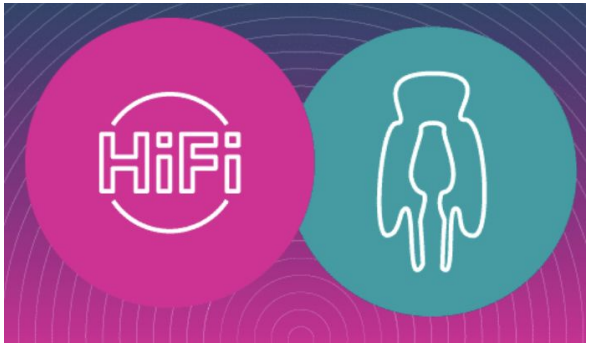
De-novo assembly of whole human genomes or smaller loci is now feasible for well-equipped labs



Typical applications in human genetics:

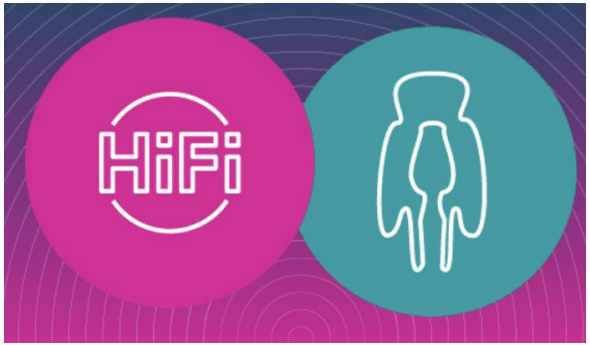
- Long or complex variation
- Hard to map genomic regions (e.g. recurrent disease loci)
- Impossible to map genomic regions (e.g. centromers / telomers)
- Pan-genomics, population genetics, …

Data requirements



Minimum: ~30-fold coverage LRS data [Li, Durbin 2025]
T2T-approaching: >60-fold coverage, often multi-technology [e.g. HPRC]

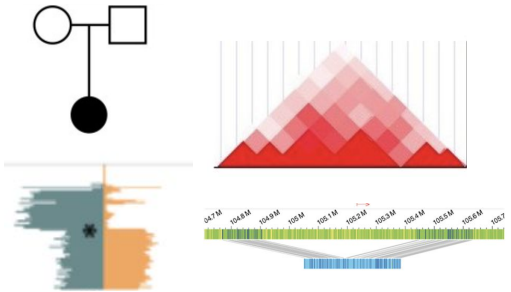
Data requirements



Minimum: ~30-fold coverage LRS data [Li, Durbin 2025]
T2T-approaching: >60-fold coverage, often multi-technology [e.g. HPRC]

Other data types can improve phasing and contiguity:

- Parental data
- Hi-C
- Strand-Seq
- Optical genome mapping



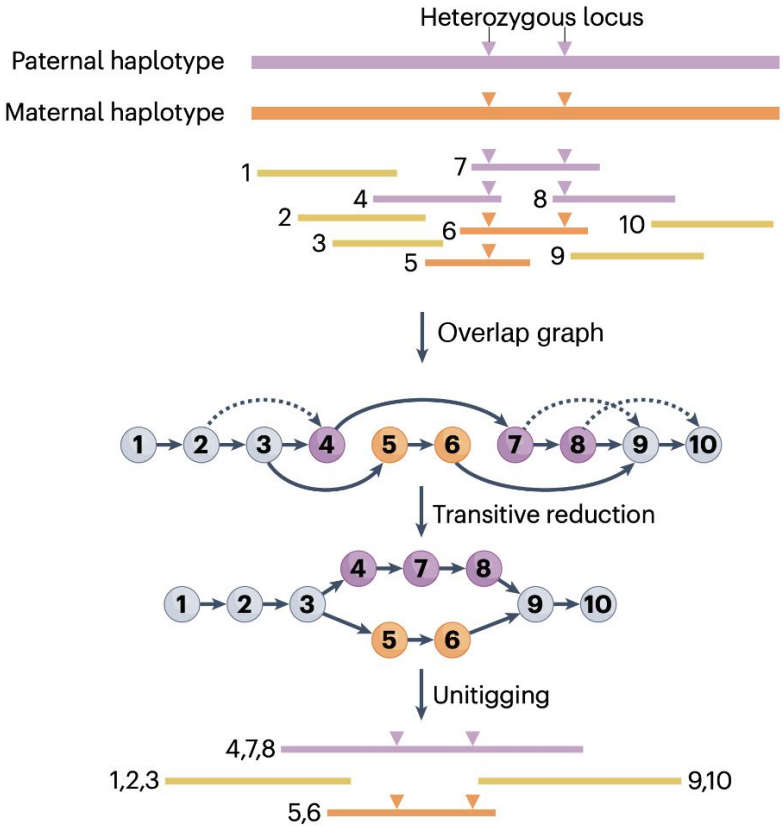
Assembly: Technical background

Phased assembly of diploid genomes

Some commonly used assembly tools for human genomes:
Verkko [Rautainen et al. 2023], **Hifiasm** [Cheng et al. 2021], **Flye** [Kolmogorov et al. 2020]

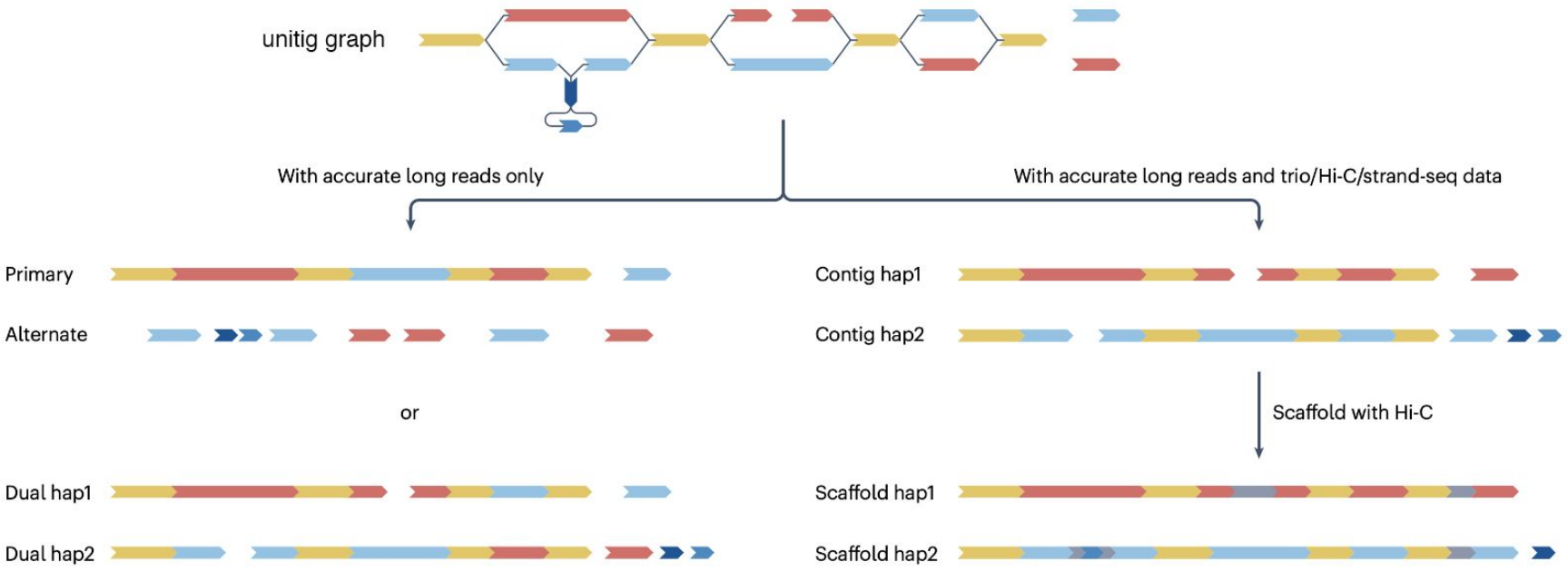
Phased assembly of diploid genomes

Some commonly used assembly tools for human genomes:
Verkko [Rautainen et al. 2023], **Hifiasm** [Cheng et al. 2021], **Flye** [Kolmogorov et al. 2020]



Hifiasm: assembly with overlap graphs
[adapted from Li & Durbin 2024]

