Chapter 2

Recombination Fraction and Linkage Map Construction

1 Map Functions

Genes are physically located on chromosomes. Because chromosomes are string-like structures, genes carried by chromosomes are arranged linearly (Morgan, 1928). In other words, a particular location of a chromosome can only be occupied by one and only one gene. Figure 1.1 shows a marker map of the barley genome (Hayes et al., 1993), where the names of markers are given in the right hand side of the chromosomes and the marker positions measured in centiMorgan (cM) are given in the left hand side. In genetics, the terms gene and locus are often used interchangeably, but a more precise description of a chromosome location is the locus. Genetic loci carried by the same chromosome are physically linked, and thus they tend to co-segregate. These loci are said to be in the same linkage group. The distribution of loci among chromosomes and the order of loci within chromosomes are called genetic map. Using observed genotypes of these loci to infer the genetic map is called genetic mapping or linkage analysis.

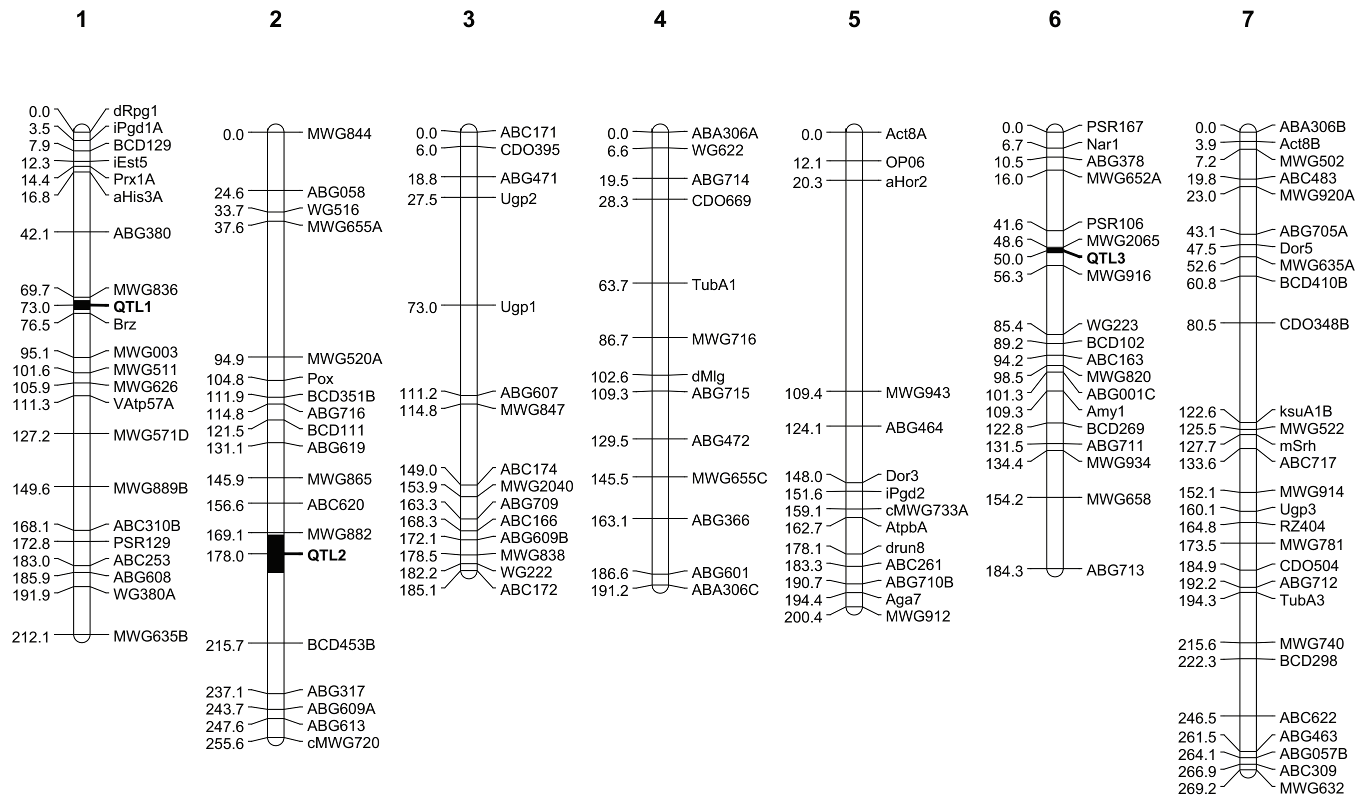

Fig. 1.1. Marker map of the barley genome (seven chromosomes). The names of markers are given in the right hand side of the chromosomes and the marker positions measured in centiMorgan (cM) are given in the left hand side of the chromosomes

1.1 Physical map and genetic map

The size of a genome is determined by the number of chromosomes and the size of each chromosome. In general, there are two types of measurements for the size of a chromosome. One measurement is the number of base pairs (bp) of the DNA molecules carried by the chromosome. The order and relative localization of all the loci in a genome with distances between consecutive loci measured by the numbers of base pairs is called the physical map. Unfortunately, one must sequence the entire chromosome to determine the size of the chromosome and sequence the entire genome to determine the genome size. During meiosis, pairs of duplicated homologous chromosomes unite in synapsis and then non-sister chromatids exchange segments (genetic material) during crossing over, which produces the recombinant gametes. This phenomenon is called crossover. The number of crossovers between two loci depends on the physical distance of the loci. More distant loci tend to have more crossovers between them. The relationship between the number of crossovers and the number of base pairs may be described by a linear function. The slope of the linear relationship varies from one organism to another and even varies across different chromosome regions within a species (Civardi et al., 1994; Chen et al., 2002). Nevertheless, under the assumption of a constant slope across the genome, one can measure the size of a chromosome by the number of crossovers and then infer the number of base pairs from the number of crossovers. Roughly, 1 cM on a chromosome encompasses 1 megabase (Mb) (1 Mb = 10^6 bp) of DNA (Chen et al., 2002). The order and relative positioning of all linked loci in a genome with distances between consecutive loci measured by the numbers of crossovers is called the genetic map, due to the fact that crossover is a genetic phenomenon. Although we can observe crossovers with some special cytogenetic technology, counting the total number of crossovers of the entire genome is still impractical. If a crossover has occurred at a point between two loci during meiosis, the gamete formed will carry a paternal allele for one locus and a maternal allele for the other locus. This mosaic gamete is called the recombined gamete or simply recombinant. Crossover, however, is a random event and it happens with a certain probability during meiosis. If no crossover has happened, the gamete will carry paternal alleles or maternal alleles for both loci. This type of gamete is called the parental gamete. If two loci are far apart, crossover between the two loci may happen twice or multiple times during meiosis. Only an odd number of crossovers will generate recombinants. The proportion of recombinants in a gametic pool is called the recombination fraction or recombination frequency. This fraction depends on the number of crossovers, although not in a linear fashion. In genetic linkage study, people often use recombination fraction in place of the number of crossovers to measure the distances between loci. The order and relative localization of linked loci with distances between consecutive loci measured by the recombination fractions is also called the genetic map. Therefore, we have two different measurements for the genetic distance between two loci: the number of crossovers and the recombination fraction.

1.2 Derivation of map functions

As mentioned early, there are two measurements of map distance between two linked loci, the average number of crossovers (x) and the recombination fraction (r). The number of crossovers itself is a random variable in the sense that it varies from one meiosis to another. The unit of map distance measured this way is Morgan (M), named in honor of geneticist Thomas Hunt Morgan. One Morgan is defined as the length of a chromosome segment bracketed by two loci that produces, on average, one crossover per meiosis. In other words, if we can observe many meioses in a genetic experiment for the segment under investigation, some meioses may produce no crossover, some may produce one crossover and others may produce more than one crossovers. But, *on average*, a segment of 1 M produces one crossover. We often use 1/100 Morgan as the unit of map distance, called one centiMorgan (cM). Note that the unit Morgan is rarely used today.

Genetic map distance measured by Morgan or centiMorgan is additive. Assume that three loci are ordered as A, B and C with x_{AB} and x_{BC} representing the map distance between A and B and the distance between B and C, respectively, then the distance between A and C is $x_{AC} = x_{AB} + x_{BC}$. Because of this property, we call the map distance measured in Morgan or centiMorgan the additive distance, as apposed to the distance measured in recombination fraction. Recombination fraction between two loci is defined as the ratio of the number of recombined gametes to the total number of gametes produced. For example, assume that we sample 100 gametes for a chromosome segment bracketed by loci A and C in a genetic experiment, if 12 gametes are recombinants and the remaining 88 gametes are parental gametes, then the recombination fraction between A and C is $r_{BC} = 0.12$. Recombination fraction is not additive in the sense that $r_{AC} \neq r_{AB} + r_{BC}$. This is because only odd numbered crossovers can generate a recombinant.

If two loci overlap, no crossover will be expected between the two loci and both x_{AC} and r_{AC} will be zero. On the other hand, if the two loci are far away (almost unlinked), an infinite number of crossovers will be expected. When the segment is sufficiently long, odd and even numbered crossovers will be roughly equal, leading to a recombination fraction of $\frac{1}{2}$. Therefore, recombination fraction ranges between 0 and $\frac{1}{2}$. The relationship between r_{AC} and x_{AC} can be described by a non-linear function, called the map function. In linkage analysis, we usually estimate the recombination frequency between loci and then convert the frequency into the additive distance and report the additive map distance. Several functional relationships have been proposed (Zhao and Speed, 1996; Liu, 1998), but the most commonly used functions are the Haldane map function (Haldane, 1919) and its extension, called the Kosambi map function (Kosambi, 1943).

