Fast and Accurate Draft Genome Patching with GPatch

Adam Diehl[1*], Department of Computational Medicine and Bioinformatics, University of Michigan, Ann Arbor, MI, USA 48109. adadiehl@umich.edu
Alan Boyle[1,2], Department of Computational Medicine and Bioinformatics, University of Michigan, Ann Arbor, MI, USA 48109, Department of Human Genetics, University of Michigan, Ann Arbor, MI, USA 48109. apboyle@umich.edu

1 . Department of Computational Medicine and Bioinformatics, University of Michigan, Ann Arbor, MI, USA, 48109
2. Department of Human Genetics, University of Michigan, Ann Arbor, MI, USA 48109
* Correspondence: adadiehl@umich.edu

Abstract

Recent advancements in sequencing technologies have yielded numerous long-read draft genomes, promising to enhance our understanding of genomic variation. However, draft genomes are typically highly fragmented, posing significant challenges for functional genomics. We introduce GPatch, a tool that constructs chromosome-scale pseudoassemblies from fragmented drafts using alignments to a reference genome. GPatch produces complete, accurate, gap-free assemblies preserving over 95% of nucleotides from draft genomes. We show that GPatch assemblies can be used as references for Hi-C data analysis, whereas draft assemblies cannot. Until complete genome assembly becomes routine, GPatch presents a necessary tool for maximizing the utility of draft genomes.

Keywords

Long-read sequencing, Genome assembly, Reference genome, Hi-C data analysis, Genomic variations, Reference-guided patching, Intraspecific variation, Genomic mapping, Structural variation, Functional genomics, Sequence alignment, Computational genomics

Background

Since the publication of the second human genome, scientists have strived to understand the functional consequences of genomic variation. This has been accompanied by numerous advances in sequencing technology that have allowed researchers to generate unprecedented amounts of data to help us reach that lofty goal. Building off the original methods used to sequence and assemble these early genomes, these methods have been adapted to efficiently identify sites of intraspecific variation within genomes and annotate loci with functional evidence. The synthesis of these methods allows researchers to infer which variants are most-likely to have functional effects, and to develop and test hypotheses about their consequences. For the majority of these methods, the first step in data analysis is mapping sequenced reads to a reference genome. Thus, the choice of a reference genome is one of the first and most important decisions that must be made in data analysis: one that can dramatically influence the outcome of an analysis, primarily by affecting where and how well sequencing reads align. Therefore, the quality of experimental results depends heavily on the quality and completeness of the reference genome used, and how closely its underlying sequence matches the genome of the sequence donor, whether it be an individual or immortalized cell line. Indeed, we expect inverse relationships between mapping rate and quality and divergence between the donor and reference genomes. Most published reference genomes to-date were constructed from multiple individuals spanning a broad range of genetic backgrounds, and often include haplotypes not actually observed in nature. As a result, we expect a variable amount of divergence between any donor and a reference genome, including both single-nucleotide and structural variants, potentially affecting mapping at numerous loci. Since presence of these variants in sequenced reads introduces mismatches, insertions, and deletions relative to the reference genome, decreased mapping quality, incorrect mappings, and unmapped reads may result, all of which contribute to a decreased ability to discern functional signatures over background noise.

Ideally, we would use a complete, Telomere-to-Telomere (T2T) personal genome matched to the donor cell line in place of a reference genome to avoid the confounding effects of such cryptic variation. However, the resources needed to generate complete, high-quality reference assemblies from a single individual remains prohibitive for a typical individual research lab. Indeed, such efforts have been mostly confined to large, consortium-based projects (1–5) and, thus far, these efforts have produced only two truly chromosome-scale genomes: T2T-CHM13 (5) and T2T-HG002 (5,6). These efforts were enabled by recent advances in long-read sequencing methods, such as Pacific Biosciences (PacBio) HiFi and Oxford-Nanopore (ONT) sequencing. Long reads have the advantage of being able to span many repetitive regions of the genome, which are problematic to assemble when their length exceeds the read length (7–10), thus enabling gaps to be closed between contigs on either side of a repeat locus. These methods have made it cost-effective to routinely achieve sequence fragments exceeding the length of most genomic repeat regions at sufficient sequencing

depths for de-novo genome assembly. However, to date, assembling these fragments into truly T2T genomes has required a combination of deep sequencing and additional data sources to assemble the entire genome into complete chromosomes. T2T-HG002, for example, achieved this milestone using a combination of HiFi data at 120X depth and ONT data at 710X (3,6), while, for T2T CHM13, 30X HiFi and 120X ONT sequencing data, in combination with additional strand-seq, Hi-C, and BioNano optical map data were required (5). Such resources are beyond the reach of most individual labs. However, another group was recently able to fully-assemble 63.3% of chromosomes across 28 diploid genomes (1) using more attainable sequencing depths of 45.7X for HiFi and 60.5X for ONT. These assemblies are not truly T2T, with an average assembly containing only 14 complete chromosomes and a variable number of unassigned contigs. Meanwhile, several large consortia, including groups from the 1000 Genomes Project (1KG) (2), Human Genome Structural Variation Consortium (HGSVC) (4), and Human Pangenome Reference Consortium (HPRC) (11), have collectively produced hundreds of publicly-available draft personal genomes, derived primarily from immortalized lymphoblastoid cell lines (LCLs), using various combinations of sequencing methods and depths. Unfortunately, these genomes all exist as sets of hundreds to thousands of unassigned contigs, making them unsuitable for use as reference genomes for most applications.

The fragmented nature of currently-available long-read personal genomes presents a problem for many functional genomics assays, particularly those relying on long-range interactions within the genome. The most-notable of these involve chromatin conformation capture, such as Hi-C. Indeed, Hi-C processing pipelines are not designed for fragmented genomes, relying on contiguity across entire chromosomes to detect cis-interactions between distant loci. In this case, genome fragmentation both hinders detection of features artificially split across contig boundaries, inflates the number of intrachromosomal (trans) contacts, and prevents construction of complete contact matrices. Furthermore, fragmentation also complicates comparisons between genomes since contigs in different assemblies are named independently, and are typically not related to chromosomes of origin. This poses challenges both in comparison of sequence content and annotations across genomes, as well as in annotating features of interest within individual assemblies and determining their locations relative to one another. Furthermore, since many genomics operations, such as Hi-C matrix construction and normalization, scale in resource requirements proportionally to the number of reference contigs, certain analyses are intractable using a fragmented draft genome. Thus, methods for assembling fragmented draft genomes into chromosome-scale pseudoassemblies are necessary. There are two common approaches for doing so: reference-free, and reference-guided methods.

Reference-free scaffolding methods often use mappings to supplementary datasets, such as optical, physical, or linkage maps (12,13), or long-range interaction data, such as Hi-C or linked reads, to arrange contigs into single molecules. These methods rely on overlaps between contigs and the map(s) used, thus they will leave out contigs lacking overlaps. Thus, there is no guarantee that these methods will yield complete, chromosome-scale scaffolds. Furthermore, the length and sequence for gaps between scaffolded contigs remains unknown since the maps used lack information to impute the content of sequence gaps. Finally, these methods all incur the extra cost of having to generate or obtain the necessary genomic maps or long-range interactions datasets, which is likely to deter most individual research labs.

By contrast, reference-guided approaches utilize sequence alignments to a complete reference genome rather than external data to assemble contigs into chromosome-scale scaffolds. Reference-guided scaffolding assembles contigs into pseudomolecules by arranging them along chromosomes based on their aligned positions, with intervening gaps filled with N characters. A handful of methods exist for reference-based scaffolding (14–21). However, since these methods all leave N-gaps in scaffolded sequences and often do not impute gap length, their ability to serve as drop-in substitutes for a reference genome is limited. Reference-guided patching goes a step further by filling N-gaps with sequence imputed from the corresponding loci in the reference genome, thus yielding contiguous, gap-free, chromosome-scale pseudoassemblies. These

assemblies contain a complete representation of their source genome, augmented with missing regions from the reference assembly, and can, in theory, be used interchangeably with a reference assembly. We are aware of only one publicly-available tool for reference-based patching: RagTag Patch (22), an extension of RaGOO (23). However, RagTag patch is presented as a beta release that has not been rigorously tested or presented in the published literature, and has not been substantively updated since October 2021.

Here we present GPatch, a utility that presents a straightforward, intuitive, and flexible approach to reference-guided patching. We show that GPatch is able to faithfully assemble full-length, chromosome-scale pseudoassemblies from long-read assemblies with varying degrees of fragmentation, producing pseudochromosomes that accurately represent the structure of their target chromosomes. In particular, we demonstrate that GPatch faithfully produces complete, chromosome-scale pseudoassemblies given only a draft genome and a reference assembly, something that RagTag Patch could not replicate. We show that GPatch assemblies compare favorably with the T2T-CHM13 reference assembly in terms of contiguity and Hi-C read mapping rates and quality, while incorporating over 95% of nucleotides from the draft assembly. Notably, we were able to construct and normalize Hi-C matrices and call loops using a GPatch assembly, whereas the unpatched source assembly could not be successfully processed. From these matrices, we were able to recover loop predictions spanning contig boundaries in the draft genome, and those that are anchored by sequences not appearing in T2T-CHM13, demonstrating the advantages of using the GPatch genome over either the draft assembly or the T2T-CHM13 reference assembly. Thus, we conclude that GPatch is a valuable tool to facilitate the use of long-read draft genomes in everyday computational analyses, particularly where genomic fragmentation would hinder data analysis and interpretation.

Results

The GPatch Algorithm

GPatch (Fig. 1) is designed to simultaneously scaffold contigs based on alignments to a reference genome and fill intervening gaps with reference sequence to yield a chromosome-scale pseudoassembly. GPatch takes an alignment of contigs from a draft assembly to a reference genome assembly, such as T2T-CHM13, in BAM format (Fig. 1A). This file need not be sorted nor indexed, and can be produced using any aligner, but we strongly recommend minimap2 (24). Initially, GPatch isolates all primary alignments passing the chosen mapping quality threshold (default 30) from the input BAM. Next, 5’ and 3’ breakpoints in the reference sequence are inferred for each alignment by padding start and end coordinates with the lengths of any 5’ or 3’ soft-clipped regions found in the alignment’s CIGAR string. Alignments are then sorted by position and filtered to remove nested alignments. Optionally, alignments can be filtered to exclude alignments falling entirely within blacklist regions. The final, filtered, position-sorted alignments (Fig. 1B) are then processed with the core GPatch algorithm (Fig. 1C-F).

