RESEARCH

WhatsHap: fast and accurate read-based phasing

Marcel Martin[1†], Murray Patterson[2†], Shilpa Garg[6,7], Sarah Fischer[6], Nadia Pisanti[3], Gunnar W Klau[4], Alexander Schönhuth[4] and Tobias Marschall[6,7*]

*Correspondence:
t.marschall@mpi-inf.mpg.de
[6]Center for Bioinformatics, Saarland University, Campus E2.1, 66123, Saarbrücken, Germany
[7]Max Planck Institute for Informatics, Saarbrücken, Germany
Full list of author information is available at the end of the article
†Equal contributor

Abstract

Read-based phasing allows to reconstruct the haplotype structure of a sample purely from sequencing reads. While phasing is a required step for answering questions about population genetics, compound heterozygosity, and to aid in clinical decision making, there has been a lack of an accurate, usable and standards-based software.

WhatsHap is a production-ready tool for highly accurate read-based phasing. It was designed from the beginning to leverage third-generation sequencing technologies, whose long reads can span many variants and are therefore ideal for phasing. WhatsHap works also well with second-generation data, is easy to use and will phase not only SNVs, but also indels and other variants. It is unique in its ability to combine read-based with genetic phasing, allowing to further improve accuracy if multiple related samples are provided.

Keywords: read-based phasing; haplotype assembly; long reads

1 Background

Variant calling finds isolated differences between an individual's genome and a known reference. The most common variants are single nucleotide variants (SNVs), insertions and deletions (indels). The reported information for a variant includes its *genotype*, which is the set of alleles at that locus, such as A/A (homozygous) or G/A (heterozygous) in a diploid individual. Heterozygous genotypes by definition allow no conclusions about which of the two alleles is paternal and which one is maternal – G/A and A/G are equivalent. Thus, the full sequence of nucleotides in an individual chromosome – the haplotype – cannot be reconstructed. *Phasing* is the process of inferring the correct relationship (*cis* or *trans*) between alleles at multiple variant loci using additional information besides genotypes, and allows to reconstruct these haplotypes. For example, phasing the two heterozygous variants G/A and C/T may lead to the result that the correct haplotypes are G–T (one chromosome) and A–C (the other chromosome).

Phasing allows to obtain crucial insight into various aspects of population genetics, such as population structure, migration and environmental pressure [1, 2]. On an individual level, phased variant data can significantly aid clinical decision making [3]. It is important in investigating compound heterozygosity and allele-specific expression.

1.1 Approaches

Molecular phasing involves physically separating the chromosomes before further processing (such as sequencing). To our knowledge, few wet-lab researchers have

succeeded in doing this for arbitrary samples, but the readily available CHM1 cell line (which is haploid) could be considered to fall in this category.

Genetic (pedigree-based) phasing makes use of family relationships between individuals. For example, if the genotypes at a locus are A/A in the mother, T/T in the father, and A/T in the child, then the A allele of the child is on the maternal haplotype, and the T on the paternal one. Similar conclusions can be made in other cases and in larger pedigrees as long as the variant is homozygous in at least one individual.

Population-based (statistical) phasing uses regions of shared ancestry, obtained from a reference panel created from many individuals.

Finally, *read-based phasing* uses sequencing reads spanning at least two heterozygous variants to infer phase. It is also called *haplotype assembly* since the individual reads may be seen as haplotype fragments that need to be assembled to obtain larger fragments encompassing more variants. Read-based phasing has started to become more and more attractive with the arrival of long-read sequencing technologies since these cover a larger number of variants, allowing to attain a better phasing quality in terms of haplotype block lengths.

WhatsHap, which we introduce here as a usable tool, and whose core algorithm has been previously described [4], is a read-based phasing tool, but it can additionally employ pedigree information if provided with multiple, related samples, thus combining read-based and genetic phasing [5].

1.2 Tools

Until recently, DNA microarrays would be typically used for genotyping, and a range of state-of-the-art tools exist that can phase from the resulting data. SHAPEIT [6] is an example of a software for statistical, population-based phasing and Merlin [7] for genetic, pedigree-based phasing.

Driven by the advances in third-generation sequencing [8, 9, 10] and the corresponding growth in read length, read-based phasing has been gaining attention [11] because it allows to link consecutive variant alleles directly from single reads.

Read-based phasing can be formulated as the NP-hard *Minimum Error Correction* problem [12]. To solve it, heuristic [13, 14, 15] and exact approaches [16, 17, 18, 19] have been suggested, often accompanied by an implementation of the algorithm. Arguably, the main focus has often been on creating proof-of-principle implementations, not on providing stable software. Exceptions to this are GATK's [20] ReadBackedPhasing command, which is one of few tools that directly support standard Variant Call Format (VCF) files for both input and output. It can even be made to work with long reads when the correct command-line parameters are used. Its underlying algorithm is not published, and it is no longer maintained. HapCUT [13] on the other hand is Open Source and even supports phasing of indels, but requires pre-processing of input data and produces non-standard output. Recently, phASER [21] was published with a focus on phasing from RNA sequencing data, but it can also be used for whole genomes, even supporting standard VCF output.

For production use, a software is needed that is easy to use, gives very good phasing results, works with standard formats (VCF and BAM), supports a variety of sequencing technologies and is maintained. WhatsHap fulfills these criteria while

also being based on a solid theoretical framework [4]. It can phase indels, works well with (potentially error-prone) long reads, and can combine read-based and genetic phasing [5]. WhatsHap is available as Open Source software under the MIT license. It has been in production use in various research groups and is being continually maintained and improved.

2 Algorithm and implementation

We describe the main phasing algorithm first since it is central for an understanding of the pre-processing steps.

2.1 Phasing algorithm

WhatsHap solves the *weighted minimum error correction problem* (wMEC) for diploid samples. We summarize here the previously described algorithm [4].

Conceptually, the algorithm works on a *allele matrix* with one column per heterozygous variant and one row per read. Allowed values are 0, 1 and "-", which denote that the read supports the reference allele, the alternative allele, or that the read does not cover the variant, respectively. Paired-end reads occupy a single row, but have a stretch of "-" values in the middle. The goal of the MEC problem is to correct the smallest number of entries from 0 to 1 or vice versa such that the rows can be partitioned into two haplotypes without conflicts. The wMEC version of the problem allows weights in the matrix instead of only 0 and 1.

