METHOD

Bcmap: fast alignment-free barcode mapping for linked-read sequencing data

Richard Lüpken[1*], Thomas Krannich[2] and Birte Kehr[1,2,3*]

[*]Correspondence: richard.luepken@bih-charite.de; birte.kehr@ukr.de
[1]Genome Informatics, Berlin Institute of Health at Charité–Universitätsmedizin Berlin, 10117 Berlin, Germany
[2]Algorithmic Bioinformatics, Leibniz Institute for Immunotherapy (LIT), 93053 Regensburg, Germany
Full list of author information is available at the end of the article

Abstract

The bottleneck for genome analysis will soon shift from sequencing cost to computationally expensive read alignment. Here, we propose to circumvent full read alignment computation on linked-read sequencing data when only portions of the genome are of interest. Our novel barcode mapping approach 'bcmap' indexes and places barcoded read sets on the reference genome, thereafter allowing the extraction of reads originating from genomic regions of interest directly from FASTQ files. We demonstrate that bcmap is accurate and an order of magnitude faster than full read alignment. As example application, we use bcmap to locally assemble a known variant region.

Keywords: linked-read mapping; alignment-free; sequence barcodes; minimizers; targeted genome analysis; short-read data

Background

Genomic analyses frequently focus on selected regions of the genome. Especially analyses with a focus on particular diseases are often limited to few loci [1], small gene panels [2, 3] or exonic regions only [4]. As the costs for sequencing drop and the 100$ genome comes into reach, whole-genome sequencing may soon replace panel and whole-exome sequencing for a range of applications. Whole-genome sequencing provides more flexibility regarding the genomic regions to analyze, that is, an analysis can incrementally be extended to additional or larger regions. However, whole-genome data sets are substantially larger than data from targeted sequencing approaches. Handling these large amounts of data can quickly exceed available computational power and, thus, may hinder the decision to generate whole-genome data. Even with the already available amounts of whole-genome sequencing data, reducing computational cost is desirable. Cheap data analysis could, for example, enable more researchers to confirm findings regarding a single or small set of genomic loci on a large scale, using publicly available whole-genome data sets.

In addition to choosing between targeted or whole-genome sequencing, sequencing endeavors are facing a trade-off in choosing between short and long reads. Short reads are comparably cheap and can easily be obtained at large scale while offering a high per base accuracy. The confined span of short reads, however, limits their effectiveness for sequence analyses, especially in complex and repetitive regions of the genome. Long reads [5] provide the missing long-range information at the cost of high per base error rates. While circular consensus approaches like HiFi reads [6]

offer long read lengths combined with low error rates, they are currently expensive and require large amounts of input DNA.

Linked-read sequencing technologies [7, 8, 9] address the mentioned weak points of both short and long reads. Linked reads resolve some complex and repetitive regions of the genome by providing long-range information over distances similar to long reads, while retaining the high accuracy and relatively low cost of short reads. Essentially, linked reads are short reads labeled with barcodes. Reads with the same barcode originate from a small set of large DNA molecules. This equips the data with an additional level of structure which, as we show below, can be used to tackle the computational bottleneck when analyzing whole-genome data.

Previously, the barcode information of linked reads has been used to improve the quality of read alignments [10, 11]. Building on this, linked-read information has shown to assist in variant calling [12], in particular for structural variants [13, 14, 15], and haplotype phasing [7]. Additionally, the provided long-range information has shown to assist in whole-genome assemblies [16, 17, 18]. All these applications of linked reads tackle problems previously addressed with short-read data and use the barcode information for improving the quality of the results. We believe that the technology provides additional opportunities for new analyses and new ways of analysis that are to be explored, for example, improving the computational efficiency of genome analysis.

Typically, read alignment makes up the bulk of the computational analysis and hence contributes a significant portion to the computational, and consequently also to the monetary cost of genome analysis. Therefore, replacing the expensive alignment with a faster approach can substantially speed up the analysis and may eventually be necessary to establish the 100$ genome for a wider range of analyses.

Here, we introduce bcmap, an approach that quickly assigns linked-read barcodes to genomic intervals. We refer to this task as barcode mapping although we are not mapping the barcode itself but rather the sets of reads labeled with the same barcode. In previous work, it was shown how the contextual information of linked-read barcodes can be used to map reads to a reference genome which otherwise could not or only ambiguously be placed [10, 19]. While still taking advantage of this effect we use another aspect of this contextual information. A set of reads sharing a barcode contains much more positional information than a single read pair, such that it is possible to map whole barcodes to a reference with comparable accuracy to an alignment using an alignment-free approach.

In short, bcmap uses k-mer hash tables and window minimizers to swiftly map barcodes to the reference whilst calculating a mapping score. Simultaneously, bcmap constructs an index on the barcode-sorted input read files that allows us to efficiently retrieve all reads of a given barcode. After sorting the barcode mappings by genomic position, the barcode mappings together with the read file index allow the efficient retrieval of reads originating from genomic regions of interest.

When faced with WGS data but interested in a specific genomic location, bcmap can return all barcodes and reads belonging to a region of interest in approximately 60 times less time than BWA needs for computing a read alignment. As opposed to other work focusing on alignment speed [20], we achieve this speed-up on commodity hardware. In addition, we show how our barcode mapping score can be used as a

quality measure of barcode mappings and consequently for filtering the output. Finally, we demonstrate how an example targeted analysis can benefit from the tremendous savings in computational time by assembling a known non-reference sequence variant from a read set extracted with bcmap.

Results

We implemented our novel approach in the software tool bcmap that offers three commands: bcmap index, bcmap map, and bcmap get. The index command creates a k-mer index of the reference genome. The map command assigns barcodes to intervals on the reference genome, computes the barcode mapping scores and, simultaneously, creates the index on the input read files. The get command retrieves reads labeled with specific barcodes using the read file index. For simple retrieval of reads from a region of interest, we prepared a Snakemake workflow [21] that chains these bcmaps commands with standard unix commands. Our evaluations primarily focus on the accuracy of the map command while benchmarks of the computational performance consider the entire workflow for extracting reads from regions of interest.

We compared our barcode mapping, i.e. bcmap map, to read alignment even though barcode mapping and read alignment are two different tasks with two different aims. We fully acknowledge that read alignments contain much more information than barcode mappings. It is even possible to reduce a read alignment to a barcode mapping by clustering positions of aligned reads for every barcode. However, we argue that full read alignment is the task that can potentially be omitted by performing barcode mapping. Similar to barcode mappings, read alignments allow us to extract sets of reads originating from genomic regions of interest, e.g., using samtools [22].

