Simulating admixed and family genotypes: Code Documentation

Jean Morrison

September 23, 2013

This document describes how to use the collection of python scripts which can be downloaded at `http://students.washington.edu/jeanm5/`. These scripts can be used to generate genotypes over a pre-specified pedigree from founder genotypes and to generate simulated admixed genotypes from single ancestry genotypes. All scripts have a short help display which can be seen by using

```
./program_name.py -h
```

The basic process for both types of task is to assign labels over the length of a chromosome corresponding to either founder haplotypes or populations and then assign haplotypes to each label. A chromosome diagram is a simple way of representing a labelled chromosomes and consists of any number of a sequence of pairs. The first item in each pair is a label name (in these scripts labels are all integers). The second item in each pair is a position indicating the end of the segment with the given label for example:

```
1 100 3 500 1 700 2 1000
```

is a chromosome diagram for a chromosome of length 1000 base pairs. From position 1-100 and 500-700 the chromosome has label "1". Between 100 and 500 it is labelled "3" and from 700 to 1000 it is labelled "2". These might represent particular ancestors from whom segments were inherited or ancestral populations.

1 Generating pedigree chromosome diagrams with `recomb_sim.py`

This program takes as input the desired family structure and outputs chromosome diagrams for all the non-founders in the family by simulating recombination events over the given pedigree. There are two required input files - one describing the family structure and one giving a list of chromosome lengths. The output diagrams can be used with founder genotypes from reference data to create full genotype data for all the members of the pedigree. If you plan to do this step, the provided chromosome lengths should match the lengths of the chromosomes in the founder genotype data. Using the last base-pair position for markers on each chromosome is a good way to generate these lengths.

```
> ./recomb_sim.py -h
Usage: recomb_sim.py [options]


Options:
  -h, --help            show this help message and exit
  --fam=FAM             Name input fam file. (required)
  -o OUT, --out=OUT     Name an output file.
  --chr-lengths=CHR_LENGTHS
                        File containing list of chromosome lengths. Default
                        assumes base pair lengths. (required)
  --cM                  Indicates chromosome lengths are in cM not basepairs.
  --bp-to-cM=BP_CM      Recombination rate per base-pair. Default is 1Mb =
                        1cM or --bp-to-cM=1000000.
  --print-cM            Print chromosome diagrams in cM. Not yet implemented.
```

1.1 Input Files

The pedigree file provided with the `--fam` option has the same format as the PLINK fam file. It has 5 (or more) columns:

```
Family ID
Individual ID
Paternal IID
Maternal IID
Sex (1=Male, 2=Female)
```

any columns after 5 will be ignored. Founders should have both paternal and maternal IDs given as 0. Family ID is not used by the program but if these data will later be combined with other data it is useful to give everyone the same family ID and an individual ID that is unique in the whole data set. The fam file should be complete so any parental IDs should also appear somewhere as individual IDs.

The chromosome lengths file is a single column file giving the maximum length of each chromosome. There can be any number of chromosomes desired and default behavior is to assume lengths are in base pairs. The `--cM` flag can be used to indicate that lengths are in cM. The program assumes a constant recombination rate per base-pair which can optionally be specified using `--bp-to-cM` which takes default value 1000000 (1Mb = 1cM).

1.2 Output file

The file output will have $N \cdot 2 \cdot C$ rows where N is the number of individuals in the fam file and C is the number of chromosomes. Each row has 7 columns followed by the chromosome diagram given using founder genome labels. The first five columns match the five columns of the fam file. The 6th column indicates if the chromosome is maternally (M) or paternally (P) inherited. The 7th column gives the chromosome number. A chromosome diagram indicating the haplotype labels along the chromosomes follows the 7th column.

