Published online 19 April 2016 *Nucleic Acids Research, 2016, Vol. 44, No. 12* ***e112***
doi: 10.1093/nar/gkw284

An accurate clone-based haplotyping method by overlapping pool sequencing

Cheng Li†, Changchang Cao†, Jing Tu and Xiao Sun*

State Key Laboratory of Bioelectronics, School of Biological Science and Medical Engineering, Southeast University, Nanjing, Jiangsu 210002, China

Received October 13, 2015; Revised March 21, 2016; Accepted April 07, 2016

ABSTRACT

Chromosome-long haplotyping of human genomes is important to identify genetic variants with differing gene expression, in human evolution studies, clinical diagnosis, and other biological and medical fields. Although several methods have realized haplotyping based on sequencing technologies or population statistics, accuracy and cost are factors that prohibit their wide use. Borrowing ideas from group testing theories, we proposed a clone-based haplotyping method by overlapping pool sequencing. The clones from a single individual were pooled combinatorially and then sequenced. According to the distinct pooling pattern for each clone in the overlapping pool sequencing, alleles for the recovered variants could be assigned to their original clones precisely. Subsequently, the clone sequences could be reconstructed by linking these alleles accordingly and assembling them into haplotypes with high accuracy. To verify the utility of our method, we constructed 130 110 clones *in silico* for the individual NA12878 and simulated the pooling and sequencing process. Ultimately, 99.9% of variants on chromosome 1 that were covered by clones from both parental chromosomes were recovered correctly, and 112 haplotype contigs were assembled with an N50 length of 3.4 Mb and no switch errors. A comparison with current clone-based haplotyping methods indicated our method was more accurate.

INTRODUCTION

Human genomes are diploid, with the homologous chromosomes being derived from each parent, respectively (1). The process of resolving the diploid nature, which assigns each allele to different homologous chromosomes, is called haplotyping (2). For many biological and medical studies, it is very valuable to obtain chromosome-long haplotyping information. For example, haplotypes can identify genetic variants associated with altered gene expression (2), and can be used to study human migration, evolution selection and population structure. Most importantly, haplotypes are very useful in clinical diagnosis (3).

At present, several methods have been proposed to resolve haplotypes, but all of them have advantages and disadvantages. Computational methods based on population genotype data can phase common variants cheaply and accurately, but are incapable of dealing with rare or individual-specific variants (4–6). The most direct experimental method for haplotyping is physically separating the chromosomes during cell division (7,8), which requires expensive specialized devices and subtle manipulation. Alternatively, HaploSeq exploits chromosomes' spatial information based on Hi-C techniques to construct haplotype blocks (9), while leaving many variants un-phased between "blocks". Several dilution-based haplotyping methods have been reported, including single-molecule dilution (1,10,11), transposome-based virtual dilution (12) and clone pooling (13–16). In the ideal scenario, the target DNA is diluted into substantial distinct wells to guarantee that there would be no DNA fragments from both parental chromosomes overlapping in each well. However, this scenario requires enough dilution in which the target DNA will be diluted into too many wells, leading to a significant bias for single-molecule amplification before sequencing. The contiguity-preserving transposition (CPT-seq) dilution method solves the paradox by importing an indexed Tn5 transposome into target DNA to create separate virtual genomic partitions (12). However, the broad length distribution of the DNA library makes the subsequent PCR non-uniform, which would decrease the haplotyping coverage.

The clone-based dilution method will produce more accurate haplotyping because long subsections of a haploid could be extracted from the constructed clone libraries (13). Therefore, the clone-based haplotyping approach is still widely used, although substantial cost is required for clone library preparation. The basic principle behind this method is pooling clones into different pools for sequencing and then reconstructing the clones from shorter sequencing

*To whom correspondence should be addressed. Tel: +86 25 83795174; Fax: +86 25 83792349; Email: xsun@seu.edu.cn
† These authors contributed to the paper as first authors.

reads and assembling them into longer haploid fragments. However, in real experiments, clones covering the same sites from both homologous chromosomes could simultaneously appear within a pool. The overlapping parts between clones from the homologous chromosomes will produce heterozygosity, which is not informative for haplotyping (13). Hence, current clone-based haplotyping methods remove either all the overlapping clones (14) or just the overlapping parts (15), which wastes the information carried by these parts and reduces the accuracy of the assembled haplotype contigs. Considering that clones could be picked repeatedly, if clones are mixed combinatorially into different pools and sequenced (a strategy defined as overlapping pool sequencing), it would be possible to construct more accurate haplotypes containing the overlapping parts by correctly assigning variants in the overlapping parts to their original clones, based on the unique pooling pattern for each clone. In addition, because of the error-tolerance of overlapping pool sequencing strategy (17), the reconstructed clone sequences will be more accurate compared with the simple pooling, which will improve the accuracy of the assembled haplotypes.

Hence, we present a method for single individual haplotyping that could resolve haplotypes from overlapping clone parts. This method is based on the overlapping pool sequencing strategy, which originated from Group Test theories (18,19), as we described before (17,20). Overlapping pool sequencing techniques have been used to find rare variant carriers among a large number of samples in recent years. Defining samples carrying the target variant as *positive*, overlapping pool sequencing can find rare *positive* samples from a large number of samples by mixing samples combinatorially to different pools, and then decoding the sequencing results according to the pooling patterns for each sample. The DNA Sudoku design (21) and binary-based design (22) were constructed to screen rare variant carriers; however, the latter failed when more than one sample contained the variant. Shental *et al.* (23) used compressed sensing theories to resolve the group testing matrix, which makes use of the number of reads containing the target variant to improve the decoding efficiency. However, all the above designs lack well-grounded cost models, which lead to expensive non-optimized overlapping pooling design. Cao *et al.* (17) proposed an optimized overlapping pooling design strategy that achieved a better performance. At present, the overlapping pool sequencing techniques have made significant progress such that they can not only find more than one variant carrier, but also tolerate sequencing errors.

In the clone-based haplotyping scenario, considering that haplotyping is performed to ascertain all the alleles on a single chromosome and all the clones have a haploid nature, haplotyping could be realized by determining all the alleles on all the clones and then assembling these clones into haplotype contigs. Thus, the critical step is to assign allelic variants called from the sequencing data to their original clones. For a given variant site, both the reference allele (allele in the reference) and the non-reference allele (allele called by software) need to be assigned respectively. Taking the clones containing a target allele as positive samples, overlapping pool sequencing can be used to assign every allele to clones by combinatorial pooling design and decoding. Successful decoding for overlapping pool sequencing requires that the positive samples are rare. The clones are randomly constructed; therefore, very few clones span each variant site. In other words, for each variant site, the clones carrying a specified allele are rare in the whole clone library, which meets the decoding requirement. When all the alleles are correctly assigned, clones could be reconstructed by linking the alleles. Accordingly, clones that belong to the same chromosome could be assembled to form haplotype contigs by chaining together heterozygous variants.

Considering the rarity of clones carrying a certain allele, techniques from compressed sensing (CS) theory (24) can be used in the clone-based haplotyping scenario to decode which clones contain a certain allele. The CS theory is an information theory tool first used to analyze sparse signals. Following its fast development, it has been applied widely in many research fields, including image processing (25), geophysics (26), astronomy (27) and biological applications (28).