WhatsHap implements a fixed-parameter tractable (FPT) algorithm with read coverage as fixed parameter that solves the wMEC problem. The algorithm proceeds from left to right over the variants. At each position, it considers all k reads covering that position (for read pairs, also the middle section of the fragment without sequencing data is considered to be covering) and tests all 2^k possibilities of assigning the reads to the two haplotypes. The combinations are enumerated in Gray code order, and dynamic programing with some extra data structures is employed to avoid score recomputation, which taken together allows to update the cost for a subsolution in constant time.

Unlike other phasing tools, the read length does not affect the runtime of WhatsHap, which means it is ready for the ever-longer reads that might arrive with future generations of sequencing technology. Rather, the runtime is governed by the coverage of the reads.

2.2 Read selection

To ensure that read coverage does not exceed a desired threshold, WhatsHap uses a *read selection* heuristic (see [22] for details) as a pre-processing step. The heuristic starts from a coverage of zero and greedily selects the reads most informative for phasing in multiple rounds until the desired coverage is reached. Informative reads are selected based on a per-read score that reflects its suitability for phasing, in particular the number of heterozygous variants it covers. Also, a preference for high-quality over low-quality reads is taken into account in the score. In a separate step in each iteration, reads are added that would otherwise not be selected but that "bridge" islands of covered variants (connected components). This includes, for example, mate-pair reads, which have a low score because they cover few variants,

but span long distances. Note that the two parts of a paired-end or mate-pair read are treated as a single read with a gap in the middle.

It was previously demonstrated [22] how this strategy leads to superiour phasing results compared to random downsampling to the desired coverage. It is particularly powerful when feeding WhatsHap with sequencing data of mixed types, such as combining PacBio and Illumina reads.

By default, read selection reduces coverage to 15×. Earlier experiments have shown that the performance gain beyond that coverage is negligible since mostly redundant information is added [23].

2.3 Detecting alleles in reads

Before actual phasing can begin, the alleles that each read supports need to be detected. Variants are expected in standard Variant Call Format (VCF), and aligned sequencing reads are provided as one or multiple BAM files. The input VCF file contains merely a list of variants and their genotypes, possibly with aggregate information such as the number of reads supporting the reference and alternative allele at a variant position. To use reads covering a heterozygous variant for phasing, we additionally need to detect for each of those reads whether it supports the reference or alternative allele.

Two approaches for allele detection are implemented in WhatsHap, both yield a decision of either "Reference" (0), "Alternative" (1) or "Unknown/Cannot decide" (-) for a single read covering a single heterozygous variant.

The first approach is to inspect existing read alignments in the BAM file: Whenever the alignment contains a match to the reference at a SNV position, the allele is reported as "Reference", and if it is a mismatch with the appropriate other nucleotide, it is reported as "Alternative". A mismatch with one of the other two nucleotides is reported as "Unknown".

This approach has severe limitations when PacBio or Oxford Nanopore reads are involved, which come with high error-rates around 10% that include not only substitution errors, but equally also indel errors. As an example, assume we have the reference sequence TCGTGT with a heterozygous G/A variant at the third nucleotide. One optimal alignment between read TCAGT and the reference could be:

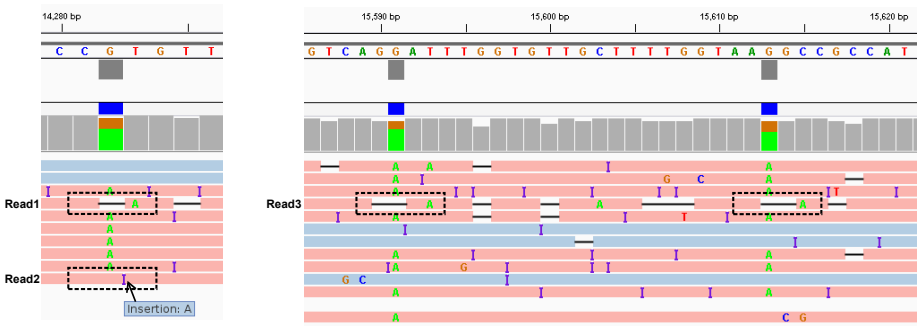
```
Read: TC-AGT
Ref.: TCGTGT
```

Since the aligner is unaware of the alternative allele, it can place the gap at either pos. 3 or 4 because both options result in the same alignment score (one deletion, one mismatch), resulting in an "Unknown" allele in the shown case. Aligning to the alternative reference instead gives a more correct alignment with a better score (match instead of mismatch):

```
Read: TCA-GT
Alt.: TCATGT
```

Current aligners are biased towards the reference sequence since they are unaware of the existence of the alternative allele. This leads to misdetected or undetected alleles when the existing read alignment is trusted. Figure 1 illustrates this on real data.

Figure 1 Noisy sequencing data benefits from re-alignment. This screenshot from IGV (Integrative Genomics Viewer) shows PacBio reads aligned to a G/A variant. In Read1, the variant is covered by a deletion and would be detected as "unknown". Re-aligning to the alternative version of the genomic sequence (with an A) would align the read's A on top of the variant, and it would therefore be detected as "alternative", which is likely the correct explanation. Read2 supports the reference allele although an A insertion is next to the variant. Depending on how alignment scores are chosen, re-alignment allows us to detect either the alternative allele or to determine that both alleles are equally likely.



The second allele detection strategy in WhatsHap is therefore based on re-aligning the read in a way similar to what some variant callers do [24]. Using the existing alignment, WhatsHap first extracts the read sequence in a window around the variant (from 10 bp upstream to 10 bp downstream). It then aligns that sequence to the corresponding section in the reference and also to an artificially created sequence that represents how the reference would look like if it incorporated the alternative allele. The allele with a higher alignment score is then assumed to be supported by the read; for equal scores, the result is "Unknown".

This strategy has multiple advantages: It not only improves phasing performance significantly for noisy long-read data (see evaluation), but as a highly desirable side-effect, the same algorithm allows us to detect and therefore phase indels and multi-nucleotide variants without extra implementation effort. In fact, any variant given by reference and alternative allele in the input VCF file, that is, everything excluding structural variants, can be detected by the re-alignment method.

While re-alignment is slower in theory, it is conceptually simpler, which allowed us to spend more time on optimizing the code. Thus, in our implementation, it does not incur any runtime overhead compared to the alignment-inspecting version.

