Chapter 6
Evolution Within Populations

6.1 Descent from One or Two Parents

Every one of us has two parents, four grandparents, eight great-grandparents, and so on. Figure 6.1 shows a random ancestry for a single individual in a population of seven individuals. The individuals are diploid, hence the two dots indicating two genes, and there are two sexes, ellipses, and boxes. As we shall see that the rapid growth of the number of ancestors has the curious effect that it is easy to have a famous forbear—but difficult to not share him with everybody [13, 41]. In contrast, each gene has a single ancestor, if we ignore recombination for now. Using simulations, we learn that in the uni-parental genealogies of genes it takes much longer to find common ancestors than in the bi-parental genealogies of individuals.

New Concepts

Name	Comment
bi-parental genealogy	the genealogy of people
uni-parental genealogy	the genealogy of genes
Wright–Fisher model	model of evolution within populations

New Programs

Name	Source	Help
`drawGenealogy`	book website	`drawGenealogy -h`
`drawWrightFisher`	book website	`drawWrightFisher -h`

6.1.1 Bi-Parental Genealogy

Problem 409 Write down the number of ancestors as a function of the number of generations back in time. How many ancestors do you have 30 generations back?

Problem 410 Trace the ancestry of individual i_4 in Fig. 6.1. Walk back in time from b_0 to b_3 and count the ancestors of i_4 in each generation. What do you observe?

B. Haubold and A. Börsch-Haubold, *Bioinformatics for Evolutionary Biologists*, https://doi.org/10.1007/978-3-319-67395-0_6

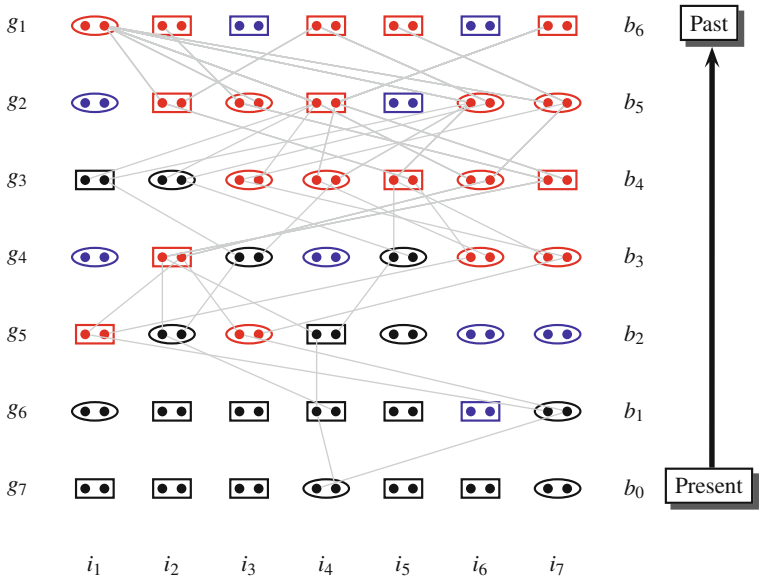

Fig. 6.1 Simulation of the ancestors of a single individual. Ellipses and boxes indicate two sexes, the dots two genes. g_i stands for generation i, b_j for j generations back

