Kojima *et al. BMC Genomics* [illegible]
DOI 10.1186/s12864-016-3294-x

BMC Genomics

RESEARCH ARTICLE Open Access

STR-realigner: a realignment method for short tandem repeat regions

Kaname Kojima, Yosuke Kawai, Kazuharu Misawa, Takahiro Mimori and Masao Nagasaki*

Abstract

Background: In the estimation of repeat numbers in a short tandem repeat (STR) region from high-throughput sequencing data, two types of strategies are mainly taken: a strategy based on counting repeat patterns included in sequence reads spanning the region and a strategy based on estimating the difference between the actual insert size and the insert size inferred from paired-end reads. The quality of sequence alignment is crucial, especially in the former approaches although usual alignment methods have difficulty in STR regions due to insertions and deletions caused by the variations of repeat numbers.

Results: We proposed a new dynamic programming based realignment method named STR-realigner that considers repeat patterns in STR regions as prior knowledge. By allowing the size change of repeat patterns with low penalty in STR regions, accurate realignment is expected. For the performance evaluation, publicly available STR variant calling tools were applied to three types of aligned reads: synthetically generated sequencing reads aligned with BWA-MEM, those realigned with STR-realigner, those realigned with ReviSTER, and those realigned with GATK IndelRealigner. From the comparison of root mean squared errors between estimated and true STR region size, the results for the dataset realigned with STR-realigner are better than those for other cases. For real data analysis, we used a real sequencing dataset from Illumina HiSeq 2000 for a parent-offspring trio. RepeatSeq and lobSTR were applied to the sequence reads for these individuals aligned with BWA-MEM, those realigned with STR-realigner, ReviSTER, and GATK IndelRealigner. STR-realigner shows the best performance in terms of consistency of the size of estimated STR regions in Mendelian inheritance. Root mean squared error values were also calculated from the comparison of these estimated results with STR region sizes obtained from high coverage PacBio sequencing data, and the results from the realigned sequencing data with STR-realigner showed the least (the best) root mean squared error value.

Conclusions: The effectiveness of the proposed realignment method for STR regions was verified from the comparison with an existing method on both simulation datasets and real whole genome sequencing dataset.

Keywords: High-throughput sequencing, Short tandem repeat, Alignment

Background

From the development of high-throughput sequencing (HTS) technologies, the detailed variant detection is enabled for each individual with whole genome sequencing analysis. For single nucleotide variants (SNVs), various types of variant calling methods have been proposed [1–4] for HTS data, and the accurate SNV detection is archived for more than a thousand of individuals in genome-wide scale [5, 6]. However, unlike SNVs, there still exists difficulty in the accurate detection of structural variations such as genome insertion, genome deletion, short tandem repeat (STR) number polymorphisms, and copy number variations, especially from data with low read coverage [7].

For repeat number polymorphisms, several studies thus far reported associations with various disease phenotypes such as CAG repeats in the Huntingtin gene with Huntington's disease [8] and CAG repeats in the androgen receptor gene with spinal and bulbar muscular atrophy [9]. From HTS data, several approaches such as lobSTR [10], RepeatSeq [11], STRViper [12], and coalescentSTR [13] have been proposed for estimating repeat numbers in STR regions. In lobSTR and RepeatSeq, repeat patterns included in sequence reads spanning the STR regions are considered for the estimation of repeat numbers.

*Correspondence: nagasaki@megabank.tohoku.ac.jp
Tohoku Medical Megabank Organization, Tohoku University, 2-1, Seiryo-machi, Aoba-ku, 980-8573 Sendai, Japan

On the other hand, STRViper and coalescentSTR estimate repeat numbers by considering difference between the actual insert size and the insert size inferred from paired-end reads aligned to the flanking regions of the target repeat. The alignment quality of sequence reads is important for accurate repeat number estimation, especially in the former approaches although usual alignment methods have difficulty in STR regions due to insertions and deletions caused by the frequent change of repeat numbers.

We propose a new dynamic programming based realignment method named STR-realigner where repeat patterns in STR regions are given as prior knowledge, and repeat patterns are used multiple times in the realignment process. Although a similar algorithm is adopted in a tool for detecting STR regions in PacBio reads based on 3-stage modified Smith-Waterman [14], consecutive STR regions can be handled in the proposed algorithm unlike the tool. In addition, clipping fragments, which are an essential feature for the realignment, are also considered in the proposed algorithm. By allowing insertions and deletions of repeat patterns in STR regions with repeatedly use of repeat units, accurate realignment of sequence reads is expected.

In a simulation study with synthetically generated HTS data for artificial diploid genomes sequence based on phased genotypes of a sample in the dataset of 1000 Genomes Project [5], we showed the effectiveness of our model by evaluating root mean squared errors between true and estimated repeat numbers with RepeatSeq or allelotype, an STR calling software in the lobSTR package, from realignment results. For real data analysis, we applied STR-realigner, ReviSTER [15], and GATK IndelRealigner to HTS data from Illumina HiSeq 2000 for a HapMap CEU parent-offspring trio and show the effectiveness of STR-realigner based on consistency in Mendelian inheritance in the estimated repeat numbers in the parent-offspring trio. Root mean squared error values were also calculated from the comparison with the gold standard STR region size obtained from high coverage PacBio sequencing data for one of samples in the parent-offspring trio, and the results from the realigned sequencing data with STR-realigner showed the least (the best) root mean squared error value.

Method

Realignment algorithm considering repeat sequence as prior knowledge

We propose a dynamic programming based algorithm named STR-realigner that realigns query read R to a genome sequence, taking into account the multiple use of repeat patterns for prespecified STR regions. We consider a genome sequence comprised of series of m subsequences $G_1, \ldots, G_m$. Let B_j be a binary variable that takes one if G_j can be used repeatedly and zero otherwise, i.e., subsequence G_j with $B_j = 1$ is for a repeat pattern in one of prespecified STR regions. Figure 1 shows an example of a genome sequence comprised of subsequences $G_1, \ldots, G_6$, where G_2, G_3, and G_5 are repeat patterns of prespecified STR regions and are repeatedly used in the proposed realignment algorithm. In the description of the proposed algorithm, $|R|$ and $|G_j|$ denote the size of R and G_j, and $R[k]$ and $G_j[k]$ denote bases at the kth position of R and G_j, respectively.

Since infinitely long deletions can be considered by using the same subsequence with $B_j = 1$ repeatedly, we limit the size of deletions to less than $|G_j|$ for subsequences with $B_j = 1$. We consider the following six types of states for the alignment of the ith position in query read R to the kth position of subsequence G_j.

1. $s_M(i, j, k)$: a state representing match or mismatch between bases at the ith position of query read R and the kth position of subsequence G_j.
2. $s_I(i, j, k)$: a state representing insertion at the ith position of query read R right after the kth position of subsequence G_j.
3. $s_D(i, j, k)$: a state representing deletion of the kth position of subsequence G_j right after the ith position of query read R.
4. $s_D(i, j, k, l)$: a state representing deletion from the $k - l + 1$ to kth positions of subsequence G_j right after the ith position of query read R. This state is considered only for subsequences with $B_j = 1$ in order to avoid deletions longer than $|G_j|$ by limiting the range of l from 2 to $|G_j| - 1$. For $l = 1$, $s_D(i, j, k)$ is used, and consecutive deletions in the same subsequence are not considered for $s_D(i, j, k)$ with $B_j = 1$. If l is longer than k, the deletion starts from the $|G_j| - l - k + 1$st position on the subsequence

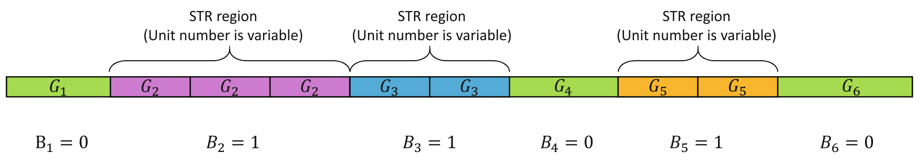

Fig. 1 An example of notations in STR-realigner. Subsequence with $B_j = 1$ can be used repeatedly in the realignment process

and the deletion part rotates from tail to head of the subsequence.

