Supplementary Notes

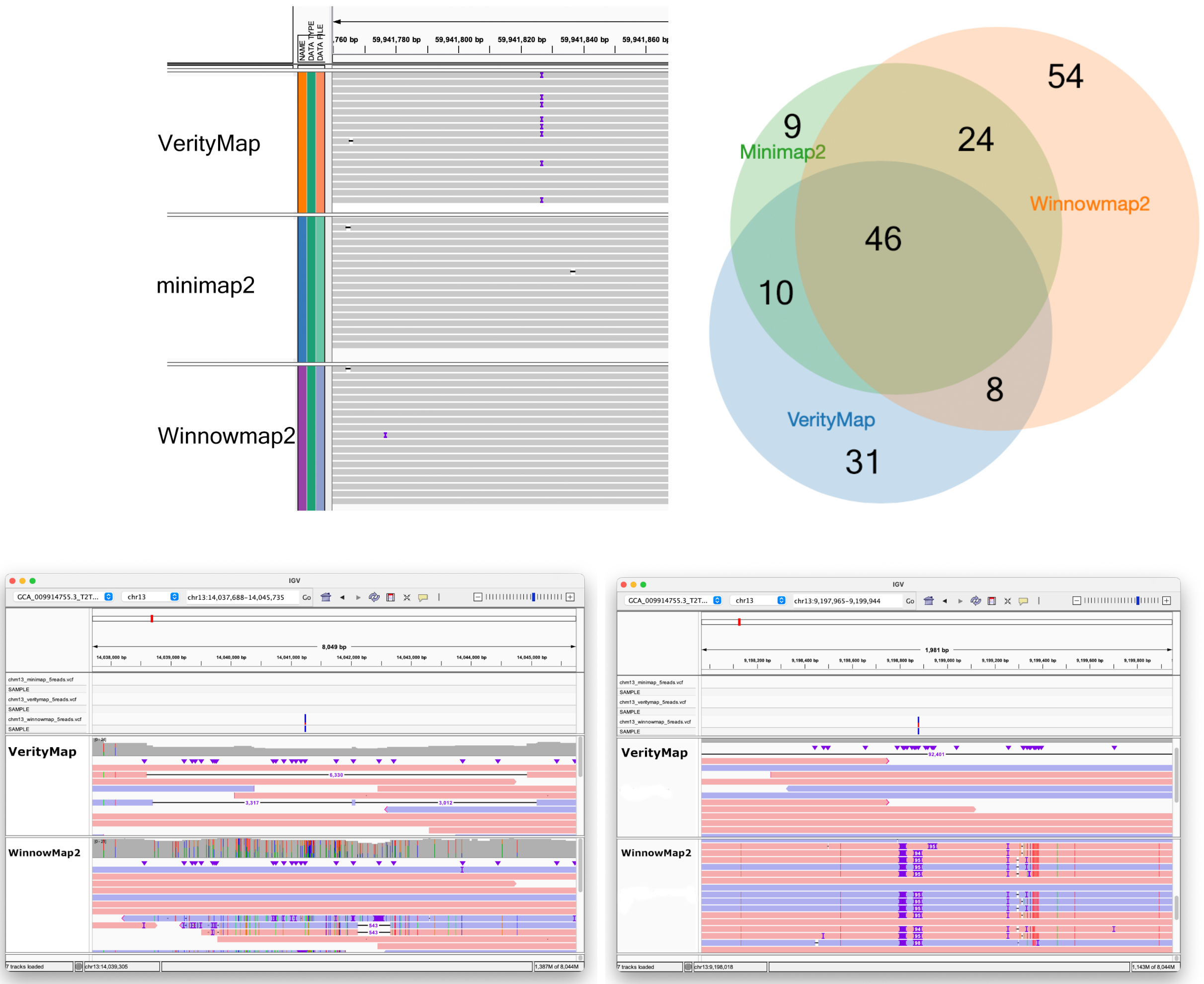
Supplementary Note 1:

"Similarities and differences between detecting mutations and detecting misassemblies"

Although the problems of detecting structural variants (SVs) and detecting misassemblies appear to be similar, there is an important difference between these two problems. Detecting SVs starts from read mapping and implicitly assumes that read mapping is a solved problem. For example, Sniffles (Sedlazeck et al, 2018) assumes that read alignments (e.g., minimap2 alignments) are given and attempts to infer SVs from these alignments. However, if read alignments are incorrect (like some minimap2 alignments in highly-repetitive regions), Sniffles will fail. VerityMap attempts to address limitations of existing mapping tools by generating accurate alignment in highly-repetitive regions that can be further used by tools like Sniffles for deriving SVs. In fact, Jain et al., *Nat Methods*, 2022 discuss using Winnowmap2's alignments as an input to Sniffles and show that it might produce better SV calls. Below we demonstrate that applying VerityMap results in more accurate mapping than Winnowmap2, suggesting that the SV calls might be more accurate as well.

To compare VerityMap, Winnowmap2, and minimap2 performance with respect to SV calling, we launched Sniffles on these three tools to align reads to CHM13 (Supplementary Figure 1, right). Since the same reads were used to generate the assembly of CHM13, the detected variants likely represent heterozygous (Het) sites within the otherwise haploid cell line (Nurk et al., 2022). Only SVs supported by at least 5 reads were reported. 46 SVs were supported by all three tools. VerityMap, Winnowmap2, and minimap2 made 31, 54, and 9 exclusive SV calls (not supported by other tools), respectively. We investigated 24 SVs that were not detected using VerityMap alignments but supported by both Winnowmap2 and Minimap2 alignments. Out of 24 SVs, 3 calls were classified by Sniffles as "BND" (breakend) and correspond to translocations between chromosomes. VerityMap is not optimized to detect such events. One SV represents a full duplication of chrM and likely is a false positive call. Three variants had low read support in VerityMap alignments and therefore were not reported by Sniffles. Five SV calls were not revealed by Sniffles, but were reported by VerityMap itself using the DistanceDiscordance test. We manually checked variants detected exclusively by Winnowmap2. First, 4 SV calls had low read support in VerityMap alignments and therefore were not reported by Sniffles. 7 of the 54 variants were not revealed by Sniffles, but were reported by VerityMap itself using the DistanceDiscordance test. The unusually high number of mismatches around some of these variants (Supplementary Figure 1, bottom) in Winnowmap2's alignments may suggest that these SVs may have incorrect position and length. Moreover, numerous variants detected exclusively by Winnowmap2 are located near clumps of mismatches (not reported by other aligners). Although we cannot claim that these variants are false positives, the detected clumps of mismatches indicate that they may represent alignment errors in highly repetitive regions.

At the same time, Sniffles detected several SVs based on VerityMap alignments that were missed by Winnowmap2 and minimap2. For example, Supplemental Figure 1, left shows an heterozygous insertion of ~2054 bp length in the centromeric region of Chromosome X that was confirmed by a manual validation of centromeric arrays (Miga et al. 2020). However, minimap2 and Winnowmap2 alignments missed this variant because of the highly repetitive structure of the centromere. Thus, VerityMap can be used as an input to SV detection tools (like Sniffles) and VerityMap's own DistanceDiscordant test can complement these tools by detecting SVs that these tools missed.



Supplementary Figure 1. Sniffles Variant calling using VerityMap, Winnowmap2, and minimap2. (Top left) HiFi read alignments to the centromeric region of Chromosome X in the CHM13 cell line. Only VerityMap detects a Het site of length ~2054 bp that has been confirmed by manual validation of centromeric array on Chromosome X (Miga et al., 2020). (Top right) Venn diagram that shows the relationship between SVs detected by Sniffles using various mapping tools. (Bottom) Clumps of mismatches in the alignments reported by Winnowmap2 in Chr13 suggest incorrect length and placement of the detected SVs.