The core GPatch recurrence loops over chromosomes in the reference assembly. We first retrieve all alignments falling on a given chromosome from the final sorted BAM file (Fig. 1B). The patched chromosome sequence is initialized as an empty string and a tracker for the current position within the reference sequence, r, is set to zero. We then loop over contig alignments, alternately incorporating contigs and, as needed, patches, until all contigs have been incorporated into the patched sequence and the end of the reference chromosome is reached. For each sorted contig, we first compare the 5’ contig breakpoint, p, to r. If $p > r$, we add a patch spanning the reference interval from r to p to the chromosome sequence, followed by the complete, unmodified contig sequence (Fig. 1C). If $p == r$, contig mappings are bookended, thus no patch from the reference is necessary and the unmodified contig sequence is appended to the chromosome string (Fig. 1D). If $p < r$, there is an overlap between neighboring contig mappings and no patch is necessary. Contigs will either be bookended directly, or optionally, the 5’ end of the overlapping contig can be trimmed at the 3’ breakpoint of the preceding contig. This process is repeated until all contigs have been placed (Fig. 1E). To terminate the patched sequence, we compare r to the length of the reference chromosome and, if necessary,

incorporate a final patch to reach the 3' terminus of the reference chromosome (Fig. 1F). The patched chromosome record is then written to output in FASTA format, while coordinates of contigs and patches are written to their own files, in BED format (Fig. 1G). By checking whether the first and last contigs encompass the 5' and 3' termini of the reference chromosome and applying terminal patches as needed, we ensure that the algorithm always produces complete chromosomes. Importantly, BED coordinates for all contigs and patches both document the exact composition of the patched genome and render the patching process fully-traceable. Excluding alignment time, this process can be completed in about 6-20 minutes for a typically-sized mammalian genome.

Analysis of Simulated Data

To evaluate the performance and accuracy of GPatch, we generated four simulated draft genomes of varying difficulties. We selected two publicly available genomes on which to model our simulated data: NA12878 from the HGSVC consortium (25), representing a highly-fragmented genome consisting of thousands of contigs, and HG002 from the HPRC consortium (11), representing a highly-contiguous genome consisting of hundreds of contigs (Additional File 1: Table S1). For each genome, the T2T-CHM13 reference genome was fragmented into a set of simulated contigs matching the length distribution of the model assembly randomly tiling each reference chromosome. This yielded the "no-indel" sets for NA12878 and HG002 pseudoassemblies which contain no structural or nucleotide-level variation relative to the reference assembly (Additional File 1: Table S2). These datasets represent an idealized case where the reference assembly contains a perfect alignment match for every contig in the respective draft assembly. To present a more realistic scenario, we used SURVIVOR (26) to randomly introduce 5,000-10,000 indels and single-nucleotide-variants at a 1% rate, starting with the no-indel pseudoassemblies, which we call the SURVIVOR datasets (Additional File 1: Table S2). As a basis for comparison, all contigs from the SURVIVOR dataset were concatenated into a pseudoassembly according to their known order in the T2T-CHM13 genome. For simplicity, we apply the term "target genome", which is used as the baseline for comparison with patched genomes, to either the complete T2T-CHM13 sequence, for the no-indel set, or to the concatenated SURVIVOR genome. All four datasets were processed with both GPatch and RagTag Patch to produce patched genomes. For both methods, we

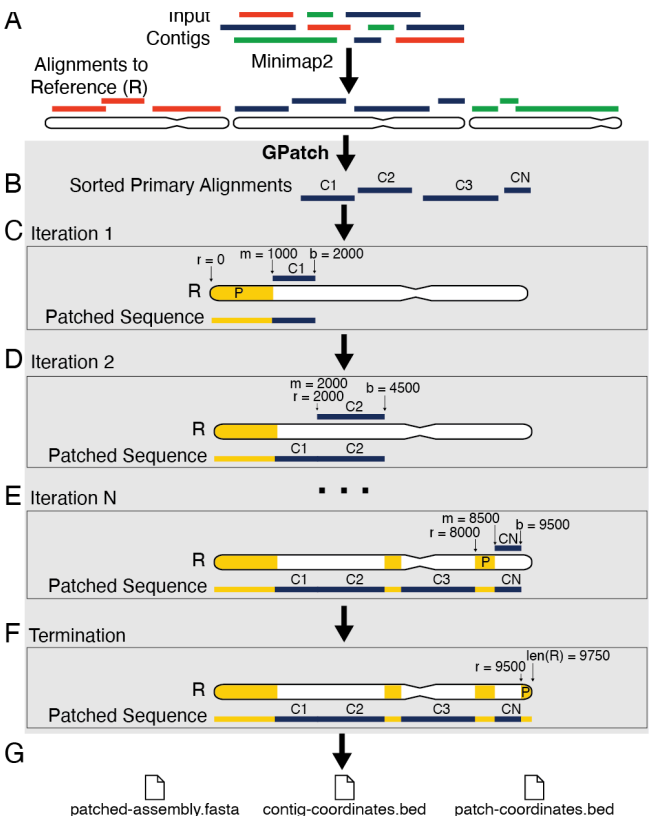

Figure 1: The GPatch Algorithm. A) Contigs are first aligned against the reference genome. GPatch takes as inputs alignments from (A), and a reference genome *R*. The grey box indicates the core GPatch recurrence, which is repeated for each reference chromosome. Primary alignments are first filtered to remove low-quality and nested alignments and sorted by position. B) Primary alignments to the current reference chromosome are retrieved and the patched sequence *S* is initialized to an empty string. C) Patching begins by drawing contig C1 from the sorted list, and setting *r*=0, while *m* and *b* are set to C1's 5' and 3' reference breakpoints. If *m* > *r*, we append a patch, *P* = *R[r:m]* to *S*, followed by the entire contig sequence, including any soft-clipped regions. Finally, *r* is set equal to *b*. D) In iteration 2, we draw contig C2 from the sorted list and set *m* and *b* to its 5' and 3' alignment breakpoints. Since *m* == *r*, no patch is necessary: the contig sequence is appended directly to *S*. Finally, *r* is set equal to *b*. E) This process is repeated until all contigs have been incorporated into *S*. F) To terminate the patched sequence, *r* is compared to the length of the chromosome, *len(R)* and, if *r* < *len(R)*, a terminal patch is appended to *S*. Steps B through F are repeated until all reference chromosomes have been patched. G) GPatch produces three output files: patched-assembly.fasta = the patched genone, contig-coordinates.bed = contig boundaries in the coordinate frame of the patched genome, and patch-coordinates.bed = patch boundaries in the coordinate frame of the reference genome.

found that the completeness of the patched assembly relied heavily on the choice of aligner and alignment parameters used to create and filter the initial alignment. Therefore, we used minimap2 as the aligner for both GPatch and RagTag patch and fixed alignment parameters across both methods. For each method, we separately optimized the patching parameters to achieve an optimal tradeoff between maximizing contig capture and placement accuracy.

We selected a set of metrics by which to compare patched genome accuracy and completeness, using comparisons of contig order, orientation, and placement in the patched genome to their known values in the target genome, and direct sequence comparisons between the patched and target chromosome sequences. Assembly completeness was evaluated using the fraction of contigs and nucleotides from the draft genome that are placed in the patched genome and the percentage of adjacent contig pairs from the target genome recovered in the patched genome. Ordering and orientation accuracy were evaluated in three ways. First, ordering edit distance was defined as the Levenshtein distance between the true and observed vectors of contig IDs along each chromosome from the patched and target genomes. This measures departures from the true contig ordering, including the effect of dropped contigs. Second, pair recall is defined as the fraction of correct adjacent pairs from the target genome recovered in the patched genome, offering another view of contig-completeness including the effects of dropped contigs. Third, we define the switch error rate as the fraction of contigs placed out-of-order in the patched sequence relative to the target sequence, calculated as half the Levenshtein distance between the observed and sorted contig ID vectors from the patched sequence, offering a view of ordering accuracy that is not affected by dropped contigs. Finally, we used bag distance to approximate the edit distance between the patched chromosome sequences and their counterparts in the target genome. Dot plots were used to visualize any departures in collinearity between the patched and target sequences.

GPatch performed exceptionally well on the no-indel dataset, placing 98.02% of NA12878 and 98.62% of HG002 contigs, and capturing 99.87% and 99.99% of nucleotides from their respective pseudoassemblies (Fig. 2A), with 100% of contigs placed with the correct chromosome, order, and orientation. We observed ordering edit distances less than 2% for both sets (Fig. 2B) and pair recall of 97.40% for NA12878 and 97.58% for HG002, respectively, indicating a low rate of dropped contigs in the patched genome. Remarkably, we were able to recover the exact sequence for all target chromosomes in the patched genome, as evidenced by identical N50, identical MD5sums, and an average normalized bag distance of 0 between patched and target genomes (Fig. 2C). Furthermore, GPatch achieved 100% pair-accuracy (Fig. 1E) and a 0% switch error rate (Fig. 1F), placing all contigs in their correct relative order. This was clearly visible in dot-plots against T2T-CHM13, where we observed a near 1:1 correspondence between patched and target genomes (Additional File 2: Fig. S1-S2). Where patch sequences occurred, they were relatively short and tended to fall near/within telomeres and pericentromeric regions (Additional File 2: Fig. 3A-B).

