Genome analysis

PopIns: population-scale detection of novel sequence insertions

Birte Kehr[1,]*, Páll Melsted[1,2] and Bjarni V. Halldórsson[1,3]

[1]deCODE genetics/Amgen, Reykjavík, Iceland, [2]Faculty of Industrial Engineering, Mechanical Engineering and Computer Science, University of Iceland, Reykjavík, Iceland and [3]Institute of Biomedical and Neural Engineering, Reykjavík University, Reykjavík, Iceland

*To whom correspondence should be addressed.
Associate Editor: Jan Korbel

Abstract

Motivation: The detection of genomic structural variation (SV) has advanced tremendously in recent years due to progress in high-throughput sequencing technologies. Novel sequence insertions, insertions without similarity to a human reference genome, have received less attention than other types of SVs due to the computational challenges in their detection from short read sequencing data, which inherently involves *de novo* assembly. *De novo* assembly is not only computationally challenging, but also requires high-quality data. Although the reads from a single individual may not always meet this requirement, using reads from multiple individuals can increase power to detect novel insertions.

Results: We have developed the program *PopIns*, which can discover and characterize non-reference insertions of 100 bp or longer on a population scale. In this article, we describe the approach we implemented in *PopIns.* It takes as input a reads-to-reference alignment, assembles unaligned reads using a standard assembly tool, merges the contigs of different individuals into high-confidence sequences, anchors the merged sequences into the reference genome, and finally genotypes all individuals for the discovered insertions. Our tests on simulated data indicate that the merging step greatly improves the quality and reliability of predicted insertions and that *PopIns* shows significantly better recall and precision than the recent tool *MindTheGap.* Preliminary results on a dataset of 305 Icelanders demonstrate the practicality of the new approach.

Availability and implementation: The source code of *PopIns* is available from http://github.com/bkehr/popins.

Contact: birte.kehr@decode.is

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

The latest version of the human reference genome (International Human Genome Sequencing Consortium, 2001; Venter *et al.*, 2001), GRCh38, is of a remarkable quality. However, the sequence of a single individual is inherently different from the reference due to sequence diversity. Some sequences are missing in the reference as they are not present in the individuals from whom the reference was constructed. Alternate haplotypes have been added to the reference genome (Horton *et al.*, 2008) to account for highly variable regions,

but they cover only a small part of the variation. The variable regions are of great biological and medical interest since their sequence diversity is known to affect phenotypes including numerous diseases (Conrad *et al.*, 2010; Stankiewicz and Lupski, 2010). Thus, the characterization of differences to the reference genome is a major task.

Differences between human genomes include single nucleotide polymorphisms (SNPs), small indels and structural variants (SVs). One type of SVs, which affect a larger piece of sequence than indels,

are insertions. Insertions can be further classified into duplications and novel sequence insertions. Duplications are insertions of sequence also present elsewhere in the genome, e. g. mobile elements. The focus of this work is on novel sequence insertions, insertions of unique sequence that is not similar to other regions of the reference genome. The evolutionary origin of a novel sequence insertion may be explained by two types of events. On the one hand, it may be an addition of sequence to the genome of a sequenced individual, e. g. lateral transfer or viral genome insertion. On the other hand, it may as well be a deletion of sequence in the individuals used to contruct the reference.

In order to successfully associate genetic differences with phenotypes, large numbers of individuals are necessary. With the enormous improvements in sequencing technology, recent years have seen a marked increase in large-scale efforts that aim at discovering variation at a population level, e.g. the HapMap project (Gibbs et al., 2003), the 1000 genomes projects (1000 Genomes Project Consortium, 2010, 2012) and the Genome of the Netherlands project (Boomsma et al., 2014). The achievements of these efforts have been the characterization of a great number of SNPs, indels, and deletions, but comparatively fewer novel insertions. For example, Mills et al. (2011) reported only 128 novel insertions in contrast to 22 025 deletions in their release set.

One reason for a smaller number of insertions discovered is the fact that their detection from short read sequencing data is challenging (Mills et al., 2011). Unlike detection of other types of variation, insertion detection requires a de novo assembly. Typical genotype callers, such as GATK (DePristo et al., 2011; McKenna et al., 2010) or FreeBayes (Garrison and Marth, 2012), use only reference-aligned read pairs and, therefore, are not suitable for calling insertions longer than the reads. Sequencing technologies that yield longer reads (English et al., 2012) show promise in simplifying insertion detection (Chaisson et al., 2014), but they are still not commonplace nor cost-effective on a large scale. Thus, insertions remain one of the most challenging types of variation to detect.

Strategies for detecting insertions incorporate either a local assembly of unknown sequence not present in the reference genome (Chen et al., 2014; Hajirasouliha et al., 2010; Holtgrewe et al., 2015; Rizk et al., 2014) or a whole-genome assembly (Bankevich et al., 2012; Gnerre et al., 2011; Li et al., 2010; Zerbino and Birney, 2008; Zimin et al., 2013). A whole-genome assembly needs to be followed by a comparison step of the assembled contigs to the reference genome for identifying the insertions. In the local assembly strategy, the positions of the insertions need to be identified either before or after assembly. For example, the strategy implemented in the program MindTheGap (Rizk et al., 2014) first identifies candidate insertion sites without a read alignment before initiating local assemblies, whereas the NovelSeq approach (Hajirasouliha et al., 2010) first assembles unaligned reads before anchoring them in the genome.

