# QTL

## Contents

# 1 Introduction

**QTL:** A quantitative trait locus (QTL) is a locus that correlates with variation of a quantitative trait in the phenotype of a population of organisms. QTLs are mapped by identifying which molecular markers (such as SNPs or AFLPs) correlate with an observed trait.

**Locus:** A genomic region that explains phenotypic differences due to genetic polymorphisms.

**Quantitative trait:** Also known as complex traits, are traits that do not behave according to simple Mendelian inheritance laws.

There are some different types of QTL:

- **eQTL:** Expression quantitative trait loci (eQTL) are genomic loci that explain variation in expression levels of mRNAs.
- **mQTL:** Methylation quantitative trait loci (mQTL) are genomic loci that explain variation in methylation levels of DNA.
- **sQTL:** Splicing quantitative trait loci (sQTL) are genomic loci that explain variation in regulation of alternative splicing of pre-mRNA.

# 2 Crosses

**Backcross** is the simplest form of crossing where F1 individuals are crossed to one of the two parental strains. In a backcross to the A strain, one may detect a QTL only if the A allele is not dominant.

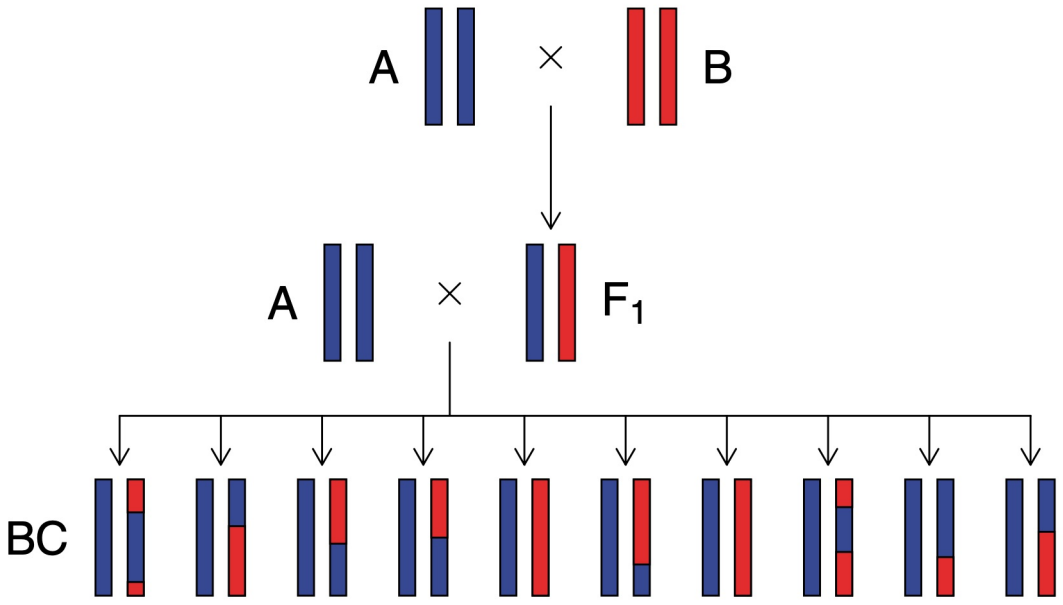



Figure 1: Backcrossing

**Intercross** is the most widely used form of crossing. The intercross allows the detection of QTL for which one allele is dominant. Moreover, the intercross allows one to estimate the degree of dominance at a QTL.

