Instruction Manual for "ChromoPainter: a copying model for exploring admixture in population data"

Garrett Hellenthal

November 28, 2012

Contents

1 Introduction

ChromoPainter is a program to explore admixture in the Single-Nucleotide-Polymorphism (SNP) data of haplotypes sampled from multiple populations. In particular, ChromoPainter takes as input: (1) the SNP data for a set of admixed *"recipient"* chromosomes, (2) the SNP data for a set of *"donor"* chromosomes thought to represent the sources of admixture in the recipient chromosomes, and (3) a genetic-map representing the recombination distance between each pair of contiguous SNPs. (Note that (3) is only necessary if the data is not specified as "unlinked" using the '-u' switch, which we highly recommend to substantially increase power – see below.) It then uses a Hidden Markov Model (HMM) analagous to that described in [1] in a manner that intuitively forms each recipient chromosome as a mosaic of the donor chromosomes (see [2] for details), capturing which donors are essential to explain the recipient DNA. One attractive feature of ChromoPainter is that it deals with an arbitrarily large set of donor chromosomes, so that it requires very little or no *a priori* information about the important donors involved in the admixture history of the recipient population.

When conditioning a set of recipient chromosomes on a set of donor chromosomes, ChromoPainter provides the following as output for each recipient individual (individuals may be either haploid or diploid):

1. stochastic sample realizations of the HMM denoting which donors the recipient chromosome copies from at each SNP – we refer to this as *"painted chromosomes"*
2. the total expected length of genetic material genome-wide copied from each donor
3. the total expected number of "*chunks"* genome-wide copied from each donor, where a "chunk" refers to a set of contiguous SNP(s) copied intact from the same donor source

The program is flexible in that it allows the user to change emission and transition probabilities for each donor population. It can also use an Expectation-Maximization (E-M) algorithm to re-estimate the proportion of genetic material copied from each donor by using the previous estimates as a new prior under the model and iterating.

2 Getting Started

After extracting the .tar file, compile in the following manner:

```
gcc -o ChromoPainter ChromoPainter.c -lm -lz
```

To compile, note that you must have "zlib" installed (e.g. `sudo apt-get install zlib1g-dev`).

The basic command line is as follows:

./ChromoPainter -g [*haplotype_infile*] -r [*recom_rate_infile*] -f [*donor_list_infile*] -o [*output_filename*]

The user-input file *haplotype_infile* provides the genetic variation information at all SNPs in the donor and recipient haplotypes. The user-input file *recom_rate_infile* provides the genetic distance between each pair of contiguous SNPs in *haplotype_infile*. The user-input file *donor_list_infile* provides information on the donor populations contained in *haplotype_infile*. The user-input name *output_filename* denotes the prefix of the output files. See section "Input Format" below for details on the format of each of the three input files and section "Output" for details on the files output by the program.

Type "./ChromoPainter -h" or "./ChromoPainter –help" to get a brief description of all command line options.

3 Input Format

There are three types of input file, all required except for in specific situations as described below: the *haplotype_infile* (always required; specified with the '-g' switch), the *recom_rate_infile* (specified with the '-r' switch), and the *donor_list_infile* (specified with the '-f' switch).

3.1 *haplotype_infile* ('-g' switch)

The file containing the genetic variation information for the donor and recipient chromosomes (i.e. *haplotype_infile*) should be in a format very similar to PHASE format [3, 4]:

- The first line of the file contains the number of donor haplotypes. (**NOTE: if the '-a' switch is specified, this first line should be 0.**)
- The second line of the file contains the total number of donor *and* recipient **individuals**. If the '-j' switch is used, indicating all individuals are haploid, this will be the total number of donor and recipient haplotypes in the file. If the '-j' switch is **not** specified, indicating individuals are diploid, this number should be half the total number of donor and recipient haplotypes in the file. (In this latter caes, note that unless the '-a' switch is used, only recipient individuals are assumed to be diploid. Thus each donor population can have any number of haplotypes representing it – the program does not ever assume that any given pair of donor haplotypes forms an individual. For this reason, the number entered here can be a fraction; for example if there are 7 total donor haplotypes and 3 diploid recipient individuals, the number entered here should be "6.5"(=[7+3*2]/2).)
- The third line contains the number of SNPs.