Both the whole-genome and the local assembly strategies face difficulties when integrating the results of many individuals. This issue is most pronounced in de novo whole-genome assembly strategies and, thus, has previously been addressed. Parrish et al. (2013) suggested a reference-guided whole-genome assembly approach that makes the results of several samples more compatible. Iqbal et al. (2012) developed Cortex, a program that rigorously assembles the whole genomes of several individuals at the same time based on colored de-Bruijn graphs. However, the tests in the Cortex paper were limited to relatively few individuals or to pooled data. Furthermore, all whole-genome assembly strategies commonly suffer from a considerable demand for computational resources.

In addition, the assembly problem demands high-coverage data (Miller et al., 2010; Zerbino et al., 2012). Assemblies from low-coverage data are typically incomplete, i.e. fragmented and with significant portions of the sequences missing (Alkan et al., 2011). Hence, the application of any of the mentioned approaches on a dataset from a single individual sequenced at insufficient coverage, results in a largely incomplete set of insertions. Polymorphic insertions with low-frequency in a population are particularly hard to detect as they are likely to appear only at heterozygous loci. Additionally, incomplete assemblies lead to greater difficulties in comparing and integrating sets of insertions across multiple individuals. If not carefully considered at the population level, all this can add to an underestimation of allele frequencies, less power to detect rare insertions, and may eventually impede association with phenotypes.

Despite the caveats mentioned, the task of analyzing large numbers of individuals can also aid in the detection of insertions. Insertions that occur within many individuals have an increased total coverage across the whole dataset. If used in the assembly step, this may reduce fragmentation and fill in gaps of the insertion sequences. The larger the number of individuals, the more likely it is that we can capture low-frequency insertions. Thus, instead of merging insertions detected from many individuals, we can take advantage of all individuals during the detection of insertions.

We have developed an approach for characterizing insertions across a large number of individuals simultaneously using a local assembly strategy. We start by assembling per individual reads that do not align to the reference genome. Subsequently, we merge the assemblies into a multi-individual contig set of higher quality. Each contig in this set is then placed into the reference genome using read-pair and split-read information. Finally, we propose a genotyping procedure that determines for each individual the number of copies it carries of an insertion in its diploid genome.

We have implemented the approach in a program called PopIns. Our tests on simulated data indicate that merging of single-individual assemblies increases the quality of insertion sequences by >20%. A comparison to MindTheGap (Rizk et al., 2014) confirms that we greatly benefit in recall from the merging step and that our approach is precise. An additional test on data from 305 whole genomes obtained with Illumina sequencers demonstrates its practicality on real data.

2 Problem formulation and approach

Given reference-aligned paired-end sequencing data from many individuals of the same species, the problem we deal with in this paper is to identify a set of long novel sequence insertions with respect to the reference genome and to determine the genotypes of all individuals for each insertion in the set.

2.1 Polymorphic long novel sequence insertions

An insertion in a multi-individual dataset is fully defined by three attributes: a position, a sequence and a genotype of each individual. The sequence of a novel sequence insertion is not similar to any part of the reference genome. By a long insertion, we here refer to a sequence longer than the sequencing reads of the given dataset. An insertion is polymorphic, if it is present in one or two copies of an individual's diploid genome and in one or more individuals within the given dataset at a frequency below 100%. The genotype of a single diploid individual determines the number of copies the individual carries of a polymorphic insertion. Possible genotypes are homozygous refererence (zero copies), heterozygous (one copy) or homozygous insertion (two copies).
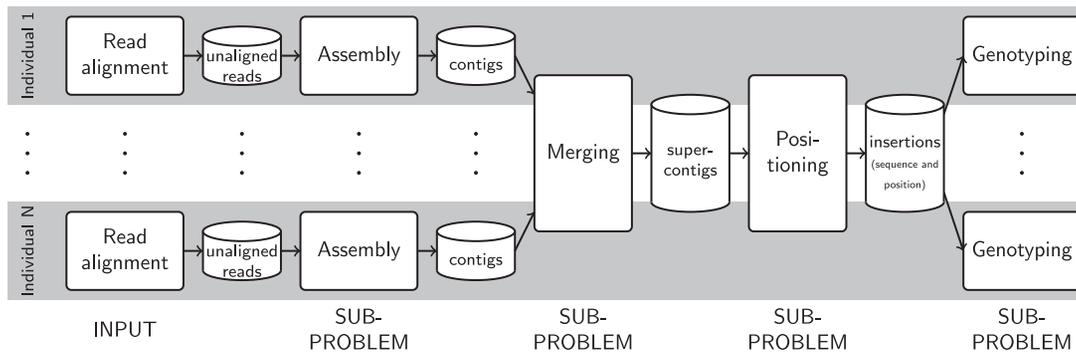
Fig. 1. Overview of our approach. Starting from a read alignment, we break up the insertion detection problem for many individuals into four subproblems: assembly, merging, positioning and genotyping

2.2 A local assembly approach for a single individual

Before we extend to multiple individuals, we consider the problem where a sequencing dataset is given for a single individual. We assume that this dataset has been aligned to a reference genome and refer to high-quality reads without alignment or with very low alignment scores as the set of unaligned reads. We define three separate subproblems, each addressing one of the three attributes of an insertion: the sequence, the position and the genotype. Consecutive solutions to these three subproblems form a local assembly approach for a single individual.

