Vignette

Rhodes, G

April 2022

To implement the Recombination Model, we proceed using hierarchical estimation. This means, computationally, that to obtain joint distribution estimates reconconstructed from an order-m Markov chain, we must

1. Convert the SNP data from the four nucelotide bases (A, G, T, C) to binary sequences,
2. Estimate the onewise marginal distribution, the pairwise marginal distribution, and all marginal distributions to and including the $(m+1)$-wise marginal distribution,
3. Use the $(m+1)$-wise marginal distribution and the m-wise marginal distribution to reconstruct the joint distribution.

The **recombinationMCCL** package therefore includes the four following elements,

1. Built-in binary SNP haplotype data from the YRI population TRIOs International HapMap Project data,
2. Functions that perform the marginal distribution estimation,
3. Functions that reconstruct the joint distribution estimation,
4. Functions that can be used to simulate ancestor distributions and descendant samples.

1 Built-in Data

The International HapMap Project has SNP haplotype data publicly available on their website. This data includes: (**1**) the variable `rsID` which is a unique ID given to the SNP site, (**2**) the variable `positionb36` which represents a distance on the chromosome measured in centromeres, and (**3**) the variables which are written in the format `NA12345_A` each of which represents a SNP haplotype, where the variable name indexes the family code and haplotype. For our purposes, the family code and haplotype aren't of interest– in this phased version of the data, only the parents' haplotypes are included so this data includes only unrelated individuals. Looking at the data below, we can see that for each SNP location on each haplotype the nucleotide base has been recorded:

```
rsID position_b36 NA19095_A NA19095_B NA19096_A NA19096_B
    NA18867_A NA18867_B NA18868_A NA18868_B NA18924_A
   NA18924_B NA18923_A NA18923_B NA18488_A NA18488_B
   NA18486_A NA18486_B
1 rs10458597 554484 C C C C C C C C C C C C C C C C
2 rs2185539 556738 C C C C C C C T C C C C C C C C
3 rs11240767 718814 C C C T C C C C T C C C C C C T
4 rs12564807 724325 A A A A A A A A A A A A A A A A
5 rs3131972 742584 A A A A A A A G A A A A A A A G
6 rs3131969 744045 A A G A A A A G A G A G G A G G
7 rs3131967 744197 T T C T T T T C T T T C C T C C
8 rs1048488 750775 C C T C C C C T C T T C T C T T
9 rs12562034 758311 G G G G G G G G G G G G G G G G
10 rs12124819 766409 A A A A A A A A A A A A A A A A
```

You will recall that, for the purposes of this model, we will convert this data into binary sequences where 0 is the major allele and 1 is the minor allele.

Consider the `rs3131969` row of the data. We can see that `A` is the major allele, or the most frequent allele on this site. Therefore we recode each `A` with a 0. This makes `G` the minor allele, so we recode each `G` with a 1. Following such a process for each SNP site gives us data in the following form:

```
NA19095_A NA19095_B NA19096_A NA19096_B NA18867_A
   NA18867_B NA18868_A NA18868_B NA18924_A NA18924_B
   NA18923_A NA18923_B NA18488_A NA18488_B NA18486_A
   NA18486_B
rs10458597 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
rs2185539 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
rs11240767 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 1
rs12564807 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
rs3131972 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1
rs3131969 0 0 1 0 0 0 0 1 0 1 0 1 1 0 1 1
rs3131967 0 0 1 0 0 0 0 1 0 0 0 1 1 0 1 1
rs1048488 0 0 1 0 0 0 0 1 0 1 1 0 1 0 1 1
rs12562034 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
rs12124819 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Displayed here are the first 10 rows of the data. For the data to be properly formatted for use in the remainder of the functions available in package, we then take the transpose so that the rows correspond to one haplotype and the columns correspond to one SNP site.