2.4 Software engineering

The command-line front-end of WhatsHap is written in Python 3 due to its simplicity and maintainability. The core dynamic programming algorithm that solves the wMEC (or PedMEC) problem is implemented in C++. The interface between Python and C++ is written in Cython, which is a Python dialect that compiles directly to C/C++ (`http://www.cython.org/`). Other performance-critical parts have been implemented directly in Cython, such as the re-alignment algorithm.

External libraries used include PyVCF (`https://github.com/jamescasbon/PyVCF`), pysam (`https://github.com/pysam-developers/pysam`) and pyfaidx [25].

2.5 Usage and file formats

WhatsHap is easily installed from the *Python Package Index* (PyPI) or Conda's bioconda channel (`https://bioconda.github.io/`) with a single `pip3 install` or `conda install` command.

WhatsHap is a command-line tool that expects – in the common case – a single Variant Call Format (VCF) file and one or more BAM files as input. The command `whatshap phase -o phased.vcf input.vcf input.bam` is typically sufficient for phasing in non-pedigree mode. Instead of BAM, phasing information can be provided to WhatsHap also as phased VCF files, such as those generated by 10X Genomics' phasing pipeline, and already phased blocks are then treated as "reads". Whatshap phases all samples it finds in a multi-sample VCF, using BAM read groups and VCF sample names to associate each read to its originating sample.

Input reads – even those from a single sample – can be distributed over several BAM (or phased VCF) files, and may use different sequencing technologies. For example, PacBio, Illumina mate-pair reads, and 10X Genomics phased blocks can be used simultaneously for phasing.

WhatsHap outputs an augmented version of the input VCF file, where phasing information for those variants that could be phased has been added. The standardized "PS" tag (phase set) is used to encode phasing results by default. The "HP" tag – compatible with GATK ReadBackedPhasing output – is also supported.

2.6 Combining read-based and genetic phasing

The core algorithm for solving the wMEC [4], and most of the features we have outlined thus far involve the phasing of a single individual. Recentely, Garg et al. [5] have extended the wMEC formalization to exploit pedigree information between a set of related individuals, referred to as *PedMEC*, and integrated a pedigree phasing mode into WhatsHap.

Given multiple samples, the corresponding reads, and a pedigree describing the relationship between samples, a solution to the PedMEC problem minimizes error corrections (like wMEC) in such a way that the Mendelian rules of inheritance imposed by the pedigree are respected and that a minimum recombination cost is incurred.

Pedigree-aware phasing thus makes use of the fact that the haplotypes of a child are a recombination of the parental haplotypes. Combining read-based and genetic phasing is principally more powerful than either method alone: Read-based phasing is limited by read length while genetic phasing, on the other hand, cannot phase variants that are heterozygous in all samples, and it cannot find recombination breakpoints.

Recall that WhatsHap can phase multiple samples that are not related if provided with a multi-sample VCF file and appropriate reads in one or more BAM or phased VCF files. WhatsHap switches to the combined read-based/pedigree phasing algorithm if it is additionally supplied with a PED file (see `http://pngu.mgh.harvard.edu/~purcell/plink/data.shtml`) that describes the relationship between the samples. Recombination events are by default assumed to be equally likely at any position. For more realistic modeling of, for example, recombination hotspots, a vector of recombination rates along the genome (genetic map) can optionally be given.

We have demonstrated earlier [5] that adding pedigree information boosts accuracy immensely. In some cases, the accuracy attained with 15× coverage on a single individual can be achieved with just 2× coverage when a mother–father–child trio is provided.

When in pedigree mode, even variants that are not connected by any read in any sample can often be connected correctly based on pedigree information, in the best case allowing to obtain chromosome-length haplotypes when phasing families.

2.7 Supporting tools

Beyond phasing, WhatsHap comes with a collection of other functionalities to handle phasing data. These functionalities are available as sub-commands, allowing to conveniently combine them into larger analysis pipelines. Available sub-commands include "unphase" to remove all phasing information present in a VCF file; "stats" to compute statistics on a phased VCF such as the number of (un)phased variants, number of phased blocks, and minimum/maximum/average block size; and "compare" to compare two or more phased VCFs with respect to pair-wise switch/flip errors, but also with respect to a simultaneous comparison of multiple phasings. The sub-command "haplotag" uses a phased VCF to add tags to each read in a BAM file that indicate which haplotype it belongs to. This allows further processing, such as splitting the reads into two BAM files according to haplotype, or to color-code reads by haplotype in a genome browser (as for example seen in Figure 1).

3 Results

3.1 Experimental setup

We evaluate WhatsHap and other phasing tools on both real and simulated datasets. The setup we explain in the following is similar to the one described earlier [5].

Real data. As evaluation sample, we chose the child (NA24385) of the Ashkenazim trio sequenced by the Genome in a Bottle Consortium (GIAB) [26]. We downloaded a consensus genotype call set (NIST_CallsIn2Technologies_05182015) for chromosome 1, containing 48 023 heterozygous SNVs and 4 265 heterozygous non-SNVs. We refer to these variants as *ground truth unphased variants*. We phased all these variants with the population-based phasing tool SHAPEIT (v2-r837) [27] with default parameters, using the reference panel provided by Phase 3 of the 1000 Genomes Project. We refer to them as *ground truth phased variants*. Furthermore, we use the Pacific Biosciences (PacBio) reads from GIAB aligned to chromosome 1. They have an average coverage of 60.2× and average mapped read-length of 8 687 bp. To analyze the effect of sequencing depth on phasing performance, we created additional datasets by downsampling the aligned reads to average coverages of 2×, 3×, 4×, 5×, 10× and 15×.

Simulated data. While population-based phasing is very accurate, it has limitations with respect to low-frequency variants. Since our ground truth phasing will therefore not be completely error free, we complement our evaluation with a simulation study. To this end, we generated two haplotypes of a "virtual child" by recombining the haplotypes of the parents NA24143 and NA24149 *in silico* as described previously [5]. The resulting virtual child has 45 625 heterozygous variants

with known phasing (ground truth), containing 42 062 SNVs and 3 563 non-SNVs. Next, we incorporated these true variants into the GRCh37 reference genome to obtain two FASTA files representing the true haplotypes. We simulated reads that mimic the read length profile and coverage of the real data using pbsim [28]. The resulting reads were then aligned to GRCh37 using BWA-MEM (0.7.12-r1039) [29]. Similar to the real dataset, we further downsampled the aligned reads to average coverages of 2×, 3×, 4×, 5×, 10×, and 15×.

