`Haploconfig`:
a program for performing haplotype-based neutrality tests conditional on the number of segregating sites

Kangyu Zhang*, Paul Marjoram†, and Noah A. Rosenberg‡

*Program in Molecular and Computational Biology
University of Southern California, 1042 W 36th Place
Los Angeles CA 90089-1113, USA
kangyuzh@usc.edu

†Department of Preventive Medicine
University of Southern California
Los Angeles CA 90089-9011, USA
pmarjora@usc.edu

‡Center for Computational Medicine and Biology
2017 Palmer Commons, 100 Washtenaw Avenue
Ann Arbor MI 48109-2218, USA
rnoah@umich.edu

The `haploconfig` software is available at
`http://rosenberglab.bioinformatics.med.umich.edu/haploconfig.html`[1]

August 2, 2007

Contents

1 Introduction

Innan *et al.* (2005) describe four statistical tests of neutrality based on the haplotype frequency distribution (the "haplotype configuration") in a sample of DNA sequences, conditional on the number of segregating sites. These include the haplotype configuration test (HCT) — an "exact test" of the haplotype configuration $\mathbf{C}$ — and tests based on three summary statistics (Hudson *et al.*, 1994; Depaulis and Veuille, 1998; Depaulis *et al.*, 2001; Markovtsova *et al.*, 2001; Wall and Hudson, 2001): the Hudson *et al.* haplotype test (HHT) based on the frequency of the most frequent haplotype (M), the haplotype number test (HNT) based on the number of haplotypes (K), and the haplotype diversity (HDT) based on the haplotype diversity (H).

This document describes the usage of `haploconfig`, a software package for implementing the haplotype tests studied by Innan *et al.* (2005). The `haploconfig` package includes two programs: `haploconfig` and `haplofreq.py`. The program `haploconfig` simulates coalescent histories conditional on a given population model — possibly including recombination, exponential population growth, and/or two-population island migration — along with a given number of segregating sites S and a value of the mutation parameter θ. Either fixed values or uniform prior distributions can be accommodated for the recombination, growth, migration, and mutation rate parameters. The `haploconfig` program can also output individual haplotypes as well as the spatial locations of polymorphisms.

Based on the simulations produced by `haploconfig`, the processing program `haplofreq.py` counts frequencies of the various haplotypes in the simulation output, obtaining approximate

HCT, HHT, HNT, and HDT P-values for the various haplotype configurations and possible values of M, K, and H. Alternatively, `haplofreq.py` can be used to indicate which configurations or values of M, K, and H have P-values smaller than a specified cutoff.

1.1 Availability

The `haploconfig` program was used to generate Figures 2 and 3 and Table 7 in INNAN *et al.* (2005). The simulation program, `haploconfig`, was written by Paul Marjoram with modifications by Kangyu Zhang. The post-processing program, `haplofreq.py`, was written by Kangyu Zhang. The software and this document are available at

`http://www.cmb.usc.edu/~noahr/haploconfig.html`.

When using `haploconfig`, please cite INNAN *et al.* (2005).

1.2 Basic overview

The `haploconfig` program is written in C++, and compiled versions are available for Linux (`haploconfig`), Unix (`haploconfig_unix`) and Windows (`haploconfig_windows`). The operating systems under which the code was compiled were Linux Redhat 3.2.2, Unix Sun Solaris 2.8, and Windows XP, with `gcc` version 3.3.2 in Linux/Unix and MC Visual C++ 6.0 in Windows. The source code is available upon request from Kangyu Zhang (kangyuzh@usc.edu).

The script `haplofreq.py` is written in python, an interpreted language. Both programs (`haploconfig` and `haplofreq.py`) are executed from a command prompt using command-line arguments, and the output of both programs is in the form of text files.