> SUBPROBLEM. (*Assembly*)
>
> INPUT: *The set of unaligned reads from a sequencing dataset of a single individual.*
>
> TASK: *Reconstruct a set of contigs, representing candidate sequences of insertions.*

We note that this problem is a classical genome assembly problem (Miller *et al.*, 2010).

> SUBPROBLEM. (*Positioning*)
>
> INPUT: *An alignment of reads to the reference genome, a set of contigs, and an alignment of the unaligned reads to these contigs.*
>
> TASK: *Determine the position on the reference genome where the contig is inserted.*

We assume a solution to this problem to provide exactly one position for each contig although in practice more than one position is possible.

> SUBPROBLEM. (*Genotyping*)
>
> INPUT: *A contig, a position, and an individual's reads that align to the contig and to the reference near the position.*
>
> TASK: *Classify the individual as homozygote reference, heterozygote, or homozygote insertion.*

2.3 A local assembly approach for multiple individuals

We now extend the single-individual problem to the case where aligned read data of multiple individuals is given. We could solve the above three subproblems for each individual separately. However, in order to obtain a single set of insertions for all individuals, an additional merging step is necessary.

If we merge the sets of insertions after solving all three subproblems, we do not benefit from multi-individual data and expect many false negative insertions due to low coverage. When genotyping is done after merging the sets, we can discover an insertion in all individuals even though the detection failed in some individuals. However, this leads to extra computational cost when solving the

positioning subproblem for similar contigs of multiple individuals. In addition, it is challenging to identify insertions in different individuals as the same insertion if the positions are inaccurate.

Therefore, we add a merging step after solving the assembly subproblem but before solving the positioning subproblem. This step merges sets of contigs into a single contig, which we call a *supercontig*. We say that a supercontig represents a contig if the full length of the contig aligns to the supercontig.

> SUBPROBLEM. (*Merging*)
>
> INPUT: *Given multiple sets of contigs, where each set corresponds to one individual.*
>
> TASK: *Reconstruct a set of supercontigs that fulfils the following two conditions*:
>
> 1. *Every input contig is represented by a supercontig in the set.*
> 2. *Any two supercontigs in the set cannot be further merged.*

We note the similarity of this subproblem to a classical assembly problem and discuss the differences in Section 5.

Our approach for detecting insertions in many individuals solves the subproblems assembly, merging, positioning and genotyping in this order as is illustrated in Figure 1.

3 Methods

This section presents our solutions to the four subproblems and describes the simulation of a test dataset. Details of our solutions to the subproblems can be found in corresponding subsections of Supplementary Section S1.

3.1 Assembly subproblem

As noted earlier, the assembly subproblem is a classical genome assembly problem. Careful work has been put into the development of tools for genome assembly from short-read data, among them *Velvet* (Zerbino and Birney, 2008), *SPAdes* (Bankevich *et al.*, 2012) and *MaSuRCA* (Zimin *et al.*, 2013). All these tools compute a set of contigs when given an individual's set of unaligned reads and, thus, can be applied to solve the assembly subproblem within our approach. We integrated *Velvet* into our implementation (Supplementary Section S1.1).

3.2 Merging subproblem

For the merging subproblem, we develop a practical solution that scales to hundreds of contigs (totaling to a length of hundreds of kilobases) from each of thousands of individuals. Formally, the inputs to the merging subproblem are sets of contigs $C_1, \ldots, C_n$, one
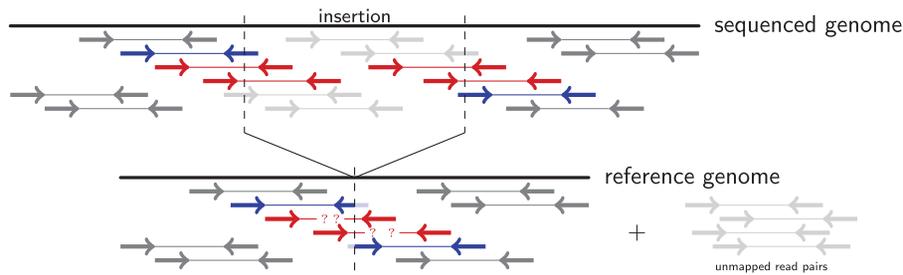
Fig. 2. The alignment of reads near a novel sequence insertion to the genome of a homozygous carrier of the insertion (top) and to the reference genome (bottom). Black read pairs correctly align to the reference genome, one end of red read pairs does not align to the reference, one end of each blue read pair is a split-read and gray read pairs do not align to the reference.

for each of $n > 1$ individuals. Let $\mathcal{C} = \cup\, C_i$. Our objective is to merge the contigs in $\mathcal{C}$ into a supercontig set $\overset{i}{S}$ of minimal size such that for each contig $c \in \mathcal{C}$ a supercontig $s \in S$ exists that represents c. We say that a supercontig s *represents* a contig c if c aligns to a substring of s with an error rate of at most ε (default $\varepsilon = 0.05$), where the error rate is calculated as the edit distance divided by the length of c.

