**Input & output format of Haplotype Assembly Software Packages**

Contents

[HapCompass 2](#_Toc14169076)

[HapCUT2 3](#_Toc14169077)

[AltHap 4](#_Toc14169078)

[MixSIH 5](#_Toc14169079)

[CSP 7](#_Toc14169080)

[PEATH 9](#_Toc14169081)

[SDhaP 9](#_Toc14169082)

[WinHap2 11](#_Toc14169083)

[WhatsHap 12](#_Toc14169084)

## HapCompass

Manual: <https://www.brown.edu/Research/Istrail_Lab/resources/hapcompass_manual.html#sec11>

**Output**

Several files will be output to the -o directory.

* OUTPUT\_PREFIX\_<ALGORITHM>\_solution.txt  
  Contains a list of blocks and phasings. A new block is shown by BLOCK <start\_position\_of\_block> <end\_position> <start\_snp\_number> <end\_snp\_number> <score>  
  then a list of SNPs in the block...  
  <snp\_id> <snp\_position> <snp\_number> <hap\_0\_allele> <hap\_1\_allele>

In the case of assembling more than 2 haplotypes, there will be additional columns for the extra haplotypes.

* OUTPUT\_PREFIX\_reduced\_representation.vcf and  
  OUTPUT\_PREFIX\_reduced\_representation.sam  
  A list of SAM reads and variants covering the blocks of variants in terms of the variant numbers instead of variant positions.
* OUTPUT\_PREFIX\_reads.sam  
  A different representation of input reads in terms of true genomic positions.
* OUTPUT\_PREFIX\_binning.txt  
  This file will only be produced if -ploidy > 2. Contains the pairwise variant phasings defined in the compass graph *GC*.
* OUTPUT\_PREFIX\_frags.txt  
  BAM input files are converted into an alternative representation that contains only the information required by HapCompass. This file will only contain the sequence reads that contain 2 or more variant alleles.

**File formats**

The HapCompass package includes sample files for all types of input.

**Variants**

HapCompass always requires specification of variants in the form of a VCF file. VCF specification can be found <https://github.com/amarcket/vcf_spec>.

**Aligned sequence reads**

There are two options for inputting sequence reads: BAM and fragment files.

A SAM file is a tab delimited file for specifying sequence reads and their properties. A BAM file is the SAM files binary version. BAM and SAM files may be manipulated using SAMTools. The BAM specification can be found <http://samtools.sourceforge.net/SAMv1.pdf>.

Fragment files are a **tab** delimited file format with reads separated by new lines. The columns are specified as follows:

**column 1:**

number of read parts, e.g. 1 for a single read, 2 for two reads (paired sequencing when both ends are informative)

**column 2:**

read name

**column 3 and on:**

the read parts. The number of read parts is defined in column 1.

A read part is a **space** delimited list beginning with a reference name and then a set of one or more position(s), allele(s), base quality score(s) trios. The position is the 0-based position of the variant in the VCF file. E.g. the first variant is position 0, the second variant is position 1, and so on. The allele is the numerically encoded allele, 0 for reference, 1 for first alternative allele, and so on. The base quality score is the ASCII of Phred-scaled base quality+33. An example of a paired read is given below <tab> denotes a tab character:

2<tab>sequence\_read\_1<tab>chr1 0 1 D 1 1 D<tab>chr1 4 0 D 5 0 D 6 0 D

## HapCUT2

Manual: <https://github.com/vibansal/HapCUT2>

**Input:**

HapCUT2 requires the following input:

* BAM file for an individual containing reads aligned to a reference genome
* VCF file containing short variant calls (SNVs and indels) and **diploid** genotypes for the same individual with respect to the reference genome

**Output Format:**

**HapCUT2 now outputs the phased variants to a VCF file "output\_haplotype\_file.phased.vcf"**

Haplotype blocks are printed to the output file. 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.

