Chapter 9

HapCUT2: A Method for Phasing Genomes Using Experimental Sequence Data

Vikas Bansal

Abstract

Rapid advances in high-throughput DNA sequencing technologies have enabled variant discovery from whole-genome sequencing (WGS) datasets; however linking variants on a chromosome together into haplotypes, also known as haplotype phasing, remains difficult. Human genomes are diploid and haplotype phasing is crucial for the complete interpretation and analysis of genetic variation.

Hapcut2 (https://github.com/vibansal/HapCUT2) is an open-source software for phasing diploid genomes using sequence data generated using different sequencing technologies and experimental methods. In this article, we give an overview of the algorithm used by Hapcut2 and describe how to use Hapcut2 for haplotype phasing of individual genomes using different types of sequence data.

Key words Haplotype, Phasing, Long reads, Proximity-ligation, Variants, Next-generation sequencing, Haplotype assembly

1 Introduction

Continued advancements in DNA sequencing technologies have made it possible to sequence human genomes routinely. However, the short read lengths of commonly used second-generation sequencing technologies provide information about variants and genotypes but very limited information about haplotypes—the combination of genetic variants that are present on a single chromosome. This is because for sequence reads to be haplotype informative, they must be long enough to cover multiple heterozygous variants. This is relatively infrequent for read lengths of a few hundred base pairs, but some partial haplotype information can sometimes be achieved. In addition, using overlaps between such haplotype-informative reads can allow longer haplotypes to be assembled [1].

Recognizing the importance of haplotype information for interpreting genetic variation in human genomes, a range of

Brock A. Peters and Radoje Drmanac (eds.), *Haplotyping: Methods and Protocols*,
Methods in Molecular Biology, vol. 2590, https://doi.org/10.1007/978-1-0716-2819-5_9,

experimental protocols and techniques have been developed that enable the reconstruction of whole-genome haplotypes using computational methods [2–6]. These include dilution-pool sequencing and long fragment read (LFR) sequencing (i.e., co-barcoding or linked-read technologies) that preserve haplotype information from long DNA fragments (tens to hundreds of kilobases) in short sequence reads (explained in Chaps. 6, 7, and 11). In addition, proximity-ligation sequencing has been shown to enable chromosome-scale haplotype phasing [7]. In recent years, third-generation technologies such as Pacific Biosciences and Oxford Nanopore have emerged which enable haplotype phasing directly from their long contiguous reads [8].

All of these sequencing protocols and technologies have the capability to generate reads or fragments with haplotype information but require computational tools to assemble the reads into long haplotypes. A number of algorithms have been developed for haplotype assembly [8–11]. The method described in this chapter, HapCUT2, is a computational method designed to enable haplotype phasing using diverse types of sequence data such as proximity-ligation (Hi-C) sequencing [6], long-read sequencing, and linked-read sequencing [5]. Here we describe how to use HapCUT2 for phasing genomes using experimental sequence data. The input to HapCUT2 consists of (i) a list of heterozygous variants—typically identified from alignment of whole-genome sequence (WGS) data to a reference genome; and (ii) haplotype fragments (sequences of alleles at heterozygous variant sites identified from sequence reads aligned to a reference genome). HapCUT2 aims to assemble a pair of haplotypes that are maximally consistent with the input set of haplotype fragments. Note that the sequence reads used for identifying the heterozygous variants can be different from those used for phasing. This allows HapCUT2 to use multiple different types of sequence data for phasing.

2 Materials

HapCUT2 is designed to run on Unix/Linux and Mac OS operating systems. All of the installation instructions and required software tools are listed below.

2.1 Software Requirements

1. htslib (version >1.2.1): https://github.com/samtools/htslib.
2. GNU make (*see* **Note 1**).
3. C compiler (e.g., gcc or clang, *see* **Note 1**).