We evaluated bcmap on simulated data of human chromosome 21 (Supplementary Section 2.2) and on real 36x coverage linked-read WGS data, i.e., the first lane of the NA12878/HG001 benchmark genome generated using the 10x Genomics Chromium controller followed by Illumina sequencing (Availability of data and materials). In all experiments, we used GRCh38 (Availability of data and materials) as the reference genome. On the real WGS data, we assessed the computational performance of bcmap in terms of running time and memory consumption (Section Computational performance). On simulated data, we compared precision-recall curves for barcode mappings output by bcmap to barcode mappings obtained by clustering reads aligned with different read aligners (Section Accuracy on simulated data). To demonstrate how bcmap can serve as a pre-processing step for targeted analyses, we show that a known non-reference sequence variant can be detected in the bcmap extracted read set of NA12878 via local assembly (Section Local sequence assembly). For justifying the choice of default parameters, we further evaluated the impact of minimizers on the index size and construction time on the real data (Section Impact of minimizers).

We limit our comparisons of bcmap to LongRanger [11], EMA [10] and BWA MEM [23]. We chose LongRanger as it is widely used for aligning linked-read data. The tool EMA was shown to outperform LongRanger in this task in terms of computational performance. Finally, BWA serves us as a baseline representing aligners that do not account for barcodes.

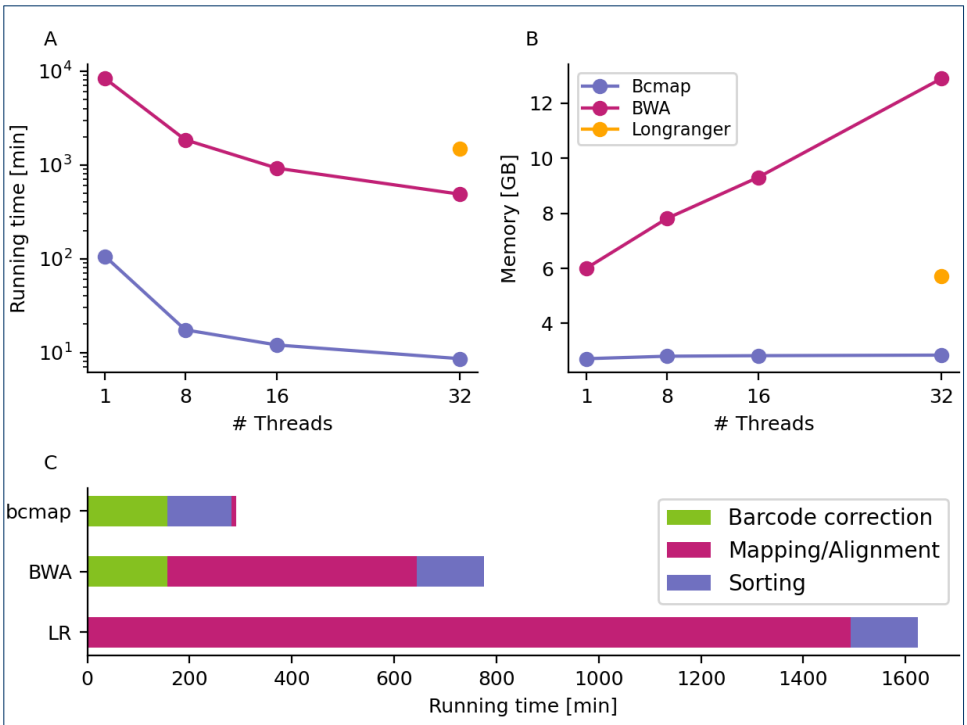

Figure 1 Comparison of running time (A) and memory consumption (B) on different numbers of threads, and workflow running times on 32 threads (C) for bcmap map, bwa mem and Longranger align (LR) on 36x coverage WGS linked-read data from the NA12878 genome. Workflow running times include barcode correction and sorting of reads in addition to barcode mapping or read alignment. For Longranger, barcode correction is included in the main align command.

Computational performance

We compared the running time and memory consumption of bcmap map to that of BWA MEM and Longranger align on the NA12878 data. We excluded EMA from this analysis as it is using BWA internally and will require a similar yet slightly more computational resources. All tests were performed on an Intel Xeon E5-2650 v2 @2.60Ghz, 16 cores x2 threading. Our assessment is not meant to compare the performance of bcmap to read aligners but rather to demonstrate potential time savings for tasks which do not require a full read alignment.

Barcode mapping with bcmap proves to be significantly faster than a read alignment, outperforming BWA roughly 60-fold in this regard (Figure 1A). The long running time of LongRanger is partially due to the fact that LongRanger includes a barcode trimming and correction step in its align command. For bcmap and BWA, barcode correction was performed with our more efficient software bcctools (`https://github.com/kehrlab/bcctools`) as a pre-processing step and its running time is not included in the given timings. In terms of memory consumption bcmap also outperforms the alignment tools, requiring less than 3GB of RAM and scaling well when using more threads (Figure 1B). Exact values of the running time and memory consumption are given in Supplementary Table 1.

Figure 1C shows the running times (wall clock times on 32 threads) for a complete mapping workflow using the compared tools. The workflows of all tools include a correction step of the barcodes and sorting step of the input reads. For the alignment

tools, the sorting was performed using samtools sort after alignment. Here, sorting is necessary in order to allow the quick extraction of reads from a region of interest. For bcmap, sorting was performed by barcode using Unix sort after barcode correction and before barcode mapping. The computational cost of sorting is similar in all workflows, which is why it is safe to exclude these steps from the comparison in Figure 1A and 1B.

Accuracy on simulated data

To evaluate the accuracy of bcmap, we simulated data using LRSIM [24] (Supplementary Section 2.2). We ran LRSIM for chromosome 21 of GRCh38 with and without introducing variants. Since LRSIM does not output the true positions of underlying simulated DNA molecules but only the true positions of individual reads on the reference genome, we used a simple clustering procedure to infer the truth set of barcode mappings from the data (Supplementary Section 2.2.3). To guarantee a conservative comparison of bcmap, we used the same clustering procedure on the alignments of reads labeled with the same barcode output by Longranger, EMA and BWA for evaluating their barcode mapping accuracy.

We compared the output of bcmap and the clustered output of Longranger, EMA and BWA to the truth set. We define a predicted barcode mapping as a true positive if there is a mapping in the truth set for the same barcode, overlapping the predicted mapping, and with start and end positions deviating by less than 20,000 bases, respectively. A predicted mapping that has no such equivalent in the truth set is defined as a false positive and a true mapping without a predicted equivalent is defined as a false negative. Furthermore, we demand a one to one correspondence for true positives, meaning that a true mapping can only qualify one predicted mapping as a true positive.