Supplementary Note 2: "The challenge of diploid-aware read mapping"

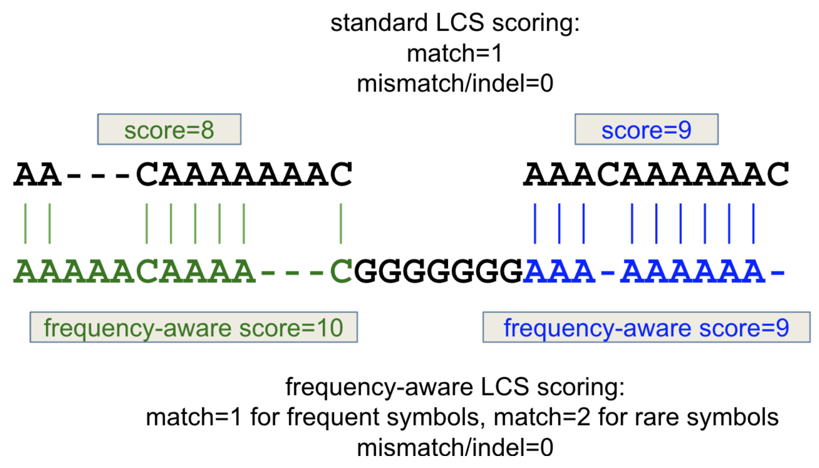
Since the definition of a solid *k*-mer requires that this *k*-mer appears only in a single contig, the ability of VerityMap to find a misassembly in a pair of highly similar haplotypes is limited. To illustrate this point, we consider a pair of substrings $String_A$ and $String_B$ of the same length K in haplotypes A and B respectively. For any string *String,* we denote $String^*$ — its corresponding (possibly error-prone) representation in the set of contigs (for simplicity, we assume that *String* is represented by a single contig). If $String_A^*$ and $String_B^*$ are identical,, there exist no solid *k*-mers in either $String_A^*$ or $String_B^*$ for any $k \leq K$. However, $String_A^*$ and $String_B^*$ might be identical in two different scenarios: $String_A$ and $String_B$ are also identical, or $String_A$ and $String_B$ are different while $String_A^* = String_B^*$ due to an overcorrection in the assembly graph. Since there are no solid *k*-mers in $String_A^*$ or $String_B^*$, VerityMap in haploid mode would report a coverage gap over both strings thus limiting its ability to distinguish between these two scenarios and potentially detect an erroneous overcorrection.

Alternatively, if $String_A = String_B = String_A^* \neq String_B^*$ (for example, if there is an erroneous indel in $String_B^*$), VerityMap will identify correct solid *k*-mers in $String_A^*$, erroneous solid *k*-mers in $String_B^*$, and subsequently map all reads originating from strings $String_A$ and $String_B$ to $String_A^*$ leading to potentially inflated coverage on $String_A^*$ and coverage gap on $String_B^*$. In particular, every solid *k*-mer in $String_B^*$ that is not shared with any other region of the genome, will not be covered by any reads. Even though it is theoretically possible to detect such a scenario with VerityMap, it is not straightforward because it, in turn, requires knowledge that both $String_A^*$ and $String_B^*$ correspond to the same string in the underlying genome. However, a clump of solid *k*-mers in $String_B^*$ with little to no coverage identifies either a sequencing drop-out or a misassembly. To distinguish these two scenarios, a user might wish to utilize an orthogonal sequencing technology (for example, Oxford Nanopores, see Discussion). A fragmented diploid assembly makes mapping challenging when one haplotype is assembled but the other haplotype has a gap. In this case, reads from the gapped haplotype may be mapped to the complete haplotype, triggering an error in VerityMap.

Since the human genome is highly inbred, stretches of DNA that are identical between two sequences and exceed typical HiFi read length, are rather common (Bankevich et al. 2022). As we show in Figure 3, VerityMap's diploid mode is effective for haplotypes that do not share long stretches of identical DNA, thus providing the first step towards developing a truly diploid-aware and error-exposing read-mapper.

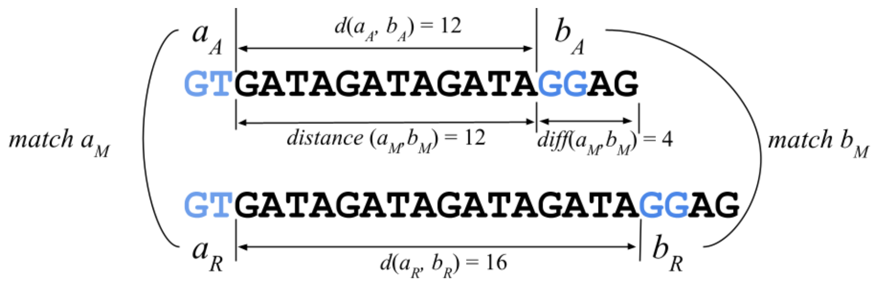
Supplementary Note 3: "Frequency-aware sequence alignment scoring"

Supplementary Figure 2 presents a contrived yet simple example illustrating the key limitation of the standard alignment in highly-repetitive regions. It presents a "genome" AAAAACAAAACGGGGGGAAAAAAAAAAA with two instances of an imperfect repeat and a "read" AACAAAAAAAC originating from the green repeat. Even though two rare nucleotides (represented by two "C" in the genome) "survived" in this read, the optimal alignment of this read to the blue repeat has larger score (9) than the score of the optimal alignment to the green repeat (8), resulting in an incorrect mapping. As shown in (Mikheenko et al. 2020), such incorrect mapping are ubiquitous in ETRs, motivating a need for a new *frequency-aware* alignment scoring in highly-repetitive regions that scores matches of rare nucleotides higher than matches of frequent nucleotides (such as "A" in the figure).



Supplementary Figure 2. Limitations of standard sequence alignment scoring. A "genome" AAAAACAAAACGGGGGGGAAAAAAAAA with two instances (green and blue) of an imperfect repeat and a "read" AACAAAAAAAC originating from the green repeat. Optimal alignment (for simplicity, we use the longest common subsequence (LCS) scoring with match score 1 and mismatch/indel score 0) to the blue repeat has larger score (9) than the score of the optimal alignment to the green repeat (8), resulting in an incorrect mapping. However, frequency-aware scoring (with an increased score for matching rare nucleotides, such as "C") results in a correct mapping of the read to the green repeat.

Supplementary Figure 3 illustrates how VerityMap implements frequency-aware *k*-mer scoring.



***a*$_M$ *and b*$_M$ are close:** $distance(a_M, b_M) < MaxJump$
synced: do not overlap
compatible: $a_M < b_M$

$bias(a_M, b_M) = \ln(diff(a_M, b_M)) = \ln(4) = 1.39$
$penalty(a_M, b_M) = \min(MisassemblyPenalty, bias(a_M, b_M)) = \min(5, 1.39) = 1.39$
$premium(a_M, b_M) = Score(b_M)*EdgeScore(a_M, b_M) = 3*1 = 3$ (if b_M is unique)
$weight(a_M, b_M) = premium(a_M, b_M) - penalty(a_M, b_M) = 3 - 1.39 = 1.61$

Supplementary Figure 3. Illustrating the definition of compatible *k*-mers. Matches of *k*-mers are *compatible* if (i) $a_M < b_M$ and (ii) a_M and b_M are synced.

Supplementary Note 4: “Extended benchmarking of long-read mapping tools”

Reference sequences:

CHM13 assembly (public release v2.0; https://github.com/marbl/CHM13; GCA_009914755.4; (Nurk et al. 2022)). ChrY in this file was assembled from sample HG002/NA24385.

Cen10. Centromeric satellite region of Chromosome 10 of CHM13 assembly.
Coordinates: Chr10:34,633,784-46,664,580.

CHM13-Cen10-interim — Cen10 in the interim version of **CHM13 assembly** predates the earliest publicly released version v0.9 of the CHM13 assembly.