3.2 Tools

We compared WhatsHap to other phasing tools that we were able to get to work with a reasonable effort of trying, which forced us to omit some tools. We settled on the ReadBackedPhasing component of the Genome Analysis Toolkit (GATK) [20]), hapCUT [13] and phASER [21]. We used ReadBackedPhasing as distributed with GATK v3.5-0-g36282e4, Git revision 844af08c of hapCUT (reporting as version 0.7), revision 8608e1d of phASER (version 0.8) with a fix for CIGAR hard-clipping available in pull request #14, and revision 09326a5 (version 0.12) of WhatsHap.

ReadBackedPhasing failed to phase any variants in PacBio data with default settings, and there is no official recommendation for the correct settings[1]. We therefore tried 144 parameter combinations on a subset of the data and determined that using options `--phaseQualityThresh 1` and `--min_base_quality_score 1` gives the best results.

Since hapCUT does not produce VCF, but a custom output format, we implemented a WhatsHap subcommand "`hapcut2vcf`" that converts the custom format to VCF. We did not use hapCUT's `--longreads` option since the help states that it should be set if the "program uses too much memory", which was not the case.

We ran all programs both with and without requesting phasing of non-SNVs (mostly indels), except ReadBackedPhasing, which does not support indel phasing. WhatsHap was additionally run in a non-realignment mode, in which it also does not support non-SNV phasing.

3.3 Measuring performance

We describe here performance metrics that we use to evaluate phasing accuracy and completeness of the different tools.

For phasing completeness, we need to consider that read-based requires reads covering two heterozygous variants. The result is therefore a partition of the list of variants into *haplotype blocks*, within which all of the variants are phased relative to each other. In the ideal case, these blocks would correspond to connected components of a connectivity graph, which is a graph with heterozygous variants as the nodes and where an edge is drawn between two variants if a read exists that covers both. In practice, different tools may choose to not use some reads or may have different notions of connectivity and will therefore output haplotype blocks that are more fragmented. Note that haplotype blocks are not necessarily contiguous. When paired-end or mate-pair reads are involved, they may not connect adjacent variants and may even be nested or interleaved. A haplotype block with only a single variant is a *singleton*.

[1] http://gatkforums.broadinstitute.org/gatk/discussion/7129/

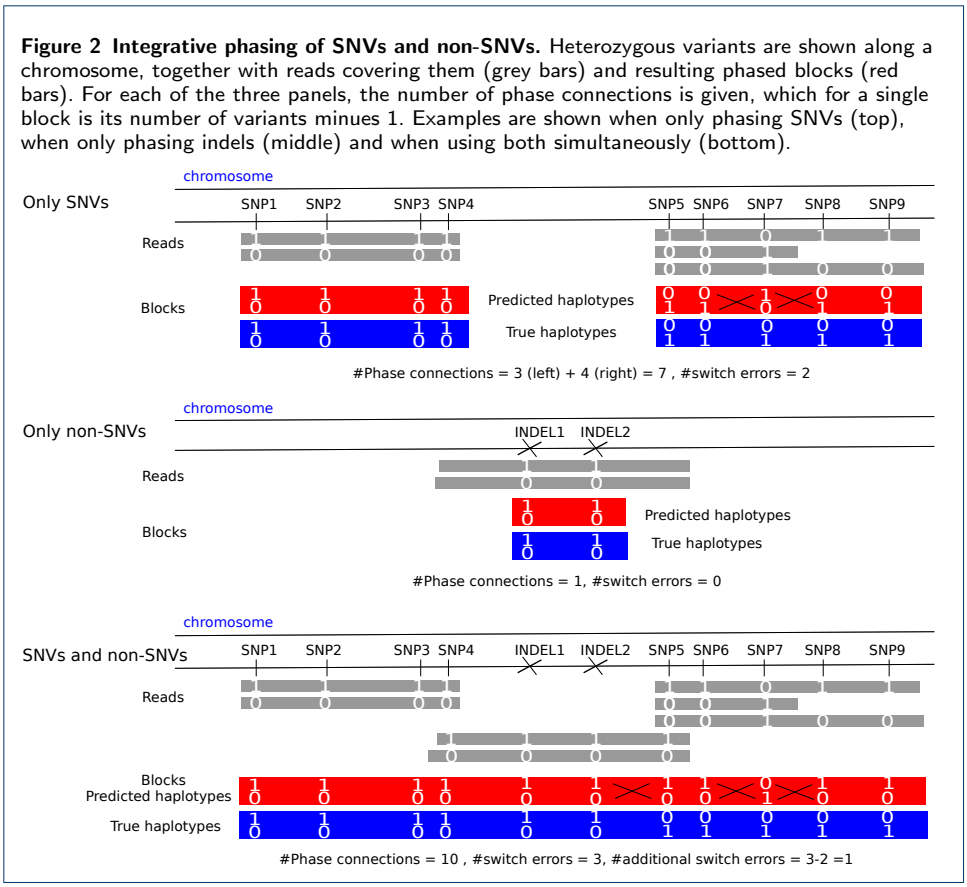


Figure 2 Integrative phasing of SNVs and non-SNVs. Heterozygous variants are shown along a chromosome, together with reads covering them (grey bars) and resulting phased blocks (red bars). For each of the three panels, the number of phase connections is given, which for a single block is its number of variants minues 1. Examples are shown when only phasing SNVs (top), when only phasing indels (middle) and when using both simultaneously (bottom).

In a VCF file, haplotype blocks are called *phase sets* and have a unique identifier, which is attached to a genotype using the `PS` tag.