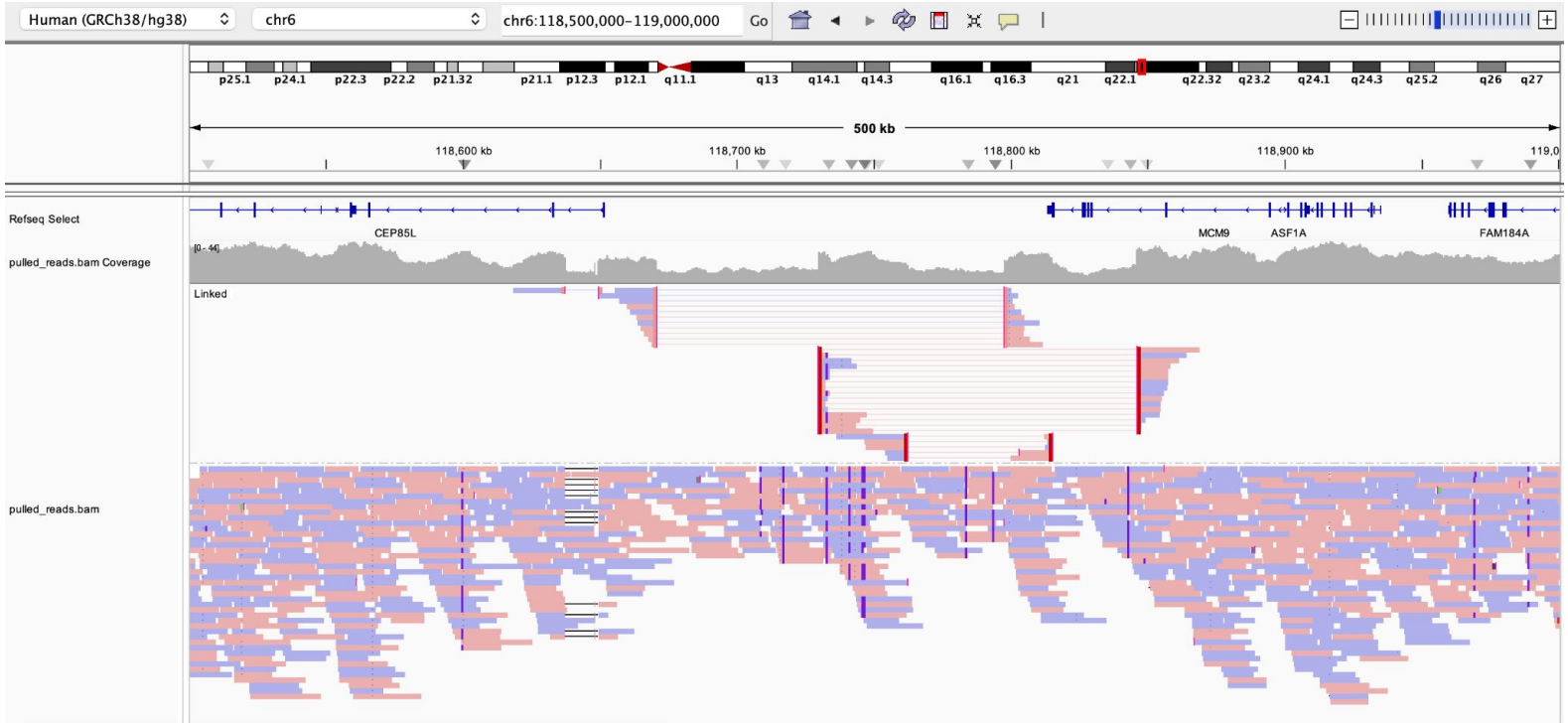
From unitigs to phased assembly



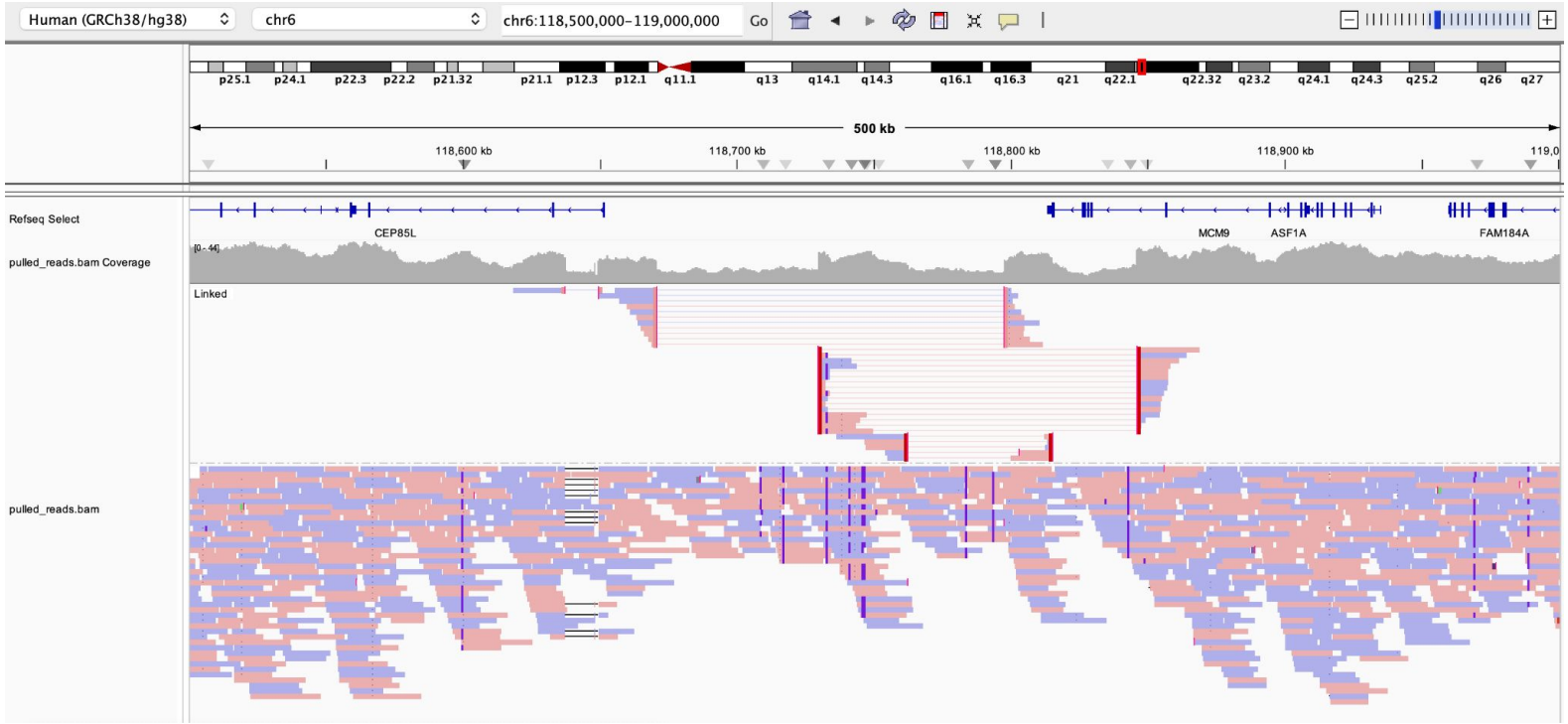
adapted from Li & Durbin 2024

Hands on: a simple local assembly

Example: HiFi-only assembly of one region



Example: HiFi-only assembly of one region




```
samtools view sample.bam "chr6:117000000-120000000" -b > reads_region.bam
samtools fastq sample_local.bam > reads_region.fastq
hifiasm reads_region.fastq -o my_assembly
```

here, we could include ONT, parental reads or Hi-C data

all code on github!

gfa: graphical fragment assembly files

hifiasm outputs:

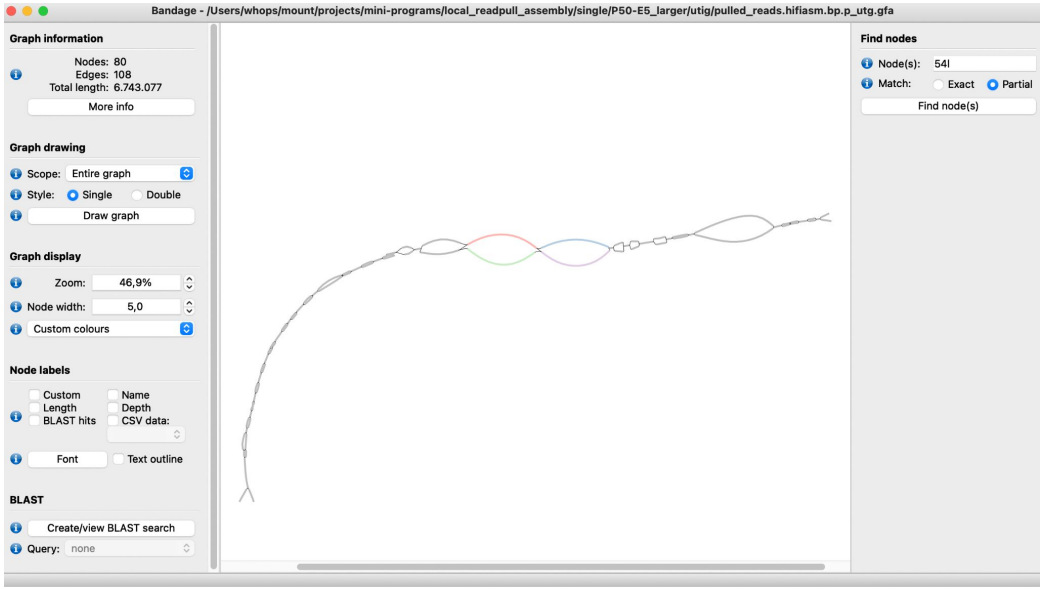
```
"main" .gfa output files
my_assembly.bp.hap1.p_ctg.gfa
my_assembly.bp.hap2.p_ctg.gfa
my_assembly.bp.p_ctg.gfa
my_assembly.bp.p_utg.gfa
my_assembly.bp.r_utg.gfa

other outputs:
my_assembly.bp.hap1.p_ctg.lowQ.bed
my_assembly.bp.hap1.p_ctg.noseq.gfa
my_assembly.bp.hap2.p_ctg.lowQ.bed
my_assembly.bp.hap2.p_ctg.noseq.gfa
my_assembly.bp.p_ctg.lowQ.bed
my_assembly.bp.p_ctg.noseq.gfa
my_assembly.bp.p_utg.lowQ.bed
my_assembly.bp.p_utg.noseq.gfa
my_assembly.bp.r_utg.lowQ.bed
my_assembly.bp.r_utg.noseq.gfa
my_assembly.ec.bin
my_assembly.ovlp.reverse.bin
my_assembly.ovlp.source.bin
```