```
"rs10458597" "rs2185539" "rs11240767" "rs12564807" "
   rs3131972" "rs3131969" "rs3131967" "rs1048488" "
   rs12562034" "rs12124819"
"NA19095_A" 0 0 0 0 0 0 0 0 0 0
"NA19095_B" 0 0 0 0 0 0 0 0 0 0
```

```
"NA19096_A"  0 0 0 0 0 1 1 1 0 0
"NA19096_B"  0 0 1 0 0 0 0 0 0 0
"NA18867_A"  0 0 0 0 0 0 0 0 0 0
"NA18867_B"  0 0 0 0 0 0 0 0 0 0
"NA18868_A"  0 0 0 0 0 0 0 0 0 0
"NA18868_B"  0 1 0 0 1 1 1 1 0 0
"NA18924_A"  0 0 1 0 0 0 0 0 0 0
"NA18924_B"  0 0 0 0 0 1 0 1 0 0
"NA18923_A"  0 0 0 0 0 0 0 1 0 0
"NA18923_B"  0 0 0 0 0 1 1 0 0 0
"NA18488_A"  0 0 0 0 0 1 1 1 0 0
"NA18488_B"  0 0 0 0 0 0 0 0 0 0
"NA18486_A"  0 0 0 0 0 1 1 1 0 0
"NA18486_B"  0 0 1 0 1 1 1 1 0 0
```

The full data frame is available in the package stored as `yri_trio_1`.

2 Marginal Estimation Functions

2.1 Onewise Estimates

The function `estimates_m0` is used to calculate onewise estimates. This function takes three parameters:

1. `n` is the number of descendants in the sample
2. `L` is the length of the SNP sequences in the sample
3. `descendents` is a data frame of descendant samples structured such that each row represents one descendent and each column represents on SNP

site (the number of rows should be equal to `n` and the number of columns should be equal to `L`)

The onewise estimates are calculated following Equation **??**. The output is a matrix with two rows, one for $\hat{\pi}_s(0)$ and one for $\hat{\pi}_s(1)$. It has L columns, one for each of the sites $s = 1, \ldots, L$.

For example, the output generated for the $L = 10$ subset of the `yri_trio_1` is

```
onewise <- estimates_m0(descendents = yri_trio_1[(1:16)
    ,(1:10)], L=10, n=16)
onewise
```

```
"site_1" "site_2" "site_3" "site_4" "site_5" "site_6" "
    site_7" "site_8" "site_9" "site_10"
"pi_0" 1 0.9375 0.8125 1 0.875 0.5625 0.625 0.5625 1 1
"pi_1" 0 0.0625 0.1875 0 0.125 0.4375 0.375 0.4375 0 0
```

This tells us that $\hat{\pi}_3(0) = 0.8125$, or that the estimated marginal probability that the ancestor has a 0 on site 3 is 81.25% for this subset of the data.

2.2 Pairwise Estimates

The function `estimates_m1` is used to calculate the pairwise marginal estimates. Since the pairwise estimates are dependent on the onewise marginal estimates, this function involves an unseen call to `estimates_m0`. This function takes four parameters:

1. `L` is the length of the SNP sequences in the sample
2. `q` is the chosen probability of recombination (value between 0 and 1)
3. `n` is the number of descendants in the sample

4. `d` is a data frame of descendant samples structured such that each row represents one descendent and each column represents one SNP (the number of rows should be equal to `n` and the number of columns should be equal to `L`)

The pairwise estimates are calculated following Equation **??**. The output is a matrix with

1. four rows; $\hat{\pi}_{s,s+1}(0,0)$, $\hat{\pi}_{s,s+1}(0,1)$, $\hat{\pi}_{s,s+1}(1,0)$, and $\hat{\pi}_{s,s+1}(1,1)$
2. $L-1$ columns; each column represents one of the sites $s = 1, \ldots, L-1$

For example, the output generated for the $L = 10$ subset of the `yri_trio_1` with a recombination probability of $q = 0.01$ is

```
pairwise <- estimates_m1(L=10, q=0.01, n=16, d=yri_trio_
    1[(1:16),(1:10)])
pairwise
```