We apply a two-step approach to solve the merging subproblem. It is based on the assumption that, in a minimal set of supercontigs, any two contigs c_p and c_q represented by the same supercontig $s \in S$ are connected by a path of contigs $c_p = c_{i_1}, \ldots, c_{i_r} = c_q$ such that any two adjacent contigs c_{i_x} and $c_{i_{x+1}}$, $i_1 \le i_x < i_r$ locally align to each other. Following this assumption, our first step aligns and partitions the original set of contigs $c \in \mathcal{C}$ into new sets $D_1, \ldots, D_k$, such that any two contigs that are elements of the same set D_j, $1 \le j \le k$ are connected by a path and any two contigs that are connected by a path are elements of the same set. The second step determines the supercontigs for each set $D_1, \ldots, D_k$. In the ideal case, all contigs in a set D_j, $1 \le j \le k$, originate from the same insertion and can be merged into exactly one supercontig.

In order to partition the contigs in $\mathcal{C}$ into sets $D_1 \ldots D_k$, we apply a union-find data structure on sets of contigs. We iteratively add the contigs in $\mathcal{C}$ to an initially empty union-find instance $\mathcal{D}$ such that, in the end, $\mathcal{D}$ represents the sets $D_1 \ldots D_k$. We unify two sets whenever we find an alignment between the current contig and any contig in each of the two sets. Instead of aligning all pairs of contigs with dynamic programming, we pre-filter for potential alignments with the fully sensitive k-mer counting algorithm *SWIFT* (Rasmussen *et al.*, 2006). Supplementary Section S1.2.1 gives further details on our partitioning step.

Subsequently we construct the supercontigs, separately for each set $D_1, \ldots, D_k \in \mathcal{D}$. This construction requires a multiple sequence alignment, which is known to be an NP-hard problem (Wang and Jiang, 1994). Thus, we apply an iterative approach similar to progressive alignment (Feng and Doolittle, 1987). In order to explore the possibility that the contigs do not assemble into a single contiguous supercontig, we use a graph data structure. Graph nodes are labeled with substrings of the contigs; the graph edges are directed and represent adjacencies of the contig substrings. After adding all contigs to the graph, we obtain supercontigs by following all paths through the graph. Supplementary Section S1.2.2 provides the details on how we iteratively add contigs to the graph.

3.3 Positioning subproblem

Our solution to the positioning subproblem uses information from read pairs as well as split reads. For a given insertion, we define an *anchoring read pair* as a pair of read ends where one end aligns to the reference genome and the other end to the insertion. A *split read*

is a read end without a full-length alignment to the reference genome or insertion, but with a position, the *split position*, that divides the read end into a prefix aligning to the reference genome and a suffix aligning to the insertion, or a prefix aligning to the insertion and a suffix aligning to the reference genome. Figure 2 shows how anchoring read pairs and split reads of an insertion align to the reference genome and to the genome of a homozygous carrier of the insertion. For anchoring read pairs, the orientations of the read ends in the alignments distinguish four different scenarios at a single genomic location (Supplementary Fig. S1). For distinguishing the four scenarios with split reads, it additionally matters whether the prefix or suffix aligns to the reference.

Our positioning approach considers the two ends of each contig separately. This allows the detection of more complex events than simple insertions, for example, an insertion of novel sequence that is followed by a mobile element. For each contig end, we first use anchoring read pairs to find the most probable locations in the reference genome and then split reads to determine the exact insertion positions at these locations.

We determine potential locations of a contig end by clustering anchoring read pairs using a greedy approach that scans linearly over the anchoring read pairs sorted by position to identify sets of read pairs that pairwisely support the same insertion. Afterwards, we compute an anchoring score between 0 and 1 for each location that takes into account all alternative locations. See Supplementary Section S1.3.1 for a more detailed description of the cluster and score computation. We keep only locations that have an anchoring score above a fixed threshold α (default $\alpha = 0.3$) and those that are shorter than twice the maximum allowed insert size (default $d_{max} = 800$) as all others are unlikely to be correct.

For each of the remaining locations, we search for the exact insertion position using split reads. Although the anchoring read pairs can give a first estimate of this position, split reads provide exact predictions of the position at base pair resolution. We collect potential split reads, align them using the split alignment function from the SeqAn C++ library (Döring *et al.*, 2008), and determine the insertion position from the whole set of split-aligned reads. See Supplementary Section S1.3.2 for details.