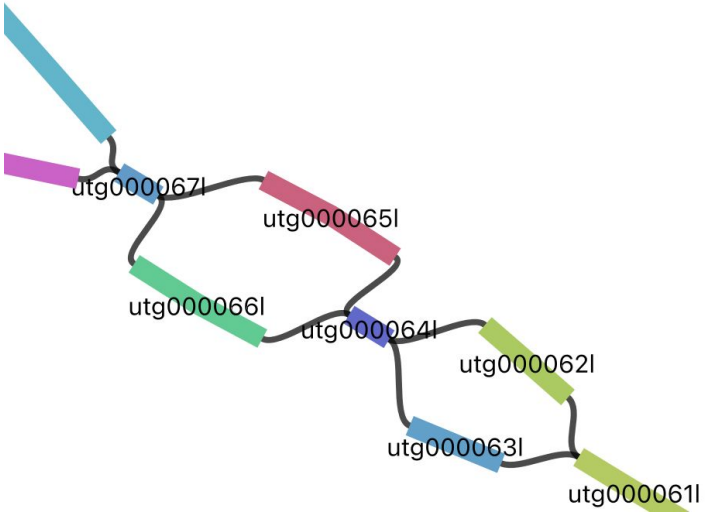
```
S       utg000001l      TCCTGCACACCTATTAAAATGGCTAATACTAAAAAGAATGACCTAGGAATTATTGGAGAGAATGTGAAACAATTGGAAACCTCATACATTGCTGGTG
S       utg000002l      AATACTATTCAGCCATAAAAAGAATGATATCATATTCTTTGCAGCAACAAAGATGGAACTGGAGGCCATTATCCTAAGCTAACTAACACCGGGGCAG
S       utg000003l      CTGTTGTGCTAGCAAATACTAGGTCTTATTCATTCTTCCTAACTATTTTTTTGTACCCATTAACCATCCATCACTTCCTTCCCACCAATCCCCCACT
S       utg000004l      ACGATATTGATTCTTCCTACCCATGAGCATGGAATGTTCTTCCATTTGTTTGTATCCTCTTTTATTTCATTGAGCAGGGGTTTGTAGTTCTCCTTGA
S       utg000005l      AGAAAATCCAGCCCGTGCCATCTTCCTCTGATTATTCCAGGGAAGACCTAGGCTCACATTAGCAGGGCTTAGCTGCTTCCCGAGAGCAAACAGGCGA
S       utg000006l      TGGTAGATTTTAGAATAAGTACCATGTGGCACTCAGAAGAATGTATATTCTGTTGATTTGGGCTAGAAAGTTCTGTAGACATCTACTAGGTCCACTT
S       utg000007l      GAGGAACTGGTACCATTCCTTCTGAAACTATTCCAATCAATAGAAAAAGAGGGAATCCTCCCTAACTCACTTTATGAGGCCAGCATCATTCTGATAC
S       utg000008l      GTTCCCATCATTTTGTTAGGCCTCACGATTAAATATGAAACCAGCCAAGAATACTAAACATTTGAGAAAGATTCTCACATTGAATCAGATTCCAAAA
S       utg000009l      GGGAACTCAGAATCTTCTGTAGGTGAATAACTGATATCTAAATTTAATGTTTTGGGGAAAAGATATTTTAAAAAAGATACTAGCCATATCATTACAT
S       utg000010l      TGTCATTTTGGGTGTTAAAAATAAGTCAAGCGGACTTAAAACTTCTTACCCATACCAGGAGAAAATTATTTCAGAGCTACCTCATCAGATGTGCCTC
[...]
L       utg000001l      +       utg000003l      +       19822M  L1:i:85183      L2:i:0
L       utg000002l      +       utg000003l      +       15094M  L1:i:82821      L2:i:0
L       utg000003l      +       utg000004l      -       20551M  L1:i:153718     L2:i:0
L       utg000003l      +       utg000005l      -       9994M   L1:i:164275     L2:i:0
L       utg000003l      -       utg000001l      -       19822M  L1:i:154447     L2:i:0
L       utg000003l      -       utg000002l      -       15094M  L1:i:159175     L2:i:0
L       utg000004l      +       utg000003l      -       20551M  L1:i:9079       L2:i:0
L       utg000004l      -       utg000006l      -       16976M  L1:i:12654      L2:i:0
L       utg000004l      -       utg000007l      +       983M    L1:i:28647      L2:i:0
L       utg000005l      +       utg000003l      -       9994M   L1:i:26037      L2:i:0
L       utg000005l      -       utg000007l      +       17941M  L1:i:18090      L2:i:0
L       utg000006l      +       utg000004l      +       16976M  L1:i:83417      L2:i:0
L       utg000006l      -       utg000008l      +       17697M  L1:i:82696      L2:i:0
L       utg000007l      +       utg000008l      +       11045M  L1:i:66696      L2:i:0
L       utg000007l      -       utg000005l      +       17941M  L1:i:59800      L2:i:0
L       utg000007l      -       utg000004l      +       983M    L1:i:76758      L2:i:0
L       utg000008l      +       utg000009l      -       17683M  L1:i:1300       L2:i:0
[...]
```

S: Segment / Sequence
L: Link

Assembly graph visualization with *Bandage*

bandage -> load graph…. -> my_assembly.p_utig.gfa



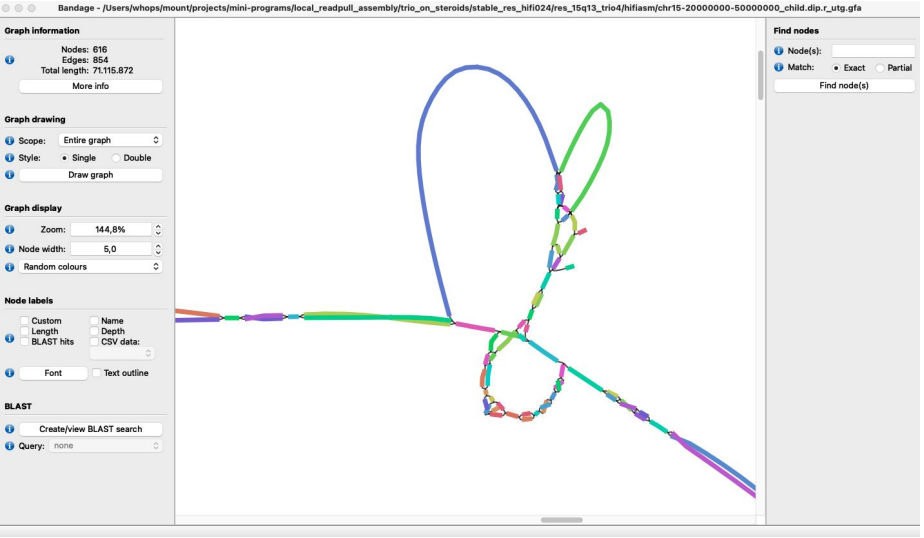


nodes: Segments (“utigs”)
connections: Links

Excursion: interpreting assembly (unitig / contig) graphs

Excursion: Understanding assembly graphs (.gfa)

HiFi reads; 5Mbp repeat-rich region

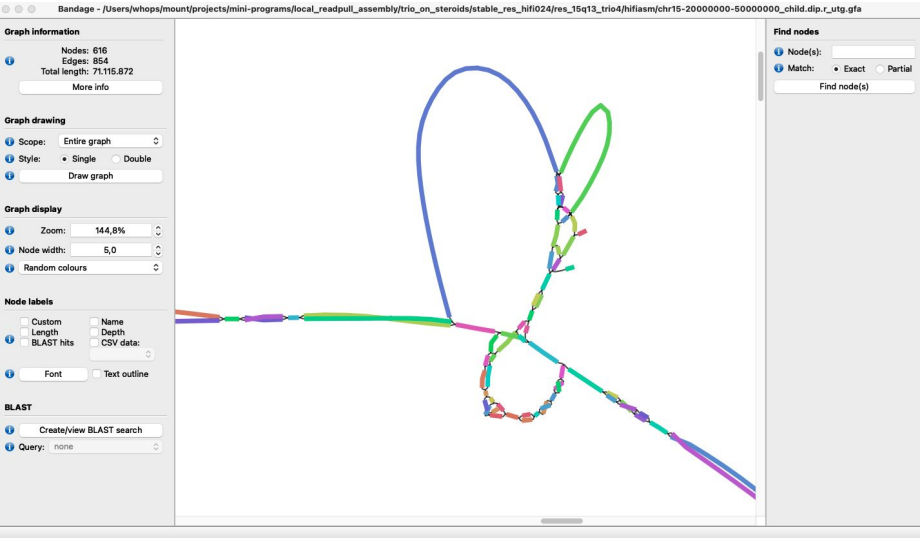


Adverse signs:

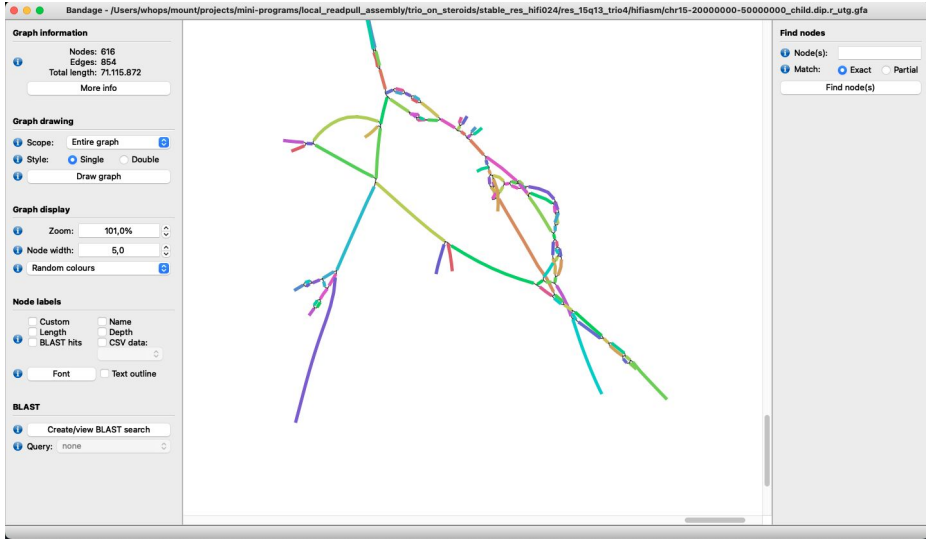
- branching / tangles
- ‘tips’: loose ends

Excursion: Understanding assembly graphs (.gfa)