```
site_1 site_2 site_3 site_4 site_5 site_6 site_7
    site_8 site_9
pi_0,0 0.9375 0.75 0.8125 0.875 0.5625 0.5625
    0.5014993687 0.5625 1
pi_0,1 0.0625 0.1875 0 0.125 0.3125 0 0.1235006313 0 0
pi_1,0 0 0.0625 0.1875 0 0 0.0625 0.0610006313 0.4375 0
pi_1,1 0 0 0 0 0.125 0.375 0.3139993687 0 0
```

This tells us that $\hat{\pi}_{2,3}(0,0) = 0.75$, or that the estimated marginal probability that the ancestor has a 0 on site 2 and a 0 on site 3 is 75% for this subset of the data when the recombination probability is 0.01.

2.3 Threewise Estimates

The function `estimates_m2` is used to calculate the threewise marginal estimates. Since the threewise marginal estimates are dependent on the onewise and pairwise marginal estimates, this function includes unseen calls to `estimates_m0` and `estimates_m1`. This function takes four parameters:

1. `L` is the length of the SNP sequences in the sample
2. `q` is the chosen probability of recombination (a value between 0 and 1)
3. `n` is the number of descendants in the sample
4. `descend` is a data frame of descendant samples structured such that each row represents one descendant and each column represents one SNP site (the number of rows should be equal to `n` and the number of columns should be equal to `L`)

The threewise estimates are calculated following Equation **??**. The calculation is done via ten helper functions which calculate the cubic equation coefficients, calculate the roots of the cubic equation, and select which of these roots are real. The output is a matrix with

1. eight rows; $\hat{\pi}_{s,s+1,s+2}(0,0,0)$, $\hat{\pi}_{s,s+1,s+2}(0,0,1)$, $\hat{\pi}_{s,s+1,s+2}(0,1,0)$, $\hat{\pi}_{s,s+1,s+2}(0,1,1)$, $\hat{\pi}_{s,s+1,s+2}(1,0,0)$,$\hat{\pi}_{s,s+1,s+2}(1,0,1)$, $\hat{\pi}_{s,s+1,s+2}(1,1,0)$, and $\hat{\pi}_{s,s+1,s+2}(1,1,1)$
2. $L-2$ columns; each column represents one of the sites $s = 1, \ldots, L-2$

For example, the output generated for the $L = 10$ subset of the `yri_trio_1` with a recombination probability of $q = 0.01$ is

```
threewise <- estimates_m2(L=10, q=0.01, n=16, descend =
    yri_trio_1[(1:16), (1:10)])
threewise
```