By contrast, RagTag patch performed poorly despite extensive optimization of patching parameters and experimentation with alternate aligners. With default patching parameters, RagTag Patch dropped most contigs from the output assembly, producing a small set of incomplete scaffolds and falling well short of yielding the desired complete patched assembly. Optimizing input parameters improved performance marginally but the best performance was about 47% contig placement (Additional File 1: Table S3). We also noted there is not a

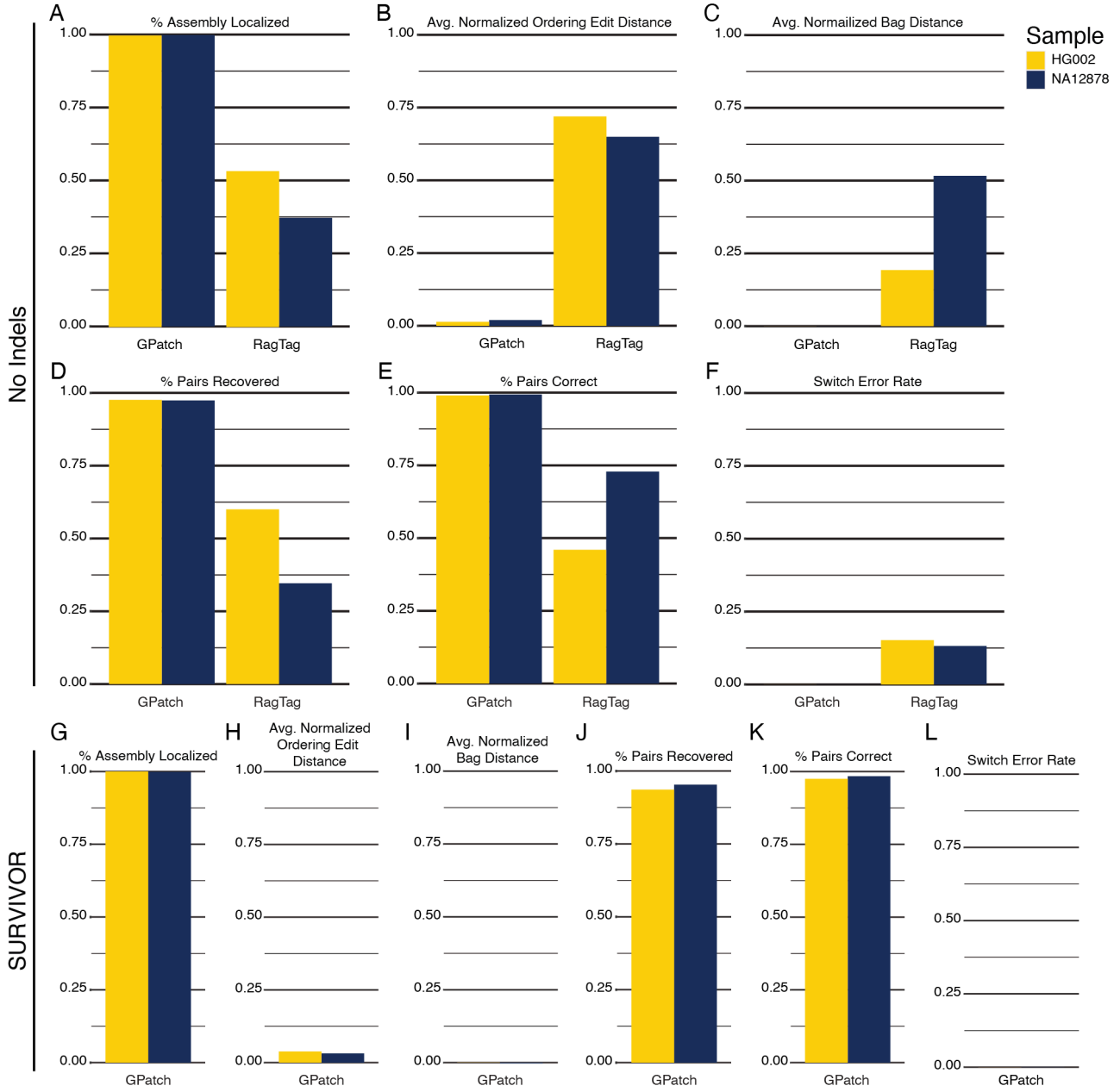

Figure 2: Simulation Analysis Results. All panels: Maize bars = HG002, Blue bars = NA12878. A and G) The percentage of nucleotides from the target assembly localized in the patched assembly. B and H) Average Levenshtein distance between the observed and actual contig ordering vectors divided by the length of the target sequence. C and I) Average bag distance between patched and target chromosome sequences divided by target sequence length. D and J) The fraction of adjacent contig pairs from the target genome recovered in the patched genome. E and K) The fraction of adjacent pairs in the patched genome that are also found in the target genome. F and L) The fraction of contigs placed out-of-order relative to neighboring contigs in the patched sequence.

1-to-1 relationship between reference chromosomes and scaffolds in the RagTag patch results. First, entire chromosomes were omitted from the patched genome, with scaffolds recovered for only nine chromosomes in NA12878 and twelve in HG002, many of which were truncated and/or contained large, likely-spurious, rearrangements relative to T2T-CHM13 (Additional File 2: Fig. S4-S5). Second, we noted that, in NA12878, chromosomes 5, 12, and X were each split into three separate scaffolds. Third, we observed that three pairs of chromosomes in the HG002 results (14-21; 3-19; 4-15) were incorrectly joined into hybrid scaffolds. Finally, we noted that scaffolds always begin and end with a contig sequence, such that scaffolds are always 5' and/or 3' truncated in the event that contig sequence does not capture the entirety of the telomere. In all, RagTag patched assemblies captured only 43.07% of contigs for NA12878 and 47.39% for HG002, representing only 37.16% and 53.41% of their target genomes, respectively (Fig. 2A). Likewise, pair recall of 34.60% for

NA12878 and 60.0% for HG002 (Fig. 2D), and ordering edit distances of 64.93% and 71.91%, respectively, (Fig. 2B) indicate a high rate of dropped contigs. Furthermore, the combination of pair accuracies of only 72.86% and 45.95% (Fig. 2E), switch error rates of 13.20% and 15.97% (Fig. 2F) for NA12878 and HG002, respectively, indicate relatively high rates of contigs being placed in the wrong relative order when compared to their neighbors in the target genome. Normalized bag distances of 51.65% and 19.25% (Fig. 2C) suggest large-scale sequence differences between the patched and target genomes. Interestingly, dot-plots between the patched genomes and T2T-CHM13 show that RagTag Patch was able to successfully scaffold three chromosomes from NA12878 and nine from HG002 achieving close similarity with their respective targets (Additional File 2: Fig. S4-S5).

To evaluate the effects of nucleotide-level and structural variation relative to the reference assembly on patching, SURVIVOR assemblies for NA12878 and HG002 were processed with GPatch and RagTag patch using the alignment and patching parameters optimized for the no-indel set. GPatch maintained its high performance even in the presence of variation, localizing 96.88% of NA12878 and 96.15% of HG002 contigs, representing 99.83% and 99.96% of their respective input genomes (Fig. 2G) and recovering 95.29% and 93.60% of adjacent contig pairs (Fig. 2J), with ordering edit distance remaining below 5% in both patched assemblies (Fig. 2H). Placement accuracy remained high, with 99.99% of NA12878 and 100% of HG002 contigs placed on the correct chromosome and strand, and pair-accuracy exceeding 97% (Fig. 2K) and negligible switch error rates (Fig. 2L) for both patched assemblies. Likewise, average normalized bag distances of 0.24% for NA12878 and 0.04% for HG002 (Fig. 2I) indicate minimal sequence content divergence from the target genome. Regardless, dot-plots show nearly one-to-one correspondence between patched and target genomes (Additional File 2: Fig. S6-S7). In contrast to the no-indel set, while still relatively short compared to contig lengths, patches were distributed more broadly and randomly across chromosomes (Additional File 2: Fig. 3C-D). Not surprisingly, we observed many more patches in the NA12878 genome compared to HG002, which we attribute to more uncertainty in the alignment of shorter contigs to the reference genome, particularly in repetitive regions. We were unable to obtain results for comparison from RagTag patch, which consistently failed to scaffold any contigs from either SURVIVOR pseudoassembly, despite extensive optimization of input parameters, and even when allowing RagTag patch to use its preferred aligner, nucmer (27).

Patching NA12878 and HG002 Draft Genomes

To evaluate GPatch's ability to construct chromosome-scale assemblies from biological data, we patched the same contig-scale assemblies for NA12878 and HG002 used to model contig-length distributions in our simulation analysis. This represents a highly-fragmented assembly, NA12878, comprised of 10,406 contigs with a median length of 16,248bp, and a highly contiguous assembly, HG002, comprised of 762 contigs with a median length of 40,201bp. Contigs were initially mapped to T2T-CHM13 with minimap2 (24), and resulting BAM files were processed with GPatch using optimized parameter values from our simulated data analysis. Patched genomes were evaluated for completeness using contig placement (Fig. 3A), the fraction of nucleotides from the input assembly localized in the patched genome (Fig. 3B), and for composition using the fraction of nucleotides in the patched assembly derived from contigs (Fig. 3C), with the goals of maximizing the rates for each metric. We also used recently-published chromosome-scale assemblies for both NA12878 (T2T-NA12878) (1) and HG002 (T2T-HG002) (25) as target assemblies for direct sequence comparison via dot-plots.

We were surprised to find that only 22.02% of HGSVC NA12878 contigs, and 18.77% of HPRC HG002 contigs were placed in their respective patched genomes (Fig. 3A). However, there was a distinct bias toward placement of longer contigs, with the median lengths of placed contigs for NA12878 (20,362bp) and HG002 (302,129bp) being significantly longer than unplaced contigs (NA12878: 15,609bp, p-value=3.71e-157, HG002: 32,233bp, p-value=2.25e-61, Wilcoxon Rank-Sum tests). As a result, patched assemblies still captured the