2.2 Installation

1. Htslib is a C library for processing high-throughput sequencing data formats. It can be installed as follows:

```
git clone https://github.com/samtools/htslib.git
cd htslib
autoreconf –i
./configure
make
```

Note that htslib has its own dependencies (e.g., libssl-dev) that need to be installed. Once htslib is installed, the environment variable LD_LIBRARY_PATH variable should be updated as follows:

```
export LD_LIBRARY_PATH="$LD_LIBRARY_PATH:/path_to_htslib_-
directory"
```

2. Subsequently, HapCUT2 can be installed from source using the following steps:

```
git clone https://github.com/vibansal/HapCUT2
cd HapCUT2
make
```

Before running make, the Makefile should be edited to set the HTSLIB variable to the directory where htslib is installed. The HapCUT2 codebase on github includes a script "install.sh" that automates these instructions.

2.3 Installation Using Conda

Conda is an open-source package management system that automatically installs all dependencies. Installation instructions for Miniconda (a conda installer) are available at: https://docs.conda.io/en/latest/miniconda.html.

1. Once Miniconda is installed, HapCUT2 can be installed using a one-line command:

```
conda config --add channels defaults
conda config --add channels bioconda
conda config --add channels conda-forge
conda install -c bioconda hapcut2
```

3 Methods

3.1 Input Requirements

HapCUT2 requires two files as input: a variant file (in VCF format, *see* **Note 2**) with the list of variants (and genotypes) to be phased (for a single individual, *see* **Note 3**) and one or more BAM/CRAM files with reads aligned to a reference genome (*see* Fig. 1). The VCF file and the aligned reads file should correspond to the same reference genome version (*see* **Note 5**). The reference genome sequence is not required for phasing single-nucleotide variants (SNVs) but is needed for phasing insertion/deletion variants or indels.

3.2 Extraction of Haplotype-Informative Reads

The ExtractHAIRs (Extract HAplotype Informative Reads) program is part of HapCUT2 and is designed to process aligned reads in a sorted BAM file and heterozygous variants from a VCF file (with variant calls) to create the haplotype "fragment file." It can efficiently process paired-end sequence data as well as third-generation datasets such as PacBio reads. For analyzing Hi-C data, the fragment file format was modified to store extra information for each read including the data type (Hi-C or long read), variant start index of the second read, and paired-end insert size. This file represents each haplotype informative read or fragment as a list of heterozygous variants (indices of variants in the VCF file) and the corresponding alleles (with quality values) at each variant site. Consecutive heterozygous variants covered by a single read are compressed to form a single block. This format was first utilized in the assembly of the whole-genome Sanger sequence data for HuRef [12]. Each line of this file records haplotype information from a single fragment as follows:

Column 1 is the number of blocks (consecutive set of variants covered by the fragment).

Column 2 is the fragment id.

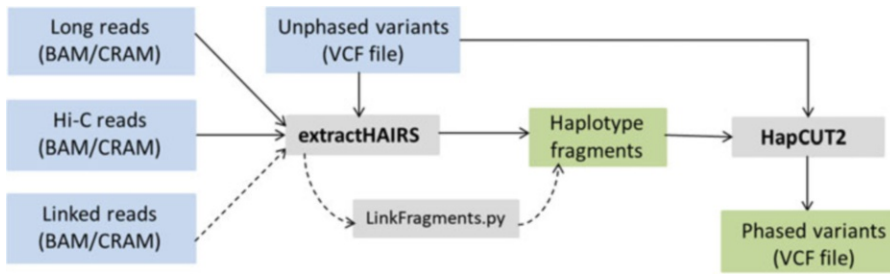

Fig. 1 Overview of haplotype phasing using HapCUT2 for different sequencing data types. Haplotype phasing is a two-step process involving extraction of "haplotype fragments" from the aligned reads (**Step 1**) and assembly of the fragments into haplotypes (**Step 2**). For linked-reads, an additional step is required to link the short reads into haplotype fragments using barcode information (shown using dotted lines)

Column 3 is the offset of the first block of variants covered by the fragment followed by the alleles of the variants in this block.