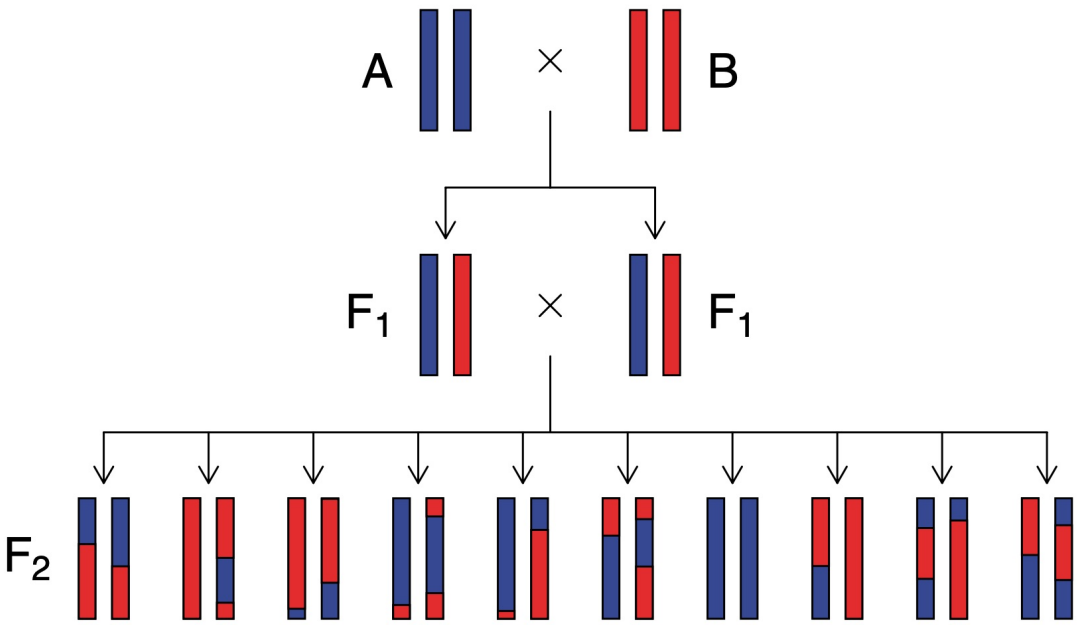



Figure 2: Intercrossing

**Recombinant inbred lines (RIL)** is another form beginning with an intercross, and then mating pairs of F2 siblings, followed by a parallel series of repeated sibling mating to construct a new panel of inbred lines whose genomes are a mosaic of the two initial lines. In RIL we can achieve better mapping resolution, but it is expensive.

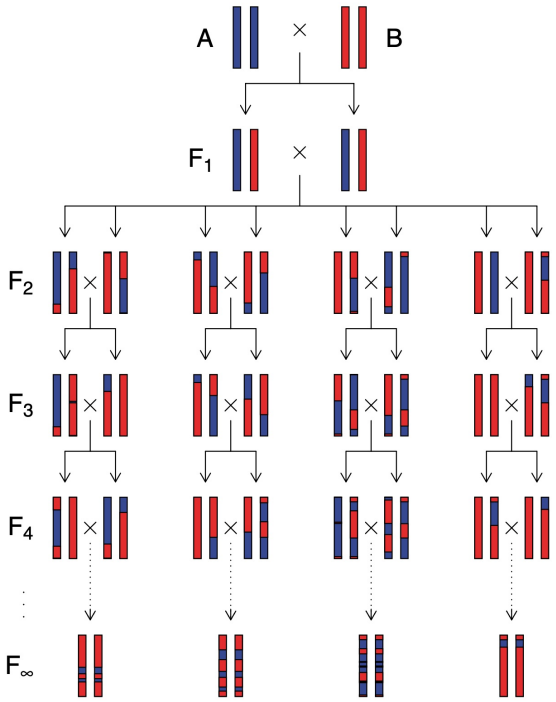



Figure 3: Recombinant inbred lines

# 3 QTL mapping

The key idea in QTL mapping is to obtain phenotype data on a number of backcross or intercross progeny and then identify regions in the genome where genotype is associated with the phenotype. The genotype is observed at genetic markers (SNP, microsatellites, etc.). The principal goals of QTL mapping are:

- First, we seek to detect QTL (and, potentially, interactions among QTL).
- Second, we seek confidence regions for the locations of the QTL.
- Finally, we seek to estimate the effects of the QTL (i.e., the effect, on the phenotype, of substituting one allele for another).

# 4 Import data

Let's import and check *listeria* dataset. The listeria data set is from Boyartchuk et al. (2001). This is an intercross using the C57BL/6ByJ and BALB/cByJ inbred mouse strains. There are 120 female intercross individuals (though only 116 were phenotyped). Mice were injected with Listeria monocytogenes; the phenotype is survival time (in hours). A large proportion of the mice (35/116) survived past the 240-hour time point and were considered to have recovered from the infection; their phenotype was recorded as 264.

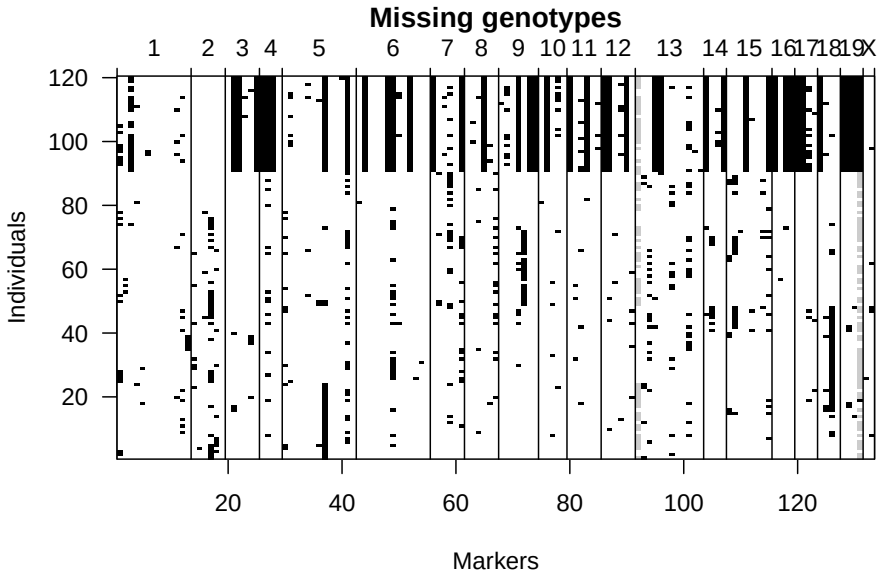
Let's check the data now:

```
# Import data
library(qtl)
data("listeria")

# Check data summary
summary(listeria)
```