- The fourth line contains the letter “P” followed by a vector of the basepair positions of each SNP, in monotonically increasing order. The basepair positions do not strictly need to be in monotonically increasing order if you are including genetic information from multiple chromosomes, as specified in *recom_rate_infile* below. However, within each chromosome, basepairs must be in order.

- The fifth line contains a vector of “S”s, with one “S” per SNP (this line is ignored in the current implementation, but some value must still be specified in line 5).

- The remaining lines of the file contain the genetic variation information of each donor and recipient haplotype, with each row a new haplotype and each column the allelic type at each **biallelic** SNP, in the same order as the “positions” line. The accepted allelic type values are “0”, “1”, “A”, “G”, “C”, or “T”. **There should be NO missing values!!** There should be no spaces between columns for the genotype rows. Donor haplotypes are listed in the initial rows of haplotypes, followed by recipient haplotypes in the final rows of the input file. If individuals are diploid, each pair of 2 contiguous rows should be the two haplotypes for a single individual.

For example, consider a file with 4 donor haplotypes and 6 recipient haplotypes (which may be for 3 recipient individuals), with genetic information collected at 6 SNPs with basepair positions at 100, 200, 300, 400, 500, and 600; the *haplotype_infile* might look like the following:

```
4
5
6
P 100 200 300 400 500 600
SSSSSS
010101
011101
111101
001101
011000
001100
001001
001011
001001
001111
```

The first donor haplotype’s allele types across the 6 SNPs is `010101`. The second donor haplotype’s allele types across the 6 SNPs is `011101`.
The first recipient haplotype’s allele types across the 6 SNPs is `011000`. The second recipient haplotype’s allele types across the 6 SNPs is `001100`. (If diploid, these two haplotypes are the genetic information for recipient individual 1.)

Etc...

If a *donor_list_infile* is specified, donor haplotypes should be ordered as listed in *donor_list_infile* as described below.

3.2 *recom_rate_infile* ('-r' switch)

The file format of *recom_rate_infile* is as follows. There should be a header line followed by one line for each SNP in *haplotype_infile*. Each line should contain two columns, with the first column denoting the basepair position values given in *haplotype_infile*, in the same order. The second column should give the genetic distance per basepair between the SNP at the position in the first column of the same row and the SNP at the position in the first column of the subsequent row. The last row should have a "0" in the second column (though this is not required – this value is simply ignored by the program). Genetic distance should be given in Morgans, or at least the relevant output files assume this value is in Morgans.

If you are including genetic information from multiple chromosomes, put a "-9" (or any value < 0) next to the last basepair position of the preceeding chromosome. For example, to specify one chromosome with SNPs at basepairs 100 and 300 with recombination rate 0.01 Morgan per basepair between them, and a second chromosome with SNPs at basepairs 250 and 450 with recombination rate 0.02 Morgan per basepair between them, the *recom_rate_infile* should look as follows:

```
start.pos recom.rate.perbp
100 0.01
300 -9
250 0.02
450 0
```

In general, specifying "-9" (or any value < 0) in the second column indicates that the recombination rate is infinite between the SNP at the position in the first column of the same row and the SNP at the position in the first column of the subsequent row. In this case, the position in the subsequent row can be less than the position specified in the row containing the "-9". Within a chromosome, however, basepairs must always be given in monotonically increasing order.

Note that the *recom_rate_infile* does not need to be specified if the switch '-u' is specified, indicating that SNPs are unlinked. With the exception of being able to specify "-9" as described above to indicate multiple chromosomes, any recom rate information from a provided *recom_rate_infile* will be ignored if the '-u' switch is used.