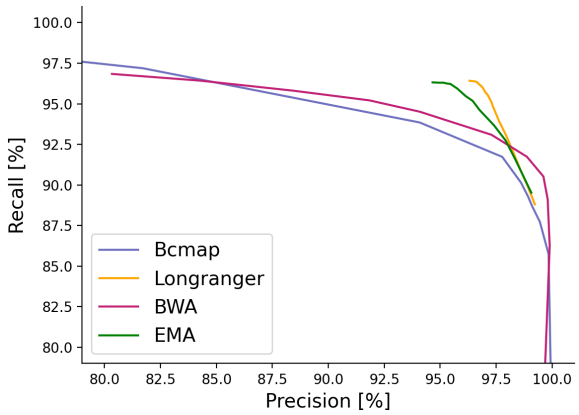

Figure 2 Precision-recall curve for barcode mappings with bcmap, BWA and EMA compared to the truth set on simulated data without variation.

Figure 2 and Supplementary Figure 3 display the resulting precision-recall curves for different mapping score thresholds of bcmap and different minimum numbers of reads per cluster for the read aligners for the simulated data sets without and with variation. The accuracy achieved by bcmap is almost as good as the accuracy achieved by BWA. The results for EMA and LongRanger are very similar to each

other, with Longranger achieving consistently better precision than EMA. Both barcode-aware aligners produce arguably better precision recall ratios than BWA and bcmap, whereas BWA and especially bcmap can reach higher peak values for both precision and recall.

Evaluation of barcode mapping scores

On the simulated data, we observed that higher mapping scores in bcmap lead to a steadily increasing precision. This observation indicates that our mapping score does indeed serve as a reliable quality measure. To further confirm that our mapping score is suitable for distinguishing true positive from false positive barcode mappings, we examined the score distributions in different data sets.

The discrete values for the score are expected to vary between different data sets and different reference genomes. For example, a larger reference with fewer unique or rare k-mers will lead to lower scores overall (see Methods). This makes it difficult to define a general score threshold. A threshold that works for one sample or reference genome may harm precision or recall in another setup.

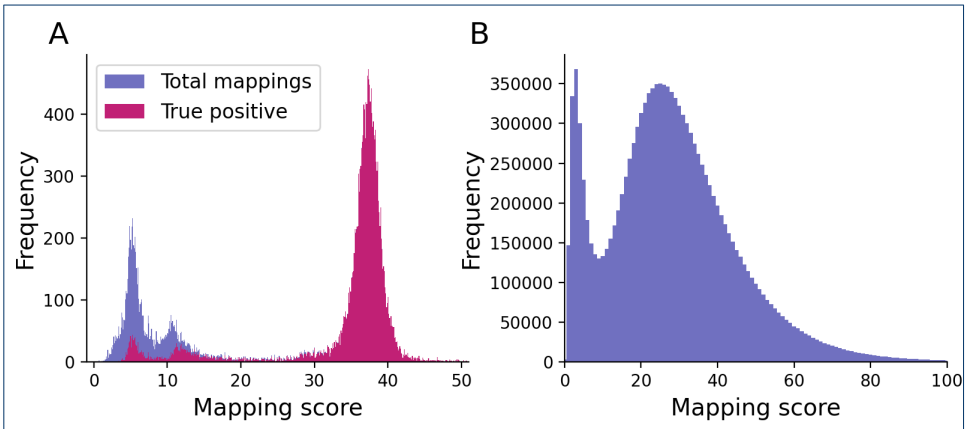

Figure 3 Mapping score distribution for (A) simulated data with true positives highlighted and (B) NA12878 data.

However, in all our tests the score distribution followed a certain pattern shown in Figure 3: Two peaks of frequent scores with a dip in between. Figure 3A shows the true positive rate based on this distribution. Mappings with scores on the right side of the dip are almost exclusively true while mappings with lower scores can be either true or false. On real data the pattern allows us to infer a score threshold for reliable barcode mappings, e.g., a mapping score of at least 10 for the NA12878 data (Figure 3B).

Local sequence assembly

As an example downstream analysis, we tested whether bcmap can be applied as the initial step of a local sequence assembly workflow for a given target region of the human genome. We selected the genomic position chr17:17,831,079 on the GRCh38 human reference genome that was previously found to be the breakpoint of a 766 bp non-reference sequence variant associated with a lower risk of myocardial

infarction [25]. The Genome-in-a-Bottle consortium previously reported NA12878 to be a carrier of this variant [26].

We applied bcmap to the linked-read data of NA12878 and extracted the set of reads labeled with barcodes that had a mapping overlapping a 50 kbp window around the variant position, i.e., 25 kbp to each side (Supplementary Section 2.6.1). Note that this read set is contaminated with a high proportion of reads originating from other genomic positions as a single barcode can label multiple DNA molecules. Given that the physical coverage of linked-reads is low, most of these spurious reads do not lead to unitigs larger than a single short read.

The set of reads was passed to the Minia assembler [27] of the GATB library [28] and the graph structure output by Minia was further simplified using Bifrost [29] (Supplementary Section 2.6.2). As a result, we obtained a set of 747 unitigs with a total length of 373894 bp, an N50 of 840 bp, and a median length of 254 bp.

We aligned the unitigs to the reference genome using BLAT [30] and found the alignment of one of the ten longest unitigs (of length 4887 bp), to span the expected breakpoint (Supplementary Section 2.6.3). Within this unitig, a continuous subsequence of 766 bp that did not align to the reference genome was confirmed to be the sought variant (Supplementary Section 2.6.4).

Impact of minimizers on computational performance

The barcode mapping approach in bcmap owes its computational performance to the use of minimizers. We quantified the impact of minimizers in bcmap on the construction time and size of the k-mer index build for the reference genome as well as on barcode mapping.

We constructed the k-mer index of GRCh38 using 31-mers and minimizer window sizes w spanning from 31 to 61, where $w = 31$ corresponds to the case that no minimizers are used. The time for constructing and the disc space required by the index (Table 1) scale according to the number of minimizers that are required to cover the reference (Equation 1 in Methods).

For the same parameters, we mapped the barcodes of the NA12878 data (36x coverage) using the GRCh38 index. The impact on the memory consumption of barcode mapping is is dominated by the size of the index. The impact on the running time of barcode mapping is even squared compared to indexing since both the number of evaluated minimizers and the number of hits per evaluated minimizer scale up. While the use of minimizers is clearly beneficial in terms of computational performance (Table 1), the choice of w has no measurable effect on the accuracy of the barcode mapping over the tested range of parameter values (Supplementary Figure 4).