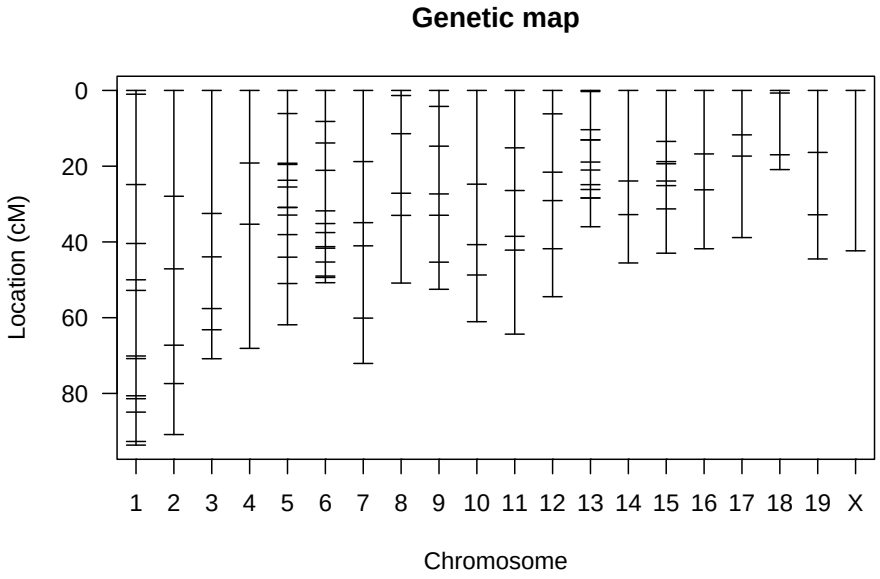
```
##     F2 intercross
##
##     No. individuals:    120
##
##     No. phenotypes:     2
##     Percent phenotyped: 96.7 100
##
##     No. chromosomes:    20
##         Autosomes:      1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
##         X chr:          X
##
##     Total markers:      133
##     No. markers:        13 6 6 4 13 13 6 6 7 5 6 6 12 4 8 4 4 4 4 2
##     Percent genotyped:  88.5
##     Genotypes (%):
##           Autosomes:        CC:25.8      CB:48.9      BB:24.4  not BB:0.0
##                         not CC:0.9
##        X chromosome:        CC:51.7      CB:48.3
```

Let's plot data summaries:

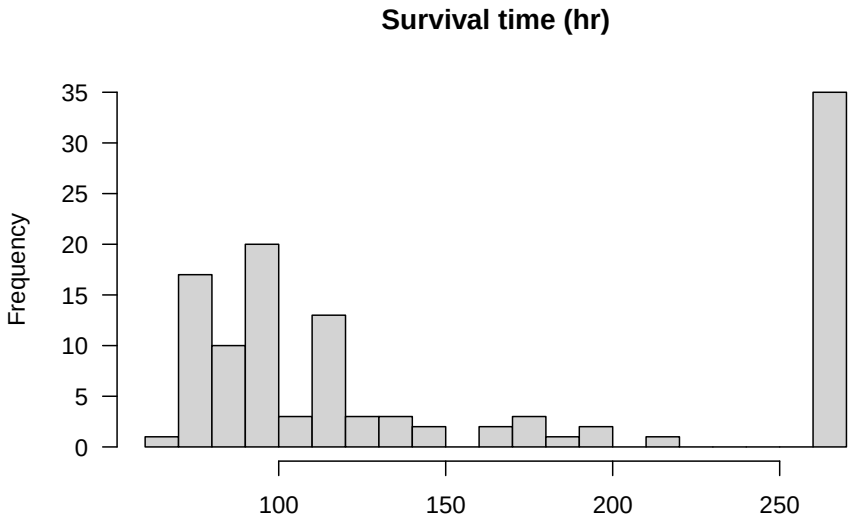
```
# The pattern of missing genotype data
plotMissing(listeria)
```



