README for SelectionHapStats

Nandita Garud
ngarud@stanford.edu
Stanford University
November 2014

SelectionHapStats is a repository of Python scripts to identify genomic loci under natural selection and R scripts to visualize the genomic signatures at loci under selection. This work is described briefly below and in greater detail in the arXived paper: Recent selective sweeps in North American *Drosophila melanogaster* show signatures of soft sweeps (http://arxiv.org/abs/1303.0906).

Introduction:

Evolutionary adaptation is a process in which beneficial mutations increase in frequency in response to selective pressures. If these mutations were previously rare or absent from the population, adaptation should generate a characteristic signature in the genetic diversity around the adaptive locus, known as a selective sweep. Such selective sweeps can be distinguished into *hard* selective sweeps, where only a single adaptive mutation rises in frequency, or *soft* selective sweeps, where multiple adaptive mutations at the same locus sweep through the population simultaneously. Here we present a new statistical method that can identify both hard and soft sweeps in population genomic data.

Our statistical test for detecting hard and soft sweeps is based on the reasoning that selective sweeps will elevate the frequencies of the genetic material surrounding the adaptive mutation, also known as a *haplotype*. The increase of haplotype population frequencies in both hard and soft sweeps can be captured using haplotype homozygosity. If p_i is the frequency of the i^{th} most common haplotype in a sample, and n is the number of observed haplotypes, then haplotype homozygosity is defined as $H1 = \Sigma_{i=1,...n} p_i^2$. However, as we demonstrate in our paper, H1 has better ability to detect hard sweeps than soft sweeps. To have a better ability to detect hard and soft sweeps using homozygosity statistics, we developed a modified homozygosity statistic, $H12 = (p_1 + p_2)^2 + \Sigma_{i>2} p_i^2 = H1 + 2p_1p_2$, in which the frequencies of the first and the second most common haplotypes are combined into a single frequency.

In order to gain intuition about whether a sweep identified with H12 can be easily generated by hard sweeps versus soft sweeps under several evolutionary scenarios, we developed a new homozygosity statistic, H2/H1, where $H2 = \Sigma_{i>1} p_i^2 = H1 - p_1^2$ is haplotype homozygosity calculated using all but the most frequent haplotype. We expect H2 to be lower for hard sweeps than for soft sweeps because in a hard sweep, only one adaptive haplotype is expected to be at high frequency and the exclusion of the most common haplotype should reduce haplotype homozygosity precipitously. When the sweep is soft, however, multiple haplotypes exist at high frequency in the population and the exclusion of the most frequent haplotype should not decrease the haplotype homozygosity to the same extent. Conversely H1, the homozygosity calculated using all haplotypes, is expected to be higher for a hard sweep than for a soft sweep as we described above. The ratio H2/H1 between the two should thus increase monotonically as a sweep becomes softer, thereby offering a summary statistic that in combination with H12 can be used to test whether the observed haplotype patterns are likely to be generated by hard or soft sweeps.

Note that we intend H2/H1 to be measured near the center of the sweep where H12 is the highest, otherwise further away from the sweep center mutations and recombination events will decay the haplotype signature and hard and soft sweep signatures may look indistinguishable. In order to assess if a sweep's H12 and H2/H1 values can be easily generated under hard and soft sweeps, the H12 and H2/H1 values generated in extensive simulations under a broad range of evolutionary scenarios of sweeps of varying softness must be compared with the observed values of H12 and H2/H1.

Scripts in this repository:
The following scripts can be found in this repository and can be used to calculate H12 and H2/H1, identify H12 peaks in the genome, and visualize the signatures of selection.

H12_H2H1.py calculates H12 and H2/H1 using genomic data as an input.
H12peakFinder.py identifies peaks of H12 using the chromosome-wide scan output of H12_H2H1.py
H12_viz.R visualizes the chromosome-wide H12 scan output from H12_H2H1.py
hapSpectrum_viz.R visualizes the haplotype frequency spectra in specific defined windows.
hapData_viz.R visualizes the haplotype diversity and missing data structure in specific defined windows.

