Haplous

Haplotype identification and comparison library

Sirkku Karinen
Computational Systems Biology Lab
University of Helsinki

2012

Introduction

Haplous is a Java- and a Anduril-library for haplotype identification and comparison. It can be used to identify *shared haplotypes* (SH) from data containing phased genotypes of samples. Haplous is not limited *per se* into any measurement technology as long as the data can be transformed into marker sequences that indicate which allele is inherited from maternal, and which from paternal side.

Haplous consists of Java library Haplous.jar, which includes methods for converting data between a set of data formats, haplotype extraction and for haplotype filtering. Haplous produces a list of SHs, and this list is human-readable, but can also be read using methods in the library.

Haplous.jar is wrapped into Anduril components, which makes it easy to connect parts of the analysis into other analysis tools in one pipeline. We also provide an example Anduril pipeline for users to get familiar with Haplous and its capabilities.

Basic functionalities

Haplous uses phased haplotypes and predicts the SHs by comparing consecutive SNP alleles with each other using a fixed-sized sliding window with user adjustable parameters for window-size (w), mismatches (m) and length of identical regions (l). The window of w markers is slid over the haplotyped SNP alleles and windows having at most m differences are identified as SHs. Shorter identical regions

of *l* markers are identified as SHs as well. Missing markers are treated as matches. Haplous extracts SHs separately from maternal and paternal chromosomes from each sample, therefore two SHs found from the same region of same sample reveals the location of homozygous regions.

The running time and the memory requirements of Haplous grow polynomially with the number of samples and linearly with the number of markers. Therefore, Haplous is applicable in genome-wide studies in rare diseases, where the number of samples is less than 500. For larger sample sizes, Haplous allows user to define those samples that are used as a reference in the comparison, and the rest of the samples are skipped. This is a very useful option in case-control settings, since Haplous can be used to compare cases to controls, but skip the control-case and the control-control comparisons. Additionally, each sample can be analyzed independently from each other, which enables parallelization of each run. Anduril automatically parallelizes the execution.

Haplous gives an estimate for informativeness of each SH. This informativeness describes how rare given SH is and is defined as a joint probability of the alleles in SH estimated from the allele frequencies by multiplying allele frequencies a_i from the first marker of SH (i) to the last marker (n), *i.e.*, $\prod_i^n P(a_i)$. User may set a threshold (t) for informativeness. In such case only those SHs that have informativeness below t are selected to the results. The value of informativeness is between zero and one. Zero denotes that the alleles of SH could never be seen in the population, while one means that alleles of SH are always observed in that region and thus the region is completely uninformative.

Simply scanning all SHs that meet the criteria would produce a huge list of SHs that are abundant in the population but not particularly interesting regarding the phenotype in question. To find the interesting SHs, we make use of the expert knowledge of the user: the user defines the rules that are used to decide which SHs are interesting. The rules was used to define the features that an interesting SH needs to have, and these features are defined for cases and controls separately. If a SH has these features, it is considered interesting. The rules are set as thresholds for number of cases or controls that share the SH. Pseudo code for the algorithm is:

```
SH_INTERESTING(SHs):

cases, caseHet, caseHom, caseOperator  = USER_INPUT
controls, controlHet, controlHom, controlOperator = USER_INPUT

FOR sh in SHs DO
  IF evaluate(sh, cases, caseHet, caseHom, caseOperator) AND
  NOT evaluate(sh, controls, controlHet, controlHom, controlOperator) THEN
    save(sh)
  END
END

function evaluate(data SH, string samples, int het, int hom, enum operator)

  IF operator == AND THEN
    IF numOfHeterozygous(samples, SH) >= het AND
    numOfHomozygous(samples, SH) >= hom THEN
      return TRUE
    END

  ELSE IF operator == OR THEN
    IF numOfHeterozygous(samples, SH) >= het OR
    numOfHomozygous(samples, SH) >= hom THEN
      return TRUE
    END

  ELSE
    return FALSE
  END
```

Briefly, the inference algorithm takes the thresholds and list of cases and controls as an input, calculates the number of cases and controls sharing each SH, and evaluates whether a SH, has the features of interesting SH taking into account both cases and controls. These evaluations for cases and controls are produced with the same function but the return value is negated for the control rule. The rules follow a natural deduction of which one example could be: “The SH is interesting if it is shared by one or more homozygous case samples and not shared by any control samples.”

