# `LDhelmet`: Fine-scale recombination rate estimation from population genetic data

Andrew H. Chan
Department of Electrical Engineering and Computer Sciences
University of California, Berkeley

Paul A. Jenkins
Department of Statistics
University of Warwick

Yun S. Song
Department of Statistics
Department of Electrical Engineering and Computer Sciences
University of California, Berkeley

Version 1.7

Copyright © 2012 Andrew H. Chan, Paul A. Jenkins, Yun S. Song.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license can be found at `http://www.gnu.org/licenses/fdl.txt`.

# Contents

# 1 Introduction

LDhelmet is a software program for statistical inference of fine-scale crossover recombination rates from population genetic data. LDhelmet expects the input data to be phased and aligned DNA sequences. The statistical model assumes the sample of DNA sequences is randomly drawn from a single population, and that the population followed a neutral evolution with constant population size. However, the statistical inference can still be accurate even with mild deviations from these assumptions. LDhelmet can handle sample sizes of up to 50 individuals (haplotypes), and is suitable for whole-genome sequence analysis.

LDhelmet is an implementation of the method described in
Chan AH, Jenkins PA, Song YS (2012) Genome-Wide Fine-Scale Recombination Rate Variation in Drosophila melanogaster. PLoS Genet 8(12).

# 2 Installation

LDhelmet has the following dependencies:

1. Boost C++ Libraries, version 1.37 or above.
2. GNU Scientific Library, version 1.15 or above.

If you have access to the package manager of your operating system, using it is the easiest way to install the dependencies. Sections 2.1 and 2.2 provide instructions for manual installation of the dependencies on Linux and Mac.

Follow these steps in this order to install LDhelmet:

1. Install Boost as described in Section 2.1, if necessary.
2. Install GSL as described in Section 2.2, if necessary.
3. Install LDhelmet as described in Section 2.3.

## 2.1 Installing the Boost C++ Libraries

Download the Boost C++ Libraries as a tar archive from www.boost.org. The name of the tar archive will have the form boost_x_xx_x.tar.gz, where x_xx_x is the version number (e.g., 1_51_0). The version of Boost must be at least 1.37.

**Assumptions:** For instructive purposes, we will make the following assumptions:

- The desired installation directory is named my_boost and is located in /home/gradstud.
- The tar archive is called boost_1_51_0.tar.gz. **If the version of your Boost archive is different, be sure to use the appropriate file name.**

**Instructions:** Follow these steps at the shell (or using the Terminal app if on a Mac):

1. Create the directory in which you intend the Boost installation to reside by typing

   ```
   mkdir /home/gradstud/my_boost
   ```

2. Move boost_1_51_0.tar.gz to /home/gradstud/my_boost.
3. Change directory to /home/gradstud/my_boost by typing

   ```
   cd /home/gradstud/my_boost
   ```

4. Untar the archive by typing

```
tar xzvf boost_1_51_0.tar.gz
```

5. A directory called `boost_1_51_0` will be created in `/home/gradstud/my_boost`. Change to this directory by typing

   ```
   cd boost_1_51_0
   ```

6. In `boost_1_51_0` will be a file named `bootstrap.sh`. Run the bootstrap script by typing

   ```
   ./bootstrap --prefix=$PWD/..
   ```

   Bootstrapping will require a few seconds.

7. Build and install the `Boost` libraries by typing

   ```
   ./b2 install
   ```

   Building and installation will require some time.

8. The following directories should have been created in `/home/gradstud/my_boost`:

   (a) `include`: Contains the `Boost` header files.

   (b) `lib`: Contains the `Boost` compiled libraries.

   If any of these directories is missing from `/home/gradstud/my_boost`, then the installation failed. Please contact your system administrator.

9. Installation of the `Boost` libraries is complete. You will need to modify `LDhelmet`'s `Makefile` to reflect the installation directory. If you manually installed both `Boost` and `GSL`, follow the instructions in Section 2.4 instead. Otherwise, change the following four lines in the `Makefile`:

   ```
   INC_FLAG =
   LIB_FLAG =
   #INC_FLAG = -I/home/gradstud/my_boost/include -I/home/gradstud/my_gsl/include
   #LIB_FLAG = -L/home/gradstud/my_boost/lib -L/home/gradstud/my_gsl/lib
   ```

   to

   ```
   #INC_FLAG =
   #LIB_FLAG =
   INC_FLAG = -I/home/gradstud/my_boost/include
   LIB_FLAG = -L/home/gradstud/my_boost/lib
   ```

   where `/home/gradstud/my_boost` is your `Boost` installation directory.