5. $s_L(i)$: a state representing left clipping that ends at the ith position of query read R.
6. $s_R(i)$: a state representing right clipping that starts at the ith position of query read R.

The following penalties are considered in the proposed realignment algorithm.

- $p_{m,j}$: penalty for match of bases between query read R and subsequence G_j. Usually, the penalty is set to a minus value, i.e., the penalty is used for rewarding.
- $p_{mis,j}$: penalty for mismatch of bases between query read R and subsequence G_j.
- $p_{io,j}$ and $p_{ie,j}$: penalties for open and extension of insertion on subsequence G_j, respectively.
- $p_{do,j}$ and $p_{de,j}$: penalties for open and extension of deletion on subsequence G_j, respectively.
- p_c: penalty for clipping.

In the proposed dynamic programming algorithm, penalty and traceback information for state s are stored in functions $P(s)$ and $T(s)$, respectively. In the first step of the dynamic programming, penalty and traceback information of states for the first position in query read R are initialized in the following algorithm.

Algorithm 1: Initialize penalty and traceback information of states for the first position in query read R

```
1. For subsequence ID $j$ from 1 to $m$, perform the following steps:
    (a) For position $k$ from 1 to $|G_j|$ in subsequence $G_j$, perform the following steps:
        (i) Set $P(s_M(1,j,k))$ to $p_{m,j}$ if $G_j[k] = R[1]$ holds, otherwise set $P(s_M(1,j,k))$ to $p_{mis,j}$.
        (ii) Set $P(s_I(1,j,k))$ to infinity.
        (iii) Set $P(s_D(1,j,k))$ to infinity.
        (iv) Set $P(s_D(1,j,k,2))$,...,$P(s_D(1,j,k,|G_j|-1))$ to infinity if $B_j = 1$ holds.
        (v) For each state $s$ considered at steps i to iv, set $T(s)$ to NULL.
2. Set $P(s_L(1))$ and $T(s_L(1))$ to $p_c$ and NULL, respectively.
```

The best penalties for the alignment up to the ith position of query read R for each state is updated by using the best penalties of states for the $i-1$st position of query read R in Algorithm 2, where traceback information is also updated. Algorithm 3 given below updates

Algorithm 2: Update penalties and traceback information of states for positions 2 to $|R|$ in query read R

```
1. For position $i$ from 2 to $|R|$ in query read $R$, perform the following steps:
    (a) For subsequence ID $j$ from 1 to $m$, perform the following steps:
        (i) For position in subsequence $G_j$, $k$ from 1 to $|G_j|$, perform the following steps:
            (A) Use Algorithm 3 to update penalties and traceback information for $s_M(i,j,k)$.
            (B) Use Algorithm 5 to update penalties and traceback information for $s_I(i,j,k)$.
            (C) Use Algorithm 6 to update penalties and traceback information for $s_D(i,j,k)$.
            (D) Use Algorithm 7 to update penalties and traceback information for $s_D(i,j,k,l)$ for $l$ from 2 to $|G_j|-1$ if $B_j=1$ holds.
    (b) Set $P(s_L(i))$ and $T(s_L(i))$ to $p_c$ and NULL, respectively.
    (c) Use Algorithm 8 to update penalty and traceback information for $s_R(i)$.
```

penalty and traceback information for states representing match or mismatch. Algorithm 4 given below is used for obtaining states that are in preceding subsequences and can be traced from $s_M(i,j,1)$. Algorithm 5 given below updates penalty and traceback information for states representing insertion. Algorithm 6 given below updates penalty and traceback information for states representing deletion.

For subsequence G_j with $B_j = 1$, consecutive deletions in the same subsequence are handled with $s_D(i,j,k,l)$, and hence $s_D(i-1,j,k-1)$ is not considered at step 6 of Algorithm 6 for traceback. Procedures for updating penalty and traceback information for states representing consecutive deletions for subsequence G_j with $B_j = 1$ is given as Algorithm 7. Algorithm 8 given below updates penalty and traceback information for $s_R(i)$. Finally, an algorithm for traceback is given as Algorithm 9. By following states from head to tail in Q obtained with the above algorithm, the realignment result with the best penalty is obtained.

Algorithm 3: Update penalty and traceback information for $s_M(i,j,k)$

```
1. Let $\mathcal{S}$ be an empty set of states.
2. Add $s_L(i-1)$ to $\mathcal{S}$.
3. Add $s_M(i-1,j,|G_j|)$ to $\mathcal{S}$ if $k=1$ and $B_i=1$ hold.
4. Add $s_M(i-1,j,k-1)$ to $\mathcal{S}$ if $k>1$ holds.
5. Add $s_I(i-1,j,|G_j|)$ to $\mathcal{S}$ if $k=1$ and $B_i=1$ hold.
6. Add $s_I(i-1,j,k-1)$ to $\mathcal{S}$ if $k>1$ holds.
7. Add $s_D(i-1,j,k-1)$ to $\mathcal{S}$ if $k>1$ holds.
8. Add $s_D(i-1,j,k-1,2),\ldots,s_D(i-1,j,k-1,|G_j|-1)$
   to $\mathcal{S}$ if $B_j=1$ and $k>1$ hold.
9. Add $s_D(i-1,j,|G_j|)$ and
   $s_D(i-1,j,|G_j|,2),\ldots,s_D(i-1,j,|G_j|,|G_j|-1)$ to $\mathcal{S}$ if
   $k=1$ and $B_j=1$ hold.
10. Add states obtained with Algorithm 4 to $\mathcal{S}$ if $k=1$
    holds.
11. Set $T(s_M(i,j,k))$ to $\arg\max_{s\in\mathcal{S}} P(s)$.
12. Set $P(s_M(i,j,k))$ to $P(T(s_M(i,j,k)))+p_{m,j}$ if
    $G_j[k]=R[i]$ holds, otherwise set $P(s_M(i,j,k))$ to
    $P(T(s_M(i,j,k)))+p_{mis,j}$.
```

Algorithm 4: Obtain a set of candidate preceding states in preceding subsequences for subsequence G_j

```
1. Let $\mathcal{S}$ be an empty set of states.
2. Set $j'$ to $j-1$.
3. Go to step 10 if $j'<1$ holds.
4. Add $s_M(i-1,j',|G_{j'}|)$ to $\mathcal{S}$.
5. Add $s_I(i-1,j',|G_{j'}|)$ to $\mathcal{S}$.
6. Add $s_D(i-1,j',|G_{j'}|)$ to $\mathcal{S}$.
7. Add $s_D(i-1,j',|G_{j'}|,2),\ldots,s_D(i-1,j',|G_{j'}|,|G_{j'}|-1)$
   to $\mathcal{S}$ if $B_{j'}=1$ holds,
8. Decrement $j'$.
9. Go back to step 3 if $B_{j'}=1$ holds.
10. Output $\mathcal{S}$.
```

Algorithm 5: Update penalty and traceback information of $s_I(i,j,k)$

```
1. Let $\mathcal{S}$ be an empty set of states.
2. Add $s_M(i-1,j,k)$ to $\mathcal{S}$.
3. Add $s_I(i-1,j,k)$ to $\mathcal{S}$.
4. Add $s_D(i-1,j,k)$ to $\mathcal{S}$.
5. Add $s_D(i-1,j,k,2),\ldots,s_D(i-1,j,k,|G_j|-1)$ to $\mathcal{S}$ if
   $B_j=1$ holds.
6. Set $P(s_I(i,j,k))$ to infinity and perform the following
   steps for each $s\in\mathcal{S}$:
     (a) Set penalty $p$ to $P(s)+p_{ie,j}$ if $s$ is a state
         representing insertion, otherwise set $p$ to
         $P(s)+p_{io,j}$.
     (b) Set $P(s_I(i,j,k))$ and $T(s_I(i,j,k))$ to $p$ and $s$,
         respectively if $P(s_I(i,j,k))>p$ holds.
```

