Unfavorable Haplotype Finder

Software Tool

Jeremy T. Howard[1]
Francesco Tiezzi[1]
Christian Maltecca[1]

[1]North Carolina State University, Raleigh, NC, USA

Contents

Introduction

The Unfavorable Haplotype Finder program is method that is designed to identify haplotypes contained within an run of homozygosity (ROH) that reduce the performance of an animal, but are not lethal. The overall objective is to identify these haplotypes across a variety of traits and the ones that are have a consistent unfavorable effect across multiple traits would be great candidates to use in mating designs in tandem with lethal haplotypes. The methods are described in detail in Howard et al. (2017).

The reference for the Unfavorable Haplotype Finder program:

- Howard, J.T., F. Tiezzi, Y. Huang, K. A. Gray & C. Maltecca. A heuristic method to identify runs of homozygosity associated with reduced performance in livestock. Journal of Animal Science 95 (10), 4318-4332.

Overview of Program

The program is ran in three stages and each one is described below:

Stage 1

- Step 1: Tabulate Means of non-ROH and unique ROH for sliding windows (start with largest desired length).

- Step 2: Combine nested windows

Before (each haplotype contains same set of animals):

Start	End	Haplotype
132	182	22002220002200222200000222220220202222022000200202
133	183	20022200022002222000002222202202022220220002002020
134	184	00222000220022220000022222022020222202200020020200
135	185	02220002200222200000222220220202222022000200202000
136	186	22200022002222000002222202202022220220002002020000
137	187	22000220022220000022222022020222202200020020200002
138	188	20002200222200000222220220202222022000200202000022
139	189	00022002222000002222202202022220220002002020000220
140	190	00220022220000022222022020222202200020020200002202

After:

Start	End	Haplotype
132	190	220022200022002222000002222202202022220220002002020200002202

- Step 3: Reduce window size by 5 until lowest desired length is reached.

- Step 4: Combine nested windows .

Before (each haplotype contains same set of animals):

Start	End	Haplotype
614	656	200020220022002222200000200022020002000020
614	646	2000202200220022222000002000 2202
614	651	20002022002200222220000020002202000 20
614	661	200020220022002222200000200022020002000020020 22
614	671	20002022002200222220000020002202000200002002022220202 20002
614	666	2000202200220022222000002000220200020000200202220202
614	641	2000202200220022222 00000200

After:

Start	End	Haplotype
614	641	2000202200220022222 00000200

Stage 2

- Determine the significance of each window that passed Stage 2 using a model that allows:
  - 1.) Fixed Environmental Effects
  - 2.) Additive effect of animal based on pedigree
  - 3.) Permanent effect of animal
- This step can be done with any program, but is parallelized in this program to make it faster.
- A window is kept if a contrast between a unique ROH versus non-ROH is greater than certain cutoff.