## 2.2 Installing the `GNU Scientific Library`

Download the `GNU Scientific Library` from `www.gnu.org/software/gsl/`. The name of the `tar` archive will have the form `gsl-x.xx.tar.gz`, where `x.xx` is the version number (e.g., `1.15`). The version of `GSL` must be at least 1.15.

**Assumptions:** For instructive purposes, we will make the following assumptions:

- The desired installation directory is named `my_gsl` and is located in `/home/gradstud`.
- The `tar` archive is called `gsl-1.15.tar.gz`. **If the version of your `GSL` archive is different, be sure to use the appropriate file name.**

**Instructions:** Follow these steps at the shell (or using the `Terminal` app if on a Mac):

1. Create the directory in which you intend the `GSL` installation to reside by typing

   ```
   mkdir /home/gradstud/my_gsl
   ```

2. Move `gsl-1.15.tar.gz` to `/home/gradstud/my_gsl`.

3. Change directory to `/home/gradstud/my_gsl` by typing

   ```
   cd /home/gradstud/my_gsl
   ```

4. Untar the archive by typing

   ```
   tar xzvf gsl-1.15.tar.gz
   ```

5. A directory called `gsl-1.15` will be created in `/home/gradstud/my_gsl`. Change to this directory by typing

   ```
   cd gsl-1.15
   ```

6. Configure the installation by typing

   ```
   ./configure --prefix=$PWD/..
   ```

7. Build and install the `GSL` library by typing

   ```
   make && make install
   ```

   Building and installation will require some time.

8. The following directories should have been created in `/home/gradstud/my_gsl`:

   (a) `include`: Contains the `GSL` header files.
   (b) `lib`: Contains the `GSL` compiled libraries.
   (c) `bin`: Contains the `GSL` binaries.
   (d) `share`: Contains the `GSL` shared libraries.

   If any of these directories is missing from `/home/gradstud/my_gsl`, then the installation failed. Please contact your system administrator.

9. Installation of the `GSL` library is complete. You will need to modify `LDhelmet`'s `Makefile` to reflect the installation directory. If you manually installed both `Boost` and `GSL`, follow the instructions in Section 2.4 instead. Otherwise, change the following four lines in the `Makefile`:

   ```
   INC_FLAG =
   LIB_FLAG =
   #INC_FLAG = -I/home/gradstud/my_boost/include -I/home/gradstud/my_gsl/include
   #LIB_FLAG = -L/home/gradstud/my_boost/lib -L/home/gradstud/my_gsl/lib
   ```

   to

   ```
   #INC_FLAG =
   #LIB_FLAG =
   INC_FLAG = -I/home/gradstud/my_gsl/include
   LIB_FLAG = -L/home/gradstud/my_gsl/lib
   ```

   where `/home/gradstud/my_gsl` is your `GSL` installation directory.

## 2.3 Installing LDhelmet

Download the LDhelmet tar archive. The name of the archive will have the form ldhelmet_x.x.tgz, where x.x is the version number (e.g., 1.0).

**Assumptions:** For instructive purposes, we will make the following assumptions:

- The desired installation directory is named ldhelmet and is located in /home/gradstud.
- The tar archive is called ldhelmet-1.0.tar.gz. **If the version of your LDhelmet archive is different, be sure to use the appropriate file name.**
- Boost and GSL are installed on your system.

**Instructions:** Follow these steps at the shell (or using the Terminal app if on a Mac):

1. Create the directory in which you intend the LDhelmet installation to reside by typing

   ```
   mkdir /home/gradstud/ldhelmet
   ```

2. Move ldhelmet_1.0.tgz to /home/gradstud/ldhelmet.
3. Change directory to /home/gradstud/ldhelmet by typing

   ```
   cd /home/gradstud/ldhelmet
   ```

4. Untar the archive by typing

   ```
   tar xzvf ldhelmet_1.0.tgz
   ```

5. A directory called ldhelmet_1.0 will be created in /home/gradstud/ldhelmet. Change to this directory by typing

   ```
   cd ldhelmet_1.0
   ```

6. Modify Makefile according to Section 2.4 as necessary.
7. Build LDhelmet by typing

   ```
   make
   ```

8. An executable binary called ldhelmet should have been created in /home/gradstud/ldhelmet.
9. Installation of LDhelmet is complete. If you manually installed Boost or GSL, follow the instructions in Section 2.4 to modify the Makefile appropriately. Refer to Section 3 for instructions on program usage.