HiFi reads; 5Mbp repeat-rich region



HiFi reads; (peri) centromeric region on chr15

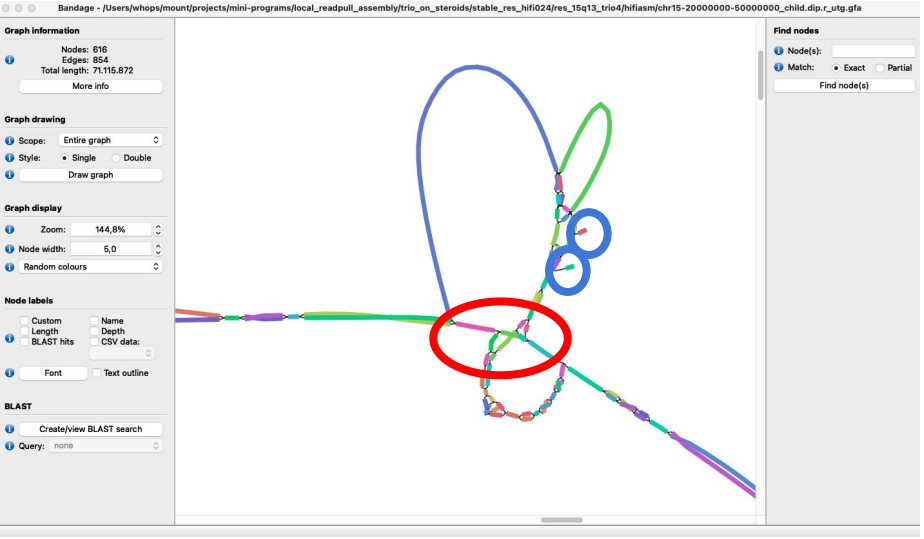


Adverse signs:

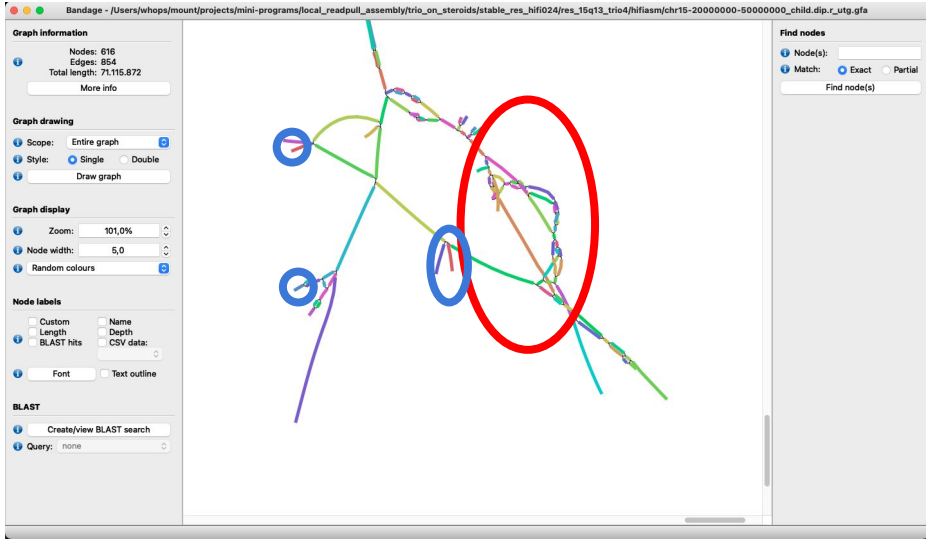
- branching / tangles
- ‘tips’: loose ends

Excursion: Understanding assembly graphs (.gfa)

HiFi reads; 5Mbp repeat-rich region



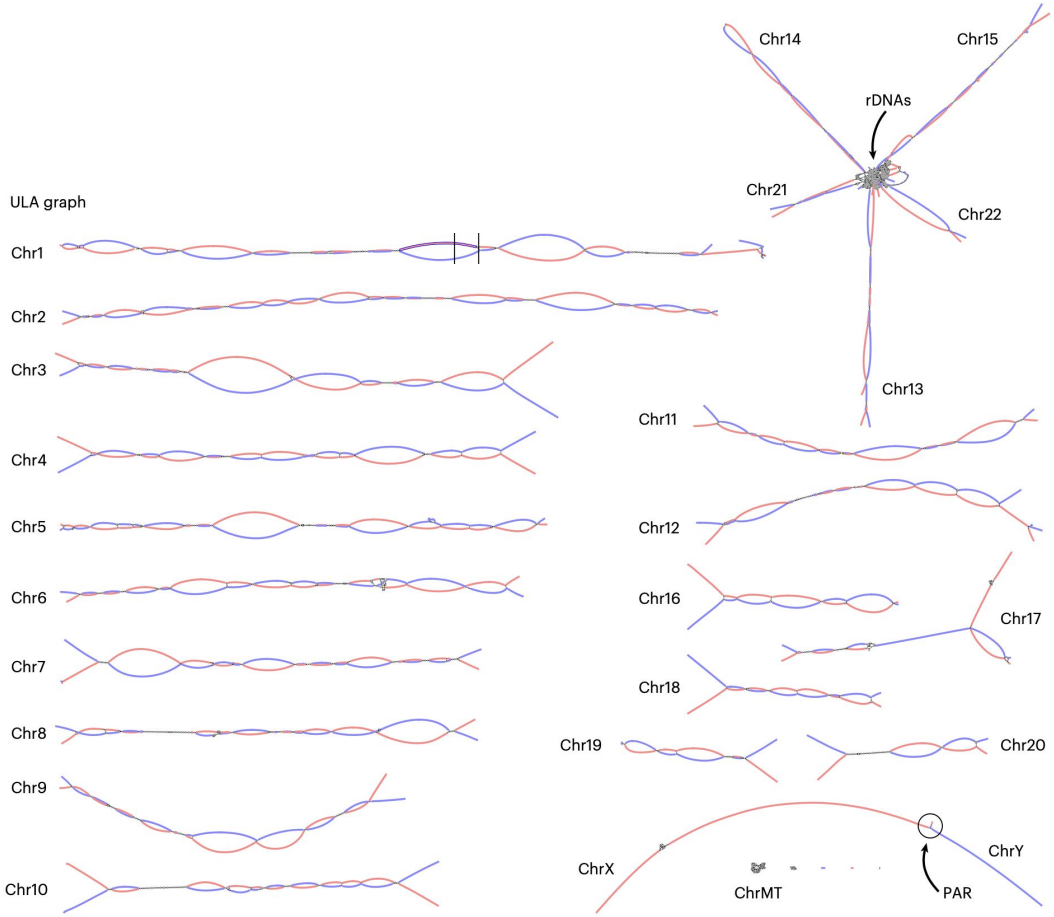
HiFi reads; (peri) centromeric region on chr15



Adverse signs:

- branching / tangles
- ‘tips’: loose ends

Excursion: Verkko assembly with >100X Hifi + ~80X ONT UL

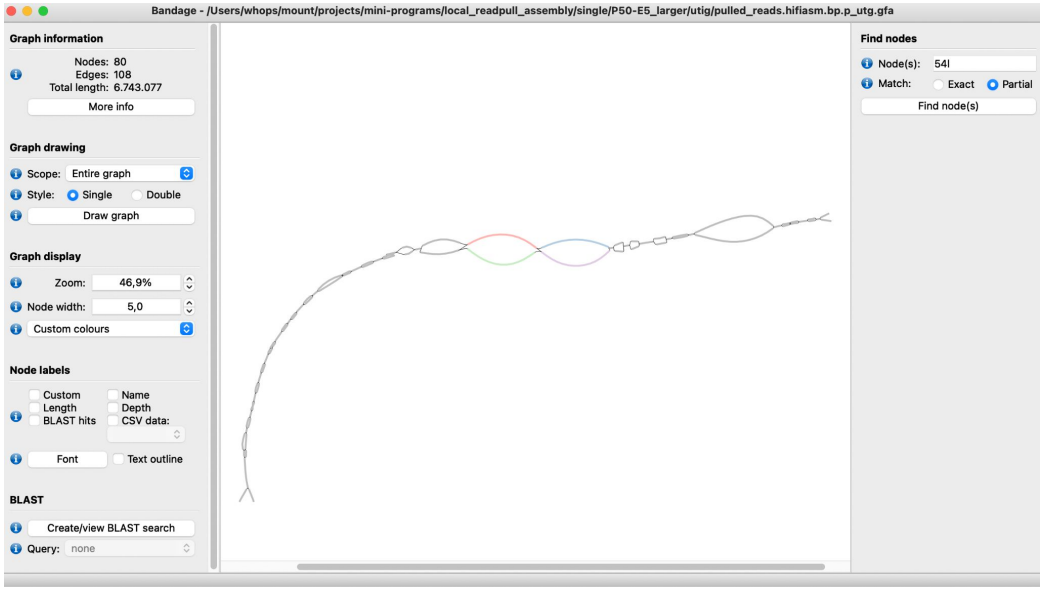


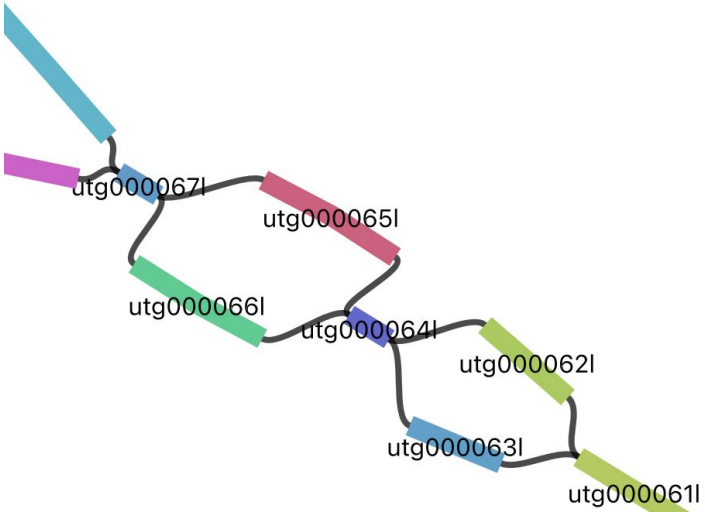
adapted from Rautiainen et al. 2023

Excursion over

Assembly graph visualization with *Bandage*

bandage -> load graph…. -> my_assembly.p_utig.gfa

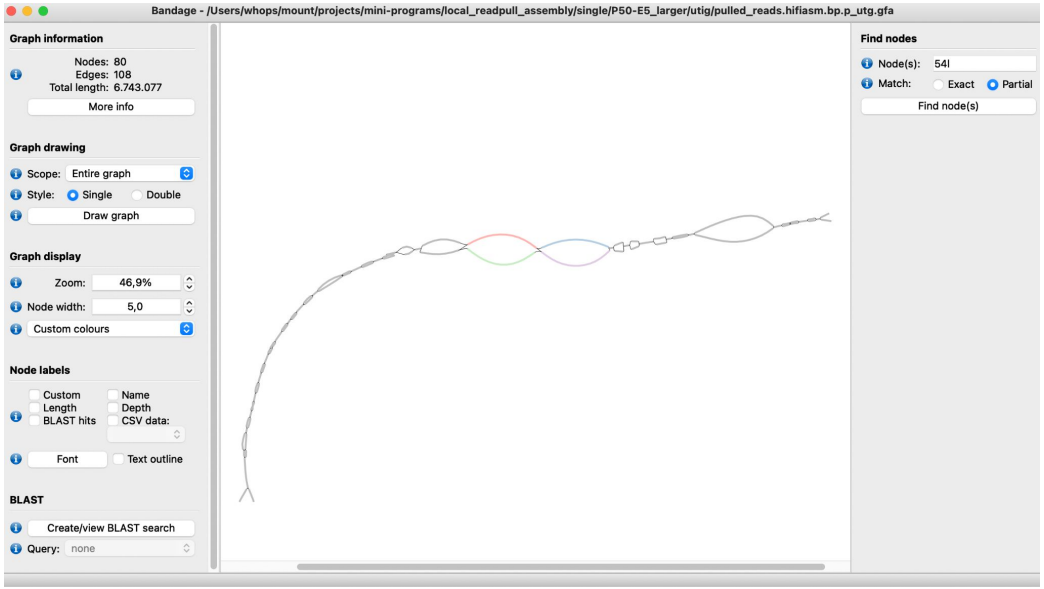


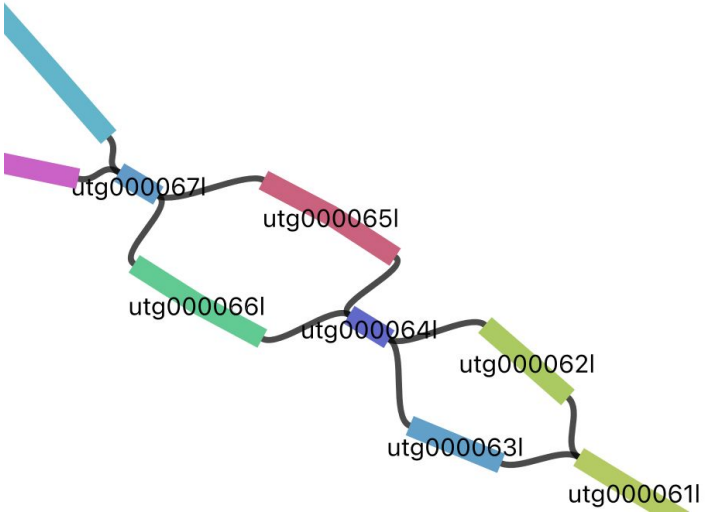


nodes: Segments (“utigs”)
connections: Links

Assembly graph visualization with *Bandage*

bandage -> load graph…. -> my_assembly.p_utig.gfa



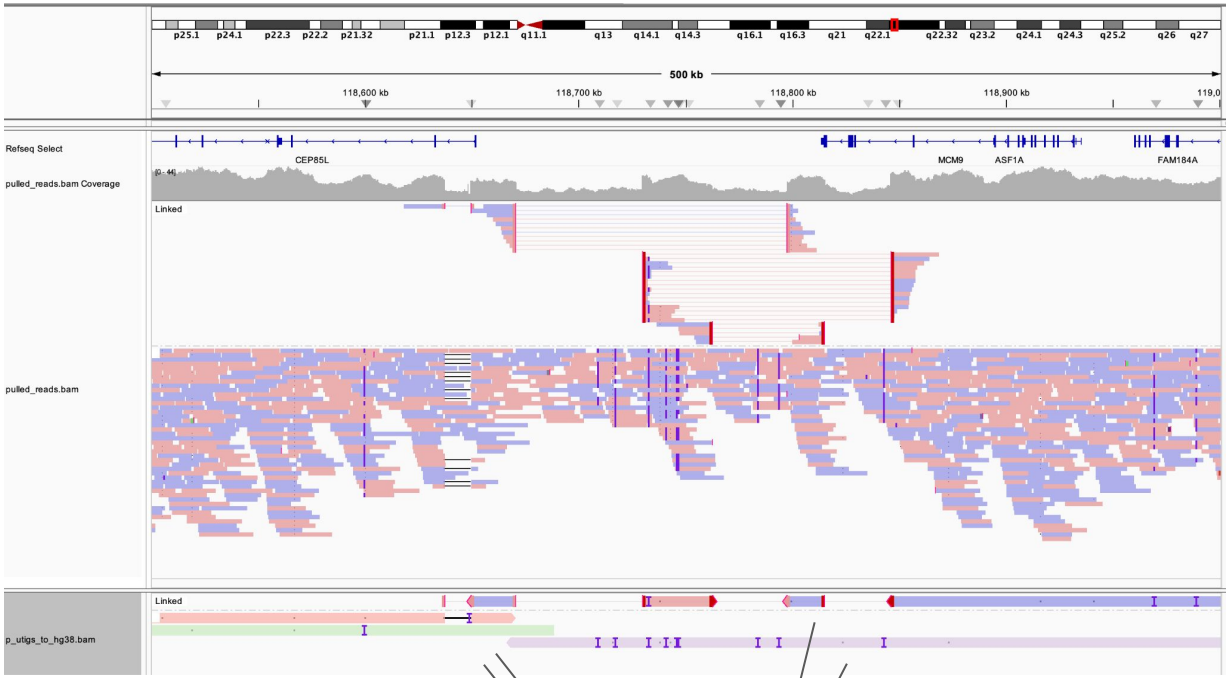


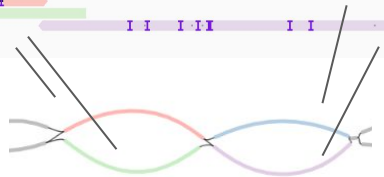
nodes: Segments (“utigs”)
connections: Links

no tips or tangles

manipulating gfa files - mapping segments back to the reference

```
gfatools gfa2fa unitigs.gfa > unitigs.fa
minimap2 -x asm5 -a hg38.fa unitigs.fa > aligned_utigs.sam
```

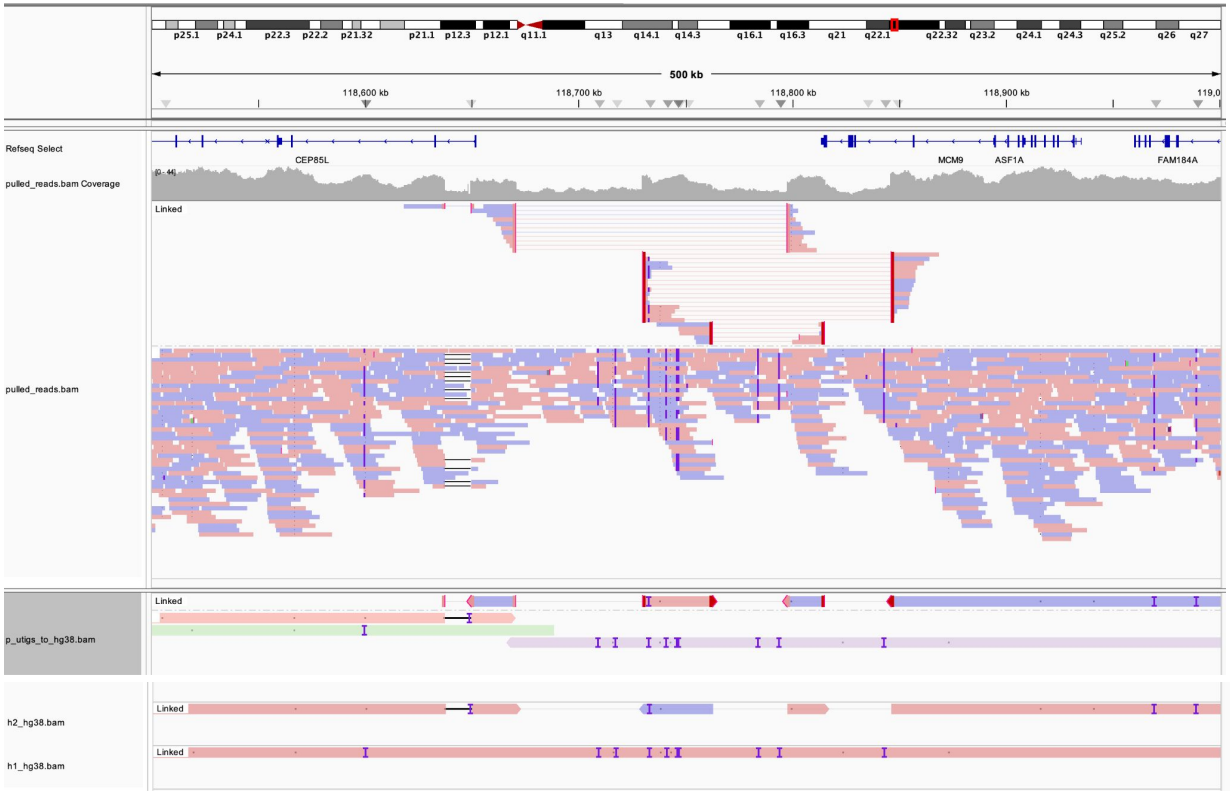




unitigs.gfa

… and this is what asm_hap1 and asm_hap2 in our case are

```
gfatools gfa2fa unitigs.gfa > unitigs.fa
minimap2 -x asm5 -a hg38.fa unitigs.fa > aligned_utigs.sam
```

