2020

Creating an ensemble method combining haplotype estimates

MATHIJS VAN KOOTEN

SUPERVISORS: DICK DE RIDDER, SINA MAJIDIAN

Table of contents

Abstract

Haplotypes describe alleles that are co-located on the same chromosome and inherited together. They are a major factor in explaining inheritance and expression patterns. The haplotypes of an individual can be estimates via high throughput sequencing from which SNPs are identified. These SNPs are then treated as one big puzzle, fitting SNPs together on so called haplotype blocks, with the least amount of errors. Inferring an individual's haplotype in a feasible manner is possible for diploid organisms but remains a challenge in polyploids as the assignment of SNPs to their respective haplotype becomes much more challenging given the increasing number of possible combinations. For haplotyping polyploids, many tools have been released, all applying a different approach to the puzzle. Here, we show the potential improvement that can be made by combining haplotype estimates made by different tools, as well as applying three approaches of doing so. Our results show that SNP positions with an error in any of their phasings typically are not shared among the estimates of the tested tools. Furthermore, the ensemble approaches we have tested show the need to apply a method that does not treat each allele of each haplotype independently. Ultimately, we suggest to expand the input features for a machine learning approach by using the reads and VCF files that are also used by the haplotyping tools themselves.

1. Introduction

Since the advent of high throughput sequencing, the DNA of many species has been sequenced and their genomes have been analysed. This made it possible to analyse organisms on a more fundamental level: traits can be linked to specific genomic regions; gene expression can be predicted and genetic differences within and between species can be compared. A major factor in explaining inheritance and expression patterns are haplotypes, describing alleles that are co-located on the same chromosome and inherited together. What distinguishes alleles on different haplotypes are single nucleotide polymorphisms (SNPs), i.e. single DNA positions with a differing nucleotide. Using the human reference genome, the HapMap consortium created a comprehensive library of SNPs within the human population (Gibbs et al., 2003), identifying variations that allows for identification of genes and genetic variations affecting disease. As haplotypes describe the co-location of one's alleles, they can be used to predict inheritance patterns, as well as explain gene expression patterns and likelihood of genetic disease (Roe et al., 2017). Considering the average human gene size lies between 10 and 15 kbp (Sutherland, 1996), obtaining a haplotype assembly that spans a large distance on the genome, spanning many alleles with few errors, is of interest.

There are various techniques to detect SNPs and to estimate whether they are located on the same haplotype, so-called phasing. SNP microarrays are widely used and can detect variants via probes that target various known SNPs. Although cheap in use, they require a known sequence and SNP location however, making them expensive to develop. In addition, SNP arrays generally give unphased genotypes (Browning & Browning, 2011). With the advances in high throughput sequencing however, it has become feasible to generate an individual's genome and infer the haplotypes contained herein. For inferring one's haplotype, multiple methods exist. The most straightforward method, in principle, is to physically sequence a single haplotype. One can do so by separating the chromosomes (Luo et al., 2018), or by using haploid cells. Another relatively straightforward way to obtain a haplotype assembly is long read sequencing (Roe et al., 2017), in which reads span large stretches of a haplotype, containing many variants. In practice however, separating chromosomes is a laborious task, requiring specialised devices; and long read sequencing techniques, currently have a high error rate (Weirather et al., 2017) and relatively low throughput. As such, short-read sequencing is still typically used for haplotype phasing.

Phasing using sequencing reads poses a challenge, as the sequencing strategy involves the full genome, including multiple haplotypes, in a single run. As such, in order to make out the haplotypes afterwards, various algorithms can be used to assign alleles to a specific haplotype (Browning & Browning, 2011). A major drawback of this differentiation is that seeking out a solution with as few flips as possible, using the so-called minimum error correction (MEC), is NP-hard (Cilibrasi, van Iersel, Kelk, & Tromp, 2007). As MEC is NP-hard, many haplotype assemblers have been made, each with its own approximation method and accompanying error rate.

Using sequencing reads to differentiate haplotypes from a single sequencing run poses an issue. In a diploid organism, most SNPs are considered to consist of two variant nucleotides, known as the bi-allelic assumption. It seems clear that in bi-allelic SNPs, the differing nucleotides lie on either haplotype and are easily recognised as such by comparison to the reference genome. In a polyploid genome however, SNPs can consist of all four nucleotides, spread across a number of haplotypes. This means that multiple SNPs are required in order to determine which haplotype the region should be assigned to (Aguiar & Istrail, 2013). Further complicating matters are base calling errors during sequencing, potentially leading to erroneous haplotype assembly (Motazedi, Finkers, Maliepaard, & de Ridder, 2018). To estimate the haplotypes from sequencing reads, computational methods are required (Browning & Browning, 2011). There are three main haplotyping optimisation strategies:

minimum error correction (MEC), minimum SNP removal (MSR) and minimum fragment removal (MFR). These strategies try to assemble the reads into a specified number of haplotypes using as few mutations as possible. As their names imply, MEC tries to flip bases in the reads in order for them to correspond to a haplotype; MSR, instead of flipping bases removes entire SNP positions that are not in correspondence; and MFR removes entire reads with a non-corresponding SNP.

As different haplotype approximation methods can result in different estimates, these various solutions potentially complement each other. Combining these outputs therefore holds potential to result in a haplotype estimate that contains fewer errors. Here, we present a method to combine the output of a variety of haplotype assemblers using short read data. Decisions regarding what output to use will be based on each tool's performance in terms of haplotyping errors.