Cen7. Centromeric satellite regions of Chromosome 7 of CHM13 assembly.
Coordinates: Chr7:55,414,368-68,714,496.

Cen9. Centromeric satellite regions of Chromosome 9 of CHM13 assembly.
Coordinates: Chr9:39,952,789-81,694,033.

CenX. Centromeric satellite regions of Chromosome X of CHM13 assembly.
Coordinates: ChrX:52,820,107-65,927,026.

ASat-X. Alpha satellite region of Chromosome X of CHM13 assembly (ChrX:57,587,233-61,241,121)

HG002 assembly (public release v0.7; [Citation error])

ASat-X-HG002. Alpha satellite regions of Chromosome X of HG002 assembly (ChrX:55,900,000-59,100,000).

Supplementary Table 1 summarizes information about all datasets with artificial misassemblies that VerityMap was benchmarked against.

ChrXDiploid — a synthetic diploid Chromosome X generated by combining **ChrX** from **CHM13** and **HG002**.

Supplementary Table 1. Description of used datasets.

Reference	**Dataset name**	**Misassembly type**	**Length (kbp)**	**Position (Mbp)**
Cen7	**Cen7$_{del5}$**	deletion	5	3
	Cen7$_{del10}$	deletion	10	10
	Cen7$_{del15}$	deletion	15	11
	Cen7$_{ins5}$	insertion	5	3
	Cen7$_{ins10}$	insertion	10	10
	Cen7$_{ins15}$	insertion	15	11
Cen9	**Cen9$_{del5}$**	deletion	5	40
	Cen9$_{del10}$	deletion	10	20
	Cen9$_{tandem}$	tandem duplication	10	20
	Cen9$_{dup}$	duplication	10	From 20, duplicated at 25
	Cen9$_{ins5}$	insertion	5 (Chr1:120,000,000 - 120,005,000)	40
	Cen9$_{ins10}$	insertion	10 (Chr1:120,000,000 - 120,010,000)	20
	Cen9$_{reloc}$	relocation	10	From 20, to 30
ASat-X and	**CenX-Diploid -Switch**	Each assembly was cut at one of canonical HOR units and merged	**Contig1:** ASat-X:1-2,097,938 + ASat-X-HG002:2,089,097-3,200,000	

ASat-X-HG002		in such a way to mimic a **haplotype-switch error**.	**Contig2:** ASat-X-HG002:1-2,089,096 + ASat-X:2,097,939-3,653,888

Each row corresponds to a reference sequence (first column) with a single artificial misassembly. Second (third) column shows the dataset name (the type of the introduced misassembly). Fourth (fifth) column indicates the length (position) of the indel.

Long read datasets:

CHM13-SimHiFi dataset was simulated using pbsim2 (Ono, Asai, and Hamada 2021) with the following command

```
./src/pbsim --prefix chm13_sim_hifi --sample-fastq chm13_pacbio_hifi_20kb.fastq
chm13.draft_v2.0.fasta
```

Cen7Sim dataset was simulated from **Cen7** using pbsim2 with the following command

```
./src/pbsim --prefix cen7_sim --sample-fastq chm13_pacbio_hifi_20kb.fastq cen7.fasta
```

Cen9Sim dataset was simulated from **Cen9** using pbsim2 with the following command

```
./src/pbsim --prefix cen9_sim --sample-fastq chm13_pacbio_hifi_20kb.fastq cen9.fasta
```

Cen9Sim-Het dataset was constructed by mixing **Cen9Sim** dataset with reads simulated from **Cen9**$_{\mathbf{del10}}$ using pbsim2 with the following commands

```
./src/pbsim --prefix cen9_sim_del --sample-fastq chm13_pacbio_hifi_20kb.fastq cen9.del.fasta

cat cen9_sim_0001.fastq cen9_sim_del_0001.fastq > chr9_censat_het.fasta.fastq
```

CenXDiploid-Sim dataset was constructed by mixing reads simulated from **ASat-X** dataset with reads simulated from **ASat-X-HG002** using pbsim2 with the following commands

```
./src/pbsim --prefix asatx_chm13 --sample-fastq chm13_pacbio_hifi_20kb.fastq ASatX.fasta

./src/pbsim --prefix asatx_hg002 --sample-fastq chm13_pacbio_hifi_20kb.fastq ASatX_HG002.fasta

cat asatx_chm13_0001.fastq asatx_hg002_0001.fastq > cenX.diploid.sim.fastq
```

CHM13-RealHiFi dataset contains real HiFi (20 kbp library; accession numbers: SRX7897685-8) reads for the CHM13 sample generated by the T2T Consortium. This dataset is extended by reads originating from the ChrY of HG002 sample.

ChrXDiploid-RealHiFi dataset contains HiFi reads from **CHM13-RealHiFi** recruited to **ChrXDiploid** using Winnowmap2.

CHM13-LJA is an assembly of **ChrXDiploid-RealHiFi** reads with LJA (Bankevich et al., 2022).

Supplementary Table 2 benchmarks VerityMap, Winnowmap2, and minimap2 when aligning simulated reads from the Cen7Sim read-set to the $\text{Cen7}_{\text{del5}}$, $\text{Cen7}_{\text{del10}}$, $\text{Cen7}_{\text{del15}}$, $\text{Cen7}_{\text{ins5}}$, $\text{Cen7}_{\text{ins10}}$, $\text{Cen7}_{\text{ins15}}$ assemblies. Supplementary Table 3 benchmarks VerityMap on the CHM13-SimHiFi dataset. We also aligned the CHM13-RealHiFi dataset to CHM13 assembly to reveal possible assembly errors and Het sites (Supplementary Table 4).

We have applied VerityMap in the FragmentedVerityMap mode (Supplementary Note "Validation of fragmented haploid and diploid assemblies") for verification of the **CHM13-LJA** assembly. VerityMap identified 54 sites with putative misassemblies or heterozygous sites with indel sizes ranging in length from 6474 bp to 73,449 bp. Supplementary Figure 4 shows several examples of putative misassemblies detected in the alignments generated by VerityMap. Specifically, Supplementary Figure X, top, contains an insertion of length ~7132bp that is supported by all bridging reads, indicating a likely misassembly. On the other hand, Supplementary Figure 4, middle, shows a *heterozygous* insertion of length 2057bp in contig 12 that maps to centromere on Chromosome X. Since its length coincides with the length of the canonical high-order-repeat in this centromere, the insertion likely corresponds to an insertion of a full tandem repeat unit (Miga et al., 2020, Altemose et al., 2022). We further launched Sniffles (Sedlazeck et al., 2018) on alignments generated by VerityMap with command

```
sniffles    --input    chm13-lja.bam    --vcf    chm13_lja.1000.vcf
--cluster-merge-pos 50 --minsupport 10 -t 12 --minsvlen 2000
```

Sniffles reported eight insertion sites — four deletions (lengths 6803, 2026, 2057, 3554 bp) and four insertions (7404, 6199, 8930, 6079 bp). Although this experiment shows VerityMap's capability to identify potential issues in fragmented assemblies, a more extensive benchmark is necessary to rigorously measure the statistical power of VerityMap when evaluating fragmented haploid or diploid assemblies.

Supplementary Table 2. Benchmarking VerityMap, Winnowmap2, and minimap2 when aligning simulated reads from the Cen7Sim read-set to the Cen7$_{del5}$, Cen7$_{del10}$, Cen7$_{del15}$, Cen7$_{ins5}$, Cen7$_{ins10}$, Cen7$_{ins15}$ assemblies.