Note also that, in order to allow numerical stability in C, **the minimum allowed recombination rate is 1×10^{-15} Morgan per basepair**. Any values below this in the second column of *recom_rate_infile* will be fixed automatically to this value.

3.3 *donor_list_infile* ('-f' switch)

The file *donor_list_infile* provides information on the number of haplotypes from each donor population in *haplotype_infile*. This file is not always required, for instance if you condition a subset of haplotypes (or individuals) on every other haplotype (or individual) using the '-a' switch, but is useful for tidying the output files. There should be one row in *donor_list_infile* for each donor population. There are 2-4 columns per row. The first column gives the donor population label, and the second column gives the number of haplotypes from that donor population. The optional third column (which must be provided only if either the '-p' or '-m' switches are used) gives the *a priori* probability of copying from each donor population (default is equally likely to copy from each donor *chromosome*, so that *a priori* a recipient haplotype is more likely to copy from donor populations with more chromosomes). The optional fourth column (which must be provided only if the '-m' switch is used) gives the mutation (or emission) probability from each donor population; i.e. the probability of a mutation given the recipient is copying from the given donor population at a SNP (default is that mutation probabilities are the same across all donor populations, with rate as described in [2]). If the '-m' switch is specified but the '-p' switch is not, column 3 must still be entered though it is not used. Note that if the '-im' switch is given, denoting that you wish to use E-M to maximize over the mutation probabilities, each donor haplotype will have its own maximum mutation probability value (i.e. while either all haplotypes or haplotypes with the same donor pop label will begin with the same mutation probabilities, they may not have the same probabilities after any number of E-M iterations).

Donor haplotypes in *haplotype_infile* must be ordered according to the rows in *donor_list_infile*. For example, consider the following *donor_list_infile* example consisting of three donor populations with 4, 8, and 6 chromosomes, respectively:

```
pop1 4 0.5 0.0002
pop2 8 0.3 0.0004
pop3 6 0.2 0.0001
```

In this example, *haplotype_infile* must contain 18 rows of donor haplotypes, ordered such that the first 4 rows are the haplotypes from pop1, the next 8 rows are the haplotypes from pop2, and the next 6 rows are the haplotypes from pop3. (Any subsequent lines in *haplotype_infile* will contain the recipient haplotype information.) The output files for .chunkcounts.out, .chunklengths.out, .prop.out, .regionchunkcounts.out, and .regionsquaredchunkcounts.out described below will then contain columns representing the amount of genetic material copied from each of pop1, pop2, and pop3. If the '-p' switch is used, then the *a priori* probability of copying from each pop1 haplotype will be $0.5/4 = 0.125$, the *a priori* probability of copying from each pop2 haplotype will be $0.3/8 = 0.0375$, and the *a priori* probability of copying from each pop3 haplotype will be $0.2/6 = 0.0333$. If the '-m' switch is used, then the probability of mutation at any SNP given you copy from a pop1 haplotype is 0.0002, the probability of mutation given you copy from a pop2 haplotype is 0.0004, and the probability of mutation given you copy from a pop3 haplotype is 0.0001. (Note that you can use the above *donor_list_infile* example without specifying the '-p' and/or '-m' switches; in this case columns 3 and 4 will be ignored.)

Note that the third column of *donor_list_infile*, if specified, must sum to 1. Furthermore, in order to allow numerical stability in c, no value in the third column can be below 1×10^{-15} times the corresponding number of haplotypes in the second column. Values below this threshold will be set automatically to the threshold value.

See "Examples of Usage" below and the provided sample files for examples of each type of input file.

4 Output

There are several output files for ChromoPainter.

4.1 .samples.out

The first line of the output file with suffix .samples.out contains information on the input files and commands used to run ChromoPainter.

