Instruction Manual for "ChromoPainterV2: a copying model for exploring admixture in population data"

Garrett Hellenthal

August 11, 2014

Contents

1 Introduction

ChromoPainterv2 updates the previous program ChromoPainter [1] that explores ancestry using Single-Nucleotide-Polymorphism (SNP) data of haplotypes sampled from multiple populations. These updates largely involve making the program easier to use, and making the input and output from ChromoPainterv2 compatible with companion program GLOBETROTTER [2] that identifies, dates and describes past itermixing (i.e. "admixture") events.

In particular, ChromoPainterv2 takes as input: (1) the SNP data for a set of admixed *"recipient"* chromosomes, (2) the SNP data for a set of *"donor"* chromosomes (e.g. thought to represent the sources of admixture or ancestry 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.) It then uses a Hidden Markov Model (HMM) analagous to that described in [3] in a manner that intuitively forms each recipient chromosome as a mosaic of the donor chromosomes (see [1] for details), capturing which donors are essential to explain the recipient DNA. One attractive feature of ChromoPainterv2 is that it deals with an arbitrarily large set of donor chromosomes, so that it requires no *a priori* information about the important donors (i.e. ancestral populations) involved in the ancestral history of the recipient population.

When painting a set of recipient chromosomes conditional on a set of donor chromosomes, ChromoPainterv2 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 – this is of particular use for input in companion program fineSTRUCTURE [1] that clusters individuals into genetically homogeneous groups

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 ChromoPainterv2 ChromoPainterv2.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:

./ChromoPainterv2 -g [*haplotype_infile*] -r [*recom_rate_infile*] -f [*population_list_infile*] [*start_ind*] [*end_ind*] -t [*label_infile*] -o [*output_filename*]

The user-input file *haplotype_infile* provides the genetic variation information at all SNPs in the donor and recipient haplotypes (which can be in any order). The user-input file *recom_rate_infile* provides the genetic distance between each pair of contiguous SNPs in *haplotype_infile*. The user-input file *label_infile* provides population labels for all individuals in *haplotype_infile*, and *population_list_infile* provides information on which populations to use as donors and/or recipients (as well as how many recipient individuals to paint, by specifying *start_ind* and *end_ind*). 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 four input files and section "Output" for details on the files output by the program.

Type "./ChromoPainterv2 -h" or "./ChromoPainterv2 –help" to get a brief description of all command line options.

3 Changes from previous version

The new updates for ChromoPainterv2 involve making the program easier to use, and making the input and output from ChromoPainterv2 compatible with companion program GLOBETROTTER [2] that identifies, dates and describes past itermixing (i.e. "admixture") events. There are the following specific changes:

1. The *haplotype_infile* ('-g' switch) format has changed slightly (see Section 4.1). Compared to the input file for the previous version of ChromoPainter, the first row has been removed, the second row now lists the total number of haplotypes in the file (rather than the total number of individuals) and the fourth row of "SSSSS..." has been removed. In contrast to the original version, donor and recipient haplotypes can be in any order. (Though – as before – every two consecutive rows should contain the DNA from the two haplotypes of a single individual, if dealing with diploids.)

2. There is a new (required) input file *label_infile* ('-t' switch) to specify the individual and population labels for each individual in *haplotype_infile* (see Section 4.3). (NOTE: this is the same file described as **input.file.ids** in the instructions for the GLOBETROTTER program.)

3. The population labels supplied by the user in *label_infile* ('-t' switch) will be used to concatonate individuals into populations automatically, i.e. so that individuals can be listed in any order in *haplotype_infile* ('-g' switch) and *label_infile*, e.g. irrespective of population label. (This is in contrast to the previous version of ChromoPainter, where the haplotypes from the same donor population had to be listed together in consecutive rows in *haplotype_infile*, and listed before the recipient haplotypes.)