1.3 Calculating IBD for simulated chromosome diagrams

Exact identity by decent proportions can be calculated for pairs of individuals in the pedigree chromosome diagrams using the script `diagram_ibd.py`.

```
> ./diagram_ibd.py -h
Usage: diagram_ibd.py [options] simulated_data.txt



Options:
  -h, --help      show this help message and exit
  --out=O         Output file name.
  --chrom_num=CN  Number of chromosomes
```

This script accepts one argument which is a file output by `recomb_sim.py`. The `--chrom_num` should give the total number of chromosomes in the file and defaults to 22. The output file written by this script has 10 columns:

`ID1` Individual ID for first individual

`ID2` Individual ID for second individual

`IBD_Mb` Total length of sharing in Mb

`IBD1` Proportion of total length along which 1 chromosome is shared IBD

`IBD2` Proportion of total length along which 2 chromosomes are shared IBD

`IBD_proportion` Proportion of the genome shared IBD

`N_segs` Number of segments shared IBD

`Relationship` Relationship between ID1 and ID2

`MRCA1` Distance to most recent common ancestor counting from ID1

`MRCA2` Distance to most recent common ancestor counting from ID2

The column labelled `Relationship` can have values 'PC' for parent-child, 'FSib' for full siblings, 'HSib' for half siblings, 'Yes' for other relationships, 'No' for unrelated and 'SELF' when ID1 and ID2 are the same. The various measures of identity by descent are well defined when there is no inbreeding in the pedigree and there are only three possible Jacquard sharing states. In this case the `IBD_Mb` has a maximum length of twice the total genetic length for a duplicated individual or identical twins and `IBD_proportion` is simply the total sharing divided by the maximum possible. This is equal to $\frac{1}{2}$`IBD1` + `IBD2`. If ID1 and ID2 are the same the maximum value for `IBD_Mb` is just the total genetic length since sharing is between only two haplotypes rather than two pairs of haplotypes.

1.4 Example

A simple example is provided in the files `example.fam` and `example.chr_lengths.txt`. This is a four member pedigree containing two parents (founders) with two male children. There is one chromosome with length 200 Mb. Running

```
./recomb_sim.py --fam=example.fam --chr-lengths=example.chr_lengths.txt \
        --out=example.chr_diag.txt
```

generates the output file

```
fam1 Ann 0 0 2 P 1 1 200000000
fam1 Ann 0 0 2 M 1 2 200000000
fam1 Carlos 0 0 1 P 1 3 200000000
fam1 Carlos 0 0 1 M 1 4 200000000
fam1 Raymond Carlos Ann 1 P 1 3 24782415 4 194935390 3 200000000
fam1 Raymond Carlos Ann 1 M 1 2 170257379 1 200000000
fam1 Harry Carlos Ann 1 P 1 4 691112 3 17009702 4 108551739 3 200000000
fam1 Harry Carlos Ann 1 M 1 1 76658041 2 92995800 1 137028345 2 200000000
```

Every run of this command will produce different results because the simulated recombination process is random. The results above will be used in later examples and can be found in the file `example.chr_diag.txt.save`. In these results, Ann (the female founder) has chromosomes labelled 1 and 2. Carlos (the male founder) has chromosomes labelled 3 and 4. Raymond's paternally inherited chromosome contains two recombination events. It begins labelled 3, switches to the 4 label around 25 Mb and switches back to 3 from 195 Mb to the end. The other recombinant chromosomes can be read similarly.

To calculate IBD between pairs of individuals with these diagrams run

```
./diagram_ibd.py --out=example.chr_diag.IBD --chrom_num=1 example.chr_diag.txt
```

This generates

```
ID1 ID2 IBD_Mb IBD1 IBD2 IBD_proportion N_segs Relationship MRCA1 MRCA2
Ann Ann 0.0000 NA NA 0.0000 0 SELF 0 0
Ann Raymond 200.0000 1.0000 0.0000 0.5000 2 PC 0 1
Ann Harry 200.0000 1.0000 0.0000 0.5000 4 PC 0 1
Raymond Raymond 0.0000 NA NA 0.0000 0 SELF 0 0
Raymond Carlos 200.0000 1.0000 0.0000 0.5000 3 PC 1 0
Raymond Harry 154.7193 0.6102 0.0817 0.3868 5 FSib 1 1
Carlos Carlos 0.0000 NA NA 0.0000 0 SELF 0 0
Carlos Harry 200.0000 1.0000 0.0000 0.5000 4 PC 0 1
Harry Harry 0.0000 NA NA 0.0000 0 SELF 0 0
```