Discussion

Here, we demonstrate the performance of our novel barcode mapping approach on simulated data and on data generated with the 10X Chromium technology [7]. We do not demonstrate bcmap's performance on linked-read data generated with other technologies, e.g., MGI's stLFR data [8] or TELL-Seq data [9]. In contrast to other linked-read analysis tools that are tied to special linked-read protocols (e.g., Longranger is tied to 10x Genomics linked reads), our barcode mapping approach

Table 1 Impact of minimizer window size w on the performance of bcmap.

	w	31	41	51	61
bcmap index	time	62:57 min	12:10 min	7:04 min	5:38 min
	space	37.67 GB	6.63 GB	3.52 GB	2.41 GB
bcmap map	time	22.25 h	26:49 min	12:42 min	8:35 min
	memory	41.01 GB	7.48 GB	4.05 GB	2.85 GB

We show wall clock time and disc space or memory usage of bcmap index and bcmap map for different values of w. The tests of the indexing were performed on Intel(R) Xeon(R) Gold 5115 CPU @ 2.40GHz (40 CPUs) using multithreading. The timings for the mapping were recorded on Intel Xeon E5-2650 v2 @2.60Ghz, 16 cores x2 threading using multithreading.

implemented in bcmap is independent of the given type of paired-end linked-read data. Requiring as input a barcode-sorted pair of FASTQ files, bcmap will most likely neither need an update to work with future linked-read data types.

For targeted approaches, the slightly inferior mapping accuracy of bcmap compared to barcode aware alignment methods may pose an issue for some applications. However, bcmap achieved the highest recall in our comparison, and false positives appeared mainly at low scores. Therefore, it is possible to counteract the overall lower accuracy by validating mappings with a low score using a barcode-aware aligner. As the alignment would only have to be performed for a small subset of the available reads, we expect this approach to still be reasonably efficient and yield better results than any single method in our comparison. We leave the experiment of validating barcode mappings as a promising future direction.

Alongside the genomic positions, the barcode mapping returns a mapping score. The provided score can be used in various ways. The score can be defined as a quality weight factor for downstream analyses or it can be used to define a score threshold based on the distribution shown in Figure 3, limiting the output to high-precision mappings. The predictable nature of score distributions can be used to configure the precision versus recall trade-off. As there are still some true mappings reported at low scores we do not recommended to immediately discard all mappings with low scores. Depending on the downstream analyses, it can be worthwhile to retain the entire mapping set and take the mapping score into account.

For both the index of the reference genome and the barcode mapping bcmap uses window minimizers. Using minimizers for the index reduces the required memory significantly and improves the running time. Intuitively, this bears the risk of missing out on potential k-mer hits whenever small variants or sequencing errors change the minimizer in a read to a k-mer which is present in the reference but not in the minimized index. One could expect this to lead to a drop in mapping quality. At the same time, the contextual information of a minimizer hit (the k-mer is the minimizer for both the read and the reference window) improves the mapping evidence. Our experiments have shown that the benefits of a minimized index outweigh the disadvantages and lead to an increase in both precision and recall of barcode mappings (data not shown). Nevertheless, bcmap offers the possibility to build the k-mer index without minimizers although we discourage from doing so.

Another possible use case for our approach is approximate mapping of long reads. Long reads provide large amounts of contextual sequence information, which is why we hypothesize that an approximate mapping approach works similarly well for long reads as it does for linked reads. In addition, it seems rather straightforward

to adopt our implementation to work on long reads. Exploring the pitfalls of this will be subject of future work.

Conclusion

With bcmap we introduce a tool that allows for fast retrieval of linked read sets originating from genomic regions of interest. While being significantly faster and more memory efficient than alignment based approaches, bcmap yields results of comparable quality. Consequently bcmap contributes towards shifting the computational bottleneck of genome analyses from read alignment to other pre-processing steps like barcode correction and sorting of reads. Thanks to these properties, bcmap makes it possible to effectively apply any kind of targeted genome analysis method to linked-read whole-genome sequencing data without the need for computing a full read alignment. Prospectively, bcmap can be used to assist genome-wide local assembly approaches [13] by providing the required local read sets much more efficiently than read aligners.

Methods

The approach implemented in bcmap takes as input a barcode-sorted pair of FASTQ files. It solves the task of barcode and read retrieval from regions of interest by first creating a barcode index file and a read index file from the input set of barcoded read pairs. With the help of the created index files, reads from genomic regions of interest can then be retrieved efficiently from the input FASTQ file. The barcode index maps genomic intervals to barcodes and allows the efficient retrieval of barcodes in given regions of interest. The read index file stores positions of barcodes in the input FASTQ file and allows efficient retrieval of the reads labeled with a given barcode. For creating the barcode index file, bcmap additionally builds a k-mer index of the reference genome. This index needs to be built only once per reference genome and can be used for analyzing any number of genomes. The index of the reference genome is analogous to the index files used by read aligners and stores the genomic positions of all window minimizers present in the reference.

Below we describe details of the three bcmap commands. In short, the index command creates the k-mer reference index, the map command creates the barcode and the read index using the reference index, and the get command can be used to retrieve reads of specific barcodes using the read index.

Bcmap index

The k-mer index created by the bcmap index command stores the genomic positions of all minimizers present in the reference. It is constructed in a way that allows one to look up the frequency of a minimizer before accessing all associated positions. This feature allows for faster read analysis in the barcode mapping step (Section Bcmap map).

To index the reference genome we used a minimized open-addressing k-mer index [31]. An open-addressing index follows the same principles as a regular hash table but it allows for ambiguity in the hashing. Therefore, it massively reduces the size of the index for large values of k (i.e. k=31) as the required memory scales with the length of the indexed sequence instead of the number of possible k-mers.

The size advantage comes at the cost of slightly longer lookup times because hash collisions caused by the introduced ambiguity have to be resolved. We resolve collisions by performing a predefined bit-shift on the hash value until the correct index position is found. The lookup procedure of an open-addressing index is shown in Supplementary Figure 1.

Because we are using minimizers to process the read data later on, we can use the same minimizing strategy for constructing the index. We used a window based minimizing technique where a minimizer is the k-mer with the lowest hash value within a fixed size window w (default $k = 31$, $w = 61$). In contrast to global minimizers, where a subset of k-mers are predefined as minimizers, window minimizers take more contextual information into account and hence retain more of the unique sequence information. To avoid bias towards certain sequences and thereby reduce collisions during index look ups, we used a random order to determine minimizers. The random order is achieved by performing a bit-wise *xor* operation between the bit-vector representation of the k-mer and a consistently reused random bit-vector of length $2k$.