```
site_1 site_2 site_3 site_4 site_5 site_6 site_7 site_8
pi_000 0.75 0.75 0.750793127 0.5625 0.5625 0.5014993687
   0.5014993687 0.5625
pi_001 0.1875 0 0.061706873 0.3125 0 0.0610006313 0 0
pi_010 0.0625 0.1875 0 0 0.0625 0 0.1235006313 0
pi_011 0 0 0 0.125 0.25 0 0 0
pi_100 0 0.0625 0.124206873 0 0 0 0.0610006313 0.4375
pi_101 0 0 0.063293127 0 0 0.0625 0 0
pi_110 0 0 0 0 0 0.0610006313 0.3139993687 0
pi_111 0 0 0 0 0.125 0.3139993687 0 0
```

This output tells us that $\hat{\pi}_{2,3,4}(0,0,0) = 0.75$, or that the estimated marginal probability that the ancestor has a 0 on site 2, 0 on site 3, and 0 on site 4 is 75% for this subset of the data when the recombination probability is 0.01.

2.4 Fourwise Estimates

The function `estimates_m3` is used to calculate the fourwise marginal estimates. Since the fourwise marginal estimates are dependent on the onewise, pairwise, and threewise marginal estimates, this function includes unseen calls to `estimates_m0`, `estimates_m1`, and `estimates_m2`. This function takes four parameters:

1. `L` is the length of the SNP sequences in the sample
2. `q` is the chosen probability of recombination (a value between 0 and 1)
3. `n` is the number of descendants in the sample
4. `d` is a data frame of the descendant samples structured such that each row represents one descendant and each column represents one SNP site (the

number of rows should be equal to `n` and the number of columns should be equal to `L`)

The fourwise estimates are calculated following Equation **??**. The calculation is done via 14 helper functions which calculate the cubic equation coefficients, calculate the roots of those cubic equations, and selects which of the roots are real. The output is a matrix with

1. sixteen rows: $\hat{\pi}_{s,s+1,s+2,s+3}(0,0,0,0)$, $\hat{\pi}_{s,s+1,s+2,s+3}(0,0,0,1)$, $\hat{\pi}_{s,s+1,s+2,s+3}(0,0,1,0)$, $\hat{\pi}_{s,s+1,s+2,s+3}(0,0,1,1)$, $\hat{\pi}_{s,s+1,s+2,s+3}(0,1,0,0)$, $\hat{\pi}_{s,s+1,s+2,s+3}(0,1,0,1)$, $\hat{\pi}_{s,s+1,s+2,s+3}(0,1,1,0)$, $\hat{\pi}_{s,s+1,s+2,s+3}(0,1,1,1)$, $\hat{\pi}_{s,s+1,s+2,s+3}(1,0,0,0)$, $\hat{\pi}_{s,s+1,s+2,s+3}(1,0,0,1)$, $\hat{\pi}_{s,s+1,s+2,s+3}(1,0,1,0)$, $\hat{\pi}_{s,s+1,s+2,s+3}(1,0,1,1)$, $\hat{\pi}_{s,s+1,s+2,s+3}(1,1,0,0)$, $\hat{\pi}_{s,s+1,s+2,s+3}(1,1,0,1)$, $\hat{\pi}_{s,s+1,s+2,s+3}(1,1,1,0)$, and $\hat{\pi}_{s,s+1,s+2,s+3}(1,1,1,1)$
2. $L-3$ columns; each column represents one of the sites $s = 1, \ldots, L-3$

For example, the output generated for the $L = 10$ subset of the `yri_trio_1` with a recombination probability of $q = 0.01$ is

```
fourwise <- estimates_m3(L=10, q=0.01, n=16, d =
    yri_trio_t[(1:16), (1:10)])
fourwise
```

```
site_1 site_2 site_3 site_4 site_5 site_6 site_7
pi_0000 0.75 0.75 0.438293127 0.5625 0.5014993687
    0.5014993687 0.5014993687
pi_0001 0 0 0.3125 0 0.0610006313 0 0
pi_0010 0.1875 0 0 0.0625 0 0.0610006313 0
```

```
pi_0011 0 0 0.061706873 0.25 0 0 0
pi_0100 0.0625 0.124206873 0 0 0 0 0.1235006313
pi_0101 0 0.063293127 0 0 0.0625 0 0
pi_0110 0 0 0 0 0.0610006313 0 0
pi_0111 0 0 0 0.125 0.1889993687 0 0
pi_1000 0 0.000793127 0.124206873 0 0 0 0.0610006313
pi_1001 0 0.061706873 0 0 0 0 0
pi_1010 0 0 0 0 0 0.0625 0
pi_1011 0 0 0.063293127 0 0 0 0
pi_1100 0 0 0 0 0 0.0610006313 0.3139993687
pi_1101 0 0 0 0 0 0 0
pi_1110 0 0 0 0 0 0.3139993687 0
pi_1111 0 0 0 0 0.125 0 0
```

This output tells us that $\hat{\pi}_{2,3,4,5}(0,0,0,0) = 0.75$, or that the estimated marginal probability that the ancestor has 0 on site 2, 0 on site 3, 0 on site 4, and 0 on site 5 is 75% for this subset of the data when the recombination probability is 0.01.

3 Joint Estimation Functions

3.1 Order-1 Reconstruction

To estimate the joint distribution from the pairwise marginal distribution, use the function `ancestor_pair_estimation`. According to Equation (**??**), this will be based on the pairwise marginal estimates (from `estimates_m1`) in the numerator and the onewise marginal estimates (from `estimates_m0`) in the denominator. This function takes three parameters,

1. `L` is the length of the SNP sequences in the sample

2. `pairs_est` is a data frame of the pairwise marginal estimates, the output of `estimates_m1`

3. `ones_est` is a data frame of the onewise marginal estimates, the output of `estimates_m0`

The output of this function is a data frame with one column and 2^L rows. Each row represents one of the estimates $\pi_{1,\ldots,L}(x_1, \ldots, x_L)$. These estimates are ordered according to their bitwise value.

For example, a portion of the output generated for the $L = 10$ subset of the `yri_trio_1` with a recombination probability of $q = 0.01$ is