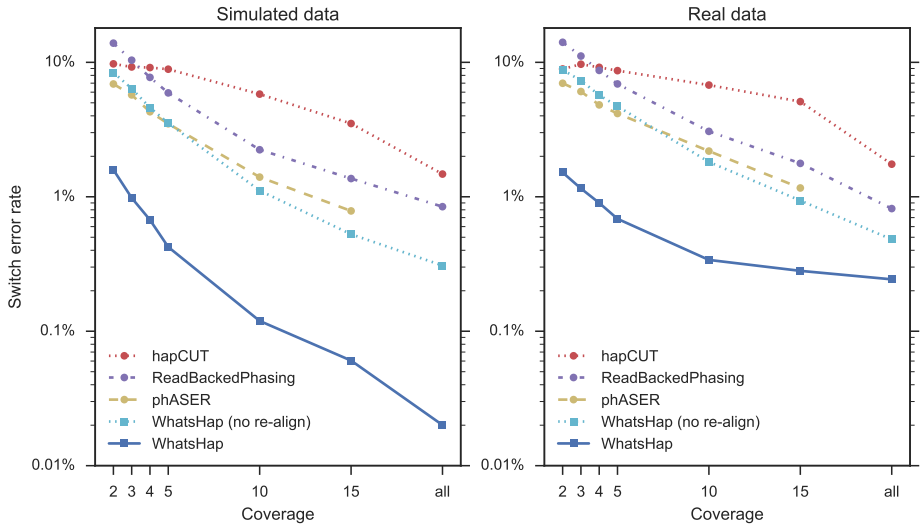
In a block of size k, we say that there are $k-1$ *phase connections* between adjacent variants, assuming the variants are sorted by position. In our evaluation, we count phase connections because they take the block structure into account; the number of phase connections is equal to the number of phased variants minus the number of blocks.

In our results, we will give the *phase connection ratio*, which is the total number of phase connections divided by the possible number of phase connections ($n-1$ for a chromosome with n heterozygous variants).

Phase can be seen as a relationship between variants along a phase connection: It can be either *cis*, meaning that the alleles are on the same haplotype or *trans* if they are on different haplotypes.

Errors within phased blocks are counted in terms of *switch errors*, which occur when the meaning of cis/trans is reversed from one point on forward within a block. Switch errors are calculated by traversing the haplotype blocks from left to right and computing the number of times a jump is needed from one haplotype to the other in order to reconstruct the true haplotype. Note that flip errors (an isolated variant assigned to the wrong haplotype) are counted as two switch errors in this model. For the *phasing error rate*, we sum the switch errors over all blocks and divide by the total number of phase connections. Switch errors are detected by comparison with the ground truth phasing (SHAPEIT result or simulated known phasing).

Figure 3 Phasing switch errors on simulated and real PacBio data, SNVs only. All tools give fewer errors with increasing coverage. phASER at full coverage is not shown since the run did not complete within 24 hours. Note logarithmic scaling of the y axis, which emphasizes the difficulty of lowering the error rate the closer it already is to zero.



3.4 Results on SNVs only

As one would expect, the five evaluated algorithms consistently perform better when coverage increases. For both simulated and real data, switch error rate decreases (Figure 3) and the phase connection ratio increases (Figure 4).

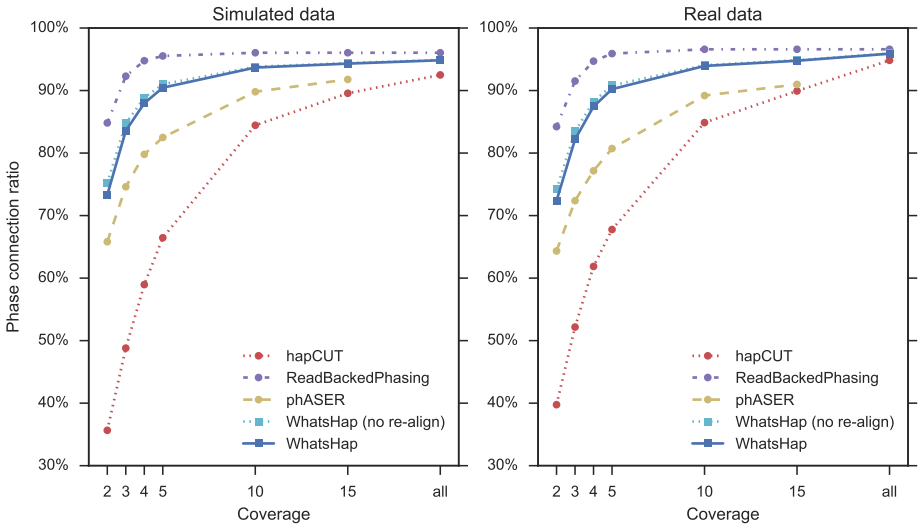
WhatsHap with the re-alignment algorithm is clearly the tool with the lowest switch error rate at all coverages. In real data and at full coverage, it achieves a switch error rate of 0.24%. The second best is WhatsHap without realignment at 0.48% and full coverage. While the phASER run for full coverage did not finish within 24 hours, the trend shows that phASER could possibly also achieve a rate of around 0.5% in this data set.

For simulated data, the shapes of curves and the resulting ranking of tools is identical, suggesting that the simulation pipeline mimics realistic conditions. WhatsHap achieves an extremely low error rate of 0.02%, corresponding to just eight switch errors at full coverage. Second best is WhatsHap without re-alignment at 0.3% switch error rate in simulated data. Again, phASER might have been able to achieve similar rates if we had waited longer than 24 hours.

Of note is that the switch error rate curves look nearly identical in simulated and real data, except that they are shifted by one order of magnitude towards higher errors in real data. Recall that phasing errors in real data are computed by comparison to the phasing made by SHAPEIT. Statistical phasing is not perfect and makes errors at rare and *de novo* variants. Therefore, some "baseline" errors remaining for all tools in real data may actually be explained by errors made by SHAPEIT.

Figure 4 shows how many phase connections the tools were able to make. All tools, as expected, connect more variants when coverage increases. At first sight, ReadBackedPhasing seems to be the winner as it connects more variants than the other

Figure 4 Completeness of phasing on simulated and real PacBio data, SNVs only. All tools make more phase connections as coverage increases. phASER at full coverage is not shown since the run did not complete within 24 hours. The results for ReadBackedPhasing are higher than theoretically possible, see main text.



tools. Already at coverage 5, it connects ca. 95% of all variants in both real and simulated data. However, when we inspected the haplotype blocks it outputs together with the reads and variants in IGV, we found examples where ReadBackedPhasing put variants into the same phased block that clearly had no direct or indirect connection by any reads. We can only presume that ReadBackedPhasing sometimes connects variants that are not connected by reads.