## 2.4 Modifying the Makefile

If you performed a manual installation of either the Boost libraries or the GSL library, you will need to modify LDhelmet's Makefile to reflect the installation directories as described in Sections 2.1 and 2.2.

If you installed both the Boost libraries and the GSL library manually, change the following four lines in the Makefile:

```
INC_FLAG =
LIB_FLAG =
#INC_FLAG = -I/home/gradstud/my_boost/include -I/home/gradstud/my_gsl/include
#LIB_FLAG = -L/home/gradstud/my_boost/lib -L/home/gradstud/my_gsl/lib
```

to

```
#INC_FLAG =
#LIB_FLAG =
INC_FLAG = -I/home/gradstud/my_boost/include -I/home/gradstud/my_gsl/include
LIB_FLAG = -L/home/gradstud/my_boost/lib -L/home/gradstud/my_gsl/lib
```

where `/home/gradstud/my_boost` is your `Boost` installation directory and `/home/gradstud/my_gsl` is your `GSL` installation directory.

# 3 Program usage

## 3.1 Overview

`LDhelmet` consists of the following 5 components:

1. `find_confs`: Scans DNA sequence files (FASTA or FASTQ files) and produces a haplotype configuration file (with extension `.conf`).
2. `table_gen`: Computes the likelihood lookup table (with extension `.lk`) from a given haplotype configuration file.
3. `pade`: Computes the Padé coefficient table (with extension `.pade`) from a given haplotype configuration file.
4. `rjmcmc`: Performs reversible-jump MCMC from input sequence files. A likelihood lookup table must be precomputed prior to running this component. A Padé coefficient table can optionally be used to improve the accuracy of the inference. `rjmcmc` outputs the estimated variable recombination rates (with extension `.post`).
5. `post_to_text`: Generates a human-readable text file from a `.post` file (with extension `.txt`).
6. `max_lk`: Computes a maximum likelihood estimate from input sequence files. A likelihood lookup table and optionally a Padé coefficient table must be computed prior to using this component. `max_lk` assumes a constant recombination rate across the entire input sequence. **Note that the maximum likelihood estimate is output to the screen, not to a file.**

To execute a component, type (assuming your current directory contains `ldhelmet`):

```
./ldhelmet <component_name> [options]
```

For example, to run the `find_confs` component, type

```
./ldhelmet find_confs --num_threads 10 -w 50 -o output.conf input1.fasta input2.fasta
```

Descriptions of some of the command-line options (e.g., `--num_threads`, `-w 50`, etc.) are provided in the following sections.

## 3.2 Quickstart

The following are included in the `LDhelmet` distribution in the directory `example_scripts`:

1. An example simulated data file: `input.fasta`.
2. An example mutation transition matrix file: `mut_mat.txt`.
3. A set of `bash` scripts for running on the example data.
   (a) `find_confs.bash`: Takes `input.fasta` and generates a haplotype configurations list.
   (b) `table_gen.bash`: Takes the output from `find_confs` and generates a likelihood table.
   (c) `pade.bash`: Takes the output from `find_confs` and generates a Padé coefficients table.

(d) `rjmcmc.bash`: Takes `input.fasta`, `mut_mat.txt`, and the outputs from `table_gen` and `pade`, and performs rjMCMC to estimate the recombination rate map used to simulate `input.fasta`.

(e) `post_to_text.bash`: Takes the output from `rjmcmc` and generates a text file with the results of the rjMCMC sampling procedure.

(f) `max_lk.bash`: Takes `input.fasta`, `mut_mat.txt`, and the outputs from `table_gen` and `pade`, and finds the maximum likelihood estimate for the recombination rate assuming a constant recombination rate.

(g) `run_all.bash`: Runs all the aforementioned scripts in sequence.

To test `LDhelmet` on the example dataset, go to the `example_scripts` directory and type

```
bash run_all.bash
```

Note that you must be using the `bash` shell for the scripts to run properly. Because certain steps rely on outputs from previous steps, the scripts will need to be run in sequence for them to work on the first pass. Once the output files are generated (e.g., the likelihood table), the other components (e.g., `rjmcmc` and `max_lk`) can be run without needing to regenerate the prerequisite lookup tables. Running all the scripts will take some time.

### 3.3 Example workflow

An example workflow is described below.

1. The DNA sequence files we wish to analyze are `chr2.fastq` and `chr3.fastq`.

   (Both FASTA and FASTQ files are acceptable input formats.)