3.4 Genotyping subproblem

Given the sequence of an insertion, its position on the reference genome, and an individual's sequence read data. Our solution to the genotyping subproblem constructs the sequence of the two possible alleles at the insertion position R (reference) and A (alternate, i.e. insertion) in a window of size $2w$ (default $w = 50$), re-aligns the set of reads $\mathcal{R}$ to both alleles using dynamic programming (Supplementary Section S1.4 for details), and computes a relative likelihood for an

Table 1. Results of all steps of *PopIns* on simulated data

	Full sequence/both ends		Partial sequence/at least 1 end		Total simulated
Assembled (single individual)	62.3%	4 243	75.5%	5 111	6 768
Supercontigs after merging	85%	85	90%	90	100
Locations (from anchoring read pairs)	90%	90	90%	90	100
Positioned (with split reads)	84%	84	90%	90	100
Correctly genotyped	85.4%	5 783	90.5%	6 122	6 768

The column *full sequence/both ends* considers only those predictions where no more than 4 bp are missing from one of the insertion sequence's ends or both ends are positioned/genotyped. The column *partial sequence/at least 1 end* includes also predictions where one end of the insertion sequence is truncated or only one end is positioned or correctly genotyped. The numbers include one insertion sequence that is split into two supercontigs.

individual to carry the insertion on 0, 1 or 2 of his chromosomal copies based on alignment scores. As above, we consider both ends of the insertions separately.

We compute the likelihood based on the alignment scores of all reads $r \in \mathcal{R}$. Given a read r aligned to one of the two allele sequences $S \in \{A, R\}$ with score s_S, we assume $P(r|S) \sim e^{s_S}$. As we are only interested in relative likelihoods of R and A, we compute

$$P(r|S) \sim \frac{e^{s_S}}{e^{s_A} + e^{s_R}}$$

where s_A and s_R are the alignment scores to A and R, respectively. To ensure that a single read does not have too large of an impact in the joint computation of the genotype likelihood described later, we bound this relative likelihood from below by a small constant c (default $c = 0.0001$).

If we assume that reads are sampled from the two alleles of an individual with equal probability, we get the genotype likelihoods:

$$P(r|R, R) \sim \frac{e^{s_R}}{e^{s_A} + e^{s_R}}$$

$$P(r|R, A) \sim \frac{1}{2}\frac{e^{s_R}}{e^{s_A} + e^{s_R}} + \frac{1}{2}\frac{e^{s_A}}{e^{s_A} + e^{s_R}} = \frac{1}{2}$$

$$P(r|A, A) \sim \frac{e^{s_A}}{e^{s_A} + e^{s_R}}$$

Under the assumption that the reads in $\mathcal{R}$ are independent, we get the likelihoods for an individual's genotype $(S_1, S_2) \in \{A, R\} \times \{A, R\}$:

$$P(R|S_1, S_2) \sim \Pi_{r \in \mathcal{R}} P(r|S_1, S_2).$$

Finally, we pick the genotype that has the highest likelihood as our prediction and require this likelihood to be above 0.5 by default.

3.5 Test datasets

For performance evaluation, we simulated a dataset with polymorphic insertions at random positions. Furthermore, we selected a real dataset of 305 Icelanders as described later and data from the individual NA12878 as described in Supplementary Section S2.3. Supplementary Section S2 describes how we process the sequencing reads from all datasets to obtain a set of unaligned reads as input to the subproblems.

3.5.1 Simulation of data.

Our simulation is based on human chromosome 18, GRCh37 (hg19). We simulated 100 insertions that occur at different frequencies in 100 diploid individuals. In order to get realistic insertion sequences, we deleted 100 randomly selected regions from the reference chromosome using the resulting sequence as our reference and keeping the deleted sequences as insertions. We chose the lengths of these insertions according to an exponential distribution

with a mean of 100 and always added 100 bps. Next, we uniformly sampled a frequency between 0 and 1 for each insertion and added the insertion sequence back into 200 copies of the modified reference at the assigned frequency. The 200 copies of the modified reference each equipped with a subset of the insertions represent a set of 200 simulated haplotypes.

For each haplotype, we simulated sequencing reads at a coverage of 13×. From the Mason read simulation package, version 2.0 (Holtgrewe, 2010), we used the variator tool to add SNPs and small indels and the simulator tool to generate 5 million 101 bp paired-end Illumina reads per haplotype. Finally, we merged the sequencing reads of haplotype pairs to obtain data at a coverage of 26× for 100 diploid individuals. Our dataset created with this simulation procedure has an average of 32.19 heterozygous and 35.49 homozygous insertions per individual.

3.5.2 Real data of 305 individuals.

To test our approach on real sequencing reads, we selected the data of 305 Icelanders sequenced at deCODE genetics (Gudbjartsson *et al.*, 2015). These data were generated by Illumina HiSeq instruments in paired-end mode with a read length of 101 bp and an average insert size of 402 bp. The number of reads yielded sequencing coverages between 8× and 45× (average 24×).