```
m1_joint <- ancestor_pair_estimation(L=10, m=1, pairs_est
    = pairwise, ones_est = onewise)
m1_joint_subset <- as.matrix(m1_joint[(1:10),])
m1_joint_subset
```

```
Probability
1  0.3385120738725
2  0
3  0
4  0
5  0.0833629261275
6  0
7  0
8  0
9  0
10  0
```

From this output, we read that $\hat{\pi}_{1,\ldots,10}(0, 0, 0, 0, 0, 0, 0, 0, 0, 0) \approx 0.339$. Therefore, for this subset of the data when $q = 0.01$, the estimated population fre-

quency with which the ancestor has 0 on all of $s = 1, \ldots, 10$ is about 33.9 % when reconstructed from an $m = 1$ Markov chain.

3.2 Order-2 Reconstruction

To estimate the joint distribution from the threewise marginal distribution, use the function `ancestor_three_estimation`. According to Equation (**??**), this will be based on the threewise marginal estimates (from `estimates_m2`) in the numerator and the onewise marginal estimates (from `estimates_m1`) in the denominator. This function takes three parameters,

1. `L` is the length of the SNP sequences in the sample
2. `three_est` is a data frame of the threewise marginal estimates, the output of `estimates_m2`
3. `pairs_est` is a data frame of the pairwise marginal estimates, the output of `estimates_m1`

The output of this function is a data frame with one column and 2^L rows. Each row represents one of the estimates of $\pi_{1,\ldots,L}(x_1, \ldots, x_L)$. These estimates are ordered according to their bitwise value.

For example, a portion of the output generated for the $L = 10$ subset of the `yri_trio_1` with a recombination probability of $q = 0.01$ is

```
m2_joint <- ancestor_three_estimation(L=10, m=2,
    three_est = threewise, pairs_est = pairwise)
m2_joint_subset <- as.matrix(m2_joint[(1:10),])
m2_joint_subset
```

```
Probability
1 0.397210316534293
```

```
2  0
3  0
4  0
5  0.0483152753118605
6  0
7  0
8  0
9  0
10  0
```

From this output, we read that $\hat{\pi}_{1,\ldots,10}(0,0,0,0,0,0,0,0,0,0) \approx 0.397$. Therefore, for this subset of the data when $q = 0.01$, the estimated population frequency with which the ancestor has 0 on all of $s = 1,\ldots,10$ is about 39.7 % when reconstructed from an $m = 2$ Markov chain.

3.3 Order-3 Reconstruction

To estimate the joint distribution from the fourwise marginal distribution, use the function `ancestor_four_estimation`. According to Equation (**??**), this will be based on the fourwise marginal estimates (from `estimates_m3`) in the numerator and the threewise marginal estimates (from `estimates_m2`) in the denominator. This function takes three parameters,

1. `L` is the length of the SNP sequences in the sample
2. `four_est` is a data frame of the fourwise marginal estimates, the output of `estimates_m3`
3. `three_est` is a data frame of the threewise marginal estimates, the output of `estimates_m2`

The output of this function is a data frame with one column and 2^L rows. Each row represents one of the estimates of $\pi_{1,\ldots,L}(x_1, \ldots, x_L)$. These estimates are ordered according to their bitwise value.

For example, a portion of the output generated for the $L = 10$ portion of `yri_trio_1` with a recombination probability of $q = 0.01$ is