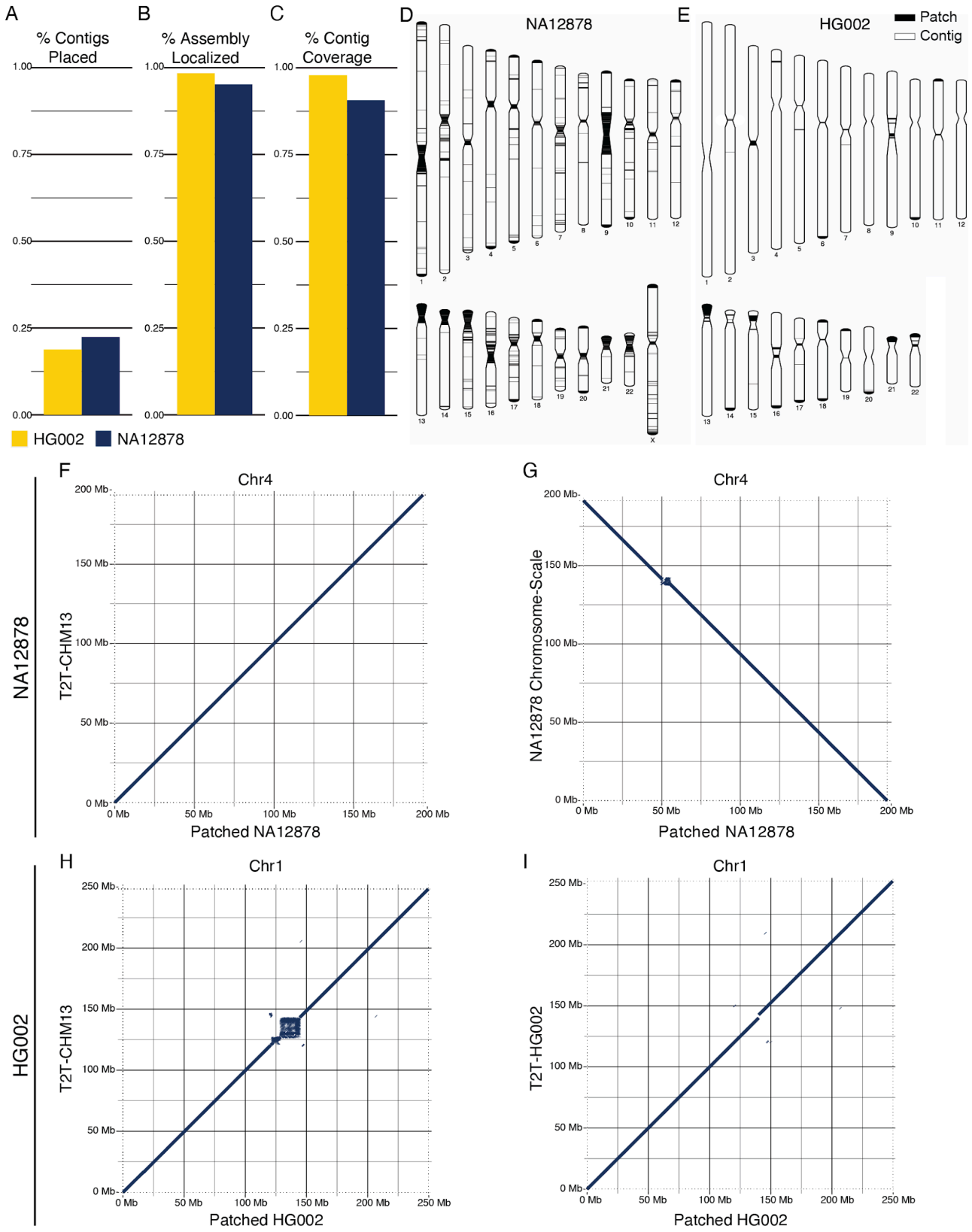

Figure 3: Patching Results for Biological Datasets. Panels A-C: Maize bars = HG002, Blue bars = NA12878. A) The fraction of contigs from the source assembly placed in the patched assembly. B) The percentage of nucleotides from the source assembly localized in the patched assembly. C) The fraction of nucleotides in the patched genome derived from contig sequences. D-E) Ideograms illustrating the locations of contigs and patches along each chromosome in the patched genome. D) HGSVC NA12878 genome patched with T2T-CHM13. E) HPRC HG002 genome patched with T2T-CHM13. F-I) Dot plots of patched chromosomes against T2T-CHM13 (F and H), or a chromosome-scale assembly for the same cell line (G and I). F) Patched HGSVC NA12878 chr4 aligned to T2T-CHM13 chr4. G) Patched HGSVC NA12878 chr4 aligned to full-length chr4 sequence from Porubsky et al. H) Patched HPRC HG002 chr1 aligned to T2T-CHM13 chr1. I) Patched HPRC HG002 aligned to T2T-HG002 chr1.

majority of nucleotides in each input assembly: 94.25% for NA12878, and 95.13% for HG002 (Fig. 3B).

Likewise, patched assemblies consisted primarily of contig-derived nucleotides: 88.9% for NA12878 and 97.85% for HG002 (Fig. 3C). As expected, patched assemblies substantially increased in N50 relative to their respective draft genomes, approaching values observed for the T2T-CHM13 reference assembly and their respective target genomes (Additional File 1: Table S4).

Consistent with our expectations, we observed that the longer contigs in the HG002 assembly yielded better performance in terms of nucleotide localization and contig coverage in the patched genome, with the GPatch HG002 genome containing fewer and smaller patches than GPatch NA12878 (Fig. 3D-E). Indeed, GPatch HG002 chromosomes 1, 8, and 12 contained only contig sequence with no intervening patches (Fig. 3E). In both patched genomes, patches tended to concentrate in/near telomeres, pericentromeric regions, and acrocentric short arms, in some cases contributing the majority of sequence in these regions. This likely reflects the repetitive nature of these regions, which tend to harbor shorter contigs and have reduced mapping ability relative to non-repetitive regions.

In order to evaluate large-scale genome structure, we compared patched genomes to the T2T-CHM13 reference genome and respective chromosome-scale target genomes using dot-plots (Fig. 3F-I). Both patched genomes were found to be highly collinear with T2T-CHM13 (Fig. S3F, S3H, Additional File 2: Fig. S8-S9), with most departures associating with pericentromeric regions (Fig. 3H). Aside from these regions, we observed large-scale structural differences between GPatch NA12878 and T2T-CHM13 on six chromosomes (Fig. 4A, Additional File 2: Fig. S8), however, no comparable divergence was noted between GPatch HG002 and T2T-CHM13 (Additional File 2: Fig. S9). All the divergent loci noted in NA12878 included at least one breakpoint within a placed contig, suggesting they represent either true structural variants or misjoins in the source assembly rather than artifacts of the patching process. Dot plots of patched genomes against their respective target genomes showed similar patterns, largely recapitulating those obtained from comparisons with the T2T-CHM13 reference (Fig. 3G, 3I, Additional File 2: Fig. S10-S11). This suggests that patched genomes indeed recapitulate their target genomes, at least at large-scale. Indeed, in some cases, we observed closer correspondence between the patched and target genomes than with the T2T-CHM13 reference, as in HG002 chr1 (Fig. 3H-I).

Automated Misjoin Correction

When examining dot plots of the GPatch NA12878 against T2T-CHM13, we noted large-scale rearrangements on chromosomes 3, 7, 9, 12, 17, and 22 (Fig. 4A, Additional File 2: Fig. S8). Since these same rearrangements were present in dot-plots between GPatch NA12878 and full-length chromosomes from a recently-published chromosome-scale NA12878 assembly (1) (Fig. 4B, Additional File 2: Fig. S9), we were able to exclude the possibility that these reflect genuine structural variants in NA12878. Therefore, these rearrangements must reflect misjoins in the HGSVC NA12878 assembly. Since these events are likely to negatively affect performance when using patched genomes as a reference for downstream analysis, we developed a method to break contigs at the coordinates of likely misjoins prior to realignment and patching with GPatch.

When applied to the HGSVC NA12878 genome, we identified 22 loci on nine chromosomes representing likely misjoins. After one round of contig-breaking, we observed no grossly-visible rearrangements on all chromosomes, except chromosome 9, when compared to T2T-CHM13 (Fig. 4C, Additional File 2: Fig. 10), and full-length NA12878 chromosomes (Fig. 4D, Additional File 2: Fig. S11). A second-round of contig-breaking identified two additional loci on chromosome 9 that appeared to originate from a contig that was absent from the initial patched assembly. The final patched genome, after two rounds of contig-breaking, had no grossly-visible rearrangements when compared to either T2T-CHM13 or full-length NA12878 chromosomes (Additional File 2: Fig. S12-S13). Overall, contig-breaking appeared to improve the quality of the patched NA12878 assembly, as indicated by improved metrics for the percentage of NA12878 nucleotides recovered in the patched genome and the fraction of the patched genome derived from contigs after each contig-breaking