Column 4 is the sequence of alleles at variants in the first block.

Column 5 is the offset of the second block of variants covered by the fragment followed by the alleles of the variants in this block.

The last column is a string with the quality values (Sanger fastq format) for all the alleles covered by the fragment (concatenated for all blocks).

The offset for each variant corresponds to the index of the variant in the VCF file (one-based). For example, if a read/fragment covers SNPs 2, 3, and 5 with the alleles 0, 1, and 0, respectively, then the fragment will be:

```
2 read_id 2 01 5 0 AAC
```

Here AAC is the string corresponding to the quality values at the three alleles (encoded using char(QV+33)). The encoding of 0/1 follows the VCF format, 0 is reference and 1 is alternate.

1. Use extractHAIRS to convert the BAM file to the compact fragment file format containing only haplotype-relevant information:

```
./build/extractHAIRS [options] --bam reads.sorted.bam \
--VCF  variants.vcf --out fragment_file
```

3.3 Assembly of Haplotypes

1. Use HAPCUT2 to assemble the fragment file into haplotype blocks:

```
./build/HAPCUT2 --fragments fragment_file --VCF  variants.vcf
\
--output haplotype_output_file
```

This two-step process (extraction of haplotype fragments and assembly of haplotypes) allows for flexibility in using different types of sequence data for haplotype phasing. Users can also generate the haplotype fragment file using a custom pipeline and subsequently use HapCUT2 for phasing. Conversely, the extractHAIRS program can be used to extract haplotype fragments for phasing using algorithms other than HapCUT2. Note that the "variants.vcf" file used for running extractHAIRS and HapCUT2 should be the same (*see* **Note 6**). A schematic overview of the input and output files and the steps for haplotype phasing using HapCUT2 is provided in Fig. 1.

3.4 Output Files

HapCUT2 outputs the assembled haplotype information in two formats. The first is a custom format which contains the haplotype information for variants in individual blocks. For a given block, column 2 represents the allele on one chromosome copy (0 for reference, 1 for variant), while column 3 represents the allele on the other copy. The format of this file is described in detail here: https://github.com/vibansal/HapCUT2/blob/master/outputformat.md

HapCUT2 also outputs the phasing information in the VCF format using the genotype (GT), phase set (PS), and the phasing quality (PQ) fields. As per the VCF specification, a phase set is defined as "a set of phased genotypes to which this genotype belongs." A detailed description of these fields can be obtained from the VCF specification document here: https://samtools.github.io/hts-specs/VCFv4.2.pdf

3.5 Phasing Using Third-Generation Reads

For long-read sequencing technologies such as Pacific Biosciences SMRT and Oxford Nanopore, the average error rate of the reads is much greater than that of second generation reads and is dominated by insertion/deletion errors. This increases the uncertainty in the allele information provided by individual reads at heterozygous variant sites. To address this, the extractHAIRS module performs local realignment in a small window around each heterozygous variant for each read overlapping it that accounts for the uncertainty in the base-to-base alignment. This realignment uses a pair-Hidden Markov Model (HMM) and the parameters of this model can be estimated directly from the data itself ("--ep 1") option. It is also recommended to provide the reference genome fasta file (*see* **Note 4**) as input for improving the accuracy of the haplotype fragments. In addition, "--pacbio 1" and "--ont 1" options should be used for running extractHAIRS on Pacific Biosciences and Oxford Nanopore reads respectively.

3.6 Phasing Using Hi-C Reads

HapCUT2 models Hi-C specific error modalities (i.e., trans errors) and estimates the trans-error rate using an iterative algorithm. For phasing with Hi-C reads or other types of proximity-ligation-based paired-end reads, the "--hic" option should be set to 1. extractHAIRS considers read pairs for which both ends map to the same chromosome and for which the insert size is smaller than a specified threshold (default value equal to 40 Megabases) as a single haplotype fragment. The "--maxIS" parameter can be used to decrease or increase this threshold.