In this paper, we first describe the design and decoding procedure for single individual haplotyping using overlapping pool sequencing. Employing a pooling design algorithm, the clones are pooled combinatorially following a specific pooling pattern and then sent for sequencing. Using the decoding algorithm from the field of compressed sensing theories, each allele can be assigned back to its original clones. Accordingly, clones could be reconstructed by linking these alleles and further assembled into haplotype contigs. Next, we conducted an experiment to phase haplotypes of a HapMap individual, NA12878, *in silico*, to verify the accuracy and efficiency of our method. Finally, a comparison experiment employing current clone-based haplotyping methods on the same individual was conducted, which showed that our method achieved a higher accuracy.

MATERIALS AND METHODS

Overlapping pool sequencing

Overlapping pool sequencing is a newly developed strategy that identifies rare variant carriers (positive samples) from a large number of samples, according to the results of a few sequencing tests. Suppose there are n samples in which s samples contain the target variant ($s \ll n$). The n samples are pooled into t pools ($t < n$) and each pool is sequenced; subsequently, the samples containing the variant could be found according to the pooling pattern and sequencing result. The pooling pattern is based on a pooling matrix M. The M is a $t \times n$ binary matrix $M = \{m_{ij}\}$, in which the rows are indexed by pools $A_1,\ldots,A_t \subset \{1\ldots n\}$, the columns are indexed by samples $S_1,\ldots,S_n$, and $m_{ij} = 1$ if and only if the jth sample is contained in the ith pool (Figure 1A).

In the clone-based haplotyping scenario, clone libraries are constructed by extracting long subsections of a haploid and are then distributed randomly and independently in wells of plates. The most naïve method to obtain clone sequences is sequencing clones one by one and the haplotypes could be assembled accordingly. However, there are so many clones that the sequencing cost would be unaffordable and the sequencing errors are difficult to eliminate. The currently available strategies reduce the sequencing cost by

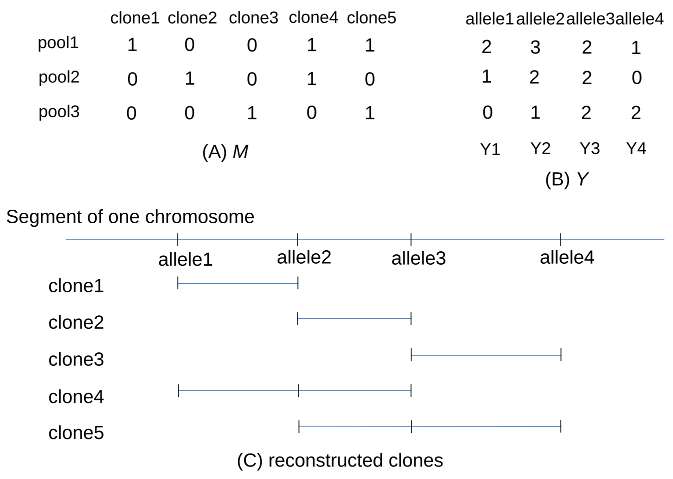

Figure 1. Illustration of alleles assignment. (**A**) Five clones are pooled into three pools, which means pool #1 contains clones 1, 4 and 5; pool #2 contains clones 2 and 4; and pool #3 contains clones 3 and 5. (**B**) The sequencing results. For example, allele 2 is sequenced three, two and one times in pools #1, #2 and #3, respectively. (**C**) According to M and Y, the vector x for every allele could be solved and the five clones could be reconstructed accordingly, as shown in (C). For example, the sequencing result of allele 2 is $(3, 2, 1)^T$, equaling the dot-product of the vector M with the target vector x of $(1, 1, 0, 1, 1)^T$, which means allele 2 is contained in clone 1, clone 2, clone 4 and clone 5, but not in clone 3.

pooling clones, but could not obtain haplotype information from overlapping clone parts within a pool, and also could not correct sequencing errors. Based on the above introduction, overlapping pool sequencing might solve these problems by pooling clones combinatorially following a certain pattern, sequencing all the pools, and then decoding to assigning alleles to their original clones. Clones can then be reconstructed by linking all the assigned alleles and can be further assembled into haplotype contigs. Hence, the key step is to assign the alleles called from sequenced data to their original clones. The clones carrying a certain allele are rare in the whole clone library; therefore, the rare clones carrying one target allele are called as positive samples, and the allele assigning problem is converted to finding rare positive ones from a large number of samples, which can be solved efficiently by employing techniques from the field of compressed sensing.

In general, to construct haplotypes employing overlapping pool sequencing, clones are first pooled combinatorially into different pools and then sequenced. Next, all the alleles are identified by calling variants from the sequencing results and each allele should be independently assigned to its original clones. Finally, clones are reconstructed by linking these alleles and further assembled into haplotype contigs. The impact of sequencing errors can be relieved and haplotypes with high accuracy can be phased accordingly because of the error-tolerance of pooling and the decoding process.

Pooling design

In overlapping pool sequencing, the pooling strategy should be designed initially to guarantee its ability. Before the pooling design, we only know the total clone number (n) and clone coverage (c) for a given genome. According to these known conditions, a pooling matrix M could be constructed by employing a random size-k design (29), where a constant number of clones are randomly chosen and pooled into each pool. Therefore, there are three parameters for random size-k design: the number of pools (t), the data throughput for each pool (dt), and the percent of clones for each pool (k). To achieve a correct decoding rate with the maximum efficiency, the optimal combination of the three parameters is chosen on the basis of simulation experiments. First, some reasonable values of the parameters are selected. Next, the simulation experiments under all combinations of selected values of the three parameters are conducted, and many replicates under every combination are performed. Then, keeping the decoding rate beyond a threshold, the optimal design parameters are selected based on the whole cost of the sequencing experiment (17,20). When the optimal pooling pattern has been identified, the pooling matrix M can be constructed, and the clones should be pooled combinatorially according to M and then sequenced.

Normally the clone number (n) is $>100\,000$. Studies in the field of overlapping pool sequencing (30) have proved that it is more efficient to split a large library into smaller blocks of samples, which is called repeated blocks design. Hence, the clones are first randomly divided into several blocks and then pooled "intra-block". The number of clones that span a variant site follows a Poisson distribution; therefore, we could calculate the block number (b), the clone numbers in each block (cb), and the maximum number (s) of clones that span a variant, i.e. the maximum number of positive samples in group testing. Thus the pooling design for each block should guarantee the ability to find the correct set of clones when the variant is covered by, at most, s clones.

Assigning alleles to clones

After sequencing the pooled clones, the next critical procedure is to call the variants and assign the alleles to the correct clone set. Currently, variants may be obtained from the next generation sequencing data by various methods (31). Subsequently, alleles are assigned to their original clones one by one, including both the reference allele and the non-reference allele for every variant site. Considering that few clones actually carry a certain allele, the allele assignment problem can be solved using compressed sensing (CS). In CS, the unknown sparse vector x, with few elements that are non-zero, can be reconstructed by solving the equation:

$$Mx = Y \tag{1}$$

In the allele assignment, M is the pooling matrix $M = \{m_{ij}\}$, and for each allele, the ith element in Y represents how many times the allele is sequenced in the ith pool, which could be obtained from the sequencing results. M and Y are known; therefore, vector x could be obtained by solving the equation. The elements in x are either 1 or 0, meaning the clone contains the allele or it does not. Figure 1 shows an example. Suppose five clones are pooled into three pools, according to Figure 1A. This means that pool #1 contains clone 1, 4 and 5; pool #2 contains clone 2 and 4; and pool #3 contains clone 3 and 5. The sequencing results are shown in Figure 1B. In this example, allele 2 is sequenced three, two and one times in pools #1, #2 and #3, respectively. According to M and Y, the vector x for every allele could be solved and the five clones could be reconstructed accordingly, as shown in Figure 1C. For example, the sequencing result of allele 2 is $(3, 2, 1)^T$, equaling the dot-product of the vector M with the target vector x of $(1, 1, 0, 1, 1)^T$, which means allele 2 is contained in clone 1, clone 2, clone 4 and clone 5, but not in clone 3.