Each block starts with a block header with the following format:

BLOCK: offset: <SNV offset> len: <SNV span of block> phased: <# SNVs phased> SPAN: <base pair span of block> fragments <# of fragments in block>

Following the header, there is one line per SNV with the following tab-delimited fields:

1. VCF file index (1-based index of the line in the input VCF describing variant)
2. allele on haploid chromosome copy A (0 means reference allele, 1 means variant allele)
3. allele on haploid chromosome copy B (0 means reference allele, 1 means variant allele)
4. chromosome
5. position
6. reference allele (allele corresponding to 0 in column 2 or 3)
7. variant allele (allele corresponding to 1 in column 2 or 3)
8. VCF genotype field (unedited, directly from original VCF)
9. discrete pruning status (1 means pruned, 0 means phased)
10. switch quality: phred-scaled estimated probability that there is a switch error starting at this SNV (0 means switch error is likely, 100 means switch is unlikely)
11. mismatch quality: phred-scaled estimated probability that there is a mismatch [single SNV] error at this SNV (0 means SNV is low quality, 100 means SNV is high quality)

Field 9 describes the status of the SNV under the discrete SNV pruning method introduced by RefHap (an SNV is pruned if there are equal reads consistent and inconsistent with the phase), with the slight difference that reads are assigned to haplotypes using likelihoods in our implementation. Use the option "--discrete\_pruning 1" to automatically prune SNPs ('- -' phasing) based on the value of this field.

The values in field 10 and field 11 are quality scores that range from 0 to 100 (less confident to more confident). Field 10 is useful for controlling switch errors, especially on data types such as error-prone SMRT reads. It is empty by default (".") unless switch error scores are computed using "--error\_analysis\_mode 1" (compute switch error confidence but leave blocks intact and all SNVs phased for manual pruning later).

Important note: flag "--split\_blocks 1" (compute switch error confidence and automatically split blocks using the value of --split\_threshold) is currently broken, for the time being use "--error\_analysis\_mode 1" and manually split using field 10.

Field 11 is useful for controlling mismatch (single SNV) haplotype errors, similarly to field 9. The default behavior of HapCUT2 is to prune individual SNVs for which this confidence is less than 6.98 (probability of error 0.2), as these are highly likely to be errors.

## AltHap

Manual: <https://github.com/realabolfazl/AltHap>

**Input file format**

The input file format is as follows

Number of reads   
Number of columns   
Number\_of\_contiguos\_segments Read\_identifier Position\_of\_first\_SNP\_segment Continuous\_bases\_in\_read Position\_of\_next\_SNP\_segment Continuous\_bases\_in\_read ..... Quality\_scores (in fastq format)

* Example for Biallelic:

5568   
22801   
2 chr22\_SPA9\_8733 2 0 5 0 ==   
1 chr22\_SPH2\_1940 3 100 C==

**Output file format**

The output file contains MEC score, CPU Time, and Recovered Haplotypes. Subsequently, the phased haplotype is printed in the following format: first haplotype second haplotype third haplotype ...

* Example for Biallelic:

MEC: 353   
CPU Time: 38.363051   
Recovered Haplotype:   
0 1 1   
1 0 1   
0 0 1   
1 1 1   
0 1 1   
1 0 0   
1 0 1   
1 1 1

## MixSIH

Manual: <https://sites.google.com/site/hmatsu1226/software/mixsih>

Format of Input\_file:  
The first line describes the number of lines of Input\_file (which corresponds to the number of SNP fragments - 1).  
After first line, each line describes a SNP fragment as below.  
\segment\_num \fragment\_name \start\_site1 \sequence1 \start\_site2 \sequence2.....  
where \segment\_num is the number of the segments which don't have gaps in the segments,  
\fragment\_name is the name of the fragment,  
\start\_allele'i' is the first site's position of i-th segment (1-origin),  
\sequence'i' is the sequence of i-th segment.  
The fragments must be sorted by the value of the third column and these can be sorted as follows:  
    **sort -n -k 3 frag.txt > frag\_sorted.txt**  
  