4. In *label_infile* ('-t' switch), you can also now specify whether to exclude any individuals from analysis, rather than having to make new input files every time you want to remove individuals for e.g. quality control reasons.

5. The format for *population_list_infile* ('-f' switch) (called *donor_list_infile* in the instructions for the previous version of ChromoPainter) has changed slightly (see Section 4.4). In particular, matching the labels provided in *label_infile* ('-t' switch), you now specify which population you wish to use as donors and/or recipients (and no longer need to specify the number of haplotypes in each). This allows the flexibility of excluding populations, and/or for altering donor and recipient populations, without having to make a new *haplotype_infile* ('-g' switch) as in the previous version.

6. You can now specify how many recipient individuals to paint (i.e. rather than painting all of them), as part of the new '-f' switch (see Section 4.4). (In the previous version, this convenience was only allowed when specifying each individual to copy from every other individual by using the '-a' switch.)

7. When using the '-a' switch specifying that each individual should copy from every other, and when also providing a *label_infile* ('-t' switch), you can now specify that you only want individuals with the populations labels specified in *population_list_infile*, ('-f' switch) to copy from one another. This is useful if you want to perform analyses (for example, that will subsequently use fineSTRUCTURE) on only a subset of populations contained in your *haplotype_infile* ('-g' switch), without having to make a new such file.

8. We have removed the '-c' switch from the previous version that allows you to "self-copy" from members of your own population, as now you simply should specify a recipient population as a donor as well in file *population_list_infile* ('-f' switch) if you want to "self-copy". (As before, an individual will not copy from themself.) For the same reason, when specifying the '-m' switch, you no longer input the mutation rate for the recipient population (i.e. if it is a donor as well) from the command line as in the

previous version, as you now input this value into *population_list_infile* instead.

9. The output file with suffix .prop.out has now been changed to .prior-probs.out. This is to avoid confusion that these are inferred proportions, as they instead – as before – reflect the prior probability of copying from each donor population in the final ChromoPainterv2 run (i.e. last E-M iteration).

10. All output files (see Section 5) now contain labels for individuals based on those provided by the user in *label_infile* ('-t' switch). (As a consequence, we have removed the '-y' switch from the previous version that allows you to alter the labels in output files, which was only applicable when using the '-a' switch in the previous version.)

11. The default value for the "switch parameter" ('-n' switch) is now 400,000 divided by J, with J the total number of donor haplotypes. (This differs from the previous version in that J used to be the total number of haplotypes in *haplotype_infile*. Though as before we recommend inferring the "switch parameter" using E-M iterations.)

4 Input Format

There are four 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), the *label_infile* (specified with the '-t' switch), and the *population_list_infile* (specified with the '-f' switch). See the text below and the files provided with this program for examples of each type of input file.

For most analyses, we recommend having a single *label_infile*, in addition to one *haplotype_infile* and *recom_rate_infile* per chromosome. Ideally you then only ever have to alter your *population_list_infile*, which is relatively easy to toggle, to perform different analyses (e.g. to specify different donor and/or recipient populations when painting) using your data.

4.1 *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 [4, 5]:

- The first line of the file contains the total number of donor *and* recipient haplotypes.
- The second line contains the number of SNPs.

- The third 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 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. If individuals are diploid, each pair of 2 contiguous rows should be the two haplotypes of a single individual.

For example, consider a file with 10 haplotypes (which may be for e.g. 2 donor individuals and 3 recipient individuals, assuming diploidy), 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:

```
10
6
P 100 200 300 400 500 600
010101
011101
111101
001101
011000
001100
001001
001011
001001
001111
```

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

If a *label_infile* is specified (which must be true if not specifying the '-a' switch), individuals should be ordered in consecutive rows as corresponding to *label_infile* (see Section 4.3).

4.2 *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:** In contrast to ChromoPainterv2, for companion program GLOBETROTTER – used to identify and date admixture events – currently there must be only one *recom_rate_infile* per chromosome.)

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.