In detail, the index consists of three tables, the identifier table, the directory table and the positions table. The size of the tables depends on the size of the indexed reference l, the k-mer length k and the size of the minimizer windows w. The identifier table stores an identifier of the contained minimizer and is used to resolve ambiguity from the open addressing. We choose the size of the identifier table as

$$S_I = \frac{l}{\frac{w-k+1}{2}} = \frac{2l}{w-k+1} \tag{1}$$

If a minimizer stays the minimizer for B_m consecutive minimizer windows we say that it is active for B_m bases. As the maximum amount of bases for which a minimizer can stay active is $w - k + 1$, on average a minimizer will be active for about half this long [32]. Hence, on average S_I minimizers are necessary to cover the entire reference sequence. Because some minimizers will appear multiple times in the reference, fewer different minimizers than S_I will have to be stored in the index. This circumstance ensures that placing minimizers in the identifier table results in only a small number of collisions. The directory table links the identifier table to the positions table and is of size $S_I + 1$. The positions table stores the reference positions of minimizers ordered by minimizers. The size of the positions table is determined during index construction so that the table exactly fits the number of minimizers. If a minimizer is stored at index position i of the identifier table, then position i of the directory table points to the first entry in the positions table that holds an occurrence of this minimizer. Position $i + 1$ of the directory table points to one entry past the last reference position of the minimizer (the first position of the next minimizer). The difference between position i and $i + 1$ of the directory hence equals the number of occurrences of the minimizer in the reference genome. To save memory, we store no more then 16 bases of a minimizer as an identifier in the identifier table. For $k > 16$ this bears the theoretical risk of storing the positions of multiple minimizers in the same entry. However, as the whole base content is used

for the hashing to determine the table position, collisions of this kind are extremely unlikely.

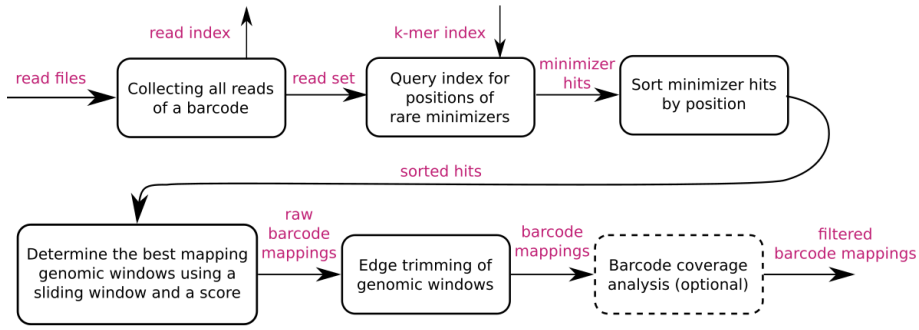

Figure 4 Workflow of the bcmap map command. The map command creates a read index in the very first step and the barcode index as the final output. It takes as input the barcode-sorted reads in FASTQ format and the k-mer index of the reference genome.

Bcmap map

An overview of our workflow for mapping barcodes to the reference genome is depicted in Figure 4. Note that the intermediate steps of this workflow are performed separately for each barcode in the read files. Only the first and the optional last step are performed at the whole-genome level. We iterate over the read file and initially collect all reads belonging to one barcode, barcode by barcode. During the iteration we write out the read index.

We calculate all minimizers of the reads labeled with the same barcode and look them up in the k-mer reference index. If a minimizer m appears a sufficiently small number of times (default up to 20 times) in the reference genome, its reference positions $P_m = \{p_m\}$ are extracted and added to a list M. We refer to these positions as the minimizer hits. In addition, we retrieve the abundance A_m of this minimizer in the reference genome and the number of bases B_m for which the minimizer is active in the read sequence. Once the list of minimizer hits M is created for one barcode, we sort it by position on the reference genome.

Next, we cluster the hits in list M using two criteria and a cluster score. The maximum range criterion requires hits of the same cluster $C \subseteq M$ to originate from the same reference contig and limits the range a cluster spans on the contig by an upper threshold (default 300,000). The maximum gap size criterion limits the distance between two adjacent hits in a cluster to a maximum size (default 20,000 chosen empirically, see Supplementary Figure 5). From all minimizer hits within a cluster, $p_m \in C$, we accumulate our cluster score as

$$\text{ClusterScore} = \sum_{p_m \in C} \frac{B_m}{f(A_m)} \tag{2}$$

The active bases B_m of each minimizer increase the contributed score linearly. Rarer k-mer hits contribute more towards the score, scaling inverted logarithmic as a function of the abundance A_m:

$$f(A_m) = \begin{cases} \log(A_m) & \text{if } A_m > 1 \\ 0.2 & \text{if } A_m = 1 \end{cases} \tag{3}$$

We identify initial clusters that conform with the two criteria and maximize the cluster score by iterating over the sorted list of minimizer hits. Hits are added to a cluster as long as both criteria are met. If the maximum range criterion is hurt, multiple cluster configurations spanning the same genomic region may be possible. In this case, we choose the cluster with the highest cluster score. If the maximum gap size criterion is hurt, the current cluster is saved and a new cluster is initialized with the new minimizer hit. We note that this clustering procedure is a heuristic and does not optimize a global score. Example results of this initial clustering procedure are illustrated in Figure 5A.

After the initial clustering, every cluster is submitted to a trimming post processing step (Figure 5B). The aim of the cluster trimming is to remove a false expansion of mappings which are caused by spurious random minimizer hits. For this purpose, two sliding windows are applied, running inwards from the start and the end of the initial cluster, respectively. These sliding windows check if a cluster score per base threshold (default 100) is met for the edges of the cluster and, if necessary, trim the clustered minimizer hit set until this condition is met at both ends. The final mapping score for the cluster is calculated by dividing the cluster score by the length of the genomic region L the trimmed cluster is spanning:

$$\text{MappingScore} = \frac{100}{L} \cdot \sum_{p_m \in C} \frac{B_m}{f(A_m)} \tag{4}$$

The division by L is done to prevent a bias towards larger molecules reaching higher mapping scores. We multiply the score by 100 to obtain a more tangible range of scores.

The reference contig, start and end position of the cluster, the barcode and the mapping score are stored in a BED formatted output file. After sorting the BED file, extracting all barcodes mapped to a user-defined region is straightforward, e.g. using BEDtools [33].

The mapping score can be used in downstream analyses to evaluate the reliability of barcode mappings. Generally speaking, mappings with a higher score are more likely to be correct. To allow the user to choose a valid score threshold, bcmap creates a score histogram similar to the one shown in Figure 3A. The distribution is automatically traced and output by bcmap and can be visualised using a provided python script. As this distribution is sample-, parameter- and reference genome-specific, no score threshold is predefined in bcmap but it can be set by the user.

To provide an additional filtering step for the barcode mappings, bcmap offers an option for a coverage analysis. Based on the coverage analysis, it can filter the barcode mappings using the mapping score. In this optional step, bcmap first determines an upper bound for the minimum score threshold from the score histogram.