2. A haplotype configuration file must be created from the DNA sequence files. `find_confs` is used to accomplish this. To create the haplotype configuration file, we must decide on the window size, which is specified as a number of SNPs. The recommended window size is 50 SNPs. The `-w` option specifies the window size. `-w 50` specifies a window size of 50 SNPs. We also need to decide on an output file name, which in this example will be `output.conf`. To speed up the computation, we will use 10 threads, in which case we will only experience a speedup if our computer has at least 10 processor cores. The `--num_threads 10` command-line option specifies this. We create the haplotype configuration file by typing

   ```
   ./ldhelmet find_confs --num_threads 10 -w 50 -o output.conf chr2.fastq chr3.fastq
   ```

   The output is the file `output.conf`.

3. Now that we have the haplotype configuration file (`output.conf`), we need to compute the likelihood lookup table. The likelihood lookup table is computed by the `table_gen` component. The likelihood lookup table requires several parameters:

   (a) $\theta$, the population-scaled mutation rate, in units of 1/bp, specified with option `-t`.

   (b) The grid of $\rho$ values, the population-scaled recombination rates, in units of 1/bp, specified with option `-r`.

   The grid of $\rho$ values is specified in the following format:

   $$\rho_0 \ \delta_0 \ \rho_1 \ \delta_1 \ \ldots \ \rho_n,$$

   where $\rho_0$ is the first $\rho$ value (usually 0.0), $\delta_0$ is the increment to use from $\rho_0$ to $\rho_1$, $\delta_1$ is the increment to use from $\rho_1$ to $\rho_2$, etc., until $\rho_n$.

   For example, the command-line option

   ```
   -r 0.0 0.5 3.0 1.0 10.0
   ```

specifies the following grid of $\rho$ values:

$$(0.0, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0)$$

The recommended grid of $\rho$ values is

```
-r 0.0 0.1 10.0 1.0 100.0
```

which is a sequence that increments by 0.1 from 0.0 to 10.0, and increments by 1.0 from 10.0 to 100.0, providing a higher resolution of $\rho$ values for smaller values of $\rho$ (where the Fisher information is greater).

Suppose we have separately estimated $\theta$ to be 0.1/bp (e.g., using Watterson's estimator on another dataset).

We will call the output file `output.lk`, and again, we wish to use 10 threads.

To compute the likelihood lookup table, type

```
./ldhelmet table_gen --num_threads 10 -c output.conf -t 0.01 \
-r 0.0 0.1 10.0 1.0 100.0 -o output.lk
```

The above command should be typed on a single line in the shell by removing the "\" from the command. For example,

```
./ldhelmet table_gen --num_threads 10 -c output.conf -t 0.01 -r 0.0 0.1 10.0 1.0 100.0 -o output.lk
```

(The "\" is called the *line-continuation* character, which allows us to split a command onto multiple lines to aid readability in this document.)

The output of the computation is the file `output.lk`.

4. (This step is optional but recommended.) We would also like to use Padé coefficients in the sampling step. The `pade` component computes the Padé coefficients used in an asymptotic sampling formula from a haplotype configuration file. The use of the asymptotic sampling formula improves accuracy of the sampling step. The Padé coefficient table requires the following parameters:

   (a) $\theta$, the population-scaled mutation rate, in units of 1/bp, specified with option `-t`.

   (b) The number of Padé coefficients, specified with option `-x`.

   The more Padé coefficients we compute, the more accurate the Padé approximation will be in the sampling step but the longer the computation time of the Padé coefficients will be. We recommend 11 coefficients. The Padé coefficients will be stored in a file named `output.pade`. As before, suppose we have estimated $\theta = 0.1$/bp, and we again wish to use 10 threads. To compute the Padé coefficients, type

   ```
   ./ldhelmet pade --num_threads 10 -c output.conf -t 0.01 -x 11 -o output.pade
   ```

   The output of the computation is the file `output.pade`.

5. At this stage, we have generated the following files:

   (a) `output.conf`: The haplotype configuration file, produced by the `find_confs` component.

   (b) `output.lk`: The likelihood lookup table, produced by the `table_gen` component.

   (c) `output.pade`: The Pade coefficient table, produced by the `pade` component.

   We can optionally provide the following files in the sampling step:

   (a) Mutation matrix: A $4 \times 4$ mutation matrix, with file format specified in Section 4.2. If a mutation matrix is not provided, the Jukes-Cantor mutation matrix is used by default. The mutation matrix specifies the transition probabilities of a mutation.