Dataset	**# correctly / incorrectly mapped reads**			**# alignments correctly / incorrectly extended through breakpoint**			**# alignments clipped at breakpoint**		
	VerityMap	**Winnowmap2**	**minimap2**	**VerityMap**	**Winnowmap2**	**minimap2**	**VerityMap**	**Winnowmap2**	**minimap2**
Cen7$_{del5}$	12665 / **2**	**12839** / 17	**12839** / 17	**14 / 0**	0 / 26	1 / 22	**7**	1	3
Cen7$_{del10}$	12664 / 2	12854 / 2	**12855 / 1**	**5 / 0**	0 / 0	4 / 0	**10**	9	7
Cen7$_{del15}$	12665 / **2**	**12841** / 15	12837 / 19	**5 / 0**	0 / 0	0 / 0	**18**	16	16
Cen7$_{ins5}$	12665 / 2	**12856 / 0**	**12856 / 0**	**15 / 0**	4 / 0	**15 / 0**	0	0	0
Cen7$_{ins10}$	12665 / 2	**12856 / 0**	**12856 / 0**	**16 / 0**	8 / 0	15 / 0	0	0	0
Cen7$_{ins15}$	12665 / 2	12853 / 3	**12856 / 0**	**17 / 0**	2 / 0	16 / 0	0	0	0

The concept of a correctly/incorrectly mapped read is defined in Supplementary Note "Measuring performance of mapping software". Only primary alignments were taken into account. Total number of reads 12,856. The majority of reads incorrectly mapped by minimap2 and Winnomap2 have secondary alignments to the correct positions: in cases of nearly perfect duplications, minimap2 and Winnomap2 might incorrectly choose primary alignments, while VerityMap classifies a read as "unalignable". The best value for each column group is in bold.

Supplementary Table 3. Benchmarking VerityMap on the CHM13-SimHiFi dataset.

	#reads	#mapped reads (%)	#incorrectly mapped reads (%)	#unmapped reads (%)	assembly length	#uncovered bases (%)
Chr1	137,498	137,041 (99.67%)	7 (0.01%)	457 (0.33%)	248,387,328	229,715 (0.09%)
Chr2	133,994	133,628 (99.73%)	1 (0.00%)	366 (0.27%)	242,696,752	319,001 (0.13%)
Chr3	110,494	110,342 (99.86%)	2 (0.00%)	152 (0.14%)	201,105,948	29,908 (0.01%)
Chr4	106,860	106,495 (99.66%)	2 (0.00%)	365 (0.34%)	193,574,945	268,311 (0.14%)
Chr5	99,647	99,510 (99.86%)	3 (0.00%)	137 (0.14%)	182,045,439	64,688 (0.04%)
Chr6	93,834	93,318 (99.45%)	1 (0.00%)	516 (0.55%)	172,126,628	443,153 (0.26%)
Chr7	88,558	88,311 (99.72%)	0 (0.00%)	247 (0.28%)	160,567,428	50,898 (0.03%)
Chr8	79,644	79,511 (99.83%)	2 (0.00%)	133 (0.17%)	146,259,331	61,728 (0.04%)
Chr9	82,395	80,519 (97.72%)	15 (0.02%)	1,876 (2.28%)	150,617,247	1,303,243 (0.87%)
Chr10	74,169	73,994 (99.76%)	1 (0.00%)	175 (0.24%)	134,758,134	55,485 (0.04%)
Chr11	74,362	74,176 (99.75%)	4 (0.01%)	186 (0.25%)	135,127,769	109,070 (0.08%)
Chr12	73,256	73,161 (99.87%)	1 (0.00%)	95 (0.13%)	133,324,548	14,811 (0.01%)
Chr13	62,184	59,898 (96.32%)	3 (0.00%)	2,286 (3.68%)	113,566,686	3,534,552 (3.11%)
Chr14	55,289	54,779 (99.08%)	5 (0.01%)	510 (0.92%)	101,161,492	685,702 (0.68%)
Chr15	55,312	53,665 (97.02%)	10 (0.02%)	1,647 (2.98%)	99,753,195	2,362,049 (2.37%)
Chr16	53,156	52,570 (98.90%)	8 (0.02%)	586 (1.10%)	96,330,374	137,417 (0.14%)
Chr17	46,280	46,042 (99.49%)	1 (0.00%)	238 (0.51%)	84,276,897	127,915 (0.15%)
Chr18	44,429	43,944 (98.91%)	3 (0.01%)	485 (1.09%)	80,542,538	311,643 (0.39%)
Chr19	34,124	34,062 (99.82%)	0 (0.00%)	62 (0.18%)	61,707,364	9,188 (0.01%)
Chr20	36,684	36,583 (99.72%)	0 (0.00%)	101 (0.28%)	66,210,255	37,219 (0.06%)

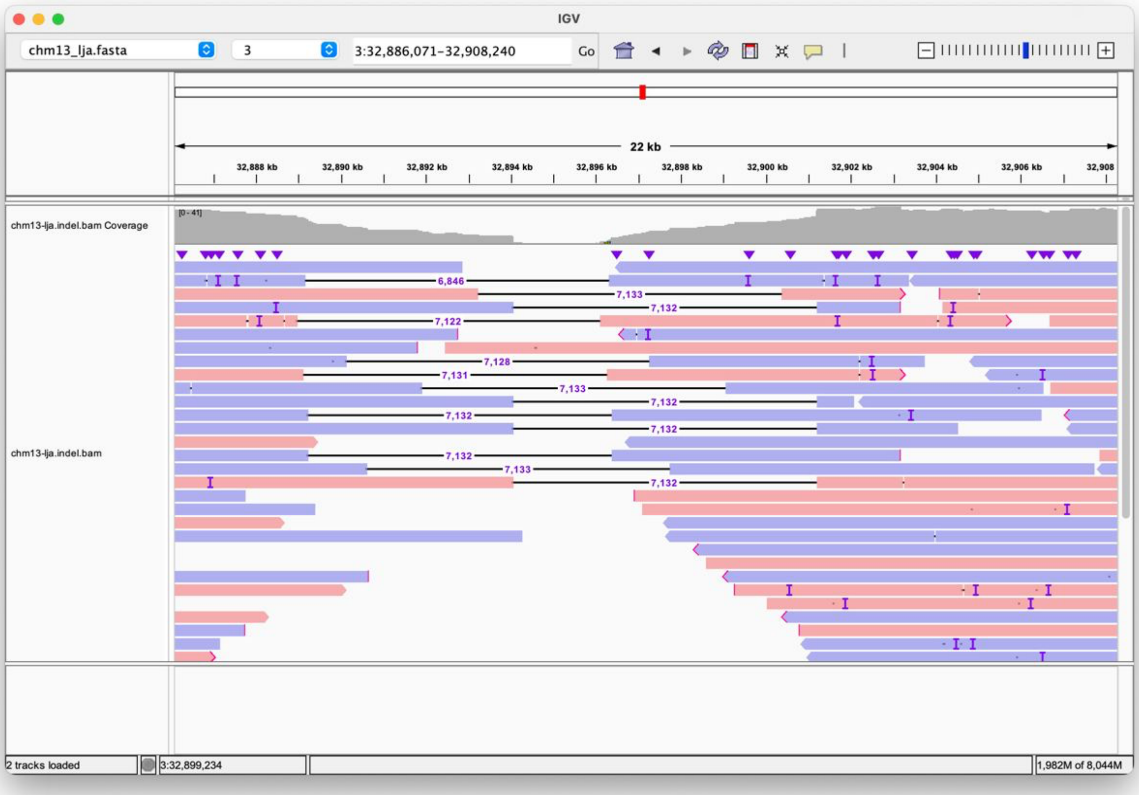
Chr21	24,977	23,540 (94.25%)	0 (0.00%)	1,437 (5.75%)	45,090,682	2,458,718 (5.45%)
Chr22	28,350	27,716 (97.76%)	4 (0.01%)	634 (2.24%)	51,324,926	868,795 (1.69%)
ChrX	84,614	84,284 (99.61%)	3 (0.00%)	330 (0.39%)	154,259,566	288,607 (0.19%)
ChrY	60,185	57,376 (95.33%)	18 (0.03%)	2809 (4.67%)	62,460,029	1,012,008 (1.62%)