2. Methods

For this project, we set out to create a more accurate haplotype estimate, by combining haplotype estimates of multiple haplotyping tools. We first created the haplotypes themselves *in silico* out of a 30kb reference sequence by generating three copies while introducing SNPs, to gain a triploid ground truth. Then, we simulated reads for this triploid and created a variant calling file via FreeBayes (Garrison & Marth, 2012). Another vcf file was generated directly from the ground truth. These vcf files were then used as input to the tested haplotyping tools whose estimates were used in both performance assessment and multiple ensemble methods. Three ensemble methods were applied: a majority vote combination and two random forest models using a different output format. Multiple haplotypes were generated to both gain a sufficiently sized dataset for the ensemble approaches, as well as assess the variability in estimate performance.

1. Haplotype & read simulation

In order to measure the accuracy of haplotype estimates, a comparison to the ground truth haplotype was made. For this, we use simulated haplotypes based on chromosome 1 of *Solanum tuberosum* genome version 3.0, downloaded from EnsemblPlants version 42. Specifically, the 30kb region downstream from 6Mb was used for its absence of unknown nucleotides. This region was used for all simulations, as haplotypes are defined by SNPs and thus the specific underlying sequence has no impact. Using this 30kb region, a triploid was simulated via Haplogenerator, part of haplosim (Motazedi, 2016) by creating three copies of the reference and introducing SNPs at random positions. Triploids were generated at a SNP rate of 1 percent.

Next, reads were simulated using ART Illumina (Huang, Li, Myers, & Marth, 2011). In order to simulate a single sequencing run, the three separate FASTA files of the simulated triploid genome were concatenated. Paired-end Mi-Seq reads were simulated at 20x coverage, insert sizes of 800bp, and read lengths of 250bp.

2. SNP calling & pre-processing

Input for the haplotyping tools consists of a variant call format (VCF) which contains SNP calls and their genotypes. As the ground truth is known in our simulations, two methods were applied to gain these vcf files. First, a vcf file was created from the ground truth haplotypes via the hap2vcf python script, part of 10xpipeline (Majidian, n.d.-a). This script takes the haplotypes contained in the ground truth that was generated by Haplosim, and converts it to a vcf file with all SNP positions having the same quality scores.

The second method uses the simulated reads and FreeBayes (version 1.3.1) to accomplish the generation of a vcf file. FreeBayes was run with a specified ploidy of three. Before submitting this vcf file to the haplotyping tools, MNPs in it were broken up into individual SNPs using the break_vcf script, part of hap10 (Majidian, n.d.-b), as the tested haplotyping tools would otherwise ignore these features entirely.

3. Haplotype estimation

Using the vcf file, three tools were used to estimate the haplotypes, Haptree (Berger, Yorukoglu, Peng, & Berger, 2014) (v0.1), HapCompass (Aguiar & Istrail, 2012) (v0.8.2) and SDhaP (Das & Vikalo, 2015) (poly, 2016/06/03). As HapCompass is able to use the vcf file without further modification, no further modifications were made to its input. To use the vcf in HapTree and SDhaP, it has to be converted into a fragment file. For this conversion we used HapCut2 HAIRS (Edge, Bafna, & Bansal, 2017). As HAIRS only works on diploids, the vcf was modified to mimic a diploid. These modifications were executed via an awk command (Appendix 4). In addition, Haplosim fragmentpoly was used to convert the HAIRS output to both HapTree and SDhaP specific input formats.

4. Ensemble preparation & methods

For combining the haplotype estimates by the three tools tested, we applied both a majority vote, and a random forest machine learning classification method. Before either of these are applied, as the alleles in the haplotype estimates are only phased together within blocks of each estimate, it is required to figure out the corresponding haplotypes across the tools, so that each of our input features represents the same haplotype. This was done by reordering the output of each tool such that each column between tools represents the same haplotype. This reordering was executed via a python script (Appendix 1) that finds two corresponding haplotypes by determining which combination has the fewest differences. As the ground truth is known, we applied two approaches regarding the determination of the reference haplotype. The first method is to always take the ground truth as reference and reorder all tool estimates to it (GT reordering). The second, more realistic approach without using the ground truth, finds the best combination between two tool estimates and reorders accordingly (DN reordering). The reference in this case is determined by a set of rules, illustrated in Figure 2, that ensure that each haplotype block is only reordered once and used as a reference for as long as possible, in order to minimise potential errors in the reordering.

When using the tool estimates to determine corresponding haplotypes, we are also able to merge certain blocks. This is possible when SNPs between estimates are phased together in different blocks, meaning these SNPs can be treated as an overlap between the blocks, finding corresponding haplotypes between them, and subsequently treated as one, as is illustrated in Figure 1.

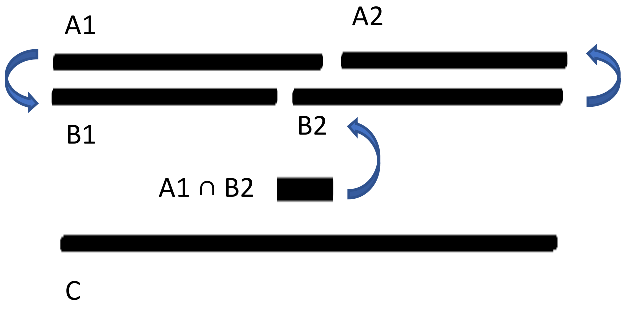


Figure 1: Illustration of a de novo reordering, including a merger of blocks. Here, A1-2 and B1-2 represent two blocks from tools A and B respectively. C represents the combined block in the majority vote. Block C is started at a starting position where no tool blocks are reordered. Then, following the rules of selecting the reference block, A1 is used to reorder B1. After this, the next starting position is that of B2, at which position, A1 is already reordered and used to reorder B2 via the appropriate overlap. The same applies from B2 to A2. For C, this means only one block is created, effectively merging the separate blocks of the tool estimates.