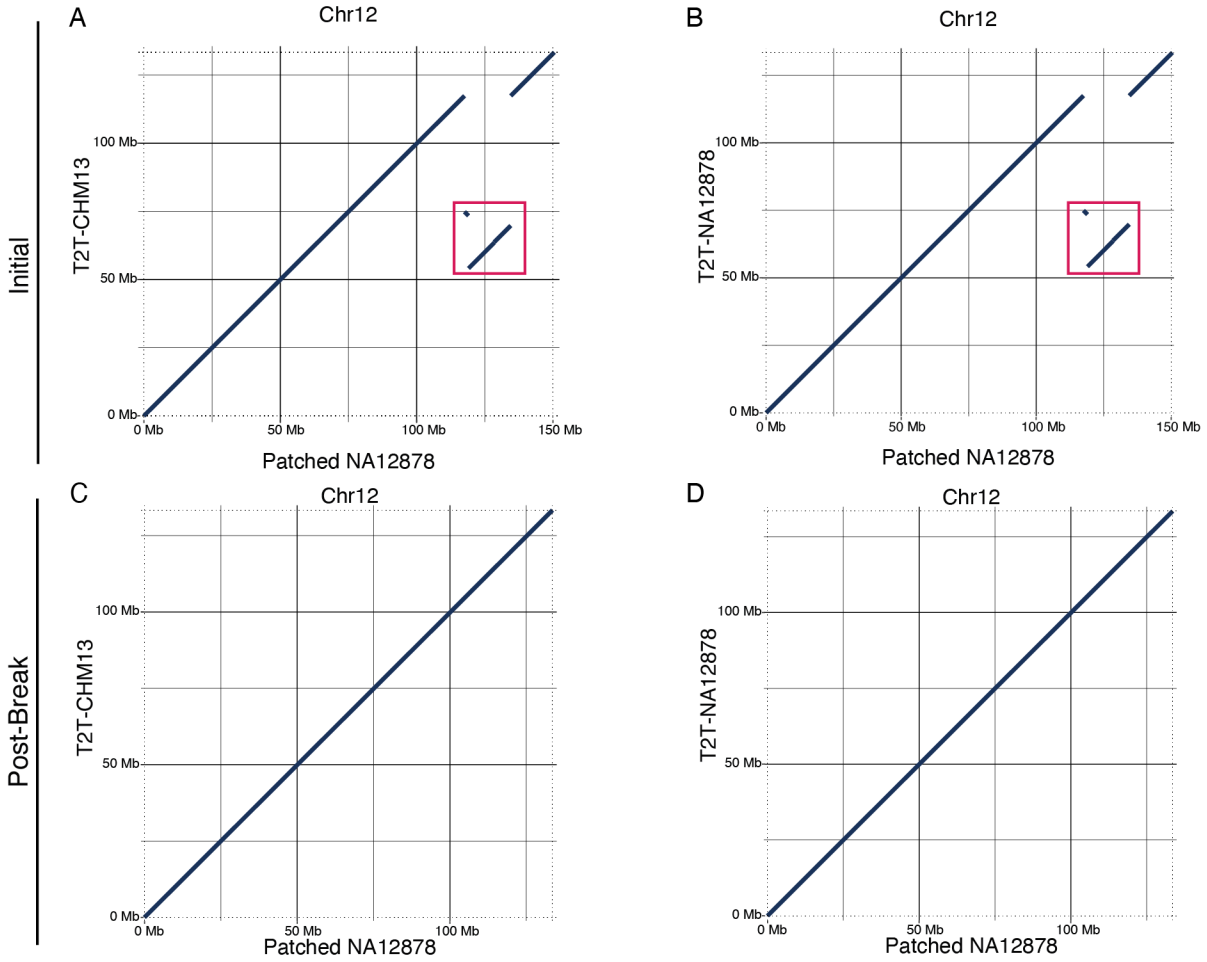

Figure 4: Identifying and Correcting Likely Misjoins. All Panels: Dot plots of patched NA12878 chromosomes against T2T-CHM13 (A and C), or a chromosome-scale assembly NA12878 (B and D). A) Patched HGSVC NA12878 chr12 aligned to T2T-CHM13 chr12. The red box indicates the location of two apparent misjoins. B) Patched HGSVC NA12878 chr12 aligned to full-length chr12 sequence from Porubsky et al. The red box indicates the location of two apparent misjoins. C) Patched NA12878 chr12 after a single-round of contig-breaking at misjoins visible in A and C, aligned to T2T-CHM13 chr12. D) Patched HGSVC NA12878 chr12 after a single round of contig-breaking at misjoins visible in A and C, aligned to aligned to full-length chr12 sequence from Porubsky et al.

iteration (Additional File 1: Table S4). This suggests that contig-breaking at likely misjoins is beneficial for overall patching performance.

Although we did not observe any large-scale rearrangements in GPatch HG002 (Additional File 2: Fig. S16-S17) comparable to those we saw in GPatch NA12878 (Additional File 2: Fig. S14-S15), our script did detect 48 potential misjoins based on the alignment to T2T-CHM13. However, breaking contigs at the corresponding loci actually decreased performance relative to the initial patched genome as indicated by drops in both the fraction of contig nucleotides captured and contig nucleotide coverage (Additional File 1: Table S4) while dot-plots against T2T-CHM13 and T2T-HG002 remained largely unchanged (Additional File 2: Fig. S16-S17). Furthermore, contig-breaking caused the patched genome to grow in size, mostly reflecting a 38.5% increase in patch nucleotide content. We conclude that contig-breaking is not beneficial when alignments between the patched genome and the reference do not reveal any large-scale divergence. Therefore, caution should be exercised when applying contig-breaking, paying particular attention to its effects on contig nucleotide recovery and patched genome size and composition.

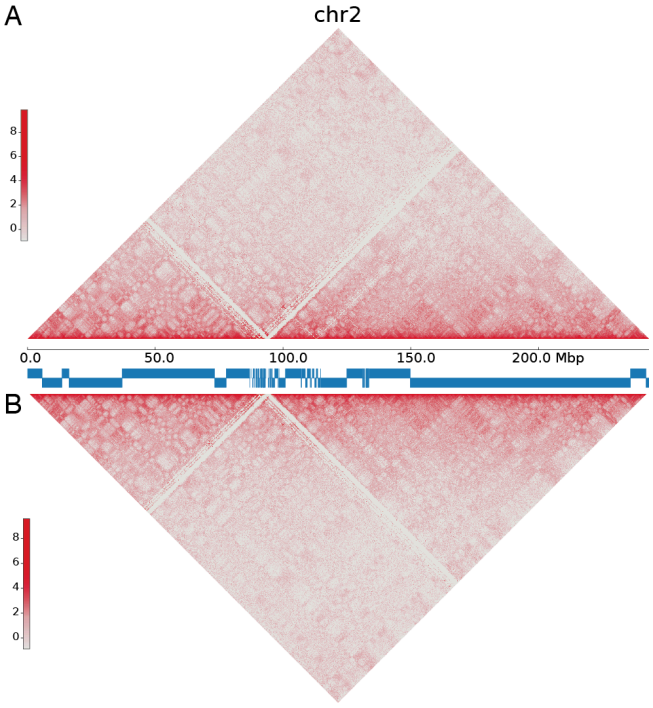

Figure 5: NA12878 Hi-C Heat Maps. All panels: Hi-C heat maps showing contact density at 80kb resolution with balanced normalization, with the X-axis indicating the position on human chromosome 2. A) NA12878 Hi-C data mapped to the HGSVC NA12878 assembly patched with T2T-CHM13. Blue bars below the X-axis indicate the positions of contigs from the original NA12878 assembly within patched chromosome 2, whereas blank regions indicate patches taken from the reference sequence at corresponding loci. B) The same NA12878 Hi-C data mapped to T2T-CHM13.

Patched Genomes Recapitulate Hi-C Patterns in T2T-CHM13

To show that GPatch assemblies are interchangeable with polished reference assemblies in genomic analyses, we obtained publicly available Hi-C sequencing data for NA12878 (28) and processed it through the loop-calling stage using the 4D-Nucleome Hi-C analysis pipeline (29) using GPatch NA12878 as the reference genome. For comparison, we analyzed the same dataset using the T2T-CHM13 assembly, and the unpatched HGSVC NA12878 assembly (Additional File 1: Table S5), using the same Hi-C data and analytical pipeline. At the mapping level, we noted similar rates of overall read mapping (99.44% and 99.45%, respectively) and mate mapping (98.94% and 98.95%, respectively) between GPatch NA12878 and T2T-CHM13. However, GPatch NA12878 actually outperformed HGSVC NA12878 (% Mapped = 99.09%, % Mate Mapped = 98.35%). Notably, reads where the mate maps to a different chromosome occurred at a ~53% higher rate in HGSVC NA12878 (39.35%) than in either GPatch NA12878 or T2T-CHM13 (25.74% and 25.83%, respectively). This disparity remains evident when mapped reads are decomposed to individual pairs of genomic contact loci, with 35.37% of read pairs split across contigs in HGSVC NA12878 compared to 20.95% in GPatch NA12878 and 21.01% in T2T-CHM13. While many of these likely represent legitimate interchromosomal contacts, some reflect “cryptic” intrachromosomal contacts which cannot be recovered in the draft genome because their anchor loci are split between different contigs. GPatch allows many of these cryptic contacts to be recovered.

Pairs files for T2T-CHM13 and Patched NA12878 were assembled into Hi-C matrices and normalized with Juicer Tools (30). However, for HGSVC NA12878, matrix construction failed with an out-of-memory error, having exhausted all available RAM (1.5TB) on a memory-optimized compute node. Hi-C matrices were plotted as heat maps for visual comparison (Additional File 2: Fig. S18-S19). We observed very few grossly-visible differences between heat maps prepared using GPatch NA12878 and T2T-CHM13, with GPatch NA12878 heat maps largely recapitulating patterns seen with T2T-CHM13 (Fig. 5, Additional File 2: Fig. S18-S19). These results suggest that GPatch assemblies can substitute for a polished reference assembly while being composed mostly of sequence derived from the draft assembly, whereas the unpatched draft assembly cannot serve interchangeably as a reference assembly.

GPatch Enables Recovery of Cryptic Loops from the HGSVC NA12878 Genome

Hi-C matrices for both T2T-CHM13-mapped and Patched NA12878-mapped data were processed with hiccups (30) to produce chromatin loop predictions at 5kb and 10kb resolution. At both resolutions, we found comparable numbers of loops at comparable average sizes at using both data mapped to GPatch NA12878 and T2T-CHM13 (Additional File 1: Table S5), showing that the GPatch genome is, again, able to recapitulate results produced using a polished reference assembly. However, we did identify notable differences between the GPatch NA12878 and T2T-CHM13 data. First, we observed many loops at both resolutions for which the 5’

and 3' anchors map to different contigs in HGSVC NA12878 assembly. We call these cryptic loops, since they would be impossible to recover in data mapped directly to HGSVC NA12878. We recovered 126 cryptic loops (~1.29% of all loops) at 5kb resolution, and 599 cryptic loops (~2.99% of all loops) at 10kb resolution, demonstrating that GPatch allows recovery of genomic features that would otherwise be missed using an unpatched draft assembly.

In order to directly compare loop predictions from GPatch NA12878 with those from T2T-CHM13, we first lifted over GPatch NA12878 loop anchor loci at both resolutions to the T2T-CHM13 coordinate frame. These were then intersected with T2T-CHM13 loop anchors using BEDtools (31). Lifted and intersected data were then used to reconstruct individual loops in order to determine the number of fully-shared, partially-shared, and NA12878-only loops (Additional File 1: Table S5). In total, we observed 88.86% of GPatch NA12878 loops at 5kb resolution and 95.21% of loops at 10kb resolution that shared at least one anchor with T2T-CHM13. This leaves a substantial fraction of loops for which neither loop anchor is shared with T2T-CHM13 data, with ~95% of these existing in sequence that is physically present in both assemblies. Notably, we detected 66 loops at 5kb resolution and 137 loops at 10kb resolution for which at least one anchor is in sequence unique to GPatch NA12878. This included four 5kb loops and five 10kb loops for which neither anchor could be mapped to T2T-CHM13. Thus, GPatch assemblies allow recovery of genomic features that would otherwise be missed when using a reference assembly.

Discussion

The combination of long-read sequencing and advanced algorithms for de-novo genome assembly have yielded an abundance of publicly-available draft personal genomes. By capturing donor-specific variations relative to polished reference assemblies, these genomes carry the promise of increased sensitivity and specificity when used as the reference assembly for downstream genomic analysis. However, these draft genomes exist as sets of hundreds to thousands of individual contigs, which are typically not assigned chromosomal identity. This presents several challenges that, ultimately, render these draft genomes unsuitable to replace a polished reference assembly for most genomic assays, particularly those targeting spatial relationships between genomic loci. Until de-novo assembly software and/or long-read sequencing advance to the point where complete, T2T assemblies can be reasonably built in-house within a typical research lab, these challenges will limit the utility of long-read draft genomes. At present, this remains impractical and methods are needed to bridge the gap between long-read draft genomes and polished reference assemblies. Here we present GPatch, which utilizes alignments to a polished reference assembly to order and orient contigs from draft genomes into complete, chromosome-scale pseudoassemblies.