Algorithm 6: Update penalty and traceback information for $s_D(i,j,k)$

```
1. Let $\mathcal{S}$ be an empty set for states.
2. Add $s_M(i-1,j,|G_j|)$ to $\mathcal{S}$ if $k=1$ and $B_j=1$ hold.
3. Add $s_M(i-1,j,k-1)$ to $\mathcal{S}$ if $k>1$ holds.
4. Add $s_I(i-1,j,|G_j|)$ to $\mathcal{S}$ if $k=1$ and $B_j=1$ hold.
5. Add $s_I(i-1,j,k-1)$ to $\mathcal{S}$ if $k>1$ holds.
6. Add $s_D(i-1,j,k-1)$ to $\mathcal{S}$ if $k>1$ and $B_j=0$ hold.
7. Add states obtained with Algorithm 4 to $\mathcal{S}$ if $k=1$
   holds.
8. Set $P(s_D(i,j,k))$ to infinity and perform the following
   steps for each $s\in\mathcal{S}$:
     (a) Set penalty $p$ to $P(s)+p_{de,j}$ if $s$ is a state
         representing deletion, otherwise set $p$ to
         $P(s)+p_{do,j}$.
     (b) Set $P(s_D(i,j,k))$ and $T(s_D(i,j,k))$ to $p$ and $s$,
         respectively if $P(s_D(i,j,k))>p$ holds.
```

Algorithm 7: Update penalty and traceback information for $s_D(i,j,k,l)$ with $l\in[2,|G_j|-1]$

```
1. Set $P(s_D(i,j,k,l))$ to $P(s_D(i,j,k,l-1))+p_{de,j}$ if $l>2$
   holds, otherwise set $P(s_D(i,j,k,l))$ to
   $P(s_D(i,j,k))+p_{de,j}$.
2. Set $T(s_D(i,j,k,l))$ to $s_D(i,j,k,l-1)$ if $l>2$ holds,
   otherwise set $P(s_D(i,j,k,l))$ to $s_D(i,j,k)$.
```

Algorithm 8: Update penalty and traceback information for $s_R(i)$

```
1. Let $\mathcal{S}$ be an empty set for states.
2. For subsequence ID $j$ from 1 to $m$, perform the
   following steps:
     (a) For position $k$ from 1 to $|G_j|$ in subsequence $G_j$,
         perform the following steps:
           (i) Add $s_M(i-1,j,k)$ to $\mathcal{S}$.
3. Get $\hat{s}=\arg\max_{s\in\mathcal{S}} P(s)$.
4. Set $T(s_R(i))$ and $P(s_R(i))$ to $\hat{s}$ and $P(\hat{s})+p_c$, respectively.
```

Algorithm 9: Trace back states for the realignment result with the best penalty

```
1. Let $\mathcal{S}$ be a set of states representing match or
   mismatch for the $|R|$th position in query read $R$.
2. Add $s_R(i)$ to $\mathcal{S}$ for all $i\in[2,|R|]$.
3. Get $\hat{s}=\arg\max_{s\in\mathcal{S}} P(s)$.
4. Let $\mathcal{Q}$ be an empty queue and add $\hat{s}$ to $\mathcal{Q}$.
5. Set $\hat{s}$ to $T(\hat{s})$
6. If $\hat{s}$ is not NULL, add $\hat{s}$ to $\mathcal{Q}$ as its head element and go
   back to step 5.
```

Figure 2 summarize a relationship of the above nine algorithms considered in STR-realigner as a flowchart. After initialization of penalty and traceback information for first query position with Algorithm 1, penalty and traceback information are updated for other query positions with Algorihtm 2 in a dynamic programming manner. Then, a realignment with the best penalty is obtained from traceback information with Algorithm 9.

Time and space complexities of STR-realigner

Time complexity analysis

For each position i in query read R, updating penalty and traceback information takes $O(1)$ time for $s_M(i,j,k)$, $s_I(i,j,k)$, and $s_D(i,j,k)$ for $k > 1$ and subsequence G_j with $B_j = 0$. For $k > 1$ and subsequence G_j with $B_j = 1$, updating information for $s_M(i,j,k)$ and $s_I(i,j,k)$ requires $O(|G_j|)$ time while updating information for $s_D(i,j,k)$ and $s_D(i,j,k,l)$ requires $O(1)$ time. For $k = 1$, states for tail positions of preceding subsequences are additionally considered until reaching to subsequence G_j with $B_j = 0$ or $j = 1$ as in Algorithm 4. This process additionally requires $O\left(\sum_{x=j'}^{j} |G_x|\right)$ time for $s_M(i,j,1)$, where j' is one or the index for the first subsequence $G_{j'}$ with $B_{j'} = 0$ reached from G_j. However, since the best state and its corresponding penalty before G_{j-1} are already considered for updating information for $s_M(i,j-1,1)$, by using this information, we need to newly consider only states in subsequence G_{j-1}, and hence the additionally required time complexity is reduced to $O(|G_{j-1}|)$. Thus, with the modification of the algorithm according to the above argument, updating information for states $s_M(i,1,1),\ldots,s_M(i,m,1)$ requires $O\left(\sum_j |G_j|\right)$ time in total. Since the same optimization can be applied to updating information for states representing insertion, updating information for states with $k = 1$ requires $O\left(\sum_j |G_j|\right)$ time in total as well. In addition, for $s_L(i)$ and $s_R(i)$, $O(1)$ time and $O\left(\sum_j |G_j|\right)$ time are required, respectively. Thus, updating penalties and traceback information for all the states requires $O\left(\sum_j |G_j| + \sum_{j \in \{j' | B_{j'}=1\}} |G_j|^2\right)$ time for each position in query read R, and hence the time complexity of the proposed algorithm is $O\left(|R| \cdot \left(\sum_j |G_j| + \sum_{j \in \{j' | B_{j'}=1\}} |G_j|^2\right)\right)$ time.

Space complexity analysis

The order of the number of states for each position in query read R is $O\left(\sum_j |G_j|\right)$ for $s_M(i,j,k)$, $s_I(i,j,k)$, and $s_D(i,j,k)$. For $s_D(i,j,k,l)$, the order is $O\left(\sum_{j \in \{j' | B_{j'}=1\}} |G_j|^2\right)$, and for $s_L(i)$ and $s_R(i)$, the order is $O(1)$. Thus, storing values from functions P and T requires $O\left(|R| \cdot \left(\sum_j |G_j| + \sum_{j \in \{j' | B_{j'}=1\}} |G_j|^2\right)\right)$ space. However, $P(s_D(i,j,k,l))$ can be obtained by calculating $P(s_D(i,j,k)) + (l-1) \cdot p_{de,j}$, and $T(s_D(i,j,k,l))$ is given by $s_D(i,j,k,l-1)$ for $l > 2$ and $s_D(i,j,k)$ for $l = 2$. Thus, the order of the space required for functions P and T can be reduced to $O\left(|R| \cdot \left(\sum_j |G_j|\right)\right)$ by calculating functions P and T for $s_D(i,j,k,l)$ with $O(1)$ time when their values are required. The space required for updating for each state is less than the order of the number of states and is negligible, compared to spaces required for P and T. Thus, with the above modification, the proposed algorithm requires $O\left(|R| \cdot \left(\sum_j |G_j|\right)\right)$ space.

Practical implementation

Irregular repeat patterns are often contaminated in the provided STR regions detected by some Bioinformatics tools [16, 17], and those irregular repeat patterns worsen the quality of the alignment of the proposed algorithm due to the difference of the actual sequence and the assumed repeat pattern. In order to address this issue, we extract maximal regions containing repeat patterns consecutively with some pre-specified error rate from the target STR region. The extracted region is used for a new target STR region for STR-realigner.

In order to use the realignment result from the proposed algorithm for resequenced data, parts of the query read aligned to G_j with $B_j = 1$ are again realigned to the corresponding STR region of the reference genome. However, the quality of the alignment is also worsened due to irregular patterns in the STR region. Thus, we consider a subsequence for a repeat pattern right after the target STR region and set lower deletion penalty to the target STR region. For penalty, the following setting were used in our study: $p_{m,i} = -1$, $p_{mis,i} = 4$, $p_{io,i} = 6$, $p_{ie,i} = 1$, $p_{do,i} = 6$, $p_{de,i} = 1$, and $p_c = 5$. These parameter values are the same as the default values in BWA-MEM. For subsequences corresponding to target STR regions for lower deletion penalty, $p_{do,i}$ is set to 4.