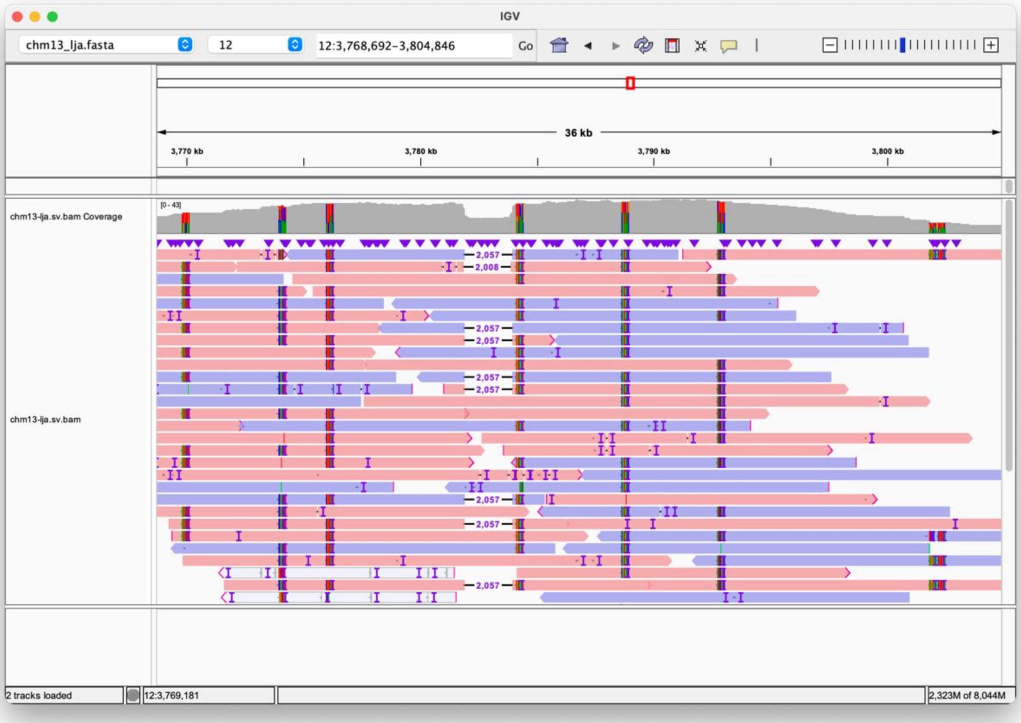
Supplementary Table 4. Benchmarking VerityMap on the CHM13 assembly when aligning real HiFi reads.

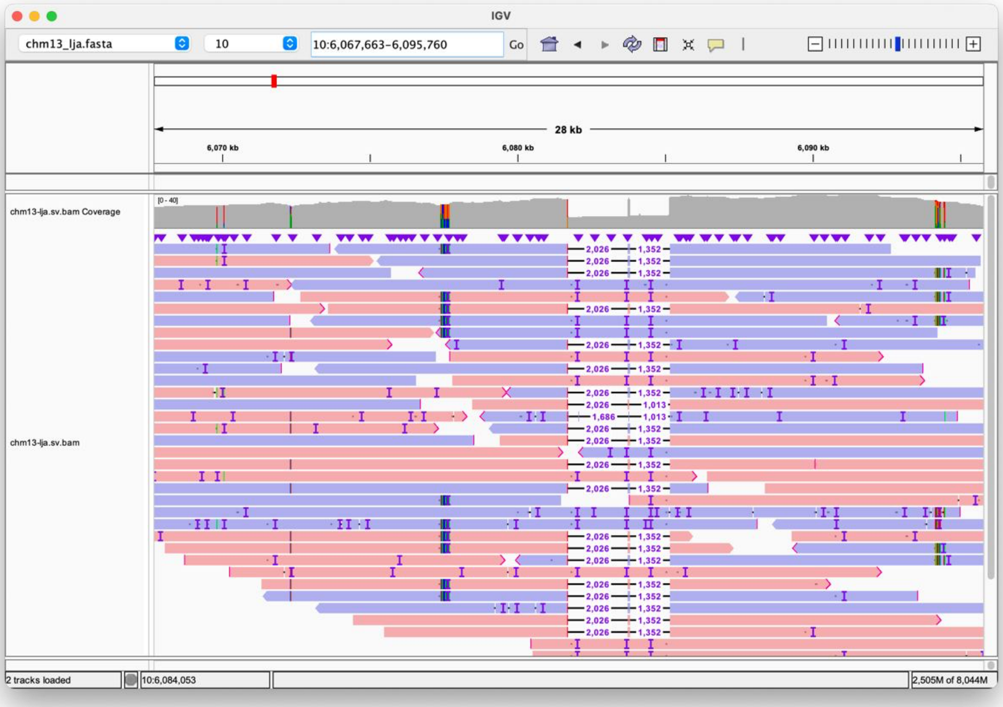
	# mapped reads	**# Het sites**	**assembly length**	**uncovered bases (bp)**
Chr1	457,122	20	248,387,328	21,344 (0.01%)
Chr2	440,316	3	242,696,752	114,112 (0.05%)
Chr3	363,497	0	201,105,948	2,890 (0.00%)
Chr4	348,380	2	193,574,945	129,201 (0.07%)
Chr5	330,073	0	182,045,439	1,993 (0.00%)
Chr6	309,725	3	172,126,628	203,809 (0.12%)
Chr7	288,274	3	160,567,428	2,552 (0.00%)
Chr8	262,933	1	146,259,331	6,363 (0.00%)
Chr9	280,431	8	150,617,247	868,459 (0.58%)
Chr10	242,712	5	134,758,134	10,471 (0.01%)
Chr11	244,050	4	135,127,769	13,192 (0.01%)
Chr12	239,358	1	133,324,548	2,682 (0.00%)
Chr13	193,133	5	113,566,686	256,392 (0.23%)
Chr14	179,606	1	101,161,492	4,365 (0.00%)
Chr15	178,759	5	99,753,195	99,329 (0.10%)
Chr16	178,109	7	96,330,374	20,909 (0.02%)
Chr17	147,723	1	84,276,897	9,531 (0.01%)
Chr18	144,694	2	80,542,538	92,612 (0.11%)
Chr19	103,865	5	61,707,364	5,492 (0.01%)
Chr20	119,269	0	66,210,255	3,208 (0.00%)

Chr21	75,810	0	45,090,682	24,647 (0.05%)
Chr22	86,907	5	51,324,926	65,110 (0.13%)
ChrX	269,221	5	154,259,566	32,364 (0.02%)
ChrY	113,893	0	62,460,029	198,722 (0.3%)

Total number of reads: 5,567,158, mapped: 5,483,967 (98.51%). The fraction of uncovered bases does not exceed 0.3% for all chromosomes except Chromosome 9 (0.58%). rDNA arrays with coordinates Chr13:5,964.549-9,123,208 (length 3,158,660 bp), Chr14:2,109,599-2,707,543 (length 597,945 bp), Chr15:2,518,973-4,655,338 (length 2,136,366 bp), Chr21:3,132,131-5,512,426 (length 2,380,296 bp), Chr22:4,796,484-5,509,436 (length 712953 bp) are considered models by the T2T Consortium until longer sequencing reads enable more accurate reconstruction (Supplementary Note "Assembly of rDNA arrays" in (Nurk et al. 2022)). Thus, we do not count bases in these regions as uncovered.







Supplementary Figure 4. Examples of putative misassemblies or Het sites identified in CHM13-LJA assembly. Top (middle, bottom) suggests an insertion of length ~7kbp (2057bp, 2026+1352bp).