In addition, for phasing Hi-C reads, it is recommended to align the reads using BWA using the -5, -S, and -P options. This preserves the mate-pair information in the aligned reads file.

```
BWA mem -5SPM reference.fasta read1.fq read2.fq > reads.sam
```

3.7 Phasing Using Linked-Reads (Co-barcoded Reads)

Multiple experimental methods have been developed to generate linked-reads derived from sequencing of labeled short DNA fragments originating from a single long DNA fragment with barcodes inside hundreds of thousands to millions of separate compartments. The original DNA molecules from linked-read data can be extremely long (>100 kb). The format used by HapCUT2 for storing haplotype fragments is well-suited for storing haplotype information from linked-reads but requires additional pre-processing to generate the linked-reads:

1. First, the extractHAIRS program is used to convert the input BAM file to a compact fragment file format containing only haplotype-relevant information:

```
 ./build/extractHAIRS --10X 1 --bam reads.sorted.bam --VCF
variants.vcf\
 --out unlinked_fragment_file
```

2. The LinkFragments.py script can then be used to link fragments into barcoded molecules. It is assumed that reads with the same barcode occurring within a maximum distance (default 20 kb) belong to the same molecule and reads separated by more than this distance are assigned to separate molecules. This distance can be controlled using the -d parameter in the LinkFragments script. The LinkFragments.py script uses the BX or MI tags (these tags contain the barcode information) present on individual reads to generate fragments. The LinkFragments.py script generates a fragment file that can be used for phasing using HapCUT2.

```
Python3 utilities/LinkFragments.py –-bam reads.sorted.bam\
--VCF variants.vcf –-fragments unlinked_fragment_file\
--out linked_fragment_file
```

A readme file is available in the HapCUT2 github repository (linkedreads.md) that describes these steps in more detail.

3.8 Phasing Using Data from Multiple Sequencing Technologies

If you have data from different technologies or in multiple BAM files for the same individual, follow these steps below.

1. Generate a single VCF file with variants, preferably using second-generation sequencing data or another source of high-accuracy reads.
2. Run extractHAIRS on each of the BAM/CRAM files independently and concatenate (linux "cat" command) the output files. The "--nf 1" option should be used when running extractHAIRS when combining data from different sequencing technologies. This outputs the fragments in the Hi-C fragment

file format. The output files can then be concatenated and processed with HapCUT2 (also need to use the "--nf 1" option, *see* **Note 7**).

3. Run HapCUT2 with the combined output file from (2) and the VCF file from (1).

The set of commands for phasing using data from two different technologies (PacBio long reads and Hi-C reads) are as follows:

```
./build/extractHAIRS --pacbio 1 --ep 1 --nf 1 --bam pacbio_reads.bam --VCF variants.vcf --out pacbio_fragment_file --fasta reference.fasta
./build/extractHAIRS --hic 1 --bam HiC_reads.bam --VCF variants.vcf --out HiC_fragment_file --maxIS 10000000
cat pacbio_fragment_file HiC_fragment_file > combined_fragment_file
./build/HAPCUT2 --hic 1 --fragments combined_fragment_file --VCF variants.vcf --output haplotype_output_file
```

3.9 Phasing Indels

HapCUT2 can phase indels or short insertion/deletion variants (included in the VCF file). For this, extractHAIRS performs indel realignment when extracting haplotype-informative reads from BAM files (use "--indels 1" option for this and provide a reference fasta file). HapCUT2 requires the fasta file to be indexed using samtools or a similar program.