In Illumina reads, bases at positions after homopolymer regions are highly erroneous because the same phasing is accumulated in synthesis during the Illumina sequencing process in homopolymer regions. Figure 3 shows an example of erroneous bases around a homopolymer region where a lot of clippings occur around a long homopolymer comprised of A bases in GRCh37 due to sequencing errors. Since sequence reads with such highly erroneous bases worsen the quality of realignment with STR-realigner, we additionally implemented an option that skips the realignment with STR-realigner for homopolymer regions with some specified size such as 15.

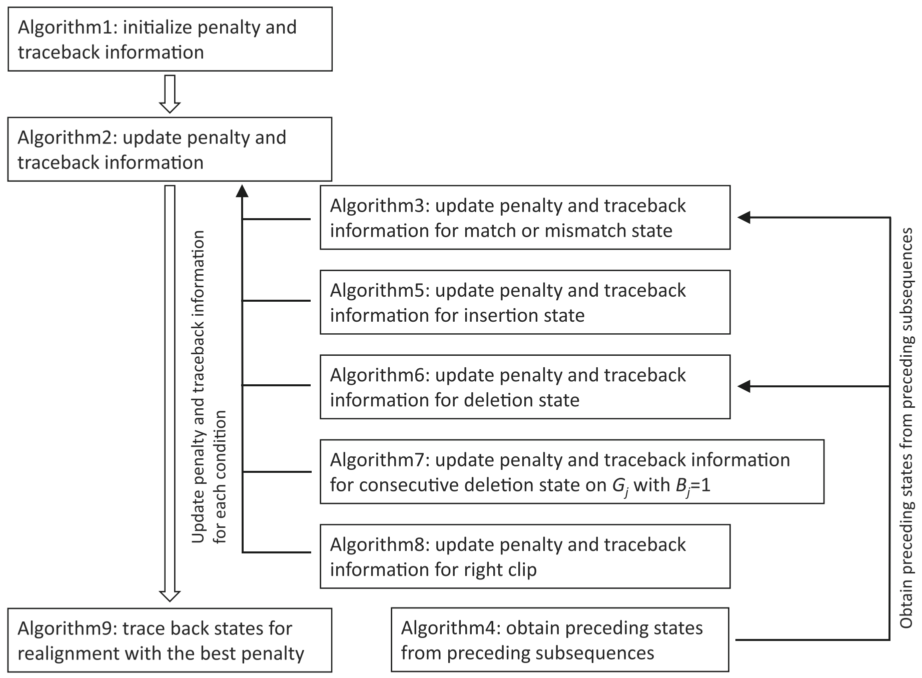
Algorithm1: initialize penalty and traceback information
Algorithm2: update penalty and traceback information
Update penalty and traceback information for each condition
Algorithm3: update penalty and traceback information for match or mismatch state
Algorithm5: update penalty and traceback information for insertion state
Algorithm6: update penalty and traceback information for deletion state
Algorithm7: update penalty and traceback information for consecutive deletion state on G_j with B_j=1
Algorithm8: update penalty and traceback information for right clip
Obtain preceding states from preceding subsequences
Algorithm9: trace back states for realignment with the best penalty
Algorithm4: obtain preceding states from preceding subsequences

Fig. 2 A flowchart of algorithms considered in STR-realigner. After initialization of penalty and traceback information for first query position with Algorithm 1, penalty and traceback information are updated for other query positions with Algorihtm 2 in a dynamic programming manner. Then, a realignment with the best penalty is obtained from traceback information with Algorithm 9

Each mapping tool has its specific characteristics in the aligned reads. For example, a deletion exists in the start position of an STR region in the reads aligned with some mapping tool while a deletion exists in the end position of the STR region in the alignment result of another mapping tool for the same sequence reads. The performance of variant calling is worsened if such characteristics are mixed in the alignment results. Thus, all the reads aligned to a target STR region are realigned with STR-realigner in the default condition.

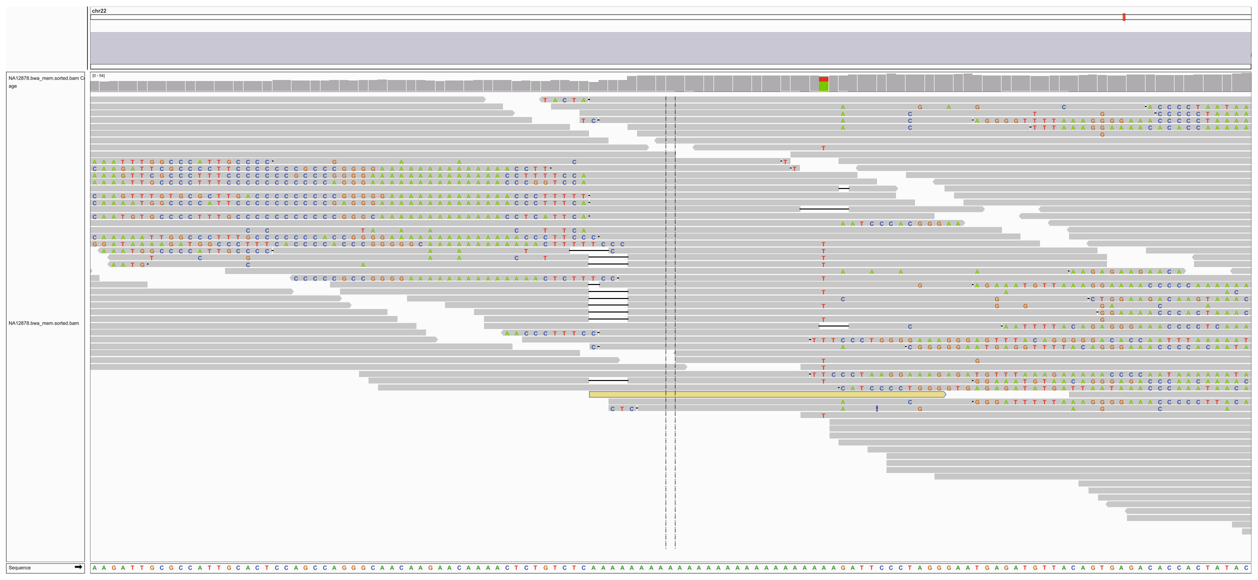
chr22
NA12878.bwa_mem.sorted.bam Coverage
NA12878.bwa_mem.sorted.bam
Sequence

Fig. 3 An alignment result around a homopolymer region. Most of the reads spanning the region contain soft clipping parts due to drastic sequencing errors after the homopolymer region

Results and discussion

Simulation analysis

From a list of STR regions provided in the RepeatSeq software package, we extracted STR regions for evaluation as follows:

- STR regions not in chromosome 22 were filtered out.
- STR regions with size longer than 100 bp were filtered out.

The maximum period, the size of repeat pattern, in the list is six. Since the length of sequence reads considered in the following experiments is 100 or 101 bp and these sequence reads cannot span STR regions > 100 bp for most of the cases, STR regions > 100 bp were filtered out. We then prepared synthetically generated diploid genome sequences of chromosome 22 based on phased genotypes for a CEU individual, NA12286, in the phase3 phased reference panel by the 1000 Genomes Project [18]. In the generation of the above genome sequences, variants located in the extracted repeat regions were ignored. The number of variants in total is 54,897. By randomly sampling repeat numbers, we generated two sets of repeat numbers for the extracted repeat regions and added STR variants to the diploid genome sequences based on the sets of repeat numbers for the evaluation. Note that repeat numbers with which the size of STR region is > 100 bp were avoided in the random sampling process. From the diploid genome sequences, we generated paired-end sequence reads in FastQ format with the read length of 100 bp and the insert size normally distributed with mean of 500 bp and standard deviation of 50 bp. In the generated reads, substitution errors were added with rate of 0.1%. Base quality scores for bases in FastQ format were set to Q30, which corresponds to 0.1% error. The read coverage of the generated data is 40×. A BAM file for the dataset was obtained by mapping the sequence reads to the reference genome (GRCh37) with BWA-MEM (0.7.12-r1039) [19]. We applied our proposed realignment method, STR-realigner, ReviSTER (0.1.7), and GATK IndelRealigner (GATK 3.4-0) to the BAM independently and generated three types of BAM files.

