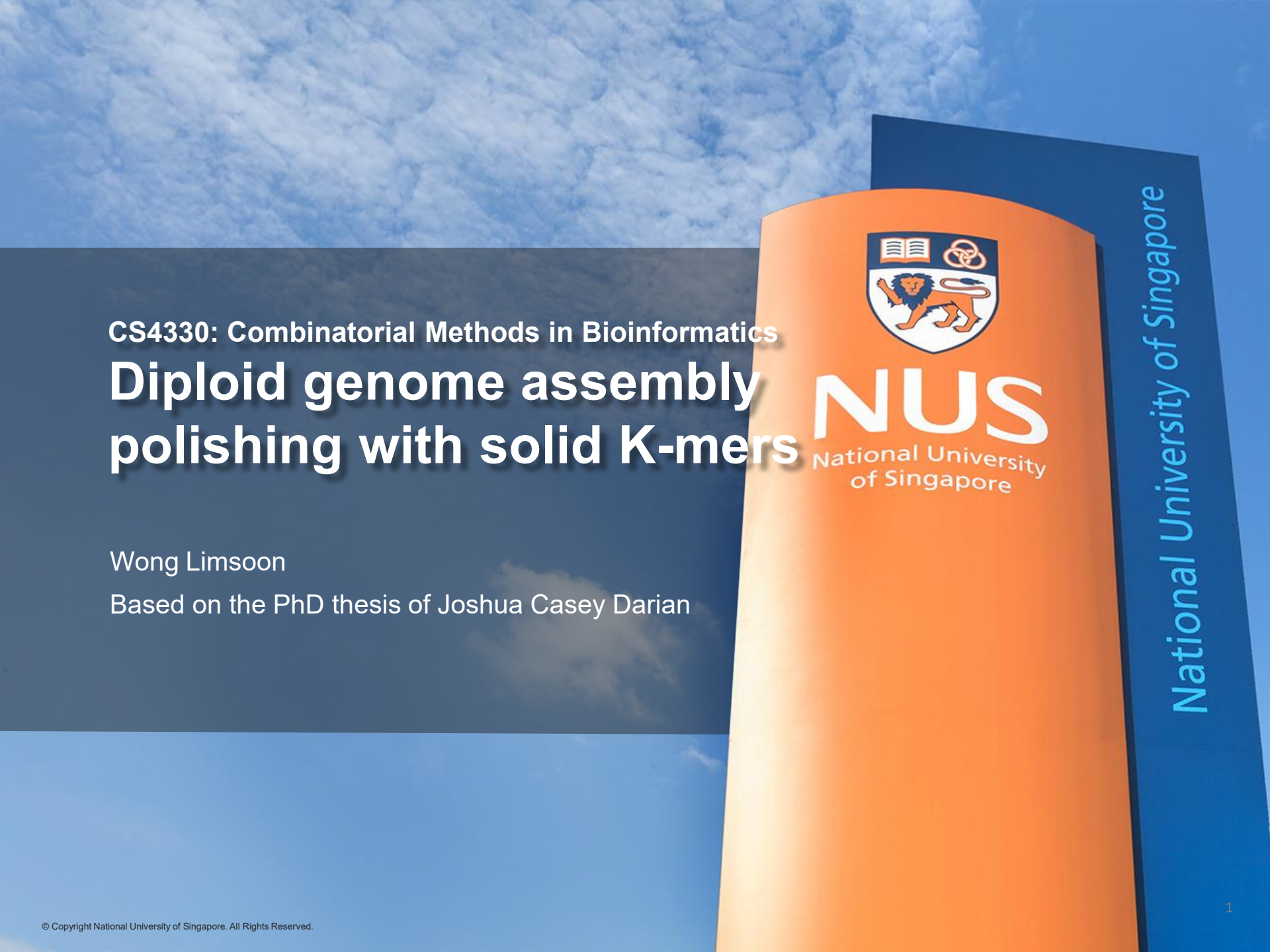


CS4330: Combinatorial Methods in Bioinformatics

Diploid genome assembly polishing with solid K-mers

Wong Limsoon

Based on the PhD thesis of Joshua Casey Darian

Genome assembly polishing

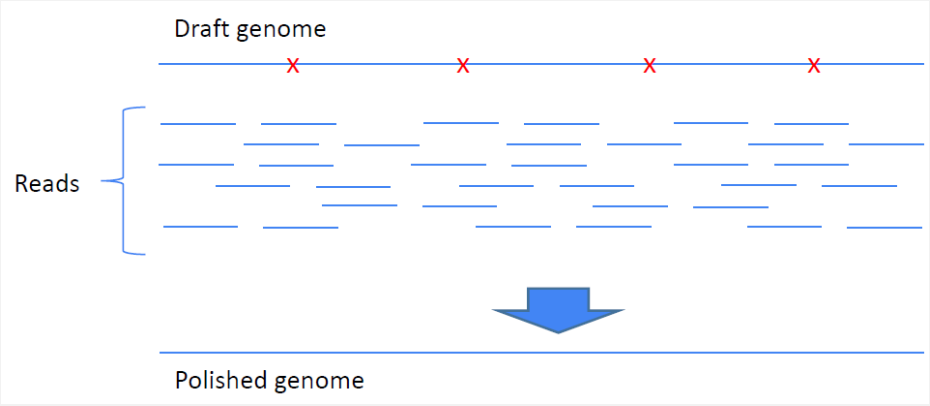
Given a set of reads, polishing reduces errors in the draft genome assembly

Existing methods polish a draft haploid genome into a polished haploid genome (i.e., not phased into diploid genome)

Two approaches

K-mer based

Alignment based



Benefits of genome assembly polishing

Normally variation calling is done by aligning an assembled genome to a reference genome

Such variation calling generates tens of thousands of false variants

Telemere-to-telomere genome assembly & polishing reduces false variants by a lot

Automated diploid assemblers & polishers

Verkko, the diploid assembler used for majority of T2T genome assembly & polishing

Long & accurate Hifi reads for base assembly

Ultra long ONT reads for contiguity

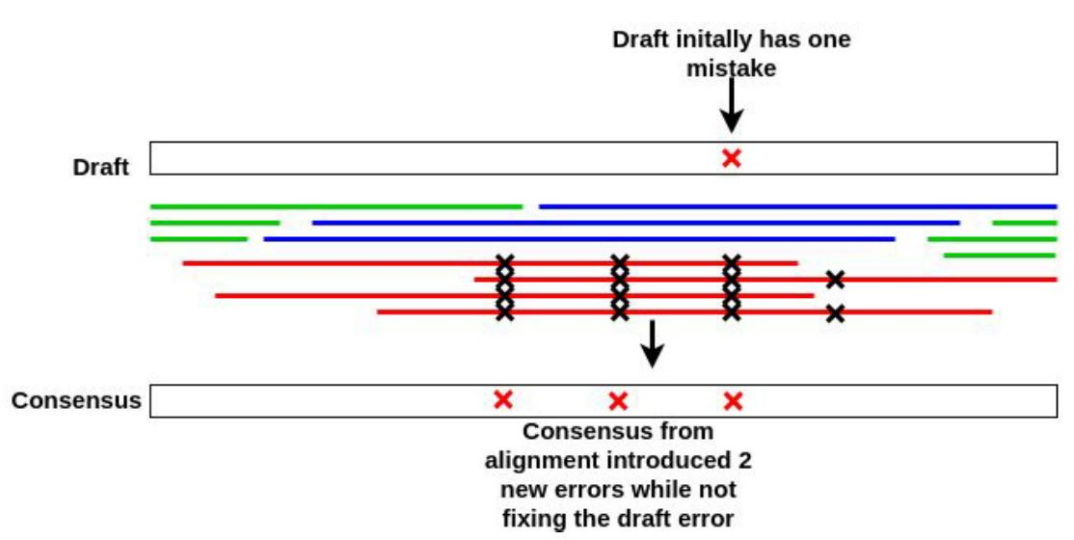
Hi-C or trio sequencing to phase the assembly