To assign alleles correctly and efficiently in substantial clones, the l_1-squared nonnegative regularization (*NNREG)* algorithm could be employed to solve equation (1) and estimate x, which achieves sparse recovery using a conventional non-negative least squares algorithm and allows for sparsity promotion, constraining x to be non-negative at the same time (32). *NNREG* recovers x by solving the optimization problem (2) for some large $\lambda > 0$.

$$x* = \arg\min \ \|x\|_{l1}^2 + \lambda^2 \, \|Mx - Y\|_{l2}^2 \quad s.t. \ x \geq 0 \qquad (2)$$

In general, *NNREG* reports a vector $x* \in R^+$. To decide which clones carry the target allele, a post-processing procedure should be performed to find the correct clone set. Similar to (23), we chose a heuristic scheme to deal with x^*. We ranked all non-zero values obtained by *NNREG*, and rounded the largest s non-zero values, setting all other values to zero to get the vector x^{*s}. We then computed a post-probability term $PP(s) = P(O|M, x^{*s})$ standing for the probability that the sequencing result O is observed when x^{*s} indicates the set of clones that carry the target allele. The details for $PP(s)$ are given in Appendix 1. Repeating this for different values of s, we selected the vector x^{*s}, which maximizes the post-probability term, because the probability of observing the sequencing results under the exact set of clones carrying the target allele should be greater than that obtained considering other sets of clones.

Haplotype assembly

Following the assignment of alleles, HapCUT (33) was employed to assemble haplotypes. HapCUT requires two kinds of input data: clone sequences and the set of variants covered by them. Clones could be reconstructed by linking alleles that belong to each clone. Meanwhile, the variant set could be obtained by collecting all the alleles assigned to at least one reconstructed clone.

The variants would be called as heterozygous if both reference alleles and non-reference alleles were observed, and as homozygous if only non-reference alleles were observed. Hence, in the assembly, variants that are homozygous are discarded, as they are not useful for haplotype phasing. Likewise, all sites with more than two alleles are discarded, as all variant sites should be bi-allelic for a diploid genome. The input to HapCUT can be represented as a matrix, X, where each row represents a clone and each column represents a heterozygous variant. Alleles for each heterozygous variant are arbitrarily relabeled as 0 and 1, and an entry in the matrix, $X[i][j]$, is either '0', '1' or '–' depending on the allelic value of position j in clone i.

In the absence of any errors, the rows of fragment matrix X can be partitioned into two disjointed sets, such that every column is homozygous in each set (34). Furthermore, the consensus values can be used to phase the two haplotypes. However, the sequencing process, for instance, will produce errors in fragments, and perfect bi-partitions cannot be achieved. On the basis of computing max-cuts in graphs derived from the matrix X, HapCUT partitions the clones to minimize the error correction, which is also known as the minimum error correction (MEC) objective, which produces significantly more accurate inferred haplotypes.

RESULTS

Experimental design

First, we downloaded the SNP variants for a HapMap sample NA12878 (http://www.stanford.edu/~kuleshov/NA12878.vcf.gz, Supplementary Table S1). Replacing bases on the corresponding site of the reference genome hg19 with these variants, we could artificially produce two haplotype sequences for the individual NA12878. Next, clones were generated by taking sequences randomly from these two haplotypes. The length of the clones followed a Poisson distribution, where the average length was approximately 140 kb (Supplementary Figure S1). The whole clone coverage was set as 6×, similar to Lo's research (13). Ultimately, 130 110 clones were generated.

Considering the large clone number, a repeated blocks design was employed to split these clones into smaller blocks, where an optimized overlapping pool strategy was conducted repeatedly for each block, which reduced the decoding complexity (30). According to the above analysis, we could calculate the maximum number of clones that span a variant in different block designs, i.e., the maximum number of positive samples in group testing (Supplementary Table S2, Supplementary Figure S2). Hence, we finally split the clones into eight blocks, each containing approximately 16 264 clones and more than 99.27% of the variants were covered by, at most, three clones. Accordingly, given a variant site, the overlapping pool sequencing design should guarantee the ability to find the correct set of clones when the site was covered by, at most, three clones.

For each block with 16 264 clones, we applied a random size-k design to pool the clones combinatorially (29), where a constant number of clones was randomly chosen and pooled into each pool. Therefore, there were three parameters for random size-k design: the number of pools (t), the data throughput for each pool (dt), and the percent of total clones that were pooled in each pool (k). As described in the Methods section, the three parameters were chosen on the basis of simulation calculations under all combinations of the parameter values. First, some reasonable values of the parameters were selected. For t, according to our previous design experience of finding three positive samples from over 10 000 samples, 50, 60, 70 and 80 pools were chosen. For k, because each pool containing too many or too few clones would reduce the decoding efficiency, 20, 30, 40 and 50% were chosen. For dt, because we chose 15 000, 20 000, 25 000 and 30 000 as the overall sequencing depth for each pool and average clone length was 140 kb, 2.1, 2.8, 3.5 and 4.2 GB were chosen. The correct decoding rates for different combinations of various k, t and dt were then calculated and 1000 replicates were conducted for each scenario. The results are shown in Figure 2. It was obvious

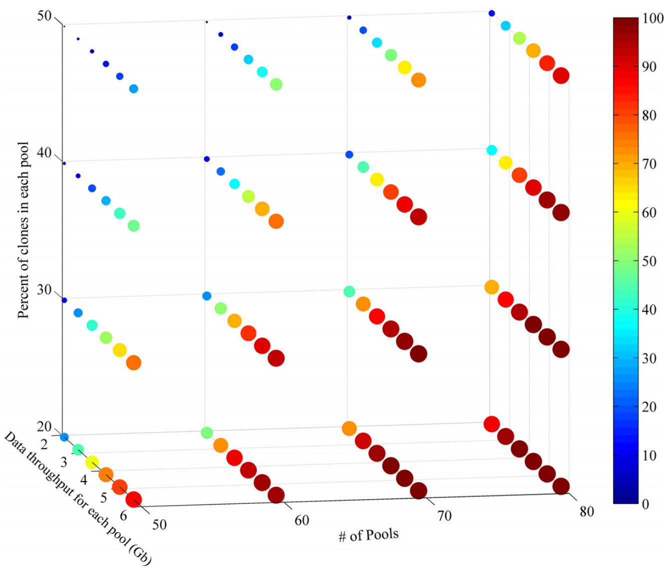

Figure 2. The correct decoding rate for different combinations of various *k* (the percent of clones that are pooled in each pool), *t* (the number of pools) and *dt* values (the data throughput for each pool). The color and size of the circle denote the correct decoding rate for each scenario.

that setting the parameter *k* (the percent of clones in each pool) at 20% (3252 clones per pool) produced the best performance, because the sequencing depth for each clone was higher when fewer clones were contained in the pool as the data throughput was fixed. However, when *k* was less than 20%, because fewer clones within a pool would reduce the encoding space, it was hard to construct a pooling matrix with each column being unique to any other column, meaning clones could possess identical pooling signatures, which prevented correct decoding.