Example of Input\_file:  
16  
1 frag1 1 000  
1 frag2 1 0001  
2 frag3 1 11 4 111  
2 frag4 2 11 5 11  
1 frag5 3 000  
1 frag6 4 000000  
1 frag7 4 1111  
3 frag8 5 11 8 1 10 1  
1 frag9 6 000  
1 frag10 7 0000  
1 frag11 7 1011  
1 frag12 7 111  
1 frag13 8 010  
1 frag14 9 11  
1 frag15 9 00  
  
  
Format of Output\_file1:  
The Output\_file1 is composed of the list of the blocks.  
Each block contains a header which is composed of the relative position of the first site,  
the number of the　sites in the block and the number of the sites which can be phased.  
Each block consists of columns as follows:  
 Col1: relative position of the site  
 Col2: probability that the phase is (0,1)  
 Col3: probability that the phase is (1,0)  
 Col4: exp(*connectivity*) of the site  
  
Example of Output\_file1:  
BLOCK: offset: 1 len: 10 phased: 10  
1    0.127    0.873    0.000  
2    0.102    0.898    4.424  
3    0.102    0.898    7.351  
4    0.224    0.776    6.891  
5    0.072    0.928    10.566  
6    0.072    0.928    11.954  
7    0.073    0.927    11.126  
8    0.166    0.834    12.211  
9    0.149    0.851    8.469  
10    0.078    0.922    7.296  
\*\*\*\*\*\*\*\*  
  
  
Format of Output\_file2:  
The Output\_file2 is composed of the list of the blocks.  
Each block contains a header which is composed of the relative position of the first site,  
the number of the sites in the block, the number of the sites which can be phased.  
Each block consists of columns as follows:  
 Col1: relative position of the site  
 Col2: allele in the first haplotype  
 Col3: allele in the second haplotype  
  
Example of Output\_file2:  
BLOCK: offset: 1 len: 10 phased: 10  
1    0    1  
2    0    1  
3    0    1  
4    0    1  
5    0    1  
6    0    1  
7    0    1  
8    0    1  
9    0    1  
10    0    1  
\*\*\*\*\*\*\*\*

------------------------  
Extract reliable regions  
------------------------  
extract\_reliable\_region.rb divides the haplotypes so that MC of the divided regions are higher than threshold.  
  
Usage:  
    ruby extract\_reliable\_region.rb <Input\_file1> <Input\_file2> <Output\_file1> <Output\_file2> <threshold>  
  
Example:  
    ruby extract\_reliable\_region.rb profile.txt haplotype.txt profile\_6.txt haplotype\_6.txt 6.0  
  
Format of Input\_file1 and Output\_file1:  
It is the same format of Output\_file1 of MixSIH.  
  
Format of Input\_file2 and Output\_file2:  
It is the same format of Output\_file2 of MixSIH.

## CSP

Manual: [**https://sites.google.com/site/hmatsu1226/software/csp**](https://sites.google.com/site/hmatsu1226/software/csp)

Usage:

ruby CSP1.rb <Genotype\_file> <Fragment\_file> <Output\_file1> <PHASE\_file1> <PHASE\_file2> <N> <W>

Example of running CSP1:

ruby CSP1.rb example/genotype.txt example/fragment.txt out/csp1.txt phase/input.txt phase/output.out 11 5

Genotype\_file:

This contains population genotypes information.

Format of the file is

<chromosome number> <chromosome position> <refSNP> <base1> <base2> <genotype1> <genotype2> <genotype3> ...

where <genotype(n)> is n-th individual genotype of a SNP.

<genotype1> has to be an individual who is the target of the dilution-based sequencing.

Example of the file is as follows.