1.3 Python

If python (version 2.3 or later) is already installed, the `haplofreq.py` script can be executed immediately. For Linux/Unix, it must be verified that the file is executable, using `chmod 744 haplofreq.py` (similarly `chmod 744 haploconfig` will make `haploconfig` executable). Additionally, the first line of the `haplofreq.py` program must be edited to let the script know the location for python in the system. The command `which python` can be used to find the default path to python; for example, if the location is `/usr/bin/python`, the first line of the script should be `#!/usr/bin/python`.

In Windows, verify that a path for python.exe exists by typing "path" in a command window (obtained by running `cmd` under "Run" in the Start Menu). To add a path for python, right click the "My Computer" icon, click "Properties" menu and select the "Advanced" tab. Then choose "Environment Variables" and modify the system variable "path"; alternatively, type `path /?` in the command window and follow instructions for adding to the path. If python does not recognize the input file for `haplofreq.py`, try placing both `haplofreq.py` and the input file in the same directory as python itself.

If python is not already installed, it can be downloaded from

`http://www.python.org/download`

For Linux/Unix, follow the instructions in the package for compiling (it is necessary to have administrator privileges in advance); for Windows, a `.msi` file is distributed for direct installation. Under Linux/Unix, typing `python` will enter the interpretive mode, in which python commands can be executed interactively. To execute a written script such as `haplofreq.py`, the command is `python script_name` or simply `script_name`. Introductory documentation for python is available at `http://www.python.org/moin/BeginnersGuide`.

Note that if the version of python specified by `which python` is too old (prior to 2.3), the command `python ./haplofreq.py` will produce an error message such as "ImportError: No module named optparse." Similarly, if the version specified on the top line of `haplofreq.py` is old, the command `./haplofreq.py` (omitting `python` in front) will produce the same error. If this problem arises, type `whereis python` to determine if version 2.3 or later is present on the system. If a recent version is present, replace the first line of `haplofreq.py` with the appropriate path, and run the program using `./haplofreq.py` (without the leading `python`). If no recent version is present, the solution is to install a newer version of python.

For example, on one of the Linux systems that we have used, `which python` indicates that the default location for python is `/usr/bin/python`. The version in `/usr/bin/python` is outdated, and when we run `python ./haplofreq.py` we obtain an error message, regardless of the location specified on the first line of `haplofreq.py`. However, `whereis python` finds an updated version in `/usr/local/bin/python2.3`. When the first line of `haplofreq.py` is modified to `#!/usr/local/bin/python2.3`, the command `./haplofreq.py` runs the program properly.

1.4 Included files

The `haploconfig` package includes the following files:

`haploconfig` (Executable `haploconfig` program for Linux)

`haploconfig_unix` (Executable `haploconfig` program for Unix)

`haploconfig_windows.exe` (Executable `haploconfig` program for Windows)

`haploconfig_output` (Example output file for `haploconfig`)

`haplofreq.py` (Executable script for processing output of `haploconfig`)

`haplofreq_table` (Example output file for `haplofreq.py` similar to Tables 5 and 6 of Innan *et al.* [2005])

`haplofreq_Kpvalue` (Example output file for `haplofreq.py` listing configurations that are significant for the haplotype number test below a particular cutoff)

2 Running `haploconfig`

2.1 Command-line arguments

To run `haploconfig`, several parameters must be specified on the command line:

`-a number_of_accepted_configurations`

`-n sample_size` (the program does not support sample sizes larger than 300)

`-s target_number_of_mutations`

`-t mutation_rate_theta` ($\theta = 2N\mu$, where N is haploid population size, and μ is DNA sequence length times mutation rate per base pair per generation)

It is recommended that an output file name is given; otherwise, the program will output the results to the screen. A file `debug_output` may also appear; this file may contain information pertaining to the debugging of the program, and can be ignored.