The subsequent lines contain stochastically-derived samples indicating which donor haplotypes each recipient haplotypes copies at each SNP under the model. Recipient haplotypes appear in the order specified in *haplotype_infile*. Each recipient haplotype has one row denoting the haplotype index label (=1,...,x for x recipient haplotypes), followed by s rows corresponding to the number of "painted chromosome" samples requested using the '-s' switch (default is s=10). If there are L total SNPs, each "painted chromosome" row has L+1 columns, with the first column corresponding to the sample number (=1,...,s) and the subsequent columns denoting the index of the donor haplotype copied at each SNP, with SNPs ordered as in *haplotype_infile*. For D donor haplotypes, these index labels are 1,...,D corresponding to the donor haplotype's order in *haplotype_infile* going top to bottom (i.e. the first donor haplotype row listed in *haplotype_infile* has label "1", the second donor haplotype row listed in *haplotype_infile* has label "2", etc...). If the '-a' switch is used, labels correspond to the corresponding haplotype rows as ordered in *haplotype_infile*.

4.2 .chunkcounts.out

The output file with suffix .chunkcounts.out contains a matrix with donor populations as columns and recipient individuals as rows (this may be one "population" per donor individual if the '-a' switch is specified, with "individuals" being haploid or diploid). Each matrix entry contains the expected number of "chunks" (i.e. blocks of contiguous SNPs) that the given recipient individual copies from each donor population across all SNPs under the model (after i E-M iterations as specified by the '-i' switch). A "chunk" can intuitively be thought of as analagous to a segment of DNA inherited intact from a single ancestral donor source, with left and right endpoints marked by recombinations, though imperfections of the model and other issues caution strongly against strictly interpretating "chunks" in this manner. If the unlinked switch '-u' is specified (indicating that each SNP is independent from all other SNPs), a "chunk" corresponds to a single SNP.

4.3 .chunklengths.out

The output file with suffix .chunklengths.out contains a matrix with donor populations as columns and recipient individuals as rows (this may be one "population" per donor individual if the '-a' switch is specified, with "individuals" being haploid or diploid). Each matrix entry contains the expected total genetic length of DNA that the given recipient individual copies from each donor population across all SNPs under the model (after i E-M iterations as specified by the '-i' switch). These entries are given in cM assuming *recom_rate_infile* provides basepair values in Morgans; more generally the sum across columns in each row of the .chunklengths.out matrix should be a factor of 100 larger than the total sum of genetic distances across all basepairs provided in *recom_rate_infile* (if recipient individuals are diploid, it will be a factor of 200 greater, as genetic lengths are summed across the recipient individual's two haplotypes). This file is empty if the unlinked switch '-u' is specified.

4.4 .prop.out

The output file with suffix .prop.out contains a matrix with donor populations as columns and recipient individuals as rows (this may be one "population" per donor individual if the '-a' switch is specified, with "individuals" being haploid or diploid). Each matrix entry contains the expected proportion of "chunks" (i.e. blocks of contiguous SNPs, or single SNPs if the '-u' switch is specified) that the given recipient individual copies from each donor population across all SNPs under the model (at the ith E-M iteration as specified by the '-i' switch). If the '-i' switch is not specified, this will simply return the user-specified probabilities in *donor_list_infile* (or default values corresponding to an equal proportion of copying from each donor chromosome if the '-p' switch is not specified). Note that the values in this file are the ones used as *a priori* copying proportions for the results generated in the output files with suffixes

.samples.out, .chunkcounts.out, chunklengths.out, .regionchunkcounts.out, .regionsquaredchunkcounts.out, and .copyprobsperlocus.out. **Furthermore, in order to allow numerical stability in c, no value can be below 1×10^{-15} times the corresponding number of haplotypes from that donor population. Values below this threshold will be set automatically to the threshold value during each iteration of the E-M algorithm.**

4.5 .regionchunkcounts.out

The output file with suffix .regionchunkcounts.out contains values very similar to those described for the output file with suffix .chunkcounts.out. The only difference is that, rather than tabulating the expected number of "chunks" copied from each donor population across all SNPs in *haplotype_infile*, it tabulates values only across the maximum number of "regions" across all SNPs such that each region contains k chunks (with k specified by the '-k' switch; default is k=100). The second column gives the number of regions of size k. For use in finestructure as described in Lawson et al [2].