4.3 *label_infile* ('-t' switch)

The file *label_infile* provides the population and identifier labels for each individual in *haplotype_infile*. This file is not required if you paint a subset of

haplotypes (or individuals) using every other haplotype (or individual) using the '-a' switch. (However, if you do specify it when using the '-a' switch, it will use the individual labels from *label_infile* in all output files, which may be convenient. By providing *label_infile* when using the '-a' switch, you can also use the '-f' switch file to analyse only a subset of populations in your *haplotype_infile*. See Section 4.4.) You can also use this file to identify individuals to exclude from analysis. Note that this is the same file described as **input.file.ids** in the instructions for the GLOBETROTTER program.

The format of *label_infile* is one row per individual, with three columns giving the individual identifier (column 1), the individual's population label (column 2) and an indicator for whether or not to exclude the individual from an analysis (column 3 – use a "0", i.e. "zero", to exclude the given individual; all other values in this column will be ignored). Note that the population labels given in column 2 of *label_infile* must match those given in *population_list_infile* (see Section 4.4). Any individuals with population labels not containined in *population_list_infile* will be ignored in analysis.

An example for *label_infile* with 5 individuals might look like:

Individual1 Pop1 1
Individual2 Pop1 0
Individual3 Pop2 1
Individual4 Pop1 1
Individual5 Pop2 1

The above specifies to remove Individual2 from analysis, so that there are two individuals from Pop1 with individual identifiers Individual1 and Individual4, and two individuals from Pop2 with individual identifiers Individual3 and Individual5 in this analysis. Again note that the haplotype information provided in the corresponding *haplotype_infile* must match exactly the order of this file, so that the initial row(s) of *haplotype_infile* (ignoring the three header rows in that file) must contain the haplotype(s) of Individual1, the next row(s) the haplotype(s) of Individual2 (even though this individual is excluded from analysis), the next row(s) the haplotype(s) of Individual3, etc.

4.4 *population_list_infile* ('-f' switch)

The file *population_list_infile* provides information on the populations to be used as donors and/or recipients when running ChromoPainterv2. This file is not required if you paint a subset of haplotypes (or individuals) using every other haplotype (or individual) using the '-a' switch (and will be ignored in that case, unless you also specify a *label_infile* using the '-t' switch – see below).

With the '-f' switch, you also specify which recipient individuals to paint. For example, specifying '-f [*population_list_infile*] 5 20' will paint recipient individuals 5-20, inclusive, selecting these 16 individuals based on their ordering

in *label_infile* (as always ignoring any individuals marked for exclusion by '0' in *label_infile*, so that the 5th through 20th non-excluded recipient individuals will be painted). Use '-f [*population_list_infile*] 0 0' to paint all recipient individuals. When specifying to copy each individual using all other individuals as donors with the '-a' switch, any '-f' switch values are ignored.

There should be one row in *population_list_infile* for each donor population and one for each recipient population. There are 2-4 columns per row. The first column gives the donor population label, and the second column specifies whether that population is a donor ("D") or recipient ("R"). If you wish to specify a population as both, it should be listed in two different rows, one with a second column containing "D" and the other with the second column containing "R".

The third and fourth columns are optional and can be empty, unless either the '-p' or '-m' switches are specified (these columns will be ignored for any recipient populations). The optional third column gives the *a priori* probability of copying from each donor population. (The 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. (The default is that mutation probabilities are the same across all donor populations, with rate as described in [1]. Use "-9" to specify this default mutation rate for any given population, i.e. row.) If the '-m' switch is specified but the '-p' switch is not, column 3 must still be entered though it is not used.