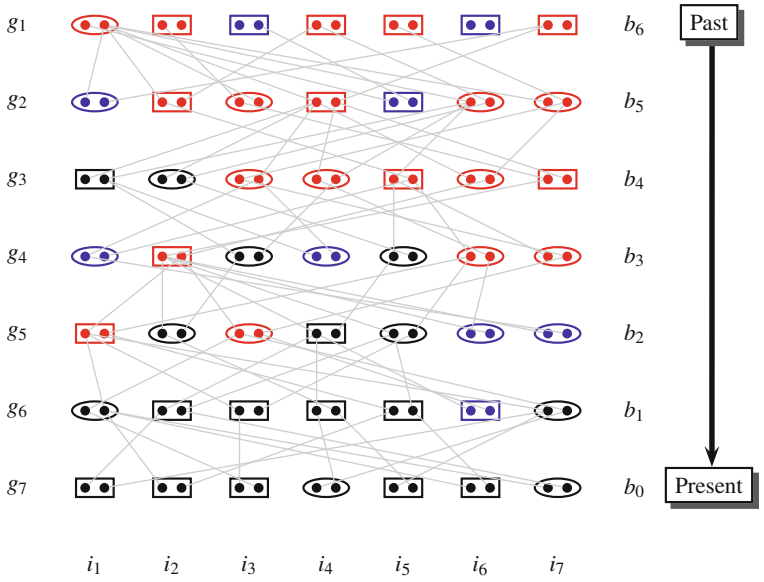

Fig. 6.2 Same as Fig. 6.1 but with all lines of descent included for moving forward in time

Problem 411 Figure 6.2 shows the same simulation as Fig. 6.1, but this time all lines of descent are included, not only those leading to i_4. So we can walk from any individual forward in time to visit all its descendants. By doing this, can you explain the color coding of black, blue, and red individuals?

Problem 412 Create the directory `Descent` and change into it. Figures 6.1 and 6.2 were drawn using the program `drawGenealogy`, which generates LaTeX output. Use it to simulate new random family histories using options similar to the above Figures. Typeset and display the figures using the commands `latex`, `dvips`, and `gv`.

Problem 413 `drawGenealogy` also has a non-graphical mode. Use it to determine the number of generations until the first universal ancestor appears for populations of sizes 10, 20, 50, 100, 200, 500, and 1000. Compute averages over 100 iterations every time and plot the result. Compare your results to the expectation of $\log_2(N)$, where N is the population size [13]. This can be specified in `gnuplot` by

```
f(x) = log(x) / log(2)
plot f(x)
```

Problem 414 Use `drawGenealogy` to determine the average number of generations until all present-day individuals have identical ancestors. Compare your results to the expectation of $1.77 \times \log_2(N)$ [13].

6.1.2 Uni-Parental Genealogy

Problem 415 In contrast to individuals, who have two parents, genes only have one. Figure 6.3 shows the third version of Fig. 6.1, where the lines of descent are restricted to those of the two genes in individual i_4. What can you say about the common ancestor of those two genes? How does this compare to the ancestry of individuals?

Problem 416 Populations are often modeled as consisting of genes without reference to individuals or gender. This abstract model of a population is called the Wright–Fisher model in honor of two founding figures of population genetics, Ronald A. Fisher (1890–1962) and Sewall Wright (1889–1988). Figure 6.4 shows two generations in the evolution of a Wright–Fisher population. To get from one generation to the next, ancestors are simply picked at random, as indicated by the arrows. To see how this works, extend Fig. 6.4 for one generation by manually drawing another set of eight genes in g_3. To determine the ancestor of the first gene, use

```
awk -v seed = $RANDOM 'BEGIN {srand(seed); print int(
     rand() * 8 + 1)}'
```

and draw the appropriate arrow. Then repeat for the remaining genes.