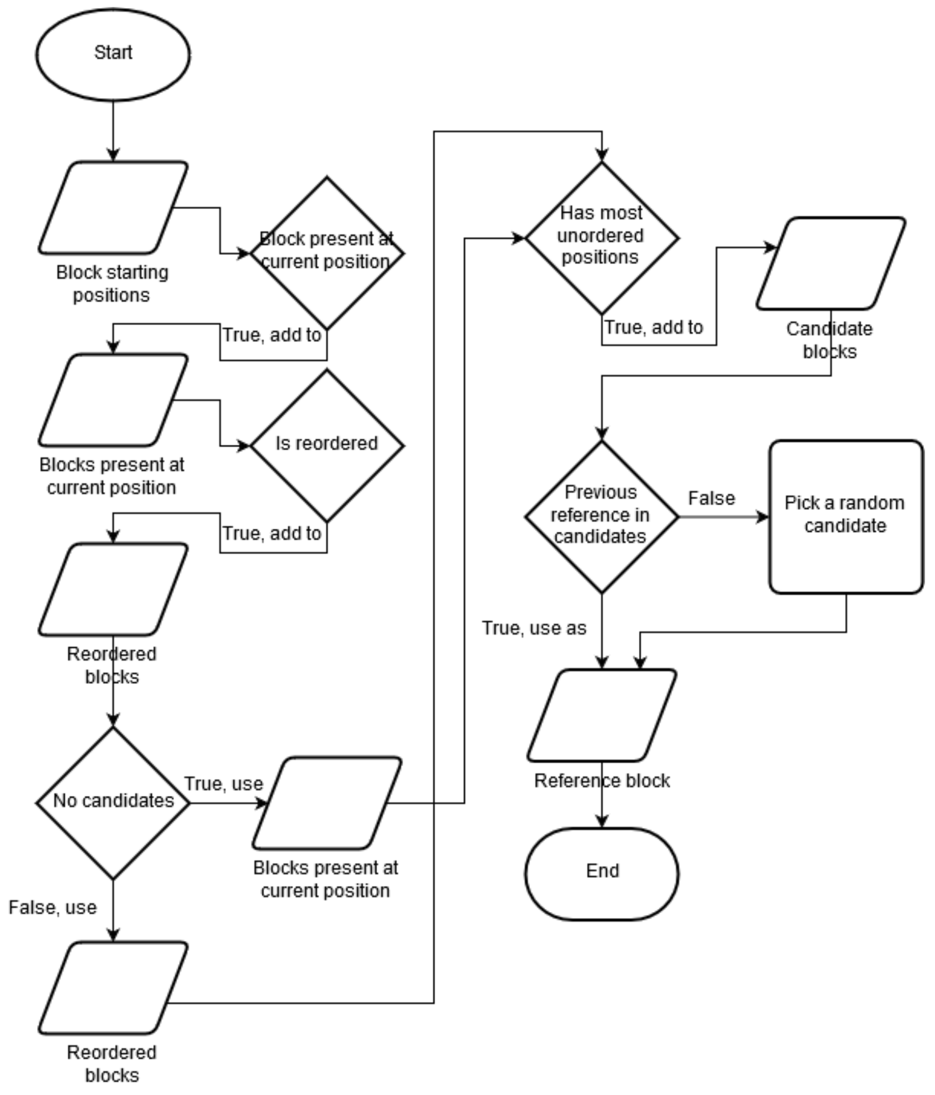


Figure 2: Flowchart illustrating the decisions made in the selection of reference block, part of the de novo reordering of haplotype blocks. Starting data are all tool estimates of which the block starting positions are used to loop through. For each block starting position, of all tools, the overlapping blocks are calculated and with these, it is checked whether any of them are already reordered. If none are reordered, all remain a candidate. Then, it is checked which of the candidates contains the most SNP positions yet to be reordered. The final decision is checking whether the previous reference block is still a candidate, in which it is used. Otherwise, a random block is chosen from the remaining candidate blocks.

The random forest models were trained using nine features for each SNP position, representing the three alleles of the three tool estimates. These tool estimates were reordered to the ground truth, to ensure features represent the same haplotype across the input. As outcomes, two strategies were applied. First, the three alleles of the ground truth were used as separate features (RF3). Second, a discrete value representing a specific combination of alleles was used (RF1). The reasoning behind these two approaches lies in how the random forest can use the in- and output features as these are related to one another which could result in a difference in performance. For test and training sets, half of the 60 simulated haplotypes and their estimates were used, having a combined size of around 5000 SNPs each. As no missing input features are allowed, only SNP positions present in all three tool estimates were used.

5. Performance assessment

To assess the performance of a haplotype estimate, three main statistics were calculated: allelic correlation, perfect solution rate and percentage of correctly phased alleles. For all three statistics, an estimate perfectly resembling the ground truth would score 1. Hapcompare, part of Haplosim, was used to calculate allelic correlation and perfect solution rate. Allelic correlation is a weighted metric of the fraction of correctly phased alleles and block size. Perfect solution rate describes the average accuracy of phasing for any possible SNP pair. A python script (Appendix 2) was used to calculate the fraction of correctly phased alleles in an estimate. Another python script (Appendix 3) was used to gather statistics regarding block size and fraction of correct alleles at said block size.

6. Variability assessment

In order to assess the variability in the estimated haplotypes of each tool, as well as to create a suitably sized training set for the random forest model, multiple haplotypes were generated from the 30kb reference sequence, with a total of at least 10.000 SNP positions. Given the varying number of SNPs generated at different SNP rates, multiple simulations were run. Each simulation consisting of using the 30kb reference to create a triploid, leading to a haplotype estimate from the different tools and ensemble methods, described below.