For example, consider the following *population_list_infile* example consisting of three populations:

```
Pop1 D 0.5 0.0002
Pop2 R 0.3 0.0004
Pop3 D 0.2 0.0001
```

In this example, any individuals with labels Pop1 or Pop3 specified in *label_infile* (given they are not excluded by specifying a "0" in *label_infile*) will be used as donors to paint any individuals with label Pop2 in *label_infile*. (Note that you can specify as many donor and recipient populations as you like.) The output files for .chunkcounts.out, .chunklengths.out, .priorprobs.out, .regionchunkcounts.out, and .regionsquaredchunkcounts.out described below will then contain columns representing the amount of genetic material each individual (rows in those output files) copy from each of Pop1 and Pop3. If you also wish Pop2 to "self-copy", i.e. so that individuals from Pop2 can be painted using any other individuals with label Pop2 (excluding themself), you should add a fourth row (in any or-

der) to the above with "Pop 2 D ...".

If the '-p' switch is specified, then the *a priori* probability of copying from each Pop1 haplotype will be $0.5/n_1$, with n_1 the number of haplotypes included in the analysis with label Pop1, and the *a priori* probability of copying from each Pop3 haplotype will be $0.2/n_3$, with n_3 the number of haplotypes included in the analysis with label Pop3. (Note that if you allow self-copying, individuals from e.g. Pop1 will still copy each other individual from Pop1 with *a priori* probability $0.5/n_1$, so that the total *a priori* probability of copying from Pop1 is $0.5(n_1 - 1)/n_1$ for Pop1 recipient individuals compared to 0.5 for other recipient individuals.) If the '-m' switch is used, then the probability of mutation at any SNP given you copy from a Pop1 haplotype is 0.0002 and the probability of mutation given you copy from a pop3 haplotype is 0.0001. These columns will be ignored for Pop2, as it is specified as a recipient here. (Note again that you can use the above *population_list_infile* example without specifying the '-p' and/or '-m' switches; in this case columns 3 and 4 will be ignored.)

The third column of *population_list_infile*, if specified using '-p', should sum to 1. If it does not (as is the case in the example above), ChromoPainterv2 will rescale these values to sum to 1.0, so that the user-input values are proportional numbers. 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.

Note: When specifying the '-a' switch so that all individuals copy from every other individual, if you specify a *label_infile* ('-t' switch, see Section 4.3) and a *population_list_infile*, ChromoPainterv2 will only include individuals with population labels (as specified in *label_infile*) contained in *population_list_infile*. So for example, if you specify '-t', '-f', and '-a' using the above *population_list_infile* example, each individual with population label Pop1, Pop2 and Pop3 will be painted using all other individuals from these three populations as donors (excluding themself, as always), and individuals from any other populations will be ignored. (Note that whether populations are specified as recipients "R" or as donors "D" in *population_list_infile* is ignored if using the '-a' switch in this manner.) As usual any individuals that are specified to be excluded from analysis (i.e. with a "0" in the third column of *label_infile*) will NOT be included as either a donor or recipient, even if from one of these three populations.

5 Output

There are several output files for ChromoPainterv2.

5.1 .samples.out

The first line of the output file with suffix .samples.out contains information on the input files and commands used to run ChromoPainterv2. 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* and *label_infile*. Each recipient haplotype has one row denoting the haplotype index label (either "1" or "2", depending on ploidy) and individual identifier, 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* and *recom_rate_infile*. For D haplotypes contained in *haplotype_infile*, these index labels are 1,...,D corresponding to the row in *haplotype_infile* (ignoring the initial three header rows) containing the haplotype used to paint that particular SNP for that particular sample. (I.e. the first haplotype row listed in *haplotype_infile* has label "1", the second haplotype row listed in *haplotype_infile* has label "2", etc....)

5.2 .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 that is most closely related ancestrally to the single copied donor relative to all other donors, with left and right endpoints marked by ancestral recombinations. However, imperfections of the model and other issues caution 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.

5.3 .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 filled with zeros if the unlinked switch '-u' is specified.

5.4 .priorprobs.out

The output file with suffix .priorprobs.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) used as a prior for specifying the probability each 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 prior probabilities in *population_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, mutationprobs.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.