Scanning of SHs may produce considerably large numbers of regions that are all almost equally promising for further studies. Haplous gives scores for SHs, which are stored in a file with information about the range, score and samples sharing the particular haplotype. This allows straightforward identification and post-processing of the most interesting homozygous and heterozygous chromosomal regions.

The score calculated by Haplous emphasizes the number of cases and controls that share the haplotype as well as the length of the SH. The score is calculated according to the following formula: $M(C_a\text{-}C_o)$,

where M is the number of markers in the chromosomal region of SH, C_a is the number of times cases share the SH and C_o number of times controls share the SH. Note that in the case of homozygous loci, each homozygous sample shares the SH twice, which increases either C_a or C_o. For an underlying chromosomal region a score is assigned similarly $M_r(C_{ar}\text{-}C_{or})$, where C_{ar} is the number of cases and C_{or} the number of controls having any SH in a given marker. M_r is the number of markers in the chromosomal region that get the same score from $(C_{ar}-C_{or})$. The upper or lower limits of the scores depend on the parameter values.

When to use Haplous

Haplous is targeted for identical by descent (IBD) analysis. Haplous identifies the haplotypes shared by multiple individuals instead of IBD regions between two samples. With unrelated cases and controls, Haplous can be used for as large as 1000 sample cohorts. When the mode of inheritance is unclear or there are possible phenocopies, Haplous can substitute linkage analysis in pedigrees. Haplous can also identify compound-heterozygous regions, which is highly useful especially in rare diseases.

Haplous components for Anduril

Please, read component documentations for details. This document gives only a very brief overview to these components. The package includes also an example Anduril pipeline script examplePipeline.and.

SNPHaplotypeMatrix

Transforms various data formats into suitable form for Haplous.

HaplotypeScan

Scans SHs from one or two haplotype matrices.

HaplotypeFilter

Filters SHs according to given rules

HaplotypeScore

Gives a score to the SHs found from the data.

HaplotypeSamples

This component:
1) calculates the number of times samples are sharing the haplotype,
2) lists sharing samples,
3) returns haplotype length distribution (full length, not SHs that are spliced)
4) returns SH length distribution

Additional components

Text2CSV or CSVCleaner

Component that transforms text files into CSV format. Text2CSV is used in the example script, and it is highly useful when further processing the Haplous results.

Haplous function for Anduril

HaplotypeAnnotator

Annotates the chromosomal regions covered by the SHs. It uses either rsIDs of the SNPs or the locations in the given map-file.

Input data

Haplous reads data from unphased .ped format, and from phased Merlin .chr format. A Plink software is a good tool for producing needed input files.

Output data

The results have one row for each SH. Because SHs are compared against one sample, direction of comparison plays a role in the results. If sample A shares a SH with samples B and C, it does not indicate that also B and C share a SH in the same chromosomal region. This is due to mismatches and missing data. The SH extraction and SH filtering (HaplotypeScan, HaploFilter -components) produces these SH files. The result file does not have headings, but the data columns for each row are following:

1. Chromosome
2. Name of the sample that was used for comparing SHs
3. ALLELE1 or ALLELE2 separating the maternal and paternal haplotypes.
4. SH start marker
5. SH end marker
6. Samples that share the SH. Each sample is given by sample name and ALLELE1 or ALLELE2 and separated by semicolon.

Scored haplotypes (produced by HaplotypeScore-component) have the same columns, except the score is in the first column, and the SHs are sorted in descending order.

HaplotypeScore component produces also scores for regions. This format is a CSV file without the column headers. Region scores are sorted according to region positions. This allows easier plotting of the results (in Anduril using Plot2d-component). Data in the columns are:

1. Chromosome
2. Start marker index
3. End marker index
4. Score