4 Results

We implemented our solutions to the merging, positioning and genotyping subproblems in a program called *PopIns* using the SeqAn C++ library (Döring *et al.*, 2008). We assess the performance of *PopIns* on the simulated dataset, compare its results to that of the program *MindTheGap* (Rizk *et al.*, 2014), and demonstrate its practicality on the data of 305 Icelanders. Supplementary Section S3.3.1 and Table S2 describe an additional experiment on a single Icelandic individual that quantifies the benefit from the merging step. Further, Supplementary Section S3.3.2–S3.3.4 compare the performance of *PopIns* on data from the NA12878 individual to insertions validated by Kim *et al.* (2013) and insertions predicted in the 1000 genomes project (1000 Genomes Project Consortium, 2010).

4.1 Performance on simulated data

Table 1 summarizes the results of *PopIns* on our simulated dataset. The unaligned reads assemble into an average of 51.45 contigs per individual, which align to 75.5% of the 6768 simulated insertion sequences using *Stellar* (Kehr *et al.*, 2011) with a minimal length of 50 and a maximal error rate of 5%. After merging, the supercontigs align to 90% of the 100 simulated insertion sequences. With anchoring read pairs we find approximate locations for all supercontig ends. The anchoring score always points to the correct location if multiple locations are suggested. For 85 supercontigs, *PopIns* finds

Table 2. Comparison of *PopIns* and *MindTheGap* on simulated data

	Precision (%)	Recall (%)
PopIns	99.0	85.4
MindTheGap	81.9	69.5

an exact insertion position for both ends using split reads. For the remaining five supercontigs, *PopIns* finds an exact insertion position only for one end. These supercontigs turn out to be truncated with respect to the insertion sequence by five or more base pairs at the other end. Finally, *PopIns* genotypes 172 ends of insertions in all individuals correctly, reports a false position as heterozygous in 23 individuals, and discards 6 false positions in all individuals by typing them as homozygous reference. In relation to the 6768 single-individual insertions, this yields a recall of 85.4% and a precision of 99% after genotyping when counting only those insertions where both ends are fully characterized. Supplementary Section S3.1 provides further details of our evaluation on simulated data.

Furthermore, we made an experiment of pooling the unaligned reads of all individuals before assembly with velvet (using the same parameters except for an extended coverage window of [2 10000]). This yields only 14 contigs, a small number possibly reflecting assumptions made by velvet (e.g. on even coverage), a program that was developed for whole-genome assembly of a single individual.

4.2 Comparison to *MindTheGap*

We use our simulated dataset for comparison of *PopIns* to *MindTheGap* (Rizk *et al.*, 2014). We ran the index, find, and fill commands of *MindTheGap* with default parameters for each individual separately and selected all predicted insertions longer than 99 bps. In contrast to *PopIns*, *MindTheGap* first identifies potential insertion positions and then assembles their sequences. It reports each insertion as heterozygous or homozygous, which we interpret as the individuals genotype in our comparison. In many cases, *MindTheGap* reports several sequences per position. Following Rizk *et al.* (2014), we count only one sequence per position for calculating the precision, which gives a favorable representation of results of *MindTheGap*. We aligned all sequences to the set of simulated insertions (using *Stellar* as above) and counted a simulated insertion as found if 90% or more of the sequence aligned to a predicted sequence.

Table 2 shows the recall and precision of *PopIns* in comparison to *MindTheGap*. *PopIns* clearly outperforms *MindTheGap* in terms of recall. Further, the predictions of *PopIns* are more reliable in terms of precision, even though we count only one sequence per position for *MindTheGap*. We observed a much smaller difference in recall and precision between homozygous and heterozygous insertions for *PopIns* than for *MindTheGap* (data not shown). We count predictions per individual and consider an insertion as true positive if the position, sequence and genotype is correct.

4.3 Running times

PopIns and *MindTheGap* were run exclusively on Cisco 4 GHz machines with 16 GB of memory in a compute cluster. The total CPU time used by *PopIns* to finish all steps from assembly to genotyping for all individuals was 3:30 h. In comparison, *MindTheGap* used 35:54 h in total for the index, find and fill steps of all individuals. Thus, the computations of *PopIns* take ∼10 times less time. The difference is smaller between the actual walltimes that include

I/O operations but difficult to quantify as the walltimes depended heavily on the total cluster load. Supplementary Section S3.2 lists factors that influence running time in *PopIns*.

4.4 Performance on data of 305 human individuals

We tested the practicality of *PopIns* on the data of 305 Icelanders. After processing the data as described in Supplementary Section S2.2, we obtain an average of 35 531 unaligned reads per individual. The reads assemble into an average of 691 contigs per individual with an average N50 of 334 and totaling to an average length of 209 kbp. The merging step reduces the set of 210 892 contigs to 8437 supercontigs including 6141 contigs that are unique to one individual. We identify two sets of individuals (of size four and six) that have extra contigs, which are likely to originate from bacterial species (contamination). We exclude supercontigs that were only found in the contaminated individuals and thereby reduce the set of supercontigs to a size of 2226 of which only 401 are unique to one individual.