The only line of this file with random variability is the relationship between the two brothers Raymond and Harry. On average, full siblings with unrelated, non-inbred parents will share two segments IBD over half the genome and will share one segment over a quarter of the genome. In this short realization we see that Raymond and Harry share one segment over 61.0% of the genome and two segments over 8.2% of the genome giving them a total sharing proportion of 0.38%.

2 Generating admixed chromosome diagrams with `admixture_sim.py`

This program works similarly and has similar output files to `recomb_sim.py` except that rather than founder genome labels, chromosome diagrams use population labels. This program simulates the segmental ancestries for as many individuals as desired under the scenario that *npop* populations came together in specified proportions $\alpha_1 \ldots \alpha_{npop}$ and have been randomly mating for λ generations.

```
> ./admixture_sim.py -h
Usage: admixture_sim.py [options] alpha1 alpha2 ... alpha_npops


Options:
  -h, --help            show this help message and exit
  -o OUT, --out=OUT     Name an output file.
  --chr-lengths=CHR_LENGTHS
                        File containing list of chromosome lengths.
  --bp-to-cM=BP_CM      Recombination rate per basepair. Default is 1Mb = 1cM.
                        If chromosome lengths are in cM use --bp-to-cM=0 or
                        --bp-to-cM=1.
  -N N, --number=N      Number of admixed individuals to produce.
  --npop=NPOP           Number of populations.
  --lambda=LAM          Number of generations since admixture.
```

The only input file is the chromosome lengths file. The output file has contains $N \cdot 2 \cdot C$ lines where N is the desired number of admixed individuals and C is the number of chromosomes in the chromosome lengths file. This file has the same format as the output of `recomb_sim.py` except that labels in the chromosome diagram represent population labels. After the options are specified $npop - 1$ values summing to less than 1 and representing the initial mixing proportions from each of the populations after the first should be provided.

2.1 Example

Using the command

```
./admixture_sim.py --chr-lengths=example.chr_lengths.txt --npop=2 \
        --lambda=5 --out=example.admx.txt --number=3 0.4
```

will produce the dialogue

```
1: 0.6
2: 0.4
```

indicating that the initial admixture event resulted in a population in which 60% of the individuals were from population 1 and 40% were from population 2. The command generates chromosome diagrams for three individuals after 5 generations of random mating. The resulting chromosome diagram file contains six lines:

```
1 1 0 0 1 M 1 2 57058985 1 67460746 2 127952310 1 169258489 2 171934363 1 187586380 2 200000000
1 1 0 0 1 P 1 2 48810488 1 52858886 2 200000000
2 2 0 0 1 M 1 2 30958085 1 37911007 2 135976457 1 145475180 2 200000000
2 2 0 0 1 P 1 2 4675714 1 11870518 2 38053730 1 183787165 2 188674404 1 199439017 2 200000000
3 3 0 0 1 M 1 1 11128554 2 78248923 1 120414030 2 181879177 1 200000000
3 3 0 0 1 P 1 2 88687757 1 200000000
```

These diagrams are read in the same way as diagrams from `recomb_sim.py` except that labels represent populations rather than founder haplotype labels.