Consider three ordered loci A-B-C. If the probability of crossover between A and B and that between B and C are independent, then the probability of double crossovers between A and C should take the product of r_{AB} and r_{BC}. The recombination frequency between A and C should be

$$r_{AC} = r_{AB}(1 - r_{BC}) + r_{BC}(1 - r_{AB}) = r_{AB} + r_{BC} - 2r_{AB}r_{BC}. \tag{1.1}$$

The independence of crossovers between two nearby intervals is called no interference. However, it is often observed that if the two intervals under consideration are too close, the crossover of one interval may completely prevent the occurrence of the crossover of the other interval. This phenomenon is called complete interference, in which case the recombination fraction between A and C is described by

$$r_{AC} = r_{AB} + r_{BC}. \tag{1.2}$$

This relationship was given by Morgan and Bridges (1916). Under the assumption of complete interference, the recombination frequencies are additive, like the additive map distance measured by the average number of crossovers. When loci A and C are very closely linked, the recombination fractions are approximately additive, even if interference is not complete. In practice, an intermediate level of interference is most likely to occur. Therefore, the recombination fraction between A and C can be described by

$$r_{AC} = r_{AB} + r_{BC} - 2cr_{AB}r_{BC}. \tag{1.3}$$

where $0 \leq c \leq 1$ is called the coefficient of coincidence whose complement $i = 1 - c$ is called the coefficient of interference. Equations (1.1) and (1.2) are special cases of equation (1.3), where $c = 1$ for equation (1.1) and $c = 0$ for equation (1.2).

We now develop a general map function to describe the relationship between the recombination frequency r and the additive map distance x measured in Morgan. Let the map function be $r = r(x)$, indicating that r is a function of x. The above equation can be rewritten as

$$\underbrace{r(x_{AB} + x_{BC})}_{r_{AC}} = \underbrace{r(x_{AB})}_{r_{AB}} + \underbrace{r(x_{BC})}_{r_{BC}} - 2c\,\underbrace{r(x_{AB})}_{r_{AB}}\underbrace{r(x_{BC})}_{r_{BC}}. \tag{1.4}$$

Let $x = x_{AB}$ be the map distance between loci A and B and $\Delta x = x_{BC}$ be the increment of the map distance (extension from locus B to locus C). We can then rewrite the above equation as

$$\underbrace{r(x + \Delta x)}_{r_{AC}} = \underbrace{r(x)}_{r_{AB}} + \underbrace{r(\Delta x)}_{r_{BC}} - 2c\,\underbrace{r(x)}_{r_{AB}}\underbrace{r(\Delta x)}_{r_{BC}}, \tag{1.5}$$

which can be rearranged into

$$r(x + \varDelta x) - r(x) = r(\varDelta x) - 2cr(x)r(\varDelta x). \tag{1.6}$$

Dividing both sides of the equation by $\varDelta x$, we have

$$\frac{r(x + \varDelta x) - r(x)}{\varDelta x} = \frac{r(\varDelta x) - 2cr(x)r(\varDelta x)}{\varDelta x}. \tag{1.7}$$

Recall that when $\varDelta x$ is small, $r(\varDelta x) \approx \varDelta x$. Therefore, we can take the following limit

$$\lim_{\varDelta x \to 0} \frac{r(x + \varDelta x) - r(x)}{\varDelta x} = \lim_{\varDelta x \to 0} \frac{r(\varDelta x)}{\varDelta x} - 2c \lim_{\varDelta x \to 0} \frac{r(x)r(\varDelta x)}{\varDelta x}. \tag{1.8}$$

Because $\lim_{\varDelta x \to 0} \frac{r(x+\varDelta x)-r(x)}{\varDelta x} = \frac{d}{dx} r(x)$ and $\lim_{\varDelta x \to 0} \frac{r(\varDelta x)}{\varDelta x} = 1$, we have

$$\frac{d}{dx} r(x) = 1 - 2cr(x). \tag{1.9}$$

This is a differential equation, which can be easily solved if c is specified. This differential equation was derived by Haldane (1919) and is the theoretical basis from which the following specific map functions are derived.

1.3 Haldane map function

If c is treated as a constant, we can solve the above differential equation for function $r(x)$. We then obtain

$$x = -\frac{1}{2c} \ln[1 - 2cr(x)]. \tag{1.10}$$

Solving for $r(x)$, we get

$$r(x) = \frac{1}{2c}(1 - e^{-2cx}). \tag{1.11}$$

When $c = 1$ is assumed, i.e., there is no interference, we have obtained the following well known Haldane map function (Haldane, 1919)

$$r(x) = \frac{1}{2}(1 - e^{-2x}). \tag{1.12}$$

Note that when using the Haldane map function, the genetic distance x should be measured in Morgan, not in centiMorgan.

1.4 Kosambi map function

Kosambi (1943) expressed the coefficient of incidence as a function of the recombination frequency, i.e., $c = 2r(x)$, based on the notion that $c = 0$ when the two intervals are very close (complete interference) and $c = 1$ when the two intervals are virtually not linked (no interference). Substituting $c = 2r(x)$ into (1.9), we have

$$\frac{d}{dx}r(x) = 1 - 4r^2(x). \tag{1.13}$$

Solving this differential equation for $r(x)$, we have

$$x = \frac{1}{4}\ln\frac{1+2r(x)}{1-2r(x)}. \tag{1.14}$$

A rearrangement on equation (1.14) leads to the famous Kosambi map function (Kosambi, 1943),

$$r(x) = \frac{1}{2}\left(\frac{e^{4x}-1}{e^{4x}+1}\right). \tag{1.15}$$

The main difference between Haldane and Kosambi map functions is the assumption of interference between two consecutive intervals. Haldane map function assumes no interference whereas Kosambi map function allows interference to occur. This can be reflected by the following two equations under the two map functions. Under the Haldane map function,

$$r_{AC} = r_{AB} + r_{BC} - 2r_{AB}r_{BC}, \tag{1.16}$$

whereas under the Kosambi map function,

$$r_{AC} = \frac{r_{AB} + r_{BC}}{1 + 4r_{AB}r_{BC}}. \tag{1.17}$$

Similar to the Haldane map function, when using the Kosambi map function, the genetic distance x should be measured in Morgan, not in centiMorgan.

Kosambi map function often fits data better than Haldane map function because it takes into account interference. Haldane map function has an attractive Markovian property, which can be used elegantly to analyze multiple loci. Figure 1.2 illustrates the difference between the two map functions. For the same map distance between two loci, the recombination fraction calculated based on the Haldane map function is always smaller than that calculated based on the Kosambi map function.

The Haldane and Kosambi map functions are the most popular ones used in genetic mapping. There are many other map functions available in the literature. Most of them are extensions of these two map functions. These additional map functions can be found in Liu (1998) . It is important to understand that the map functions apply to diploid organisms. For polyploid organisms, e.g., autotetraploids, the map function can be different due to problems of multiple dosage of allelic inheritance, the null allele, allelic segregation distortion, mixed bivalent and quadrivalent pairing in meiosis. For example, the maximum recombination frequency can be 0.75 in tetraploid species rather than 0.5 in diploid. Details of tetraploid map function and map construction can be found in the seminal studies conducted by Luo et al. (2004) and Luo et al. (2006).

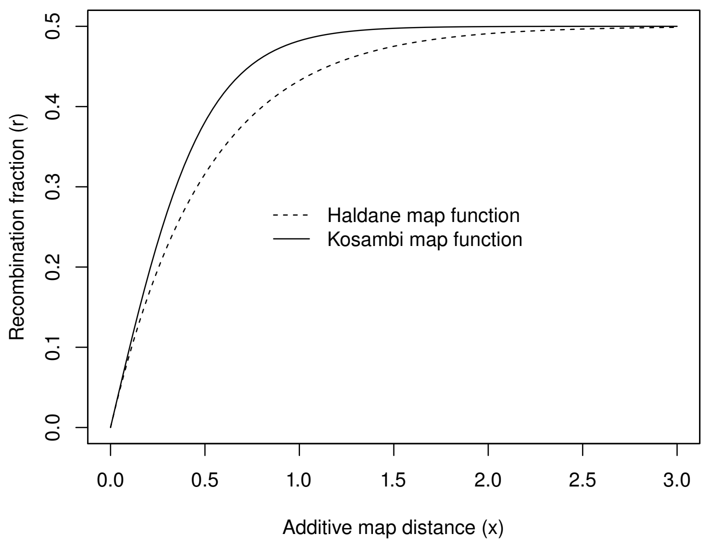

Fig. 1.2. Comparison of the Haldane and Kosambi map functions

2

Recombination Fraction

Recombination fraction (also called recombination frequency) between two loci is defined as the ratio of the number of recombined gametes to the total number of gametes produced. Recombination fraction, denoted by r throughout the book, however, has a domain of $0 \leq r \leq 0.5$, with $r = 0$ indicating perfect linkage and $r = 0.5$ meaning complete independence of the two loci. In most situations, gametes are not directly observable. Therefore, special mating designs are required to infer the number of recombined gametes. When a designed mating experiment cannot be carried out, data collected from pedigrees can be used for estimating recombination fractions. However, inferring the number of recombined gametes in pedigrees is much more complicated than that in designed mating experiments. This book only deals with designed mating experiments.

2.1 Mating designs

Two mating designs are commonly used in linkage study, the backcross (BC) design and the $\mathtt{F}_2$ design. Both designs require two inbred lines, which differ in both the phenotypic values of traits (if marker-trait association study is to be performed) and allele frequencies of marker loci used for constructing the linkage map. We will use two marker loci as an example to show the mating designs and methods for estimating recombination fraction. The BC design is demonstrated in Fig. 2.1. Let A and B be the two loci under investigation. Let A_1 and A_2 be the two alleles at locus A, and B_1 and B_2 be the two alleles at locus B. Let $\mathtt{P}_1$ and $\mathtt{P}_2$ be the two parents that initiate the line cross. Since both parents are inbred, we can describe the two-locus genotype for $\mathtt{P}_1$ and $\mathtt{P}_2$ by $\frac{A_1B_1}{A_1B_1}$ and $\frac{A_2B_2}{A_2B_2}$, respectively. The hybrid progeny of cross between $\mathtt{P}_1$ and $\mathtt{P}_2$ is denoted by $\mathtt{F}_1$ whose genotype is $\frac{A_1B_1}{A_2B_2}$. The horizontal line in the $\mathtt{F}_1$ genotype separates the two parental gametes, i.e., A_1B_1 is the gamete from $\mathtt{P}_1$ and A_2B_2 is the gamete from $\mathtt{P}_2$. The $\mathtt{F}_1$ hybrid crosses back to one of the parents to generate multiple BC progeny, which will be used for linkage study. The BC population is a segregating population. Linkage analysis can only be conducted in such a segregating population. A segregating population is defined as a population that contains individuals with different genotypes. The two parental populations and the $\mathtt{F}_1$ hybrid population are not segregating populations because individuals within each of the three populations are genetically identical. The BC progeny generated by $\mathtt{F}_1 \times \mathtt{P}_1$ is called $\mathtt{BC}_1$ whereas the BC population generated by $\mathtt{F}_1 \times \mathtt{P}_2$ is called $\mathtt{BC}_1^{'}$.