For GATK IndelRealigner, USE_READS was used for --consensusDeterminationModel option. RepeatSeq (v0.8.2) was applied to the original BAM file and the three types of realigned BAM files, and sizes of variants in the target STR regions were obtained. Table 1 shows call rates of results from RepeatSeq using the original BAM file and the three types of realigned BAM files. The call rate indicates the rate of results with STR region size estimated as a non-NA value. For all the STR periods other than period of 1, call rates of results from the BAM file realigned with STR-realigner are higher than those from other BAM files.

Table 2 summarizes the root mean squared errors (RMSE) between estimated and true STR region size for each BAM file for all the STR regions. In the calculation of RMSE, the size in the reference genome was assigned for the region size estimated as NA value. For all the STR periods other than period of 1, the results from the BAM file realigned with STR-realigner show the best RMSE value. The RMSE value from the results based on the BAM file realigned with GATK IndelRealigner is slightly better than that based on the original BAM file. In order to examine the performance excluding the results estimated as NA value, we summarized RMSE for STR regions where results were commonly estimated as a non-NA value on all the four types of BAM files in Table 3. Similarly to the results for all the STR regions, the results from the BAM file realigned with STR-realigner show the best RMSE value for all the STR periods other than period of 1. The RMSE value from the results based on the BAM file realigned with GATK IndelRealigner is slightly better than that based on the original BAM file. We also applied allelotype (4.0.0) [10], an STR calling software in lobSTR package, to the original BAM file and the three types of realigned BAM files. Tables 4, 5 and 6 show call rates, RMSE values averaged on all the regions, and RMSE values averaged on commonly called regions for results from allelotype, respectively. In these tables, the

Table 1 Call rate of STR calling results with RepeatSeq using the original BAM file of 40× and those realigned with STR-realigner, ReviSTER, and GATK IndelRealigner. The best result is underlined

Period	No. of regions	STR-realigner	ReviSTER	IndelRealigner	Original BAM
1	5345	0.878	0.878	0.873	0.872
2	1160	0.799	0.794	0.785	0.784
3	517	0.834	0.807	0.799	0.803
4	1433	0.840	0.766	0.771	0.783
5	668	0.856	0.811	0.819	0.819
6	472	0.881	0.850	0.852	0.856
Total	9595	0.859	0.841	0.838	0.840

Table 2 Root mean squared error (RMSE) between true and estimated repeat numbers with RepeatSeq using the original BAM file of 40× and those realigned with STR-realigner, ReviSTER, and GATK IndelRealigner for all the STR regions. The best result is underlined

Period	No. of regions	STR-realigner	ReviSTER	IndelRealigner	Original BAM
1	5345	3.726	2.700	8.273	8.461
2	1160	2.648	4.648	8.213	8.539
3	517	2.151	4.022	8.242	8.601
4	1433	3.199	5.726	9.523	9.597
5	668	3.885	6.701	10.156	10.325
6	472	2.431	5.976	10.185	10.437
Total	9595	3.421	4.162	8.705	8.900

Table 3 Root mean squared error (RMSE) between true and estimated repeat numbers with RepeatSeq using the original BAM file of 40× and those realigned with STR-realigner, ReviSTER, and GATK IndelRealigner for commonly called STR regions. The best result is underlined

Period	No. of regions	STR-realigner	ReviSTER	IndelRealigner	Original BAM
1	4659	3.858	2.718	8.694	8.891
2	900	2.471	4.815	8.735	9.084
3	410	2.402	3.265	8.548	8.930
4	1084	3.152	4.428	9.688	9.796
5	536	3.600	5.855	10.219	10.422
6	399	2.248	5.593	10.498	10.793
Total	7988	3.484	3.741	9.038	9.253

Table 4 Call rate of STR calling results with allelotype using the original BAM file of 40× and those realigned with STR-realigner, ReviSTER, GATK IndelRealigner, and allelotype with --realign option. The best result is underlined

Period	No. of regions	STR-realigner	ReviSTER	IndelRealigner	--realign option	Original BAM
1	5345	1.000	1.000	0.998	0.998	0.998
2	1160	0.994	0.994	0.984	0.984	0.984
3	517	0.992	0.992	0.988	0.988	0.988
4	1433	0.991	0.992	0.988	0.987	0.987
5	668	0.997	0.997	0.990	0.990	0.990
6	472	0.994	0.994	0.992	0.989	0.989
Total	9595	0.997	0.997	0.993	0.993	0.993

Table 5 Root mean squared error (RMSE) between true and estimated repeat numbers with allelotype using the original BAM file of 40× and those realigned with STR-realigner, ReviSTER, GATK IndelRealigner, and allelotype with --realign option for all the STR regions. The best result is underlined

Period	No. of regions	STR-realigner	ReviSTER	IndelRealigner	--realign option	Original BAM
1	5345	1.104	1.054	4.148	4.181	4.152
2	1160	2.638	2.679	5.454	5.762	5.477
3	517	2.778	2.746	4.818	4.768	4.818
4	1433	2.651	2.631	5.386	5.398	5.406
5	668	2.301	2.114	6.617	6.660	6.617
6	472	3.137	3.178	6.071	5.936	6.094
Total	9595	1.959	1.933	4.861	4.914	4.870

Table 6 Root mean squared error (RMSE) between true and estimated repeat numbers with allelotype using the original BAM file of 40× and those realigned with STR-realigner, ReviSTER, GATK IndelRealigner, and allelotype with --realign option for commonly called STR regions. The best result is underlined

Period	No. of regions	STR-realigner	ReviSTER	IndelRealigner	--realign option	Original BAM
1	5333	1.009	<u>0.955</u>	4.024	4.058	4.028
2	1141	<u>2.475</u>	2.489	5.086	5.420	5.111
3	511	2.472	<u>2.435</u>	4.490	4.435	4.490
4	1414	<u>2.270</u>	2.371	5.157	5.152	5.177
5	661	2.058	<u>1.865</u>	6.263	6.308	6.263
6	467	<u>2.476</u>	2.977	5.888	5.747	5.912
Total	9527	<u>1.729</u>	1.755	4.649	4.702	4.659

results for the original BAM file realigned by allelotype with --realign option are also included. The results for the BAM files realigned with STR-realigner and ReviSTER gave the highest call rate for all the STR periods other than period of 4 and the case considering all the periods. The results for STR-realigner gave the best RMSE value for STR periods of 2, 4, and 6 and the case considering all the periods for commonly called regions although the results for STR-realigner are slightly worse than those for ReviSTER in total for Table 5.

In order to examine the performance on lower coverage data, we downsampled the original BAM file from 40× to 10× and estimated repeat numbers with RepeatSeq and allelotype using the downsampled BAM file and BAM files obtained by applying the realignment methods to the downsampled BAM file. Table 7 shows call rates of results from RepeatSeq for the downsampled BAM files. For all the STR periods other than period of 1, call rates of results from the BAM file realigned with STR-realigner are higher than those from other BAM files. Tables 8 and 9 respectively summarize RMSE values for each BAM file for all the STR regions and the regions where results were commonly estimated as a non-NA value. In the calculation of RMSE, the size in the reference genome was set for the region size estimated as NA value for Table 8. In both Tables 8 and 9, the results from the BAM file realigned with STR-realigner shows the best RMSE values for all the STR periods other than period of 1. The RMSE value from the results based on the BAM file realigned with GATK IndelRealigner is slightly better than that based on the original BAM file.

Tables 10, 11 and 12 show call rates, RMSE values averaged on all the regions, and RMSE values averaged on commonly called regions for results from allelotype, respectively. The results for the BAM file realigned with STR-realigner gave the highest call rate for all the STR periods other than period of 1. In total, STR-realigner gave the best results in both considering all the STR regions and commonly called regions.

The results for the sequencing data of 10× are always worse than those of 40× in all the cases.