Another indication that the results are unrealistic for a purely read-based phaser is that the haplotype blocks that WhatsHap outputs are actually the connected components of the selected reads. Up to an average coverage of around 15, when WhatsHap keeps nearly all reads, the WhatsHap curve therefore represents the best possible phase connection ratio given the reads. Since ReadBackedPhasing shows a higher ratio, it clearly obtains haplotype blocks that are larger than theoretically possible, obtained at the expense of a higher switch error rate.

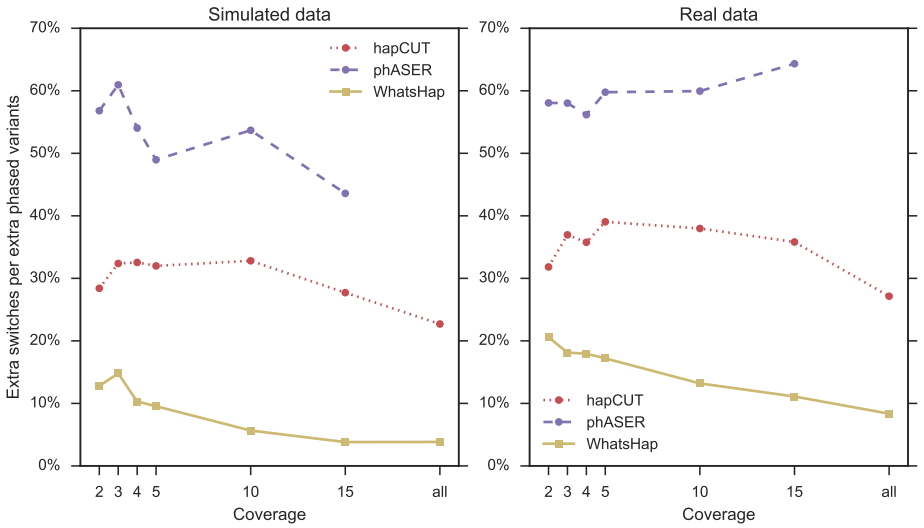
Of all the tools, the two WhatsHap algorithms come therefore closest to the theoretically best possible phase connection ratio.

3.5 Results on non-SNVs

WhatsHap, hapCUT and phASER are able to phase insertions and deletions. WhatsHap additionally phases "complex" variants such as `CGA` → `CTCC` that are neither SNV, insertion nor deletion (WhatsHap does not phase structural variations).

To assess how the three tools handle non-SNVs, we compare the results for phasing only SNVs to the results for jointly phasing both SNVs and non-SNVs. Figure 6 illustrates that we not only expect to see additionally phased non-SNVs, but also an increase in the number of phased SNVs as previously unconnected blocks can become connected due to reads that cover SNVs as well as non-SNVs. Note that

Figure 5 Additional switch errors when phasing non-SNVs. Enabling phasing of non-SNVs, the number of phased variants increases (Δp), but so does the number of switch errors (Δe). The y axis shows $\frac{\Delta p}{\Delta e}$.



phasing only non-SNVs is not useful for assessing performance as this will lead to an extremely fragmented phasing (even long reads rarely cover more than one non-SNV).

When phasing of non-SNVs is enabled, the number of phased variants increases (Δp), but so does the number of switch errors (Δe). We define the *switch error rate over additionally phased variants* as the number of additional switch errors divided by the number of additionally phased variants, $\frac{\Delta p}{\Delta e}$. The quantity can be interpreted similar to a phasing error rate, but it takes into account not only switch errors among the phased non-SNVs, but even additional errors among SNVs. Also, since Δe explicitly includes SNVs, any loss of previously phased SNVs is reflected by an increases of ratio. For example, a rate of 10% means that ten additionally phased variants come at a price of one additional switch error.

As we can see in Figure 5, the switch error rate over additionally phased variants is in general much higher than the switch error rate in the SNV-only case. Regardless of coverage, HapCUT and phASER are at a rate of over 20%. That is, every fifth additionally phased variant introduces an additional switch error.

WhatsHap is clearly the best of the tools, achieving a rate of 3.8% in simulated data and 8.3% in real data at full coverage. As mentioned before, part of the errors in real data might be explained by errors in the ground truth phasing. However, even on simulated date the switch error rates in the SNV-only case are at least an order of magnitude lower, which indicates that phasing of non-SNVs is more difficult than phasing SNVs.

Finally, Figure 6 shows how many non-SNVs could actually be phased by the programs when instructed to do so. WhatsHap is again the best choice at any coverage. At full coverage, WhatsHap is able to phase 97% of all non-SNVs in both simulated and real data, while none of the other tools reach more than 90% phased non-SNVs.

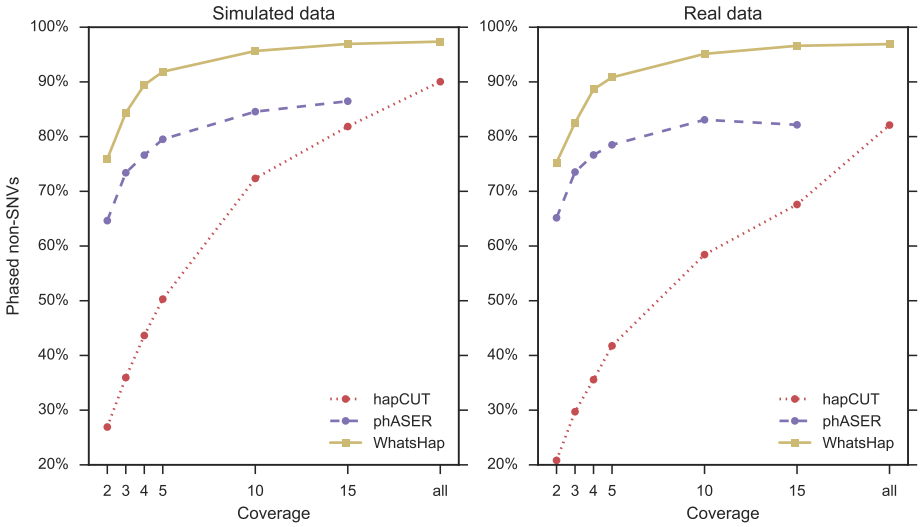


Figure 6 Fraction of non-SNVs that can be phased. The y axis shows how many non-SNVs were phased by each relative to the total number of heterozygous non-SNVs in the dataset.

Table 1 Running time of phasing tools on human chromosome 1, measured on the *artificial child* dataset. Values given are in CPU seconds, actual (wall-clock) times are higher due to I/O.