3. Results & discussion

1. Individual tool estimates are imperfect

In order to establish the potential improvements by combining haplotype estimates made by different tools, the performance of those tools needs to be assessed. For this, we used 60 simulated triploids and calculated the allelic correlation, perfect solution rate, and the fraction of correct alleles in the individual solutions, as defined in the performance assessment section of the methods.

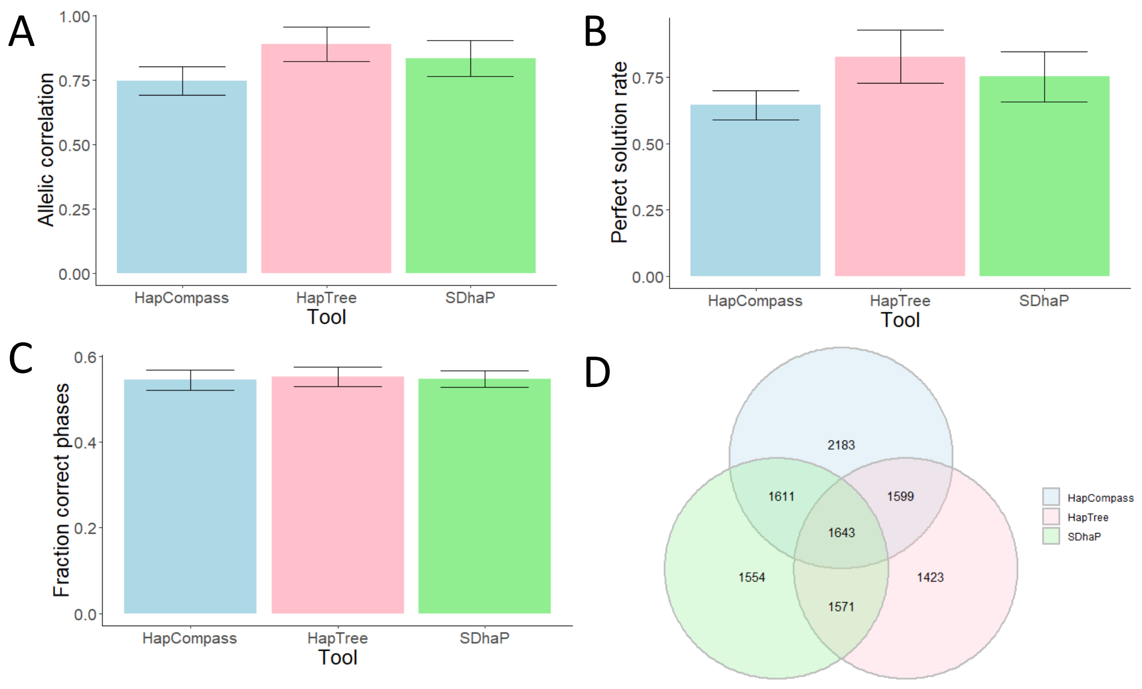


Figure 3: Performance metrics at 1% SNP rate, based on ground truth generated vcf files. A: Allelic correlation, a weighted statistic of the fraction of correctly phased alleles and block sizes, B: Perfect solution rate, C: Percentage of correctly phased alleles, D: Overlap of positions having an incorrectly phased allele.

As can be seen in Figures 3A-C, some variation is present between estimated haplotypes in all performance statistics. In general, haplotype estimates made by HapTree are the most accurate, followed by SDhaP. HapCompass is outperformed by the other two tools. None of the tool estimates perfectly resembles the ground truth however. When it comes to the percentage of correctly phased alleles, which does not take blocks into account, all tools perform similarly at around 55%. As shown in Figure 3-D, SNP positions with a phase error in any haplotype between tools overlap in such a way that selectively picking tool estimates for each position could result in a far more accurate haplotype estimate, reducing the original number of positions with a non-perfect phasing by more than 85%. Shown in Figure 4 are the performance statistics when the tools get a FreeBayes generated vcf file, sing simulated reads, as input. As can be seen, the performance is very similar. Some SNP positions are not called by FreeBayes due to low coverage but these represent is a very small fraction. We therefore use the ground truth based vcf files for our other analyses.

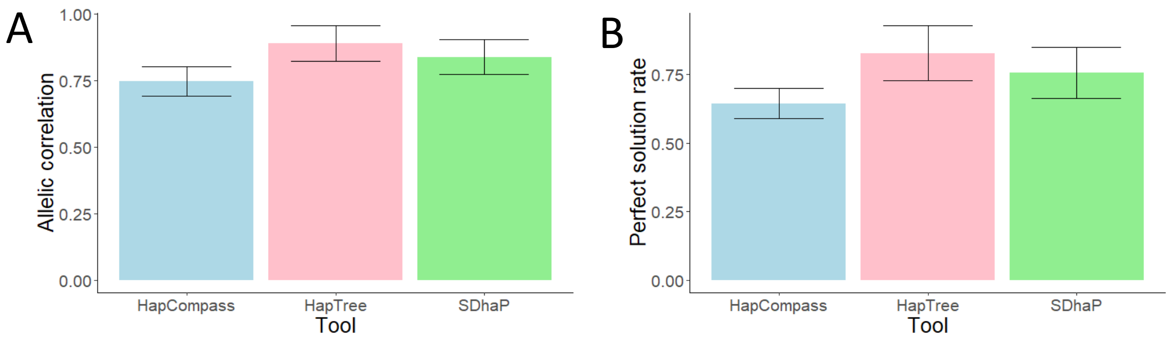


Figure 4: Performance metrics at 1% SNP rate, using FreeBayes generated vcf files. A: Allelic correlation, a weighted statistic of the fraction of correctly phased alleles and block sizes, B: Perfect solution rate.