Calculating H12 and H2/H1 with H12_H2H1.py:

Preprocessing steps:
H12_H2H1.py takes in an input file consisting of all the SNPs in a population sample (see below for the format). Recommended preprocessing steps include:

1. Convert any non-ATGC site into an 'N' (i.e. Y should be changed to an N) – this is a requirement because the script cannot handle any non-ATGCN letter.
2. Remove whole strains that have very poor data quality and high amounts of missing data. Lots of missing data can result in artificially inflated H12 values.
3. Remove any closely related strains such as siblings or first-cousins
4. Remove any invariant sites

Input file format:
The input file is a line-by-line listing of all the coordinates on the chromosome where there is a polymorphism and the nucleotide states in each individual in the sample. For example: CoordX, ind1, ind2, ind3,... etc. The file must be sorted from smallest coordinate to largest coordinate.

In this documentation, I use an example file from chromosome 3R from the Drosophila Genetic Reference Panel (DGRP), with a sample size of 145 individuals: Chr3R_DGRP_n145_inputData.txt. Because this file is relatively large and takes time to process with the H12_H2H1.py script, I also include a truncated example input file: Chr3R_DGRP_n145_truncatedFile.txt.

The input file should look something like this:
3948,A,A,A,N,T,A,A,A,T,…etc
7387,A,N,G,G,G,G,G,A,G,…etc
19921,T,G,T,T,T,G,G,T,G,T,…etc

Etc..

Required input arguments:

- Input file
- Sample size

Optional input arguments (though, it is highly recommended you specify the values most appropriate for your data rather than rely on default values):

- --outFile, -o: Output file name (default is stdout).
- --window, -w: Analysis window size in terms of SNPs. Note that if you specify an even integer, then 1 additional SNP will be added as the center SNP (default is 400).
- --jump, -j: Distance between the centers of analysis windows (in terms of SNPs) (default is 50).
- --distanceThreshold, -d: Tolerable hamming distance between unique haplotypes. Haplotypes with this difference or less will be combined as a single haplotype (default is 0).
- --singleWindow, -s: Calculate H12 in a single analysis window instead of genome-wide, centered around a specified coordinate. If the coordinate is not in the file or is too close to the ends of the files such that a definable window cannot be created, then an error is given.

Example run of the script:

```
python H12_H2H1.py Chr3R_DGRP_n145_inputData.txt 145 –o
Chr3R_H12_output_400_50.txt –w 400 –j 50 –d 0
```

To calculate H12 in a single window defined at a particular coordinate:

```
python H12_H2H1.py Chr3R_DGRP_n145_truncatedFile.txt 145 –o
Chr3R_H12_output_400_50_coord53034.txt  –w 401 –j 50 –d 0 –s 53034
```

(Here the coordinate is 53034)

To view the help page:

```
python H12_H2H1.py –help
```

Output of H12_H2H1.py:

- Field 1: Center position of the analysis window
- Field 2: Left coord of analysis window
- Field 3: Right coord of analysis window
- Field 4: K (number of unique haplotypes)
- Field 5: Haplotype frequency spectrum (The number of haplotypes in each haplotype group is separated by ',')
- Field 6: DGRP strain numbers in each of the haplotype groups, separated by '( )' . The strain numbers correspond to the order in which the genotypes are entered in the input file (i.e. 1= ind1)
- Field 7: H1 (same as regular homozygosity)
- Field 8: H2
- Field 9: H12
- Field 10: H2/H1

Identifying H12 peaks with H12peakFinder.py:

To assess whether the observed H12 values calculated with H12_H2H1.py are unusually high as compared to expectations under neutrality, the highest H12 value expected under realistic demographic models can be estimated using a program such as *msms.* This critical value of H12 ($H12_o$) can be then used to identify unusually high H12 values relative to neutral expectations.

To call individual sweeps, first all windows with H12 > $H12_o$ are identified. Then consecutive windows with H12> $H12_o$ are grouped together and considered to belong to the same 'peak'. The window with the highest H12 value among all windows in a peak is used to represent the H12 values of the entire peak.

Required input argument:

- Input file -- Use the output of a chromosome-wide scan of H12 and H2/H1 computed with H12_H2H1.py.

Optional input argument:

- --threshold, -t: The critical H12 value used to identify extreme outliers. If a negative number or no number is inputted, then the median H12 value is calculated from the input data and used as a critical value.