Therefore, we next fixed *k* as 20% and calculated the correct decoding rate for various *t* and *dt* values (Supplementary Figure S3). The results showed that the correct decoding rates were lower when fewer pools or a lower data throughput were used. Obviously, there was a tradeoff between the number of pools and the data throughput. Hence, numerous simulations needed to be performed to verify whether a pair of pool number and data throughput could succeed in achieving high accuracy. Here, we set the threshold for the correct decoding rate as 95%. The results showed that 50 pools required more than 5.6GB, 60 pools required 4.9GB, 70 pools required 3.5GB and 80 pools required 2.8GB of data throughput (Supplementary Figure S3). Considering that 50 and 60 pools needed more sequencing data and 80 pools increased the cost of sequencing library preparation, we chose the design with 70 pools and 3.5GB data for each pool, meaning each pool had about 7.68× sequencing depth for each clone and the overall sequencing depth per clone was about 107×. We next verified the decoding ability of the chosen design for variants covered by various numbers of clones (Supplementary Figure S4). The results showed that the chosen design was successful for more than 95% of variants covered by three clones and could even assign correctly 80% of variants covered by four clones, which was consistent with our expectation.

Simulation

According to the chosen overlapping pool design, we employed *pIRS* (35) to simulate 100 bp Illumina sequencing reads for each clone and then mixed them. The average sequencing error rate was set as 1%, which decided the 99% base calling accuracy, and the simulated sequencing depth followed a Poisson distribution (Supplementary Figure S5). Furthermore, mixing bias was also added to the simulation procedure to bring it closer to a real situation. Based on the study conducted by Shental *et al.* (23), a random variable following a Gaussian distribution was added to each non-zero element of the pooling matrix to simulate mixing bias. The standard deviation of the Gaussian distribution was 0.05, reflecting up to 5% average noise in the mixed quantities of each sample.

For each block with 16 264 clones, sequencing of 70 pools was conducted by mixing reads for each clone *in silico*. Bwa-0.7.12 (36) was used to map the reads back to hg19, and GATK 3.4–46 (37) was used to call variants for each pool. We applied our decoding algorithm to assign these alleles to their original clones. We only analyzed the results for chromosome 1 of the human genome because of a limited computation capability.

To assemble haplotype contigs based on the allele-assigned results, HapCUT (33) required both clone sequences and the corresponding variant set. The clones were selected first, each of which carried more than one allele and was located in chromosome 1.

In the clone selecting procedure, 10 914 clones carrying more than one allele were located in chromosome 1 initially. Nevertheless, 1044 of the 10 914 were found to belong to other chromosomes, which were mis-assigned to chromosome 1 and were defined as false positive clones. Actually, these false positive clones resulted from false read mapping and could be relieved by longer sequencing reads, which produced more accurate read mapping. We implemented two experiments with different read lengths (50 and 75 bp) to reconstruct clones. The results proved that the numbers of false positive and false negative clones were both reduced for longer reads (see Supplementary Table S3), because longer reads could be mapped more accurately than shorter reads and resulted in fewer alleles that were mis-assigned to incorrect clones. Meanwhile, these false positive clones could also be reduced when the alleles for the whole genome are considered and assigned. Remarkably, more alleles were carried by true positive clones than by false positive clones (Supplementary Figure S6), which means the false positive clones represent only a small fraction of the clones of other chromosomes that are mis-anchored by mis-mapping reads. When considering all the chromosomes, the reconstructed clones would be anchored correctly according to the majority alleles, despite a few alleles being assigned to them that are mis-anchored to chromosome 1.

Alternatively, to filter out these false positive clones, we chose at least 10 reference alleles that are located uniformly between the assigned alleles for each clone to verify whether the clones belong to chromosome 1. The candidate clones were retained if the assignment for more than 70% of the chosen alleles supported the clone belonging to chromosome 1, and were filtered out otherwise. Finally, 9881 clones were left, of which only 53 false positive clones that belong to other chromosomes were mis-assigned to chromosome 1 (Supplementary Figure S6). However, 39 clones belonging to chromosome 1 were also filtered out, mainly resulting from the intra-chromosome repeat sequences and the limited capability of the design to assign those alleles that were covered by too many clones. Meanwhile, 854 clones were missed in the recovered clone set (Supplementary Figure S7). We found that the majority of these false negative clones spanned very few or no variants (Supplementary Figure S8).

To obtain the recovered variant set for HapCUT, all the alleles that were assigned to at least one selected clone were collected. The results showed that we could recover ∼93.5% of the variants from all those input (Table 1, Supplementary Figure S9), which was defined as the variants recovery accuracy. In chromosome 1, for all the 221 009 variants that were covered by clones from both parental chromosomes, 220 734 (99.9%) were recovered correctly. However, for the 25 577 variants that were covered by clones from at most one chromosome, most of them were missed or called incorrectly. In theory, homozygous variants covered by clones from only one chromosome should be recovered correctly; however, heterozygous variants would be mis-identified as homozygous if only the non-reference allele was sampled by clones, and would be lost when only the reference allele was sampled by clones.

On the basis of the clone sequences and the recovered variant set, HapCUT (33) was used to assemble clones into haplotypes. In summary, 112 haplotype contigs were assembled to form haplotypes with an N50 length of 3.41 Mb (Table 2). Homozygous variants were neglected in HapCUT; therefore, we only analyzed the heterozygous variants to measure the accuracy of the assembled haplotypes. When comparing the assembled haplotypes with the input individual haplotypes, a variant could fall into one of five categories: (i) matching, variants with alleles identical in both homologous chromosomes; (ii) mis-matching, variants with at least one allele different in the homologous chromosomes; (iii) switch, variants with alleles switched and a crossover was needed to recover the true phase; (iv) false negative variants, variants that were missed in the assembled haplotypes, and (v) false positive variants, false variants that occurred only in the assembled haplotypes. These values were defined as the haplotyping accuracy to describe the matching degree between the assembled and the simulated input haplotypes. We maximized the number of matching variants to anchor the assembled haplotype contigs to the homologous chromosomes, respectively, and counted the number of variants falling in each category (Table 3). The results showed that no mis-matching, switch and false positive variants existed in the assembled haplotypes and only 9783 false negative variants occurred. These variants were missed because they were covered by clones from at most one chromosome, and were mis-identified as homozygous or lost during variant calling. The results revealed that our method achieved highly accurate haplotypes and could repress switch errors, indicating that our method represents an accurate option for individual haplotyping.

Comparison with current methods

To verify the effectiveness of our method, we conducted an experiment to compare the performance of our method with current clone-based haplotyping methods. There are several dilution-based haplotyping methods, including long fragment read (LFR) technology (1), statistically aided, long-read haplotyping (SLRH) (11), fosmid-based (14,15) and BAC-based methods (13). Alternative implementations of these methods mainly differ in the length of the original fragments, the number of pools sequenced and other parameters, which directly affect the cost versus haplotype length trade-off. For example, LFR maximized the clone coverage to 57.6×, which resulted in low sequencing depth for each clone (<2×) for a given total read coverage (1). SLRH sheared DNA into fragments of 10 kb and the total clone coverage was only 3.84× which needed extra statistical data for haplotyping (11) (Supplementary Table S4). Lo analyzed the effects of haplotype length on different pooling parameters and found that the length of clones (or fragments) should be chosen as high as possible to haplotype longer contigs (13). The results of an experiment using 129 024 BAC clones with 140 kb (6× clone coverage) pooled into 24 pools reported an N50 contig length of 2.4 M (13), which performed better than other dilution-based methods (Supplementary Table S4). Meanwhile, more clones need much more sequencing data in our strategy and 6× clone coverage is sufficient to provide variants recovery of 93.5% in our simulation. Hence, we simulated experiments based on Lo's design to compare with our OPS method.