Then, it calculates a local score specific barcode mapping coverage for fixed-size reference windows (see Supplementary Figure 2). Wherever the local barcode mapping coverage drops significantly below the expected uniform distribution, it adjusts the mapping score threshold to match the expected distribution. This process allows for correctly distinguishing false positive mappings from low scoring mappings caused by genomic events like segmental duplications.

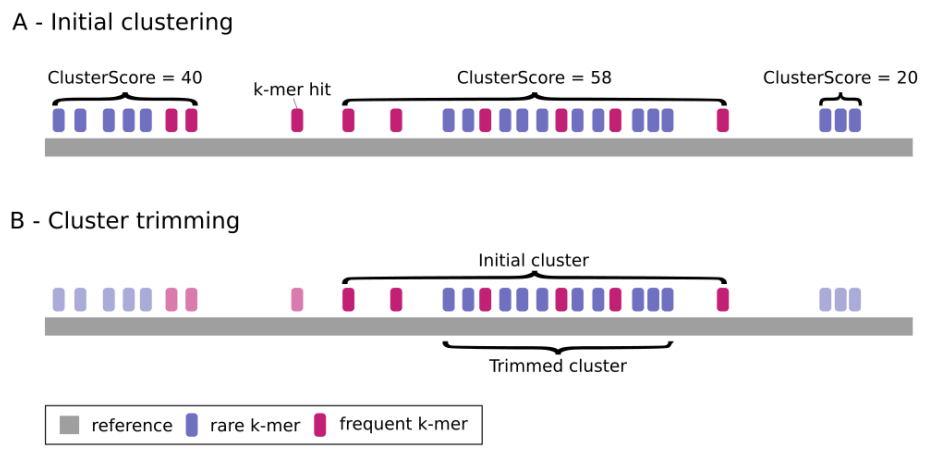

Figure 5 Processing k-mer hits into barcode mappings. The Clustering of k-mer hits A) uses a sliding window, the clusterScore, a maximum cluster size and a maximum gap threshold. Clusters are extended by the next k-mer hit as long as no rules are violated. In this example three clusters are formed. The lone k-mer hit between cluster one and two does not belong to cluster one as this would hurt the maximum gap size criterion. It does not belong to cluster two either because otherwise the maximum cluster size would be exceeded and the shown configuration yields a higher clusterScore than the alternative would. The window trimming B) uses a score-per-base threshold to trim cluster edges. Cluster two is trimmed to the displayed size because the original flanking regions did not meet the score-per-base threshold and were deemed to be incorrect.

Bcmap get

The bcmap get command provides a way to quickly access all reads sharing a specific barcode. For this purpose we use the read index that is constructed during the initial iteration over the read file in the bcmap map comand. The index is queried using a binary search and provides the FASTQ file positions of the reads of interest. Bcmap get then retrieves the reads from the input FASTQ files and returns FASTQ files containing all reads labeled with the user-specified barcodes.

Availability of data and materials

The source code of bcmap is available from GitHub, `https://github.com/kehrlab/bcmap/`. The scripts used for testing and benchmarking of bcmap are available from GitHub, `https://github.com/kehrlab/bcmapScripts`. The linked-read data of sample NA12878 that was used for testing is available from AWS Amazon, `http://s3-us-west-2.amazonaws.com/10x.files/samples/genome/2.1.4/NA12878_WGS_v2/NA12878_WGS_v2_fastqs.tar`. The human reference genome GRCh38 which was used for testing of the software is available from UCSC, `https://hgdownload.soe.ucsc.edu/goldenPath/hg38/bigZips/hg38.fa.gz`.

Competing interests

The authors declare that they have no competing interests.

Funding

This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – project number 400728090 (Research Unit FOR 2841).

Authors' contributions

RL and BK designed the study. RL implemented the method and performed the analyses. BK conceived the study. TK performed the local sequence assembly experiment. BK and RL wrote the manuscript. All authors read and approved the final manuscript.

Acknowledgements

We thank Brian Caffrey for his debugging assistance during the development phase. Computation has been performed on the HPC for Research cluster of the Berlin Institute of Health.

Author details

[1]Genome Informatics, Berlin Institute of Health at Charité–Universitätsmedizin Berlin, 10117 Berlin, Germany. [2]Algorithmic Bioinformatics, Leibniz Institute for Immunotherapy (LIT), 93053 Regensburg, Germany. [3]University of Regensburg, 93053 Regensburg, Germany.