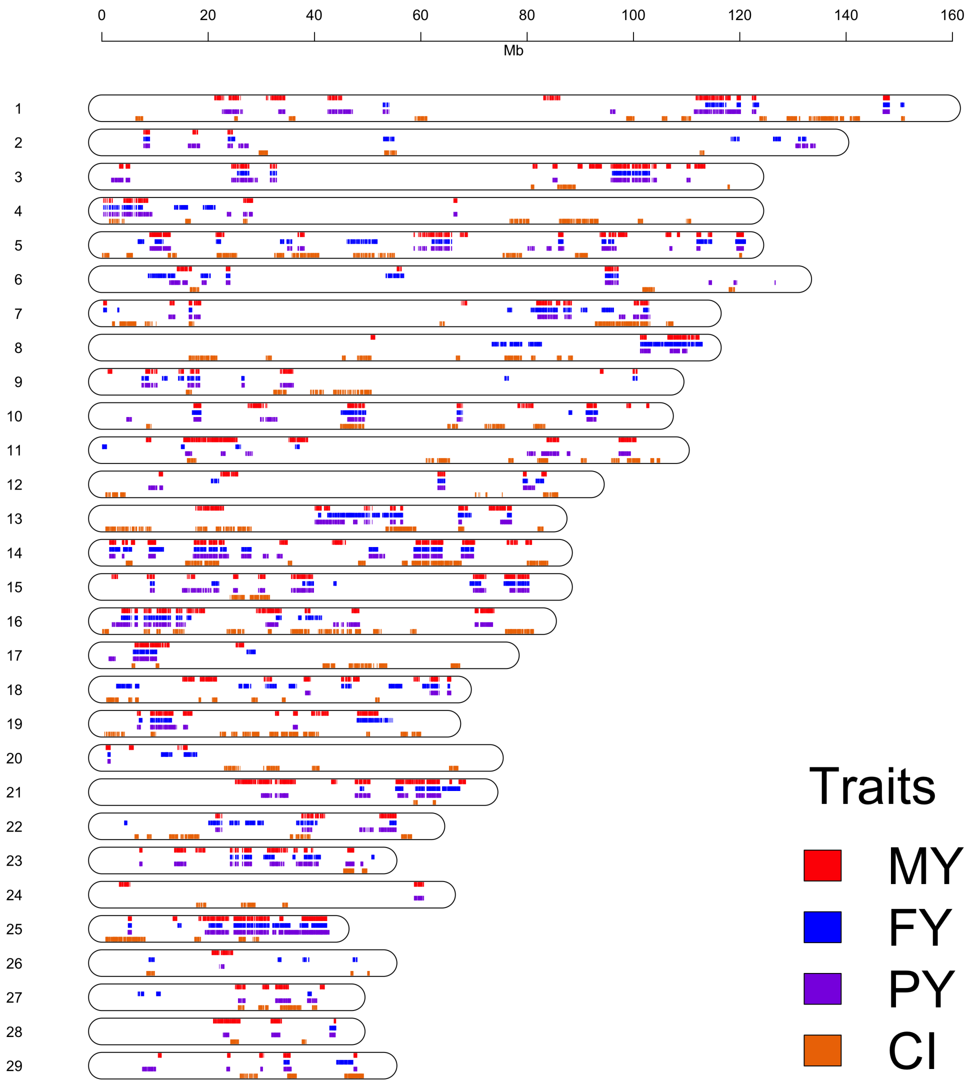
Stage 3

- Remove nested windows

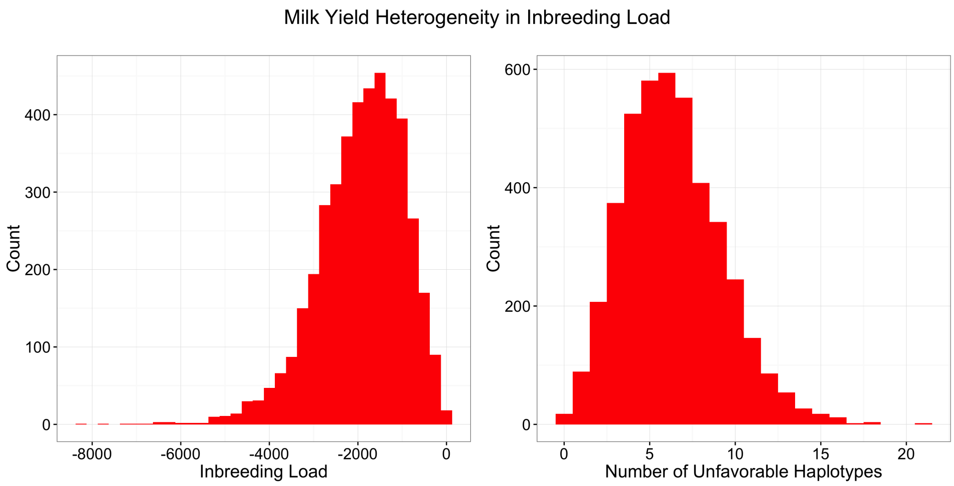
Example: (only keep Window 2)

Window 1	Window 2
Animal	Animal
1	1
2	2
3	3
4	4
5	5
6	6
-	7
-	8
-	9

Outlined below is a plot that illustrates regions that had significant unfavorable haplotypes contained in a ROH across multiple traits. Main take away from the plot is that certain regions of the genome have an effect across multiple traits and represent regions sensitive to inbreeding.