1 52066 rs28402963 T C 10 01 01 00 01 00 00 00 10 00 00

1 695745 . G A 10 00 00 00 00 00 00 10 00 00 00

1 766409 rs12124819 A G 01 01 00 00 00 10 01 11 11 00 11

1 801628 . C T 01 00 01 00 00 00 00 00 00 00 00

1 805678 . A T 01 -- -- -- -- -- -- -- -- -- --

1 805716 . A G 01 -- -- -- -- -- -- -- -- -- --

1 806222 . G A 01 00 00 10 10 00 11 01 00 00 10

In our paper, we generated this file from CEU genotypes, which were downloaded from 1000 genomes project (ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/pilot\_data/release/2010\_07/trio/snps/CEU.trio.2010\_03.genotypes.vcf.gz and ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/pilot\_data/release/2010\_07/low\_coverage/snps/CEU.low\_coverage.2010\_07.genotypes.vcf.gz).

Fragment\_file:

This contains SNP fragments.

Please see MixSIH page for detailed explanation.

Output\_file1:

This contains the haplotypes and these probabilities of the target individual for each SNP fragment.

Because we use sliding-window calculation, a SNP fragment appears many times.

Format of the file is

<SNP fragment name>

<haplotype1\_1> , <haplotype1\_2> , <probability of haplotype1>

<haplotype2\_1> , <haplotype2\_2> , <probability of haplotype2>

...

Example of the file is as follows.

frag1

001 , 110 , 0.100

000 , 111 , 0.899

frag2

0001 , 1110 , 1.000

frag3

00000 , 11111 , 1.000

frag3

00000 , 11111 , 1.000

frag3

00001 , 11110 , 0.670

00011 , 11100 , 0.330

PHASE\_file1:

This is a temporal file to create input file for PHASE.

PHASE\_file2:

This is a prefix of the output files of PHASE.

N:

N is the number of individual genotypes.

W:

W is the sliding-window width.

We use W=5 for default setting.

In the second step, CSP for each SNP fragment are calculated using the results of CSP1.rb.

Usage:

ruby CSP2.rb <Output\_file1> <Fragment\_file> <Output\_file2> <W>

Example of running CSP2:

ruby CSP2.rb out/csp1.txt example/fragment.txt out/csp2.txt 5

Output\_file1:

This is the output file of CSP1.rb.

Output\_file2:

This contains the CSP values for each SNP fragment.

Format of the file is

<SNP fragment name> <CSP>

## PEATH

Manual: <https://github.com/jcna99/PEATH>

To run PEATH, use the following command:

./PEATH <input\_file> <output\_file> (param)

<input\_file> is an input matrix for sequence reads and

<output\_file> contains phased haplotype.

**Data Sets:**

We used three data sets for experiments.

1. Fosmid dataset (Duitama et al. 2012) which has been widely used to assess and compare SIH algorithms.
2. Simulated dataset which was generated based on Fosmid data.
3. HuRef dataset (Levy et al. 2007) which has been the most widely used in SIH related articles.

## SDhaP

Manual: <https://sourceforge.net/projects/sdhap/>

To run SDhaP, use the following command

./hap /home/local/altair/Fosmid/chr1.matrix.SORTED /home/local/altair/Fosmid/phase.txt

The second argument is the input file and the third argument is the output file containing the phased haplotypes.

Input file format:

The input file format for the diploid case is as follows

Number of reads

Number of columns

Number of contiguos segments, Read identifier, Position of first SNP segmen,t Continuous bases in read, Position of next SNP segment, Continuous bases in read ..... Quality scores (in fastq format)

Example:

SDhaP(diploid)

5568

22801

2 chr22\_SPA9\_8733 2 0 5 0 ==

1 chr22\_SPH2\_1940 3 100 C==

Output file format:

The output file contains phased haplotypes for each haplotype block. the length of the block, the number of reads and the MEC per block. Subsequently, the phased haplotype is printed in the following format:

SNP position first haplotype second haplotype

Example:

Block 1 Length of haplotype block 329 Number of reads 93 Total MEC 77

2 0 1

3 1 0

4 1 0

5 0 1

6 1 0

7 0 1

Block 2 Length of haplotype block 50 Number of reads 9 Total MEC 2

1589 1 0

1590 1 0

1591 1 0

1592 1 0

1593 0 1

1594 0 1

1595 0 1

For higher ploidy, there are K phased haplotypes instead of 2.