We now use $\mathtt{BC}_1$ progeny as an example to demonstrate the BC analysis. The gametes generated by the $\mathtt{P}_1$ parent are all of the same type A_1B_1. However, the $\mathtt{F}_1$ hybrid can generate four different gametes, and thus four distinguished genotypes. Let r be the recombination fraction between loci A and B. Let n_{ij} be the number of gametes of type A_iB_j or the number of genotype of $\frac{A_iB_j}{A_1B_1}$ kind for $i, j = 1, 2$. The four genotypes and their frequencies are given in Table 2.1. This table provides the data for the maximum likelihood estimation of recombination fraction. The maximum likelihood method will be described later.

The $\mathtt{F}_2$ mating design requires mating of the hybrid with itself, called selfing and denoted by the symbol $\otimes$ (see Figure 2.2 for the $\mathtt{F}_2$ design). When selfing is impossible, e.g., in animals and self incompatible plants, intercross between different $\mathtt{F}_1$ individuals initiated from the same cross is required. The progeny of selfing $\mathtt{F}_1$ or intercross between two $\mathtt{F}_1$ hybrids is called an $\mathtt{F}_2$ progeny. An $\mathtt{F}_2$ family consists of multiple $\mathtt{F}_2$ progeny. The $\mathtt{F}_2$ family represents another segregating population for linkage analysis. Recall that an $\mathtt{F}_1$ hybrid

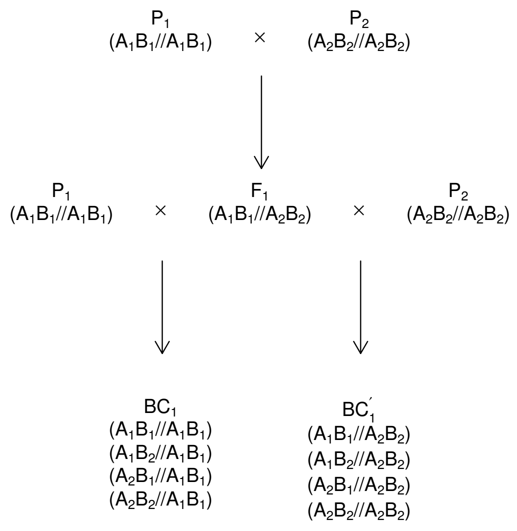

Fig. 2.1. The backcross (BC) mating design. The BC progeny generated by $\mathtt{F}_1 \times \mathtt{P}_1$ is called $\mathtt{BC}_1$ whereas the BC population generated by $\mathtt{F}_1 \times \mathtt{P}_2$ is called $\mathtt{BC}_1^{'}$

.

Table 2.1. Count data of two locus genotypes collected from a $\mathtt{BC}_1$ family

Genotype	Count	Frequency	Type
$\frac{A_1B_1}{A_1B_1}$	n_{11}	$\frac{1}{2}(1-r)$	parental
$\frac{A_1B_2}{A_1B_1}$	n_{12}	$\frac{1}{2}r$	recombinant
$\frac{A_2B_1}{A_1B_1}$	n_{21}	$\frac{1}{2}r$	recombinant
$\frac{A_2B_2}{A_1B_1}$	n_{22}	$\frac{1}{2}(1-r)$	parental

can generate four possible gametes for loci A and B jointly. Therefore, selfing of $\mathtt{F}_1$ can generate 16 possible genotypes, as illustrated in Table 2.2. Let n_{ij} be the number of individuals combining the ith gamete from one parent and the jth gamete from the other parent, for $i, j = 1, \ldots, 4$. The frequencies of all the 16 possible genotypes are listed in Table 2.4. This table is the basis from which the maximum likelihood estimation of recombination fraction will be derived.

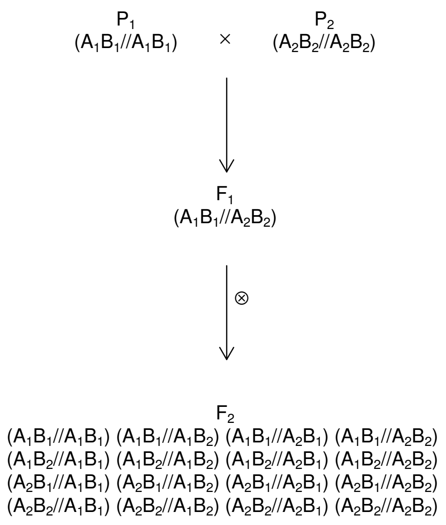
P1
(A1B1//A1B1)
×
P2
(A2B2//A2B2)
F1
(A1B1//A2B2)
⊗
F2
(A1B1//A1B1) (A1B1//A1B2) (A1B1//A2B1) (A1B1//A2B2)
(A1B2//A1B1) (A1B2//A1B2) (A1B2//A2B1) (A1B2//A2B2)
(A2B1//A1B1) (A2B1//A1B2) (A2B1//A2B1) (A2B1//A2B2)
(A2B2//A1B1) (A2B2//A1B2) (A2B2//A2B1) (A2B2//A2B2)

Fig. 2.2. The $\mathtt{F}_2$ mating design

Table 2.2. The 16 possible genotypes and their observed counts in an $\mathtt{F}_2$ family.

	A_1B_1	A_1B_2	A_2B_1	A_2B_2
A_1B_1	$\frac{A_1B_1}{A_1B_1}, n_{11}$	$\frac{A_1B_1}{A_1B_2}, n_{12}$	$\frac{A_1B_1}{A_2B_1}, n_{13}$	$\frac{A_1B_1}{A_2B_2}, n_{14}$
A_1B_2	$\frac{A_1B_2}{A_1B_1}, n_{21}$	$\frac{A_1B_2}{A_1B_2}, n_{22}$	$\frac{A_1B_2}{A_2B_1}, n_{23}$	$\frac{A_1B_2}{A_2B_2}, n_{24}$
A_2B_1	$\frac{A_2B_1}{A_1B_1}, n_{31}$	$\frac{A_2B_1}{A_1B_2}, n_{32}$	$\frac{A_2B_1}{A_2B_1}, n_{33}$	$\frac{A_2B_1}{A_2B_2}, n_{34}$
A_2B_2	$\frac{A_2B_2}{A_1B_1}, n_{41}$	$\frac{A_2B_2}{A_1B_2}, n_{42}$	$\frac{A_2B_2}{A_2B_1}, n_{43}$	$\frac{A_2B_2}{A_2B_2}, n_{44}$

2.2 Maximum likelihood estimation of recombination fraction

In a BC design, the four types of gametes are distinguishable. Therefore, the recombination fraction can be directly calculated by taking the ratio of the number of recombinants to the total number of gametes. We use $\mathtt{BC}_1$ as an example to demonstrate the method. The count data are given in Table 2.1. Let $n_p = n_{11} + n_{22}$ be the number of individuals carrying the parental gametes and $n_r = n_{12} + n_{21}$ be the number of recombinants. The estimated recombination fraction between loci A and B is simply.

$$\hat{r} = \frac{n_{12} + n_{21}}{n_{11} + n_{12} + n_{21} + n_{22}} = \frac{n_r}{n_r + n_p}. \tag{2.1}$$

We use a hat above r to indicate estimation of r. The true value of recombination fraction is not known but if the sample size is infinitely large, the estimated r will approach to the true value, meaning that the estimation is unbiased.

We now prove that $\hat{r}$ is the maximum likelihood estimate (MLE) of r. We introduce the ML method because it provides a significance test on the hypothesis that $r = 0.5$. To construct the likelihood function, we need a probability model, a sample of data and a parameter. The probability model is the binomial distribution, the data are the counts of the two possible genotypes and the parameter is r. The binomial probability of the data given the parameter is

$$\Pr(n_r, n_p|r) = \frac{n!}{n_r!n_p!}\left(\frac{1}{2}\right)^{n_r+n_p} r^{n_r}(1-r)^{n_p}, \tag{2.2}$$

where $n = n_r + n_p$ is the sample size. The value of r for $0 \le r \le 0.5$ that maximizes the probability is the MLE of r. Two issues need to be emphasized here for any maximum likelihood analysis, including this one. First, the probability involves a factor that does not depend on the parameter,

$$\texttt{const} = \frac{n!}{n_r!n_p!}\left(\frac{1}{2}\right)^n. \tag{2.3}$$

It is a constant with respect to the parameter r. This constant is irrelevant to the ML analysis and thus should be ignored. Secondly, the r value that maximizes a monotonic function of the probability also maximizes this probability. For computational convenience, we can maximize the logarithm of the probability. Therefore, it is the log likelihood function that is maximized in the ML analysis. The log likelihood function is defined as

$$L(r) = n_r \ln r + n_p \ln(1 - r). \tag{2.4}$$

To find the MLE of r, we need to find the derivative of $L(r)$ with respect to r,

$$\frac{d}{dr}L(r) = \frac{n_r}{r} - \frac{n_p}{1-r}. \tag{2.5}$$

Letting $\frac{d}{dr}L(r) = 0$ and solving for r, we have

$$\hat{r} = \frac{n_r}{n_r + n_p}. \tag{2.6}$$

which is identical to that given in equation (2.1).