After aligning the unaligned reads to the set of supercontigs, anchoring read pairs suggest 263 732 locations for the supercontig ends, of which 2686 have an anchoring score above 0.3 and are supported by more than one anchoring read pair. For making a manual evaluation of the predicted insertions possible, we restricted further analysis to chromosome 18. Of all locations, 3968 fall on chromosome 18, of which 66 have an anchoring score above 0.3 and are supported by more than one anchoring read pair. Among the 66 locations, 46 can be paired into 23 records that explain the two ends of one supercontig. At 37 of the 66 locations we can determine the exact position from an unambiguous set of split reads, including 14 pairs for the two ends of one supercontig. At 36 of the 37 positions, the genotyping algorithm determines at least one individual to be a carrier of the insertion. The insertion frequencies in the 305 individuals range from 0.5 to 100%. Supplementary Figure S2 displays example read alignments for three individuals around one of the insertions, which visualizes that the data strongly supports the different genotype calls.

5 Discussion

We have introduced *PopIns*, a method for discovering and genotyping novel sequence insertions. *PopIns* takes advantage of the information provided by many individuals, increasing the quality of the insertion sequences and significantly improving on our ability to determine the correct insertion position. Our local assembly approach reduces computational requirements as compared to whole-genome assembly (Iqbal *et al.*, 2012). On the downside, our approach depends on a read alignment and, thus, will be biased against the reference.

The major novelty of our approach is the addition of a merging step to a local assembly strategy. The merging subproblem is *per se* an assembly problem but significantly different from the classical genome assembly problem. The input sequences of the merging subproblem are themselves assembled contigs and, hence, can contain artifacts from misassemblies. In addition, they vary in length and we expect them in the best case not to be much shorter than the supercontigs. Similar to transcriptome assembly, the coverage is uneven depending on the number of individuals that are carriers of an insertion and, finally, more than two haplotypes are possible for each insertion locus as the contigs originate from many individuals.

Nevertheless, our solution to the merging subproblem is similar to the overlap-layout-consensus (OLC) approach for genome

assembly (Miller *et al.*, 2010). The use of the union-find data structure is similar to the graph used in the overlap phase of OLC approaches; our sets of contigs correspond to connected components in this graph. Our approach differs from the OLC approach in the layout phase by allowing for branching components when constructing supercontigs.

When finding insertion positions, *PopIns* greedily clusters anchoring read pairs by location, while other SV detection methods solve a maximum clique problem (Marschall *et al.*, 2012; Rausch *et al.*, 2012). Our approach can lead to very long intervals for a single location. But since we cluster the anchoring read pairs only per contig and not over the whole dataset, we observe only very few abnormally long intervals. We discard these applying a length threshold as they are unlikely to lead to a clear insertion position.

In our evaluation, we were not always able to find the insertion position for both contig ends, which can have several reasons. If we find a single location with anchoring read pairs but no clear position with split reads, the contig is likely not to contain the whole insertion. The split alignment algorithm we used penalizes all gaps in the reference. Allowing for a large gap in both the reference and the reads may help in narrowing down the position of these insertions. Another reason may be non-unique sequence being inserted together with the novel sequence. The set of unaligned reads will not assemble into contigs of non-unique sequence (e.g. mobile elements), thus, these are missing in our approach. In many cases, this leads to many low-scoring locations suggested by read pairs that anchor to known occurrences of the repeated sequence. Finally, we observe read pairs connecting several contigs, suggesting insertions of novel sequence interspersed with non-unique sequence. An additional scaffolding step of supercontigs would be necessary to fully characterize these cases.

In many cases, we observe the same short sequence repeated at the two ends of an insertion (often referred to as target site duplications). If the repeated sequences become too long, our genotyping approach has difficulties in distinguishing the reference allele from the alternate allele. This may potentially be improved by focusing the computation on the unique part of the sequence.

Our results on simulated data do not reflect all of these limitations of *PopIns* that are due to the complex structure of real genomic sequences. Still, we could show the practicality of the approach on real data, where it yields many novel sequence insertions. Therefore, we are expecting a rich set of polymorphic insertion when applying *PopIns* to a larger number of individuals, which will open up the door to include novel sequence insertions in genome-wide association studies.

Conflict of Interest: none declared.

References

1000 Genomes Project Consortium. (2010) A map of human genome variation from population-scale sequencing. *Nature*, **467**, 1061–1073.

1000 Genomes Project Consortium. (2012) An integrated map of genetic variation from 1,092 human genomes. *Nature*, **491**, 56–65.

Alkan,C. *et al.* (2011) Limitations of next-generation genome sequence assembly. *Nat. Methods*, **8**, 61–65.

Bankevich,A. *et al.* (2012) SPAdes: a new genome assembly algorithm and its applications to single-cell sequencing. *J. Comput. Biol.*, **19**, 455–477.

Boomsma,D.I. *et al.* (2014) The genome of the Netherlands: design, and project goals. *Eur. J. Hum. Genet.*, **22**, 221–227.

Chaisson,M.J.P. *et al.* (2014) Resolving the complexity of the human genome using single-molecule sequencing. *Nature*, **517**, 608–611.