`-o output_file_name`

Specifying the seed of the random number generator is optional, but recommended (otherwise the program will always use the same seed):

`-d seed`

To incorporate recombination, exponential population growth, or island migration, additional parameters can be added. The island migration model has two populations with symmetric migration, and half of the specified sample size is allocated to each of these populations (half the sample size plus and minus 1/2 in the case of an odd sample size). The default if no recombination, growth, or migration parameters are specified is a simulation using a single constant-size population with no recombination, growth, or migration.

`-r recombination_rate_rho` ($\rho = 2Nr$, where N is haploid population size, and r is DNA sequence length times recombination rate per base pair per generation)

`-g exponential_growth_rate_beta` (β is such that at t time units of N generations in the past, population size was $N \exp[-\beta t]$)

`-m migration_rate_gamma` ($\gamma = 2Nm$, where $N/2$ is the haploid population size for each of the two populations, and m is the fraction of individuals per population who migrate each generation)

The arguments `-t`, `-r`, `-g`, and `-m` all utilize a single value of the appropriate parameter. Alternatively, to simulate from a uniform prior distribution for mutation rate, recombination rate, population growth rate or migration rate, the following arguments can be used:

`-p mutation_rate_lower_bound mutation_rate_upper_bound`

`-q recombination_rate_lower_bound recombination_rate_upper_bound`

`-e growth_rate_lower_bound growth_rate_upper_bound`

`-l migration_rate_lower_bound migration_rate_upper_bound`

There are two final optional arguments that can be specified without numerical parameters. If `-i` is specified without also specifying `-m` or `-l`, the simulation will be of a single panmictic population, and the `-i` option will be ignored:

`-h` (In addition to the standard output of $\mathbf{C}$, M, K, and H, each individual haplotype and all locations of polymorphisms, in the interval $[0, 1]$, will be printed)

`-i` (If specified together with the `-m` or `-l` options, a two-population migration model will be used and the sample size in both of the subpopulations will be binomially distributed with mean equal to half the sample size)

Simply typing `./haploconfig` will display the possible command-line options.

2.2 Example

For example,

```
./haploconfig -a 10000 -o haploconfig_output -n 30 -s 10 -t 1.5 -g 2.0 -q 1 5 -m 5
```

will generate 10000 genealogies, all of which have sample size $n = 30$ and $s = 10$ segregating sites. The mutation parameter is set at $\theta = 1.5$, the growth parameter is set at $\beta = 2.0$, the recombination parameter ρ is chosen from a uniform distribution on $[1, 5]$, and the migration parameter γ is set at $\gamma = 5.0$. Output will be written to the file `haploconfig_output`.

2.3 Output

Each genealogy is printed on one line of the output file. Two sample lines are as follows:

```
GraphNo 29 Theta 1.5 Rho 2.18579 Beta 2 Gamma 5 TimeToGMRCA 1.14694
Config:  4 0 1 1 0 2 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 M 7 K 9 H 0.833333

GraphNo 31 Theta 1.5 Rho 4.00198 Beta 2 Gamma 5 TimeToGMRCA 1.34844
Config:  7 0 0 0 1 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 M 12 K 10 H 0.76444
```

Each quantity is listed after the word describing it. The `GraphNo` refers to the number of ancestral recombination graphs simulated; for example, `GraphNo 31` indicates that this is the 31st ancestral recombination graph that has been simulated. Of these 31, all except two (numbers 29 and 31) have been discarded because they have not had the target number of segregating sites. The `TimeToGMRCA` is the time to the grand MRCA of the ancestral recombination graph. The haplotype configuration is printed as described in Innan *et al.* (2005), where the ith component of the configuration vector represents the number of haplotypes with frequency i.

3 Running `haplofreq.py`

3.1 Command-line arguments

To run `haplofreq.py`, an input file must be specified. An output file is optional; if no output file is specified, the program will output results to the screen.