2.3 Standard error and significance test

A parameter is a fixed but unknown quantity. The estimate of the parameter, however, is a variable because it varies from one sample to another. As the sample size increases, the estimate will approach to the true value of the parameter, provided that the estimate is unbiased. The deviation of the estimate from the true parameter can be measured by the standard error of the estimate. In this section, we will learn a method to calculate the standard error of $\hat{r}$. To calculate the standard error, we need the second derivative of the log likelihood function with respect to r and obtain a quantity called information, from which the variance of the estimated r can be approximated. Let us call the first derivative of $L(r)$ with respect to r the score function, denoted by $S(r)$,

$$S(r) = \frac{d}{dr}L(r) = \frac{n_r}{r} - \frac{n_p}{1-r}. \tag{2.7}$$

The second derivative of $L(r)$ with respect to r is called the Hessian matrix, denoted by $H(r)$,

$$H(r) = \frac{d}{dr}S(r) = \frac{d^2}{dr^2}L(r) = \frac{n_p}{(1-r)^2} - \frac{n_r}{r^2}. \tag{2.8}$$

Although $H(r)$ is a single variable, we still call it a matrix because in subsequent chapters we will deal with multiple dimension of parameters, in which case $H(r)$ is a matrix. From $H(r)$ we can find the information of r, which is

$$I(r) = -\mathtt{E}[H(r)] = \frac{\mathtt{E}(n_r)}{r^2} - \frac{\mathtt{E}(n_p)}{(1-r)^2}. \tag{2.9}$$

The symbol E represents expectation of the data given the parameter value. Here, the data are referred to n_r and n_p, not n, which is the sample size (a constant). Suppose that we know the true parameter r, what is the expected number of recombinants if we sample n individuals? This expected number is $\mathtt{E}(n_r) = rn$. The expected number of the parental types is $\mathtt{E}(n_p) = (1-r)n$. Therefore, the information is

$$I(r) = -\mathtt{E}[H(r)] = \frac{rn}{r^2} - \frac{(1-r)n}{(1-r)^2} = \frac{n}{r(1-r)}. \tag{2.10}$$

The variance of the estimated r takes the inverse of the information, with the true parameter replaced by $\hat{r}$,

$$\mathtt{var}(\hat{r}) \approx I^{-1}(\hat{r}) = \frac{\hat{r}(1-\hat{r})}{n}. \tag{2.11}$$

Therefore, the standard error of $\hat{r}$ is

$$\mathtt{se}(\hat{r}) = \sqrt{\mathtt{var}(\hat{r})} = \sqrt{\frac{\hat{r}(1-\hat{r})}{n}}. \tag{2.12}$$

The standard error is inversely proportional to the square root of the sample size and thus approaches zero as n becomes infinitely large.

When we report the estimated recombination fraction, we also need to report the estimation error in a form like $\hat{r} \pm \mathtt{se}(\hat{r})$. In addition to the sample size, the estimation error is also a function of the recombination fraction, with the maximum error occurring at $r = \frac{1}{2}$, i.e., when the two loci are unlinked. To achieve the same precision of estimation, it requires a larger sample to estimate a recombination fraction between two loosely linked loci than between two closely linked loci.

Because of the sampling error, even two unlinked loci may look like being linked as the estimated r may be superficially smaller than 0.5. How small an $\hat{r}$ is sufficiently small so that we can claim that the two loci are linked in the same chromosome? This requires a significance test.

The null hypothesis for such a test is denoted by $H_0 : r = \frac{1}{2}$. Verbally, H_0 is stated that the two loci are not linked. The alternative hypothesis is $H_A : r < 1/2$, i.e., the two loci are linked on the same chromosome. When the sample size is sufficiently large, we can always use the z-test to decide which hypothesis should be accepted. Here, we will use the usual likelihood ratio test statistic to declare the statistical significance of $\hat{r}$. Let $L(r)|_{r=\hat{r}} = L(\hat{r})$ be the log likelihood function evaluated at the MLE of r using equation (2.4).

Let $L(r)|_{r=\frac{1}{2}} = L(1/2)$ be the log likelihood function evaluated under the null hypothesis. The likelihood ratio test statistic is defined as

$$\lambda = -2[L(1/2) - L(\hat{r})]. \tag{2.13}$$

where

$$L(\hat{r}) = n_r \ln \hat{r} + n_p \ln(1 - \hat{r}). \tag{2.14}$$

and

$$L(1/2) = -n \ln 2 = -0.6931n. \tag{2.15}$$

If the null hypothesis is true, λ will approximately follow a Chi-square distribution with one degree of freedom. Therefore, if $\lambda > \chi^2_{1,1-\alpha}$, we will claim that the two loci are linked, where $\chi^2_{1,1-\alpha}$ is the $(1 - \alpha) \times 100$ percentile of the central χ^2_1 distribution and α is the Type I error determined by the investigator. In human linkage studies, people often use LOD (log of odds) score instead. The relationship between LOD and λ is

$$\texttt{LOD} = \frac{\lambda}{2\ln(10)} \approx 0.2171\lambda. \tag{2.16}$$

Conventionally, $\texttt{LOD} > 3$ is used as a criterion to declare a significant linkage. This converts to a likelihood ratio criterion of $\lambda > 3 \times \ln(100) = 13.81551$. The LOD criterion has an intuitive interpretation. A LOD of k means that the alternative model (linkage) is 10^k times more likely than the null model.

2.4 Fisher's scoring algorithm for estimating r

The $\texttt{F}_2$ mating design is demonstrated in Figure 2.2. The ML analysis described for the $\texttt{BC}_1$ mating design is straightforward. The MLE of r has an explicit form. In fact, there is no need to invoke the ML analysis for the BC design other than to demonstrate the basic principle of the ML analysis. To estimate r using an $\texttt{F}_2$ design, the likelihood function is constructed using the same probability model (multinomial distribution), but finding the MLE of r is complicated. Therefore, we will resort to some special maximization algorithms. The algorithm we will learn is the Fisher's scoring algorithm (Fisher, 1946).

Let us look at the genotype table (Table 2.2) and the table of genotype counts and frequencies (Table 2.4) for the $\texttt{F}_2$ design. If we were able to observe all the 16 possible genotypes, the same ML analysis used in the BC design would apply here to the $\texttt{F}_2$ design. Unfortunately, some of the genotypes listed in Table 2.2 are not distinguishable from others. For example, genotypes $\frac{A_1B_1}{A_1B_2}$ and $\frac{A_1B_2}{A_1B_1}$ are not distinguishable. These two genotypes appear to be the same because they both carry an A_1B_1 gamete and an A_1B_2 gamete. However, the

Table 2.3. The counts (in parentheses) and frequencies of the 16 possible genotypes in an $\mathtt{F}_2$ family.

	A_1B_1	A_1B_2	A_2B_1	A_2B_2
A_1B_1	$(n_{11})\ \frac{1}{4}(1-r)^2$	$(n_{12})\ \frac{1}{4}r(1-r)$	$(n_{13})\ \frac{1}{4}r(1-r)$	$(n_{14})\ \frac{1}{4}(1-r)^2$
A_1B_2	$(n_{21})\ \frac{1}{4}r(1-r)$	$(n_{22})\ \frac{1}{4}r^2$	$(n_{23})\ \frac{1}{4}r^2$	$(n_{24})\ \frac{1}{4}r(1-r)$
A_2B_1	$(n_{31})\ \frac{1}{4}r(1-r)$	$(n_{32})\ \frac{1}{4}r^2$	$(n_{33})\ \frac{1}{4}r^2$	$(n_{34})\ \frac{1}{4}r(1-r)$
A_2B_2	$(n_{41})\ \frac{1}{4}(1-r)^2$	$(n_{42})\ \frac{1}{4}r(1-r)$	$(n_{43})\ \frac{1}{4}r(1-r)$	$(n_{44})\ \frac{1}{4}(1-r)^2$

origins of the two gametes are different for the two genotypes. Furthermore, the four genotypes in the minor diagonal of Table 2.2 actually represent four different linkage phases of the same observed genotype (double heterozygote). If we consider the origins of the alleles, there are four possible genotypes for each locus. However, the two configurations of the heterozygote are not distinguishable. Therefore, there are only three observable genotypes for each locus, making a total of nine observable joint two-locus genotypes, as shown in Table 2.4. Let m_{ij} be the counts of the joint genotype combining the ith genotype of locus A and the jth genotype of locus B, for $i, j = 1, \ldots, 3$. These counts are the data from which a likelihood function can be constructed.

Table 2.4. The nine observed genotypes and their counts in an $\mathtt{F}_2$ population

	B_1B_1	B_1B_2	B_2B_2
A_1A_1	$A_1A_1B_1B_1\ (m_{11})$	$A_1A_1B_1B_2\ (m_{12})$	$A_1A_1B_2B_2\ (m_{13})$
A_1A_2	$A_1A_2B_1B_1\ (m_{21})$	$A_1A_2B_1B_2\ (m_{22})$	$A_1A_2B_2B_2\ (m_{23})$
A_2A_2	$A_2A_2B_1B_1\ (m_{31})$	$A_2A_2B_1B_2\ (m_{32})$	$A_2A_2B_2B_2\ (m_{33})$

Before we construct the likelihood function, we need to find the probability for each of the nine observed genotypes. These probabilities are listed in Table 2.5. The count data in the second column and the frequencies in the third column of Table 2.5 are what we need to construct the log likelihood function, which is

$$\begin{aligned}L(r) &= \sum_{i=1}^{3}\sum_{j=1}^{3} m_{ij}\ln(q_{ij}) \\ &= [2(m_{11}+m_{33}) + m_{12} + m_{21} + m_{23} + m_{32}]\ln(1-r) \\ &\quad + [2(m_{13}+m_{31}) + m_{12} + m_{21} + m_{23} + m_{32}]\ln(r) \\ &\quad + m_{22}\ln[r^2 + (1-r)^2]. \end{aligned} \tag{2.17}$$