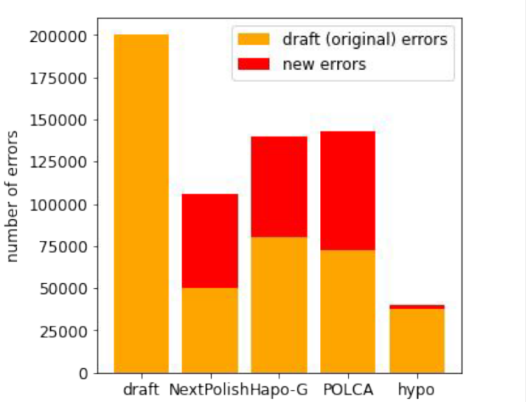
Rautiainen et al., Nature Biotech 41:1474-1482, 2023

Hifiasm: Hifi reads + Hi-C or trio data

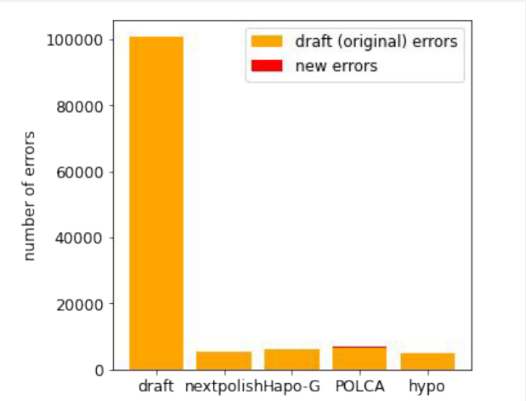
FALCOM-Phase: Any PacBio read, quality is low

Biggest mistake of alignment-based polishing





Error count in high coverage regions



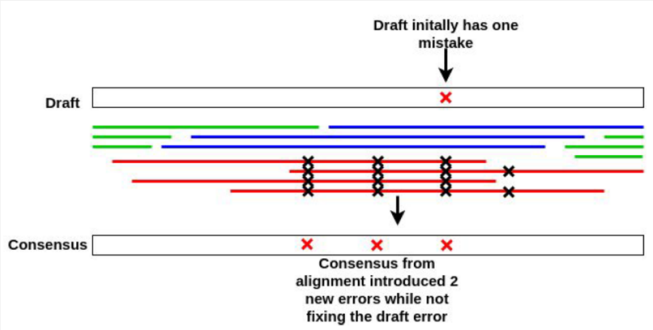
Error count in normal coverage regions

Misaligned reads in high-coverage regions

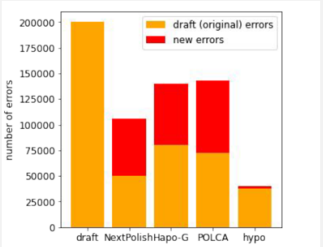
Exercise

Explain this counter intuitive observation

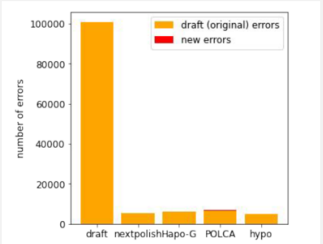
Biggest mistake of alignment-based polishing



Misaligned reads in high-coverage regions



Error count in high coverage regions

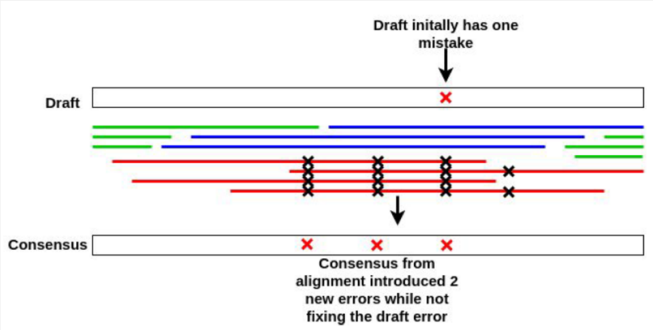


Error count in normal coverage regions

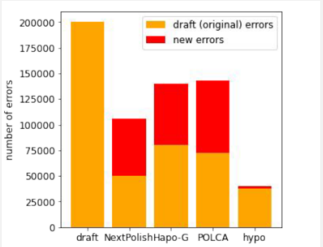
Exercise

Would down sampling improve polishing?

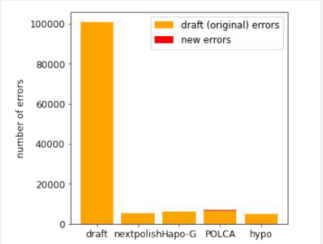
Biggest mistake of alignment-based polishing