Real data analysis

For real human sequencing data, we used 101 bp paired-end sequencing data of a CEU parent-offspring trio NA12878, NA12891, and NA12892 analyzed in the 1000 Genomes Project. NA12891 and NA12892 are parents of NA12878. The data was sequenced on Illumina HiSeq 2000 with the read coverage of 50× and the average insert size of 300 bp. Sequence reads were mapped to the reference genome (GRCh37) with BWA-MEM and stored in BAM format. The data was obtained from the Illumina Platinum Genomes Project through the European Nucleotide Archive under the

Table 7 Call rate of STR calling results with RepeatSeq using the original BAM file of 10× and those realigned with STR-realigner, ReviSTER, and GATK IndelRealigner. The best result is underlined

Period	No. of regions	STR-realigner	ReviSTER	IndelRealigner	Original BAM
1	5345	0.874	<u>0.876</u>	0.854	0.848
2	1160	<u>0.790</u>	0.788	0.760	0.756
3	517	<u>0.830</u>	0.803	0.778	0.774
4	1433	<u>0.831</u>	0.759	0.735	0.736
5	668	<u>0.859</u>	0.793	0.774	0.774
6	472	<u>0.871</u>	0.824	0.807	0.814
Total	9595	<u>0.854</u>	0.836	0.813	0.809

Table 8 Root mean squared error (RMSE) between true and estimated repeat numbers with RepeatSeq using the original BAM file of 10× and those realigned with STR-realigner, ReviSTER, and GATK IndelRealigner for all the STR regions. The best result is underlined

Period	No. of regions	STR-realigner	ReviSTER	IndelRealigner	Original BAM
1	5345	5.261	5.128	8.904	9.126
2	1160	4.779	5.984	8.879	9.162
3	517	3.441	5.409	8.629	8.880
4	1433	4.466	6.724	10.060	10.195
5	668	4.951	7.489	11.058	11.159
6	472	4.854	7.179	10.385	10.652
Total	9595	4.966	5.809	9.308	9.517

Table 9 Root mean squared error (RMSE) between true and estimated repeat numbers with RepeatSeq using the original BAM file of 10× and those realigned with STR-realigner, ReviSTER, and GATK IndelRealigner for commonly called STR regions. The best result is underlined

Period	No. of regions	STR-realigner	ReviSTER	IndelRealigner	Original BAM
1	4505	5.461	5.363	9.047	9.184
2	860	4.878	6.307	9.238	9.436
3	396	3.531	4.862	8.520	8.817
4	1023	4.449	5.954	9.828	9.899
5	497	5.188	6.835	10.504	10.638
6	371	4.622	6.771	10.451	10.773
Total	7652	5.129	5.712	9.323	9.474

Table 10 Call rate of STR calling results with allelotype using the original BAM file of 10× and those realigned with STR-realigner, ReviSTER, GATK IndelRealigner, and allelotype with --realign option. The best result is underlined

Period	No. of regions	STR-realigner	ReviSTER	IndelRealigner	--realign option	Original BAM
1	5345	0.999	1.000	0.992	0.992	0.992
2	1160	0.990	0.988	0.972	0.973	0.972
3	517	0.985	0.983	0.977	0.977	0.977
4	1433	0.983	0.983	0.973	0.973	0.973
5	668	0.994	0.991	0.969	0.969	0.969
6	472	0.987	0.985	0.979	0.979	0.979
Total	9595	0.994	0.994	0.984	0.984	0.984

Table 11 Root mean squared error (RMSE) between true and estimated repeat numbers with allelotype using the original BAM file of 10× and those realigned with STR-realigner, ReviSTER, GATK IndelRealigner, and allelotype with --realign option for all the STR regions. The best result is underlined

Period	No. of regions	STR-realigner	ReviSTER	IndelRealigner	--realign option	Original BAM
1	5345	2.477	2.695	6.017	6.009	6.009
2	1160	3.545	3.740	6.948	6.885	6.899
3	517	3.566	3.728	6.898	6.843	6.843
4	1433	3.754	3.818	7.476	7.417	7.431
5	668	3.910	4.469	8.760	8.762	8.762
6	472	4.319	4.265	8.232	8.217	8.233
Total	9595	3.116	3.309	6.752	6.727	6.732

Table 12 Root mean squared error (RMSE) between true and estimated repeat numbers with allelotype using the original BAM file of 10× and those realigned with STR-realigner, ReviSTER, GATK IndelRealigner, and allelotype with −−realign option for commonly called STR regions. The best result is underlined

Period	No. of regions	STR-realigner	ReviSTER	IndelRealigner	−−realign option	Original BAM
1	5304	2.413	2.656	5.740	5.732	5.732
2	1128	3.336	3.420	6.440	6.371	6.386
3	505	3.057	3.090	6.456	6.396	6.396
4	1394	3.398	3.479	7.045	6.980	6.995
5	647	3.766	4.199	8.015	8.017	8.017
6	462	3.688	3.872	7.907	7.892	7.909
Total	9440	2.906	3.099	6.363	6.336	6.341

study accession PRJEB3381 (http://www.ebi.ac.uk/ena/data/view/ERP001960). STR-realigner, ReviSTER, and GATK IndelRealigner were applied to these BAM files. RepeatSeq was then applied to the original BAM files and these realigned BAM files and sizes of the target STR regions were estimated. In order to examine the performance, we considered the consistency in Mendelian inheritance in called regions, where the estimated region size for NA12878 is a non-NA value. STR regions used for the evaluation are the same as the regions in simulation analysis in Section 1. We counted STR regions where the estimated size for NA12878 consistent with those for her parents, NA12891 and NA12892 in terms of Mendelian inheritance as well as the STR regions with inconsistent results.

In Table 13, the number of regions with consistent sizes in terms of Mendelian inheritance (#CR) and the number of inconsistent estimation results (#IR) are summarized. Note that the larger number is better for consistent regions while the smaller number is better for inconsistent regions. The results for STR period of 1 without skipping homopolymer regions with size > 15 are in parentheses.

For STR periods of 3, 4, and 6, results from BAM files realigned with STR-realigner gave the best results in both consistent and inconsistent regions.

In total, results from BAM files realigned with STR-realigner gave the best results in both consistent and inconsistent regions. Results for GATK IndelRealigner are consistent in more regions than those for the original BAM files although the results for GATK IndelRealigner contains the most inconsistent regions.

In Table 14, the number of regions with consistent sizes in terms of Mendelian inheritance (#CR) and the number of inconsistent sizes (#IR) on results estimated with allelotype are summarized. For STR periods of 2, 3, 5, and 6, results from BAM files realigned with STR-realigner gave the best results in consistent regions. In addition, for STR periods of 2 and 5, results from BAM files realigned with STR-realigner also gave the best results in inconsistent regions. In total, the results for STR-realigner gave the best performance in both consistent and inconsistent regions.

Figure 4 shows an IGV view where STR-realigner effectively works on realigning inserted repeat patterns in an STR region comprised of GGAT repeats located at

Table 13 The numbers of estimated repeat numbers matched and mismatched with parents in terms of Mendelian inheritance

Period	STR-realigner		ReviSTER		IndelRealigner		Original BAM	
	#CR	#IR	#CR	#IR	#CR	#IR	#CR	#IR
1	1305 (1,416)	533 (563)	1319	540	1314	531	1298	531
2	280	82	269	80	242	90	242	87
3	63	5	56	7	56	6	57	5
4	196	28	183	34	169	38	169	33
5	41	15	46	18	44	16	44	15
6	35	9	34	12	33	13	33	13
Total	1920	672	1907	691	1858	694	1843	684

The number of consistent regions (#CR), and the number of inconsistent regions (#IR) based on estimated repeat numbers with RepeatSeq in a parent-offspring trio, NA12878, NA12891 and NA19892, for the original BAM files, those realigned with STR-realigner, ReviSTER, and GATK IndelRealigner are summarized. Values in parentheses for STR-realigner are the result without filtering long homopolymer regions. The best result is underlined

Table 14 The numbers of estimated repeat numbers matched and mismatched with parents in terms of Mendelian inheritance

Period	STR-realigner		ReviSTER		IndelRealigner		- -realign option		Original BAM	
	#CR	#IR	#CR	#IR	#CR	#IR	#CR	#IR	#CR	#IR
1	1772 (1834)	777 (860)	1773	770	1770	773	1621	897	1775	776
2	345	81	328	94	323	90	317	97	328	84
3	70	6	69	4	66	7	65	8	65	7
4	222	24	224	20	219	23	214	28	216	26
5	75	27	75	29	75	28	73	31	73	28
6	73	28	67	28	67	27	68	22	70	26
Total	2557	943	2536	945	2520	948	2358	1083	2527	947