(b) Ancestral allele priors: A list of priors specifying the distribution over the ancestral allele for every segregating site. The file format is specified in Section 4.3. If ancestral allele priors are not provided, the stationary distribution of the mutation matrix is used by default.

Suppose we have a custom mutation matrix in a file named `mutMatrix.txt`, and suppose we also have a list of custom ancestral allele priors in a file named `ancPriors.txt`.

For the rjMCMC procedure, we will use a burn-in of 100,000 iterations, and run the Markov chain for 1,000,000 iterations (in addition to the burn-in). Because we used a window size of 50 SNPs when we ran the `find_confs` component, we can use a window size of up to 50 SNPs with the `rjmcmc` component. In this example, we will use a window size of 50 SNPs.

The block penalty "smooths" the rjMCMC estimates of the recombination map. Higher block penalties will cause the inference to tend toward maps with less variation. Lower block penalties will encourage more variation in the map. It is difficult to provide hard and fast rules for determining the optimal block penalty parameter. A practical method to choose the block penalty is through empirical simulations to assess accuracy under different scenarios. For human, with an average $\rho$ of 1/kbp, we recommend a block penalty of 5.0. For *Drosophila melanogaster*, with an average $\rho$ of 10/kbp, we recommend a block penalty of 50.0. The block penalty can be any non-negative real number. In this example, we use a block penalty of 50.0. As before, we will use 10 threads.

Recall we have two sequence files we wish to analyze: `chr2.fastq` and `chr3.fastq`. We will first process `chr2.fastq`. Do so by typing

```
./ldhelmet rjmcmc --num_threads 10 -w 50 -l output.lk -p output.pade \
-w 50 -b 50.0 -s chr2.fastq -m mutMat.txt -a ancPriors.txt \
--burn_in 100000 -n 1000000 -o output_chr2.post
```

This will output a file named `output_chr2.post`.

Likewise, to process `chr3.fastq` with the same parameters, type

```
./ldhelmet rjmcmc --num_threads 10 -w 50 -l output.lk -p output.pade \
-w 50 -b 50.0 -s chr3.fastq -m mutMat.txt -a ancPriors.txt \
--burn_in 100000 -n 1000000 -o output_chr3.post
```

This will output a file named `output_chr3.post`.

Note that the output files `output_chr2.post` and `output_chr3.post` are in binary format. To extract text results from these output files, we must complete the following step.

6. To post-process the data, we need to extract the desired results from the binary data. The output is in binary format because the sampling procedure records a large amount of data, and the binary format requires significantly less disk space than a text-based format. To obtain text-formatted output, we use the `post_to_text` component.

   Suppose we want the mean, 2.5th percentile, 50th percentile (median), and the 97.5th percentile. We type

   ```
   ./ldhelmet post_to_text -m -p 0.025 -p 0.50 -p 0.0975 -o output.txt output.post
   ```

   `-m` specifies we want the mean, `-p 0.025` specifies we want the 2.5th percentile, and the other `-p` options are similar. `-o output.txt` specifies the name of the output file, and `output.post` indicates the binary data file (generated in the previous step) from which to extract the text results.

   The format of `output.txt` is as follows:

   (a) The first line is a comment: "`#parameters`".

   (b) The second line lists the parameters used in the sampling step.

(c) The third line is a header that labels the columns that follow. The first two columns are `left_snp` and `right_snp`, which indicate the left SNP position and right SNP position, respectively, of every interval (every consecutive pair of SNPs defines an interval). The columns from the third column on are labels for the mean and percentiles.

(d) The following lines are the SNP positions, mean, and percentiles for the interval between every consecutive pair of SNPs.

(e) The recombination rates are in units of 1/bp.

### 3.4 Displaying help text

To display the help text of any of the components, simply issue the command without command-line arguments. For example, to display the help text for `table_gen`, type

```
./ldhelmet table_gen
```

The help text provides a summary of the command-line options available for the given component.

## 4 File formats

### 4.1 Sequence data

`LDhelmet` handles FASTA and FASTQ sequence formats, and assumes all the sequences are of the same length. Further assumptions for these file formats are as follows:

1. FASTA: Every DNA sequence consists of a comment line followed by one or more sequence lines. The comment line begins with `>` followed by a comment. The sequence lines consist of strings of A, C, G, T and N (N indicates a missing base). A single sequence may be split across multiple lines. The file should not contain any extraneous empty lines. `LDhelmet` assumes the DNA sequence is specified only in *capital* letters.