Misaligned reads in high-coverage regions



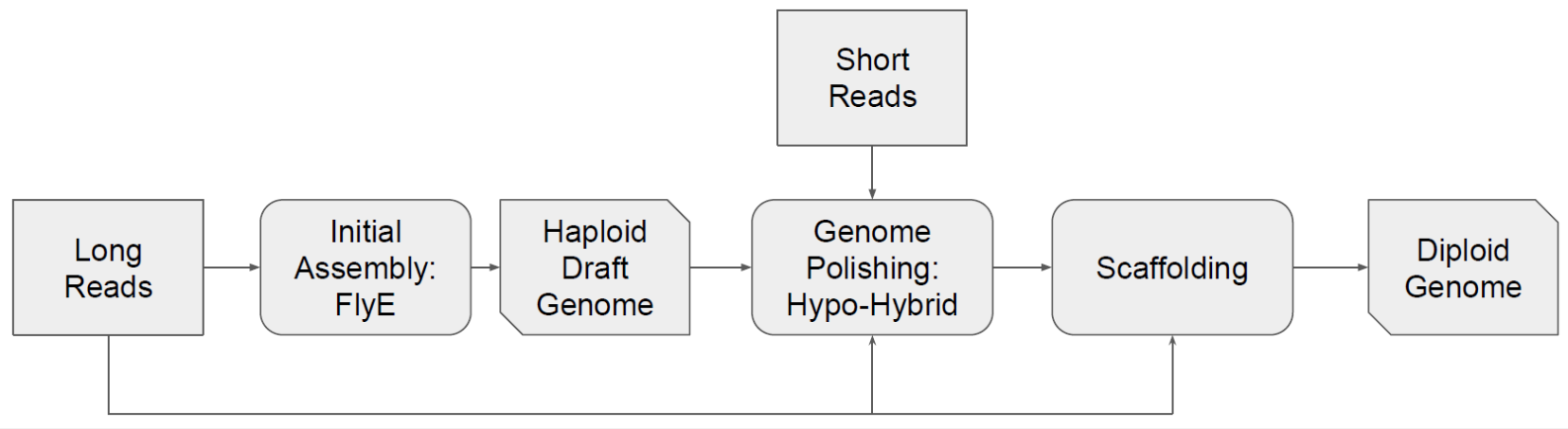
Error count in high coverage regions



Error count in normal coverage regions

Here is another way to mitigate the influence of high coverage…

Hypo-hybrid



Use cheap Illumina short reads to polish ONT assembly

Use solid K-mers instead of alignment-based polishing

Identify error-free, non-repetitive regions

Solid K-mers

Find high-confidence K-mers that occur exactly once in the genome

Solid regions

Identify error-free and repeat-free regions

In CHM13 reference, solid K-mers cover ~70% of the genome & solid regions span ~60% of the FlyE assembly

Exercise

How do find solid K-mers that occur exactly once in the genome?

Identify error-free, non-repetitive regions

Solid K-mers

Find high-confidence K-mers that occur exactly once in the genome

Solid regions

Identify error-free and repeat-free regions

In CHM13 reference, solid K-mers cover ~70% of the genome & solid regions span ~60% of the FlyE assembly

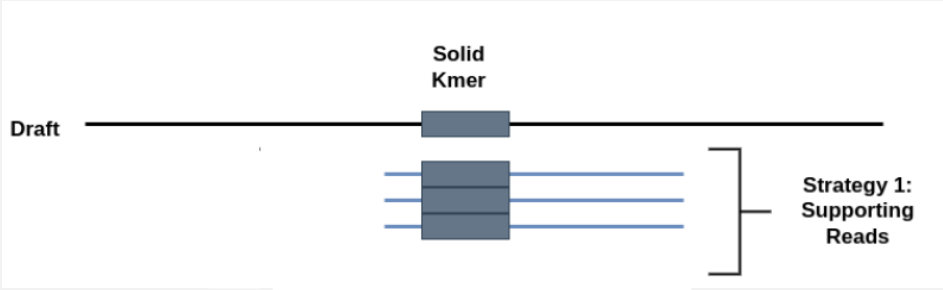
Finding solid regions

Draft assembly has lots of errors: hundreds per 100k nt

Regions covered by solid K-mers: ~10 errors per 100k nt

Some regions covered by solid K-mers are still erroneous, due to K-mers being wrongly mapped

Check if many reads that aligned to a region covered by a solid K-mer contain this solid K-mer at the aligned position



Exercise

For HG002, this removed 27% of good solid K-mers, and 98% of wrongly mapped K-mers

Suggest a better strategy

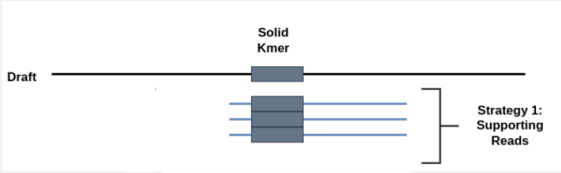
Finding solid regions

Draft assembly has lots of errors: hundreds per 100k nt

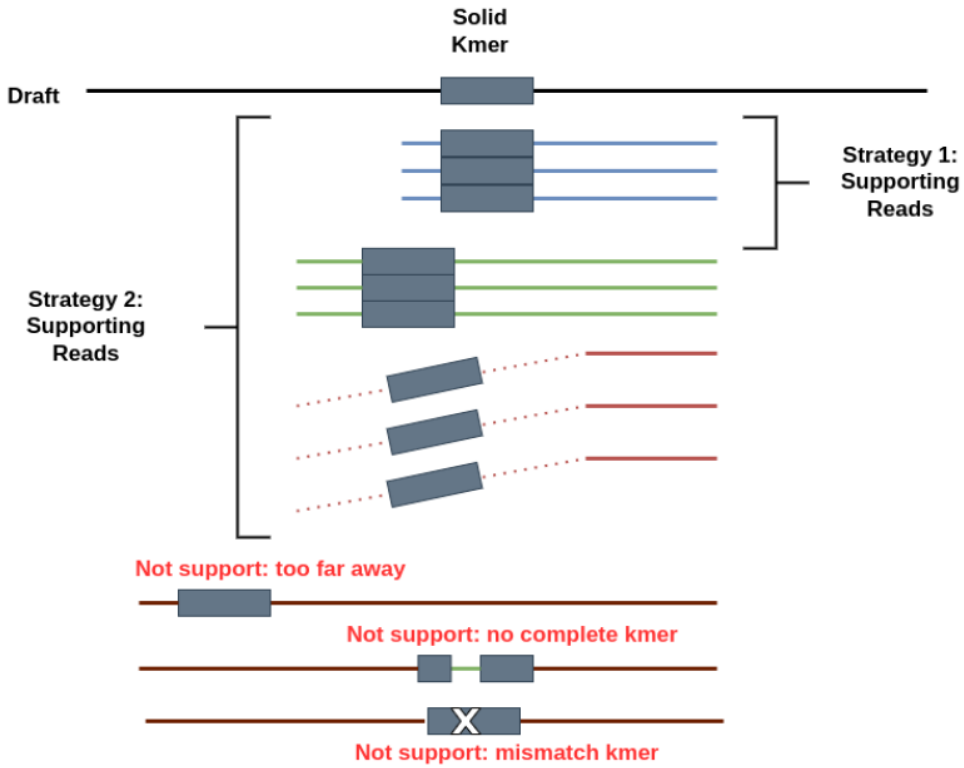
Regions covered by solid K-mers: ~10 errors per 100k nt

Some regions covered by solid K-mers are still erroneous, due to K-mers being wrongly mapped

Check if many reads that aligned to a region covered by a solid K-mer contain this solid K-mer at the aligned position



The second strategy removed similar amounts of wrongly mapped K-mers but kept a lot more good ones



Effects of filtering solid K-mers

Sample	Feature	Full Draft Assembly	Covered by Solid Kmers	Solid Regions Strategy 1	Solid Regions Strategy 2
CHM13	Size	2,711,920,893	2,055,139,644	1,498,732,251	**1,608,140,053**
	# Errors	11,840,076	277,630	1,381	1,450
	# Errors / 100kbp	436.59	**13.51**	**0.09**	0.09
HG002	Size	2,619,488,442	1,927,720,519	1,411,288,322	**1,552,277,331**
	# Errors	5,548,093	153,236	2,491	2,516
	# Errors / 100kbp	211.8	**7.95**	**0.18**	0.16

Just scanning for solid kmers in the draft is erroneous.