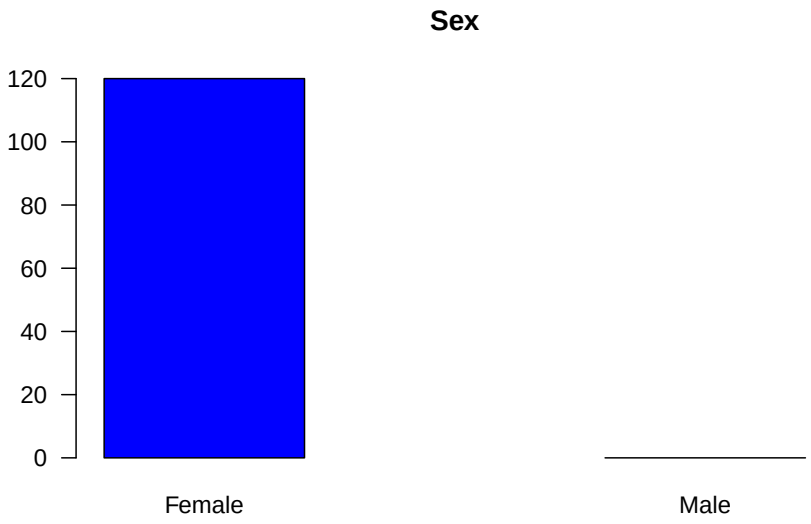

```
# The genetic map of the typed markers
plot.map(listeria)
```




```
# A histogram of the phenotype
plotPheno(listeria, 1, xlab = "", main = "Survival time (hr)")
```




```
# A bar plot of the sexes
plotPheno(
  listeria,
  2,
  xlab = "",
  main = "Sex",
  col = "blue",
  space = 1.5,
  names.arg = c("Female", "Male")
)
```




Other functions for checking data includes:

```
# Number of individuals
nind(listeria)
```

```
## [1] 120
```

```
# Number of phenotypes
nphe(listeria)
```

```
## [1] 2
```

```
# Number of markers
totmar(listeria)
```

```
## [1] 133
```

```
# Number of chromosomes
nchr(listeria)
```

```
## [1] 20
```

```
# Numbers of markers on individual chromosomes
nmar(listeria)
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19  X
## 13  6  6  4 13 13  6  6  7  5  6  6 12  4  8  4  4  4  4  2
```

Now let's check geno and pheno data:

```
# Marker geno data for chr 1
## Backcross codes: 1 for homozygotes and 2 for heterozygotes
## Intercross codes: 1 for AA, 2 for AB, 3 for BB, 4 for not BB, 5 for not AA
listeria$geno[[1]]$data[1:2, 1:4]
```

```
##      D10M44 D1M3 D1M75 D1M215
## [1,]      3    3     3      2
## [2,]     NA    3     3      3
```

```
# On the chr X, all individuals are coded with genotypes 1/2
listeria$geno[[20]]$data[1:2, 1:2]
```

```
##      DXM186 DXM64
## [1,]      2     2
## [2,]      2     2
```

```
# Positions of the markers, in cM, for chr 1
listeria$geno[[1]]$map[1:4]
```

```
##     D10M44       D1M3      D1M75     D1M215
##  0.0000000  0.9967536 24.8477329 40.4136087
```

```
# Chromosome class of chr 19 and chr X
sapply(listeria$geno, class)[19:20]
```

```
##  19   X
## "A" "X"
```

```
# Pheno data
## Rows correspond to individuals and columns correspond to phenotypes
listeria$pheno[1:3, ]
```

```
##      T264    sex
## 1 118.317 female
```

```
## 2 264.000 female
## 3 194.917 female
```

Let's create a matrix containing the pairwise recombination fractions and LOD scores. Values on the diagonal are the number of individuals that were genotyped for the corresponding marker. Values above the diagonal are LOD scores for a test of linkage; values below the diagonal are estimated recombination fractions.

```
listeria <- est.rf(listeria)
listeria$rf[1:3, 1:3]
```

```
##              D10M44        D1M3      D1M75
## D10M44 202.00000000  39.0229439   6.387192
## D1M3     0.01541617 234.0000000   9.034179
## D1M75    0.23261277   0.1977423 192.000000
```

# 5 Quality control

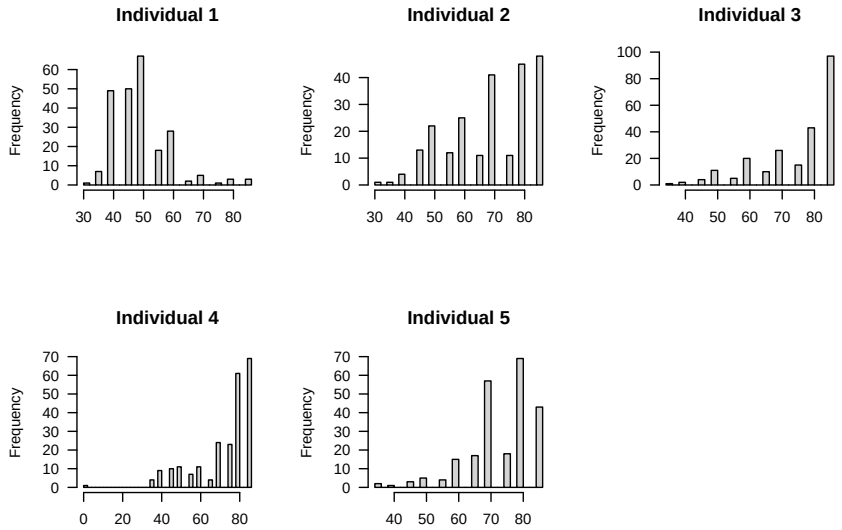
## 5.1 Phenotypes

We begin by considering the example data, ch3a:

```
# Import data
library(qtl)
library(qtlbook)
data(ch3a)
```