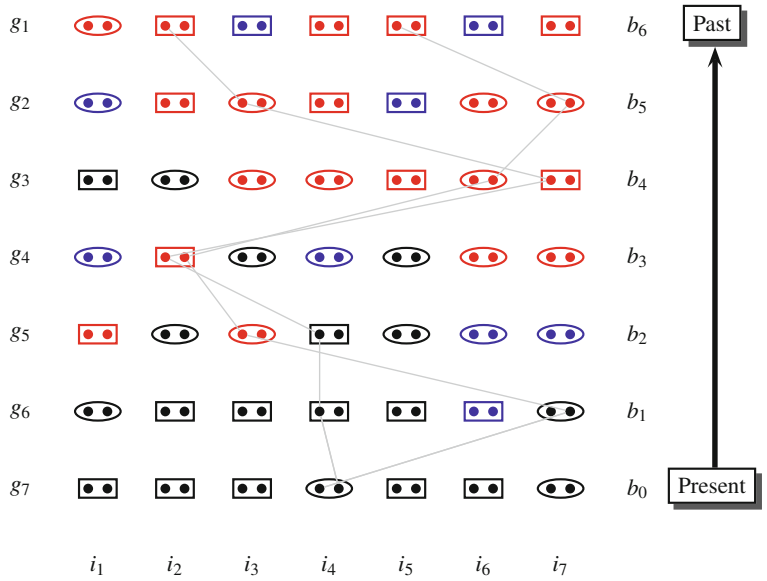

Fig. 6.3 Tracing the genes of individual i_4

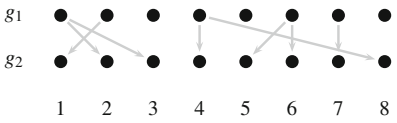

Fig. 6.4 Two generations of a population size 8 under the Wright–Fisher model

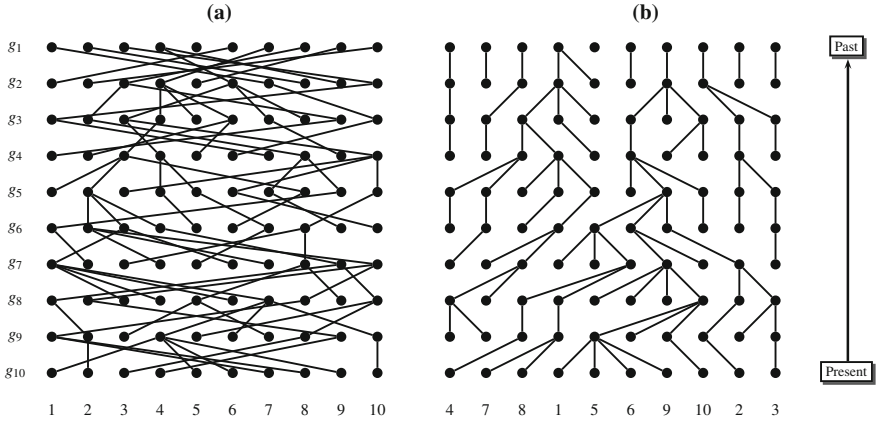

Fig. 6.5 Tangled (**a**) and untangled (**b**) versions of the same Wright–Fisher simulation

Fig. 6.6 Two generations of a population under the Wright–Fisher model consisting of eight genes

Problem 417 Figure 6.5a depicts ten generations of a Wright–Fisher population consisting of ten genes. With all the crisscrossing ancestral lines connecting the genes between generations, it is a bit hard to see what is going on. It is possible to untangle the lines of descent such that no two lines cross. Figure 6.5b shows the untangled version of Fig. 6.5a. This makes it much simpler to trace ancestry back in time. Does Fig. 6.5 contain a common ancestor of all genes?

Problem 418 Use the program `drawWrightFisher` to simulate Wright–Fisher populations in LaTeX. With a population size of ten, how often do you observe a common ancestor for the entire population within ten generations back? Use `-a` for ancestor tracing, `-t` for wrapping the LaTeX code, and carry out, say, ten trials.

Problem 419 When going one generation back in time, the number of lineages that eventually end up in the common ancestor either stays the same, or decreases. For example, in Fig. 6.6 that number goes from originally eight in b_0 to five in b_1. You can think of each gene (dot) in Fig. 6.6 as randomly picking its ancestor in the preceding generation. Occasionally two genes pick the same ancestor. What is the probability of this occurring?

Problem 420 If two genes pick an identical ancestor, their lineages fuse and the number of ancestral lineages is reduced by one. To simulate such "ancestor events", we first generate N random integers between 0 and $N-1$:

```
BEGIN{
    # seed the random number generator
    srand(seed)
    for(i=0; i<N; i++)
         print int(rand()*N)
}
```

Copy this code in the file `trace1.awk`, and when running it, pass the value of N, and a unique seed for the random number generator:

```
awk -f trace1.awk -v seed=$RANDOM -v N=10
```

Use `sort`, `unique`, and `wc` to count the number of ancestral lineages remaining after going back one generation. Repeat this a couple of times. Do you ever observe ten lineages remaining?