Taking the 130 110 clones of sample NA12878 constructed in our experiment as the source data, we randomly pooled these clones into 24 pools following Lo's strategy, with each pool containing 5421 clones. Meanwhile, in our method, the 130 110 clones were pooled into 560 pools because of the overlap pooling strategy. In Lo's method, dilution of more pools would lead to fewer overlapping clones in each pool and more effective alleles could be used to link clones. To retain consistency with our method, the experiment employing Lo's method with 130 110 clones being pooled into 560 pools was also conducted and each pool contained about 232 clones.

For the experiments with 24 pools and 560 pools, the sequencing depth for each clone was set as 30× to guarantee that more than 99% of base pairs in the clone were covered by at least 15 sequence reads. After generating artificial reads for each pool by running *pIRS*, we pooled together all the sequencing data and called the variants using GATK 3.4–46 (37) (Supplementary Figure S10). The results of 24 and 560 pools were very similar, because variant calling is based on the sequencing data, on which the number of pools would have little influence. Compared with our method, Lo's results presented some false positive variants that may have resulted from sequencing errors and false mapping. However, this situation could be relieved in our method by the error-tolerance of overlapping pool sequencing.

Clones were reconstructed using *targetcut* in the SAMtools library (38) to identify regions of enriched coverage (i.e. clone contigs). Next, we broke up the identified regions where significant changes in coverage occur, representing

Table 1. Comparison between the recovered variants set and the input variants set

Category	# of variants covered by clones from at most one chromosome	# of variants covered by clones from both parental chromosomes	In total
Matching	9819	220 734	230 553
Mis-matching	7581	31	7612
Missing	8177	244	8421
In total	25 577	221 009	246 586

Table 2. The statistics of the assembled haplotypes for chromosome 1

	OPS[a]	Lo's method[b]	Lo's method[c]	Perfect Assembly[d]
Number of contigs	112	177	109	110
Total length	204 296 736	200 122 742	204 613 934	204 643 568
Average length	1 824 078	1 130 636	1 877 192	1 860 396
Max	8 854 891	7 038 610	8 854 891	10 616 834
Min	1655	1259	1655	1655
N50	3 415 257	2 027 638	3 415 257	3 577 657
Avg het	1138.0	717.6	1166.6	1160.3

[a]The overlapping pooled clone sequencing-based haplotyping (our method).
[b,c]Lo's clone-based haplotyping method (13). Twenty-four pools and 560 pools were used, respectively, for [b] and [c].
[d]The assembly for the perfect scenario, which has been defined as a comparison baseline.
'Avg het' represents the average number of heterozygous variants on a haplotype contig.

Table 3. The accuracy of the assembled haplotypes

	OPS[a]	Lo's method[b]	Lo's method[c]	Perfect Assembly[d]
Match	127 454	122 620	126 617	127 631
Mismatch	0	0	0	0
Switch	0	4361	507	0
False negative	9783	9179	10 140	9773
False positive	0	27	33	0

[a]The overlapping pooled clone sequencing-based haplotyping (our method).
[b,c]Lo's clone-based haplotyping method (13). Twenty-four pools and 560 pools were used, respectively, for [b] and [c].
[d]The assembly for the perfect scenario, which has been defined as a comparison baseline.
Homozygous variants were neglected in HapCUT and the numbers of heterozygous variants are shown.

the overlapping parts of the clones. Details of the protocol can be found in (13) and the length distributions of reconstructed clones on chromosome 1 are shown in Supplementary Figure S11. We still used HapCUT to assemble the haplotypes and the results are shown in Table 3. In the assembled haplotypes, our method produced 127 454 matched variants, which was higher compared with Lo's method, regardless of whether 24 or 560 pools were used. For the switch errors, in the experiment with 24 pools, 4361 switch errors occurred, while in the experiment employing Lo's method with 560 pools, there were only 507 switch errors, representing a significant decrease. However, there were no switch errors in our method. The results of the matched variants and the switch errors fully proved that the haplotypes generated by our method were much more accurate than those produced by Lo's method employing 24 and 560 pools.

To establish a comparison baseline for these assembly haplotypes, we considered the perfect scenario where all alleles were recovered and in consistent with the original clones, and all the alleles were assigned to the correct clones with 100% accuracy. Using these alleles and clones, we employed HapCUT to assemble haplotype contigs, defined as the perfect assembly. Subsequently, we compared the perfect assembly with the assembled haplotype contigs generated by our method (OPS) and Lo's method (the details are shown in Tables 2 and 3, Figure 3, and Supplementary Figures S12 and S13). In Table 3, 9773 false negative errors appeared in the perfect assembly scenario because the clones in the perfect assembly were also randomly extracted from long subsections of a haploid, which might miss covering some variant sites. These missed variants were counted in as the false negative errors (Supplementary Figure S12).

Compared with Lo's method, whether using 24 or 560 pools, our method (OPS) produced results that were more similar to the perfect assembly, which proved that our method was more accurate. Figure 3 shows the capability of different methods to resolve alleles into their haploid clones. Every scatter point in Figure 3 represents a reconstructed clone and the coordinates of the point represent the number of alleles whose genotype was identical to one of the haplotypes for the individual NA12878. The figure shows that almost all the alleles on the same clone reconstructed by our method supported only one haplotype (Figure 3A), which was similar to the perfect scenario (Figure 3B). For Lo's method, considering that the overlapping clone parts within a pool were broken and eliminated, both reconstructed clones before and after the break procedure were presented in Figure 3C–F, respectively. These figures showed that the alleles on some clones reconstructed by Lo's method indeed represented different haplotypes, which meant errors had occurred during the clone reconstruction. Supplementary