These data have five related phenotypes. Let's create histograms of the phenotypes:

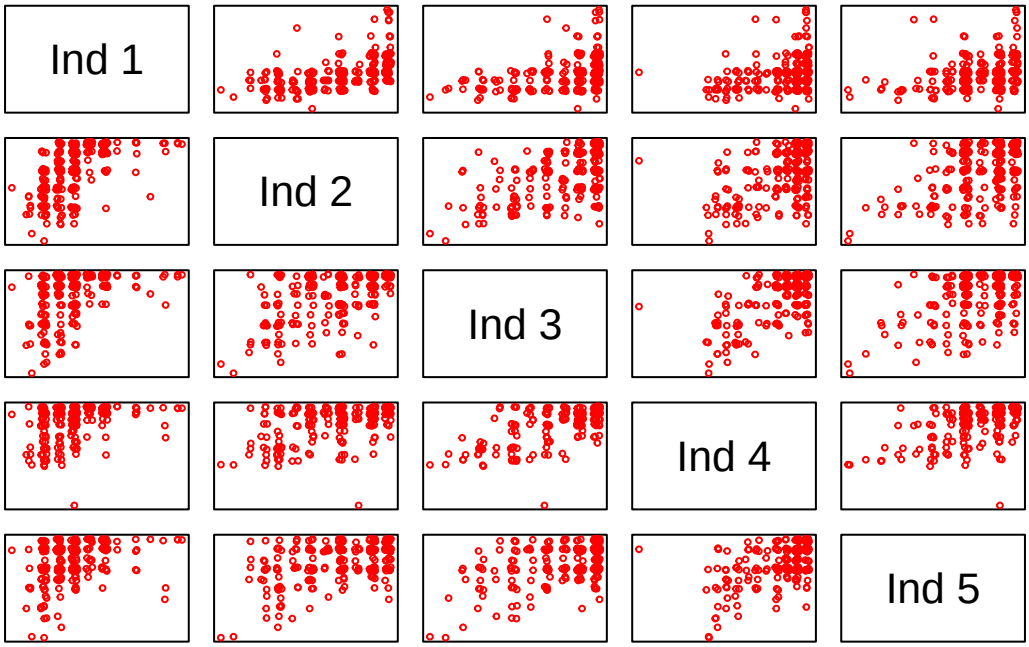
```
par(mfrow = c(2, 3))
for (i in 1:5) {
  plotPheno(ch3a, pheno.col = i, main = paste("Individual", i), xlab = "")
}
```




Now let's create scatterplots of the phenotypes against one another:

```
pairs(
  jitter(as.matrix(ch3a$pheno)),
  cex = 0.6,
```

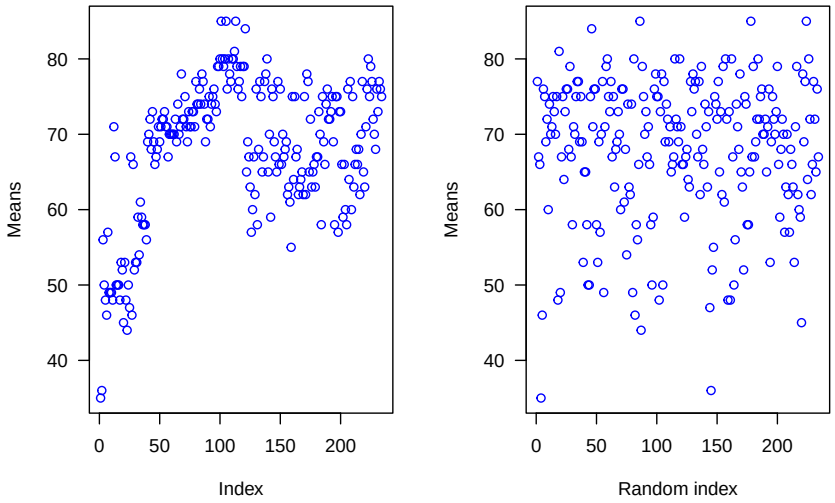
```
  las = 1,
  col = "red",
  labels = c("Ind 1", "Ind 2", "Ind 3", "Ind 4", "Ind 5"),
  xaxt = 'n',
  yaxt = 'n'
)
```




Each panel contains the data for one phenotype plotted against the data for another phenotype. The individual with 0 at the fourth phenotype now stands out.

We know plot the individuals' phenotypes against their index, which may correspond to the order in which they were measured:

```
par(mfrow = c(1, 2),
    las = 1,
    cex = 0.8)
means <- apply(ch3a$pheno, 1, mean)
plot(means, xlab = "Index", ylab = "Means", col = "blue")
plot(sample(means), xlab = "Random index", ylab = "Means", col = "blue")
```




There is a clear pattern in the average phenotype that is not seen in the case that the data have been randomized.