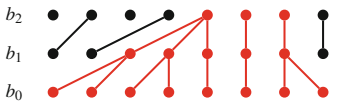

Fig. 6.7 The Wright–Fisher population of Fig. 6.6 traced back one more generation

Problem 421 We saw in Problem 419 that the probability of two genes having the same ancestor in the previous generation is $1/N$. So the probability of two genes *not* having a common ancestor is $1 - 1/N$. The probability of three genes not having the same ancestor is

$$\left(1 - \frac{1}{N}\right)\left(1 - \frac{2}{N}\right)$$

and hence the probability of N genes not having the same ancestor is

$$P_{\mathrm{n}} = \left(1 - \frac{1}{N}\right)\left(1 - \frac{2}{N}\right) \ldots \left(1 - \frac{N-1}{N}\right). \tag{6.1}$$

What is P_{n} for $N = 10$? Check your result through simulation.

Problem 422 Figure 6.7 shows two steps back in time rather than just the single one in Fig. 6.6. If we wish to simulate this second step, we need to distinguish the population size from the number of lineages that end up in the present, or ancestral lineages. We have already seen that after the first generation the number of ancestral lineages is usually smaller than the population size. We now wish to simulate the effect of going back one step when starting from an arbitrary number of ancestral lineages, n, given the population size, N. Here is an edited version of `trace1.awk` for doing this:

```
BEGIN{
     srand(seed)
     # initialize the population
     for(i=0; i<N; i++)
          pop[i] = 0
     # sample with replacement
     for(i=0; i<n; i++){
          j = int(rand()*N)
          pop[j] = 1
     }
     # count the number of lineages
     nn = 0;
     for(i=0; i<N; i++)
          nn += pop[i]
     print nn
}
```

Save your version as `trace2.awk` and run it a few times. How often do you observe a reduction in n if you start with $N = 100$ and $n = 10$?

Problem 423 To calculate the probability of an ancestor event when considering a small number of n lineages in a large population of N genes, we can modify Eq. 6.1 to get

$$P_{\mathrm{n}} = \left(1 - \frac{1}{N}\right)\left(1 - \frac{2}{N}\right) \ldots \left(1 - \frac{n-1}{N}\right).$$

Since N is large, we can ignore terms with N^{-2} or smaller to get

$$P_{\mathrm{n}} \approx 1 - \frac{1}{N} - \frac{2}{N} - \ldots - \frac{n-1}{N}.$$

The complement of this is the probability of an ancestor event

$$P_{\mathrm{a}} = \frac{1 + 2 + \ldots + n - 1}{N} = \frac{n(n-1)}{2N}.$$

What is the probability of observing an ancestor event if $N = 1000$ and $n = 10$? Check your result through simulation.

Problem 424 Finally, we would like to trace the lineages back to their common ancestor. As we have just seen, once the number of ancestral lineages is much smaller than the population size, any further reduction in lineages takes many generations on average. To simulate exactly how many, wrap the code in `trace2.awk` in a `while` loop that repeats until the number of lineages is 1; in other words, until the most recent common ancestor of n genes has been reached.

```
BEGIN{
     srand(seed)
     g = 0    # number of generations
     while(n > 1){
         g++
        for(i=0; i<N; i++)
             pop[i] = 0
        for(i=0; i<n; i++){
             j = int(rand()*N)
             pop[j] = 1
        }
        nn = 0   # new n
        for(i=0; i<N; i++)
             nn += pop[i]
        if(nn < n)
            print g "\t" nn
        n = nn
     }
}
```

Save this as `trace3.awk` and plot the number of lineages as a function of the number of generations for $N = 100$ and $n = 100$.

Problem 425 Run `trace3.awk` a couple of times. What happens to the time to the most recent common ancestor as you switch between the two most extreme values for n possible, $n = 2$ and $n = N$?

Problem 426 The expected time to the most recent common ancestor is [49, p. 76]

$$E\{T_{\text{MRCA}}\} = 2N\left(1 - \frac{1}{n}\right). \tag{6.2}$$

What is the expected TMRCA for $n = 2$ and $n \to \infty$?