`-i input_file_name`

`-o output_file_name`

The input file for `haplofreq.py` is an output file from `haploconfig`. Optionally, a value of α can be selected, and only those configurations whose P-values are at most α displayed:

`-a alpha -s statistic`

In this case the "statistic" must be `C` or `c` for the haplotype configuration test, `M` or `m` for the Hudson *et al.* haplotype test, `K` or `k` for the haplotype number test, and `H` or `h` for the haplotype diversity test. The test is assumed to be two-tailed, except with the haplotype configuration test, which is one-tailed.

To display the version of the software, use the following option:

`--version` will display current version

3.2 Example

For example, in Linux/Unix, the command

```
./haplofreq.py -i haploconfig_output -o haplofreq_table
```

will produce in `haplofreq_table` a table similar to Tables 5 and 6 of Innan *et al.* (2005).

```
./haplofreq.py -i haploconfig_output -o haplofreq_Kpvalue -a 0.05 -s K
```

will produce in `haplofreq_Kpvalue` a list of all configurations for which the two-tailed P-value for the haplotype number test is at most 0.05. The corresponding commands in Windows are

```
python haplofreq.py -i haploconfig_output -o haplofreq_table
```

and

```
python haplofreq.py -i haploconfig_output -o haplofreq_Kpvalue -a 0.05 -s K
```

3.3 Output

The top line of the output file from `haplofreq.py` lists the command used in producing the file, and the second line specifies the column headings. When the `-a` and `-s` options are not used, the column headings are the same as those in Table 6 of Innan *et al.* (2005):

```
Configuration(c) P[C==c] P[C<=c](Cumulative probability)
P[M>=M(c)] P[M<=M(c)] P[K<=K(c)] P[K>=K(c)] P[H<=H(c)] P[H>=H(c)]
```

Each additional line gives the appropriate probabilities for a single haplotype configuration, and the lines are sorted by probability of the haplotype configuration:

```
(1,2,1,0,1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0) 0.0001 0.0638
0.0927 0.9389 0.0099 0.9984 0.0658 0.935
```

When the `-a` and `-s` options are used, the order in which information is printed is somewhat different. For example:

```
K P[K<=K(c)] P[K>=K(c)] Configuration(c) P[C<=c] M H
```

Only the configurations with P-values smaller than the specified value of α are shown, and they are sorted by the value of the statistic specified by the `-s` option:

```
4 0.0001 1.0 (0,0,1,1,0,0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)
0.0001 12 0.677778
```

4 Final comments

The accepted genealogies produced by `haploconfig` can be used in a variety of ways, not all of which will involve `haplofreq.py`. For example, in Innan *et al.* (2005), to evaluate the power of the various tests, output files from `haploconfig` under models with one or more among recombination, exponential population growth, and island migration were compared to files generated without any of these phenomena. With the use of alternative specialized scripts, `haploconfig` can aid in such calculations. A C++ program for calculating power is available upon request from Kangyu Zhang.

References

Depaulis, F., S. Mousset and M. Veuille, 2001 Haplotype tests using coalescent simulations conditional on the number of segregating sites. Mol. Biol. Evol. **18:** 1136–1138.

Depaulis, F., and M. Veuille, 1998 Neutrality tests based on the distribution of haplotypes under an infinite-site model. Mol. Biol. Evol. **15:** 1788–1790.

Hudson, R. R., K. Bailey, D. Skarecky, J. Kwiatowski and F. J. Ayala, 1994 Evidence for positive selection in the superoxide dismutase (*Sod*) region of *Drosophila melanogaster*. Genetics **136:** 1329–1340.

Innan, H., K. Zhang, P. Marjoram, S. Tavaré and N. A. Rosenberg, 2005 Statistical tests of the coalescent model based on the haplotype frequency distribution and the number of segregating sites. Genetics **169:** 1763-1777.

Markovtsova, L., P. Marjoram and S. Tavaré, 2001 On a test of Depaulis and Veuille. Mol. Biol. Evol. **18:** 1132–1133.

Wall, J. D., and R. R. Hudson, 2001 Coalescent simulations and statistical tests of neutrality. Mol. Biol. Evol. **18:** 1134–1135.