## 5.2 Segregation distortion

The genotypes should appear in the expected proportions. Apparent distortion may indicate genotyping problems. We consider the example data, ch3b:

```
# Import data
library(qtl)
library(qtlbook)
data(ch3b)
```

Let's inspect the genotype frequencies at each marker:

```
gt <- geno.table(ch3b)

# Inspect data for one marker
gt[1, ]
```

```
##      chr missing AA AB BB not.BB not.AA AY BY   P.value
## c1m1   1       9 30 75 30      0      0  0  0 0.4345982
```

The last column in the output is a p-value for a chi-squared test of Mendelian proportions (1:2:1 in an intercross). Now we check for extreme distortions:

```
gt[gt$P.value < 1e-7, ][1:4, 3:7]
```

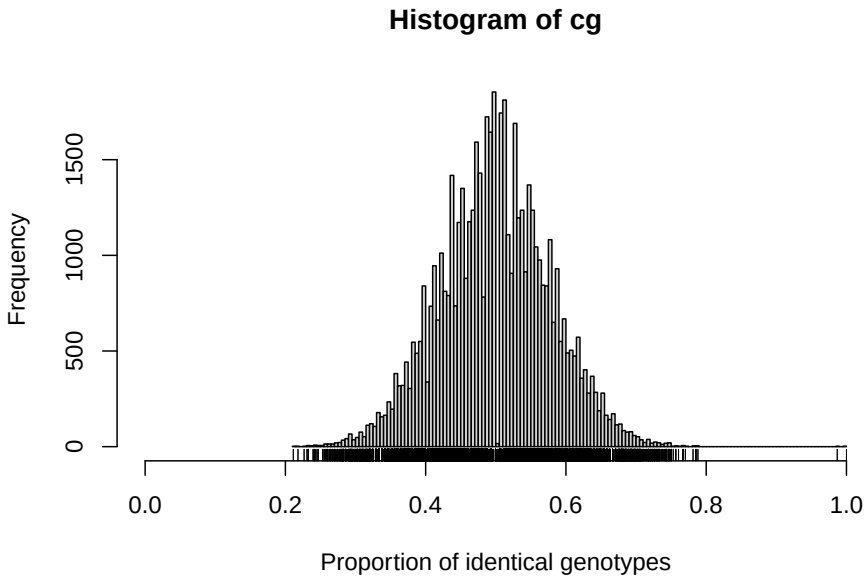
```
##       AA AB  BB not.BB not.AA
## c4m7  31  0 113      0      0
## c6m9  28  0 116      0      0
## c7m3  62  2  40      0      0
## c10m8 36  4  40      0      0
```

It is likely that these problems are due to genotyping errors.

## 5.3 Compare individuals' genotypes

It is occasionally useful to compare the genotype data for each pair of individuals from a cross, to identify pairs that have unusually similar genotypes. These may indicate sample mix-ups:

```
cg <- comparegeno(ch3a)
hist(cg,
     breaks = 200,
     xlab = "Proportion of identical genotypes",
     xlim = c(0, 1))
rug(cg)
```




With the following code, we can identify the pairs of individuals with very similar genotype data:

```
which(cg > 0.9, arr.ind = TRUE)
```

```
##     row col
## 138 138   5
## 55   55  12
## 12   12  55
## 5     5 138
```

Individuals 5 and 138 have identical genotypes at all 86 markers at which they were both typed; individuals 12 and 55 have the same genotype at 75/76 markers. Real backcross individuals shouldn't show such similarity in their genotypes, and so these individuals' data should be viewed with suspicion.

## 5.4 Check marker order

It is critical that one check that markers are placed on the correct chromosomes and in the correct order.

### 5.4.1 Pairwise recombination fractions

The first thing to do is to estimate, for each pair of markers, the recombination fraction between them, "r", and calculate a LOD score for the test of r = 1/2. Markers on different chromosomes should not appear linked, and for markers on the same chromosome, the estimated recombination fraction should be smaller for more closely linked markers:

```
# Import data
library(qtl)
library(qtlbook)
data(ch3c)

# Estimate recombination fractions
ch3c <- est.rf(ch3c)

# Check problematic markers
checkAlleles(ch3c)
```

```
##    marker chr index diff.in.max.LOD
## 2    c1m3   1     2        21.37793
## 3    c1m4   1     3        15.00967
## 32   c7m1   7     1         6.69111
## 33   c7m2   7     2        10.76894
```