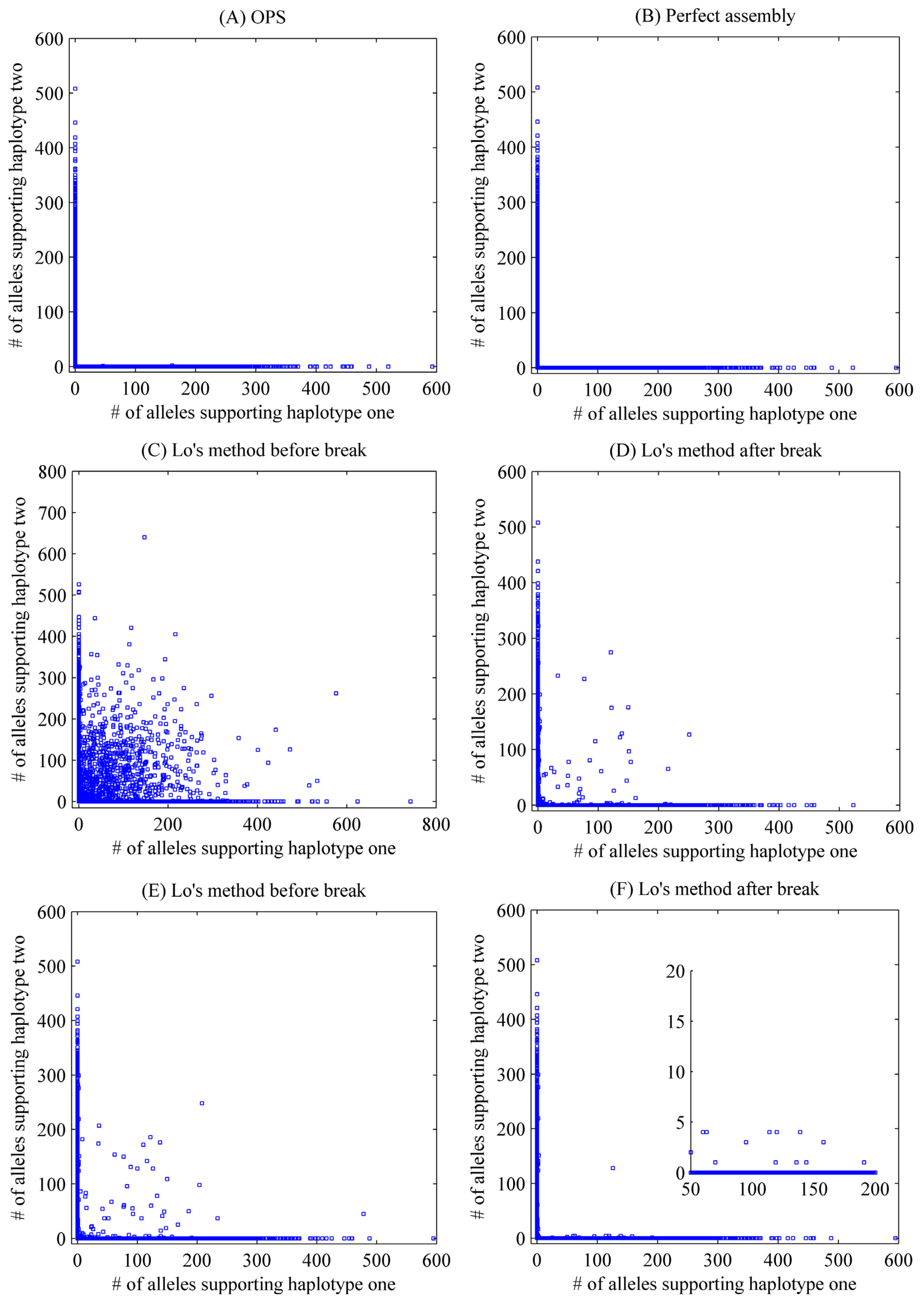

Figure 3. The number of alleles in each reconstructed clone sequence that support each haplotype in the diploid individual. Every scatter point in Figure 3 represents a reconstructed clone and the coordinates of the point represent the number of alleles whose genotype was identical to one of the haplotypes for the individual NA12878. Subplots stand for the reconstructed clones for (**A**) our method; (**B**) perfect assembly; (**C**) Lo's method before the break where 24 pools were used; (**D**) Lo's method after the break where 24 pools were used; (E) Lo's method before the break where 560 pools were used; (F) Lo's method after the break where 560 pools were used.

Figure S12 shows that more switch errors appeared in Lo's method using 24 pools. We inferred that our method could repress switch errors because of accurate allele assignment. To support this, we conducted an experiment to determine why switch errors appeared in Lo's method. Clones covering the switch location and carrying variants from both parental sides that appeared in Lo's strategy were eliminated and the remaining clones were assembled. The results showed that the switch errors were avoided (Supplementary Table S5). This experiment proved that the incorrect assignment of variants in the simple-pooling haplotyping strategy is the main reason for switch errors. In our method, on the basis of accurate allele assignment, the majority of the reconstructed clones were pure haplotype contigs, which could repress switch errors.

More examples of error assembly in Supplementary Figure S13 showed that, compared with our method or the perfect assembly, Lo's method tended to either link some disjunct haplotype contigs or break some contigs. These errors might result from incorrect genotyping of variants and retained overlapping clones.

However, the high accuracy of our method needs substantial sequencing data of 1960GB (3.5GB $\times$ 560). Actually, the sequencing data could be reduced. In our simulation, the chosen parameters (70 pools, 3.5GB sequencing data and 20% clones per pool) aimed to recover >95% variants covered by three clones. The decoding requirement is too high, since alleles covered by three clones constitute a very small fraction of the alleles and the majority of the alleles were covered by fewer than three clones for each block, with 0.75 clone coverage according to the Poisson distribution (Supplementary Figure S2). To prove that the sequencing data could be reduced in our method, we implemented a simulation to calculate the accuracy of the assembled haplotypes under different amounts of sequencing data (see Supplementary Table S6). The correct-assigning rates for variants covered by 1, 2 and 3 clones and the average rate were calculated respectively, based on which the haplotyping were simulated employing HapCUT (Supplementary Table S6). The results indicated that the correct-assigning accuracy dropped as the amount of sequencing data was reduced (Supplementary Figure S14). According to this curve, we simulated another experiment using half as much sequencing data (980GB), where the results met our expectations and proved that our method could retain a fairly high haplotyping accuracy using less sequencing data, although 511 switch errors occurred because of a few incorrectly assigned alleles (Supplementary Table S7).

The program's timings are listed in Supplementary Table S8, according to our simulation experiment on a Dell T630 workstation with 32 CPUs (encoding is hardly time-consuming and is not listed). Results showed that the decoding took less time than other steps. However, the calling variants step for a whole genome took about 71 days which cost too much time. Because of the limited disk storage of our workstation, variants were called from each pool separately and merged at last in our experiment. Actually, if the hardware of a workstation allowed, the sequencing data from all the pools could be integrated together and used to call variants, which could save the time dramatically. We conducted another experiment to call variants from the integrated data of a whole block, not from separate pools, and the estimated time for calling variants of one genome dropped from 71 days to about 12 days (Supplementary Table S9). Meanwhile, the statistic of alleles called from separate pools and the whole block showed that the calling accuracy of the two strategies had no big differences (Supplementary Figure S15).

Significantly, to make our method easier to use, OPShap (in Perl) for encoding and decoding, with detailed instructions, is available online at http://bioinfo.seu.edu.cn/OPShap.

DISCUSSION

We proposed a clone-based haplotyping method employing overlapping pool sequencing to design pooling patterns and then decode them. First, the clones are randomly divided into several blocks in which the clones are pooled 'intra-block', so that the pooling and decoding complexity are dramatically reduced. The clones in every block are then pooled combinatorially following the optimal pooling pattern according to the random size-k design and sent for sequencing. The alleles called from the sequencing result are then assigned to the correct clone set using techniques derived from the compressed sensing theories. On the basis of the correct assignment of alleles, the individual's haplotypes could be phased by assembling the reconstructed clones.

Compared with current simple-pooling haplotyping strategy, our method achieves higher accuracy and longer haplotype contigs. Significantly, in our simulation experiment, the switch errors produced in Lo's method were repressed in our method, and false linkages or truncations of haplotype contigs were also corrected. Meanwhile, our method is sequencing error-tolerant, because random size-k design could obtain an error-tolerant matrix M, which could correct a small number of sequencing errors. When an allele of a clone was sequenced as the wrong base pair in a pool but sequenced correctly in other pools, this sequencing error could be corrected based on the majority of correct sequencing results. The capability to correct sequencing errors was demonstrated by the accuracy of variant calling in the experiment. Furthermore, compared with the variants recovery accuracy of 93.5% under 6$\times$ clone coverage in our method, LFR recovered an average of 92.5% variants under very high haploid fragment coverage (38–116) (1), and both Lo (13) and SLRH (11) need extra data to fill gaps between haplotyping contigs to recover over 90% of variants. Therefore, we believe our method performs better than other dilution-based methods and is reliable for phasing the chromosomes. Besides the improvement in accuracy, our method could construct all the clones and locate them on genomes, which are valuable by-products that are achieved at no extra cost. This kind of information might be used for other biological research, such as the discovery of large genomic inversions.