4.6 .regionsquaredchunkcounts.out

The output file with suffix .regionsquaredchunkcounts.out contains values corresponding to .regionchunkcounts.out, but where expected number of "chunks" copied per donor population have been squared within each k-chunk "region" (with k specified by the '-k' switch; default is k=100) and summed across regions. The second column gives the number of regions of size k. For use in finestructure as described in Lawson et al [2].

4.7 .EMprobs.out

The output file with suffix .EMprobs.out gives for each individual the expected log-likelihood value from the model under values of "donor copying proportions", "mutation (emission) probability" and "recombination rate scaling constant" N_e at each E-M iteration. For each individual, there is one line denoting the individual label, followed by i lines corresponding to the number of E-M iterations specified by the '-i' switch. Each of these i lines gives the E-M iteration number, the expected log-likelihood value of the individual's SNP data under the model (there is one value for each haplotype if individuals are diploid), and the values of N_e and (global) mutation rate used to calculate the expected log-likelihood at the given E-M step. (**Note that N_e here should not be interpreted as "effective population size". Loosely speaking, it might be related to "effective population size" divided by the total number of donor haplotypes. But caution should be exercised with any such interpretation!**)

4.8 .mutationprobs.out

The output file with suffix .mutationprobs.out contains a matrix with donor populations as columns and recipient individuals as rows (this may be one "population" per donor individual if the '-a' switch is specified, with "individuals" being haploid or diploid). Each matrix entry contains the expected number of SNPs that the given recipient individual copies from each donor population *with error* (i.e. emission) across all SNPs under the model (after i E-M iterations as specified by the '-i' switch).

4.9 .copyprobsperlocus.out

The output file with suffix .copyprobsperlocus.out contains for each recipient haplotype the expected probability you copy from each donor population at each SNP. The first column gives the basepair position and each subsequent column gives the probability you copy from each donor population as specified in the first row header. Recipient haplotypes are stacked on top of one another. For example, if there are L SNPs, following the initial header line the next $L+1$ rows will correspond to recipient haplotype 1 (as ordered by row in *haplotype_infile*). The first line in these $L+1$ rows gives the haplotype id, which will correspond to the row in *haplotype_infile* containing the given recipient haplotype (with the first recipient haplotype getting label "1"), and the remaining L rows give the probabilities per SNP. The next $L+1$ rows correspond to recipient haplotype 2. Etc. **Note that basepairs are listed in reverse order relative to the basepairs in *haplotype_infile* and *recom_rate_infile*.**

5 List of Options

The list of ChromoPainter options that can be specified at the command line are as follows:

-g <*haplotype_infile*> (REQUIRED; no default)

-r <*recom_rate_infile*> (REQUIRED; no default – unless using '-u' switch)

-f <*donor_list_infile*> (REQUIRED; no default – unless using '-a' switch)

-i <int> number of E-M iterations for estimating parameters (default=0; with the exception of file with suffix prop.out, the main output files all contain values generated *after* this number of E-M steps) – you can specify any subset of '-in', '-ip', '-im', or '-iM' below to maximize over (with the exception that '-im' and '-iM' cannot both be specified)

-in maximize over recombination scaling constant (N_e) using E-M

-ip maximize over copying proportions using E-M

-im maximize over mutation (emission) probabilities using E-M, such that each donor haplotype has its own mutation probability

-iM maximize over global mutation (emission) probability using E-M, such that each donor haplotype has the same mutation probability

-s <int> number of samples per recipient haplotype (default=10)