- 186 haplotypes that when in homozygous form (i.e. are a ROH) result in reduced performance across all three traits milk yield traits.
- Low Frequency within set of genotypes utilized:
  - Mean (Minimum - Maximum): 0.032 (0.007 - 0.13)
- Potential to further understand the variation in genetic load across individuals.



Computing Environment

The code is written in C++11 language using object oriented techniques and the application runs on Linux platforms. Furthermore, external libraries are used including Intel MKL and Eigen.

EIGEN Library:
EIGEN is freely available at: Eigen Site
Once at the site, one just needs to download the latest stable release package and uncompress it. For example, the current downloaded package is called "eigen-eigen-07105f7124f9.tar". In order to use it you just have to place it in the file where all of the other Haplofinder files are located and uncompress the file. Once the file is uncompressed it will be a folder and using the current release the folder will be called "eigen-eigen-07105f7124f9". This will serve as your path in the make file outlined below.

Intel MKL Library:
Intel MKL is a commercial library and is available for purchase. However, there is an opportunity to obtain the Intel MKL library (for Linux) free of charge for non-commercial use at the following website: Intel MKL Site
The Intel MKL can sometimes be tricky to download and link, but there is a step-by-step protocol within the folders that is downloaded or instruction can be obtained at Intel MKL Guide and depending on the computing system you are running, a guide to linking the Intel MKL libraries can be found at Intel MKL Linking Guide

C++11 Version:
The C++11 standards start being supported in gcc 4.7 or newer. You would install or update to the correct version of gcc using the normal package manager or installer, depending on what type of OS you are using. Some website that can be used a reference include:
gcc helper 1 gcc helper 2

Compiling:
Once both EIGEN, Intel MKL libraries and gcc version 4.4 or newer have been installed and the folders placed in the directory where all of the Haplofinder source code files are located the last thing you have to do is change the path for EIGEN and Intel MKL libraries.

For lines 13 and 14 you need to change the path which aligns with where the files are located.

After changing the path all you need to do is type "make" and an executable file called "Haplofinder" will be created in your working directory.

Running the Program

At the current time executable files are available only for linux operating environments. To run the program place the following file in a folder:

- Haplofinder.

Prior to running the program you will need to check the permissions of the files and if so make them executable (i.e. type "chmod 755 Haplofinder"). Once the permissions have been changed you need to generate a parameter file and place it in the same folder as Haplofinder. If you are new to the program an example is outlined. The parameter file is read by searching for key words that are capitalized and then followed by a colon. Therefore any other phrase will not utilized.

To run the program type in "./Haplofinder" and then the name of your parameter file.

Program Parameters

A parameter file that specifies all the parameters is outlined below. Not all of them are required for the program to run, which was done to reduce the complexity of running the software. The parameters in bold are not mandatory for the program to run and should be removed. All key words are in capital letters and the parameter(s) specified are separated by spaces. If you want the program to skip over the parameter just insert in values in the key word and the program will no longer recognize the key word. For example change "VAR_PERMANENT" to "VAR_PER!!!!MANENT" and it will skip over that parameter. Only parameters after the key words impact the simulation.

```
-------| Parameter File |-------
THREADS: 4
-------| Files to Read In |-------
MAP_FILE: haplo_map_.txt
PHENO_FILE: haplo_phenotype.txt
GENO_FILE: haplo_genotype.txt
PEDIGREE: Pedigree_File.txt.SRT
-------| Location of Variables |-------
ID: 1
CLASS: 2,3
COV: 4
PHENOTYPE: 5
-------| Variance Components |-------
VAR_RESIDUAL: 0.05
VAR_ANIMAL: 0.025
VAR_PERMANENT: 0.0
-------| Running Parameters |-------
CUTOFF: data 1000
UNFAV_DIRECTION: low
WINDOW: 50,45,40,35,30,25,20,15
ONE_SIDED_T_CUTOFF: 2.326
SUBTRACT_MEAN: no
MINIMUM_FREQ: 0.0075
ASREML_CHECK: no
```

A description of all the parameters that can be specified in the program is outlined below along with helpful hints and suggestions. A complete example going from start to finish is outlined in Example 1.