References

1. Jang, S.S., Lim, B.C., Yoo, S.-K., Shin, J.-Y., Kim, K.-J., Seo, J.-S., Kim, J.-I., Chae, J.H.: Targeted linked-read sequencing for direct haplotype phasing of maternal DMD alleles: a practical and reliable method for noninvasive prenatal diagnosis. Sci Rep **8**(1), 8678 (2018)
2. Viswanathan, S.R., Ha, G., Hoff, A.M., Wala, J.A., Carrot-Zhang, J., Whelan, C.W., Haradhvala, N.J., Freeman, S.S., Reed, S.C., Rhoades, J., Polak, P., Cipicchio, M., Wankowicz, S.A., Wong, A., Kamath, T., Zhang, Z., Gydush, G.J., Rotem, D., Love, J.C., Getz, G., Gabriel, S., Zhang, C.-Z., Dehm, S.M., Nelson, P.S., Van Allen, E.M., Choudhury, A.D., Adalsteinsson, V.A., Beroukhim, R., Taplin, M.-E., Meyerson, M.: Structural alterations driving castration-resistant prostate cancer revealed by linked-read genome sequencing. Cell **174**(2), 433–447 (2018)
3. Greer, S.U., Nadauld, L.D., Lau, B.T., Chen, J., Wood-Bouwens, C., Ford, J.M., Kuo, C.J., Ji, H.P.: Linked read sequencing resolves complex genomic rearrangements in gastric cancer metastases. Genome Med **9**(1), 57 (2017)
4. Akbari, P., Gilani, A., Sosina, O., Kosmicki, J.A., Khrimian, L., Fang, Y.-Y., Persaud, T., Garcia, V., Sun, D., Li, A., Mbatchou, J., Locke, A.E., Benner, C., Verweij, N., Lin, N., Hossain, S., Agostinucci, K., Pascale, J.V., Dirice, E., Dunn, M., Regeneron Genetics Center, DiscovEHR Collaboration, Kraus, W.E., Shah, S.H., Chen, Y.-D.I., Rotter, J.I., Rader, D.J., Melander, O., Still, C.D., Mirshahi, T., Carey, D.J., Berumen-Campos, J., Kuri-Morales, P., Alegre-Díaz, J., Torres, J.M., Emberson, J.R., Collins, R., Balasubramanian, S., Hawes, A., Jones, M., Zambrowicz, B., Murphy, A.J., Paulding, C., Coppola, G., Overton, J.D., Reid, J.G., Shuldiner, A.R., Cantor, M., Kang, H.M., Abecasis, G.R., Karalis, K., Economides, A.N., Marchini, J., Yancopoulos, G.D., Sleeman, M.W., Altarejos, J., Della Gatta, G., Tapia-Conyer, R., Schwartzman, M.L., Baras, A., Ferreira, M.A.R., Lotta, L.A.: Sequencing of 640,000 exomes identifies GPR75 variants associated with protection from obesity. Science **373**(6550), 8683 (2021)
5. Ardui, S., Ameur, A., Vermeesch, J.R., Hestand, M.S.: Single molecule real-time (SMRT) sequencing comes of age: applications and utilities for medical diagnostics. Nucleic Acids Research **46**(5), 2159–2168 (2018)
6. Wenger, A.M., Peluso, P., Rowell, W.J., Chang, P.-C., Hall, R.J., Concepcion, G.T., Ebler, J., Fungtammasan, A., Kolesnikov, A., Olson, N.D., Töpfer, A., Alonge, M., Mahmoud, M., Qian, Y., Chin, C.-S., Phillippy, A.M., Schatz, M.C., Myers, G., DePristo, M.A., Ruan, J., Marschall, T., Sedlazeck, F.J., Zook, J.M., Li, H., Koren, S., Carroll, A., Rank, D.R., Hunkapiller, M.W.: Accurate circular consensus long-read sequencing improves variant detection and assembly of a human genome. Nat Biotechnol **37**(10), 1155–1162 (2019)
7. Zheng, G.X.Y., Lau, B.T., Schnall-Levin, M., Jarosz, M., Bell, J.M., Hindson, C.M., Kyriazopoulou-Panagiotopoulou, S., Masquelier, D.A., Merrill, L., Terry, J.M., Mudivarti, P.A., Wyatt, P.W., Bharadwaj, R., Makarewicz, A.J., Li, Y., Belgrader, P., Price, A.D., Lowe, A.J., Marks, P., Vurens, G.M., Hardenbol, P., Montesclaros, L., Luo, M., Greenfield, L., Wong, A., Birch, D.E., Short, S.W., Bjornson, K.P., Patel, P., Hopmans, E.S., Wood, C., Kaur, S., Lockwood, G.K., Stafford, D., Delaney, J.P., Wu, I., Ordonez, H.S., Grimes, S.M., Greer, S., Lee, J.Y., Belhocine, K., Giorda, K.M., Heaton, W.H., McDermott, G.P., Bent, Z.W., Meschi, F., Kondov, N.O., Wilson, R., Bernate, J.A., Gauby, S., Kindwall, A., Bermejo, C., Fehr, A.N., Chan, A., Saxonov, S., Ness, K.D., Hindson, B.J., Ji, H.P.: Haplotyping germline and cancer genomes with high-throughput linked-read sequencing. Nat Biotechnol **34**(3), 303–311 (2016)
8. Wang, O., Chin, R., Cheng, X., Wu, M.K.Y., Mao, Q., Tang, J., Sun, Y., Anderson, E., Lam, H.K., Chen, D., Zhou, Y., Wang, L., Fan, F., Zou, Y., Xie, Y., Zhang, R.Y., Drmanac, S., Nguyen, D., Xu, C., Villarosa, C., Gablenz, S., Barua, N., Nguyen, S., Tian, W., Liu, J.S., Wang, J., Liu, X., Qi, X., Chen, A., Wang, H., Dong, Y., Zhang, W., Alexeev, A., Yang, H., Wang, J., Kristiansen, K., Xu, X., Drmanac, R., Peters, B.A.: Efficient and unique cobarcoding of second-generation sequencing reads from long DNA molecules enabling cost-effective and accurate sequencing, haplotyping, and de novo assembly. Genome Res. **29**(5), 798–808 (2019)
9. Chen, Z., Pham, L., Wu, T.-C., Mo, G., Xia, Y., Chang, P.L., Porter, D., Phan, T., Che, H., Tran, H., Bansal, V., Shaffer, J., Belda-Ferre, P., Humphrey, G., Knight, R., Pevzner, P., Pham, S., Wang, Y., Lei, M.: Ultralow-input single-tube linked-read library method enables short-read second-generation sequencing systems to routinely generate highly accurate and economical long-range sequencing information. Genome Res **30**(6), 898–909 (2020)
10. Shajii, A., Numanagić, I., Berger, B.: Latent variable model for aligning barcoded short-reads improves downstream analyses. Res Comput Mol Biol **10812**, 280–282 (2018)
11. Marks, P., Garcia, S., Barrio, A.M., Belhocine, K., Bernate, J., Bharadwaj, R., Bjornson, K., Catalanotti, C., Delaney, J., Fehr, A., Fiddes, I.T., Galvin, B., Heaton, H., Herschleb, J., Hindson, C., Holt, E., Jabara, C.B., Jett, S., Keivanfar, N., Kyriazopoulou-Panagiotopoulou, S., Lek, M., Lin, B., Lowe, A., Mahamdallie, S., Maheshwari, S., Makarewicz, T., Marshall, J., Meschi, F., O'Keefe, C.J., Ordonez, H., Patel, P., Price, A., Royall, A., Ruark, E., Seal, S., Schnall-Levin, M., Shah, P., Stafford, D., Williams, S., Wu, I., Xu, A.W.,