Filtered based on reads, solid regions are now almost error-free.

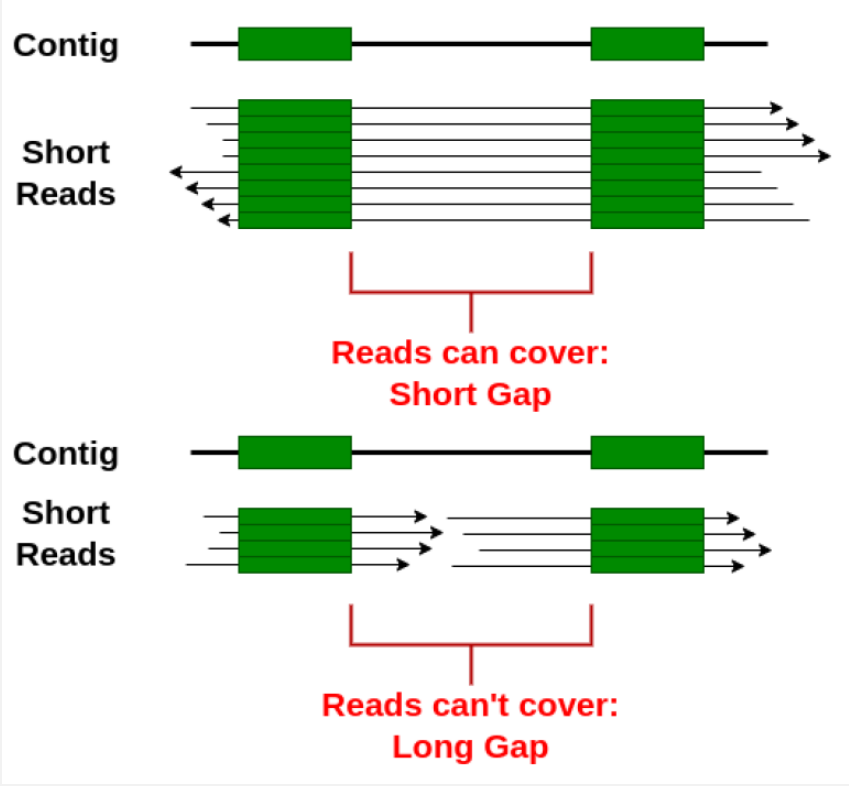
With better filtering, more solid regions can be found.

Short and long gaps

The solid regions identified earlier are assumed to be error-free and unique

Excluding them from the draft genome yields a set of remaining segments, which are called “gaps”

These gaps, both short and long, still need to be corrected



Short and long gaps

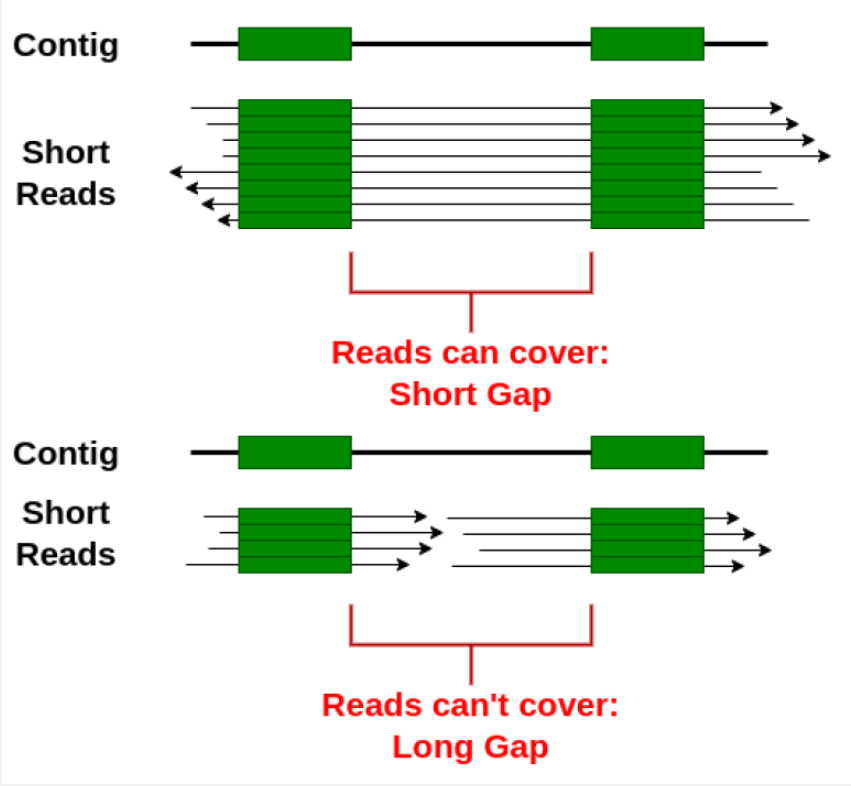
A read R covers a gap G if R contains the two solid K-mers flanking G

G is called a short gap if

G is < 100nt in length

At least 5 reads cover G

G is called a long gap otherwise

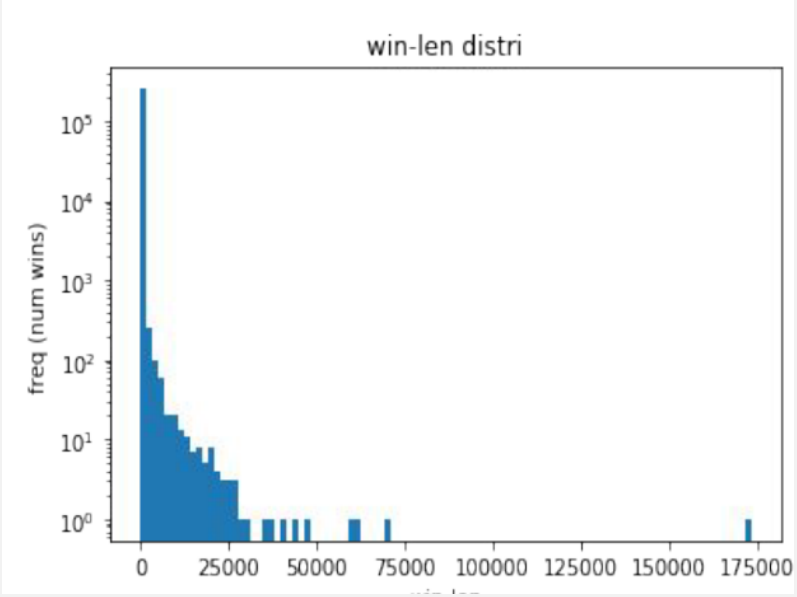


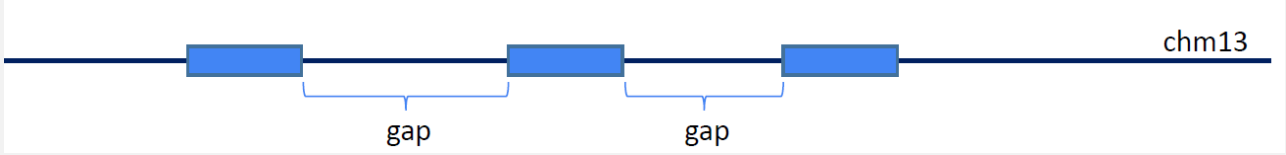
Distribution of gap sizes

Most gaps are < 3000nt

Very short gaps (< 100nt) are easy to polish: Illumina reads are longer than these gaps