Optimization Parameters

THREADS

Description: - Declares the number of threads used for parallel processing.
Value: - Integer value based on number of cores available.
Usage: - "THREADS: 4".
Type: - Optional. Default is 1.
Note:

- Running the program on a single thread will take a large amount of time. The program is designed to run multiple models at a time and therefore the running time is greatly reduced.

Files to Read In

MAP_FILE

Description:

- The file where SNP location is stored and ordered by chromosome then position. Column 1 is chromosome and column 2 is SNP position with a space delimiter.

Value: - String.
Usage: - "MAP_FILE: haplo_map_.txt".
Type: - Mandatory.
Note:

- Example files are provided as a template and is described in Example 1.

PHENO_FILE

Description:

- The file where ID, phenotype and the variables that are fixed in the model are located. Separator is space delimiter. No animals can have missing datapoints.

Value: - String.
Usage: - "PHENO_FILE: haplo_pheno.txt".
Type: - Mandatory.
Note:

- Example files are provided as a template and described in Example 1. A log file, referred to as "LogFile" is created that prints out how each line is interpreted.

GENO_FILE:

Description:

- The file where ID and genotype are located. Separator is space delimiter. The genotype is represented as a string of 0, 1, or 2 (i.e. 021012). Can't have any missing genotypes. It is assumed that the first row of map file is the first genotype, second row is the second genotype etc.....

Value: - String.

Usage: - "GENO_FILE: haplo_geno.txt".

Type: - Mandatory.

Note:

- Example files are provided as a template and described in Example 1.

PEDIGREE

Description:

- The pedigree file where ID, Sire and Dam is outlined and the delimiter is a space. The ID's can be any format (i.e. alphanumeric or numeric), but they have to be sorted so that parents come before progeny.

Value: - String.

Usage: - "PEDIGREE: Pedigree_File.txt.SRT".

Type: - Mandatory.

Note:

- Example files are provided as a template and described in Example 1. If one is generating variance components with ASREML the sorted pedigree file using the !SORT command can be utilized.

Location of Variables

ID

Description: - Location in phenotype file that has the animal ID variable.
Value: - Integer.
Usage: - "ID: 1".
Type: - Mandatory.

CLASS

Description: - Location in phenotype file that has the class type fixed effects.
Value: - Integer.
Usage: - "CLASS: 2,3".
Type: - Optional.
Note:

- For each class variable the first level is zeroed out to make the coefficient matrix invertible. At the current time the program doesn't check to see if more dependencies exist. Therefore small CG should be removed or aggregated. It is advisable that when you get the variance component estimates you check to see how many levels got zeroed out for each class variable. If more than one effect is zeroed out for each variable this program WILL NOT GENERATE THE CORRECT RESULTS!!! When generating least-square means (LSM) for each unique ROH and non-ROH the class effects are averaged over. No animals can have missing observations.

COV

Description: - Location in phenotype file that has the covariate type fixed effects..
Value: - Integer.
Usage: - "COV: 4".
Type: - Optional.
Note:

- When generating least-square means (LSM) for each unique ROH and non-ROH the average covariate value is utilized in the LSM estimate

---

PHENOTYPE
Description: - Location in phenotype file that has the phenotype.
Value: - Integer.
Usage: - “PHENOTYPE: 5”.

Variance Components

VAR_RESIDUAL
Description: - Residual variance of null model without haplotype effect.
Value: - Double.
Usage: - “VAR_RESIDUAL: 0.05”.
Type: - Mandatory.

VAR_ANIMAL
Description: - Additive genetic variance of null model without haplotype effect.
Value: - Double.
Usage: - “VAR_ANIMAL: 0.025”.
Type: - Mandatory.

VAR_PERMANENT
Description:
- Permanent animal variance of null model without haplotype effect. If absent assumed not included.
Value: - Double.
Usage: - “VAR_PERMANENT: 0.01”.
Type: - Optional.

Running Parameters

WINDOW

Description:

- Window sizes to scan across the genome and starts at largest number.

Value: - Integers separated by a comma.