Supplementary Note 5: “Downsampling solid *k*-mers in non-repetitive regions”

Although the highly repetitive regions in the newly complete assembled genome constitute the biggest challenge for read mapping, they constitute less than 8% of the human genome (Nurk et al. 2022). Consequently, most *k*-mers in the genome are solid (~98% of 301-mers in the complete assembly are solid). Such abundance of solid *k*-mers in non-repetitive regions inflates the size of the compatibility graph and thus increases the running time. For each window *W* of size *L* (the default value *L* = 30 kbp), we define *SolidFraction*(*W*) — a fraction of solid *k*-mers in window *W*. VerityMap classifies a window *W* as *non-repetitive* if *SolidFraction(W)* exceeds the threshold *MinSolidFraction* (the default value *MinSolidFraction* = 0.9), and *repetitive* otherwise. VerityMap selects all solid *k*-mers in a repetitive window but downsamples them by selecting *L * DownFreq* of random rare *k*-mers in a non-repetitive window (the default value *DownFreq* = 0.02). Such downsampling reduces the complexity for indexing of all solid *k*-mers compared to alternative approaches. For example, KMC3 does not allow generation of such a downsampled database and requires generation of the complete database that has O(|*Genome*|) memory footprint. The frequency query to such a database has O(|*Genome*| * *k*) complexity. On the other hand, the size of the database that is generated by VerityMap is

$$O\left(|Genome| \left(RepetitiveFraction / L + (1 - RepetitiveFraction) \right) \right),$$

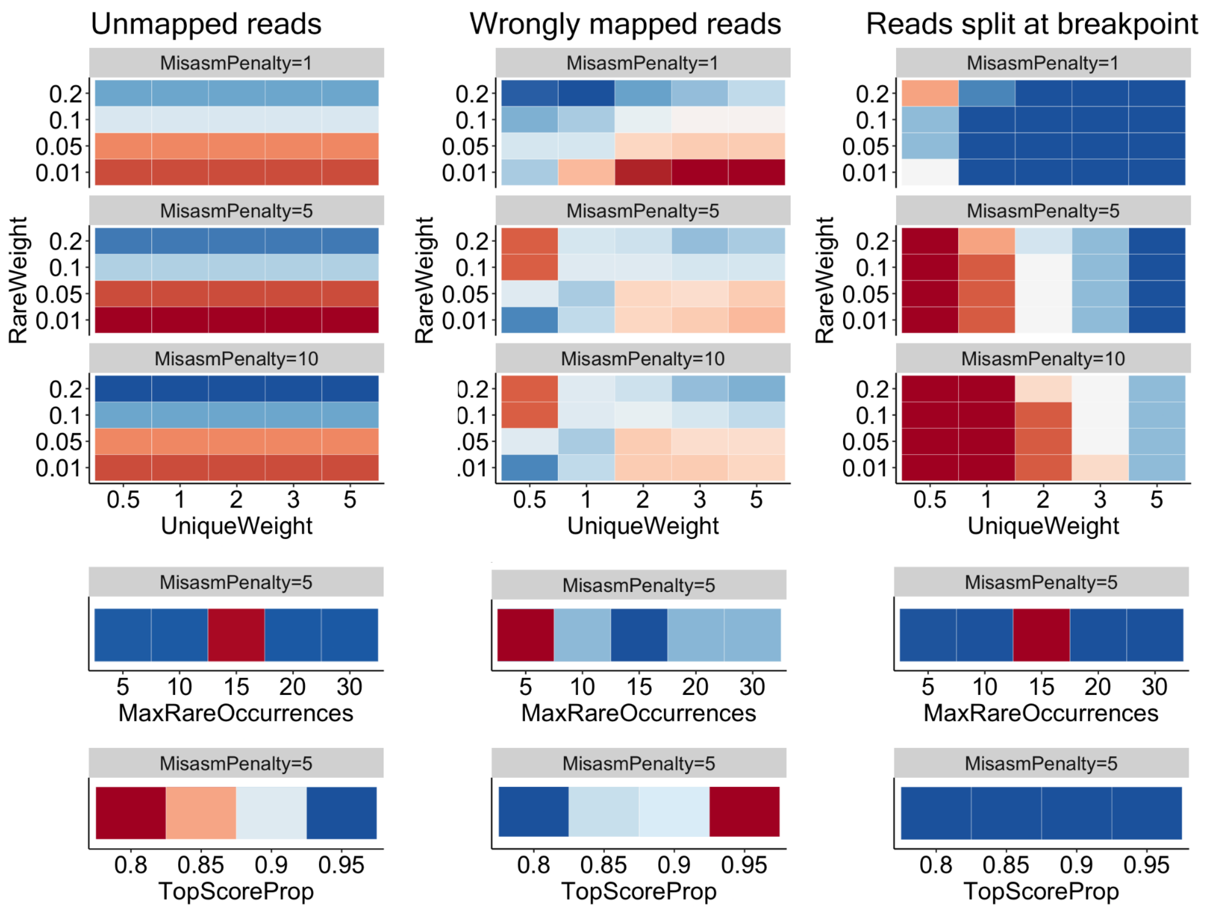
where *RepetitiveFraction* is the proportion of bases of *Genome* in non-repetitive regions. Since VerityMap is storing rolling hashes of *k*-mers in the database (rather than the *k*-mers themselves), the complexity of a query to such database is O(|*Genome*|).

Supplementary Note 6: “VerityMap parameters”

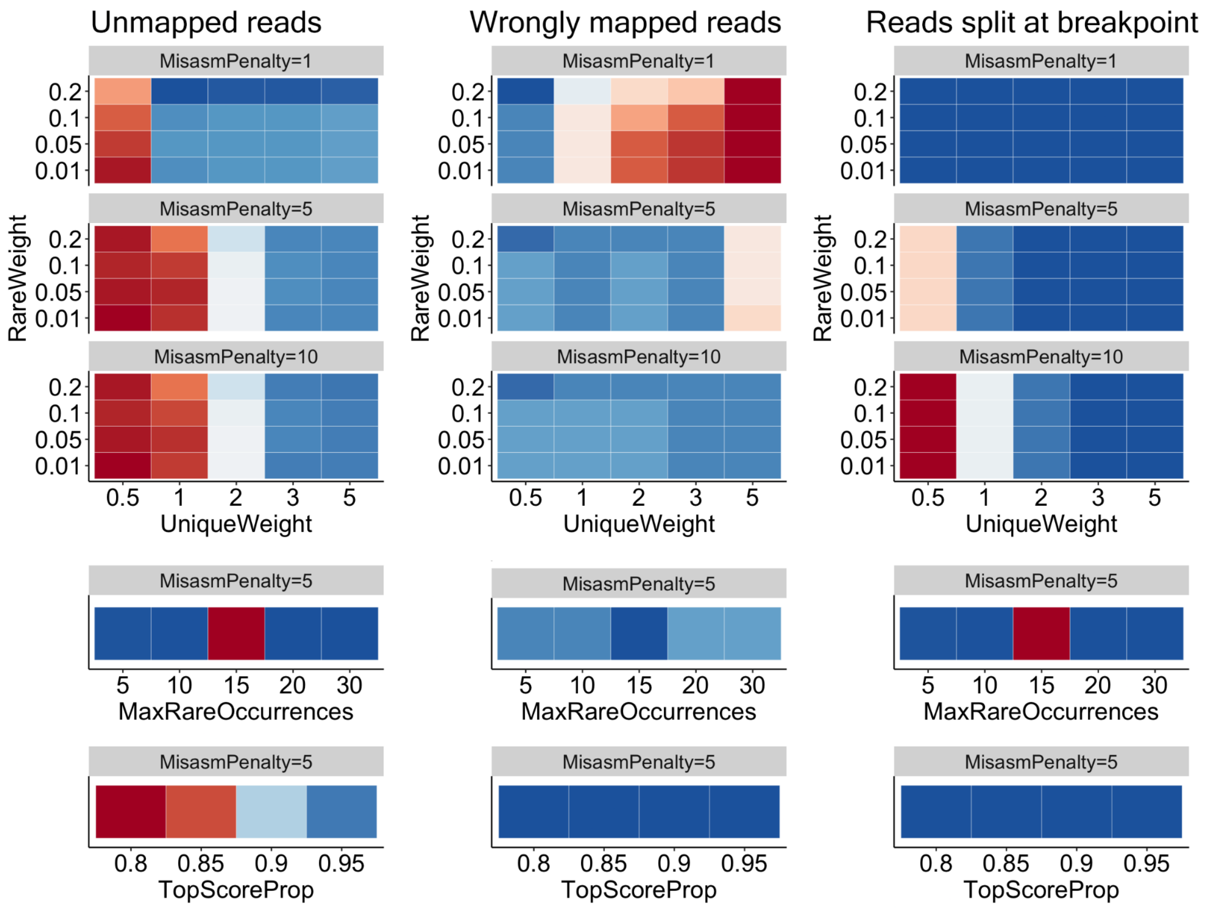
Parameters of count-min sketch and Bloom filter. VerityMap defines the CMS parameters as follows: $Bits = \lceil \log_2(MaxRareOccurrences) \rceil$, $CMSNumber = 5$, $CMSSize = \lceil 2^{\lceil \log(|Contig|) \rceil} e^{ExpBase} \rceil$, where the default value $ExpBase = 0.01$.