The derivative of $L(r)$ with respect to r is

$$
\begin{aligned}
S(r) &= \frac{d}{dr}L(r) \\
&= -\frac{2(m_{11}+m_{33})}{1-r} + \frac{(m_{12}+m_{21}+m_{23}+m_{32})(1-2r)}{r(1-r)} \\
&\quad - \frac{2m_{22}(1-2r)}{1-2r+2r^2} + \frac{2(m_{13}+m_{31})}{r}.
\end{aligned} \tag{2.18}
$$

Table 2.5. Frequencies of the nine observed genotypes in an $\mathtt{F}_2$ population

Genotype	Count	Probability
$A_1A_1B_1B_1 = \frac{A_1B_1}{A_1B_1}$	$m_{11} = n_{11}$	$q_{11} = \frac{1}{4}(1-r)^2$
$A_1A_1B_1B_2 = \frac{A_1B_1}{A_1B_2}, \frac{A_1B_2}{A_1B_1}$	$m_{12} = n_{12} + n_{21}$	$q_{12} = \frac{1}{2}r(1-r)$
$A_1A_1B_2B_2 = \frac{A_1B_2}{A_1B_2}$	$m_{13} = n_{22}$	$q_{13} = \frac{1}{4}r^2$
$A_1A_2B_1B_1 = \frac{A_1B_1}{A_2B_1}, \frac{A_2B_1}{A_1B_1}$	$m_{21} = n_{13} + n_{31}$	$q_{21} = \frac{1}{2}r(1-r)$
$A_1A_2B_1B_2 = \frac{A_1B_1}{A_2B_2}, \frac{A_1B_2}{A_2B_1}$	$m_{22} = n_{14} + n_{23} +$	$q_{22} = \frac{1}{2}[r^2 + (1-r)^2]$
$\frac{A_2B_1}{A_1B_2}, \frac{A_2B_2}{A_1B_1}$	$n_{32} + n_{41}$	
$A_1A_2B_2B_2 = \frac{A_1B_2}{A_2B_2}, \frac{A_2B_2}{A_1B_2}$	$m_{23} = n_{24} + n_{42}$	$q_{23} = \frac{1}{2}r(1-r)$
$A_2A_2B_1B_1 = \frac{A_2B_1}{A_2B_1}$	$m_{31} = n_{33}$	$q_{31} = \frac{1}{4}r^2$
$A_2A_2B_1B_2 = \frac{A_2B_1}{A_2B_2}, \frac{A_2B_2}{A_2B_1}$	$m_{32} = n_{34} + n_{43}$	$q_{32} = \frac{1}{2}r(1-r)$
$A_2A_2B_2B_2 = \frac{A_2B_2}{A_2B_2}$	$m_{33} = n_{44}$	$q_{33} = \frac{1}{4}(1-r)^2$

The MLE of r is obtained by setting $S(r) = 0$ and solving for r. Unfortunately, there is no explicit solution for r. Therefore, an iterative algorithm is resorted to solve for r. Before introducing the Fisher's scoring algorithm (Fisher, 1946), we first try the Newton method, which also requires the second derivative of $L(r)$ with respect to r,

$$
\begin{aligned}
H(r) &= \frac{d}{dr}S(r) = \frac{d^2}{dr^2}L(r) \\
&= -\frac{2(m_{11}+m_{33})}{(1-r)^2} - \frac{(m_{12}+m_{21}+m_{23}+m_{32})(1-2r+2r^2)}{r^2(1-r)^2} \\
&\quad + \frac{8m_{22}r(1-r)}{(1-2r+2r^2)^2} - \frac{2(m_{13}+m_{31})}{r^2}.
\end{aligned} \tag{2.19}
$$

The Newton method starts with an initial value of r, denoted by $r^{(t)}$ for $t = 0$, and update the value by

$$
r^{(t+1)} = r^{(t)} - \frac{S(r^{(t)})}{H(r^{(t)})}. \tag{2.20}
$$

The iteration process stops if

$$|r^{(t+1)} - r^{(t)}| \leq \epsilon, \tag{2.21}$$

where ϵ is a small positive number, say 10^{-8}.

The derivation of the Newton method is very simple. It uses the Taylor series expansion to approximate the score function. Let $r^{(0)}$ be the initial value of r. The score function $S(r)$ can be approximated in the neighborhood of $r^{(0)}$ by

$$\begin{aligned} S(r) &= S(r^{(0)}) + \frac{d}{dr}S(r^{(0)})(r - r^{(0)}) + \frac{1}{2!}\frac{d^2}{dr^2}S(r^{(0)})(r - r^{(0)})^2 + \dots \\ &\approx S(r^{(0)}) + \frac{d}{dr}S(r^{(0)})(r - r^{(0)}). \end{aligned} \tag{2.22}$$

The approximation is due to ignorance of the higher order terms of the Taylor series. Recall that $H(r^{(0)}) = \frac{d}{dr}S(r^{(0)})$ and thus

$$S(r) \approx S(r^{(0)}) + H(r^{(0)})(r - r^{(0)}). \tag{2.23}$$

Letting $S(r) = 0$ and solving for r, we get

$$r = r^{(0)} - \frac{S(r^{(0)})}{H(r^{(0)})}. \tag{2.24}$$

We have moved from $r^{(0)}$ to r, one step closer to the true solution. Let $r = r^{(t+1)}$ and $r^{(0)} = r^{(t)}$. The Newton's equation of iteration (2.20) is obtained by substituting r and $r^{(0)}$ into equation (2.24).

The Newton method does not behave well when r is close to zero or 0.5 for the reason that $H^{-1}(r)$ can be easily overflowed. The Fisher's scoring method is a modified version of the Newton method for avoiding the overflow problem. As such, the method behaves well in all range of the parameter in the legal domain $0 \leq r \leq \frac{1}{2}$. In the Fisher's scoring method, the second derivative involved in the iteration is simply replaced by the so called expectation of the second derivative. The iteration equation becomes

$$r^{(t+1)} = r^{(t)} - \frac{S(r^{(t)})}{\mathtt{E}[H(r^{(t)})]}, \tag{2.25}$$

where

$$\mathtt{E}[H(r^{(t)})] = -\frac{2n[1 - 3r^{(t)} + 3(r^{(t)})^2]}{r^{(t)}(1 - r^{(t)})[1 - 2r^{(t)} + 2(r^{(t)})^2]}. \tag{2.26}$$

Let $I(r^{(t)}) = -\mathtt{E}[H(r^{(t)})]$ be the Fisher's information. The iteration process can be rewritten as

$$r^{(t+1)} = r^{(t)} + I^{-1}(r^{(t)})S(r^{(t)}). \tag{2.27}$$

Assume that the iteration converges at the $t + 1$ iteration. The MLE of r is $\hat{r} = r^{(t+1)}$. The method provides an automatic way to calculate the variance of the estimate,

$$\mathtt{var}(\hat{r}) \approx I^{-1}(\hat{r}) = \frac{\hat{r}(1-\hat{r})(1-2\hat{r}+2\hat{r}^2)}{2n(1-3\hat{r}+3\hat{r}^2)}, \tag{2.28}$$

where $n = \sum_{i=1}^{3}\sum_{j=1}^{3} m_{ij}$ is the sample size. Note that when $\hat{r} \to 0$, $\hat{r}^2$ becomes negligible and $1 - 2\hat{r} \approx 1 - 3\hat{r}$, leading to $1 - 2\hat{r} + 2\hat{r}^2 \approx 1 - 3\hat{r} + 3\hat{r}^2$. Therefore,

$$\mathtt{var}(\hat{r}) \approx \frac{\hat{r}(1-\hat{r})}{2n}. \tag{2.29}$$

Comparing this variance with the one in the BC design shown in equation (2.11), we can see that the variance has been reduced by half. Therefore, using the $\mathtt{F}_2$ design is more efficient than the BC design.

2.5 EM algorithm for estimating *r*

The EM algorithm was developed by Dempster et al. (1977) for handling missing data problems. The algorithm repeatedly executes an E-step and an M-step for iterations. The E-step stands for expectation and the M-step for maximization. The problem of estimating recombination fraction in $\mathtt{F}_2$ can be formulated as a missing value problem, and thus solved by the EM algorithm. The derivation of the EM algorithm is quite involved and will be introduced later when we deal with a simpler problem. We now only give the final equation of the EM iteration. Recall that the $\mathtt{F}_1$ hybrid can produce four possible gametes, two of them are of parental type (A_1B_1 and A_2B_2) and the other two are recombinants (A_1B_2 and A_2B_1). Therefore, an $\mathtt{F}_2$ progeny can be classified into one of three categories in terms of the number of recombinant gametes contained: 0, 1 or 2. From Table 2.4 we can see that each of the following observed genotypes carries one recombinant gamete: $A_1A_1B_1B_2$, $A_1A_2B_1B_1$, $A_1A_2B_2B_2$ and $A_2A_2B_1B_2$, and each of the following observed genotypes carries two recombinant gametes: $A_1A_1B_2B_2$ and $A_2A_2B_1B_1$. Let $n_1 = m_{12} + m_{21} + m_{23} + m_{32}$ be the number of individuals of category 1 and $n_2 = m_{13} + m_{31}$ be the number of individuals of category 2. The double heterozygote $A_1A_2B_1B_2$ is an ambiguous genotype because it may carry 0 recombinant gamete: $(\frac{A_1B_1}{A_2B_2}, \frac{A_2B_2}{A_1B_1})$ or two recombinant gametes: $(\frac{A_1B_2}{A_2B_1}, \frac{A_2B_1}{A_1B_2})$. The number of double heterozygote individuals that carry two recombinant gametes is $n_{23} + n_{32}$. Unfortunately, this number is not observable. If it were, we would be able to take the ratio of the number of recombinant gametes to the total number of gametes in the $\mathtt{F}_2$ progeny ($2n$) to get the estimated recombination fraction right way,

$$\hat{r} = \frac{1}{2n}[2(n_{23} + n_{32} + n_2) + n_1] \tag{2.30}$$

The EM algorithm takes advantage of this simple expression by substituting the missing values $(n_{23} + n_{32})$ by its expectation. The expectation, however, requires knowledge of the parameter, which is what we want to estimate. Therefore, iterations are required. To calculate the expectation, we need the current value of r, denoted by $r^{(t)}$, and the number of double heterozygote individuals (m_{22}). Recall that the overall proportion of the double heterozygote is $\frac{1}{2}[r^2 + (1-r)^2]$, where $\frac{1}{2}r^2$ represents the proportion of individuals carrying two recombinant gametes and $\frac{1}{2}(1-r)^2$ represents the proportion of individuals carrying no recombinant gametes. The conditional expectation of $n_{23} + n_{32}$ is

$$\mathtt{E}(n_{23} + n_{32}) = \frac{(r^{(t)})^2}{(r^{(t)})^2 + (1 - r^{(t)})^2} m_{22} = w^{(t)} m_{22}. \tag{2.31}$$