Example run of the script:

```
python H12peakFinder.py Chr3R_H12_output_400_50.txt -o
Chr3R_H12_peaks.txt -t 0.02
```

To view the help page:

```
python H12peakFinder.py –help
```

Output of H12peakFinder.py:

- Fields 1-10 are the same as the output from H12_H2H1.py for the analysis window in the H12 peak with the highest H12 value.
- Field 11: Smallest edge coordinate of the peak
- Field 12: Largest edge coordinate of the peak

Recommended post-processing steps:

1. Remove any H12 analysis windows and peaks overlapping regions with low recombination rates. These regions can result in artificially high H12 values. Recombination maps can either be estimated from the data, but more preferably, can be measured in an independent sample using crossing-over methods. The demographic models used to estimate the critical H12 value used as an input for the H12peakFinder.py script should also use this recombination rate threshold.
2. Focus on only the most empirically extreme peaks – i.e. focus on only the top 50 peaks genome-wide, depending on the genome size. There will be many low-level peaks, which may be a consequence of some unaccounted artifact in the data.
3. Check if there is any correlation between the locations of peaks with locations of inversions in the data. Are there any other confounding variables that could result in peaks?

4. Check if any of your peaks could be due to lots of missing data. Inadequate control for missing data could result in spurious peaks.

Visualizing the H12 scan and H12 peaks with H12_viz.R:

H12_viz.R is an R script which generates a visualization of the output of H12_H2H1.py and H12peakFinder.py. In Figure 1, I have highlighted the top 10 peaks on chromosome 3R of the DGRP data in order to focus on the most extreme selection events and to avoid picking up any noise.

Required input arguments:

- H12 scan (output from H12_H2H1.py)
- Peaks (output from H12peakFinder.py)
- Output file name (where the PDF should print to)
- The top n number of peaks to highlight (i.e. top 10)

Example run of the script:

```
Rscript H12_viz.R Chr3R_H12_output_400_50.txt Chr3R_H12_peaks.txt
H12Scan.pdf 10
```

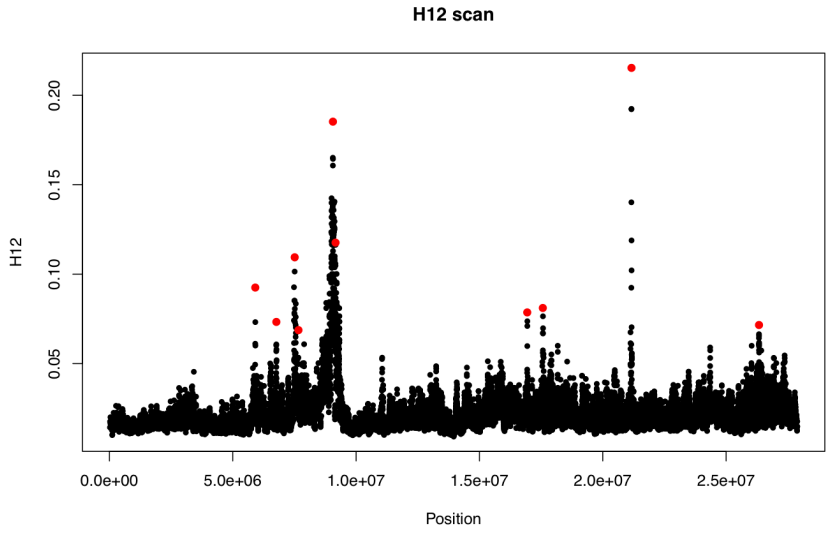

Figure 1: H12 scan for chromosome 3R of the DGRP data. Each dot represents the H12 value measured in a 400 SNP analysis window. The red dots highlight the peaks identified by H12peakFinder.py. This figure was generated with H12_viz.R

Visualization of haplotype frequency spectra for top peaks with hapSpectrum_viz.R

To gain intuition as to whether the most extreme candidates for sweeps resemble hard or soft sweeps, we can visually inspect the haplotype frequency spectrum for the analysis window in the H12 peak with the highest H12 value. The script hapSpectrum_viz.R was written in conjunction with Pleuni Pennings at San Francisco State University (pennings@sfsu.edu).