3 Generating genotypes with `sim_to_genotypes.py`

This program will generate BEAGLE formatted genotypes for the chromosomes diagrams simulated by `recomb_sim.py` and `admix_sim.py`. The input data should be file output by one of these programs, BEAGLE formatted reference genotypes with one file per chromosome for founders or members of the founder populations, a map file containing the position and name of each marker and a file containing the population identity of each individual in the reference data for use with `admix_sim.py`. This program processes only one chromosome at a time so if you are generating data for all the autosomes it must be run 22 times.

```
./sim_to_genotypes.py -h
Usage: sim_to_genotypes.py [options] -c N -s simulated_data.txt -m data.map fg1.bgl fg2.bgl ...


Options:
  -h, --help            show this help message and exit
  --mode=MODE           To use with output of recomb_sim.py use 'IBD' mode. To
                        use with output of admixture_sim.py use 'admix' mode.
                        Admix mode requires a file with population ID for each
                        individual in the genotype files.
  -c C, --chr=C         Chromosome number. (required)
  -s S, --simulation=S  Chromosome diagram output from recomb_sim.py or
                        admix_sim.py. (required)
  -m M, --map=M         File giving marker positions in cM or bp if using bp-
                        cm-map option. File must contain exactly the snps in
                        the bgl files. (required)
  --out=OUT             Output file name.
  --chunk=CHUNK         Number of SNPs per chunk to process.
  --gzip                Indicates genotype files are gzipped
  --bp-cm-sim=BPTM      If stop points are in bp give the conversion. Defaulta
                        assumes bp and uses 1Mb=1cM. For files in cM use --bp-
                        cm-sim=0
  --bp-cm-map=BPTM_MAP  If map file is in bp give conversion to cM. Default
                        assumes map file is in bp and uses 1 Mb = 1cM. For map
                        file in cM use --bp-cm-map=0.
  --founder-ids=FIDS    Text file containing list of founder ids to use.
  --pop-ids=PID         A file with population id for all ids in bgl files.
                        Use this option with 'admix' mode.
  --write-names=WRITE   Write out haplotypes used for each FGL
  --read-names=READ     Provide haplotype assignments from a previous
                        chromosome.
  --bind-haplos         Bind founder haplotypes
```

3.1 Input files, output files, and basic usage

BEAGLE file format for SNP data records one row per marker and two columns per individual. The first row of the file is a header with two ignored columns folowed by names for each haplotype in the data. Each subsequent row has two columns identifying the marker followed by genotypes for each individual:

```
I rsid id1 id1 id2M id2P id3 id3 . . .
M rs38076 A T T T A T . . .
M rs12387 G C C G G G . . .
```

The contents of the first column are ignored by `sim_to_genotypes` but the column should be present. The second column should be a marker name present in the map file given with the `-m` option. Genotypes are interpreted as phased haplotypes which belong to the same chromosome. In the example above id1 is assumed to have haploytpes A-G and T-C rather than A-C and T-G which would also be consistent with the given genotypes. The program can accept any number of genotype files containing different individual IDs. These files are listed after the options. If multiple genotype files are given they should all include the same set of markers.

The program can run in two different modes specified using the `--mode` option. In both cases a chromosome number, a map file, and a chromosome diagram file must be given using the `-c`, `-m` and `-s` options or the corresponding `--` syntax. Chromosome number is an integer which is also present in the 7th column of the chromosome diagram file. The map file gives the position of each marker in the BEAGLE files in a two column format. The first column is the marker name and the second column is the position in either base pairs or cM (see additional options).

If the chromosome diagram file was output with `recomb_sim.py` the mode should be set as `--mode=IBD`. For this mode there must be at least as many individuals in the BEAGLE files as there are founders in the pedigree. If the chromosome diagram file was output with `admix_sim.py` the mode should be set as `--mode=admix`. In `admix` mode an additional file specifying the population for each of the ids in the genotype files is also required and is given using the `--pop-ids` option. This file has two columns. The first contains an ID which also appears in one of the headers of the genotype BEAGLE files and the second contains a population number corresponding to the population ids in the chromosome diagram. With this mode there must be at least as many genotypes for each population as there are admixed individuals in the chromosome diagram. For example, if the desired simulated genotypes are admixed between French, Yoruban and Han then it will require three reference genotypes to generate one simulated admixed individual.

Files are output in BEAGLE format.

3.2 Additional options used with both modes

`--out` Specifies the name of output file.