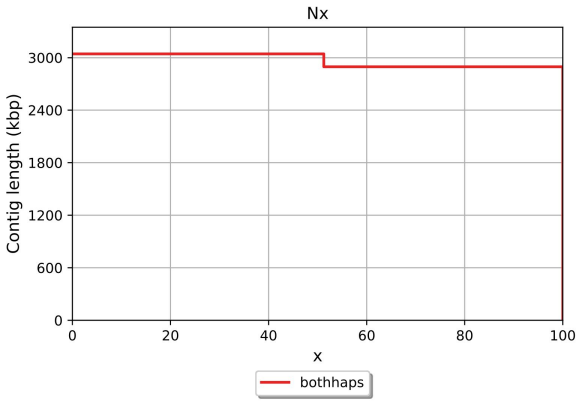


Assembly assessment

(1) Contiguity and Correctness with *Quast*
(2) k-mer based evaluation with *Merqury*
(3) Completeness with *BUSCO*

Assembly assessment

(1) Contiguity and Correctness with *Quast*
(2) k-mer based evaluation with *Merqury*
(3) Completeness with *BUSCO*



Report

	bothhaps
# contigs (>= 0 bp)	2
# contigs (>= 1000 bp)	2
# contigs (>= 5000 bp)	2
# contigs (>= 10000 bp)	2
# contigs (>= 25000 bp)	2
# contigs (>= 50000 bp)	2
Total length (>= 0 bp)	5939444
Total length (>= 1000 bp)	5939444
Total length (>= 5000 bp)	5939444
Total length (>= 10000 bp)	5939444
Total length (>= 25000 bp)	5939444
Total length (>= 50000 bp)	5939444
# contigs	2
Largest contig	3044186
Total length	5939444
GC (%)	38.20
N50	3044186
N75	2895258
L50	1
L75	2
# N's per 100 kbp	0.00

quast.py <(cat asm_hap1.fa asm_hap2.fa)

Gurevich et al. 2013

Assembly assessment

(1) Contiguity and Correctness with *Quast*
(2) k-mer based evaluation with *Merqury*
(3) Completeness with *BUSCO*

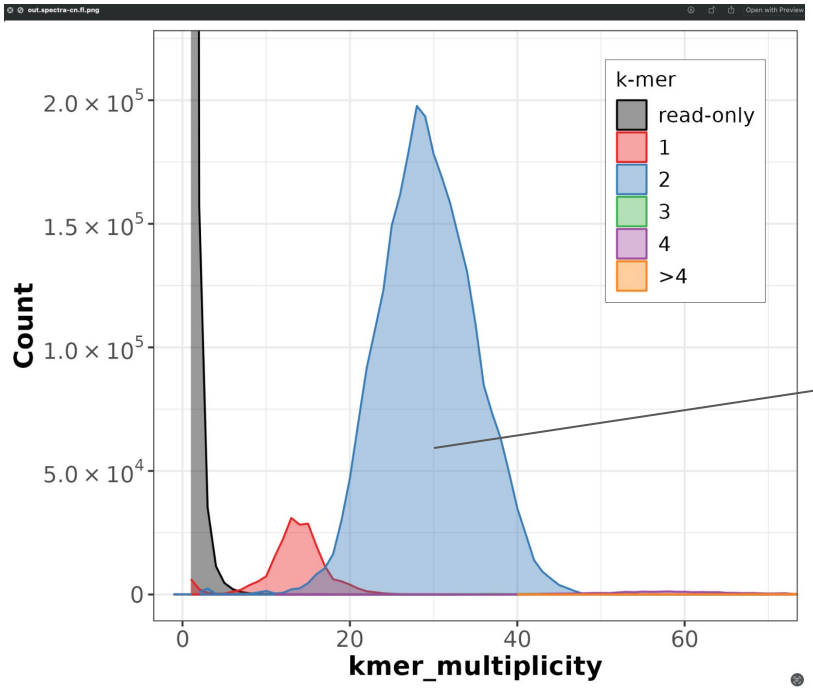
```
meryl count k=21 ../../../pulled_reads.fastq.gz output meryl_read_counts
$MERQURY/merqury.sh meryl_read_counts asm_hap1.fa asm_hap2.fa
```

```
$ cat out.qv
label     erroneous-kmers  total-kmers  QV       Error-rate
asm_hap1  15               3044166      66.296   2.34641e-07
asm_hap2  34               2895238      62.5242  5.59214e-07
Both      49               5939404      64.0576  3.92858e-07
```

Rhie et al. 2020

Assembly assessment

(1) Contiguity and Correctness with *Quast*
(2) k-mer based evaluation with *Merqury*
(3) Completeness with *BUSCO*



"k-mers which appear twice on our assembly: usually appear ~30X in the raw reads"

Rhie et al. 2020

Completeness: how many expected genes are Complete, Duplicated, Fragmented, or Missing?

(1) Contiguity and Correctness with *Quast*
(2) k-mer based evaluation with *Merqury*
(3) Gene completeness with *BUSCO (or its reimplementation 'compleasm')*

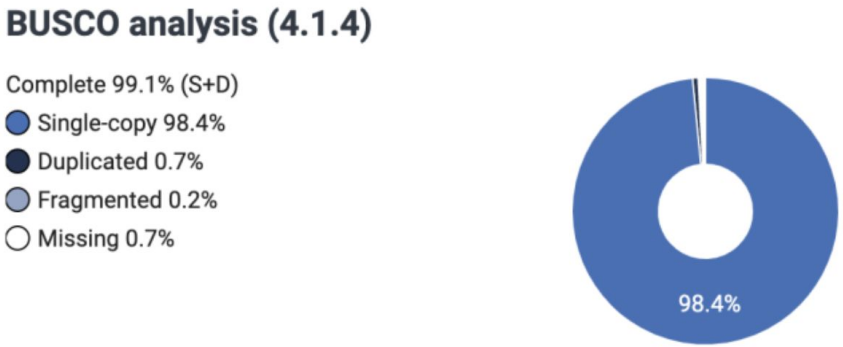
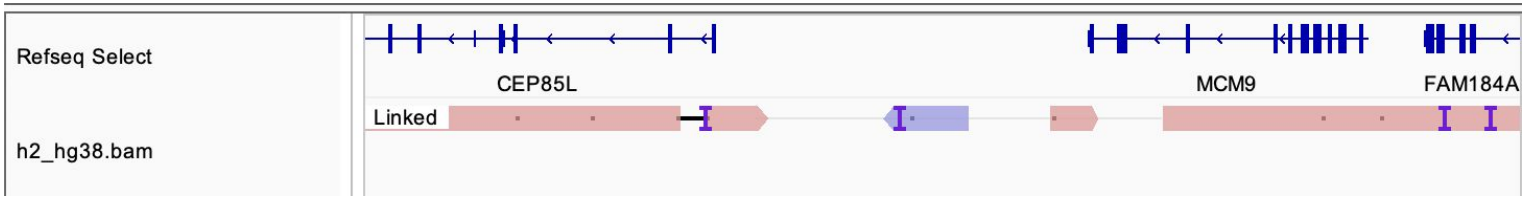


Figure: Genome institute at Washington University School of Medicine
BUSCO: Simão et al. 2015

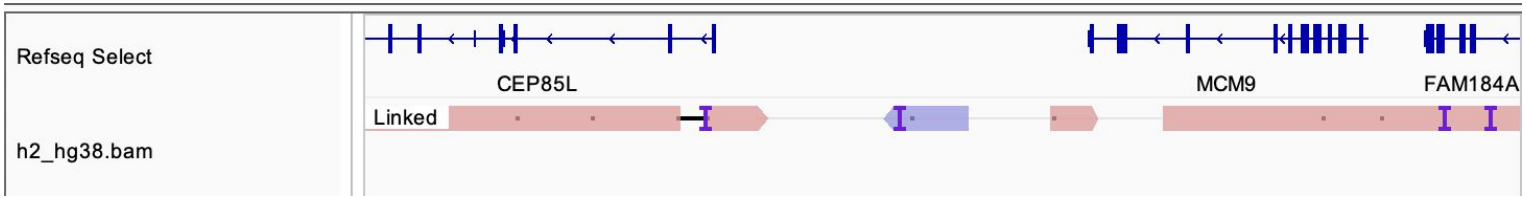
Completeness: how many expected genes are Complete, Duplicated, Fragmented, or Missing?