The iterative equation may be written as

$$r^{(t+1)} = \frac{1}{2n}\{2[\mathtt{E}(n_{23} + n_{32}) + n_2] + (n_1)\}. \tag{2.32}$$

The final equation of the EM iteration becomes

$$r^{(t+1)} = \frac{1}{2n}[2(w^{(t)} m_{22} + m_{13} + m_{31}) + (m_{12} + m_{21} + m_{23} + m_{32})]. \tag{2.33}$$

Calculating $\mathtt{E}(n_{23} + n_{32})$ using equation (2.31) represents the E-step and updating $r^{(t+1)}$ using equation (2.32) represents the M-step of the EM algorithm. The final result of the EM algorithm is so simple, yet it behaves extremely well with regard to the small number of iterations required for convergence and the insensitiveness to the initial value of r. A drawback of the EM algorithm is the difficulty in calculating the standard error of the estimate. Since the solution is identical to the Fisher's scoring method, the variance (square of the standard error) of the estimate given in equation (2.28) can be used as the variance of the EM estimate.

To test the hypothesis of no linkage, $r = \frac{1}{2}$, we will use the same likelihood ratio test statistic, as described in the BC design. The log likelihood value under the null model, however, needs to be evaluated in a slightly different way, that is $L(\frac{1}{2}) = \sum_{i=1}^{3}\sum_{j=1}^{3} m_{ij}\ln(q_{ij})$, where q_{ij} is a function of $r = \frac{1}{2}$ (see Table 2.5). The log likelihood value under the alternative model is evaluated at $r = \hat{r}$, using $L(\hat{r}) = \sum_{i=1}^{3}\sum_{j=1}^{3} m_{ij}\ln(\hat{q}_{ij})$, where $\hat{q}_{ij}$ is a function of $r = \hat{r}$ (see Table 2.5).

3

Genetic Map Construction

Gene loci are grouped into different chromosomes. Within the same chromosome, the loci are linearly arranged because the chromosome is a string like structure. The distribution of loci among chromosomes and the order of loci within chromosomes are called the genetic map. The data used to construct the genetic map are the genotypes of these loci. From the genotypic data, we can estimate all pairwise recombination fractions. These estimated recombination fractions are used to construct the linkage map. Construction of the genetic map may be better called reconstruction of genetic map because the true genetic map is already present and we simply do not know about it. This is similar to the situation where phylogeny construction is more often called phylogeny reconstruction because we are not constructing the phylogeny of species; rather, we infer the existing phylogeny using observed data. Of course, the inferred map may not be the true one if the sample size is not sufficiently large. Map construction is the first step towards gene mapping (locating functional genes). There are two steps in genetic map construction. The first step is to classify markers into linkage groups according to the pairwise LOD scores or the likelihood ratio test statistics. A convenient rule is that all markers with pairwise LOD scores greater than 3 are classified into the same linkage group. A more efficient grouping rule may be chosen using a combination of LOD score and the recombination fraction. For example, loci A and B may be grouped together if $\texttt{LOD}_{AB} > 3$ and $r_{AB} < 0.45$. Grouping markers into the same linkage group is straightforward and no additional technique is required other than comparing the LOD score of each pair of markers to a predetermined LOD criterion. If we choose a more stringent criterion, some markers may not be assigned into any linkage groups. These markers are called satellite markers. On the other hand, if we choose a less stringent criterion, markers on different chromosomes may be assigned into the same linkage group. The second step of genetic mapping is to find the optimal orders of the markers within the same linkage groups. In this chapter, we only discuss the second step of map construction.

3.1 Criteria of optimality

Given the estimated pair-wise recombination fractions for m loci on the same linkage group, we want to find the optimal order of the loci. There are $m!/2$ possible ways to arrange the m loci. The factorial of m gives the total number of permutations of all m loci. However, the orientation of a linkage map is irrelevant. For instance, ABC and CBA are considered the same order for loci A, B and C, as far as the relative positions of the loci are concerned.

We will first define a criterion of "optimality" and then select the particular order that minimizes or maximizes the criterion. The simplest and also the most commonly used criterion is the sum of adjacent recombination coefficients (sar). The criterion is defined as

$$sar = \sum_{i=1}^{m-1} \hat{r}_{i(i+1)} \tag{3.1}$$

where $\hat{r}_{i(i+1)}$ is the recombination fraction between loci i and $i+1$ for $i = 1, \ldots, m-1$, where i and $i+1$ are two adjacent loci. For m loci, there are $m-1$ adjacent recombination fractions. If there is no estimation error for each of the adjacent recombination fraction, the true sar should have the minimum value compared with any other orders. Consider the following example of three loci with the correct order of ABC. The sar value for this correct order is

$$sar_{ABC} = r_{AB} + r_{BC} \tag{3.2}$$

If we evaluate an alternative order, say ACB, we found that

$$sar_{ACB} = r_{AC} + r_{BC} \tag{3.3}$$

Remember that the true order is ABC so that $r_{AC} = r_{AB} + r_{BC} - 2r_{AB}r_{BC}$ assuming that there is no interference. Substituting r_{AC} into the above equation, we get

$$\begin{aligned} sar_{ACB} =& r_{AB} + r_{BC} - 2r_{AB}r_{BC} + r_{BC} \\ =& r_{AB} + r_{BC} + r_{BC}(1 - 2r_{AB}) \end{aligned} \tag{3.4}$$

Because $r_{BC}(1-2r_{AB}) \geq 0$, we conclude that $sar_{ACB} \geq sar_{ABC}$. In reality, we always use estimated recombination fractions, which are subject to estimation errors, and thus the marker order with minimum sar may not be the true order.

Similar to sar, we may use sad (sum of adjacent distances) as the criterion, which is defined as

$$sad = \sum_{i=1}^{m-1} \hat{x}_{i(i+1)} \tag{3.5}$$

where $\hat{x}_{i(i+1)}$ is the estimated additive distance between loci i and $i+1$ and is converted from the estimated recombination fraction using either the Haldane or Kosambi map function. Similar to the *sar* criterion, the order of loci that minimizes *sad* is the optimal order.

The sum of adjacent likelihoods (*sal*) is another criterion for map construction. Note that the likelihood refers to the log likelihood. In contrast to *sar*, the optimal order should be the one which maximizes *sal*. Define $L(\hat{r}_{i(i+1)})$ as the log likelihood value for the recombination fraction between loci i and $i+1$. The *sal* is defined as

$$sal = \sum_{i=1}^{m-1} L(\hat{r}_{i(i+1)}) \tag{3.6}$$

Both *sad* and *sal* are additive, which is a property required by the branch and bound algorithm for searching the optimal order of markers (see next section).

3.2 Search algorithms

3.2.1 Exhaustive search

Exhaustive search is an algorithm in which all possible orders are evaluated. As a result, it guarantees to find the optimal order. Recall that for m loci, the total number of orders to be evaluated is $n = m!/2$. The number of orders (n) grows quickly as m increases, as shown in the following table.

m	n
2	1
3	3
4	12
5	60
6	360
7	2520
8	20160
9	181440
10	1814400

The algorithm will use up the computing resource quickly as m increases. Therefore, this algorithm is rarely used when $m > 10$. When writing the computer code to evaluate the orders, we want to make sure that all possible orders are evaluated. This can be done using the following approach. Assume that there are five loci, denoted by A, B, C, D and E, that need to be ordered. First, we arbitrarily choose two loci, say A and B, to initiate the map. We then add locus C to the existing map. There are three possible places where we can put C in the existing map: CAB, ACB and ABC. For each of the three orders of the three locus map, we add locus D. For example, to add locus D to the existing order ACB, we need to evaluate the following four possible orders: DACB, ADCB, ACDB and ACBD. We then add locus E (the last locus) to each of the four orders of the four locus map. For example, locus E can be added to the existing order DACB in five different places, leading to EDACB, DEACB, DAECB, DACEB and DACBE. We can see that all the $3 \times 4 \times 5 = 5!/2 = 60$ possible orders have been evaluated so far (see Fig. 3.1). This ends the exhaustive search.