Coverage	2×	3×	4×	5×	10×	15×	all
WhatsHap	75	95	119	142	323	630	969
WhatsHap (no re-align)	163	223	222	353	770	1249	2028
hapCUT	75	122	155	186	284	400	543
ReadBackedPhasing	352	481	573	655	967	1375	2044
phASER	1356	1806	2082	2056	2838	3245	4302

3.6 Runtime

We measured running time of all tools on the *artificial child* dataset, chromosome 1. To be able to include all tools, only SNVs were phased. For those tools supporting non-SNV phasing, the rankings are identical with only slightly higher runtimes. Results can be seen in Table 1.

WhatsHap's runtime is clearly competitive with those of the other tools. At all coverages, WhatsHap or hapCUT is the fastest tool. HapCUT has an advantage at high coverages, while WhatsHap is faster at low coverages. WhatsHap runtimes in "no re-alignment" mode are also given for completeness and due to it being slower than the re-aligment variant of the algorithm, we can see that the additional optimization efforts that went into the algorithm have paid off.

The two other tools hapCUT and phASER consistently have the highest runtimes.

4 Discussion and Conclusions

The benefit of using long reads for the purpose of phasing is obvious, but it is hard to actually achieve the benefits using existing tools which have not been designed with error-prone long reads in mind. WhatsHap is a production-ready tool for read-based phasing that shows superior performance at any coverage level, in particular with long reads. Its phasing results are both the most complete and the most accurate compared to the other tested tools. While non-SNVs still present a challenge for

phasing tools, WhatsHap even here outperforms other tools in terms of both phasing completeness and accuracy.

WhatsHap's core algorithm is based on a sound theoretical framework [4] that delivers optimal solutions to the wMEC or PedMEC problem. In terms of accuracy, the results are therefore optimal for any algorithm based on the wMEC formulation. The FPT formulation with coverage as parameter in combination with a heuristic read selection algorithm results in an implementation that is fast in practice, and indeed we see that runtimes are dominated by I/O operations, not by the phasing algorithm itself.

The program's ability to integrate read-based and pedigree phasing is unique and is the perfect match for studies in which family trios are sequenced. As shown earlier [5], the phasing results obtained in such a setting outperform non-integrated approaches, while reducing the needed overall coverage to as low as 2×.

While our perception may be biased, we also consider WhatsHap to be the tool that is easiest to use. Options are set to sensible defaults and typically do not need to be changed by the user. Input formats are recognized automatically, and output is standards-compliant VCF, ready to be consumed by other tools. The ability of using phased VCFs as input allows interesting use cases such as "merging" phasing results. In fact, reads from any sequencing technology and previously-phased data are supported, allowing to combine data from Illumina (paired-end and mate-pairs), PacBio, Oxford Nanopore, 10X Genomics at will.

WhatsHap even comes with a set of extra tools that help in analyzing phasing results, such as obtaining statistics from phased VCFs or comparing them.

4.1 Limitations

WhatsHap ignores supplementary alignments. These occur when a read aligns to sufficently different parts of the reference such that the alignment needs to be split up into multiple parts. With reference sequences that have lower quality than the human one or that are for a more diverse species, this is more likely to happen.

4.2 Future work

WhatsHap is improved continually, especially in terms of performance, speed and the ability to work with a wide range of types of input data. For example, we have nearly finished work on a pure genetic phasing mode in which no reads are required as input if a pedigree of samples is available. More challenging will be to integrate read-based and population-based phasing. In the same way that genetic phasing helps to bridge regions insufficiently covered by reads in related samples, population-based phasing can help in the same way even in single samples. Over time, we predict that phasing tools will necessarily have to focus more on these integrative approaches that make best use of all available data.

4.3 Software availability

WhatsHap is available as Open Source software under the MIT license. The home page at `https://whatshap.readthedocs.io/` contains documentation and installation instructions. Supported operating systems include Linux and OS X. Installation requires Python 3.3 or later and a working C/C++ compiler.

Competing interests

The authors declare that they have no competing interests.

Acknowledgements

MM is supported by a grant from the Swedish Knut and Alice Wallenberg Foundation to the Wallenberg Advanced Bioinformatics Infrastructure.

Author details

[1]Science for Life Laboratory, Dept of Biochemistry and Biophysics, Stockholm University, Box 1031, SE-17121 Solna, Sweden. [2]Laboratoire de Biométrie et Biologie Évolutive (LBBE : UMR CNRS 5558), Université de Lyon 1, , Villeurbanne, France. [3]Department of Computer Science, University of Pisa, Italy. [4]Life Sciences Group, Centrum Wiskunde & Informatica (CWI), Science Park 123 1098 XG Amsterdam, The Netherlands. [5]VU University Amsterdam, The Netherlands. [6]Center for Bioinformatics, Saarland University, Campus E2.1, 66123, Saarbrücken, Germany. [7]Max Planck Institute for Informatics, Saarbrücken, Germany.