We have demonstrated that GPatch faithfully produces chromosome-scale pseudoassemblies given a fragmented draft assembly and a polished reference assembly. The resulting patched genomes consistently capture over 95% of nucleotides from their source draft assemblies, are composed of 89-98% contig sequence, and are highly collinear with both the reference and target genome assemblies, in both simulated and biological data analyses. GPatch outperformed the only competing software of which we are aware, RagTag Patch, in all metrics we tested. RagTag Patch performed poorly even on simulated data containing no indels, often partially-scaffolding chromosomes, splitting chromosomes across multiple scaffolds, and frequently dropping chromosomes from the output. Furthermore, RagTag Patch output is difficult to interpret since it does not name scaffolds according to their corresponding reference chromosomes, necessitating further steps to establish chromosomal identities. Last, since RagTag Patch will only terminate a scaffold with a contig, resulting scaffolds will always be incomplete in cases where contigs do not completely encompass the telomeres. GPatch answers all these shortcomings. By looping over reference chromosomes and tracking whether contig alignments extend to the ends of telomeres, the GPatch algorithm guarantees inclusion of full-length, gapless scaffolds for all reference chromosomes in the output, with all mapped contigs assigned

unambiguously to a chromosome. Therefore, GPatch assemblies are complete in scope and can be used interchangeably with published reference genome assemblies.

The largest potential drawback of GPatch is its reliance on contig alignments to a reference genome. We have shown that alignment quality and completeness significantly impact the content of GPatch assemblies. Indeed, alignment quality is the primary reason contigs are dropped from GPatch genomes, with many dropped contigs mapping to genomic regions that are traditionally difficult to align within, such as pericentromeric and telomeric repeat arrays. This implies that donor-specific variation within these regions is likely to be missed in GPatch alignments, thus perpetuating reference bias. It is important to note that this limitation is not unique to GPatch, but will, by definition, be shared by any reference-guided approach to genome patching. However, no currently-available software can scaffold a typical draft genome de-novo in the absence of extremely deep long-read sequencing and/or additional data that are typically unattainable for individual researchers. Furthermore, these approaches are typically much slower than reference-guided approaches and yield genomes containing unresolved N-gaps that reduce mappability. Therefore, de-novo approaches do not currently offer a satisfactory solution. Accepting a degree of reference bias seems, at present, to be a necessary compromise in utilizing draft genomes to their full potential. GPatch answers the noted shortcomings of de-novo scaffolding, yielding complete, chromosome-scale pseudoassemblies that we show can be used in place of a polished reference assembly, albeit with the inevitable inclusion of some reference bias.

We further demonstrate that GPatch assemblies, when aligned back to the reference genome, can highlight misjoins within contigs from the draft assembly. These typically appear as large-scale rearrangements visible in dot-plots between the GPatch assembly and the reference genome. The GPatch github repository includes a set of scripts designed to identify the boundaries of such events within an alignment of a GPatch genome to the chosen reference assembly and break affected contigs at the corresponding breakpoint(s), allowing the resulting contig fragments to align independently within the reference genome. We show that, after contig-breaking, assemblies that harbor misjoins perform better within GPatch than their original assemblies, incorporating more contig sequence and less patch sequence compared to patched genomes built directly from a draft assembly, and are highly collinear with both the chosen reference and target assemblies. It is important to note that contig-breaking has the potential to obscure genuine structural variants, thus increasing reference bias. For this reason, we recommend caution when choosing whether to perform contig-breaking and secondary patching, with careful consideration of dot-plots as a necessary step in deciding whether to employ it. We note that the tendency to overcorrect can be minimized by careful choice of parameters in the breakpoint-identification step, in particular the minimum rearrangement size for breakpoint flagging. Importantly, if known structural variants are present in a draft genome, contig-breaking at these loci can be prevented by simply removing the corresponding records from a text file.

Finally, we show definitively that GPatch assemblies can substitute for polished reference genomes in Hi-C data analysis without sacrificing alignment rates or quality. Indeed, the GPatch NA12878 assembly achieved mapping and pairs-construction performance comparable to or better than either the T2T-CHM13 reference or the unpatched HGSVC NA12878 assemblies. By contrast, we demonstrate that unpatched draft assembly for HGSVC NA12878 could not be used in place of reference genome, with Hi-C analysis failing at the matrix construction stage. Importantly, even if matrix construction had succeeded for HGSVC NA12878, the challenges related to interpreting over 10,000 individual contact matrices cannot be overstated. Furthermore, since cis- and trans-interactions are conflated among these matrices, and since many cis-interactions are artificially split across contigs, loop calling and other spatially-based analyses will be negatively affected, leading to cryptic loops that cannot be detected even if the computational analysis could be completed. We show that GPatch allows us to recover a large number of these cryptic loops, which would otherwise be obscured since their 5’ and 3’ anchors are split between different contigs within the unpatched genome.

Surprisingly, while up to 95% of loops we identified were at least partially-conserved (i.e., at least one anchor is shared between a GPatch NA12878 loop and a T2T-CHM13 loop), a surprisingly large number of loops included at least one anchor not shared with T2T-CHM13. Most of these are within sequence that is present in both genome assemblies, suggesting that these differences likely stem from differences in fine-mapping of reads within each assembly. That such small-scale differences can influence such a large number of loop predictions is striking, speaking to the importance of the “missing” variation captured in the GPatch NA12878 assembly but missing from T2T-CHM13 in its effects on nucleotide-level read mapping. While we cannot speculate on the biological consequences of such variation, GPatch makes it accessible to a broad range of downstream genomic methods, allowing us to gain critical information toward understanding the consequences of intraspecific variation.

Conclusions

Draft genomes based on long-read sequencing are becoming increasingly available. While these genomes offer the attractive prospect of matching the reference assembly to the sequence donor for an assay, their fragmented nature presents significant obstacles to their use in functional genomics assays. In particular, assays based on spatial relationships between loci are unduly affected by genome fragmentation. Fragmented draft genomes present problems in both data-analysis, where many methods quickly become computationally intractable as the number of contigs increases, and interpretation, with the extreme example of interpreting over 10,000 individual Hi-C contact matrices had matrix construction for HGSVC NA12878 succeeded. GPatch overcomes these challenges, reducing a large number of individual contigs into a complete set of fully-assembled pseudochromosomes, given only a draft assembly and a reference genome. We show that GPatch assemblies are functionally interchangeable with polished reference genomes while incorporating over 95% of nucleotides from the source draft genome and achieving contiguity measures comparable to the reference assembly. These features enable their use in assays that would otherwise not be practical using the unpatched draft genome. Notably, we were able to utilize a GPatch genome as the reference assembly for Hi-C data analysis and loop prediction, whereas the same analysis using the unpatched draft genome failed, despite generous resource allocations. We demonstrate that GPatch is robust to varying levels of genome fragmentation, and, more importantly, that it performs well on assemblies built using only resources already within the reach of many individual researchers. We conclude that, until it becomes realistic for individual labs to routinely build polished T2T genomes from scratch, methods for reliably assembling contigs into chromosome-scale pseudoassemblies are necessary to make the most of newly-available draft genomes. GPatch achieves this milestone, thus bridging the gap between draft genomes and polished reference assemblies.

Methods

Data Simulation

We initially selected two genomes from which to model the contig-length distribution: NA12878 from the Human Genome Structural Variation Consortium (HGSVC NA1287) (4), and HG002 from the Human Pangenome Reference Consortium (3) (Additional File 1: Table S1: Data Sources). Genomes were downloaded in FASTA format and the lengths of all contigs therein were extracted using awk and stored in text files. These were then supplied to a custom Python script, along with the T2T-CHM13 genome assembly, to break each T2T-CHM13 chromosome into a set of pseudocontigs. Briefly, we loop over reference chromosomes, initially setting a position tracker *(p)* to zero to mark the start of a chromosome. Next, we draw a length *(l)* from the contig-length distribution of the model assembly. We then extract a fragment of length *l* from the current chromosome sequence and store it as a FASTA record in the output file. Finally, we set *p* equal to the end position of the extracted sequence fragment. We continue drawing values of *l* and extracting pseudocontigs of corresponding lengths until we reach the end of the chromosome. This process is then repeated until all reference chromosomes are exhausted. The end result is a pseudoassembly consisting of

contigs closely matching the length distribution from the model assembly. In addition to the FASTA output, the script also produces a BED file documenting the coordinates of all pseudocontigs within the reference genome.

Indel simulation with SURVIVOR

Starting with the NA12878 and HG002 pseudoassemblies described in the *Data Simulation* section, we produced pseudoassemblies containing 5,000-10,000 random indels and single-nucleotide variants at a 1% rate using the SURVIVOR software package (26). As contigs with lengths less than a fixed 10kb threshold are silently dropped from SURVIVOR output, we had to post-process the SURVIVOR FASTA output to add-back a small number of short contigs in each assembly using a custom script. Since indels within the SURVIVOR output genome change the coordinate system of the pseudoassembly, we could not use the unmodified reference genome, nor pseudocontig BED coordinates based on the T2T-CHM13 reference, as the target genome for comparison. Accordingly, we produced complete target genomes from the SURVIVOR-mutated assemblies by concatenating contigs into complete chromosomes based on their known order in T2T-CHM13. BED coordinates for each contig in the frame of the target genome were stored to facilitate position-based comparisons between the patched and target genomes.

Patching Simulated Data with GPatch

BASH shell scripts were used to automate alignment and patching of the simulated NA12878 and HG002 genomes, along with realignment to the target genome and dot-plot construction with R, as described in *Dot Plot Construction* below. Pseudoassemblies were first aligned to the T2T-CHM13 reference genome (5) with minimap2 (24) using parameters: *`minimap2 <reference_genome> <pseudoassembly> -x asm20 -t 24 -a`*. The *'-x asm20*' argument was used to increase the aligner's tolerance for large gaps, while the -a argument toggles SAM output format and base-alignment within the minimap2 algorithm. SAM output was converted to BAM format on-the-fly by piping minimap2 output into SAMtools view (32). The resulting BAM and the T2T-CHM13 reference genome were then supplied to GPatch, with parameters *`GPatch.py -q <$prefix.pseudocontigs.bam> -r <reference_genome> -x <prefix> -d -m 10 -t`* where -d causes GPatch to drop reference contigs without alignments from the output, -m 10 sets the mapping quality threshold for contig alignments to 10, and -t indicates that overlapping contig alignments should not be 5' trimmed. Statistics on patched assembly length, contig content, genome completeness, and accuracy were then assembled into text files using a combination of awk and custom Python scripts.