5.5 .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 [1].

5.6 .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 [1].

5.7 .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 "switch rate" (or "recombination rate scaling constant" N_e from [1]) 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 switch rate (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!**)

5.8 .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).

5.9 .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 and individual identifier, 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*.**

6 List of Options

The list of ChromoPainterv2 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)

-t <*labels_infile*> (REQUIRED; no default – unless using '-a' switch)

-f <*population_list_infile*> $< f_1 > < f_2 >$ (REQUIRED; no default – unless using '-a' switch), plus specifying to paint recipient individuals f_1 through f_2 using all donor haplotypes (use '-f <*population_list_infile*> 0 0' to paint all recipient inds)

-i <int> number of E-M iterations for estimating parameters (default=0; with the exception of file with suffix priorprobs.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 average switch rate parameter (i.e. N_e in [1]) using E-M

-ip maximize over copying proportions using E-M

-im maximize over mutation (emission) probabilities using E-M, such that each donor population 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> average switch rate parameter start-value (i.e. as step 0 of the E-M; default=400000/J where J is the total number of donor haplotypes included in analysis – note that J might differ among individuals if allowing self-copying via use of '-f' switch)

-p specify to use prior copying probabilities supplied in *population_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 specify to use donor population mutation (emission) probabilities supplied in *population_list_infile* (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 [3])

-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 [3]) **(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)

-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 $< a_1 > < a_2 >$ paint individuals a_1 through a_2 (as ordered by row in *haplotype_infile*) using every other individual (use '-a 0 0' to paint all inds)

-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)

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

–help print help menu (which lists these options)

7 Examples of Usage

Included are samples of a *haplotype_infile* ("BrahuiYorubaSimulationChrom22.haplotypes"), *recom_rate_infile* ("BrahuiYorubaSimulationChrom22.recomrates"), *label_infile* ("BrahuiYorubaSimulation.idfile.txt"), and *population_list_infile* ("BrahuiYorubaSimulation.poplist.txt"). This example is for a simulated population (with label BrahuiYorubaSimulation) that consists of 20 individuals simulated as descendants of an admixture event occuring 30 generations ago, with 80% of the DNA contributed by the Brahui and 20% contributed by the Yoruba (this is the simulation described in Figure 1 of [2]). To run the copying model using default parameters, painting the 40 recipient haplotypes with label BrahuiYorubaSimulation using 2892 donor haplotypes from 93 populations (as defined in *population_list_infile*) and output summaries grouped by population label, type:

```
./ChromoPainterv2 -g example/BrahuiYorubaSimulationChrom22.haplotypes
-r example/BrahuiYorubaSimulationChrom22.recomrates
-t example/BrahuiYorubaSimulation.idfile.txt
-f example/BrahuiYorubaSimulation.poplist.txt 0 0
-o example/BrahuiYorubaSimulationChrom22
```

The output files "example/BrahuiYorubaSimulationChrom22.samples.out", "example/BrahuiYorubaSimulationChrom22.chunkcounts.out",
"example/BrahuiYorubaSimulationChrom22.chunklengths.out",

"example/BrahuiYorubaSimulationChrom22.priorprobs.out", "example/BrahuiYorubaSimulationChrom22.regionchunkcounts.out", "example/BrahuiYorubaSimulationChrom22.regionsquaredchunkcounts.out", "example/BrahuiYorubaSimulationChrom22.mutationprobs.out", "example/BrahuiYorubaSimulationChrom22.copyprobsperlocus.out.gz" (an empty file), and "example/BrahuiYorubaSimulationChrom22.EMprobs.out" will be generated. If you re-run the same command line, the file "example/BrahuiYorubaSimulationChrom22.samples.out" will change because the sampling of painted chromosomes is stochastic, but the values in all other output files should remain the same.