Problem 427 Simulate the average T_{MRCA} from 100 replicates for $n = 2, 5, 10, 20$ with $N = 100$ by writing the script `simTmrca.sh` to drive `trace3.awk`. Compare your results to the theoretical expectation in Eq. 6.2.

6.2 The Coalescent

Figure 6.8a shows a population of ten genes evolving for ten generations under the Wright–Fisher model. Investigations of real genes are usually restricted to small samples, say the three genes marked in blue. By untangling the lines of descent, we can see in Fig. 6.8b that these three genes are connected by a tree. If we just concentrate on this tree, we can further reduce it to the nodes where two lines of descent collide as we move from the present into the past. A different way of looking at such a collision is to say that two lines of descent merge or "coalesce" into one.

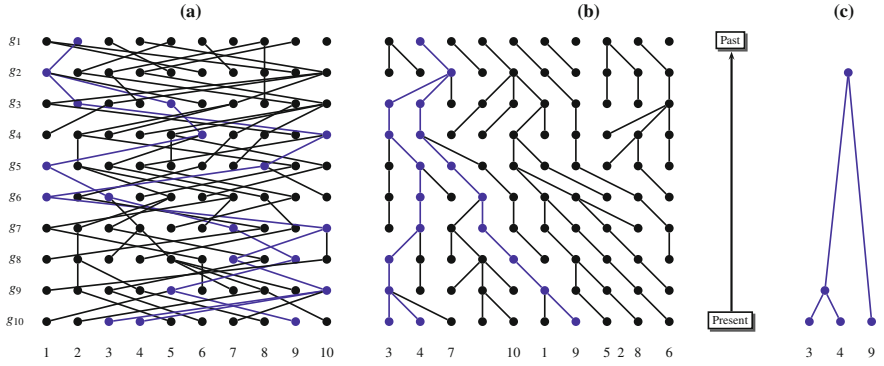

Fig. 6.8 A population under the Wright–Fisher (**a**), its untangled version (**b**), and the coalescent for three of its lineages (**c**)

The collection of such coalescence events is depicted in Fig. 6.8c and is called the "coalescent". It describes the descent of a sample of genes evolving under the Wright–Fisher model from the preset to the most recent common ancestor. As we shall see, the coalescent is a tool for simulating the evolution of genes.

New Concepts

Name	Comment
coalescent	genealogy of gene sample under Wright–Fisher model
Watterson's equation	mutations as a function of population and sample size

New Programs

Name	Source	Help
`coalescent.awk`	book website	`coalescent.awk -v h=1`
`ms`	book website	`ms`
`rpois.awk`	book website	`rpois.awk -v h=1`
`watterson`	book website	`watterson -h`
tabix	book website	`man tabix`

Problem 428 Coalescents are random trees or genealogies. To construct a coalescent, we represent it as an array

Index	1	2	3	4	5	6	7
Node	1	2	3	4	5	6	7

where nodes 1–4 refer to leaves (green) and nodes 5–7 to inner nodes (black). Draw an example tree with these nodes. What is the sample size, n, for this coalescent?

Problem 429 All nodes of a coalescent are annotated with times. The leaves get times 0:

Index	1	2	3	4	5	6	7
Node	1	2	3	4	5	6	7
Time	0	0	0	0			

To compute the times of the inner nodes, we divide the coalescent into intervals T_i, where i is the number of lines of descent in the tree during that time (Fig. 6.9). Where would T_1 be in this graph, and why is it not shown?