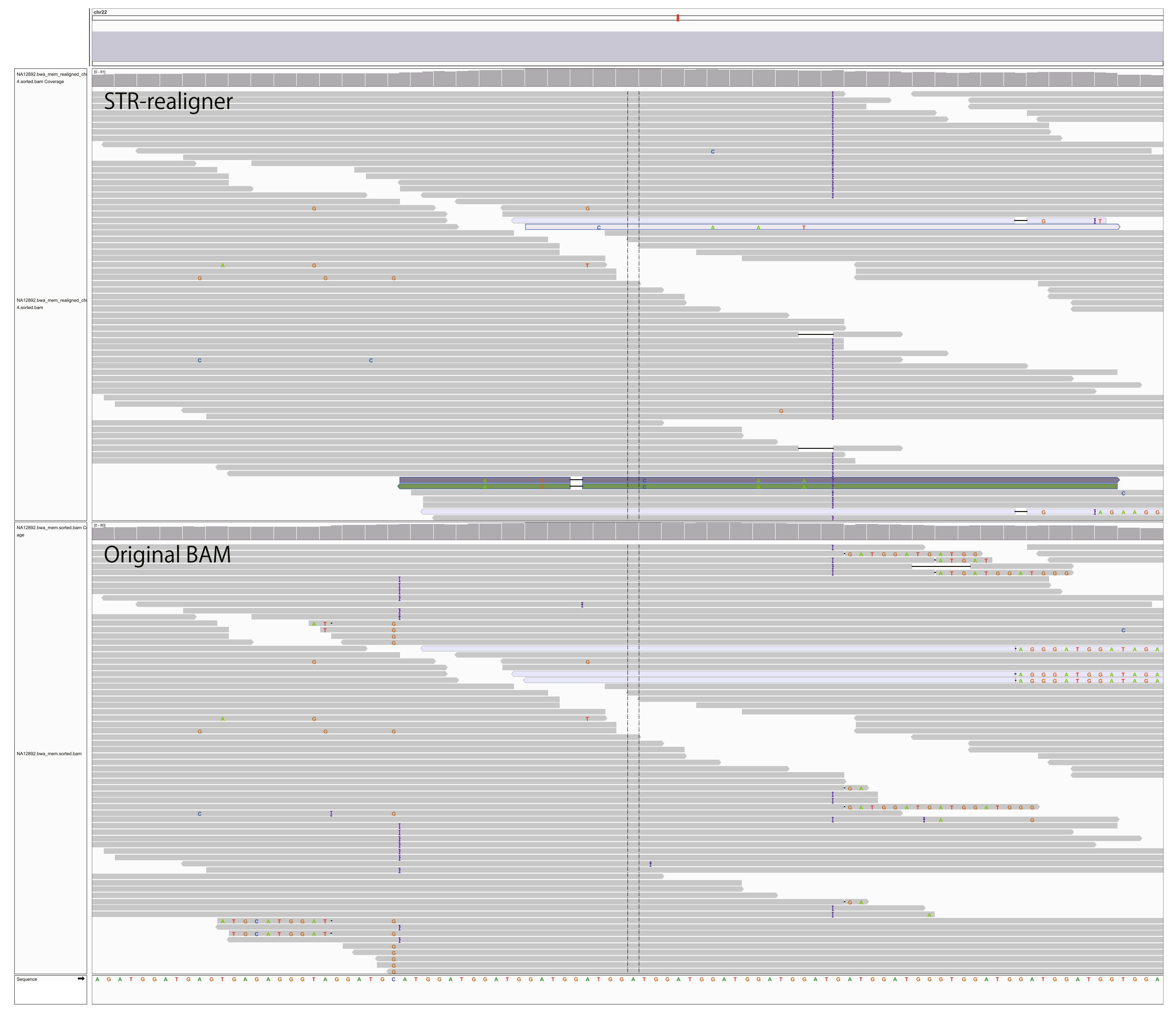

Fig. 4 Comparison of original and realigned BAM files for NA12892 in an STR region located at chr22:28045335-28045407. The top panel is a plot of sequencing data in a BAM file realigned with STR-realigner and the bottom one is a plot of sequencing data in the original BAM file

Table 15 Root mean squared error (RMSE) between the gold standard and estimated STR sizes with RepeatSeq using the original BAM file for NA12878 from HiSeq 2000 and those realigned with STR-realigner, ReviSTER, and GATK IndelRealigner for all the STR regions

Period	No. of regions	STR-realigner	ReviSTER	IndelRealigner	Original BAM
1	5345	2.429	2.397	2.431	2.430
2	1160	3.587	3.860	3.829	3.804
3	517	1.901	1.902	2.000	1.998
4	1433	2.509	2.540	2.709	2.710
5	668	3.134	2.736	3.021	3.039
6	472	2.821	2.823	2.890	2.890
Total	9595	2.656	2.660	2.724	2.721

For the gold standard, STR sizes estimated from high coverage PacBio sequencing data with allelotype are used. The best result is underlined

chr22:28045335-28045407 for a BAM file for NA12892. In the original BAM file, there exist clipping fragments in some reads due to the insertion of GGAT repeats, and the insertion of GGA repeats was missed with RepeatSeq, which caused the inconsistency of the estimated results in Mendelian inheritance. On the other hand, sequence reads with left clipping disappear and insertions are observed in the BAM file realigned with STR-realigner. In the estimated result with RepeatSeq, the insertion was included in the estimated size of the STR region.

We also evaluated root mean squared errors for NA12878 by using STR region sizes obtained from high coverage PacBio sequencing data as the gold standard. The PacBio sequencing data was obtained through BioProject PRJNA253696 with Sequence Read Archive accession numbers SRX627421 and SRX638310 [20] and its read coverage is 46× in total. Error-corrected reads with Falcon (https://github.com/PacificBiosciences/FALCON) in FASTA format were aligned to GRCh37 with BWA-MEM, and STR region size data was then obtained from variant calling results by applying allelotype to the aligned reads.

Table 15 summarizes the RMSE values between estimated STR region size with RepeatSeq and the gold standard for each BAM file for all the STR regions. In the calculation of RMSE, the size in the reference genome was assigned for the region size estimated as NA value. RMSE values for estimated results with allelotype are also summarized in Table 16. In both cases using RepeatSeq and allelotype, the results from sequencing data realigned with STR-realigner showed the best performance in total.

Comparison of computational time

Table 17 shows the computational time of STR-realigner, ReviSTER, and GATK IndelRealigner for simulation and real data analyzed in Sections 1 and 1. For the real data, the computational time for realigning the BAM file for NA12878 was measured. Computation was conducted in Intel Xeon CPU E5-2670 processors with a single thread, and computational time for each case is for single process. STR-realigner is implemented in Java. In both simulation and real data, ReviSTER required the most computational time among these methods, and STR-realigner required more computational time than GATK IndelRealigner. For STR-realigner, by limiting alignment space within some window of the original alignment result, the drastic reduction of the computational time is expected while keeping its realignment quality. In addition, the computational time can be reduced by realigning sequence reads for each STR region in a parallel manner. For memory

Table 16 Root mean squared error (RMSE) between the gold standard and estimated STR sizes with allelotype using the original BAM file for NA12878 from HiSeq 2000 and those realigned with STR-realigner, ReviSTER, GATK IndelRealigner, and allelotype with --realign option for all the STR regions

Period	No. of regions	STR-realigner	ReviSTER	IndelRealigner	--realign option	Original BAM
1	5345	2.298	2.294	2.354	2.296	2.298
2	1160	3.152	3.293	3.265	3.243	3.129
3	517	2.033	2.039	2.034	2.038	2.034
4	1433	2.453	2.582	2.687	2.600	2.454
5	668	3.033	3.006	3.271	3.008	3.031
6	472	2.698	2.739	2.765	2.739	3.090
Total	9595	2.503	2.542	2.607	2.538	2.521

For the gold standard, STR sizes estimated from high coverage PacBio sequencing data with allelotype are used. The best result is underlined

Table 17 Comparison of computational time on a simulation data for an individual with read coverage of 40× and a real dataset for NA12878

Method	Computational time (Simulation data)	Computational time (Real data)
STR-realigner	2,928.90 [s]	1,186.77 [s]
ReviSTER	5,230.72 [s]	3,618.62 [s]
GATK IndelRealigner	357.46 [s]	294.13 [s]

consumption, STR-realigner requires less than 2GB in both simulation and real data.