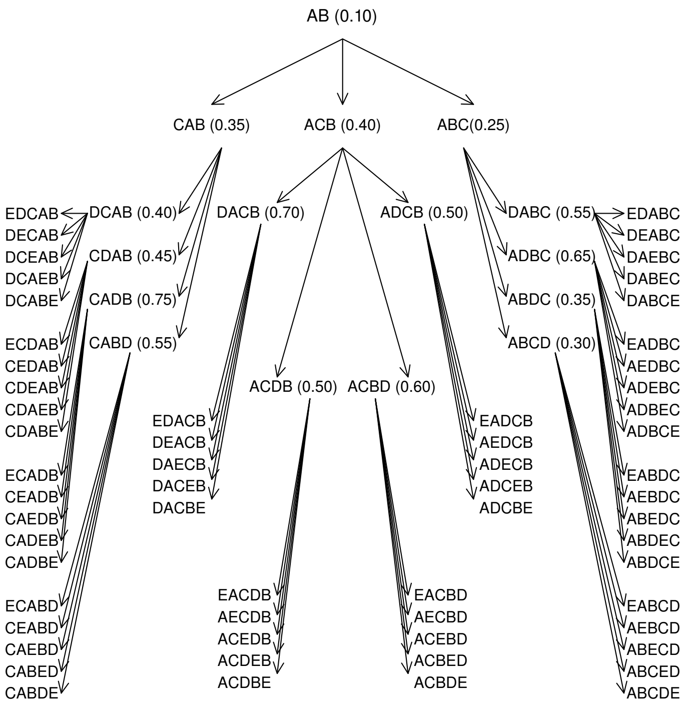

Fig. 3.1. All possible orders (60) of a map with five loci

3.2.2 Heuristic search

When the number of loci is too large to permit the exhaustive search, the optimal order can be sought via a heuristic approach that sacrifices the guarantee of optimality in favor of the reduced computing time. With a heuristic search, one starts with an arbitrary order and evaluates this particular order. The order is then rearranged in a random fashion and re-evaluated. If the new order is "better" than the initial order, the new order is accepted. If the new order is "worse" than the initial order, the initial order is kept. This completes one cycle of the search. The second cycle of the search starts with a rearrangement of the order of the previous cycle. The rearranged order is then accepted or rejected depending on whether or not the value of the order has been improved. The process continues until no further improvement is achieved for a certain number of consecutive cycles, e.g., 50 cycles. This method is also called the greedy method because it is too greedy to “climb up” the hill. It is likely to end up with a local optimum instead of a global one. Therefore, there is no guarantee that the method finds the global optimal order.

The so called random rearrangement may be conducted in several different ways. One is called complete rearrangement or global rearrangement. This is done by selecting a completely different order by random permutation. Information of the previous order has no effect on the selection of the new order. This approach is conceptually simple but may not be efficient. For example, if a previous order is already close to the optimal one, a complete random rearrangement may be far worse than this order, leading to many cycles of random rearrangements before an improved order appears. The other way of rearranging the order is called partial or local rearrangement. This is done by randomly rearranging a subset of the loci. Let $m_s(2 \leq m_s < m)$ be the size of the subset. Although m_s may be chosen in a arbitrary fashion, $m_s = 3$ may be a convenient choice. First, we randomly choose a triplet from the m loci. We then rearrange the three loci within their existing positions and leave the order of the remaining loci intact. There are $3! = 6$ possible ways to rearrange the three loci and all of them are evaluated. The best one of the six is chosen as a candidate new order for re-evaluation. If this new order is better than the order in the previous cycle, the order is updated, otherwise, the previous order is carried over to the next cycle.

The result of heuristic search depends on the initial order selected to start the search. We will use the five locus example to demonstrate a simple way to choose the initial map order. Let A, B, C, D and E be the five loci. We start with two most closely linked loci, say loci A and D. We then add a third locus to the existing two locus map. The third locus is chosen such that it has the minimum average recombination fractions from loci A and D. Assume that locus C satisfies this criterion. We then add locus C to the existing map AD. There are three places where locus C can be added: CAD, ACD and ADC. Choose the best of the three orders as the optimal map, say ACD. We then choose a next locus to add to the existing map ACD, using the same criterion, i.e., minimum average recombination fractions from loci A, C and D. Assume that locus E satisfies this criterion. There are four places that locus E can be inserted into the existing map ACD, which are EACD, AECD, ACED and ACDE. Assume that EACD is the best of the four orders. Finally, we add locus B (the last locus) to the four locus map. We evaluate all the five different orders: BEACD, EBACD, EABCD EACBD and EACDB. Assume that BEACD is the best of the five orders. This order (BEACD) can be used as the initial order to start the heuristic search.

3.2.3 Simulated annealing

Simulated annealing is a method which examines a much larger subset of the possible orders. The method was developed to prevent the solution from being trapped into a local optimum. Like the heuristic search, we start with an arbitrary order and evaluate the order using *sar*, *sad* or $-sal$ as the criterion. Note that *sal* is replaced by $-sal$ because the method always searches for the minimum value of the criterion. The score of the initial order is denoted by E_0. The order is then subject to local rearrangement in a random fashion. This involves Monte Carlo simulation for the order. Note that the rearrangement should be local rather than global. Assume that the score for the new order is E_1. If the new order is "better" than the initial order, i.e., $E_1 < E_0$, the new order is accepted. If the new order is "worse" than the initial order, i.e., $E_1 > E_0$, it is accepted with a probability,

$$\alpha = \exp\left(-\frac{E_1 - E_0}{k_b T}\right) \tag{3.7}$$

where k_b is a physical constant called Boltsman constant and T corresponds to the temperature. Once the new order is accepted, we replace E_0 by E_1 and continue the search for another order. This sampling strategy was proposed by Metropolis et al. (1953), and thus also is referred to as the Metropolis algorithm. By trial and error, it is found that $k_b = 0.95$ usually works well. However, different values should be chosen if $k_b = 0.95$ does not. The value of the temperature T can be chosen arbitrarily, say $T = 2$ or any other values. For m loci, $100m$ new orders should be examined for each value of T, and

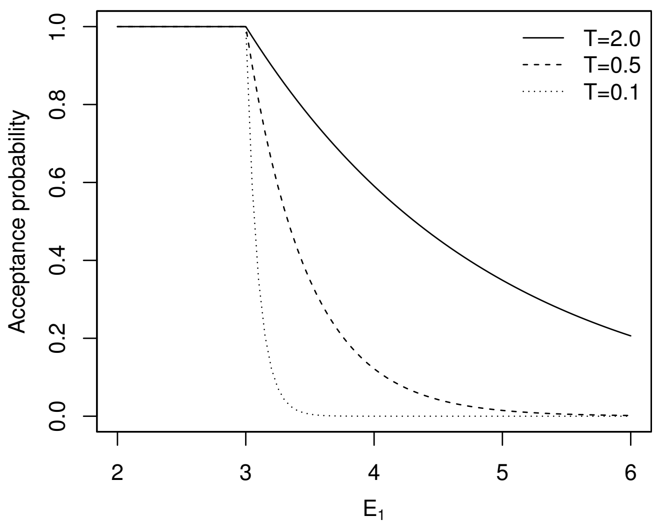

Fig. 3.2. Change of acceptance probability as E_1 deviates from $E_0 = 3.0$

then T should be changed to k_bT (the temperature has been lowered) at this point. With a pseudo code notation, the change of temperature is expressed as $T = k_bT$ when the temperature is decided to change. The algorithm stops after 100 rearrangements have failed to provide a better order. When we write the computer code to simulate the event of accepting or rejecting a new order, we do not care about whether $E_1 < E_0$ or $E_1 > E_0$. We simply let the new order to be accepted with probability α, which is defined as

$$\alpha = \min\left[1, \exp\left(-\frac{E_1 - E_0}{k_bT}\right)\right] \tag{3.8}$$

If $E_1 < E_0$, i.e., the new order is better than the old order, $\alpha = 1$, meaning that the new order is always accepted. If $E_1 > E_0$, then $\alpha < 1$, the probability of accepting the new order is not 100%. When the temperature T gets lower, it makes the acceptance of a worse order harder. This can be shown by looking at the profile of the acceptance probability as a function of the deviation of the new order E_1 from the current order E_0 (see Fig. 3.2). The initial length of the map is $E_0 = 3$. The new length E_1 ranges from 2 to 6. The Boltsman constant is $k_b = 0.95$. The three lines represent three different T values ($T = 2, 0.5, 0.1$). When $E_1 \leq E_0$, the probability of acceptance is 1. After E_1 passes $E_0 = 3$ ($E_1 > E_0$), i.e., the new order is worse than the current order, the acceptance probability starts to decrease, but very slowly for $T = 2$ (high temperature). As the temperature cools down ($T = 0.5$), the acceptance probability decreases more sharply, meaning that it is hard to accept a worse order. When $T = 0.1$, very low temperature, the acceptance probability decreases very sharply, making the acceptance of a worse order extremely difficulty. In the end, only a better order gets accepted and no worse order will be accepted. This will end the search.

The intention of allowing a worse order to be accepted is to prevent the algorithm from settling down at a local optimal order and ignoring a global optimum elsewhere in the space of possible orders. Simulated annealing was set up in a language from the observation that when liquids are cooled very slowly, they will crystallize in a state of minimum energy (Metropolis et al., 1953).

3.2.4 Branch and bound

The branch and bound method is often used in search for evolutionary trees (also called phylogenies). The method was first developed by Land and Doig (1960). It is adopted here to search for the optimal order of loci. This algorithm is not the exhaustive search but it guarantees to find the global optimum order. There will be occasions when it would require examination of all orders; but generally, it requires examination of only a small subset of all possible orders. The criterion of evaluation must be "additive". This property will make sure that the map length for a particular order with k loci cannot be shortened by adding another locus to the existing map of k loci. Both *sad* and $-sal$ follow

Table 3.1. Recombination fractions and additive distances for four marker loci

	A	B	C	D
A		$r_{AB}(0.0906)$	$r_{AC}(0.1967)$	$r_{AD}(0.2256)$
B	$x_{AB}(0.10)$		$r_{BC}(0.1296)$	$r_{BD}(0.1648)$
C	$x_{AC}(0.25)$	$x_{BC}(0.15)$		$r_{CD}(0.0476)$
D	$x_{AD}(0.30)$	$x_{BD}(0.20)$	$x_{CD}(0.05)$	

the additive rule and thus can be used as the length of a map for the branch and bound search. However, *sar* cannot be used here because it is not additive. Let us assume that there are four loci, ABCD, to be ordered. Suppose that the first two loci to be considered are AB. The next locus C can be inserted in one of three positions, corresponding to orders CAB, ACB and ABC, respectively. The fourth locus D can be inserted in four different positions for each of the three-locus orders. There will be $4!/2 = 12$ possible orders after locus D is inserted. In general, we start with two loci (one possible order) and insert the third locus to the two locus map (three possible places to insert the third locus). When the ith locus ($i = 3, \ldots, m$) is inserted into the map of $i-1$ loci, there will be i branch points (places) to insert the ith locus. Overall, there are $3 \times 4 \times \cdots \times m = (1 \times 2 \times 3 \times 4 \times ... \times m)/(1 \times 2) = m!/2$ possible orders. This process is similar to a tree growing process, as illustrated in Figure 3.1 for the example of five loci, except that this tree is drawn up side down with the root at the top. Each tip of the tree represents a particular order of the map for all the m loci, called a child. Each branch point represents an order with $m-1$ or a lower number of loci, called a parent. The initial order of two loci is the root of the tree, called the ancestor. Each member of the tree (including the ancestor, the parents and the children) is associated with an *sad* value, i.e., the length of the member.