References

1. Browning, S.R., Browning, B.L.: Haplotype phasing: existing methods and new developments. Nature Reviews Genetics **12**(10), 703–714 (2011). doi:10.1038/nrg3054
2. Tewhey, R., Bansal, V., Torkamani, A., Topol, E.J., Schork, N.J.: The importance of phase information for human genomics. Nature Reviews Genetics **12**(3), 215–223 (2011). doi:10.1038/nrg2950
3. Glusman, G., Cox, H.C., Roach, J.C.: Whole-genome haplotyping approaches and genomic medicine. Genome Medicine **6**(9), 73 (2014). doi:10.1186/s13073-014-0073-7
4. Patterson, M., Marschall, T., Pisanti, N., van Iersel, L., Stougie, L., Klau, G.W., Schönhuth, A.: WhatsHap: Weighted haplotype assembly for future-generation sequencing reads. Journal of Computational Biology **22**(6), 498–509 (2015). doi:10.1089/cmb.2014.0157
5. Garg, S., Martin, M., Marschall, T.: Read-based phasing of related individuals. Bioinformatics (Proceedings of ISMB) **32**, 234–242 (2016). doi:10.1093/bioinformatics/btw276
6. Delaneau, O., Marchini, J., Zagury, J.-F.: A linear complexity phasing method for thousands of genomes. Nature Methods **9**(2), 179–181 (2012). doi:10.1038/nmeth.1785
7. Abecasis, G.R., Cherny, S.S., Cookson, W.O., Cardon, L.R.: Merlin–rapid analysis of dense genetic maps using sparse gene flow trees. Nature Genetics **30**, 97–101 (2002). doi:10.1038/ng786
8. Ip, C.L., Loose, M., Tyson, J.R., de Cesare, M., Brown, B.L., et al.: MinION analysis and reference consortium: Phase 1 data release and analysis. F1000 Research **4** (2015). doi:10.12688/f1000research.7201.1
9. Kuleshov, V., Xie, D., Chen, R., Pushkarev, D., Ma, Z., Blauwkamp, T., Kertesz, M., Snyder, M.: Whole-genome haplotyping using long reads and statistical methods. Nature Biotechnology **32**(3), 261–266 (2014). doi:10.1038/nbt.2833
10. Roberts, R.J., Carneiro, M.O., Schatz, M.C.: The advantages of SMRT sequencing. Genome Biology **14**, 405 (2013). doi:10.1186/gb-2013-14-7-405
11. Schwartz, R.: Theory and algorithms for the haplotype assembly problem. Communications in Information & Systems **10**(1), 23–38 (2010)
12. Lippert, R., Schwartz, R., Lancia, G., Istrail, S.: Algorithmic strategies for the single nucleotide polymorphism haplotype assembly problem. Briefings in Bioinformatics **3**(1), 23–31 (2002). doi:10.1093/bib/3.1.23. http://bib.oxfordjournals.org/content/3/1/23.full.pdf+html
13. Bansal, V., Bafna, V.: HapCUT: an efficient and accurate algorithm for the haplotype assembly problem. Bioinformatics **24**(16), 153–159 (2008). doi:10.1093/bioinformatics/btn298
14. Levy, S., Sutton, G., Ng, P.C., Feuk, L., Halpern, A.L., Walenz, B.P., Axelrod, N., Huang, J., Kirkness, E.F., Denisov, G., Lin, Y., MacDonald, J.R., Pang, A.W.C., Shago, M., Stockwell, T.B., Tsiamouri, A., Bafna, V., Bansal, V., Kravitz, S.A., Busam, D.A., Beeson, K.Y., McIntosh, T.C., Remington, K.A., Abril, J.F., Gill, J., Borman, J., Rogers, Y.-H., Frazier, M.E., Scherer, S.W., Strausberg, R.L., Venter, J.C.: The diploid genome sequence of an individual human. PLoS Biol **5**(10), 254 (2007). doi:10.1371/journal.pbio.0050254
15. Mazrouee, S., Wang, W.: FastHap: fast and accurate single individual haplotype reconstruction using fuzzy conflict graphs. Bioinformatics **30**(17), 371–378 (2014). doi:10.1093/bioinformatics/btu442
16. Chen, Z.-Z., Deng, F., Wang, L.: Exact algorithms for haplotype assembly from whole-genome sequence data. Bioinformatics **29**(16), 1938–1945 (2013). doi:10.1093/bioinformatics/btt349
17. Deng, F., Cui, W., Wang, L.: A highly accurate heuristic algorithm for the haplotype assembly problem. BMC Genomics **14**(Suppl 2), 2 (2013). doi:10.1186/1471-2164-14-S2-S2
18. Fouilhoux, P., Mahjoub, A.R.: Solving VLSI design and DNA sequencing problems using bipartization of graphs. Computational Optimization and Applications **51**(2), 749–781 (2012). doi:10.1007/s10589-010-9355-1
19. He, D., Choi, A., Pipatsrisawat, K., Darwiche, A., Eskin, E.: Optimal algorithms for haplotype assembly from whole-genome sequence data. Bioinformatics **26**(12), 183–190 (2010). doi:10.1093/bioinformatics/btq215
20. DePristo, M.A., Banks, E., Poplin, R., Garimella, K.V., Maguire, J.R., Hartl, C., Philippakis, A.A., Angel, G.D., Rivas, M.A., Hanna, M., McKenna, A., Fennell, T.J., Kernytsky, A.M., Sivachenko, A.Y., Cibulskis, K., Gabriel, S.B., Altshuler, D., Daly, M.J.: A framework for variation discovery and genotyping using next-generation DNA sequencing data. Nature Genetics **43**(5), 491–498 (2011). doi:10.1038/ng.806
21. Castel, S.E., Mohammadi, P., Chung, W.K., Shen, Y., Lappalainen, T.: Rare variant phasing and haplotypic expression from RNA sequencing with phASER. Nature Communications **7**, 12817 (2016). doi:10.1038/ncomms12817
22. Fischer, S.O., Marschall, T.: Selecting Reads for Haplotype Assembly. doi:10.1101/046771
23. Patterson, M., Marschall, T., Pisanti, N., Iersel, L.v., Stougie, L., Klau, G.W., Schönhuth, A.: WhatsHap: Haplotype assembly for future-generation sequencing reads. In: Sharan, R. (ed.) Proceedings of the 18th Annual International Conference on Research in Computational Molecular Biology (RECOMB). Lecture Notes in Computer Science, pp. 237–249. Springer, ??? (2014). doi:10.1089/cmb.2014.0157

24. Garrison, E., Marth, G.: Haplotype-based variant detection from short-read sequencing. ArXiv e-prints (2012). 1207.3907
25. Shirley, M.D., Ma, Z., Pedersen, B.S., Wheelan, S.J.: Efficient "pythonic" access to FASTA files using pyfaidx. PeerJ PrePrints **3**, 1196 (2015). doi:10.7287/peerj.preprints.970v1
26. Zook, J.M., Chapman, B., Wang, J., Mittelman, D., Hofmann, O., Hide, W., Salit, M.: Integrating human sequence data sets provides a resource of benchmark SNP and indel genotype calls. Nat Biotechnol **32**(3), 246–251 (2014). doi:10.1038/nbt.2835
27. Delaneau, O., Marchini, J., The 1000 Genomes Project Consortium: Integrating sequence and array data to create an improved 1000 genomes project haplotype reference panel. Nature Communications **5** (2014). doi:10.1038/ncomms4934
28. Ono, Y., Asai, K., Hamada, M.: PBSIM: PacBio reads simulator–toward accurate genome assembly. Bioinformatics **29**(1), 119–121 (2013). doi:10.1093/bioinformatics/bts649
29. Li, H.: Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM. arXiv preprint arXiv:1303.3997 (2013)