```
m3_joint <- ancestor_four_estimation(L=10, m=3, four_est
   = fourwise, three_est = threewise)
m3_joint_subset <- as.matrix(m3_joint[(1:10),])
m3_joint_subset
```

```
Probability
1  0.390349384921934
2  0
3  0
4  0
5  0.0474807355581117
6  0
7  0
8  0
9  0
10  0
```

From this output, we read that $\hat{\pi}_{1,\ldots,10}(0, 0, 0, 0, 0, 0, 0, 0, 0, 0) \approx 0.390$. Therefore, for this subset of the data when $q = 0.01$, the estimated population frequency with which the ancestor has 0 on all of $s = 1, \ldots, 10$ is about 39 % when reconstructed from an $m = 3$ Markov chain.

4 Functions to Simulate Data

4.1 Simulate the Ancestor Distribution from Data

The function `ancestor` can be used to simulate the distribution of possible ancestor sequences. The simulated probabilities are based on a set of SNP haplotype data. This function takes two parameters,

1. `L` is the length of the ancestor sequences to be simulated
2. `hapmap_data` is a data frame structured such that each row corresponds to one SNP site and each column corresponds to one haplotype.

The output of this function is a data frame with one column and 2^L rows. Each row represents one of the possible ancestor sequences $\pi_{1,\ldots,L}(x_1,\ldots,x_L)$ and the value recorded for each sequence is the simulated probability that the ancestor has that particular sequence. The sequences are ordered by their bit-wise value.

These probabilities are simulated such that the sample proportion is taken to be the probability. A subset of the data frame of SNP haplotype data passed to the function is taken so that the number of rows is equal to L. Then for each possible ancestor sequence,

$$\pi_{1,\ldots,L}(x_1,\ldots,x_L) = \frac{n_{1,\ldots,L}(x_1,\ldots,x_L)}{n}$$

where $n_{1,\ldots,L}(x_1,\ldots,x_L)$ is the number of haplotypes in the sample that has the particular sequence $(x_1,\ldots,x_L)$ and n is the total number of haplotypes in the sequence (the number of columns in `hapmap_data`).

4.2 Simulate Descendant Sequences from Data

The function `descendant_sequences` can be used to simulate a sample of descendant sequences from a set of SNP haplotype data. This function takes five parameters,

1. `L` is the length of the descendant sequences to be simulated
2. `q` is the recombination probability to be used in the simulation
3. `n` is the number of descendant sequences to be simulated
4. `seed` is any numeric value to be used in the function for reproducibility of random results
5. `hapmap` is a data frame of SNP haplotype data structured such that each row corresponds to one SNP site and each column corresponds to one haplotype

The output of this function is a data frame with L columns, each representing a SNP site, and n rows, each representing a descendant.

These descendant sequences are simulated by first simulating an ancestral distribution returned from `ancestor`. To simulate a sequence, one of the possible 2^L sequences is randomly selected using the `sample` function such that the probability that each sequence is selected is the simulated probability returned by `ancestor` for that sequence. The selected sequence is used as the descendant sequence's "starting point". From this starting point, we simulate recombination events based on the q value passed to the function. There will be $L-1$ possible locations for a recombination event (i.e. between sites 1 and 2, between sites 2 and 3, etc.). For each of these locations, we use the `rbinom` function to return either a `0` or `1` where the probability of a `1` is the recombination probability q. If a `1` is returned for location k (where $k = 1, \ldots, L-1,$) then for SNP

sites $k+1, \ldots, L$ a different possible ancestor sequences is randomly selected and the new $x_{k+1}, \ldots, x_L$ replace those SNP sites. This is repeated for each of the n sequences to be simulated.