Rahman, N., MacArthur, D., Church, D.M.: Resolving the full spectrum of human genome variation using Linked-Reads. Genome Res. **29**(4), 635–645 (2019)
12. Wagner, J., Olson, N.D., Harris, L., Khan, Z., Farek, J., Mahmoud, M., Stankovic, A., Kovacevic, V., Yoo, B., Miller, N., Rosenfeld, J.A., Ni, B., Zarate, S., Kirsche, M., Aganezov, S., Schatz, M.C., Narzisi, G., Byrska-Bishop, M., Clarke, W., Evani, U.S., Markello, C., Shafin, K., Zhou, X., Sidow, A., Bansal, V., Ebert, P., Marschall, T., Lansdorp, P., Hanlon, V., Mattsson, C.-A., Barrio, A.M., Fiddes, I.T., Xiao, C., Fungtammasan, A., Chin, C.-S., Wenger, A.M., Rowell, W.J., Sedlazeck, F.J., Carroll, A., Salit, M., Zook, J.M.: Benchmarking challenging small variants with linked and long reads. Cell Genomics **2**(5), 100128 (2022)
13. Zhou, X., Zhang, L., Weng, Z., Dill, D.L., Sidow, A.: Aquila enables reference-assisted diploid personal genome assembly and comprehensive variant detection based on linked reads. Nat Commun **12**(1), 1077 (2021)
14. Elyanow, R., Wu, H.-T., Raphael, B.J.: Identifying structural variants using linked-read sequencing data. Bioinformatics **34**(2), 353–360 (2018)
15. Wong, K.H.Y., Ma, W., Wei, C.-Y., Yeh, E.-C., Lin, W.-J., Wang, E.H.F., Su, J.-P., Hsieh, F.-J., Kao, H.-J., Chen, H.-H., Chow, S.K., Young, E., Chu, C., Poon, A., Yang, C.-F., Lin, D.-S., Hu, Y.-F., Wu, J.-Y., Lee, N.-C., Hwu, W.-L., Boffelli, D., Martin, D., Xiao, M., Kwok, P.-Y.: Towards a reference genome that captures global genetic diversity. Nat Commun **11**(1), 5482 (2020)
16. Weisenfeld, N.I., Kumar, V., Shah, P., Church, D.M., Jaffe, D.B.: Direct determination of diploid genome sequences. Genome Res. **27**(5), 757–767 (2017)
17. Coombe, L., Zhang, J., Vandervalk, B.P., Chu, J., Jackman, S.D., Birol, I., Warren, R.L.: ARKS: chromosome-scale scaffolding of human genome drafts with linked read kmers. BMC Bioinformatics **19**(1), 234 (2018)
18. Jackman, S.D., Coombe, L., Chu, J., Warren, R.L., Vandervalk, B.P., Yeo, S., Xue, Z., Mohamadi, H., Bohlmann, J., Jones, S.J.M., Birol, I.: Tigmint: correcting assembly errors using linked reads from large molecules. BMC Bioinformatics **19**(1), 393 (2018)
19. Bishara, A., Liu, Y., Weng, Z., Kashef-Haghighi, D., Newburger, D.E., West, R., Sidow, A., Batzoglou, S.: Read clouds uncover variation in complex regions of the human genome. Genome Res. **25**(10), 1570–1580 (2015)
20. Miller, N.A., Farrow, E.G., Gibson, M., Willig, L.K., Twist, G., Yoo, B., Marrs, T., Corder, S., Krivohlavek, L., Walter, A., Petrikin, J.E., Saunders, C.J., Thiffault, I., Soden, S.E., Smith, L.D., Dinwiddie, D.L., Herd, S., Cakici, J.A., Catreux, S., Ruehle, M., Kingsmore, S.F.: A 26-hour system of highly sensitive whole genome sequencing for emergency management of genetic diseases. Genome Medicine **7**(1), 100 (2015)
21. Mölder, F., Jablonski, K.P., Letcher, B., Hall, M.B., Tomkins-Tinch, C.H., Sochat, V., Forster, J., Lee, S., Twardziok, S.O., Kanitz, A., Wilm, A., Holtgrewe, M., Rahmann, S., Nahnsen, S., Köster, J.: Sustainable data analysis with Snakemake [version 2; peer review: 2 approved]. F1000Research **10**(33) (2021)
22. Danecek, P., Bonfield, J.K., Liddle, J., Marshall, J., Ohan, V., Pollard, M.O., Whitwham, A., Keane, T., McCarthy, S.A., Davies, R.M., Li, H.: Twelve years of SAMtools and BCFtools. GigaScience **10**(2), 008 (2021)
23. Li, H.: Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM. arXiv:1303.3997 **v2** (2013)
24. Luo, R., Sedlazeck, F.J., Darby, C.A., Kelly, S.M., Schatz, M.C.: LRSim: A Linked-Reads Simulator Generating Insights for Better Genome Partitioning. Computational and Structural Biotechnology Journal **15**, 478–484 (2017)
25. Kehr, B., Helgadottir, A., Melsted, P., Jonsson, H., Helgason, H., Jonasdottir, A., Jonasdottir, A., Sigurdsson, A., Gylfason, A., Halldorsson, G.H., Kristmundsdottir, S., Thorgeirsson, G., Olafsson, I., Holm, H., Thorsteinsdottir, U., Sulem, P., Helgason, A., Gudbjartsson, D.F., Halldorsson, B.V., Stefansson, K.: Diversity in non-repetitive human sequences not found in the reference genome. Nat Genet **49**(4), 588–593 (2017)
26. Zook, J.M., Hansen, N.F., Olson, N.D., Chapman, L., Mullikin, J.C., Xiao, C., Sherry, S., Koren, S., Phillippy, A.M., Boutros, P.C., Sahraeian, S.M.E., Huang, V., Rouette, A., Alexander, N., Mason, C.E., Hajirasouliha, I., Ricketts, C., Lee, J., Tearle, R., Fiddes, I.T., Barrio, A.M., Wala, J., Carroll, A., Ghaffari, N., Rodriguez, O.L., Bashir, A., Jackman, S., Farrell, J.J., Wenger, A.M., Alkan, C., Soylev, A., Schatz, M.C., Garg, S., Church, G., Marschall, T., Chen, K., Fan, X., English, A.C., Rosenfeld, J.A., Zhou, W., Mills, R.E., Sage, J.M., Davis, J.R., Kaiser, M.D., Oliver, J.S., Catalano, A.P., Chaisson, M.J.P., Spies, N., Sedlazeck, F.J., Salit, M.: A robust benchmark for detection of germline large deletions and insertions. Nat Biotechnol **38**(11), 1347–1355 (2020)
27. Chikhi, R., Rizk, G.: Space-efficient and exact de Bruijn graph representation based on a Bloom filter. Algorithms for Molecular Biology **8**(1), 22 (2013)
28. Drezen, E., Rizk, G., Chikhi, R., Deltel, C., Lemaitre, C., Peterlongo, P., Lavenier, D.: GATB: Genome Assembly & Analysis Tool Box. Bioinformatics **30**(20), 2959–2961 (2014)
29. Holley, G., Melsted, P.: Bifrost: highly parallel construction and indexing of colored and compacted de Bruijn graphs. Genome Biology **21**(1), 249 (2020)
30. Kent, W.J.: BLAT—The BLAST-Like Alignment Tool. Genome Res. **12**(4), 656–664 (2002)
31. Weese, D.: Indices and Applications in High-Throughput Sequencing (2013). doi:10.17169/refubium-11863
32. Schleimer, S., Wilkerson, D., Aiken, A.: Winnowing: Local Algorithms for Document Fingerprinting. Proceedings of the ACM SIGMOD International Conference on Management of Data **10** (2003)
33. Quinlan, A.R., Hall, I.M.: BEDTools: a flexible suite of utilities for comparing genomic features. Bioinformatics **26**(6), 841–842 (2010)

Additional Files

Additional file 1 — Supplements.tex
Additional information about and validation of the proposed method.