## WinHap2

Manual: <http://staff.ustc.edu.cn/~xuyun/winhap/WinHAP2.0.htm>

InputFile formats:

One line per genotype, SNPs values are in {0,1,2,?}

0 - homozygous SNP with major allele

1 - homozygous SNP with minor allele

2 - heterozygous SNP

? - missing data

OutputFile formats:

Two haplotypes per genotype.

One line per haplotype, SNPs values are in {0,1}

0 - major allele SNP

1 - minor allele SNP

Sample input and output:

The input file: genotypes.in contains 9 genotypes each with 96 SNPs.

The output file: haplotypes.out is a result phased by WinHAP for genotypes.in, which contains 18 haplotypes each with 96 SNPs.

head genotypes.in

00202001011211100?020000212201000022200200100?20021002002020002202220201002100000112222010211200

202202200111112220220000010020202022100?00?02200002000020022022000000011001?00000111222011111?22

202000220111112022020000010022002020?00200100?20001002022012002000000011002100000212002012211120

0010?0020112212002202000012201200000000122100020201002002020002000000011000102000222220012221202

00122220011211200220100001120220000000020020000020200000202002210111211100?100022?211220120?2221

0012?2200112212002101000011101200000200122100?00202000000000002202222211002102000221110012202221

00001002021212100202200021220200101000022222200220000?020022020202220012220100000021222012111110

002220011120212?0?202022212202200000?00?0010120020200000202002010122221200?102022112201012011202

00022001112222200?020022110000202020?00122121002102000012012010202220211001100022111201011211222

## WhatsHap

Manual: <http://staff.ustc.edu.cn/~xuyun/winhap/WinHAP2.0.htm>

**Input data requirements**

WhatsHap needs correct metadata in the VCF and the BAM/CRAM input files so that it can figure out which read belongs to which sample. As an example, assume you give WhatsHap a VCF file that starts like this:

*##fileformat=VCFv4.1*

*#CHROM POS ID REF ALT QUAL FILTER INFO FORMAT SampleA SampleB*

chr1 100 . A T 50.0 . . GT 0/1 0/1

...

**WhatsHap comes with the following subcommands**.

| **Subcommand** | **Description** |
| --- | --- |
| phase | Phase variants in a VCF with the WhatsHap algorithm |
| stats | Print phasing statistics |
| compare | Compare two or more phasings |
| hapcut2vcf | Convert hapCUT output format to VCF |
| unphase | Remove phasing information from a VCF file |
| haplotag | Tag reads by haplotype |
| haplofasta | Write haplotypes in FASTA format from a phased VCF |
| genotype | Genotype variants |

**Creating phased references in FASTA format**

To reconstruct the two haplotypes that a phased VCF describes, the bcftools consensus command can be used. It is part of [bcftools](http://www.htslib.org/). As input, it expects a reference FASTA file and either an indexed BCF or a compressed and indexed VCF file. To work with the uncompressed VCF output that WhatsHap produces, proceed as follows:

bgzip phased.vcf

tabix phased.vcf.gz

bcftools consensus -H 1 -f reference.fasta phased.vcf.gz > haplotype1.fasta

bcftools consensus -H 2 -f reference.fasta phased.vcf.gz > haplotype2.fasta

Here, reference.fasta is the reference in FASTA format and phased.vcf is the phased VCF. Afterwards, haplotype1.fasta and haplotype2.fasta will contain the two haplotypes.