Another aspect of the haplotype estimate is the number of alleles phased together in so called blocks. As tools more often than not are unable to resolve all alleles as a single haplotype, this aspect is taken into account in both allelic correlation, and perfect solution rate, but not in the fraction of correct alleles. As can be seen in Figures 5A-C, estimates from all three tools have a similar distribution of block sizes. Most of the blocks in the solutions encompass fewer than 40 SNPs, with a similar fraction of correctly phased alleles across block sizes, shown in Figures 5D-F. Performance wise, this means the tool estimate is quite fractured, with the majority of blocks encompassing very few SNPs.

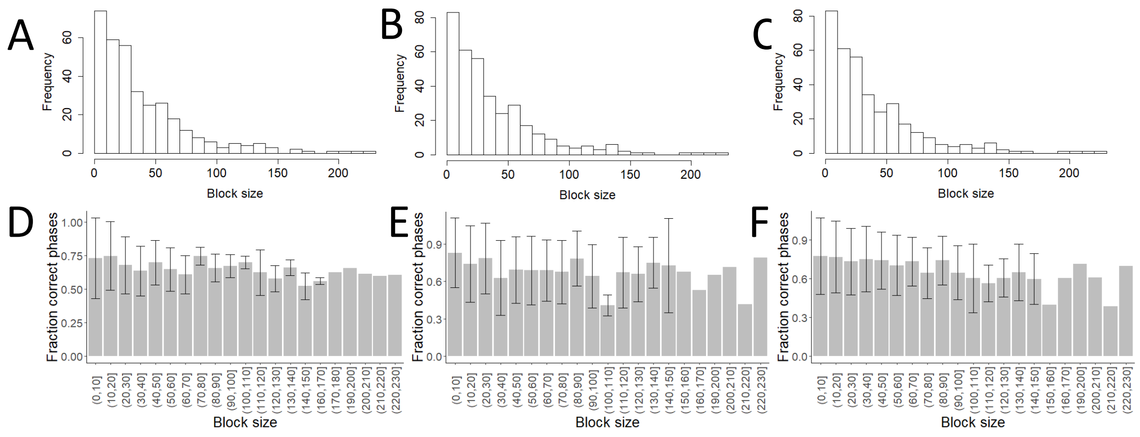


Figure 5: Block length distribution at 1% SNP rate for the tools HapCompass, HapTree and SDhaP in figures A, B and C respectively. Figures D, E and F show the percentage of correctly phased alleles in the same order. Histograms shown represent different block sizes from 0 to 230 positions in chunks of 10 each.

2. A majority vote estimate does not outperform individual tool estimates

Using the reordered haplotype estimates, the majority vote phasing of each allele was calculated via a python script (Appendix 1). As shown in Figure 6, both methods of reordering perform similarly, with a slight advantage to the GT reordering. It should be noted that the GT solution is optimal, reordering the estimates according to the minimum number of errors to the ground truth. The more realistic DN approach has the possibility of mismatching predicted haplotype estimates due to the tool estimates themselves containing errors.

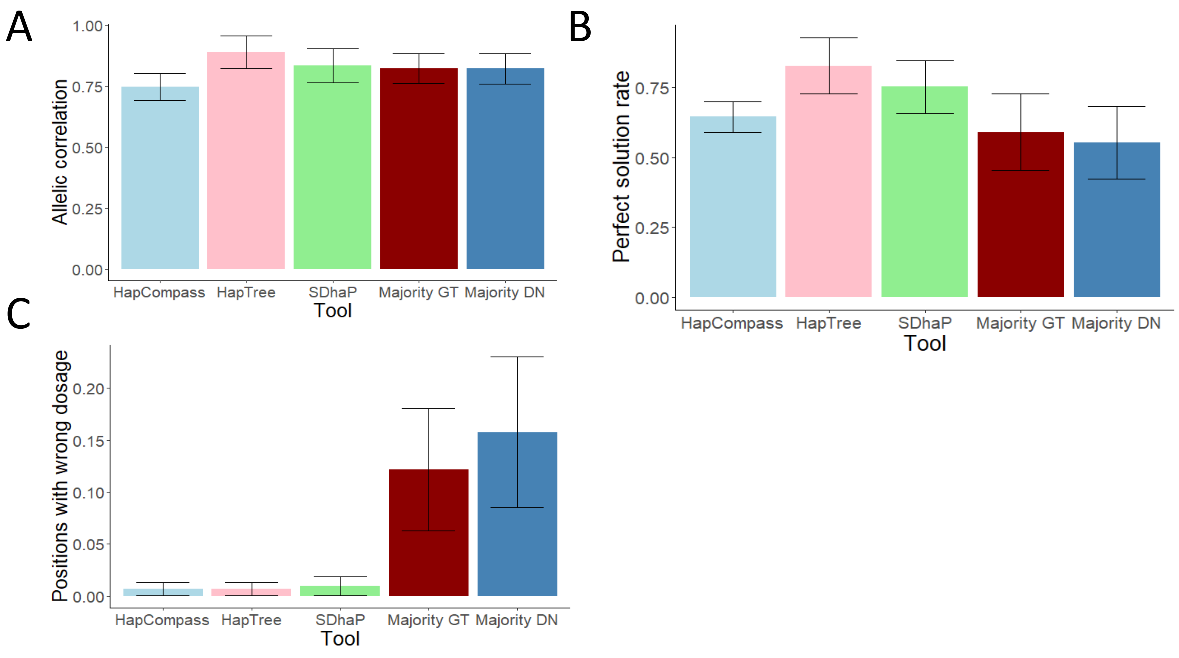


Figure 6: Majority vote performance statistics at 1% SNP rate. A: Allelic correlation, B: Perfect solution rate, C: Percentage of correctly phased alleles. Majority votes GT and DN differ in the assessment of which haplotype is which between tools and blocks, GT uses the ground truth, DN uses the tool estimates.