Problem 430 To compute T_i, start from the probability of a coalescence event derived in Probelm 423:

$$P_{\mathrm{a}} = \frac{n(n-1)}{2N}.$$

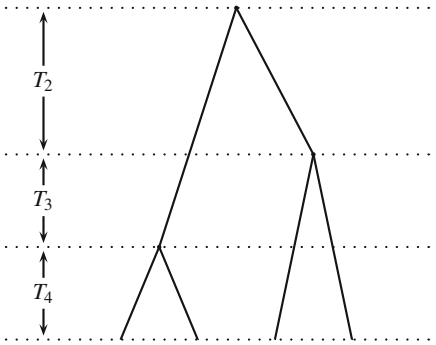

Fig. 6.9 Coalescent with time intervals T_i. These denote the regions of the tree consisting of i lines of descent

This means the time to the next coalescent event is an exponentially distributed random variable with mean $1/P_a$. Since we only know the sample size n and not the population size N, we measure time in units of $2N$ generations and compute T_4

```
awk -v s = $RANDOM 'BEGIN {srand(s); i = 4; r = -2*
  log(1-rand())/i/(i-1);print r}'
```

Use this code to compute T_i for $i = 4, 3, 2$, and fill in the coalescence times in the above table. What is the time to the most recent common ancestor in your coalescent?

Problem 431 We can automate the computation of coalescence times

```
BEGIN{
     srand(seed)
     t=0
     for(i=n;i>=2;i--){
          t -= 2*log(1-rand())/i/(i-1)
          print i, t
     }
}
```

Copy this code into the file `genCoalTimes.awk` and then run it

```
awk -v seed=$RANDOM -v n=10 -f genCoalTimes.awk
```

Wrap this code in a script to determine the average time to the most recent common ancestor (T_{MRCA}) from 100 iterations with $n = 2$ and $n = 1000$.

Problem 432 Having learned how to assign times to nodes, we next learn how to construct a random tree from them—the coalescent. This requires shuffling a set of nodes. Consider the five nodes 1, 2, 3, 4, 5; after shuffling, they might have the order 4, 5, 1, 2, 3. The way to achieve this for n nodes in node array a is as follows:

1. Pick a random number r between 1 and n.

2. Swap the $a[r]$ and $a[n]$.
3. Reduce n by 1.
4. Repeat.

This technique is also known as sampling without replacement. It depends on distinguishing between the value of an index, i, and the value of an array, a, at that position, $a[i]$. Shuffle 1, 2, 3, 4, 5 using the random indexes 1, 3, 1, 2.

Problem 433 To construct a coalescent, we need to apply the shuffling procedure to the array of leaves and internal nodes. For this, we add three auxiliary rows to our table so we can overwrite node labels:

Index	1	2	3	4	5	6	7
Node	1	2	3	4	5	6	7
Child1							
Child2							
Time	0	0	0	0			

We begin with the first internal node, 5, and pick a random child among the four leaves:

```
awk -v s = $RANDOM 'BEGIN{srand(s); print int
    (rand()*4+1)}'
```

Say, this is 1; so the fist child of 5 is 1 and we replace node 1 by the leftmost leaf in this round, 4:

Index	1	2	3	4	5	6	7
Node	1	2	3	4	5	6	7
	4						
Child1					1		
Child2							
Time	0	0	0	0			

We draw another random number, but this time only the first three leaves are candidates

```
awk -v s = $RANDOM 'BEGIN{srand(s); print int
   (rand()*3+1)}'
```

Say that is 2; then the second child of 5 is 2. In addition, 2 is replaced by 5:

Index	1	2	3	4	5	6	7
Node	1	2	3	4	5	6	7
	4	5					
Child1					1		
Child2					2		
Time	0	0	0	0			

So in each round the first child of node v is replaced by the leftmost leaf available in that round, and the second child is replaced by v. Algorithm 3 summarizes these steps. Finish the tree construction.

Algorithm 3 Generate coalescent

Require: n {sample size}
Require: tree {array of n leaves followed by $n-1$ internal nodes}
Ensure: Tree topology
1: **for** $i \leftarrow n$ **to** 2 **do**
2: $p \leftarrow i \times \text{ran}() + 1$ $\{1 \le p \le i\}$
3: tree$[2 \times n - i]$.child1 $\leftarrow$ tree$[p]$
4: tree$[p] \leftarrow$ tree$[i-1]$ {Replace child by the rightmost entry in the "leafy" part of tree}
5: $p \leftarrow (i-1) \times \text{ran}() + 1$ $\{1 \le p \le i-1\}$
6: tree$[2 \times n - i]$.child2 $\leftarrow$ tree$[p]$
7: tree$[p] \leftarrow$ tree$[2 \times n - i]$ {Replace child by parent}
8: **end for**

Problem 434 Sketch the coalescent just constructed.

