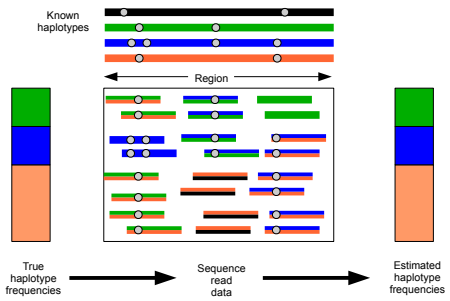
harp
Haplotype Analysis of Reads in Pools

Darren Kessner

March 6, 2013

Introduction

`harp` is a command-line program for estimating the frequencies of known haplotypes from pooled sequence data. `harp` implements an Expectation-Maximization (EM) algorithm to obtain a maximum-likelihood estimate of the haplotype frequencies.



`harp` can be run in two modes:

1. *Single reference.* Pooled reads have been mapped to a single reference, with the assumption that the haplotypes represent strains that are identical to the reference except for single-nucleotide variants. In this mode, `harp` needs:
    - single BAM file with mapped reads
    - reference sequence in FASTA format
    - SNP file in DGRP format (comma-separated table containing variants by genomic position and strain – see Examples)
2. *Multiple reference.* Pooled reads have been mapped to multiple references, one for each strain/species of interest. In this mode, `harp` needs:
    - list of BAM files (one for each mapping/species)
    - corresponding list of reference sequences in FASTA format

Citation

Details and evaluation of the method can be found in the paper:

Kessner D, Turner TL, Novembre J. 2013. Maximum Likelihood Estimation of Frequencies of Known Haplotypes from Pooled Sequence Data. Molecular Biology and Evolution (accepted January 2013, Open Access)

`http://mbe.oxfordjournals.org/cgi/content/abstract/mst016`

Usage

`harp` is written in C++ and has been tested on OSX and Linux. The program includes multiple functions, which can be run as follows:

```
harp <function_name> [arguments]
```

Running `harp` with no function name or arguments will print the usage information, including the list of available functions and parameters. Running `harp <function_name>` with no arguments will give the list of required parameters for that function.

Parameters are *(name, value)* pairs, and may be specified on the command line:

```
--name value
```

or in a configuration file:

```
name = value
```

Typical usage of `harp` proceeds in two stages:

1. *Haplotype likelihood calculation.* The harp function `like` (single reference) or `like_multi` (multiple reference) creates an intermediate binary file (extension `.hlk`) containing the computed haplotype likelihoods.
2. *Frequency estimation.* The harp function `freq` uses the computed haplotype likelihoods (`.hlk` file) to perform the frequency estimation.

This allows the haplotype likelihoods to be calculated in larger regions, with the frequency estimation performed in smaller (possibly overlapping) windows.

Examples

Included in the `harp` package are example files to demonstrate usage of `harp`. The examples use a relative path to the `harp` binary (`../../bin/harp`) so that they may be run directly from the example directory after unzipping the package.

Example 1: Single Reference

The following files may be found in `examples/example1_single_reference`:

- `example1.bam`: BAM file containing simulated mapped pooled reads. `example1.bam.bai` is an index file for faster access, created by `samtools`.
- `example1.true.freqs`: True haplotype frequencies under which the reads were drawn in the simulation used to create this example. In this example, there are 4 haplotypes at frequences .4, .3, .2, .1.
- `example1.actual.freqs`: Actual haplotype frequencies in the BAM file. Note that these differ from the true frequecies, as they do in an experimental setting.
- `dmel_chr2L.fasta`: Reference sequence in FASTA format (first 100kb of *D. melanogaster* chromosome 2L). `dmel_chr2L.fasta.fai` is an index file for faster access, created by `samtools`.
- `snps.txt`: SNP file in DGRP format. In the first row, `harp` expects the first field to be the chromosome identifier, and the second field to be "Ref" (reference), followed by strain (haplotype) identifiers. For subsequent rows, `harp` expects the first column to be position, followed by the reference base, and then the other haplotype bases. `harp` currently ignores the trailing "Coverage" field and comma, and handles files without these. `snps.txt.idx` is an index for faster access, which can be created with the `index_snp_table` tool included in the `harp` package.
- `harp_like.config, harp_freq.config`: Example `harp` configuration files for the haplotype likelihood calculation and haplotype frequency estimation, respectively.
- `run_example1.sh`: Shell script containing the `harp` commands for this example.

First we perform the likelihood calculation with this command line:

```
../../bin/harp like --bam example1.bam --region 2L:30001-50000
    --refseq dmel_chr2L.fasta --snps snps.txt --stem example1
```

We are using the `like` function, which has 4 required parameters:

- `bam`: filename of the BAM file
- `region`: genomic region under consideration
- `refseq`: filename for the reference
- `snps`: filename for the SNP file

We are also using an optional parameter `--stem example1`, which `harp` uses as the filestem for constructing output filenames (e.g. `example1.hlk`. By default, `harp` uses the BAM filename and region to create the filestem.

Alternatively, we could have specified the parameters in a configuration file (`harp_freq.config`), and we specify this on the command line:

```
../../bin/harp like -c harp_like.config
```

In either case, `harp` creates the file `example1.hlk`, which contains the computed haplotype likelihoods.

Next, we perform the haplotype frequency estimation:

```
../../bin/harp freq --hlk example1.hlk --region 2L:30001-50000
```

or:

```
../../bin/harp freq -c harp_freq.config
```

This creates the file `example1.freqs` with the estimated haplotype frequencies.