(1) Contiguity and Correctness with *Quast*
(2) k-mer based evaluation with *Merqury*
(3) Gene completeness with *BUSCO (or its reimplementation 'compleasm')*
(4) And many more! (deeppolisher, HMM-flagger, …)

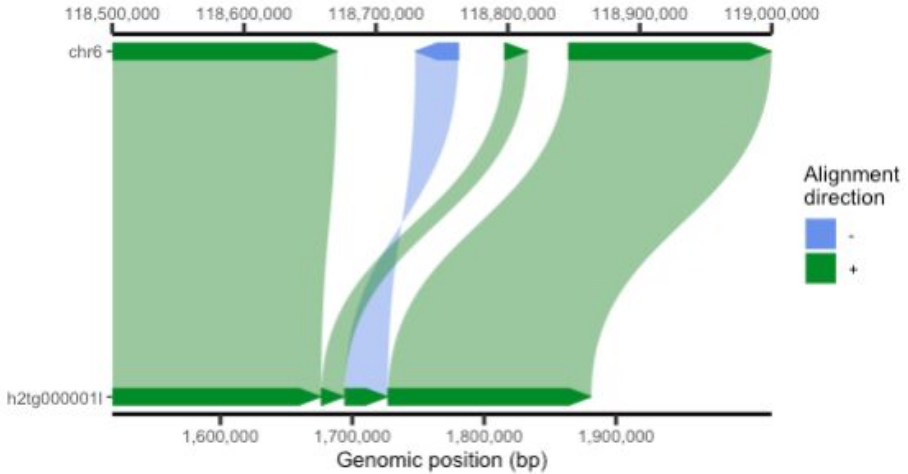
Complex regions require new visualizations



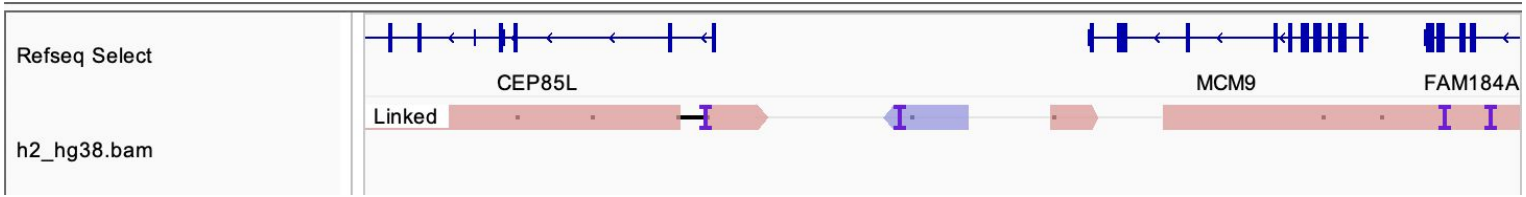
Complex regions require new visualizations



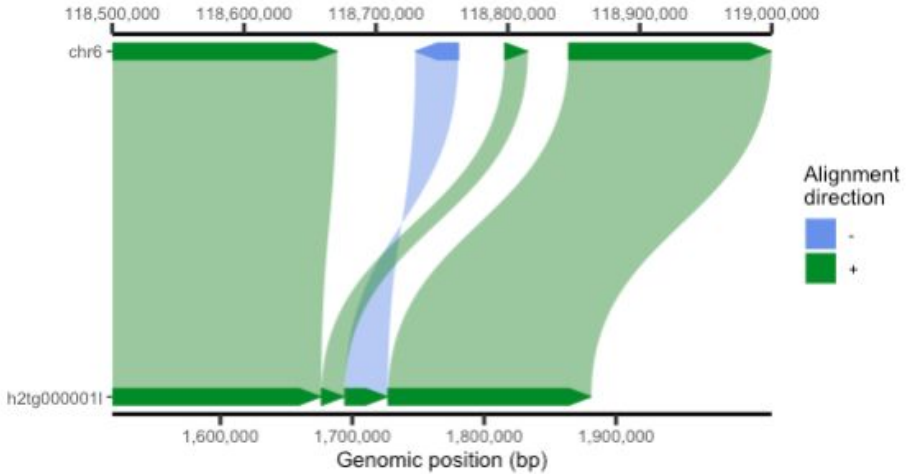
Miropeats plot [*SVbyEye*; Porubsky et al. 2025]



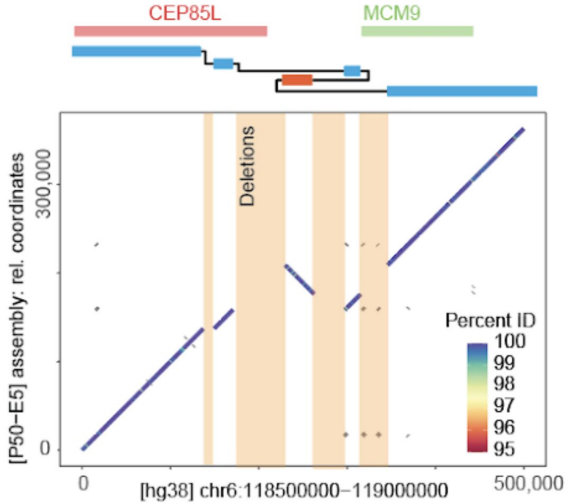
Complex regions require new visualizations



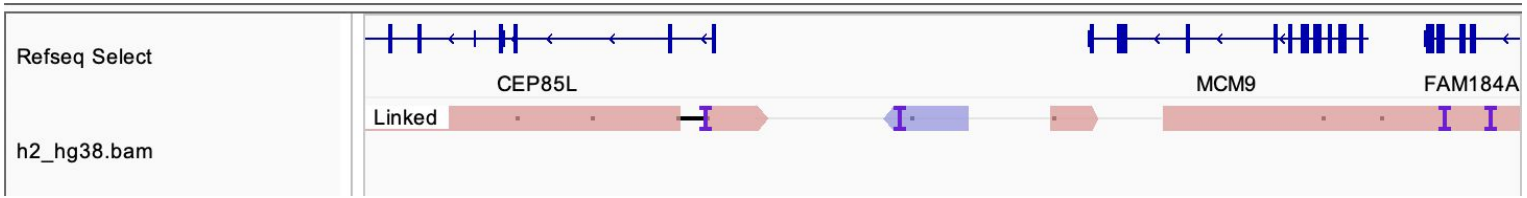
Miropeats plot [*SVbyEye*; Porubsky et al. 2025]



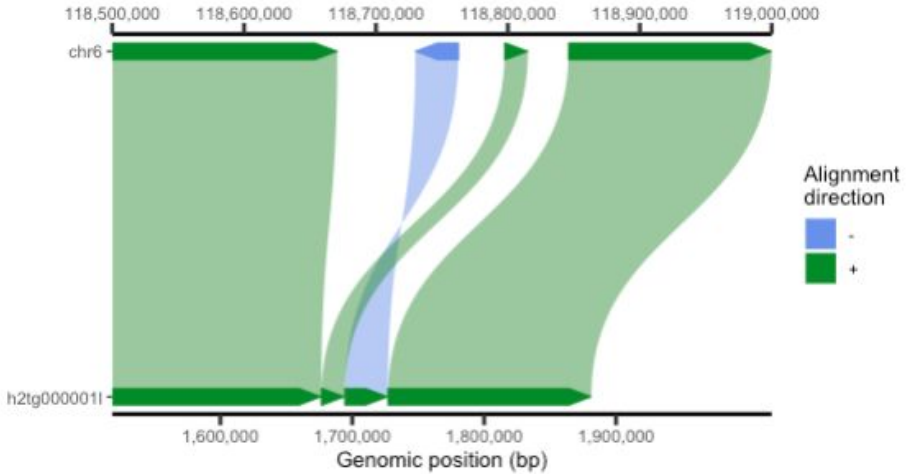
Dotplot [*NAHRWhals*; Höps et al. 2023]



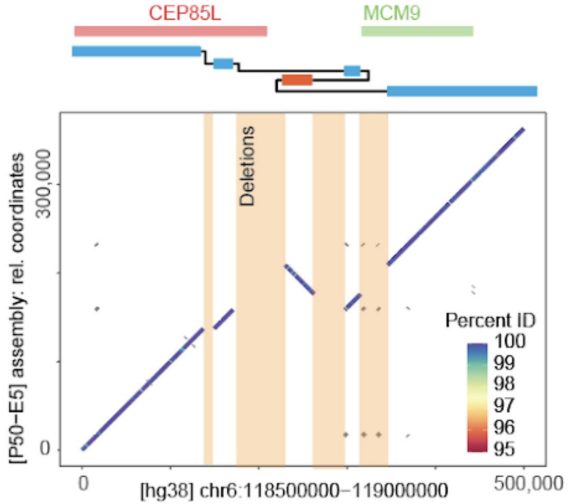
Complex regions require new visualizations



Miropeats plot [*SVbyEye*; Porubsky et al. 2025]

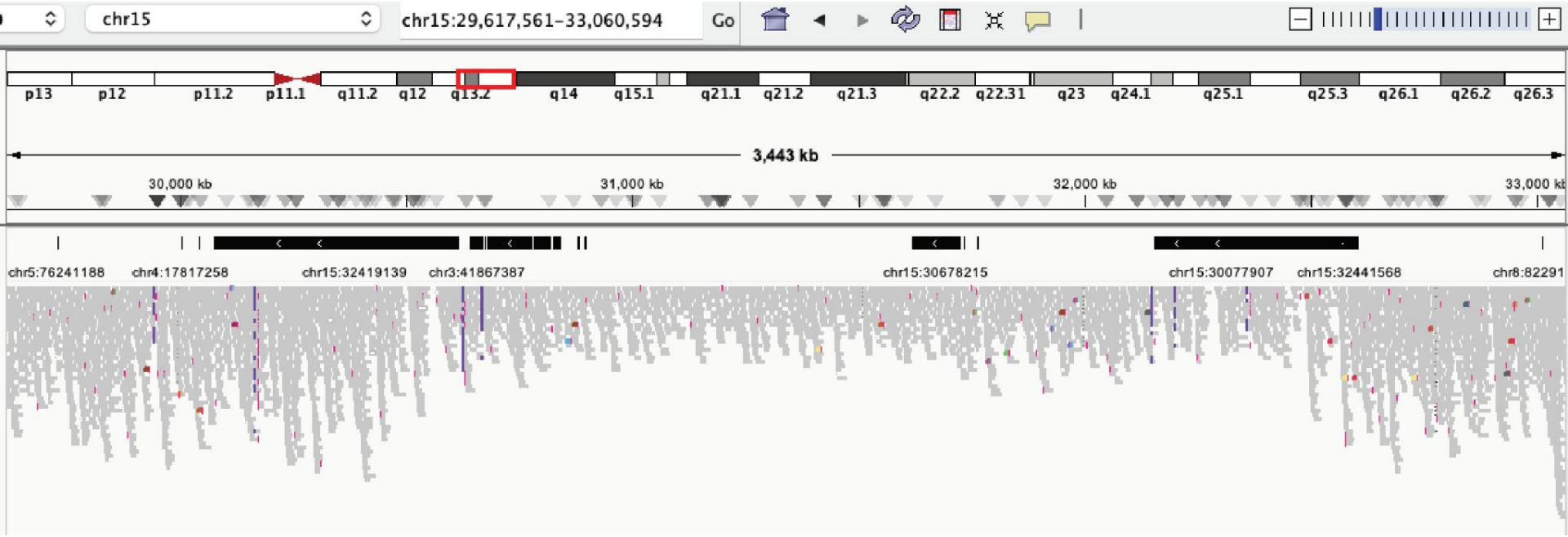


Dotplot [*NAHRWhals*; Höps et al. 2023]