Patching Simulated Data with RagTag Patch

BASH shell scripts were used to automate processing of simulated NA12878 and HG002 genomes with RagTag Patch, along with realignment for dot-plot production in R. After extensive optimization with the goal of maximizing contig recall in the results, we ran RagTag Patch with the following parameters: *'ragtag.py patch --aligner minimap2 --mm2-params "-x asm20 -c -t 24" -o $OUT_DIR -f 100 -s 10000 <pseudoassembly> <reference_genome>'.* As part of its preprocessing steps, RagTag patch renames all sequences from the query and target genomes, ostensibly to avoid naming collisions in the patching process. RagTag Patch also includes all unscaffolded contigs in the output, along with any patched results. These features made it difficult to evaluate RagTag Patch results and compare them to GPatch. To ease interpretation and comparisons with GPatch results, we utilized a custom script to filter unscaffolded contigs from RagTag Patch fasta output and rename scaffolds according to their corresponding reference chromosomes. We used a second script to process the agp-formatted data on contig-placement within scaffolds, converting it into a BED format compatible with GPatch output. Statistics on patched assembly length, contig content, genome completeness, and accuracy were then assembled into text files using the same set of awk commands and scripts used for the GPatch results.

Dot Plot Construction

For all patched genomes, we produced dot-plots in comparison to the corresponding target genome using the 'pafr' R package (https://dwinter.github.io/pafr/). We first aligned the patched genome to its respective target genome with minimap2 (24), with parameters *'minimap2 -x asm20 -t 24 <reference_fasta> <patched_fasta>'.* PAF-formatted output was read into R with pafr's 'read_paf' function, after which we used pafr's dotplot function to produce dotplots, in PDF format, for all autosomes and the X and Y chromosomes.

Ideogram Construction

We utilized the PhenoGram web application (https://visualization.ritchielab.org/phenograms/plot) to produce ideograms illustrating the locations of patches within GPatch results. We first used awk to reconfigure the patch coordinates from GPatch's patches.bed into the text format used by PhenoGram. This was supplied as the input file for PhenoGram. We supplied the chrom.sizes file for T2T_CHM13, augmented with telomere locations obtained from the UCSC Table Browser (33), as the genome. We additionally selected "Standard Algorithm" for "Phenotype Spacing", and checked the "Chromosome only" box to toggle printing only the ideogram without accompanying annotations. Resulting images were stored in PNG format and further processed locally to desaturate and improve contrast.

Patching Biological Data with GPatch

BASH shell scripts were used to automate the GPatch patching process for the HGSVC NA12878 and HPRC HG002 genomes. Initial alignment of each genome to the T2T-CHM13 reference genome was performed with minimap2 using parameters `*minimap2 <reference_genome> <contig_assembly> -x asm20 -t 24 -a*`, with output piped into SAMtools view to convert SAM output to BAM. Resulting BAM files were processed with GPatch using parameters `*GPatch.py -q <$prefix.pseudocontigs.bam> -r <reference_genome> -x <prefix> -d -m 10 -t*` where -d causes GPatch to drop reference contigs without alignments from the output, -m 10 sets the mapping quality threshold for contig alignments to 10, and -t indicates that overlapping contig alignments should not be 5' trimmed. Statistics on patched assembly length, contig content, and genome completeness were then assembled into text files using a combination of awk and custom python scripts. Finally, dot plots were prepared according to steps outlined in the *Dot Plot Construction* methods section to compare the initial patched genome to the T2T-CHM13 reference, and to matched target genomes, T2T-NA12878 (1) and T2T-HG002 (25) .

Automated Contig-Breaking

PAF alignments generated during dot-plot preparation in *Patching Biological Data with GPatch* were further processed with a custom python script, included in the GPatch GitHub repository, to identify large-scale rearrangements within the patched genome that represent likely misjoins in the draft assembly. These misjoins can be between different regions of the same chromosome or between different chromosomes, and may represent any class of structural variant. These were identified as alignments or clusters of partial-alignments, within a maximum specified distance of each other that are inverted, translocated, or duplicated in the patched genome relative to the reference genome. To locate these, we looped over contigs in the input assembly, first retrieving all overlapping partial alignments based on their coordinates in the patched genome. Partial alignments were then clustered based on their chromosome, strand, and mapped position (i.e., position in the reference/target genome) such that partial-alignments within a maximum distance from each other, on the same chromosome and strand (including overlapping and nested alignments), are merged into a single interval. Merged intervals were then filtered to identify clusters representing inversions, duplications, and translocations: i.e., those that map to the reverse strand, or whose mapped position in T2T-CHM13 has shifted by a user-configurable minimum distance threshold of 1MB. Remaining intervals were then filtered to retain only those exceeding a user-configurable size threshold of 1MB. Breakpoint loci within the patched genome were then supplied to a second Python script to identify corresponding contigs from the source assembly and break them at the inferred positions, with output written in fasta format. These assemblies were then reprocessed with GPatch following the same steps described in *Patching Biological Data with GPatch.*

Dot-plots comparing GPatch results following contig breaking to T2T-CHM13 and corresponding target genomes were prepared as described in *Dot Plot Construction.*

Hi-C Data Mapping and Processing

Hi-C read data for NA12878, in FASTQ format, were obtained from the Sequence Read Archive (SRA) repository for Rao and Huntley, 2014 (28) (Additional File 1: Table S1). Given the extraordinary amount of sequence data for NA12878 generated in this study, we selected only the two largest FASTA files for processing (Additional File 1: Table S1), totalling approximately 1.75 billion reads. We utilized a custom Hi-C analysis pipeline based on the 4DN Hi-C Processing pipeline (29). Briefly, FastQC was used to verify the quality of Hi-C read data prior to mapping to a given reference genome with BWA-MEM (34). Mapped reads with quality scores 40 and above were extracted from the BAM output, converted into pairs format, sorted, and replicates merged, using a combination of pairtools (35) commands. Pairtools was further employed to mark and exclude duplicate pairs, and select only “UU”, “UR”, and “RU” contact pairs for further processing. Restriction enzyme fragment data were then superimposed on pairs data using the *fragment_4dnpairs.pl* script from the 4DN consortium. Finally, we used Juicer (30) to construct Hi-C matrices from the final pairs file and apply normalization using the KR method (28). All analysis steps were automated with Makefiles to ensure reproducibility, and were performed on a local compute cluster using memory-optimized nodes allocated with 500GB RAM, 12 3.0 GHz Intel Xeon Gold 6154 compute cores, and 4TB HDD. Hi-C loop calling was performed using the GPU-enabled version of hiccups, on a local server equipped with an NVIDIA Titan-V GPU and 270GB available RAM, using the Juicer Tools juicer_postprocessing.sh script (30). Loop calling was performed with hiccups (30) at 10kb and 5kb resolutions for Hi-C data mapped to both the T2T-CHM13 and GPatch NA12878 assemblies. Due to unexplained crashes of the hiccups software, loop calling could only be completed for resolutions 5kb and 10kb.

Hi-C Visualization

Normalized matrices in .hic format were browsed locally in Juicebox for initial comparison across stored resolutions before identifying 80kb as the optimal resolution for visualization. Hi-C heat map plotting for individual chromosomes was then automated with a custom Python script utilizing the CoolBox API (36). Plots were generated at 80kb resolution with balanced normalization, for both T2T-CHM13 and GPatch-NA12878 based Hi-C matrices, with plots stored in SVG format. SVG images for each dataset were then converted to PDF format using the online tool: https://tools.pdf24.org/en/svg-to-pdf (37), combined into a single PDF file in Adobe Acrobat, and reduced to sets of six thumbnails per standard letter sized page (landscape format) using the MacOS Preview tool. Thumbnails were extracted from the resulting PDF files, edited, and arranged into final figures in Adobe Illustrator, with the following edits applied uniformly to each image: 1) Bilateral scaling to 120%; 2) Adjust embedded heatmap and scale images to improve contrast using the “levels” tool from the Phantasm plugin (https://astutegraphics.com/plugins/phantasm), with input levels for the RGB channel set to 50:0.5:255, and output levels set to 0:240. Final figures were saved in PDF format.

Hi-C Loop Comparison

In order to compare the positions of individual loops and loop anchors between T2T-CHM13 and GPatch NA12878, we first prepared liftover chains using the nf-lo Nextflow pipeline (38), following steps described at https://genome.ucsc.edu/cgi-bin/hgTrackUi?hgsid=1390827233_st3mhvM8SnGwC9lK4FZ9ysMqCN54&db=hg38&c=chrX&g=chm13LiftOver, with T2T-CHM13 as the target and GPatch NA12878 as the source, and using minimap2 as the aligner, using the same mapping parameters as used previously in simulated and biological data analyses. Starting with loop predictions from the patched NA12878 assembly, loop predictions at 5kb and 10kb resolution were decomposed into individual loop anchor loci by splitting each line in the BEDPE source file into separate lines for the upstream and downstream loop anchors and assigning the BED name field for both with a matched ID number, with upstream and downstream anchors tagged with _1 and _2 suffixes, respectively. Resulting BED files were processed with the UCSC LiftOver tool (33) to convert from GPatch

NA12878 frame to T2T-CHM13 frame. We first counted loops for which one or both anchors failed to liftover, indicating presence in sequence unique to the GPatch NA12878 assembly. Next, BedTools (31) was used to identify lifted GPatch NA12878 loop anchors that overlap T2T-CHM13 loop anchors, using the command `*bedtools intersect -a <NA12878 anchors> -b <T2T-CHM13 anchors> -u*`. A custom python script was then used to reassemble intersecting anchors into complete or partial loop annotations in BEDPE format, with "missing" loop anchors, from loops where only one anchor intersects a T2T-CHM13 anchor, denoted by '.' characters in the affected BEDPE fields. From this, we were able to count completely and partially shared loops between GPatch NA12878 and T2T-CHM13. Loops unique to GPatch NA12878 were identified based on records in the "unmapped" fraction of the LiftOver results.

To determine which loops have anchors that are separated by at least one contig or patch boundary in GPatch NA18278, we decomposed loops into loop intervals, defined as the maximal interval between the upstream and downstream loop anchor coordinates, and stored these in BED format. Likewise, the single-base coordinates of 5' and 3' contig boundaries, in the frame of the GPatch NA12878 assembly, were extracted from the corresponding contigs.bed file generated within the patching step. We then used bedtools intersect, with options `-a <loop_intevals.bed> -b <contig_boundaries.bed> -wa -u` to reduce intersections between loop intervals and contigs to a list of unique loop intervals that overlap at least one contig boundary. This was repeated for loop calling resolutions 10kb and 5kb. Average loop lengths for 5kb and 10kb resolutions were calculated in awk by totaling the lengths of all loop intervals from each resolution and dividing by the number of total loop predictions.