`--chunk` Specifies how many SNPs to read at a time. This can be used to control how fast the program run and how much memory it uses. Generally using a larger chunk will make it faster until memory limits are reached. The default chunk size is 100 but setting it to 1000 may yield better performance depending on the number of individuals.

`--gzip` A binary flag indicating that the genotype files are gzipped. If the option is not used BEAGLE files are assumed to be uncompressed.

`--bp-cm-sim` Specifies the conversion from base pairs to cM for the chromosome diagrams. This should be set to whatever conversion was used to generate the diagrams and defaults to 1Mb=1cM.

`--bp-cm-map` Specifies the conversion from base pairs to cM for the map file. If the map file is in base pairs there is no reason this should be different from the value given to `--bp-cm-sim`. If the map file is in cM this option should be set to 0.

`--write-names` Specifies a file name to write haplotype assignments to. This option is used for de-bugging and error checking but can also be used in combination with the `--read-names` option for `IBD` mode.

In the basic usage the haplotypes given in the BEAGLE files are randomly assigned to founder haplotypes. This option can be used to record this assignment. The format of the output file is two columns - the first giving the founder haplotype label in the chromosome diagrams and the second giving the corresponding haploytpe name from the genotype files. *Haplotype names may appear altered from the original genotype files.* If individuals don't have unique haplotype names in the BEAGLE file they will be assigned unique names. In the example BEAGLE file above individual 1 has two haplotypes that are both named `id1`. The program will internally re-name these `id1` and `id1_0` to distinguish them. The second individual in the small example has uniquely named haplotypes so these will be left unaltered.

In IBD mode the haplotype labels appearing in the output names file correspond directly with labels in the chromosome diagram. In `admix` mode each individual is internally assigned two haplotype labels for each population. The internal labels are printed to an additional file which will be appended with `.admix` (see example for more details).

3.3 Additional options for `IBD` mode

`--founder-ids` Specifies the name of a file containing a list of ids to use as founder genotypes. This option would be used if the BEAGLE genotype files contain more than the necessary number of genotypes and the user would like to specify which ones to use. If this option is not present a random selection of haplotypes from the genotype files will be used as founder haplotypes.

`--read-names` This option can be used to specify a haplotype assignment. The option was added in order to generate data with the same haplotype assignments for every chromosome but must be used carefully due to the internal re-naming process for non-unique haplotype names. The accepted file format is the same as the format of the file output by `--write-names`. To use this option do one of the following:

- Use genotype files that only contain uniquely named haplotypes.
- Generate data for the first chromosome using `--write-names=myhaplos.txt` (or specify another file name). For subsequent chromosomes use `--read-names=myhaplos.txt` and list the genotype files so that ids appear in the same order. Problems will occur if the headers of the genotype files for the first chromosome don't match the headers of the files for the subsequent chromosomes including matching the file order if there are multiple genotype files.

This option can't be used in combination with `--founder-ids`.

`--bind-haplos` This option is a binary flag that forces founders to be assigned haplotypes from the same individual in the BEAGLE files.

3.4 Example: `IBD` mode

This example uses the chromosome diagram output in section 1.3 and made up genotypes found in `example.genos.bgl`

```
I rsid id1 id1 id2 id2 id3 id3 id4 id4 id5 id5 id6 id6
M rs1 A A A T T T A T T A T T
M rs2 G A A G G G A G G A G G
M rs3 G C C C C C C G C G C C
M rs4 C C A C C A A A A C C A
M rs5 C T T T T A T T T A T T
```

The map file `example.genos.map` is in base pairs.

```
rs1 20000000
rs2 80000000
rs3 100000000
rs4 140000000
rs5 190000000
```

The command

```
./sim_to_genotypes.py --mode=IBD -c 1 -s example.chr_diag.txt -m example.genos.map \
        --out=example.IBD.bgl --write-names=example.IBD.names \
        --founder-ids=example.founderids.txt example.genos.bgl
```

will generate the dialogue:

```
Converting map bp positions to cM: 1000000=1cM
Last SNP position: 190.0
Chromosome length: 200.0
I found 4 simulated individuals.
File 1 has 12 haplotypes.
I found 12 founder haplotypes total.
There will be 1 chunks.
6 read from example.founderids.txt
6 founder haplotypes remain.
```