Main parameters. The main parameters of VerityMap were carefully defined after running the tool on **Cen7Sim** dataset and **Cen9Sim** dataset to optimize the number of incorrectly mapped reads, percent of uncovered bases in the assembly, and ability to detect the misassemblies (Supplementary Figures 5-6). Eventually we selected the following parameters:

UniqueScore=3, *RareScore*=0.1, *MisassemblyPenalty*=5, *MaxRareOccurrences*=10, *TopScoreProp*=0.9.



Supplementary Figure 5. Heatmaps showing the number of unmapped reads (Left), the number of incorrectly mapped reads (Middle), and the number of reads spanning the deletion where alignments were split in the deletion breakpoint, preventing further detection of the misassembly (Right). VerityMap was run on **Cen9$_{del10}$** dataset with different parameters. Blue color corresponds to lower values, red color corresponds to higher values.



Supplementary Figure 6. Heatmaps showing the number of unmapped reads (Left), the number of incorrectly mapped reads (Middle), and the number of reads spanning the deletion where alignments were split in the deletion breakpoint, preventing further detection of the misassembly (Right). VerityMap was run on **Cen7**$_{\mathbf{del5}}$ dataset with different parameters. Blue color corresponds to lower values, red color corresponds to higher values.

Supplementary Note 7: "Validation of fragmented haploid and diploid assemblies"

VerityMap defines a solid *k*-mer as a rare *k*-mer that appears in a single contig such that its reverse-complementary *k*-mer does not appear in any contig. Although this conservative definition is reasonable for complete assemblies, it is too restrictive for evaluation of fragmented assemblies or diploid assemblies of highly inbred organisms with long shared stretches of homologous chromosomes sharing the same sequence.

We define the count of a *k*-mer in the assembly as the number of times this *k*-mer appears in the assembly. We further define the *canonical count* of a *k*-mer as the total count of this *k*-mer and its reverse-complementary *k*-mer in the assembly. A *k*-mer is called *canonical-solid* if its canonical-count does not exceed *MaxRareOccurences*. Importantly, unlike a solid *k*-mer, a canonical-solid *k*-mer and its reverse-complementary *k*-mer can appear in several contigs (a *k*-mer and its reverse-complementary *k*-mer are simultaneously canonical-solid or not).

Below, we refer to the mode of VerityMap for evaluating fragmented haploid (diploid) assembly as *FragmentedVerityMap* (*DiploidVerityMap*). Both of these modes make use of *canonical-solid k*-mers rather than solid *k*-mers while the definition of compatibility graph remains the same. As described in Methods subsection “Compatibility graph”, VerityMap assigns the score *UniqueScore* to all unique *k*-mers and the smaller score *RareScore* to all non-unique solid *k*-mers in the assembly (default values *UniqueScore* = 3 and *RareScore* = 0.1). The score $Score(b_M)$ of a match b_M is defined as the score of the *k*-mer it represents. *FragmentedVerityMap* follows the same definition (except using the concept of canonical-solid *k*-mers rather than solid *k*-mers), while *DiploidVerityMap* introduces an additional parameter *DupScore* (default value *DupScore* = 1) for a match of a *duplicate k*-mer with canonical count equal to two. This change is motivated by the fact that most duplicate *k*-mers in a diploid assembly appear in long stretches of shared sequence between homologous chromosomes. Such duplicated *k*-mers would be unique in a consensus haploid assembly and it appears to be reasonable to assign them a higher score than for the rest of non-unique canonically-solid *k*-mers.

Since nearly all *k*-mers in non-repetitive regions are solid, VerityMap downsamples them to reduce the memory footprint as described in Supplementary Note “Downsampling solid *k*-mers in non-repetitive regions". FragmentedVerityMap and DiploidVerityMap use a slightly different method for downsampling k-mers that is inspired by a concept of a *minimizer* (Roberts et al., 2004). For each window W of size L (the default value $L = 30$), we select a k-mer $Minimizer(W)$ with the minimal canonical-count among all canonical-solid k-mers that start in the window W (ties are broken arbitrarily). The default value of L, while giving relatively modest improvement in running time, substantially reduces memory footprint.

Supplementary Note 8 "Summary of benchmarking results"

Below we define the terms "accurate", "error-exposing", and "diploid-aware" in the context of read mapping tools.

Accurate. We define the *origin* of a read as the starting position within the genome from where this read was sequenced or simulated. A read is mapped *correctly* (*incorrectly*) if its mapping position is within *MaxBias* bp away from its origin (default value 100bp). *Accuracy* for a mapping tool is defined as the ratio of correctly mapped reads out of all mapped reads. A mapping tool is called *accurate* in case of error-free assemblies if its accuracy exceeds *MinAccuracy* (default value 99%) and the percent of unalignable reads does not exceed *MaxUnaligned* (default value 5%) across a range of error-free benchmarking datasets. A mapping tool is called *partially accurate* (corresponds to "+/-" in Table 1) if these conditions are satisfied only on a subset of considered datasets. Since TandemMapper was designed for accurate read mapping to HOR arrays (rather than large regions without an HOR-like structure), it incorrectly mapped many reads in other regions. We thus classify it as *partially accurate* (or "+/-") in terms of accuracy for error-free assemblies. All other considered mapping tools (VerityMap, minimap2, Winnowmap2) are *accurate* as observed on a wide range of datasets (see Supplementary Note "Extended benchmarking of long-read mapping tools").

Error-exposing. A mapping tool is called *error-exposing* if it is accurate across a range of benchmarking datasets that contain misassembly breakpoints, and if the proportion of reads that are incorrectly spanning the breakpoint does not exceed *MaxIncorrectErrorExposing* (default value 5%). For example, Table 2 shows that VerityMap is the only tool that does not make any incorrect alignments near the misassembly breakpoints. Table 2, top shows that VerityMap is also substantially more accurate than both minimap2 and Winnowmap2. Moreover, in most cases VerityMap produces more correct mappings across the misassembly breakpoint. Since a similar scenario is observed in Supplementary Table 2, we refer to VerityMap as the only error-exposing tool in Table 2.

Diploid-aware. A mapping tool is called (fully) *diploid-aware* if it is accurate and error-exposing for a wide range of diploid datasets. It is partially diploid-aware if it is accurate and error-exposing on a limited range of diploid datasets or its scope is limited to a particular loci. Subsection “VerityMap correctly distinguishes haplotypes in diploid assemblies and identifies haplotype-switch errors” shows that VerityMap is the only partially diploid-aware mapping tool. Althoughwe use a single dataset, this dataset presents a challenging example for any read mapping tool since it contains a haplotype-switch error in a highly repetitive region of the human genome. Even though VerityMap is only partially diploid-aware, it presents a step forward towards developing a fully diploid-aware mapping algorithm (see also Supplementary Note “The challenge of diploid-aware read mapping”).