There appear to be problems on chromosomes 1 and 7. Let us look in more detail at the genotype data for the markers on chr 1:

```
# Display the map for chr 1
pull.map(ch3c, 1)
```

```
## c1m1 c1m3 c1m4 c1m5
##  8.3 49.0 59.5 89.0
```

```
# Create tables of genotypes at one marker against genotypes at another marker
geno.crosstab(ch3c, "c1m3", "c1m4")
```

```
##     c1m4
## c1m3  - AA AB BB
##   -   7  0  0  0
##   AA  0  0  3 19
##   AB  0  0 38  7
##   BB  0 22  3  1
```

```
geno.crosstab(ch3c, "c1m3", "c1m5")
```

```
##     c1m5
## c1m3  - AA AB BB
##   -   7  0  0  0
##   AA  0  2 11  9
##   AB  0  9 24 12
##   BB  0 12 12  2
```

```
geno.crosstab(ch3c, "c1m4", "c1m5")
```

```
##     c1m5
## c1m4  - AA AB BB
##   -   7  0  0  0
##   AA  0 11 10  1
##   AB  0 11 28  5
##   BB  0  1  9 17
```

It looks like marker "c1m3" is the problem: for that marker, relative to markers "c1m4" and "c1m5", the double-recombinant classes are more common than the nonrecombinant ones, while the table of two-locus genotypes for markers "c1m4" and "c1m5" looks okay.

To fix the problem, we pull out the genotypes for chr 1, swap the alleles (replacing 1's with 3's and vice versa), and then put the new data back:

```
g <- pull.geno(ch3c, 1)
g[, "c1m3"] <- 4 - g[, "c1m3"]
ch3c$geno[[1]]$data <- g
```

By a similar approach, we find that it is "c7m2" on chr 7 that is the problem. We fix it as follows:

```
g <- pull.geno(ch3c, chr = 7)
g[, "c7m2"] <- 4 - g[, "c7m2"]
ch3c$geno[[7]]$data <- g
```

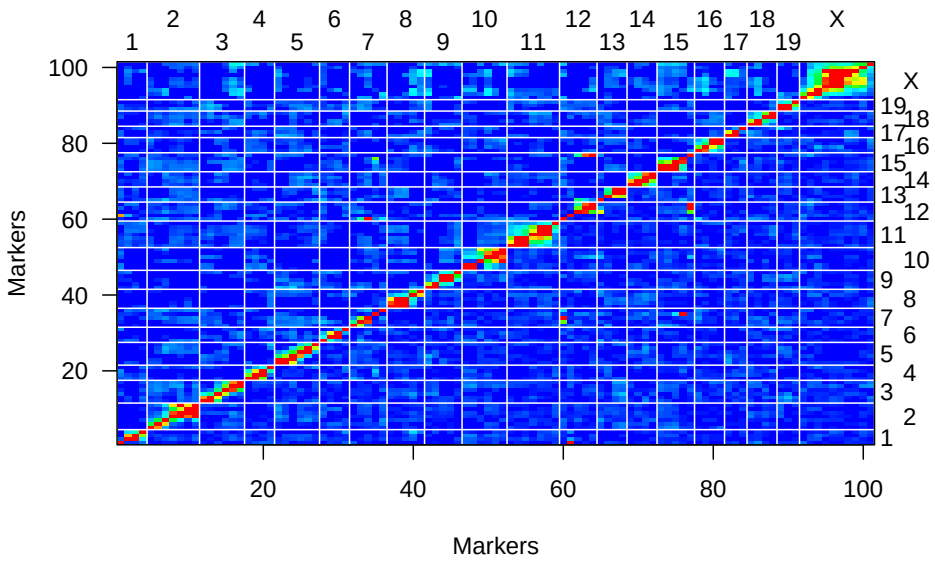
Now, let's recheck:

```
ch3c <- est.rf(ch3c)
checkAlleles(ch3c)
```

```
## No apparent problems.
```

Now plot the pairwise recombination fractions and LOD scores:

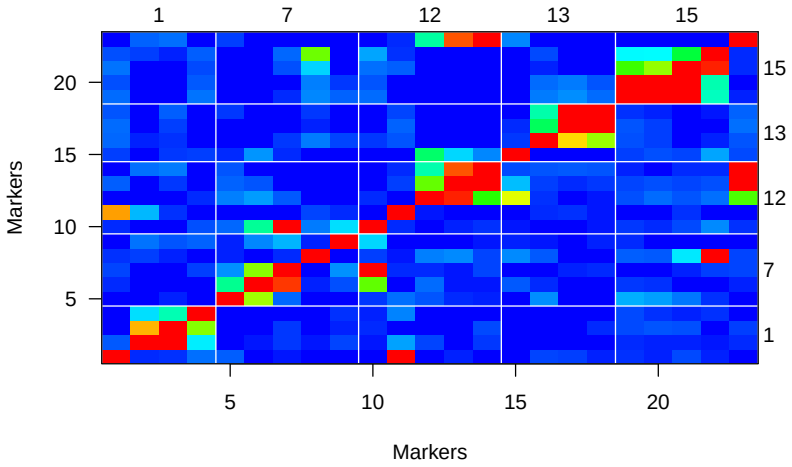
```
plotRF(
  ch3c,
  alternate.chrid = TRUE,
  main = "",
  col.scheme = "redblue"
)
```