In CHM13, short gaps + solid regions covers 74% of the ref genome and 85% of FlyE assembly





Polishing using hypo-hybrid

Input:

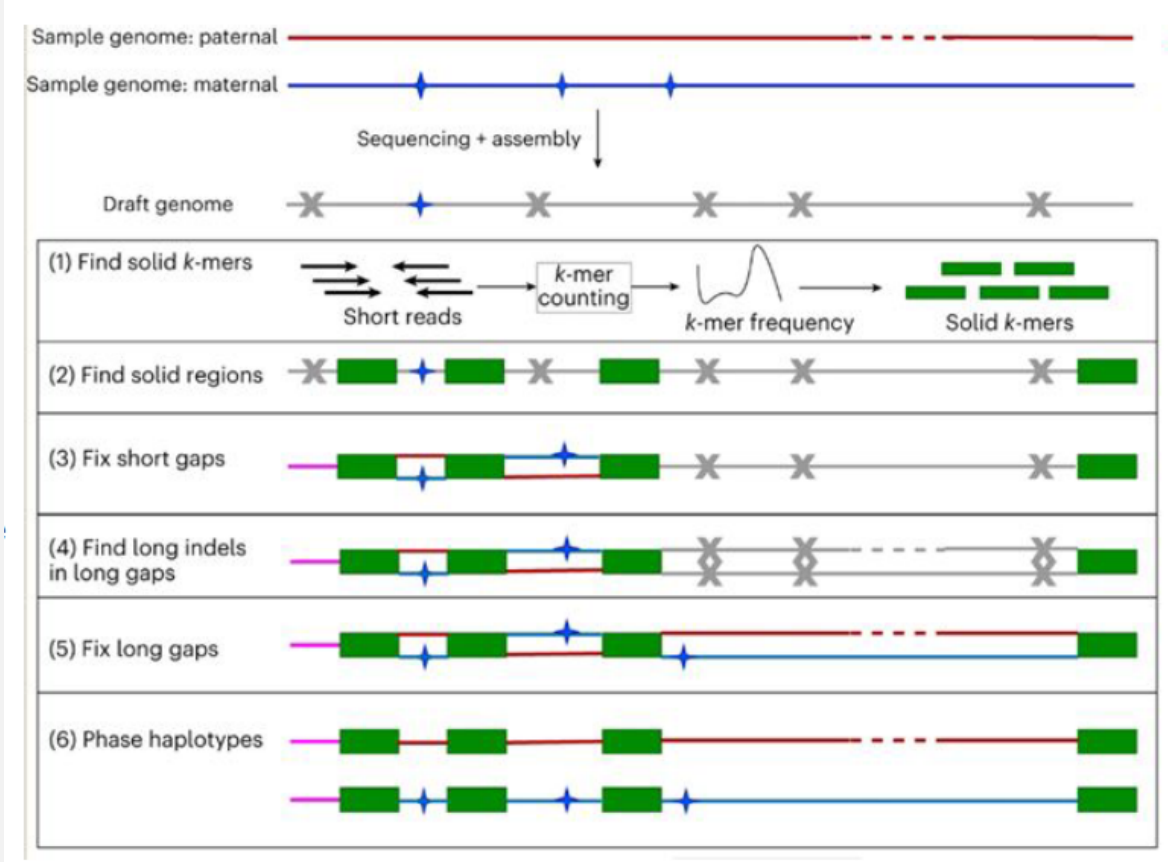
Draft genome

Short reads

Long reads

Output:

Polished diploid genome

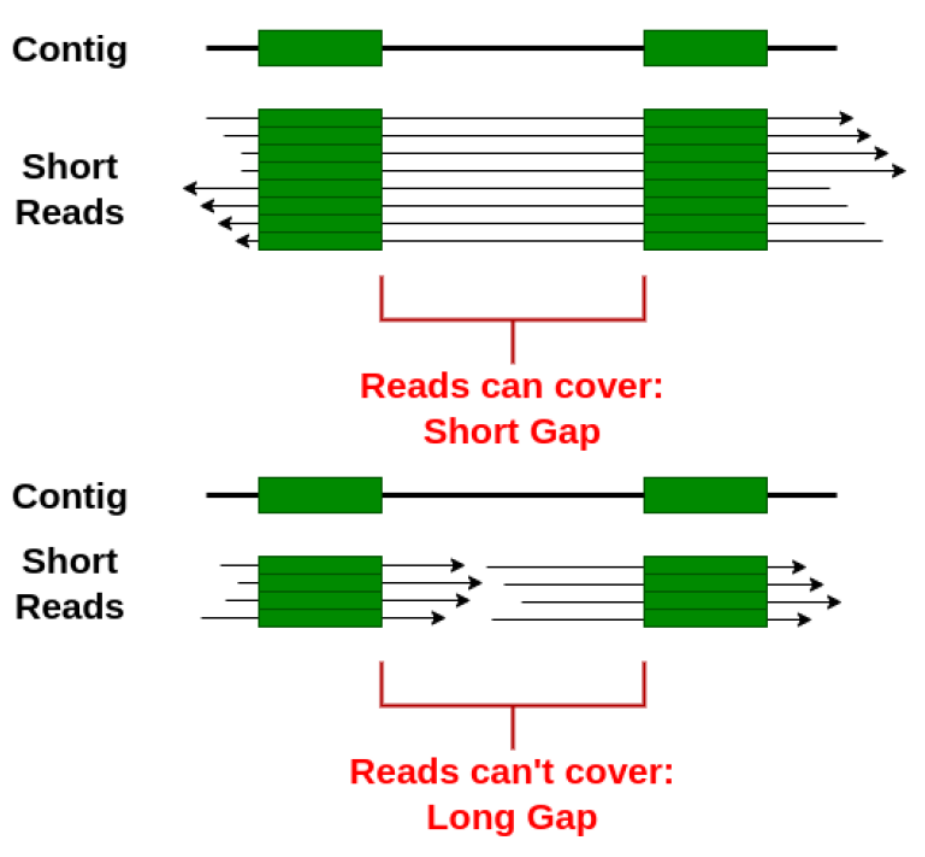


Main tasks

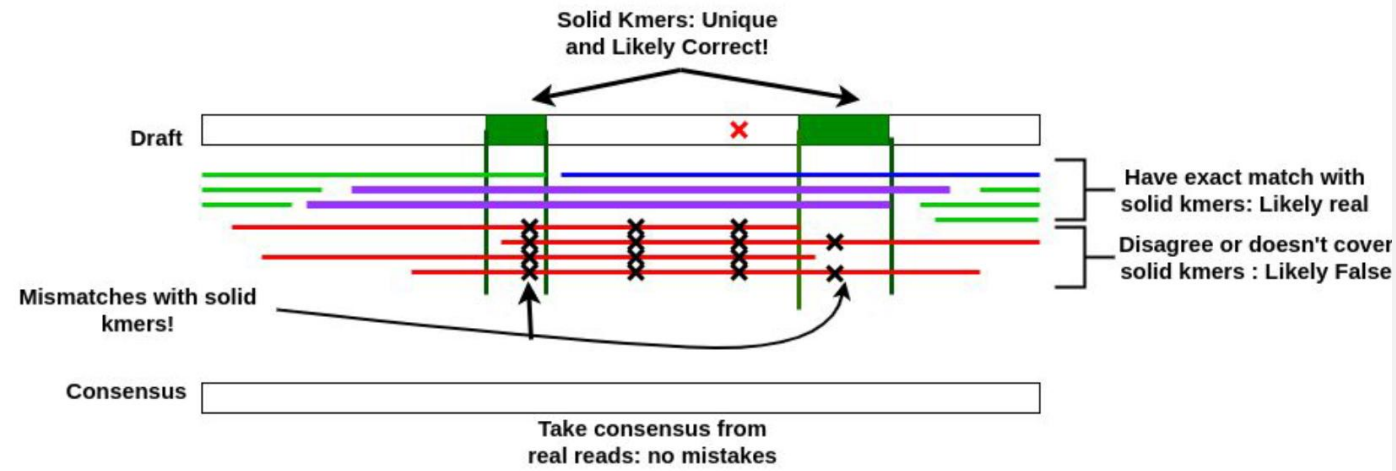
Correct short gaps

Correct long gaps

Phase the assembly



Choose which short reads to use for diploid consensus



Using the solid regions, we can choose which short reads to use for consensus.

Let X, Y = the two most freq fragments covering the gap

If count(X) > 30% * seq coverage, X is consensus #1

If count(Y) > 30% * seq coverage, Y is consensus #2

Effect of short gap correction

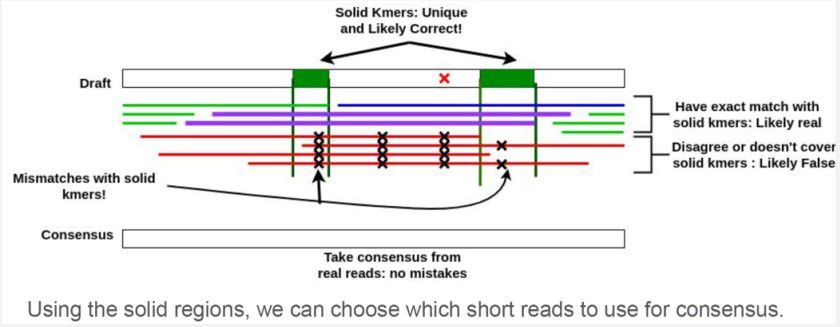
Sample	Feature	Short Gaps (draft)	Short Gaps (hypo)	Short Gaps-equivalent (NextPolish)
CHM13 Normal Coverage	Size	656,355,460	656,359,201	656,358,105
	# Errors	5,089,105	5,135	5,528
	# Errors / 100kbp	775.36	0.78	0.84
CHM13 High Coverage	Size	29,832,725	29,832,725	29,832,725
	# Errors	401,735	286	2,588
	# Errors / 100kbp	1,346.63	**0.96**	**8.68**
HG002 Normal Coverage	Size	698,806,015	698,806,799	698,806,391
	# Errors	2,513,299	4,813	5,019
	# Errors / 100kbp	359.66	0.69	0.72
HG002 High Coverage	Size	31,762,227	31,762,250	31,762,235
	# Errors	205,153	258	2,189
	# Errors / 100kbp	645.9	**0.81**	**6.89**

The presented technique shows big advantage on high-coverage regions which are more error prone

Exercise

How does hypo-hybrid avoid errors induced by high coverage region?

Choose which short reads to use for diploid consensus



Using the solid regions, we can choose which short reads to use for consensus.

Let X, Y = the two most freq fragments covering the gap

If count(X) > 30% * seq coverage, X is consensus #1

If count(Y) > 30% * seq coverage, Y is consensus #2

Finding indels in long gaps using ONT reads

Consider ONT reads aligned to both the flanking solid regions of the long gap

Cluster these ONT reads by size

Let C_1, C_2 = two largest clusters if ≥ 2 clusters exist

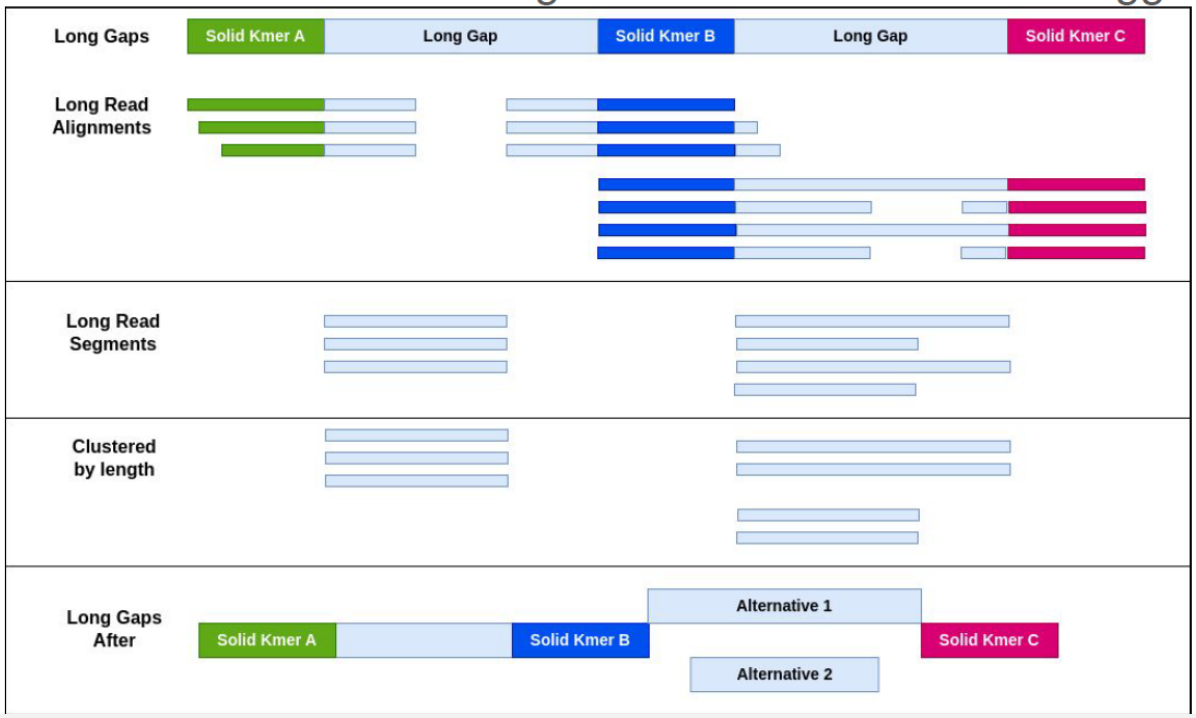
Use C_1 to generate consensus #1 for this gap

Use C_2 to generate consensus #2 for this gap, if C_2 exists and $|C_2| > 20\%$ of max of seq coverage & $|C_1|$

Remap short reads originally mapped to this long gap to these consensus

Finding indels in long gaps using ONT reads

Cluster long reads based on their size starting from the lowest. Take 1-2 biggest clusters.