An important aspect of the different solutions is the dosage of alleles, the number of reference and alternate alleles found at each SNP position. As can be seen in Figure 6C, dosage is not a major issue for any of the tools. The majority vote of both reordering methods however, show a dosage error in 10-15% of SNP positions. This high degree of dosage errors is likely the result of the tool estimates disagreeing at certain alleles, in which the majority votes predict the reference allele for all three haplotypes. This indicates that the haplotype phases should not be treated independently from one another.

3. A random forest approach performs worse, but with fewer dosage errors

Given that the majority vote did not improve haplotype estimation beyond what an individual tool can achieve, we moved our focus towards a random forest model. As we now need a test and training set, we used 30 of the simulated haplotypes and their estimates for each, randomly selected from the 60 in total. The performance of the random forest model with a single output feature, representing specific combinations of three alleles, is shown in Figures 7A-D. Both on test and training data, it is worse than that of the individual tools in terms of allelic correlation and perfect solution rate. The dosage errors are less apparent compared to the majority votes, at around 10%, but still well above the error rate of the individual tools, as shown in Figures 7E-F. The percentage of correctly phased alleles, shown in Figures 7G-H, of the random forest model is in line with that of the individual tools.

The performance of the random forest predicting three alleles as separate features, shown in Supplementary Figure 1, shows a similar allelic correlation, but a decrease in perfect solution rate to near zero. This decrease is likely explained by the percentage of positions with a dosage error at about 75 percent. This again shows that predicted alleles should take one another into account, and not be treated separately.

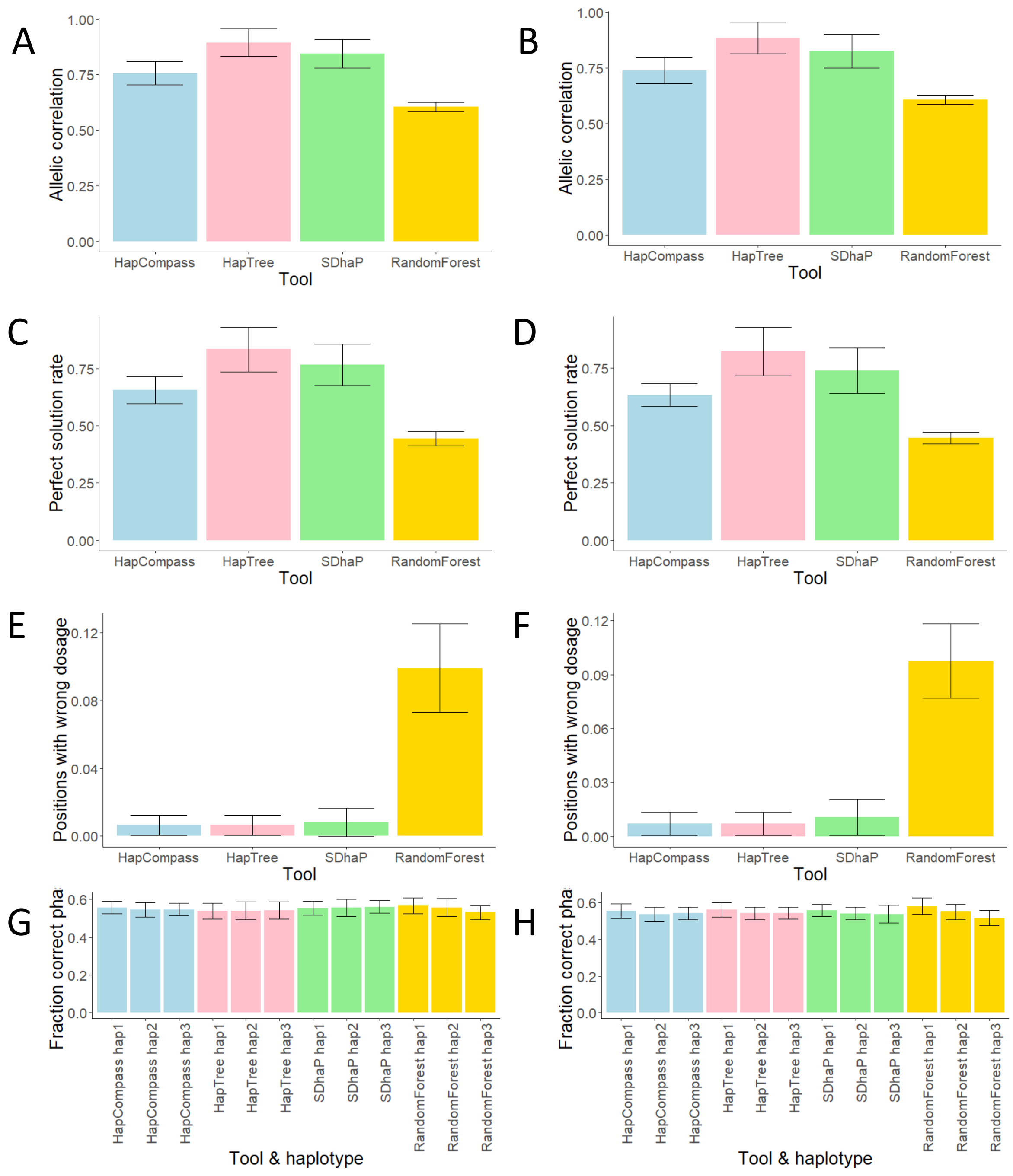


Figure 7: Various performance metrics at 1% SNP rate for the Random forest. Figures A, C, E and G are made using the training set, while Figures B, D, F and H use the test set. A & B show allelic correlation rate, C & D show the perfect solution rate, E & F show the fraction of positions with an incorrect dosage, figures G & H show the percentage of correctly phased alleles for each haplotype separately. The random forest model predicts a single output feature representing a specific combination of three phasings of a triploid.