-n <double> recombination scaling constant start-value N_e (i.e. as step 0 of the E-M; default=400000/J where J is the total number of haplotypes in <*haplotype_infile*>)

-p specify to use prior copying probabilities supplied in *donor_list_infile* (i.e. as step 0 of the E-M; default is to assume each donor haploytype is copied *a priori* with equal probability)

-m <double> specify to use mutation (emission) probabilities supplied in *donor_list_infile*; user must also provide the self-copying mutation rate following this switch, though this value will be ignored if '-c' switch is NOT used (i.e. as step 0 of the E-M; default is to use the same mutation probability for each donor population equal to fixed estimate in [1])

-M <double> specify to use a global mutation (emission) probability that is the same for all donors (i.e. as step 0 of the E-M; default equal to fixed estimate in [1]) **(Note: entering '-M 0' leads to the default value being used; to make a very small mutation rate, enter a very small number >0.)**

-k <double> specify number of expected chunks to define a 'region' (default=100 – only necessary for use in finestructure applications)

-c condition on own population's individuals (default is to condition only on donor haplotypes)

-j specify that individuals are haploid (default is to assume individals are diploid, including donor haplotypes if '-a' switch is used)

-u specify that SNPs are unlinked

-a $< a_1 >$ $< a_2 >$ condition individuals a_1 through a_2 (as ordered by row in *haplotype_infile*) on every other individual (use '-a 0 0' to do all inds) – if this option is used, the first line in *haplotype_infile* should be "0"

-b print-out zipped file with suffix .copyprobsperlocus.out containing prob each recipient copies each donor at every SNP (note: though zipped, file can be quite large with many SNPs and many donor populations)

-y do NOT print individual count numbers next to population labels in output files. This is only relevant if '-a' switch is used, and probably only necessary if you have one individual per "population" in your *donor_list_infile*. The default is to label each individual in the output files according to their population label from *donor_list_infile* concatenated with a number (e.g. "POPX1", "POPX2", etc., which would become "POPX", "POPX" if the '-y' switch is used).

-o <outfile-prefix>   (default = '*haplotype_infile*')

–help   print help menu (which lists these options)

6 Examples of Usage

Included are samples of a *haplotype_infile* ("example.haps"), *recom_rate_infile* ("example.recomrates"), and *donor_list_infile* ("example.donorlist"). To run the copying model using defaults, which will condition 24 recipient individuals on 264 donor haplotypes from 7 populations (as defined in *donor_list_infile*) and output summaries grouped by population label, type:

```
./ChromoPainter -g example.haps -r example.recomrates -f example.donorlist
```

The output files "example.haps.samples.out", "example.haps.chunkcounts.out", "example.haps.chunklengths.out", "example.haps.prop.out", "example.haps.regionchunkcounts.out", "example.haps.regionsquaredchunkcounts.out", and "example.haps.EMprobs.out" will be generated. If you re-run the same command line, the file "example.haps.samples.out" will change because the sampling of "painted chromosomes" is stochastic, but the values in all other output files should remain the same.

To estimate copying proportions over 5 E-M iterations, instead type:

```
./ChromoPainter -g example.haps -r example.recomrates -f example.donorlist -i 5
-ip
```

The same output files as before will be generated, but now – in addition to the output in "example.haps.samples.out" – the output files "example.haps.chunkcounts.out", "example.haps.chunklengths.out", "example.haps.prop.out", "example.haps.regionchunkcounts.out", and "example.haps.regionsquaredchunkcounts.out" will contain different values than before, as they are all now based on using 5 steps of the E-M algorithm to re-estimate copying proportions under the model. In addition, the output files "example.haps.EMprobs.out" will be different because it contains the expected log-likelihood values for the SNP data of each individual's haplotype under the model at each step of the E-M.