   For example,

   ```
   >sequence1
   GGACCCCNGAGA
   CTANATGACATA
   >sequence2
   ANCCCATTCACAT
   GGTTAGNCCTA
   ```

   is proper FASTA format.

   Note that in this example, every sequence is of the same length (once the lines for each sequence are concatenated).

2. FASTQ: Every DNA sequence consists of 4 lines:

   (a) Comment line 1: The line begins with "`@`".

   (b) DNA sequence consisting of A, C, G, T, and N (where N indicates a missing base).

   (c) Comment line 2: The line begins with "`+`".

   (d) Quality scores: A string of ASCII characters of the same length as the DNA sequence indicating quality scores.

   Note neither the DNA sequence (line 2) or the quality scores (line 4) may be split on multiple lines. Each must reside on a *single* line. The file should not contain any extraneous empty lines. `LDhelmet` assumes the DNA sequence is specified only in *capital* letters.

   For example,

```
@sequence1
GGACCCCNGAGACTANATGACATA
+comment for sequence 1
abcdefh!jklmnop!rstuvwxy
@sequence2
ANCCCATTCACATGGTTAGNCCNN
+comment for sequence 2
A!CDEFHIJKLMNOPQRST!VW!!
```

is proper FASTQ format.

## 4.2 Mutation matrix

The mutation matrix specifies the transition probabilities from an allele to another allele. The order of alleles is (A, C, G, T) for both the rows and the columns. The file format is a line for every row in the mutation matrix, where each line contains 4 transition probabilities (floating point numbers).

For example,

```
0.1 0.2 0.3 0.4
0.4 0.3 0.2 0.1
0.2 0.2 0.2 0.4
0.1 0.1 0.1 0.7
```

The first line specifies that allele A has a 0.1 probability to transition to A, a 0.2 probability to transition to C, a 0.3 probability to transition to G, and a 0.4 probability to transition to T. Analogously, the second line indicates allele C has 0.4, 0.3, 0.2, and 0.1 probabilities to transition to A, C, G, and T, respectively.

## 4.3 Ancestral allele priors

The ancestral allele priors specify a distribution for every SNP indicating the probability of each allele to be the ancestral allele for that site. The file format is a line for every SNP. The first item on the line is an integer specifying the SNP position (indexed from 0). This is followed by 4 floating point numbers corresponding to the probabilities for A, C, G, and T, respectively (in that order). The SNP positions must be in **ascending** order.

For example,

```
2118 0.1 0.2 0.3 0.4
7346 0.2 0.2 0.2 0.4
8769 0.1 0.1 0.1 0.7
```

The first line indicates that at position 2118, A has a 0.1 probability to be the ancestral allele, C has a 0.2 probability, G has a 0.3 probability, and T has a 0.4 probabilities.

## 4.4 SNP sequence and SNP position files

An alternative input format to FASTA and FASTQ files is the SNP sequence and SNP position files. This alternative input format is used only for the `rjmcmc` and `max_lk` components.

The SNP sequence file has the same format as the FASTA format but contains bases at segregating sites only; hence a SNP sequence file can be much more compact than a standard FASTA or FASTQ file. The SNP position file specifies the positions of the SNPs (indexed from 0) in the corresponding SNP sequence file. It contains a line for every SNP, and each line has non-negative integer indicating a SNP position.

An example SNP sequence file is

```
>sequence1
GACCGG
>sequence2
```

```
ACANCN
>sequence3
ANNTCC
```

and the corresponding SNP position file is

```
356
748
844
1022
1194
1240
```

The SNP position file in the example above has 6 lines because the example SNP sequence file has sequences of length 6.

The SNP sequence file can in fact contain non-segregating sites; however, regardless of whether the SNP sequence file contains non-segregating sites, the number of positions in the SNP position file must match the lengths of the sequences in the SNP sequence file.

# 5 Additional notes

## 5.1 Temporary files

To reduce memory consumption, the `pade` component generates temporary files in a temporary directory named `tmp_pade_xxxxxx`, where `xxxxxx` is a random string, e.g., `tmp_pade_2R2zk9`. If the `pade` computation finishes normally, then the temporary directory will be removed automatically. If, however, the computation is terminated early (e.g., through user intervention), then you will need to remove the temporary directory manually (since the program was terminated before it could remove the temporary directory itself).

# 6 Contact

Please send comments, questions, bugs in the software, or mistakes in this manual to Andrew H. Chan (`andrewhc@eecs.berkeley.edu`).
The corresponding author for the associated paper is Yun S. Song (`yss@eecs.berkeley.edu`).