References

1. Porubsky D, Dashnow H, Sasani TA, Logsdon GA, Hallast P, Noyes MD, et al. A familial, telomere-to-telomere reference for human de novo mutation and recombination from a four-generation pedigree [Internet]. Genomics. bioRxiv; 2024. Available from: https://www.biorxiv.org/content/10.1101/2024.08.05.606142v1.full.pdf

2. Gustafson JA, Gibson SB, Damaraju N, Zalusky MP, Hoekzema K, Twesigomwe D, et al. Nanopore sequencing of 1000 Genomes Project samples to build a comprehensive catalog of human genetic variation. medRxiv [Internet]. 2024 Mar 7; Available from: http://dx.doi.org/10.1101/2024.03.05.24303792

3. Liao WW, Asri M, Ebler J, Doerr D, Haukness M, Hickey G, et al. A draft human pangenome reference. Nature. 2023 May 10;617(7960):312–24.

4. Ebert P, Audano PA, Zhu Q, Rodriguez-Martin B, Porubsky D, Bonder MJ, et al. Haplotype-resolved diverse human genomes and integrated analysis of structural variation. Science. 2021 Apr 2;372(6537):eabf7117.

5. Nurk S, Koren S, Rhie A, Rautiainen M, Bzikadze AV, Mikheenko A, et al. The complete sequence of a human genome. Science. 2022 Apr 1;376(6588):44–53.

6. HG002: A complete diploid human genome [Internet]. Github; [cited 2025 Mar 26]. Available from: https://github.com/marbl/HG002

7. Simpson JT, Pop M. The theory and practice of genome sequence assembly. Annu Rev Genomics Hum Genet. 2015 Apr 22;16(1):153–72.

8. Myers EW. Toward simplifying and accurately formulating fragment assembly. J Comput Biol. 1995 Summer;2(2):275–90.

9. Bradnam KR, Fass JN, Alexandrov A, Baranay P. Assemblathon 2: evaluating de novo methods of genome assembly in three vertebrate species. Gigascience 2: 10. 2013; Available from: https://scholar.google.com/citations?user=0D3JtBAAAAAJ&hl=en&oi=sra

10. Baker M. De novo genome assembly: what every biologist should know. Nat Methods. 2012 Apr;9(4):333–7.

11. Wang T, Antonacci-Fulton L, Howe K, Lawson HA, Lucas JK, Phillippy AM, et al. The Human Pangenome Project: a global resource to map genomic diversity. Nature. 2022 Apr 20;604(7906):437–46.

12. Jiao Y, Peluso P, Shi J, Liang T, Stitzer MC, Wang B, et al. Improved maize reference genome with single-molecule technologies. Nature. 2017 Jun 22;546(7659):524–7.

13. Tang H, Zhang X, Miao C, Zhang J, Ming R, Schnable JC, et al. ALLMAPS: robust scaffold ordering based on multiple maps. Genome Biol. 2015 Jan 13;16(1):3.

14. Kolmogorov M, Armstrong J, Raney BJ, Streeter I, Dunn M, Yang F, et al. Chromosome assembly of large and complex genomes using multiple references. Genome Res. 2018 Nov 1;28(11):1720–32.

15. Aganezov S, Alekseyev MA. Multi-genome scaffold co-assembly based on the analysis of gene orders and genomic repeats. In: Bioinformatics Research and Applications. Cham: Springer International Publishing; 2016. p. 237–49. (Lecture notes in computer science).

16. Kim J, Larkin DM, Cai Q, Asan, Zhang Y, Ge RL, et al. Reference-assisted chromosome assembly. Proc Natl Acad Sci U S A. 2013 Jan 29;110(5):1785–90.

17. Jones P, Binns D, Chang HY, Fraser M, Li W, McAnulla C, et al. InterProScan 5: genome-scale protein function classification. Bioinformatics. 2014 May 1;30(9):1236–40.

18. Palmieri N, Nolte V, Chen J, Schlötterer C. Genome assembly and annotation of a Drosophila simulans strain from Madagascar. Mol Ecol Resour. 2015 Mar 1;15(2):372–81.

19. Kurtz S, Phillippy A, Delcher AL, Smoot M, Shumway M, Antonescu C, et al. Versatile and open software for comparing large genomes. Genome Biol. 2004 Jan 30;5(2):R12.

20. Tamazian G, Dobrynin P, Krasheninnikova K, Komissarov A, Koepfli KP, O'Brien SJ. Chromosomer: a reference-based genome arrangement tool for producing draft chromosome sequences. Gigascience. 2016 Aug 22;5(1):38.

21. Pop M, Kosack DS, Salzberg SL. Hierarchical scaffolding with Bambus. Genome Res. 2004 Jan 1;14(1):149–59.

22. Alonge M. RagTag: Tools for fast and flexible genome assembly scaffolding and improvement [Internet]. Github; [cited 2025 Mar 27]. Available from: https://github.com/malonge/RagTag

23. Alonge M, Soyk S, Ramakrishnan S, Wang X, Goodwin S, Sedlazeck FJ, et al. RaGOO: fast and accurate reference-guided scaffolding of draft genomes. Genome Biol. 2019 Oct 28;20(1):224.

24. Li H. Minimap2: pairwise alignment for nucleotide sequences. Bioinformatics. 2018 Sep 15;34(18):3094–100.

25. Garg S, Fungtammasan A, Carroll A, Chou M, Schmitt A, Zhou X, et al. Chromosome-scale, haplotype-resolved assembly of human genomes. Nat Biotechnol. 2021 Mar;39(3):309–12.

26. Jeffares DC, Jolly C, Hoti M, Speed D, Shaw L, Rallis C, et al. Transient structural variations have strong effects on quantitative traits and reproductive isolation in fission yeast. Nat Commun. 2017 Jan 24;8(1):14061.

27. Marçais G, Delcher AL, Phillippy AM, Coston R, Salzberg SL, Zimin A. MUMmer4: A fast and versatile genome alignment system. PLoS Comput Biol. 2018 Jan 26;14(1):e1005944.

28. Rao SSP, Huntley MH, Durand NC, Stamenova EK, Bochkov ID, Robinson JT, et al. A 3D map of the human genome at kilobase resolution reveals principles of chromatin looping. Cell. 2014 Dec

18;159(7):1665–80.

29. Hi-C Processing Pipeline [Internet]. [cited 2025 Apr 1]. Available from: https://data.4dnucleome.org/resources/data-analysis/hi_c-processing-pipeline

30. Durand NC, Shamim MS, Machol I, Rao SSP, Huntley MH, Lander ES, et al. Juicer provides a one-click system for analyzing loop-resolution Hi-C experiments. Cell Syst. 2016 Jul 27;3(1):95–8.

31. Quinlan AR. BEDTools: The Swiss-army tool for genome feature analysis. Curr Protoc Bioinformatics. 2014 Sep 8;47(1):11.12.1–34.

32. Li H, Handsaker B, Wysoker A, Fennell T, Ruan J, Homer N, et al. The Sequence Alignment/Map format and SAMtools. Bioinformatics. 2009 Aug 15;25(16):2078–9.

33. Perez G, Barber GP, Benet-Pages A, Casper J, Clawson H, Diekhans M, et al. The UCSC Genome Browser database: 2025 update. Nucleic Acids Res. 2025 Jan 6;53(D1):D1243–9.

34. Li H. bwa: Burrow-Wheeler Aligner for short-read alignment (see minimap2 for long-read alignment) [Internet]. Github; [cited 2025 Apr 1]. Available from: https://github.com/lh3/bwa

35. Open2C, Abdennur N, Fudenberg G, Flyamer IM, Galitsyna AA, Goloborodko A, et al. Pairtools: From sequencing data to chromosome contacts. PLoS Comput Biol. 2024 May 29;20(5):e1012164.

36. Xu W, Zhong Q, Lin D, Zuo Y, Dai J, Li G, et al. CoolBox: a flexible toolkit for visual analysis of genomics data. BMC Bioinformatics. 2021 Oct 10;22(1):489.

37. PDF24 Tools [Internet]. [cited 2025 Apr 17]. SVG to PDF converter. Available from: https://tools.pdf24.org/en/svg-to-pdf

38. Talenti A, Prendergast J. Nf-LO: A scalable, containerized workflow for genome-to-genome lift over. Genome Biol Evol. 2021 Sep 1;13(9):evab183.

Supplementary Information

Additional file 1: Supplementary-Figures.pdf. Supplementary Figures. Figures S1-S19.
Additional file 2: Supplementary-Tables.xlsx. Supplementary Tables. Tables S1-S5.

Declarations

Ethics approval and consent to participate

Not applicable

Consent for publication

Not applicable

Availability of data and materials

GPatch is open-source and available for download at https://github.com/adadiehl/GPatch under the MIT license. All analyses were performed using GPatch release 0.3.6, available at https://github.com/adadiehl/GPatch/archive/refs/tags/0.3.6.tar.gz. GPatch is implemented in Python, and designed to run on the Linux operating system. GPatch requires Python >= 3.7, samtools (https://github.com/samtools/samtools), biopython (https://biopython.org/), and pysam (https://github.com/pysam-developers/pysam). Custom scripts used to identify misjoin breakpoints and perform contig-breaking are included in the main github repository, and documented at

https://github.com/adadiehl/GPatch/tree/master/scripts. Tools in the scripts directory also require minimap2 (https://github.com/lh3/minimap2). All custom code and commands used in this analysis are documented and presented in the manuscript github repository at https://github.com/Boyle-Lab/GPatch-Manuscript. Sources for all publicly-available data used in this study are documented in Additional File 1: Table S1.

Competing interests

The authors declare that they have no competing interests.

Funding

This work was supported by the National Institutes of Health (R01GM144484).

Authors' contributions

AD designed and wrote the GPatch software, performed all data collection, preparation, and analysis steps, and prepared the figures and manuscript for submission. AB secured funding and provided oversight and guidance throughout the analysis. All authors read and approved the final manuscript.

Acknowledgements

The authors would like to acknowledge members of the Boyle lab for helpful discussions and feedback provided throughout the analysis and manuscript preparation processes.