Usage: - "WINDOW: 50,45,40,35,30,25,20,15'.

Type: - Optional. Default is 50,45,40,35,30,25,20,15.

Note:

- I have played around with adjusting these such as making the step size smaller as the window size decreases and reducing the smallest window and the results stay relatively consistent. This has only been utilized using medium density genotypes (i.e. 60k). My expectation that if you use higher density genotypes it should do a better job at finding the optimal haplotype that best represents the ROH haplotype.

CUTOFF

Description:

- Used to define the mean phenotype of the unfavorable haplotype that you would like to investigate with the full model. You can either specify a specific value or perform multiple (i.e. 1,000) full models with randomly chosen windows across the genome to determine the mean phenotype value that results in a t-value of at least 1.96 and any mean phenotype above or below this depending on the unfavorable direction will be investigated with the full model.

Value:

- Method:
  - value: the users specifies a value that is deemed unfavorable.
  - data: the software determines the threshold by performing the multiple models across a random region of the genome.
- Value:
  - If value was chosen: The users specifies the minimum/maximum mean phenotypic value
  - If data was chosen: The number of full models that the initial scan runs to chose a minimum/maximum mean phenotypic value.

Usage: - "CUTOFF: data 1000".
Type: - Mandatory

UNFAV
Description:
- Direction of the phenotype that is unfavorable. (Example: For the trait Number of Piglets Born Alive the direction is low).
Value: - low or high
Usage: - "DIRECTION: low".
Type: - Mandatory.

ONE_SIDED_T_CUTOFF
Description:
- One sided T-Statistic after running the full model that is regarded as being significant.
Value: - Double
Usage: - "ONE_SIDED_T_CUTOFF: 2.326".
Type: - Optional. Default is 2.326.

SUBTRACT_MEAN
Description: - Useful for simulation purposes with no fixed effects.
Value: - yes or no
Usage: - "SUBTRACT_MEAN: no".
Type: - Optional. Default is no.

MINIMUM_FREQ
Description:
- Minimum number of times a unique ROH is required in order to enter into full model.
Value: - ranging from 0.0 to 1.0
Usage: - "MINIMUM_FREQ: 0.0075".
Type: - Optional. Default is 0.0075.

ASREML_CHECK
Description:

	- This produces the output that is generated by the software and a phenotype file that can be used as input in ASREML. If worked properly the results will be very similar.
Value:	- yes or no
Usage:	- “ASREML_CHECK: no”.
Type:	- Optional. Default is no.

Output Files

The program output a file called Stage2_Regions that contains all the information on regions that were declared as significant. Below is a description on what the column heading are describing.

Chromosome: Chromosome haplotype is located.
StartPos: Nucleotide start position of haplotype.
EndPos: Nucleotide end position of haplotype.
StartIndex: Start index in full genotype string of haplotype.
EndIndex: End index in full genotype string of haplotype.
Genotype: Genotype string of unfavorable haplotype.
PhenoMean: Mean phenotype of unfavorable haplotype.
BetaEffect: Beta estimate of unfavorable haplotype.
LSM: Least square mean of unfavorable haplotype.
T-Stat: T-statistic of ROH haplotype versus non-ROH haplotype.

Walkthrough Example

Data Setup

In order to make the process easier I have created a simulated population with all of the datasets already pre-made. The pre-made datasets are:

- haplo_map.txt
  - A file that contains chromosome and position.
- haplo_geno.txt
  - A file that contains ID then genotype as a string.
  - Each row of map file is matched up with genotype string.
  - If multiple traits are being run on a similar set of individuals it is advisable to make a single genotype file. Animals that are not used in the program can still be in the genotype file, but if an animal isn't in the genotype file the program will exit.
- haplo_pheno.txt
  - A file that contains ID then phenotype.
  - It is advisable to make a phenotype file for each unique trait because their can't be any missing datapoints.
  - In the current situation low values are unfavorable.
- pedigreefile.txt
  - A file that contains ID, sire, dam information. It is already ordered so parents come before progeny.
  - If the pedigree needs to be ordered there are multiple options available including the !SORT option in AS-Reml.