These commands are collected in the shell script `run_example1.sh`, which you can run directly to avoid typing in the above commands.

Example 2: Multiple Reference

The following files may be found in `examples/example2_multiple_reference`:

- `example1.reads.fastq`: Contains the raw reads and base quality scores, in FASTQ format.
- `example2.true.freqs, example2.actual.freqs`: The true haplotype frequencies for the simulated data, and the actual realized frequencies of the reads, as in Example 1.
- `ref?.fasta`: Reference sequences in FASTA format.
- `ref?.bam`: BAM mapping file for each reference sequence. (These were created by mapping the reads in `example1.reads.fastq` to each reference sequence, using `bwa` and `samtools`).
- `refseqlist.txt, bamlist.txt`: Text files containing the lists of reference sequences and BAM files, respectively.
- `run_example2.sh`: Shell script containing the commands for this example.

In this example, there are 4 known haplotypes in the pool, as well as 1 unknown haplotype. Because of the unknown, we first analyze our read data to calculate parameter values for the haplotype likelihood filter, using the `qual_hist_fastq` tool included in the `harp` package:

```
../../bin/qual_hist_fastq 75 example2.reads.fastq example2.harp_like_multi
```

This creates the file `example2.harp_like_multi.config`, which is a text file containing three parameters needed for haplotype likelihood filtering. Now we can calculate haplotype likelihoods:

```
../../bin/harp like_multi --refseqlist refseqlist.txt --bamlist bamlist.txt
    --stem example2 -c example2.harp_like_multi.config
```

Note that we passed the configuration file directly to `harp`, but we could have passed the parameters on the command line or appended them to an existing configuration file.

Finally, we perform the haplotype frequency estimation:

```
../../bin/harp freq --hlk example2.hlk
```

which creates the file `example2.freqs` with the estimated haplotype frequencies.

These commands are collected in the shell script `run_example2.sh`, which you can run directly to avoid typing in the above commands.

Function and Parameter Summary

Running `harp` with no arguments will print the usage text, which includes a summary of functions and parameters available in `harp`. This text is reproduced here for convenience:

```
harp: Haplotype Analysis of Reads in Pools

Usage: harp function_name [args]

Available functions:
  bqi : calculate empirical base quality interpretation from monomorphic sites
  freq : estimate haplotype frequencies
  freq_stringmatch : estimate haplotype frequencies using simple string matching
  like : calculate haplotype likelihoods (create .hlk file)
  like_multi : calculate haplotype likelihoods from multilple refseqs/alignments
               (create .hlk file)
  likedump : print info from likelihood (.hlk) file

For required parameters, use 'harp function_name' with no arguments.

Parameters may be specified on the command line:
  --name value    (e.g. --bam filename.bam)
or in a configuration file:
  name = value    (e.g. bam = filename.bam)

Parameters:

Command line only:
```

```
  -c [ --config ] arg (=config.harp) filename of configuration file

Input:
  -b [ --bam ] arg      BAM filename
  --bamlist arg         text file list of BAM files
  -r [ --region ] arg   region (e.g. 2L:1001-2000)
  --refseq arg          refseq filename
  --refseqlist arg      text file list of refseq filenames
  --snps arg            SNP filename

Output:
  --stem arg                output filename stem [default: generated from BAM
                            filename and region]
  --out arg                 directory for additional output files [default:
                            (stem).output]
  --hlk arg                 haplotype likelihood filename (.hlk) [default:
                            (stem).hlk]
  --freqs arg               haplotype frequencies filename (.freqs) [default:
                            (stem).freqs]
  -v [ --verbose ]          verbose output
  --compute_standard_errors compute standard errors

Likelihood calculation (like):
  --bqi arg                       base quality interpretation filename
  --min_mapping_quality arg (=15) minimum mapping quality

Likelihood multiple-reference calculation (like_multi):
  --logl_min_zscore arg (=-inf) filter out reads below specified minimum
                                log-likelihood
  --logl_mean arg (=0)          required with logl_min_zscore
  --logl_sd arg (=1)            required with logl_min_zscore

Frequency estimation (freq):
  --window_step arg                 window step size [default: length of
                                    region, i.e. single window]
  --window_width arg                window width [default: window_step]
  --em_iter arg (=30)               EM iteration count
  --em_converge arg (=0)            EM convergence threshold
  --em_random_start_count arg (=0)  number of additional random starts
  --em_random_start_alpha arg (=1)  symmetric dirichlet parameter for random
                                    initial estimates
  --em_random_start_seed arg (=0)   seed for random initial estimates
  --em_min_freq_cutoff arg (=0.0001) EM minimum frequency cutoff
  --haplotype_filter arg            haplotype filter (0 == Ref) [e.g.
                                    1-3,5-7,10]
```

```
String matching requency estimation (freq_stringmatch):
  --max_mismatch arg (=3) maximum # of mismatches allowed
```

Building

Source code for harp is hosted on github:
`https://github.com/dkessner/harp`

`harp` uses the `samtools` API for accessing BAM files. Before building `harp`, you will need to download and build `samtools`. This can be done with the script `get_samtools.sh` in the source directory.

`harp` makes extensive use of the Boost C++ libraries, as well as the Boost build system, i.e. you will need to install Boost before building `harp` . To build, run `bjam` from the project directory, where it will find `Jamroot` for the build instructions. Executables will be placed in the `bin` subdirectory.