Required input arguments:

- Peaks (output from H12peakFinder.py). The file must be sorted from highest H12 value to lowest.
- Output file name (where the PDF should print to)
- Number of haplotype spectra to visualize (maximum is 10 with this code). Based on the number provided, the top *n* lines from the input file will be used to construct the figure.
- Sample size (the number of individuals in the sample used to calculate H12 and H2/H1)

Example run of the script:

```
Rscript hapSpectrum_viz.R Chr3R_H12_peaks.txt hapSpectrum.pdf 10 145
```

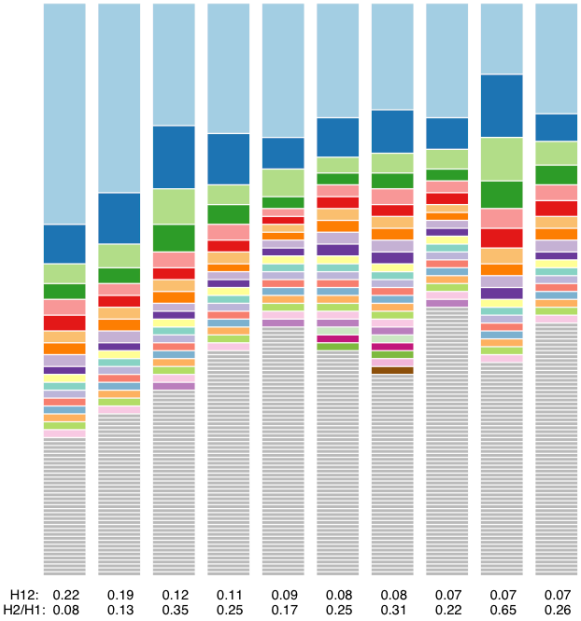

Figure 2: Haplotype frequency spectra for the top 10 peaks on chromosome 3R, corresponding to the analysis windows highlighted in red in Figure 1. This figure was generated with hapSpectrum_viz.R

Visualization of haplotype frequency spectra for any particular analysis window

hapSpectrum_viz.R can be used to visualize the haplotype frequency spectrum of any analysis window for which H12_H2H1.py was run on. For example, above I ran H12_H2H1.py using the –s option for a window centered around coordinate 53034. Here is how to run hapSpectrum_viz.R to visualize the frequency spectrum of that particular analysis window:

```
Rscript hapSpectrum_viz.R Chr3R_H12_output_400_50_coord53034.txt
hapSpectrum_coord53034.pdf 1 145
```

Visualization of haplotype diversity and missing data structure in any particular analysis window with visualizeGenomicData.sh

VisualizeGenomicData.sh generates visualizations of the diversity of the haplotypes in an analysis window and the missing data structure in the analysis window. This visualization can be very informative to gain intuition about the genetic diversity in an analysis window. For example, if the first and second most common haplotypes are very similar to each other, a sweep from standing genetic variation could have occurred. Alternatively, if the first and

second haplotypes are more differentiated than expected, then perhaps some sort of adaptive introgression event might have occurred. In addition to viewing the diversity in an analysis window, the visualization can help with understanding the missing data structure. A lot of missing data in an analysis window can result in artificially high H12 values, so such analysis windows should be excluded from any H12 scan. This visual inspection can help identify any spurious artifacts.

The bash script visualizeGenomicData.sh runs a few preprocessing steps and the R code HapData_viz.R to generate the genomic visualizations. hapSpectrum_viz.R was written in conjunction with Pleuni Pennings at San Francisco State University (pennings@sfsu.edu).

Required input arguments:

- Input genomic data, (the same input data file for H12_H2H1.py above)
- Output of H12_H2H1.py or the output of H12peakFinder.py for the analysis window of interest. The entire output of a scan can be provided.
- Center coordinate of the analysis window. If the coordinate is not in the input genomic data file and if an analysis window centered around this coordinate does not exist in the output of H12_H2H1.py or H12peakFinder.py, then the script will give an error.
- Analysis window size in terms of SNPs. Note that if you specify an even integer, then 1 additional SNP will be added as the center SNP
- Sample size
- Output file name (where the PDF should print to)

Example run of the script:

To visualize the second highest peak centered around coordinate 9060820 (this peak corresponds to the selective sweep at the locus *Ace*):

```
bash visualizeGenomicData.sh Chr3R_DGRP_n145_inputData.txt
Chr3R_H12_peaks.txt 9060820 401 145
genomicVisualization_coord9060820.pdf
```

To visualize the analysis window centered around coordinate 53034 as computed above:

```
bash visualizeGenomicData.sh Chr3R_DGRP_n145_truncatedFile.txt
Chr3R_H12_output_400_50_coord53034.txt 53034 401 145
genomicVisualization_coord53034.pdf
```