Alternatively, to estimate the switch parameter and global mutation (emission) rate over 10 E-M iterations when painting only 18 recipient haplotypes from recipient individuals 2 through 10, and output 3 painting samples instead of the default 10, type:

```
./ChromoPainterv2 -g example/BrahuiYorubaSimulationChrom22.haplotypes
-r example/BrahuiYorubaSimulationChrom22.recomrates
-t example/BrahuiYorubaSimulation.idfile.txt
-f example/BrahuiYorubaSimulation.poplist.txt 2 10
-o example/BrahuiYorubaSimulationChrom22 -i 10 -in -iM -s 3
```

The same output files as before will be generated, but now – in addition to the output in "example/BrahuiYorubaSimulationChrom22.samples.out" – the output files "example/BrahuiYorubaSimulationChrom22.chunkcounts.out", "example/BrahuiYorubaSimulationChrom22.chunklengths.out", "example/BrahuiYorubaSimulationChrom22.priorprobs.out", "example/BrahuiYorubaSimulationChrom22.mutationprobs.out", "example/BrahuiYorubaSimulationChrom22.regionchunkcounts.out", and "example/BrahuiYorubaSimulationChrom22.regionsquaredchunkcounts.out" will contain different values than before, as they are all now based on using 10 steps of the E-M algorithm to re-estimate copying proportions under the model. In addition, the output file "example/BrahuiYorubaSimulationChrom22.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.

8 Suggestions/Warnings for Running ChromoPainterv2

When running ChromoPainterv2 to paint 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. As we note in Sections 8.1-8.2, we recommend initially running the model using some number of E-M iterations (e.g. '-i 10') to estimate parameters such as the recombination scaling constant (using '-in') and mutation (emission) probabilities (using '-iM' or '-im'). 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 or some subset of chromosomes (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. If you want to estimate local ancestry along recipients' genomes, you likely want to use E-M iterations and '-ip -b' (which will infer proportions of ancestry using E-M and produce the output file .copyprobsperlocus.out.gz giving SNP-by-SNP results, respectively), again checking the .EMprobs.out file to assess convergence. However, we note that we have NOT extensively tested ChromoPainterv2's accuracy in inferring local ancestry. ChromoPainterv2 was originally intended to summarize genome-wide data in populations by tabulating how closely related they are to other populations (e.g. produce the vectors provided in the .chunkcounts.out and .chunklengths.out files). It is these summaries that are exploited in companion programs fineSTRUCTURE and GLOBETROTTER rather than the local ancestry inference on its own. (E.g. the painting samples from the .samples.out file are first "cleaned" using the .chunklengths.out results before using local inference to infer dates of admixture using GLOBETROTTER, as described in [2]. Note in Section 8.2 that this is done withOUT using the '-ip' switch, but instead assuming that a recipient copies each donor haplotype with equal probability *a priori*.)

4. Depending on the study aim, it may be worthwhile to explore a variety of applications, such as allowing self-copying (i.e. specifying a population as a donor *and* recipient in *population_list_infile*) versus not.

8.1 running ChromoPainterv2 with fineSTRUCTURE

When intending to use ChromoPainterv2 followed by fineSTRUCTURE [1] to cluster individuals into genetically homogeneous groups, we recommend the following procedure:

1. run ChromoPainterv2 twice, both times using the '-a' switch:

   (a) For some number of individuals and genetic regions (e.g. we use four chromosomes in practice, and maybe 1/10th of the total sample size),

use 10 E-M iterations to infer the switch rate and global mutation rate using '-i 10 -in -iM' (but otherwise default options). Take the final estimated values of each from the .EMprobs.out output files and average to get a single final estimate for the switch and global mutation rates. (In practice, we first weight-average values of each across chromosomes, weighting by the number of SNPs, and then average across individuals.)

(b) Run ChromoPainterv2 on all individuals and chromosomes using the fixed estimated values from (a) for the switch rate (using '-n') and global mutation rate (using '-M'), without performing any additional E-M iterations (i.e. '-i 0', which is the default).