With such high accuracy, our method still has some issues that should be resolved before being widely accepted. First, at present, longer sequencing read technologies produce reads that are substantially longer than the read lengths of the NGS platforms, such as Illumina's phased-sequencing platform (11,39). In essence, these methods produce much longer virtual reads on the basis of diluting long genome fragments into substantial pools. The mechanism is identical to those haplotyping methods that are based on dilution (10,11). Furthermore, virtual reads by these technologies are still no longer than 10 kb because of the limitation of the PCR's ability to amplify fragments longer than 10 kb (40). Hence, it is likely that the contiguity of haplotype assemblies resulting from these methods would be limited compared with those haplotyping methods that exploit much longer DNA fragments, such as fosmid or BAC clones. Besides, longer reads would lead to more accurate mapping, which might improve the accuracy of allele assignment and clone reconstruction in our method. In addition, multiple alleles in the same read could validate the allele assignment. Significantly, although single-molecule sequencing technologies, such as the Nanopore (41) and PacBio (42) sequencing platforms, allow the possibility of obtaining long sequence reads of $>$10 kb without conducting PCR, these platforms still need to solve some issues before being widely used, such as low output and the comparatively high error rate, which requires error-correcting steps based on the data obtained from second-generation sequencing platforms (43,44). Moreover, in the evolutionary research fields of population genomics, metagenomics and phylogenomics, the second-generation techniques will remain state of the art for at least the next few years (45).

Second, the need for substantial sequencing data could be reduced. The results of our calculation on different sequencing depths indicated that the correct-assigning accuracy drops slightly as the amount of sequencing data decreases. The results of our experiment using half as much sequencing data met our expectations and proved that our method could retain a fairly high haplotyping accuracy using less sequencing data. However, some switch errors occurred because of a few incorrectly assigned alleles. Users of our method could calculate the decoding results under different combinations, and then choose parameters according to their requirements, such as the sequencing cost or number of pools.

Third, our strategy requires more pools to guarantee a sufficient and distinct combinatorial pooling pattern for different clones. More pools mean more clone picking and pooling operation, which is laborious, but fortunately an automatic protocol *iPipet* (46) has been proposed recently to pick and pool clones, which could vastly relieve the efforts for the clone picking and pooling operation.

Finally, because the downstream analysis of our strategy was based on large amounts of data, the timings might be one of the major concerns of potential users. As aforementioned, variants calling strategy could be improved and the analysing time would be reduced effectively. Actually, considering the large-scale workstations, such as supercomputing centers or cloud computation platforms, the time for analysis of one genome could also be further reduced. For instance, we expect that the analysis for a whole genome could be finished in just one day with 120 CPUs' parallel calculation.

In summary, our proposed strategy could construct highly accurate haplotypes based on an individual's clones. Our method offers multiple options for individual haplotyping, almost perfect accuracy with higher cost, or cutting sequencing cost by lowering accuracy to an extent that is still ideal in most scenarios. At present, considering the sequencing cost as the main factor in selecting a haplotyping method, the parameters of our method could be adjusted to reduce the sequencing data. As the cost of sequencing continues to decrease, almost perfect haplotyping accuracy could be achieved using our strategy by improving the sequencing depth. Therefore, we conclude that, compared with current simple-pooling haplotyping approaches, our method achieves a more accurate performance and has great potential for future application.

SUPPLEMENTARY DATA

Supplementary Data are available at NAR Online.

ACKNOWLEDGEMENTS

We thank Dr. Kuleshov and his co-workers for providing access to SNP variants for the HapMap sample used in this study. We also thank the two anonymous reviewers for their helpful comments.

FUNDING

National Basic Research Program of China [2012CB316501]; National Natural Science Foundation of China [61472078]; Scientific Research Foundation of Graduate School of Southeast University. Funding for open access charge: National Basic Research Program of China [2012CB316501]; National Natural Science Foundation of China [61472078]; Scientific Research Foundation of Graduate School of Southeast University.
Conflict of interest statement. None declared.

REFERENCES

1. Peters,B.A., Kermani,B.G., Sparks,A.B., Alferov,O., Hong,P., Alexeev,A., Jiang,Y., Dahl,F., Tang,Y.T., Haas,J. *et al.* (2012) Accurate whole-genome sequencing and haplotyping from 10 to 20 human cells. *Nature*, **487**, 190–195.
2. Browning,S.R. and Browning,B.L. (2011) Haplotype phasing: existing methods and new developments. *Nat. Rev. Genet.*, **12**, 703–714.
3. Glusman,G., Cox,H.C. and Roach,J.C. (2014) Whole-genome haplotyping approaches and genomic medicine. *Genome Med.*, **6**, 73.
4. Rao,W.N., Ma,Y.M., Ma,L., Zhao,J., Li,Q.L., Gu,W.K., Zhang,K., Bond,V.C. and Song,Q. (2013) High-resolution whole-genome haplotyping using limited seed data. *Nat. Methods*, **10**, 6–7.
5. Li,Y., Willer,C.J., Ding,J., Scheet,P. and Abecasis,G.R. (2010) MaCH: using sequence and genotype data to estimate haplotypes and unobserved genotypes. *Genet. Epidemiol.*, **34**, 816–834.
6. Howie,B., Marchini,J. and Stephens,M. (2011) Genotype Imputation with Thousands of Genomes. *G3-Genes Genom. Genet.*, **1**, 457–469.
7. Fan,H.C., Wang,J.B., Potanina,A. and Quake,S.R. (2011) Whole-genome molecular haplotyping of single cells. *Nat. Biotechnol.*, **29**, 51–57.
8. Yang,H., Chen,X. and Wong,W.H. (2011) Completely phased genome sequencing through chromosome sorting. *Proc. Natl. Acad. Sci. U.S.A.*, **108**, 12–17.
9. Selvaraj,S., Dixon,J.R., Bansal,V. and Ren,B. (2013) Whole-genome haplotype reconstruction using proximity-ligation and shotgun sequencing. *Nat. Biotechnol.*, **31**, 1111–1118.
10. Kaper,F., Swamy,S., Klotzle,B., Munchel,S., Cottrell,J., Bibikova,M., Chuang,H.Y., Kruglyak,S., Ronaghi,M., Eberle,M.A. *et al.* (2013) Whole-genome haplotyping by dilution, amplification, and sequencing. *Proc. Natl. Acad. Sci. U.S.A.*, **110**, 5552–5557.