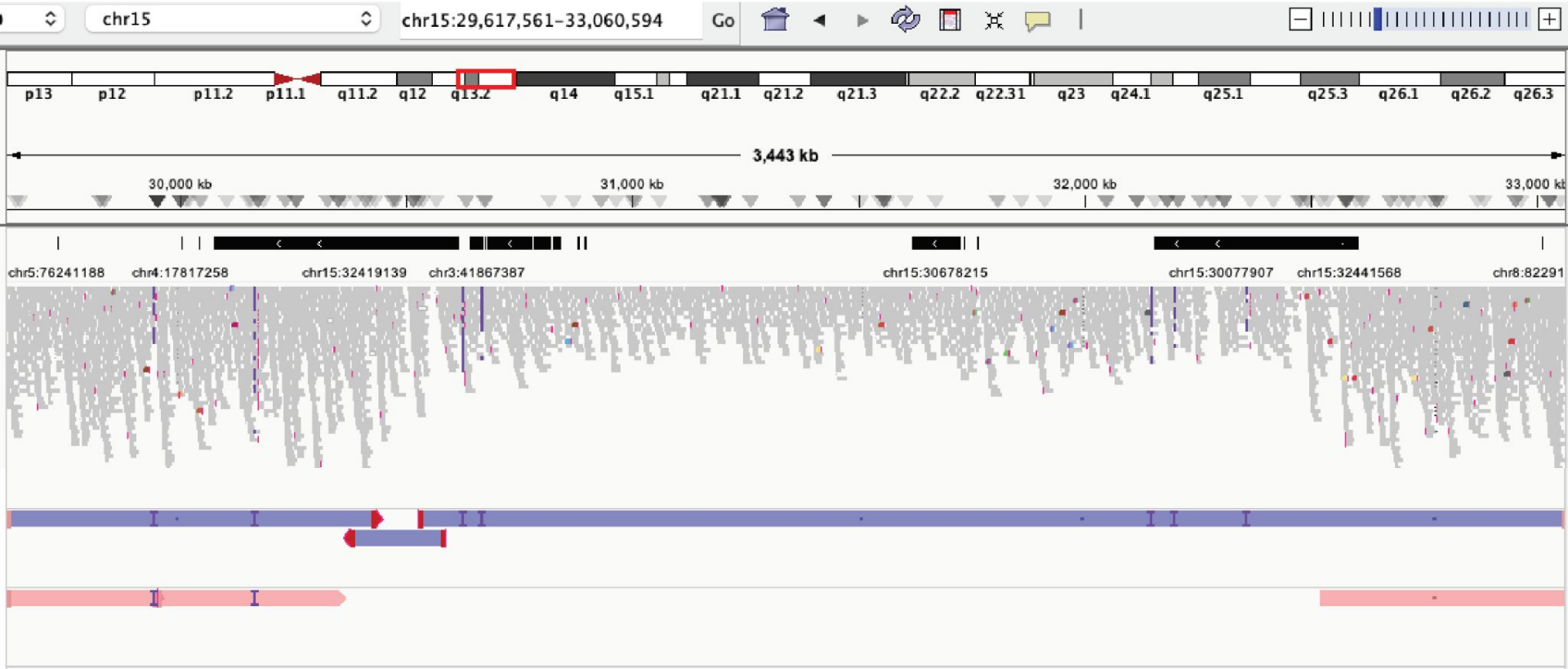
SV calling tools: e.g. *Dipcall, Smartie-sv, SVIM-asm, PAV.* [See e.g. review: Liu et al. 2024]

Example: Long duplications can obscure breakpoints



Deletion is visible, breakpoints are not.

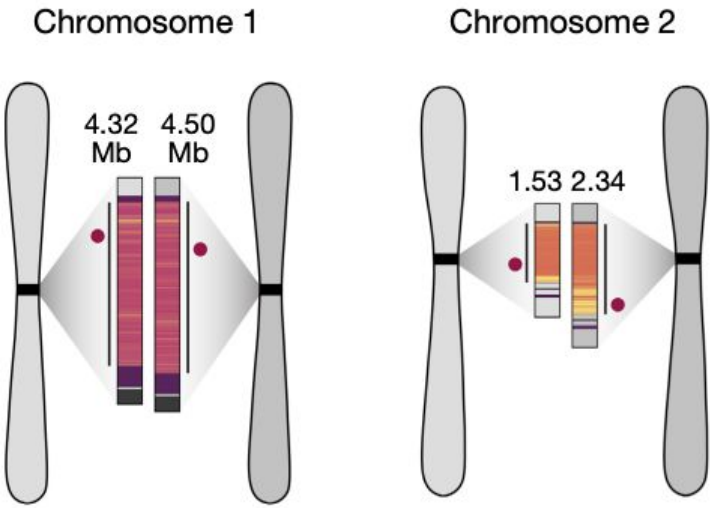
Example: Long duplications can obscure breakpoints



assembly_h1 and _h2: mapped 'back' to hg38 to reveal more detail

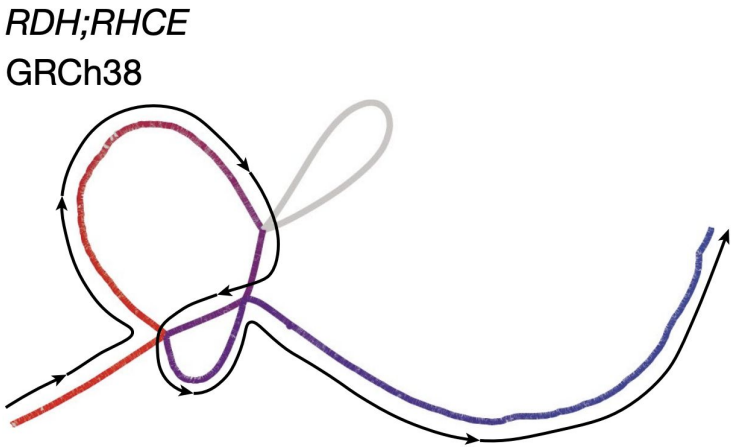
Beyond: assemblies unlock new biology

The most complex genomic loci



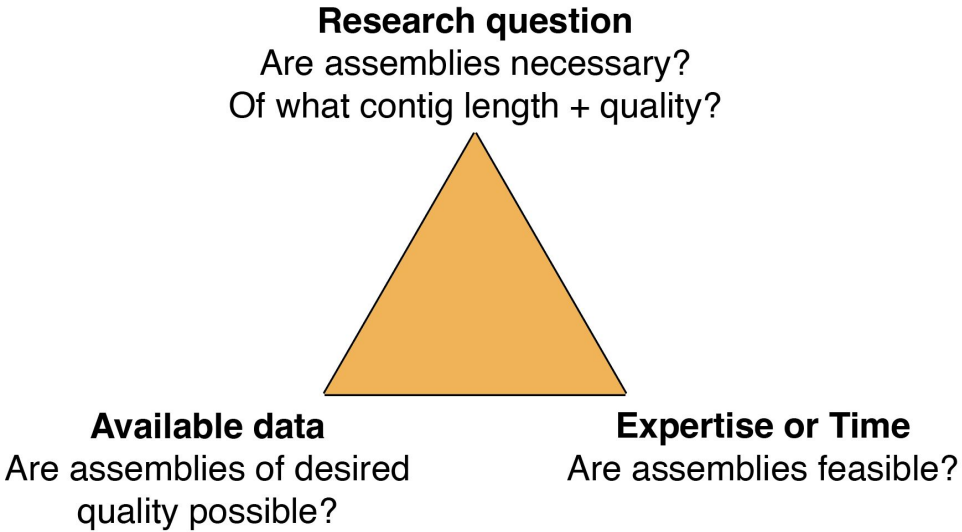
[Logsdon et al. 2024]

Pan-genome graphs to understand complex variation



[Liao et al. 2023]

Take-homes



Assemblies are only as good as the data - and always require expertise, time and careful QC.

Materials & Further reading

Genome assembly algorithms

[1] Heng Li & Richard Durbin 2025: Genome assembly in the telomere-to-telomere era; Nat. reviews genetics

[2] Rautiainen et al. 2023: Telomere-to-telomere assembly of diploid chromosomes with Verkko

[3] Kolmogorov et al. 2019: Assembly of long, error-prone reads using repeat graphs

Assembly-based variant calling

[4] Olson et al. 2023: Variant calling and benchmarking in an era of complete human genome sequences

[5] Ebler et al. 2022: Pangenome-based genome inference allows efficient and accurate genotyping across a wide spectrum of variant classes

[6] Liu et al. 2024: Tradeoffs in alignment and assembly-based methods for structural variant detection with long-read sequencing data

Tutorials

Assembly step by step:
https://training.galaxyproject.org/training-material/topics/assembly/tutorials/vgp_genome_assembly/tutorial.html

Assembly QC: (Quast, BUSCO, Mercury, Chromeister):
https://training.galaxyproject.org/training-material/topics/assembly/tutorials/assembly-quality-control/tutorial.html

https://github.com/WHops/ESHG2025_assemby_workshop