The dialogue indicates that 12 haplotypes were read from the genotype files. The file `example.founderids.txt` specifies that only individuals 1, 2, and 4 should be used resulting in 6 remaining haplotypes. The program needs at least four for this pedigree. Two output data files are generated by the command. The names file shows the random haplotype assignment:

```
1 id2
2 id1
3 id1_0
4 id4_3
5 id4
6 id2_1
```

Only haplotype labels 1-4 appear in the chromosome diagrams but assignments are made for the maximum number of labels which in this case is 6. Some of the haplotypes have been internally re-named as explained in section 3.2. The renamed haplotype is always the second one to appear in the file. From the results in section 1.3 we know that Ann's chromosomes are labelled 1 and 2. These labels have been assigned to the first haplotype from id2 and id1 respectively in the genotype file. Label 3 which is Carlos's paternal chromosome has been assigned to the second haplotype for id1 and label 4 has been assigned to the second haplotype for id 4. The assignments for labels 5 and 6 don't matter in this example.

The other output file is `example.IBD.bgl` which contains genotypes for all members of the pedigree at each marker. The paternal haplotype is given first.

```
I rsid Ann Ann Raymond Raymond Carlos Carlos Harry Harry
M rs1 A A A A T A A T
M rs2 G A G G G A G G
M rs3 G C G G G C C G
M rs4 C A C A A C C C
M rs5 C T T T T T C T
```

Now that the names file is recorded, an identical genotype file can be generated by reading in the haplotype assignment:

```
./sim_to_genotypes.py --mode=IBD -c 1 -s example.chr_diag.txt -m example.genos.map \
        --out=example.IBD2.bgl --read-names=example.IBD.names example.genos.bgl
```

Example: `admix` mode

This example uses the same genotype file as the `IBD` mode example and the output generated in section 2.1 which can be found in `example.admx.txt.save` if it has been overwritten. The file `example.popids` indicates that the first three reference genotypes have ancestry 1 and and second three have ancestry 2. Using the command

```
./sim_to_genotypes.py --mode=admix -c 1 -s example.admx.txt -m example.genos.map --pop-ids=example.p
```

generates the dialogue

```
Converting map bp positions to cM: 1000000=1cM
Last SNP position: 190.0
Chromosome length: 200.0
I found 3 simulated individuals.
File 1 has 12 haplotypes.
I found 12 founder haplotypes total.
There will be 1 chunks.
I found populaion information for 12 ids.
0 haplotypes have no population information and will be discarded.
12 founder haplotypes remain.
```

The `--write-names` option generated two files: `example.admix.names`

```
1 id1
2 id6_5
3 id3
4 id4_3
5 id3_2
6 id4
7 id5
8 id5_4
9 id6
10 id1_0
11 id2
12 id2_1
```

and `example.admix.names.admix`

```
1 1:3,11 2:7,2
3 1:10,5 2:6,8
2 1:12,1 2:9,4
```

Together these files could be used to manually generate the output BEAGLE file and can be read as follows. Individual 1 has two population 1 source haplotypes internally labelled 3 and 11. From `example.admix.names` these correspond to the first haplotype for id3 and the first haplotype for id2 in the genotype file. The population 2 haplotypes for individual 1 are 7 and 2 and these correspond to the first haplotype for id5 and the second haplotype for id6 in the genotype file. Since individual 1 has two chromosomes with population 2 ancestry at the first SNP position the genotype should be T-T. The total output BEAGLE file for this example is

```
I rsid 1 1 3 3 2 2
M rs1 T T A A T T
M rs2 G G A A G G
M rs3 C C C C C G
M rs4 C A A A C C
M rs5 T T T A T C
```