If all the possible orders were evaluated, the method would be identical to the exhaustive search. The branch and bound method, however, starts with an arbitrarily chosen order of the m locus map (a child) and assigns the length of this order to E_0, called the upper bound. It is more efficient to select the shortest map order found from a heuristic search as the initial upper bound. Once an upper bound is assigned, it is immediately known that the optimal order cannot have a value greater than E_0. Let T_0 be the map order for the selected child (the length has been chosen as the upper bound). The branch and bound algorithm starts evaluating all the siblings of T_0. The upper bound will be replaced by the shortest length of the siblings in the family if the current T_0 is not the shortest one. Once all the siblings of the current family are evaluated, we backtrack to the parent and evaluate all the siblings of the parent (the uncles of T_0). Remember that all members in the parental generation have $m-1$ loci. Any uncles whose scores are longer that E_0 will be disqualified for further evaluation because they will not produce children with scores shorter than E_0 due to the property that inserting additional loci cannot possibly decrease the score. Therefore, we can dispense with the evaluation of all children that descend from those disqualified uncles in the search and immediately backtrack and proceed down a different path. Only uncles whose scores are shorter than E_0 will be subject to further evaluation. The upper bound will be updated if a shorter member is found in the uncle's families. Once all the uncles and their families are evaluated, we backtrack to the great grandparent and the siblings of the grandparent, and evaluate the families of all the siblings of the grandparent. The process continues until all qualified families have been evaluated. The upper bound E_0 is constantly updated to ensure that it holds the length of the shortest map order among the orders evaluated so far. Constantly updating the upper bound is important, as it may enable other search paths to be terminated more quickly.

The following example is used to demonstrate the branch and bound algorithm. Let A, B, C and D be four loci with unknown order. The recombination fractions are stored in the upper triangular positions of the matrix given in Table 3.1. The additive distances converted from the recombination fractions using the Haldane map function are stored in the lower triangular positions of the matrix (Table 3.1). The 12 possible orders of the loci are given in Table 3.2 along with the *sad* score for each order. For example, the *sad* score for order CABD is

$$sad_{DBAC} = x_{BD} + x_{AB} + x_{AC} = 0.20 + 0.10 + 0.25 = 0.55. \tag{3.9}$$

From Table 3.2, we can see that the optimal order is order 12, i.e., ABCD, because its *sad* score (0.30) is minimum among all other orders. This would be the result of exhaustive search because we had evaluated all the 12 possible orders. Let us pretend that we had not looked at Table 3.2 and we want to proceed with the branch and bound method to search for the optimal order.

The entire tree of four loci is given in Figure 3.3. Each child has four loci and a length given in parentheses. Note that order ACBD has a length 0.60. The five children of ACBD do not belong to this tree of four loci. They are

Table 3.2. The 12 possible orders of a four locus map and their *sad* scores

Order	Map	*sad* score
1	DCAB	0.40
2	CDAB	0.45
3	CADB	0.75
4	CABD	0.55
5	DACB	0.70
6	ADCB	0.50
7	ACDB	0.50
8	ACBD	0.60
9	DABC	0.55
10	ADBC	0.65
11	ABDC	0.35
12	ABCD	0.30

presented here to indicate that the tree of five loci can be expanded from the tree of four loci in this way. A randomly selected order, say DCAB, is used as T_0 whose *sad* score is used as the upper bound, $E_0 = sad_{DCAB} = 0.40$. All the siblings of DCAB are evaluated for the *sad* scores. It turns out that $sad_{DCAB} = 0.40$ is the shortest order in the family and thus E_0 cannot be improved. The search is backtracked to CAB (the parent of DCAB) and the two siblings of the parent are evaluated, with scores of $sad_{ACB} = 0.40$ and $sad_{ABC} = 0.25$. Because $sad_{ACB} = sad_{DCAB} = E_0 = 0.40$, it is concluded that the optimal order cannot occur in the "lineage "descending from ACB because adding one more locus cannot possibly make the *sad* score shorter. Therefore, only children of ABC are qualified for further evaluation. The shortest order occurs in this family and it is ABCD with $sad_{ABCD} = 0.30$. We only evaluate two families out of three (2/3) to find the optimal order by using the branch and bound algorithm.

Let us use the *sad* of a different order as the upper bound to start the search and show that the branch and bound algorithm may end up with evaluating all possible orders. Assume that we choose the score of ACDB as the upper bound,

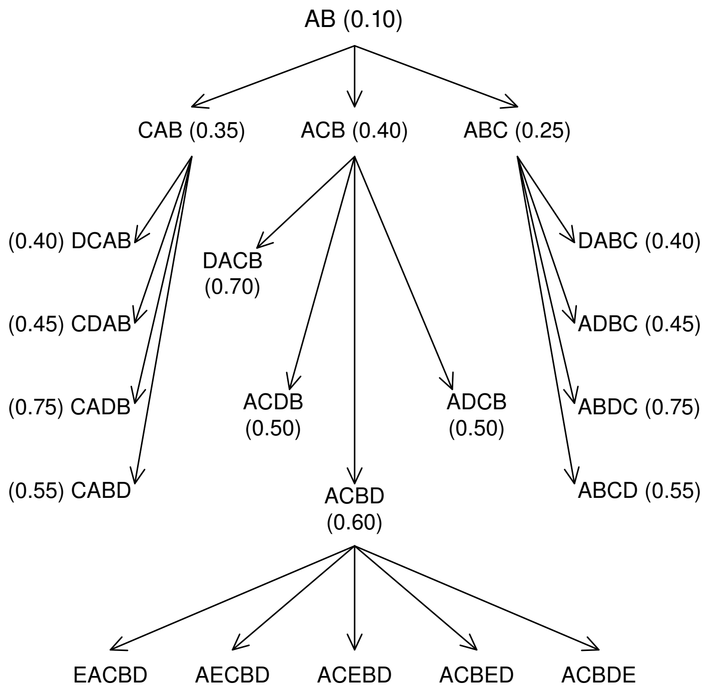

Fig. 3.3. All possible orders (12) of a map with four loci. The five locus children of order ACBD are also given to show that the tree can be expanded this way for more loci.

i.e., $E_0 = sad_{ACDB} = 0.50$. We first evaluated all the siblings of ACDB and found that the upper bound cannot be improved. We then backtracked to the parent of ACDB, which is ACB. The parent has two siblings, CAB and ABC, both are shorter than E_0 and thus both should be further evaluated. However, which of the lineages is evaluated first can make a difference regarding the efficiency of the search. Assume that the lineage under CAB is evaluated first. This leads to an improved upper bound $E_0 = sad_{DCAB} = 0.40$. This upper bound is identical to the length of the parent of the family that we started the search. This family would not have been evaluated if we had chosen $E_0 = 0.40$ as the upper bound in the beginning. Unfortunately, it was too late that we had already evaluated this family. Since ABC is shorter than $E_0 = 0.40$, the lineage under ABC is subject to further evaluation. The shortest order occurs in this lineage, which is ABCD with $sad_{ABCD} = 0.30$. All the 12 possible orders have been evaluated before the optimal order is found. However, if we had evaluated the lineage under ABC first, we would not have to evaluate the lineage under CAB, leading to a 1/3 cut of the computing load.

Finally, if the upper bound is assigned a value from a member in the ABC lineage, say $E_0 = sad_{ABDC} = 0.35$, the upper bound is immediately updated as $E_0 = sad_{ABCD} = 0.30$. This upper bound immediately disqualifies the other two lineages, leading to a 2/3 reduction of the number of orders for evaluation.

The branch and bound algorithm guarantees to find the shortest order, but the efficiency depends on the upper bound chosen and the sequence in which the paths are visited.

3.3 Bootstrap confidence of a map

In phylogeny analysis, an empirical confidence can be put on each internal branch of a particular phylogeny via bootstrap samplings (Felsenstein, 1985). This idea can be adopted here for map construction. A map with m markers have $m-1$ internal segments, similar to the internal branches of a phylogenetic tree. We can put a confidence on each segment. Let us assume that C-D-B-A-E is the optimal order we found. We want to put a bootstrap confidence on segment D-B. First, we draw a large number of bootstrap samples, say N. Each bootstrap sample contains n randomly sampled progeny from the original sample (map population) with replacement. This means that in a bootstrap sample, some progeny may be drawn several times while others may not be drawn at all. For each bootstrap sample, we estimate all the pairwise recombination fractions and construct a map (find the optimal order of the markers for that particular bootstrap sample). In the end, we will have N different maps, one from each bootstrap sample. We then count the number of maps that have segment D-B, i.e., D and B are joined together. The proportion of the maps that reserve this segment is the confidence of this segment. Let $N = 100$ be the number of bootstrap samples and $N_{D-B} = 95$

be the number of samples reserving segment D-B, the confidence for segment D-B is $N_{D-B}/N = 0.95$. Each segment of the map can be put a confidence using the same approach.

The way to place a bootstrap confidence for a segment of a map described here appears to be different from the bootstrap confidence of an internal branch of a phylogenetic tree. We simply adopted the idea of phylogenetic bootstrap analysis, not the way of confidence assignment. To fully adopt the bootstrap confidence assignment, we need to find the number of bootstrap samples that partition the loci into $\{C,D\}$ and $\{B,A,E\}$ subsets and also reserve the D-B segment. That number divided by $N = 100$ would give the confidence for the D-B segment.