Supplementary Note 9: “Measuring performance of mapping software”

The following versions of mapping tools are used in all benchmarks

- VerityMap v2.1.1-alpha
- TandemMapper (Mikheenko et al. 2020); https://github.com/ablab/TandemTools, commit ad668d)
- Winnowmap v2.03 (C. Jain et al. 2022, 2020)
- Minimap v2.24-r1122 (Li 2018, 2021).

The benchmarking was done on a server with Intel(R) Xeon(R) Platinum 8164 CPU @ 2.00GHz using 30 threads. Winnowmap2 does not respect the threads "`-t`" parameter, so we limit the resources provided to it by using `taskset` command. In order to measure, CPU time and memory usage, we use `time` command and report "User time" and "Maximum resident set size". In all benchmarks the command used for running

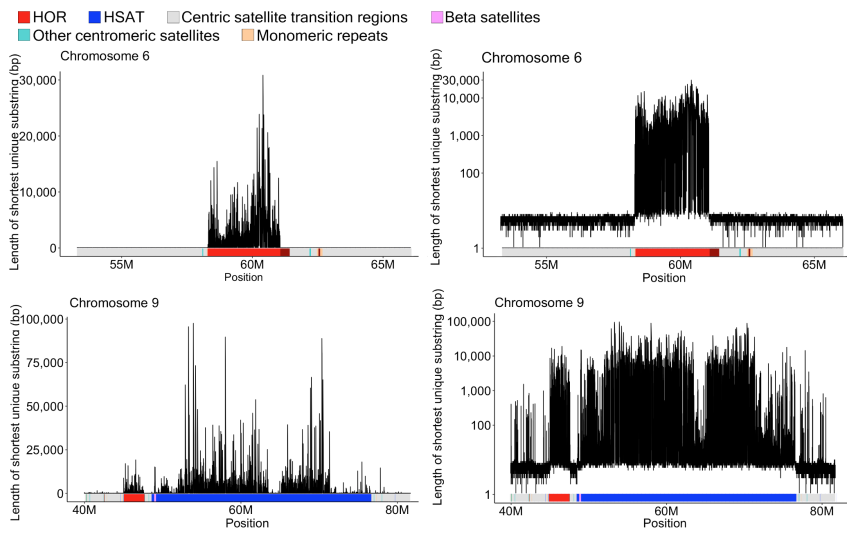
- minimap2 : `minimap2 -a -t 30 -x map-hifi {target} {queries}`
- Winnowmap2: `meryl count k=15 output merylDB {target} && meryl print greater-than distinct=0.9998 merylDB > repetitive_k15.txt && winnowmap -W repetitive_k15.txt -t 30 -ax map-pb {reference} {target}`
- VerityMap: `veritymap --reads {queries} {reference} -o {output_directory} -d hifi`

In all benchmarks, a simulated read is considered correctly mapped (otherwise, incorrectly mapped) if its starting position is within 100 bp from the read simulated position calculated for the top scoring read alignment. In all benchmarks, we consider only reads and primary read alignments longer than 5 kbp.

Supplementary Note 10: "The choice of *k*-mer size and the shortest unique substrings"

Although the *k*-mer size selected in TandemMapper (*k*=19) works well for mapping reads to many ETRs, it has several drawbacks when applied to more complex genomic regions. Two *k*-mers from a genome are called *neighbors* if they differ in a single position. Neighboring *k*-mers affect mapping performance since a single error in the assembly (or a read spanning one of them) may lead to a spurious *k*-mer match between the assembly and a read that can potentially lead to a wrong mapping. Since a short genomic *k*-mer has a higher chance to have neighbors than a long *k*-mer, the *k*-mer-based read mapping tools should ideally use a large default *k*-mer size. For example, nearly half (43%) of 19-mers in centromere X (cenX) have neighboring 19-mers within this centromere. However, neighboring *k*-mers exist only for 30% of all 31-mers, 8% of 101-mers, and 7% of 301-mers in cenX.

Moreover, using short *k*-mers can lead to ambiguous alignments even without any errors. A substring of a genome is *unique* if it appears only once in this genome. For each position of the genome, one can compute the *shortest unique substring* starting at this position and use these substrings for read mapping. If the length of the shortest unique substring in a particular region exceeds the length of a read sampled from this region, it is theoretically impossible to unambiguously map this read to the region even if the read is error-free. In highly repetitive centromeres, the lengths of the shortest unique substrings at many positions are longer than the average length of a HiFi read (20 kbp) and even exceed 100 kbp at some positions (Supplementary Figure 7). Given a window length w = 20 kbp and k = 19 bp, only 69% of windows in **Cen9** contain unique *k*-mers, while for k = 101 (301) bp, 92% (94%) of such windows contain unique *k*-mers.



Supplementary Figure 7. Length of shortest unique substrings in centromeric regions of the genome. Colors correspond to various satellites as defined in (Miga et al. 2020): high-order repeat (HOR) arrays of alpha satellites, human satellites (HSAT), centric satellite transition regions, beta satellites, monomeric alpha satellites. The y-axis shows the length of the shortest unique *k*-mer starting at this position. The y-axis in right figures is represented in logarithmic scale to better demonstrate the difference between HORs, HSATs, and other regions. (**Top**) Centromere 6 contains one of the most complex HOR arrays in the human genome (Bzikadze and Pevzner 2020; Nurk et al. 2022). The median (maximum) of the minimal length of a unique *k*-mer is 12 bp (31 kbp). (**Bottom**) Centromere 9 contains one of the most complex HSAT arrays in the human genome (Nurk et al. 2022). Together with the HOR array in centromere 6, Cen9 represented some of the most difficult-to-assemble regions in the human genome (Nurk et al. 2022). The median (maximum) of the minimal length of a unique *k*-mer is 126 bp (98 kbp).

Supplementary Note 11: “Finding a longest path in the compatibility graph”

Given a *path* in an edge-weighted graph, its weight *weight*(*path*) is defined as the total weight of all edges in this path and its *range* is defined as the distance between its starting and ending positions in the assembly. We classify a path as *valuable* if it satisfies the following conditions (i) its weight exceeds *MinWeight*, (ii) its range exceeds *MinPathRange* (the default value = 5 kbp), and (iii) it includes at least *MinUniqueKmers* solid *k*-mers as its vertices (default values *MinWeight* = 0, *MinPathRange* = 5 kbp, *MinUniqueKmers*= 0).

For each read *R*, VerityMap uses the same dynamic programming approach as in (Mikheenko et al. 2020) to find a longest path in its compatibility graph. Afterward, it removes edges of the found path from the compatibility graph and repeats the path finding process iteratively until it either finds two valuable paths or until it removes all edges in the compatibility graph (in the latter case, it either finds a single valuable path or no valuable paths). We refer to the top-scoring valuable path for a read *R* as a *PrimaryPath*(*R*) and to the second top-scoring valuable path (if it exists) as *SecondaryPath*(*R*). We classify a read as *alignable* if *weight*(*SecondaryPath*(*R*)) / *weight*(*PrimaryPath(R))* does not exceed *TopScoreProp* (default value = 0.9) or if the iterative path search returns a single valuable path *PrimaryPath*(*R*). Otherwise, a read is classified as *unalignable.* For each alignable read, VerityMap constructs the nucleotide-level alignment between every two consecutive matches in *PrimaryPath(R)* by using ksw2 (Li 2018; Suzuki and Kasahara 2018). Table 2 (top left) shows that VerityMap has fewer incorrectly mapped reads than other mapping tools in the highly repetitive $\mathbf{Cen9_{del10}}$ region.