Problem 435 Picking children can be automated:

```
BEGIN{
     srand(seed)
     print "# Pa\tC1\tC2"
     for(i=n; i>=2; i--){
          child1 = int(rand()*i+1)
          child2 = int(rand()*(i-1)+1)
          print 2*n-i+1 "\t" child1 "\t" child2
    }
}
```

Save this code in the program `pickChildren.awk` and use it together with `genCoalTimes.awk` to generate a coalescent for $n = 4$.

Problem 436 At this stage, we have coalescence times and a branching order. To achieve biological relevance, we still need mutations. They are generated as Poisson-distributed random variables for each branch with expectation

$$\lambda = t\theta/2,$$

where t is the branch length and $\theta = 4N\mu$, where μ is the number of mutations per generation per site. Use the program `rpois.awk` to generate the random variable, for example

```
awk -f rpois.awk -v lambda=0.8
```

As our coalescent has six branches, six mutation counts need to be generated. Compare the total number of mutations on the coalescent with the corresponding expectation according to Watterson's equation [50]

$$E\{S\} = \theta \sum_{i=1}^{n-1} \frac{1}{i},$$

where $E\{S\}$ is the expected number of segregating sites. This equation is implemented in the program `watterson`.

Problem 437 Use the program `coalescent.awk` to simulate two haplotype samples of size 4 with $\theta = 10$. Can you interpret the output?

Problem 438 The program `coalescent.awk` can also print the coalescent tree underlying the simulation of SNPs. Run it for single sample, extract the coalescent with `grep`, and visualize it with `new2view`. Repeat a few times to get a visual impression of the coalescent. What is the meaning of the scale?

Problem 439 The program `ms` is often used for real coalescent simulations [25]. Use it to generate two samples of size 4 with $\theta = 10$, for example,

```
ms 4 2 -t 10

ms 4 2 -t 10
60977 30522 51696

//
segsites: 6
positions: 0.0675 0.1627 0.2495 0.2952 0.3512 0.4482
010111
101000
010111
010111

//
segsites: 12
positions: 0.1269 0.2146 0.3704...
000001000010
000001000010
111110111101
000000000000
```

Row by row this output means:

- Row 1: Repetition of the command
- Row 2: Initialization of the random number generator
- Row 4: Start of the first sample
- Row 5: Number of segregating sites (mutations): 6
- Row 6: Positions of the mutations along the interval (0, 1)
- Rows 7–10: Haplotypes; 0 indicates ancestral state, 1 mutant

Use `ms` to simulate 10^4 samples of size 4 with $\theta = 10$. Use `grep` and AWK to estimate the average number of mutations. Compare this to the expectation value.

Problem 440 Let us say we observe a single mutation in a sample of four aligned DNA sequences, far fewer mutations than the number expected with $\theta = 10$. How significant is the deviation between observation and expectation?

Mouse Genome

Problem 441 Next, we investigate single nucleotide polymorphism (SNP) data in mice. We concentrate on chromosome 19, the smallest mouse chromosome, which was sampled from 17 mice. To query the SNPs in the first 5 Mb of chromosome 19, enter

```
tabix http://guanine.evolbio.mpg.de/problemsBook/chr19.
   mgp.vcf.gz 19:1-5000000
```

What is the location of the first and the last SNP on chromosome 19?

Problem 442 Count the SNPs located between the first and the last SNP (`wc -l`), and infer from this the number of SNPs per nucleotide. What is θ per nucloetide? (Remember, 17 mice were sequenced.)

Problem 443 Count the SNPs in mice between position 5,189,001 and 5,190,000. How many SNPs are expected for this interval? Is the deviation between theory and observation significant?