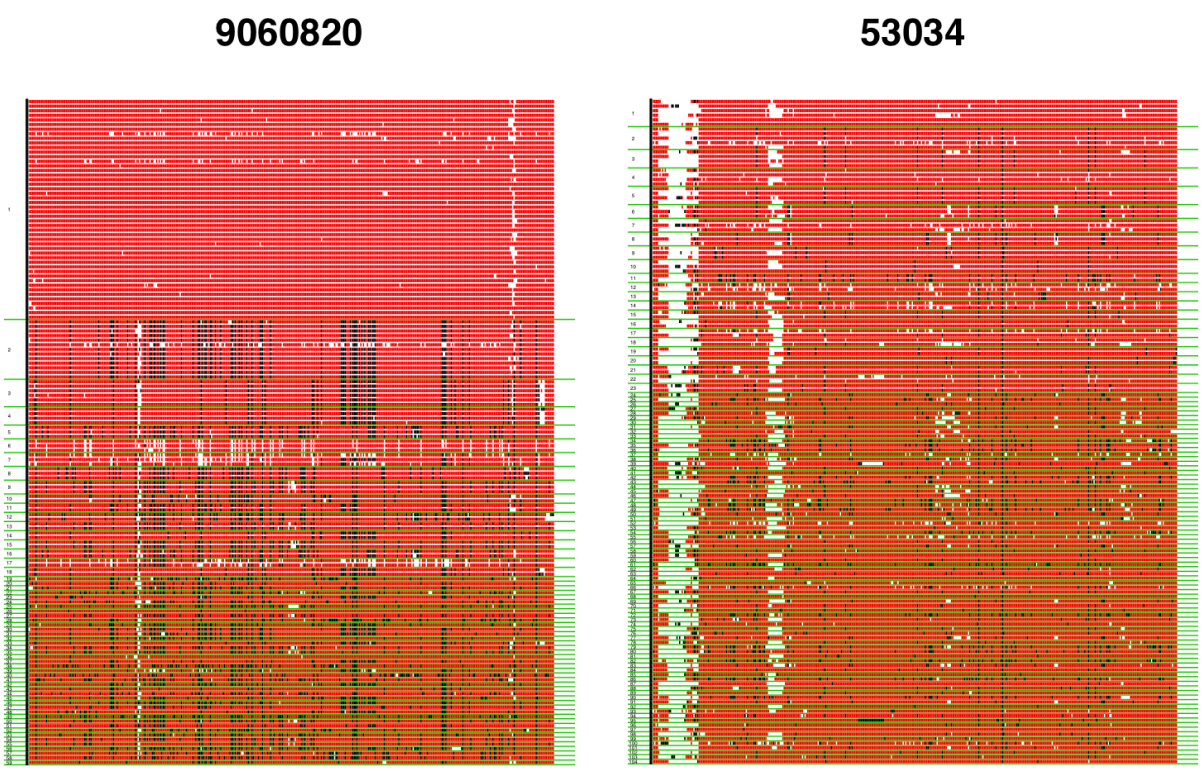

Figure 3: Genomic data visualization for two analysis windows centered around coordinates 9060820 and 53034, respectively. Haplotypes are ordered from most frequent haplotype to least frequent haplotype, and unique haplotypes are separated by a green horizontal line. Numbers of the left side indicate the rank order of the haplotypes in terms of frequency. Red indicates the major allele identified in the most common haplotype. Black indicates any alternate allele from the major allele identified in the first cluster. White indicates missing data. The analysis window centered around coordinate 9060820 coincides with the *Ace* locus, where a known soft selective sweep has occurred. The analysis window centered around coordinate 53034 is a random analysis window on chromosome 3R of the DGRP data set.

Interpretation of the H12 and H2/H1 statistics to determine if a sweep resembles a hard or soft sweep:

The H12 and H2/H1 statistics must be used in conjunction with one another in order to assess if a sweep's H12 and H2/H1 values can be easily generated under hard and soft sweeps. Using approximate Bayesian computation, the H12 and H2/H1 values in data can be compared with H12 and H2/H1 values generated in simulations under a broad range of evolutionary scenarios of sweeps of varying softness. In general, *both* a high H12 and H2/H1 is indicative of a sweep compatible with a soft sweep, but absolute values of H12 and H2/H1 must be compared with distributions generated in simulations in order to make any contextual sense (see Garud *et al.* 2014 for further description on the application of H2/H1 to Drosophila data).