11. Kuleshov,V., Xie,D., Chen,R., Pushkarev,D., Ma,Z., Blauwkamp,T., Kertesz,M. and Snyder,M. (2014) Whole-genome haplotyping using long reads and statistical methods. *Nat. Biotechnol.*, **32**, 261–266.
12. Amini,S., Pushkarev,D., Christiansen,L., Kostem,E., Royce,T., Turk,C., Pignatelli,N., Adey,A., Kitzman,J.O., Vijayan,K. *et al.* (2014) Haplotype-resolved whole-genome sequencing by contiguity-preserving transposition and combinatorial indexing. *Nat. Genet.*, **46**, 1343–1349.
13. Lo,C., Liu,R., Lee,J., Robasky,K., Byrne,S., Lucchesi,C., Aach,J., Church,G., Bafna,V. and Zhang,K. (2013) On the design of clone-based haplotyping. *Genome Biol.*, **14**, R100.
14. Kitzman,J.O., Mackenzie,A.P., Adey,A., Hiatt,J.B., Patwardhan,R.P., Sudmant,P.H., Ng,S.B., Alkan,C., Qiu,R., Eichler,E.E. *et al.* (2011) Haplotype-resolved genome sequencing of a Gujarati Indian individual. *Nat. Biotechnol.*, **29**, 59–63.
15. Suk,E.K., McEwen,G.K., Duitama,J., Nowick,K., Schulz,S., Palczewski,S., Schreiber,S., Holloway,D.T., McLaughlin,S., Peckham,H. *et al.* (2011) A comprehensively molecular haplotype-resolved genome of a European individual. *Genome Res.*, **21**, 1672–1685.
16. Burgtorf,C., Kepper,P., Hoehe,M., Schmitt,C., Reinhardt,R., Lehrach,H. and Sauer,S. (2003) Clone-based systematic haplotyping (CSH): A procedure for physical haplotyping of whole genomes. *Genome Res.*, **13**, 2717–2724.
17. Cao,C.C., Li,C., Huang,Z., Ma,X. and Sun,X. (2013) Identifying rare variants with optimal depth of coverage and cost-effective overlapping pool sequencing. *Genet. Epidemiol.*, **37**, 820–830.
18. Dorfman,R. (1943) The detection of defective members of large populations. *Ann. Math. Statist.*, **14**, 436–440.
19. Macula,A.J. (1997) Error-correcting nonadaptive group testing with d(e)-disjunct matrices. *Discrete Appl. Math.*, **80**, 217–222.
20. Cao,C.C., Li,C. and Sun,X. (2014) Quantitative group testing-based overlapping pool sequencing to identify rare variant carriers. *BMC Bioinformatics*, **15**, 195.
21. Erlich,Y., Chang,K., Gordon,A., Ronen,R., Navon,O., Rooks,M. and Hannon,G.J. (2009) DNA Sudoku-harnessing high-throughput sequencing for multiplexed specimen analysis. *Genome Res.*, **19**, 1243–1253.
22. Prabhu,S. and Pe'er,I. (2009) Overlapping pools for high-throughput targeted resequencing. *Genome Res.*, **19**, 1254–1261.
23. Shental,N., Amir,A. and Zuk,O. (2010) Identification of rare alleles and their carriers using compressed se(que)nsing. *Nucleic Acids Res.*, **38**, e179.
24. Erlich,Y., Gordon,A., Brand,M., Hannon,G.J. and Mitra,P.P. (2010) Compressed Genotyping. *IEEE Trans. Inf. Theory*, **56**, 706–723.
25. Lustig,M., Donoho,D. and Pauly,J.M. (2007) Sparse MRI: The application of compressed sensing for rapid MR imaging. *Magn. Reson. Med.*, **58**, 1182–1195.
26. Lin,T.T. and Herrmann,F.J. (2007) Compressed wavefield extrapolation. *Geophysics*, **72**, Sm77–Sm93.
27. Zhou,W.P., Li,Y., Liu,Q.S., Wang,G.D. and Liu,Y. (2014) Fast compression and reconstruction of astronomical images based on compressed sensing. *Res. Astron. Astrophys.*, **14**, 1207–1214.
28. Rao,A., Deepthi,P., Renumadhavi,C.H., Chandra,M.G. and Srinivasan,R. (2015) Compressed sensing methods for DNA microarrays, RNA interference, and metagenomics. *J. Comput. Biol.*, **22**, 145–158.
29. Hwang,F.K. (2000) Random k-set pool designs with distinct column. *Probabil. Eng. Inform. Sci.*, **14**, 49–56.
30. Kainkaryam,R.M. and Woolf,P.J. (2008) poolHiTS: A Shifted Transversal Design based pooling strategy for high-throughput drug screening. *BMC Bioinformatics*, **9**, 256.
31. Pabinger,S., Dander,A., Fischer,M., Snajder,R., Sperk,M., Efremova,M., Krabichler,B., Speicher,M.R., Zschocke,J. and Trajanoski,Z. (2014) A survey of tools for variant analysis of next-generation genome sequencing data. *Brief. Bioinformatics*, **15**, 256–278.
32. Foucart,S. and Koslicki,D. (2014) Sparse recovery by means of nonnegative least squares. *IEEE Signal Process. Lett.*, **21**, 498–502.
33. Bansal,V. and Bafna,V. (2008) HapCUT: an efficient and accurate algorithm for the haplotype assembly problem. *Bioinformatics*, **24**, I153–I159.
34. Lancia,G., Bafna,V., Istrail,S., Lippert,R. and Schwartz,R. (2001) SNPs problems, complexity, and algorithms. In: *Proceedings of the Ninth European Symposium on Algorithms, Aarhus, Denmark, Lecture Notes in Computer Science*, Springer, Heidelberg, Berlin, pp. 182–193.
35. Hu,X.S., Yuan,J.Y., Shi,Y.J., Lu,J.L., Liu,B.H., Li,Z.Y., Chen,Y.X., Mu,D.S., Zhang,H., Li,N. *et al.* (2012) pIRS: profile-based Illumina pair-end reads simulator. *Bioinformatics*, **28**, 1533–1535.
36. Li,H. and Durbin,R. (2009) Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics*, **25**, 1754–1760.
37. McKenna,A., Hanna,M., Banks,E., Sivachenko,A., Cibulskis,K., Kernytsky,A., Garimella,K., Altshuler,D., Gabriel,S., Daly,M. *et al.* (2010) The Genome Analysis Toolkit: A MapReduce framework for analyzing next-generation DNA sequencing data. *Genome Res.*, **20**, 1297–1303.
38. Li,H., Handsaker,B., Wysoker,A., Fennell,T., Ruan,J., Homer,N., Marth,G., Abecasis,G. and Durbin,R. (2009) The Sequence Alignment/Map format and SAMtools. *Bioinformatics*, **25**, 2078–2079.
39. Voskoboynik,A., Neff,N., Sahoo,D., Newman,A., Pushkarev,D., Koh,W., Passarelli,B., Fan,H., Mantalas,G., Palmeri,K. *et al.* (2013) The genome sequence of the colonial chordate, Botryllus schlosseri. *Elife*, **2**, e00569.
40. Snyder,M.W., Adey,A., Kitzman,J.O. and Shendure,J. (2015) Haplotype-resolved genome sequencing: experimental methods and applications. *Nat. Rev. Genet.*, **16**, 344–358.
41. Clarke,J., Wu,H.C., Jayasinghe,L., Patel,A., Reid,S. and Bayley,H. (2009) Continuous base identification for single-molecule nanopore DNA sequencing. *Nat. Nanotechnol.*, **4**, 265–270.
42. Eid,J., Fehr,A., Gray,J., Luong,K., Lyle,J., Otto,G., Peluso,P., Rank,D., Baybayan,P., Bettman,B. *et al.* (2009) Real-time DNA sequencing from single polymerase molecules. *Science*, **323**, 133–138.
43. Loman,N.J., Quick,J. and Simpson,J.T. (2015) A complete bacterial genome assembled *de novo* using only nanopore sequencing data. *Nat. Methods*, **12**, 733–735.
44. Jain,M., Fiddes,I.T., Miga,K.H., Olsen,H.E., Paten,B. and Akeson,M. (2015) Improved data analysis for the MinION nanopore sequencer. *Nat. Methods*, **12**, 351–356.
45. Bleidorn,C. (2016) Third generation sequencing: technology and its potential impact on evolutionary biodiversity research. *Syst. Biodivers.*, **14**, 1–8.
46. Zielinski,D., Gordon,A., Zaks,B.L. and Erlich,Y. (2014) iPipet: sample handling using a tablet. *Nat. Methods*, **11**, 784–785.