Effects of long indel corrections

Hypo-short uses only short reads. It calls INSurVeyor and SurVIndel to fix long gaps

Sample	Indel Size	Long Gaps (Draft)	Long Gaps (Hypo-short)	Long Gaps (Hypo-hybrid)
CHM13	[30, 100]bp	8105	715	708
	[100, 200]bp	385	87	11
	[200, 1000]bp	56	52	13
	>= 1000bp	51	49	21
HG002	[30, 100]bp	10785	899	753
	[100, 200]bp	408	105	8
	[200, 1000]bp	59	56	14
	>= 1000bp	81	81	28

Table 3.3: Distribution of indel errors of the long gaps in CHM13 and HG002 drafts.

Exercise

After fixing long indels using ONT long reads, still need to correct small errors in long gaps using Illumna short reads

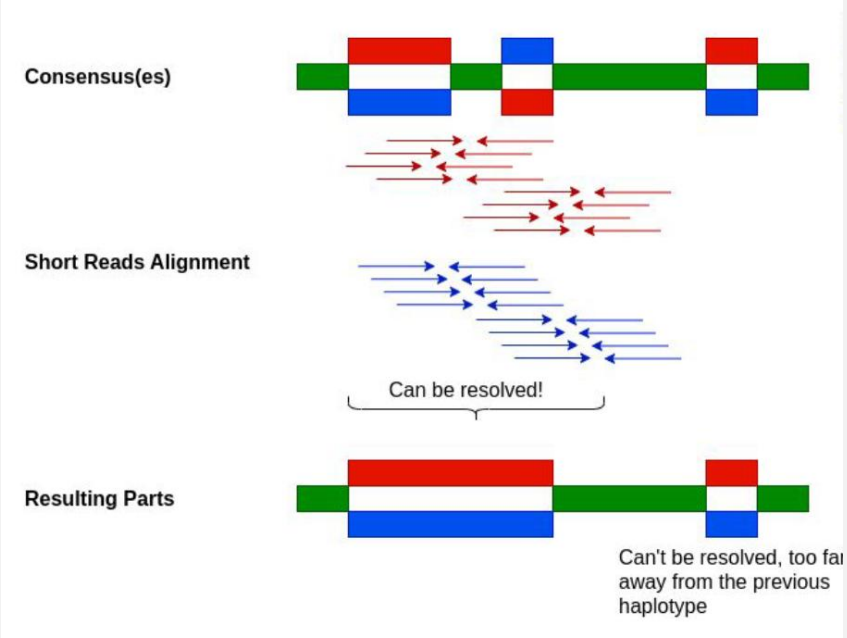
Can solid K-mers be used?

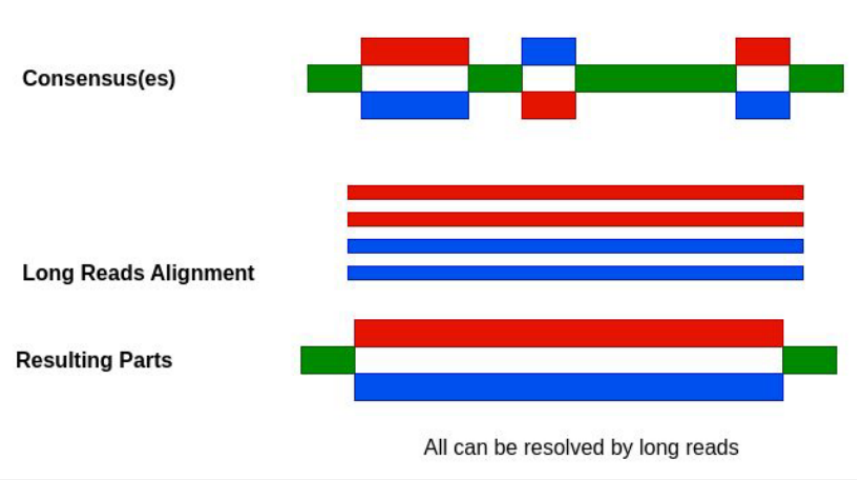
Effects of small-error correction in long gaps

Sample	Assembly	Size	# Errors	# Errors / 100kbp
CHM13	Draft	567,322,020	6,349,236	1,119.16
	Hypo-short	567,459,101	10,626	1.87
	Hypo-hybrid (before)	567,593,274	140,293	7.10
	Hypo-hybrid (after)	567,593,502	8,155	1.44
	NextPolish	567,401,292	15,891	2.80
HG002	Draft	583,562,779	2,829,641	484.89
	Hypo-short	583,566,205	11,233	1.92
	Hypo-hybrid (before)	583,600,199	89,551	6.78
	Hypo-hybrid (after)	583,601,215	9,995	1.71
	NextPolish	583,564,120	18,220	3.12

Table 3.4: The size, the number of errors and the error rate per 100kbp of long gaps in CHM13 and HG002 draft.

Phasing into diploid genome: Short vs long reads





Scoring the phasing choices

Every window has either one or two consensus

Window A with two consensus (A_1, A_2)
Window B nearest to right of A with 2 consensus (B_1, B_2)