Conclusion

We proposed a new realignment method for STR regions named STR-realigner that takes sequence reads aligned with other methods and realigns sequence reads by dynamic programing manner with the consideration of the corresponding STR repeat pattern as prior knowledge. For the simulation data analysis, we prepared synthetically generated reads aligned with BWA-MEM, those realigned with STR-realigner, those realigned with ReviSTER, and those realigned with GATK IndelRealigner. In order to evaluate the effectiveness of our proposed method, we applied RepeatSeq and allelotype to these four types of aligned reads, and the results from sequence reads realigned with the proposed method give the best RMSE value among the results from these four types of aligned reads. From the comparison of root mean squared errors between estimated and true STR region size for these four types of aligned reads, the results for the dataset realigned with STR-realigner are better than those for other datasets for most of the cases. For real data analysis, we considered a real sequencing dataset from Illumina HiSeq 2000 for a parent-offspring trio, RepeatSeq was applied to an aligned sequencing dataset with BWA-MEM, that realigned with STR-realigner, that realigned with ReviSTER, and that realigned with GATK IndelRealigner, and the results from the dataset realigned with STR-realigner shows the best performance in terms of consistency of the size of estimated STR regions in Mendelian inheritance. In addition, by using the size for STR regions obtained from high coverage PacBio sequencing data as the gold standard, the results for STR-realigner show the best RMSE values for the case considering all the periods. In both simulation and real data, ReviSTER required the most computational time among the realignment methods considered in this work, and the proposed method required more computational time than GATK IndelRealigner. However, the computational time of STR-realigner can be reduced drastically by parallel computing and limiting the search space for the realignment around the originally aligned results in some specified window size.

Abbreviations

BAM: Binary alignment/map; CEU: Central Europe; GATK: Genome analysis toolkit; GRC: Genome reference consortium; HTS: High-throughput sequencing; RMSE: Root mean squared error; STR: Short tandem repeat

Acknowledgements

This work is partially supported by grants from the Reconstruction Agency, the Ministry of Education, Culture, Sports, Science and Technology (MEXT), the Center of Innovation Program from Japan Science, Technology Agency (JST), and the Japan Agency for Medical Research and Development (AMED). All computational resources were provided by the ToMMo supercomputer system (http://sc.megabank.tohoku.ac.jp/en).

Funding

Publication costs for this work were funded by the Tohoku Medical Megabank Project (Special Account for reconstruction from the Great East Japan Earthquake).

Availability of data and materials

Sequence reads from HiSeq 2000 for NA12878, NA12891, and NA12892 are available at the European Nucleotide Archive under the study accession number PRJEB3381 (http://www.ebi.ac.uk/ena/data/view/ERP001960). Sequence reads from PacBio for NA12878 is available through BioProject PRJNA253696 with Sequence Read Archive accession numbers SRX627421 and SRX638310. A Java implementation of STR-realigner is available at https://github.com/kanamekojima/STR-realigner.

Authors' contributions

KK proposed the statistical model and implemented the program for evaluation. KK, YK, KM, TM, and MN developed fundamental environments for the evaluation using simulation and real data studies. KK, YK, KM, TM, and MN carefully checked equations and other contents in this manuscript. All authors read and approved the final manuscript.

Competing interests

The authors declare that they have no competing interests.

Consent for publication

Not applicable.

Ethics approval and consent to participate

Not applicable.

Received: 13 May 2016 Accepted: 15 November 2016
Published online: 03 December 2016

References

1. DePristo MA, Banks E, Poplin R, Garimella KV, Maguire JR, Hartl C, Philippakis AA, del Angel G, Rivas MA, Hanna M, McKenna A, Fennell TJ, Kernytsky AM, Sivachenko AY, Cibulskis K, Gabriel SB, Altshuler D, Daly MJ. A framework for variation discovery and genotyping using next-generation dna sequencing data. Nat Genet. 2011;43:491–8.
2. Kojima K, Nariai N, Mimori T, Takahashi M, Yamaguchi-Kabata Y, Sato Y, Nagasaki M. A statistical variant calling approach from pedigree information and local haplotyping with phase informative reads. Bioinformatics. 2013;29(22):2835–843.
3. Kojima K, Nariai N, Mimori T, Yamaguchi-Kabata Y, Sato Y, Kawai Y, Nagasaki M. Hapmonster: a statistically unified approach for variant

calling and haplotyping based on phase-informative reads. Lect Notes Comput Sci. 2014;8542:107–18.
4. Li H, Ruan J, Durbin R. Mapping short dna sequencing reads and calling variants using mapping quality scores. Genome Res. 2008;18(11): 1851–1858.
5. 1000 Genomes Project Consortium. An integrated map of genetic variation from 1,092 human genomes. Nature. 2012;491(7422):56–65.
6. Nagasaki JM, Yasuda KF, et al. Rare variant discovery by deep whole-genome sequencing of 1,070 japanese individuals. Nat Commun. 2015;6:8018.
7. Mimori T, Nariai N, Kojima K, Takahashi M, Ono A, Sato Y, Yamaguchi-Kabata Y, Kawai Y, Nagasaki M. iSVP: an integrated structural variant calling pipeline from high-throughput sequencing data. BMC Syst Biol. 2013;7(Suppl 6):(S8).
8. Walker FO. Huntington's disease. Lancet. 2007;369(9557):2185–28.
9. La Spada AR, Wilson EM, Lubahn DB, Harding AE, Fischbeck KH. Androgen receptor gene mutations in x-linked spinal and bulbar muscular atrophy. Nature. 1991;352(6330):77–9.
10. Gymrek M, Golan D, Rosset S, Erlich Y. lobstr: A short tandem repeat profiler for personal genomes. Genome Res. 2012;6:1154–1162.
11. Highnam G, Franck C, Martin A, Stephens C, Puthige A, Mittelman D. Accurate human microsatellite genotypes from high-throughput resequencing data using informed error profiles. Nucleic Acids Res. 2013;41(1).
12. Cao MD, Tasker E, Willadsen K, Imelfort M, Vishwanathan S, Sureshkumar S, Balasubramanian S, Boden M. Inferring short tandem repeat variation from paired-end short reads. Nucleic Acids Res. 2014;42(3).
13. Kojima K, Kawai Y, Nariai N, Mimori T, Hasegawa T, Nagasaki M. Short tandem repeat number estimation from paired-end reads for multiple individuals by considering coalescent tree. BMC Genomics. 2016;17(494).
14. Ummat A, Bashir A. Resolving complex tandem repeats with long reads. Bioinformatics. 2014;30(24):3491–498.
15. Tae H, McMahon KW, Settlage RE. Revister: an automated pipeline to revise misaligned reads to simple tandem repeats. Bioinformatics. 2013;29(14):1734–1741.
16. Benson G. Tandem repeats finder: a program to analyze dna sequences. Nucleic Acids Res. 1999;27(2):573–80.
17. Misawa K. Short tandem repeats in the human, cow, mouse, chicken, and lizard genomes are concentrated in the terminal regions of chromosomes. Gene Reports. 2016;4:280–5.
18. 1000 Genomes Project Consortium, Abecasis GR, Auton A, Brooks LD, DePristo MA, Durbin RM, Handsaker RE, Kang HM, Marth GT, McVean GA. A map of human genome variation from population-scale sequencing. Nature. 2010;467(7422):1061–1073.
19. Li H. Aligning sequence reads, clone sequences and assembly contigs with bwa-mem. arXiv:130.3997. 2013.
20. Pendleton M, Sebra R, Chun Pang AW, Ummat A, Franzen O, Rausch T, Stutz AM, Stedman W, Anantharaman T, Hastie A, Dai H, Fritz MH, Cao H, Cohain A, Deikus G, Durrett RE, Blanchard SC, Altman R, Chin C, Guo Y, Paxinos EE, Korbel JO, Darnell RB, McCombie WR, Kwok P, Mason CE, Schadt EE, A AB. Assembly and diploid architecture of an individual human genome via single-molecule technologies. Nat Methods. 2015;12(8):780–6.