As another example that uses more options, e.g. to allow individuals to copy from their own population, estimate recombination scaling constant *and* copying proportions over 5 E-M iterations, and generate 3 random "painted chromosome" samples for each haplotype under the model after copying proportions have been estimated, type:

```
./ChromoPainter -g example.haps -r example.recomrates -f example.donorlist -o
example.output -i 5 -in -ip -c -s 3
```

Here, because the '-o' switch is specified, the output names change. In particular the output files "example.output.samples.out", "example.output.chunkcounts.out", "example.output.chunklengths.out", "example.output.prop.out", "example.output.regionchunkcounts.out", "example.output.regionsquaredchunkcounts.out", and "example.output.EMprobs.out" will be generated. Note that some of the output files contain an additional column entitled "Self", which gives values for the amount of genetic material each individual copies from all haplotypes of its own population (excluding its own haplotypes).

7 Suggestions/Warnings for Running ChromoPainter

When running ChromoPainter to analyze a given set of recipient haplotypes conditional on a set of donor haplotypes, we suggest the following protocol:

1. Do NOT use the '-u' switch if you can avoid it, as you lose vital linkage disequilibrium information!!!

2. Initially run the model using some number of E-M iterations (e.g. '-i 10') to estimate parameters such as the recombination scaling constant N_e and mutation (emission) probabilities. Check the .EMprobs.out output files to see if all parameter values have converged across E-M iterations in order to determine how many iterations to use. **If the values have not moved at all from the starting values, the algorithm is likely to be stuck in a local maxima, and you should try different starting values (in practice this happens if the starting values are too high or too low, perhaps by orders of magnitude).** If dividing the data into chromosomes to run in parallel, you may want to average parameter estimates across the genome (if it is sensible to assume these parameters are constant, which it often is) and then re-run each region using these fixed parameter values (i.e. using the '-n', '-m', '-p' and/or '-M' switches and '-i 0') to get final estimates.

3. Depending on the study aim, explore a variety of applications, such as allowing self-copying (i.e. the '-c' switch) versus not, and re-estimating copying proportions using E-M (i.e. the '-ip' switch).

8 Computational Complexity

The computational complexity of ChromoPainter is $o((i+1)LDR)$ for i E-M iterations, L SNPs, D donor haplotypes, and R recipient haplotypes. When using the '-a a_1 a_2' switch, the complexity is $\approx o(2iLH(a_2 - a_1 + 1))$ where H is the total number of haplotypes in the file (if using '-a 0 0', the complexity is $\approx o(iLH^2)$). As an example, when using i=10, D=264, R=48, and L=9118, it took ChromoPainter ≈2-3 hours to run on a 2.8GHz Intel Core 2 Duo with 8Gb RAM.

9 Citation

When making use of ChromoPainter, please cite the following paper:

Lawson, D., Hellenthal, G., Myers, S., and Falush, D (2012) "Inference of population structure using dense haplotype data" *PLoS Genet* **8(1)**:e1002453

Questions? Bugs? Contact Garrett Hellenthal at ghellenthal@gmail.com. Though I have tried implementing rigorous checks for input-file format errors, if a "Segmentation Fault" occurs when running ChromoPainter, a likely explanation is that one of the input files is somehow incorrect. Another possible explanation is that the command line input is incorrect. If you encounter such a problem (or any other bug!), please email me so I can ammend the program.

References

[1] N. Li and M. Stephens. Modeling linkage disequilibrium and identifying recombination hotspots using single-nucleotide polymorphism data. *Genetics*, 165(4):2213–33, 2003.

[2] D.J. Lawson, G. Hellenthal, S. Myers, and D. Falush. Inference of population structure using dense haplotype data. *PLoS Genet*, 8(1):e1002453, 2012.

[3] M. Stephens, N.J. Smith, and P. Donnelly. A new statistical method for haplotype reconstruction from population data. *Am J Hum Genet*, 68(4):978–89, 2001.

[4] M. Stephens and P. Donnelly. A comparison of bayesian methods for haplotype reconstruction from population genotype data. *Am J Hum Genet*, 73(5):1162–9, 2003.