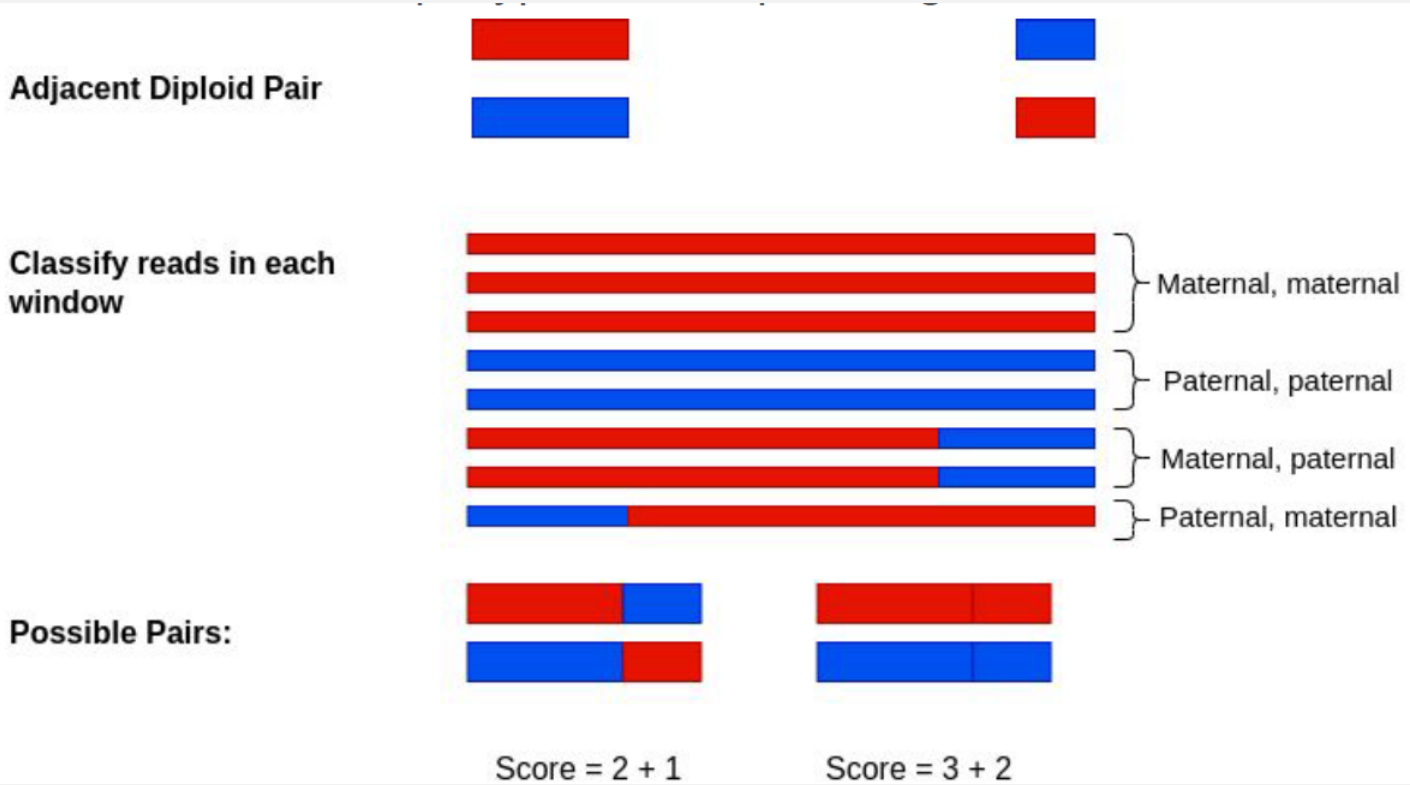
Possible phasing:
A1 B1, A2 B2
A1 B2, A2 B1

A paired end short read (or a ONT long read) supports (A_i B_j) if it covers A and B & aligned to A_i, B_j

$S(A_i B_j)$ = # of reads that support A_i B_j

Choose A1 B1, A2 B2 if S(A1, B1) + S(A2, B2) > S(A1, B2) + S(A2, B1)
Choose A1 B2, A2 B1 otherwise

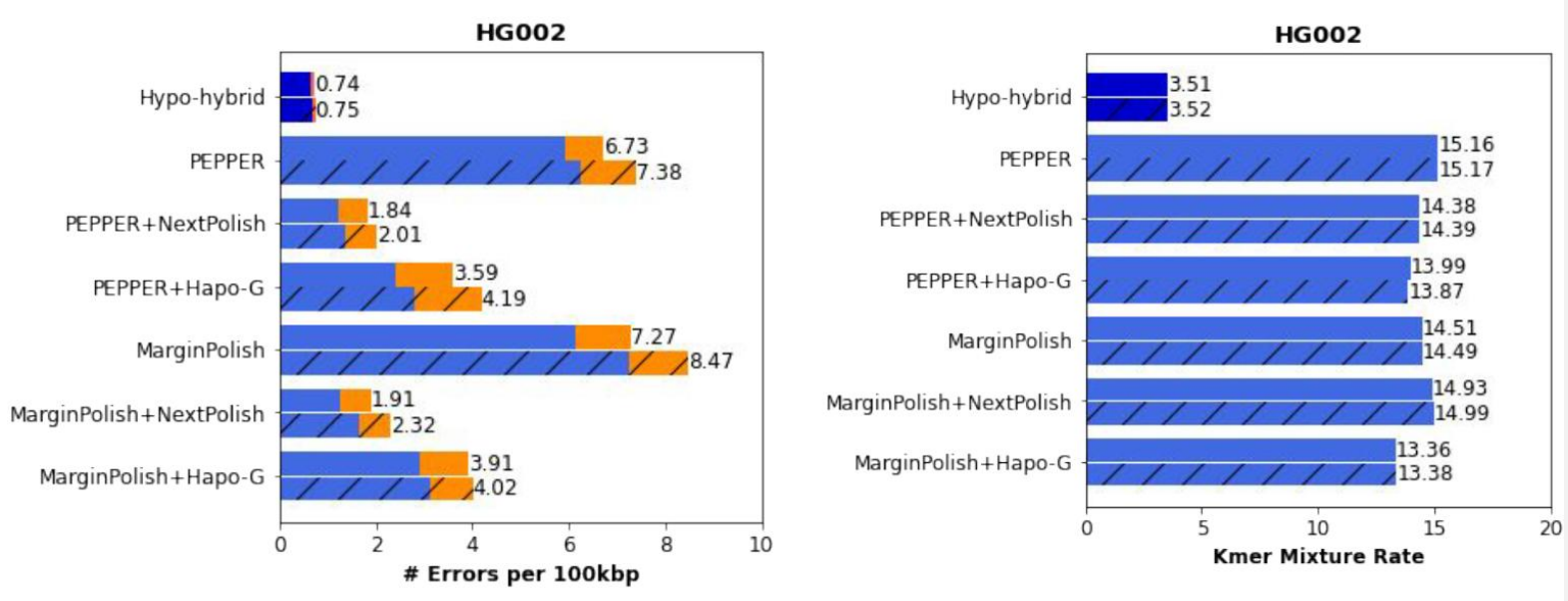
Scoring the phasing choices



Reads are classified into each haplotype in diploid region

Effects of phasing

Blue – old errors; Orange – new errors; Solid Bar – haplotype 1, Shaded Bar – haplotype 2



K-mer mixture rate:

Proportion of K-mers of lesser haplotype in each contig

Runtime performance

Polisher	Runtime	Memory Used (GB)
Hypo-hybrid	9:16:55	30.5
PEPPER	25:11:30	100.3
PEPPER + NextPolish	33:12:15	100.3
PEPPER + Hapo-G	29:17:15	100.3
MarginPolish	12:42:11	49.4G
MarginPolish + NextPolish	20:15:33	49.4G
MarginPolish + Hapo-G	17:35:19	49.4G

Table 3.7: Resources used by each hybrid polisher to polish HG002 in term of time (hour:minute:seconds) and maximum memory used.

Acknowledgements

Most of the slides were adapted from Joshua Casey Darian's PhD thesis and his viva slides

Good to read

[Hypo-hybrid] Joshua Casey Darian et al., Constructing telomere-to-telomere diploid genome by polishing haploid nanopore-based assembly. *Nat Methods* 21(4):574-583, 2024