When class fixed effects are included in the model the software still is not perfect when issues with small contemporary groups exist and therefore the issues related to confounding between variables may produce wrong results. Therefore as a check it is advisable to ensure that results are similar to the results produced by ASReml by using the "ASREML_CHECK" option.

Null Model Variance Components

Whichever program you are comfortable with you need to estimate variance components for the dataset to use in the full models in order to derive significance for a unique ROH genotype. Below is an example of an input file for AS-Reml ("asreml_null.as" in folder).

```
!Workspase 16384
File
Animal * !P
Pheno
pedigreefile.txt !ALPHA !SORT
haplo_pheno.txt !MAXIT 100 !EXTRA 2 !FCON !DDF 2
Pheno   mu !r Animal
0 0 1
Animal 1
0 0 AINV 1 !GP
```

The residual variance was estimated to be 0.722969 and the additive genetic component was estimated to be 0.204449.

These are the values that will get utilized in the full model.

Double Check with AS-REML

Prior to running the complete analysis and if AS-Reml is available it is preferred to double check with what AS-Reml would get. Below is an example parameter file that outputs a dataframe to use as input in ASREML and estimates to compare with AS-REML ("Parameter_File" in folder).

```
−−−−−−−|          Parameter File          |−−−−−−−
THREADS: 4
−−−−−−−|          Files to Read In        |−−−−−−−
MAP_FILE: haplo_map_.txt
PHENO_FILE: haplo_pheno.txt
GENO_FILE: haplo_geno.txt
PEDIGREE: pedigreefle.txt.SRT
−−−−−−−|       Location of Variables      |−−−−−−−
ID: 1
PHENOTYPE: 2
−−−−−−−|       Variance Components        |−−−−−−−
VAR_RESIDUAL: 0.722969
VAR_ANIMAL: 0.204449
−−−−−−−|       Running Parameters         |−−−−−−−
CUTOFF: data 1000
UNFAV_DIRECTION: low
ASREML_CHECK: yes
```

At the bottom of the LogFile it outlines the estimates that are derived from the software and a file called TestDFa.txt is outputted that has appended the unique ROH genotype string for each animal. The results are outlined below:

ROH ID	Beta Estimate	Least Square Mean	T-stat (non-ROH versus ROH
1	-0.308	1.078	-1.242
2	-0.183	1.203	-0.826
3	-0.924	0.461	-3.602

These results can then be compared with the true model where variance components are re-estimated ("asreml_check.as" in folder). More complex models with class fixed effects can be averaged over in ASReml by adding "!PRESENT" qualifier and then the associated class variables after it.

```
!Workspase 16384
File
Animal * !P
Haplotype * !A !SORT Pheno
pedigreefile.txt !ALPHA !SORT
TestDFa.txt !MAXIT 100 !EXTRA 2 !FCON !DDF 2
Pheno   mu Haplotype !r Animal
0 0 1
Animal 1
0 0 AINV 1 !GP
predict Haplotype !TDIFF
```

The results using AS-Reml are outlined below and are similar:

ROH ID	Beta Estimate	Least Square Mean	T-stat (non-ROH versus ROH
1	-0.307	1.083	-1.24
2	-0.182	1.208	-0.83
3	-0.926	0.464	-3.62

Running Program

Once the program has been tested on the dataset the full analysis can be run. Below is the parameter file. In the log file on around line 164 the phenotype cutoff from the previous run can be utilized (may be slightly different on your run).

```
--------| Parameter File |--------
THREADS: 4
--------| Files to Read In |--------
MAP_FILE: haplo_map_.txt
PHENO_FILE: haplo_pheno.txt
GENO_FILE: haplo_geno.txt
PEDIGREE: pedigreefle.txt.SRT
--------| Location of Variables |--------
ID: 1
PHENOTYPE: 2
--------| Variance Components |--------
VAR_RESIDUAL: 0.722969
VAR_ANIMAL: 0.204449
--------| Running Parameters |--------
CUTOFF: value 1.42227
UNFAV_DIRECTION: low
```

Once the program has finished it will output the regions that were declared as significant into the Stage2_Regions File. The columns have heading to aid in understanding.