Chen,K. *et al.* (2014) TIGRA: a targeted iterative graph routing assembler for breakpoint assembly. *Genome Res.*, **24**, 310–317.

Conrad,D.F. *et al.* (2010) Origins and functional impact of copy number variation in the human genome. *Nature*, **464**, 704–712.

DePristo,M. *et al.* (2011) A framework for variation discovery and genotyping using next-generation DNA sequencing data. *Nat. Genet.*, **43**, 491–498.

Döring,A. *et al.* (2008) SeqAn an efficient, generic C++ library for sequence analysis. *BMC Bioinformatics*, **9**, 11.

English,A.C. *et al.* (2012) Mind the gap: upgrading genomes with pacific biosciences RS long-read sequencing technology. *PloS One*, **7**, e47768.

Feng,D.F. and Doolittle,R.F. (1987) Progressive sequence alignment as a prerequisite to correct phylogenetic trees. *J. Mol. Evol.*, **25**, 351–360.

Garrison,E. and Marth,G. (2012) Haplotype-based variant detection from short-read sequencing. *arXiv preprint arXiv:1207.3907 [q-bio.GN]*.

Gibbs,R.A. *et al.* (2003) The international HapMap project. *Nature*, **426**, 789–796.

Gnerre,S. *et al.* (2011) High-quality draft assemblies of mammalian genomes from massively parallel sequence data. *Proc. Natl. Acad. Sci.*, **108**, 1513–1518.

Gudbjartsson,D. *et al.* (2015) Large-scale whole-genome sequencing of the icelandic population. *Nat. Genet.*, **47**, 435–444.

Hajirasouliha,I. *et al.* (2010) Detection and characterization of novel sequence insertions using paired-end next-generation sequencing. *Bioinformatics*, **26**, 1277–1283.

Holtgrewe,M. (2010) Mason–a read simulator for second generation sequencing data. *Technical report TR-B-10-06*, FU, Berlin.

Holtgrewe,M. *et al.* (2015) Methods for the detection and assembly of novel sequence in high-throughput sequencing data. *Bioinformatics*, in press.

Horton,R. *et al.* (2008) Variation analysis and gene annotation of eight MHC haplotypes: the MHC Haplotype Project. *Immunogenetics*, **60**, 1–18.

International Human Genome Sequencing Consortium. (2001) Initial sequencing and analysis of the human genome. *Nature*, **409**, 860–921.

Iqbal,Z. *et al.* (2012) De novo assembly and genotyping of variants using colored de Bruijn graphs. *Nat. Genet.*, **44**, 226–232.

Kehr,B. *et al.* (2011) STELLAR: fast and exact local alignments. *BMC Bioinformatics*, **12**(Suppl. 9), S15.

Kim,S. *et al.* (2013) Reprever: resolving low-copy duplicated sequences using template driven assembly. *Nucleic Acids Res.*, **41**, e128.

Li,R. *et al.* (2010) De novo assembly of human genomes with massively parallel short read sequencing. *Genome Res.*, **20**, 265–272.

Marschall,T. *et al.* (2012) Clever: clique-enumerating variant finder. *Bioinformatics*, **28**, 2875–2882.

McKenna,A. *et al.* (2010) The genome analysis toolkit: a map reduce framework for analyzing next-generation DNA sequencing data. *Genome Res.*, **20**, 1297–1303.

Miller,J.R. *et al.* (2010) Assembly algorithms for next-generation sequencing data. *Genomics*, **95**, 315–327.

Mills,R.E. *et al.* (2011) Mapping copy number variation by population-scale genome sequencing. *Nature*, **470**, 59–65.

Parrish,N. *et al.* (2013) Genome reassembly with high-throughput sequencing data. *BMC Genomics*, **14**(Suppl. 1), S8.

Rasmussen,K.R. *et al.* (2006) Efficient q-gram filters for finding all epsilon-matches over a given length. *J. Comput. Biol.*, **13**, 296–308.

Rausch,T. *et al.* (2012) Delly: structural variant discovery by integrated paired-end and split-read analysis. *Bioinformatics*, **28**, i333–i339.

Rizk,G. *et al.* (2014) MindTheGap: integrated detection and assembly of short and long insertions. *Bioinformatics*, **30**, 3451–3457.

Stankiewicz,P. and Lupski,J.R. (2010) Structural variation in the human genome and its role in disease. *Annu. Rev. Med.*, **61**, 437–455.

Venter,J.C. *et al.* (2001) The sequence of the human genome. *Science*, **291**, 1304–1351.

Wang,L. and Jiang,T. (1994) On the complexity of multiple sequence alignment. *J. Comput. Biol.*, **1**, 337–348.

Zerbino,D.R. and Birney,E. (2008) Velvet: algorithms for de novo short read assembly using de Bruijn graphs. *Genome Res.*, **18**, 821–829.

Zerbino,D.R. *et al.* (2012) Integrating genomes. *Science*, **336**, 179–182.

Zimin,A.V. *et al.* (2013) The MaSuRCA genome assembler. *Bioinformatics*, **29**, 2669–2677.