2. For each individual, sum the .chunkcounts.out, .regionchunkcounts.out, and .regionsquaredchunkcounts.out output files from 1(b) across chromosomes. If necessary, then combine files across individuals, so that there is a single .chunkcounts.out, .regionchunkcounts.out, and .regionsquaredchunkcounts.out containing results for all individuals. (There are programs available at www.paintmychromosomes.com to assist with this summation and combining across files.)

3. Run fineSTRUCTURE using the combined .chunkcounts.out, .regionchunkcounts.out, and .regionsquaredchunkcounts.out files from 2, as described in the fineSTRUCTURE instructions and at www.paintmychromosomes.com.

8.2 running ChromoPainterv2 with GLOBETROTTER

When intending to use ChromoPainterv2 followed by GLOBETROTTER [2] in order to identify, date and describe admixture events in the ancestral history of a given population, we recommend the following procedure:

1. Select a "target" population you wish to detect and describe admixture in, as well as a set of "surrogate" populations you wish to use to describe the DNA of the ancestral source groups involved in the putative admixture event(s).

2. Run ChromoPainterv2 twice to paint "target" and "surrogate" individuals using the same set of "donor" haplotypes (in practice these donors might be all individuals from e.g. the surrogate and/or target populations), both times using the '-f' and '-t' switches:

(a) For some number of "surrogate" and "target" individuals and genetic regions (e.g. we use four chromosomes in practice, and maybe 1/10th of the total sample size), use 10 E-M iterations to infer the switch rate and global mutation rate using '-i 10 -in -iM' (but otherwise default options). Take the final estimated values of each from the .EMprobs.out output files and average to get a single final estimate

for the switch and global mutation rates. (In practice, we first weight-average values of each across chromosomes, weighting by the number of SNPs, and then average across individuals.)

(b) Run ChromoPainterv2 on all "surrogate" and "target" individuals and chromosomes using the fixed estimated values from (a) for the switch rate (using '-n') and global mutation rate (using '-M'), without performing any additional E-M iterations (i.e. '-i 0', which is the default). For the "target" individuals, for each chromosome extract 10 painting samples for each haplotype (i.e. using '-s 10', which is the default), generating one .samples.out file per chromosome that contains the 10 painting samples for each of the target individuals' haplotypes.

3. For each "surrogate" and "target" individual, sum the .chunklengths.out output files from 2(b) across chromosomes. If necessary, then combine files across individuals, so that there is a single .chunklengths.out file containing results for all "surrogate" and "target" individuals, in arbitrary order so not e.g. based on the ordering in *label_infile*. (There are programs available at www.paintmychromosomes.com to assist with this summation and combining across files.)

4. Run GLOBETROTTER using the .samples.out files from 2(b) for all target individuals and the combined .chunklengths.out file from 3, as described in the GLOBETROTTER instructions.

9 Computational Complexity

The computational complexity of ChromoPainterv2 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 ChromoPainterv2 ≈2-3 hours to run on a 2.8GHz Intel Core 2 Duo with 8Gb RAM.

10 Citation

When making use of ChromoPainterv2 (and/or companion program fineSTRUCTURE), 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

If using companion program GLOBETROTTER, please also cite:

Hellenthal, G., Busby, G.B.J., Band, G., Wilson, J.F., Capelli, C., Falush, D. and Myers, S. (2014) "A Genetic Atlas of Human Admixture History" *Science* **343**:747-751

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 ChromoPainterv2, 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] D.J. Lawson, G. Hellenthal, S. Myers, and D. Falush. Inference of population structure using dense haplotype data. *PLoS Genet*, 8(1):e1002453, 2012.

[2] G. Hellenthal, G.B.J. Busby, G. Band, J.F. Wilson, C. Capelli, D. Falush, and S. Myers. A genetic atlas of human admixture history. *Science*, 343:747–751, 2014.

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

[4] 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.

[5] 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.