4. Conclusion

In this project, we combined the haplotype estimates made by three tools. Using simulated haplotypes, reads and VCF files, we assessed the performance of these individual tools and used their estimates in both a majority vote, and a random forest model, using two different ways of output, in order to combine their haplotype estimates. The performance of ensemble methods that consider the haplotypes separately are shown not to exceed that of all the individual tools. Given the introduction of dosage errors of both the majority vote and the random forest model with three haplotype phasings as its prediction (RF3), we conclude that the multiple haplotypes should be predicted together rather than individually, as was done in these two ensemble approaches. Furthermore, we conclude the performance of the random forest model that considered the possible haplotype phasings as a single solution (RF1) to be promising, as despite the decrease in performance in terms of allelic correlation, perfect solution rate and dosage errors, it still managed to slightly outperform the individual haplotyping tools in the percentage of correctly phased alleles. While this could be a result of the increased dosage errors, we believe that further research into the input features of a random forest model hold the potential to increase performance above that of individual tools.

For future recommendations, a strategy would be to take away the apparent need to obfuscate the allele combinations under a single value, allowing to take all data into account, while not resulting in a high dosage error rate. This might be done using an adapter variant of a random forest that can take into account relations between features as external input. Other machine learning approaches like a neural net might also enable this. Another option to convey the relations of features would be to add features based on the vcf and reads that the haplotyping tools have at their disposal. The tested tools have no major dosage issues and therefore, these files must contain data that might be able to be transformed into a feature suitable to a machine learning approach.

Another aspect of the current approach of gaining the input features for both the random forest, and majority vote, is the reordering based on the ground truth. For a realistic approach, this reordering would ideally use the de novo reordering but for academic purposes, these are two separate issues and the best performing approach was used for the random forest. The importance here is having consistency in the input features of a model, meaning that feature 1 corresponds to the same haplotype across that entire tools input feature. Something that is not the case in the raw output of these tool estimates.

References

Aguiar, D., & Istrail, S. (2012). HapCompass: a fast cycle basis algorithm for accurate haplotype assembly of sequence data. *Journal of Computational Biology: A Journal of Computational Molecular Cell Biology*, *19*(6), 577–590. https://doi.org/10.1089/cmb.2012.0084

Aguiar, D., & Istrail, S. (2013). Haplotype assembly in polyploid genomes and identical by descent shared tracts. *Bioinformatics*, *29*(13), i352--i360. https://doi.org/10.1093/bioinformatics/btt213

Berger, E., Yorukoglu, D., Peng, J., & Berger, B. (2014). HapTree: a novel Bayesian framework for single individual polyplotyping using NGS data. *PLoS Computational Biology*, *10*(3), e1003502. https://doi.org/10.1371/journal.pcbi.1003502

Browning, S. R., & Browning, B. L. (2011). Haplotype phasing: existing methods and new developments. *Nature Reviews Genetics*, *12*(10), 703–714. https://doi.org/10.1038/nrg3054

Cilibrasi, R., van Iersel, L., Kelk, S., & Tromp, J. (2007). The Complexity of the Single Individual SNP Haplotyping Problem. *Algorithmica*, *49*(1), 13–36. https://doi.org/10.1007/s00453-007-0029-z

Das, S., & Vikalo, H. (2015). SDhaP: haplotype assembly for diploids and polyploids via semi-definite programming. *BMC Genomics*, *16*, 260. https://doi.org/10.1186/s12864-015-1408-5

Edge, P., Bafna, V., & Bansal, V. (2017). HapCUT2: robust and accurate haplotype assembly for diverse sequencing technologies. *Genome Research*, *27*(5), 801–812. https://doi.org/10.1101/gr.213462.116

Garrison, E., & Marth, G. (2012). *Haplotype-based variant detection from short-read sequencing*. Retrieved from https://arxiv.org/abs/1207.3907

Gibbs, R. A., Belmont, J. W., Hardenbol, P., Willis, T. D., Yu, F., Yang, H., … Group, M. (2003). The International HapMap Project. *Nature*, *426*(6968), 789–796. https://doi.org/10.1038/nature02168

Huang, W., Li, L., Myers, J. R., & Marth, G. T. (2011). ART: a next-generation sequencing read simulator. *Bioinformatics*, *28*(4), 593–594. https://doi.org/10.1093/bioinformatics/btr708

Luo, D., Zhang, M., Liu, T., Cao, W., Guo, J., Mao, C., … He, J. (2018). Long range haplotyping of paired-homologous chromosomes by single-chromosome sequencing of a single cell. *Scientific Reports*, *8*(1), 1640. https://doi.org/10.1038/s41598-018-20069-x

Majidian, S. (n.d.-a). 10xpipeline. Retrieved August 1, 2019, from https://github.com/smajidian/10xpipline

Majidian, S. (n.d.-b). Hap10. Retrieved August 1, 2019, from https://github.com/smajidian/Hap10

Motazedi, E. (2016). Haplosim. Retrieved June 1, 2019, from https://git.wageningenur.nl/motaz001/Haplosim