3.10 Parallelizing HapCUT2

For genomes with multiple chromosomes, HapCUT2 can be run in parallel on individual chromosomes by splitting the VCF file by chromosome using tabix (www.htslib.org/docs/tabix.html). This requires the input file to be gzipped and indexed.

```
tabix -h variants.vcf.gz chrA > variants.chrA.vcf
```

The extractHAIRS command can then be run on individual chromosomes using the "--region" option without splitting the BAM/CRAM files as follows:

```
./build/extractHAIRS [options] --bam reads.sorted.bam \
--VCF variants.chrA.vcf --out fragment_file --region chrA
```

4 Notes

1. GNU make and C compiler are typically already installed on Linux and Mac OS systems.
2. Currently, HapCUT2 does not accept gzipped VCF files as input. It is important to provide an unzipped VCF file as input.

3. HapCUT2 is designed to phase individual genomes; therefore, the input VCF file should contain genotypes only for a single individual and should not contain non-diploid genotypes or genotypes of the form 1/3, 1/4, etc.
4. The reference genome file used for running extractHAIRS should be the same as the one used for aligning the sequence reads and indexed using samtools (www.htslib.org/doc/samtools-faidx.html).
5. If a different source of variants is used for phasing, care should be taken that the reference genome used for the variants is the same as for the aligned reads.
6. The same VCF file should be used for running extractHAIRS and HapCUT2.
7. When running HAPCUT2 on Hi-C reads (or a combination of Hi-C reads and other types of sequence reads), the option "--hic 1" should be used.

References

1. Bansal V, Halpern AL, Axelrod N, Bafna V (2008) An MCMC algorithm for haplotype assembly from whole-genome sequence data. Genome Res 18(8):1336–1346
2. Kitzman JO, Mackenzie AP, Adey A et al (2011) Haplotype-resolved genome sequencing of a Gujarati Indian individual. Nat Biotechnol 29(1):59–63. https://doi.org/10.1038/nbt.1740
3. Peters BA, Kermani BG, Sparks AB et al (2012) Accurate whole-genome sequencing and haplotyping from 10 to 20 human cells. Nature 487(7406):190–195. https://doi.org/10.1038/nature11236
4. Snyder MW, Adey A, Kitzman JO et al (2015) Haplotype-resolved genome sequencing: experimental methods and applications. Nat Rev Genet 16(6):344–358. https://doi.org/10.1038/nrg3903
5. Zheng GX, Lau BT, Schnall-Levin M et al (2016) Haplotyping germline and cancer genomes with high-throughput linked-read sequencing. Nat Biotechnol 34(3):303–311. https://doi.org/10.1038/nbt.3432
6. Chen Z, Pham L, Wu TC et al (2020) Ultralow-input single-tube linked-read library method enables short-read second-generation sequencing systems to routinely generate highly accurate and economical long-range sequencing information. Genome Res 30(6):898–909. https://doi.org/10.1101/gr.260380.119
7. Selvaraj SR, Dixon J, Bansal V et al (2013) Whole-genome haplotype reconstruction using proximity-ligation and shotgun sequencing. Nat Biotechnol 31(12):1111–1118. https://doi.org/10.1038/nbt.2728
8. Edge P, Bafna V, Bansal V (2017) HapCUT2: robust and accurate haplotype assembly for diverse sequencing technologies. Genome Res 27(5):801–812. 05
9. Bansal V, Bafna V (2008) HapCUT: an efficient and accurate algorithm for the haplotype assembly problem. Bioinformatics 24(16):i153–i159
10. Kuleshov V (2014) Probabilistic single-individual haplotyping. Bioinformatics 30(17):i379–i385
11. Patterson M, Marschall T, Pisanti N et al (2015) WhatsHap: weighted haplotype assembly for future-generation sequencing reads. J Comput Biol 22(6):498–509. https://doi.org/10.1089/cmb.2014.0157
12. Levy S, Sutton G, Ng PC et al (2007) The diploid genome sequence of an individual human. PLoS Biol 5(10):e254. https://doi.org/10.1371/journal.pbio.0050254