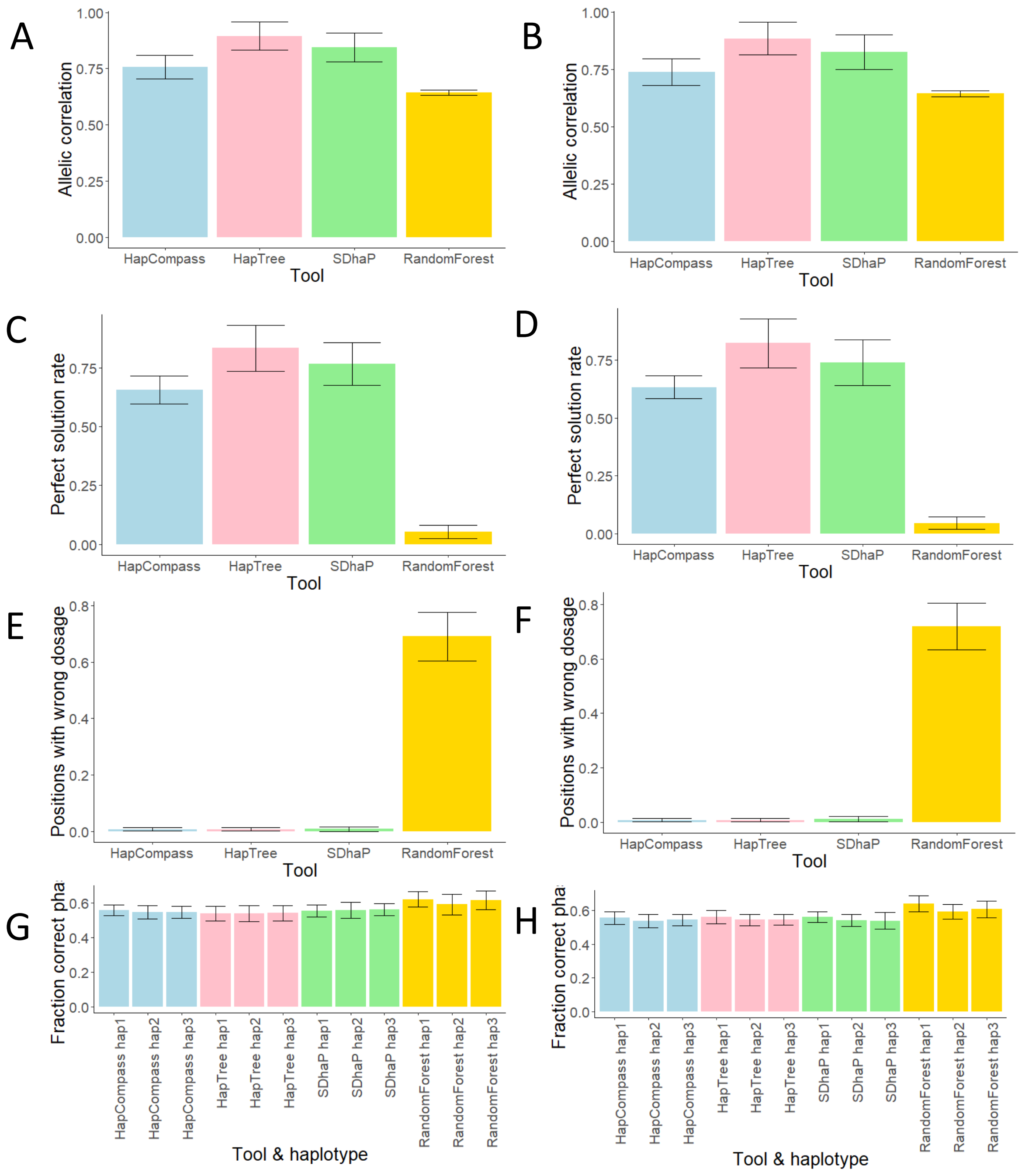
Motazedi, E., Finkers, R., Maliepaard, C., & de Ridder, D. (2018). Exploiting next-generation sequencing to solve the haplotyping puzzle in polyploids: a simulation study. *Briefings in Bioinformatics*, *19*(3), 387–403. https://doi.org/10.1093/bib/bbw126

Roe, D., Vierra-Green, C., Pyo, C.-W., Eng, K., Hall, R., Kuang, R., … Maiers, M. (2017). Revealing complete complex KIR haplotypes phased by long-read sequencing technology. *Genes and Immunity*, *18*(3), 127–134. https://doi.org/10.1038/gene.2017.10

Sutherland, G. R. (1996). Human Molecular Genetics. In *Bmj* (Vol. 312, pp. 1619–2215). https://doi.org/10.1136/bmj.312.7046.1619

Weirather, J. L., de Cesare, M., Wang, Y., Piazza, P., Sebastiano, V., Wang, X.-J., … Au, K. F. (2017). Comprehensive comparison of Pacific Biosciences and Oxford Nanopore Technologies and their applications to transcriptome analysis. *F1000Research*, *6*, 100. https://doi.org/10.12688/f1000research.10571.2

Supplementary



Supplementary Figure 1: Various performance metrics at 1% SNP rate for the Random forest using 3 output features. Figures A, C, E and G are made using the training set, while Figures B, D, F and H use the test set. A & B show allelic correlation rate, C & D show the perfect solution rate, E & F show the fraction of positions with an incorrect dosage, figures G & H show the percentage of correctly phases alleles for each haplotype separately. Random forest model was trained and predicts three output features, representing the phases of each haplotype.

Appendix

1. Python script (Compare_and_merge.py) can be found on the attached USB drive
2. Python script (collect_features.py) can be found on the attached USB drive
3. Python script (count_block_sizes.py) can be found on the attached USB drive
4. Bash script (pipeline_loopable.sh) can be found on the attached USB drive