The estimated recombination fractions between markers are in the upper left, and the LOD scores are in the lower right. Red indicates pairs of markers that appear to be linked, and blue indicates pairs that are not linked. There are a number of red points in the lower right, indicating markers on different chromosomes that appear linked. In particular, there appear to be problems on chromosomes 1, 7, 12, 13, and 15. We plot the results for just those chromosomes:

```
plotRF(
  ch3c,
```

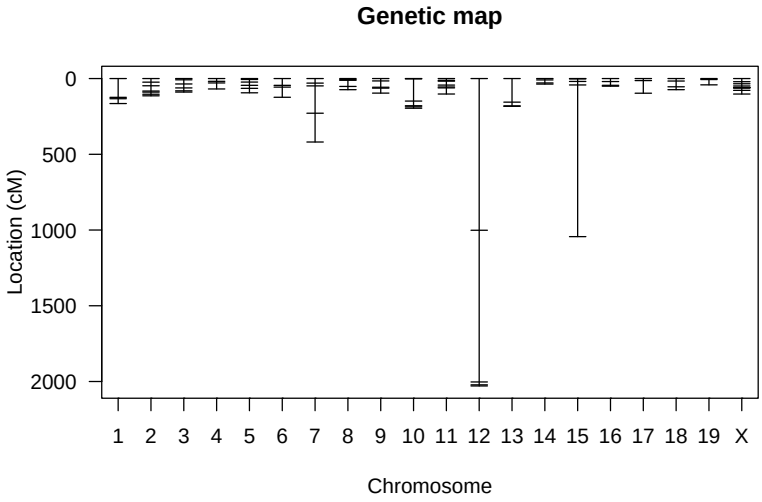
```
  chr = c(1, 7, 12, 13, 15),
  main = "",
  col.scheme = "redblue"
)
```




The results indicate that the 4th marker on chr 7 belongs on chr 15, the 1st marker on chr 12 belongs on chr 7, the 2nd marker on chr 12 belongs on chr 1, the 1st marker on chr 13 belongs on chr 12, and the 5th marker on chr 15 belongs on chr 12.

It is also valuable to use the available genotype data to re-estimate the intermarker distances of the genetic map:

```
# We assume a 0.1% genotyping error rate
nm <- est.map(ch3c, error.prob = 0.001)
plot(nm)
```



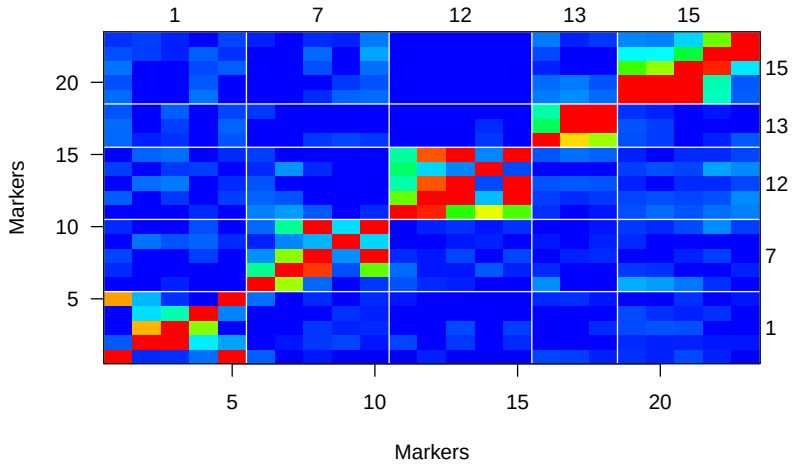

The estimated map indicates clear problems on chr 7, 12 and 15: enormous map expansion occurs as a result of markers that do not belong on those chromosomes.

Let's move these markers to the positions that they appear to be linked:

```
ch3c <- movemarker(ch3c, find.marker(ch3c, 7, index = 4), 15)
ch3c <- movemarker(ch3c, find.marker(ch3c, 12, index = 2), 1)
ch3c <- movemarker(ch3c, find.marker(ch3c, 12, index = 1), 7)
ch3c <- movemarker(ch3c, find.marker(ch3c, 13, index = 1), 12)
ch3c <- movemarker(ch3c, find.marker(ch3c, 15, index = 5), 12)
```

Let's re-check now:

```
plotRF(
  ch3c,
  chr = c(1, 7, 12, 13, 15),
  main = "",
  col.scheme = "redblue"
)
```




The markers now appear to be on the correct chromosomes, though there remain some problems with the order of markers within the chromosomes.

### 5.4.2 Rippling marker order

Now we check the order of markers within a chromosome. As the number of all possible orderings of markers is huge, we consider a sliding window of markers and consider all possible orders of the markers within the window:

```
rip <- ripple(
  ch3c,
  chr = 1,
  window = 5,
  method = "countxo",
  verbose = FALSE
)
summary(rip)
```

```
##                    obligXO
## Initial  1 2 3 4 5     197
## 1        1 5 2 3 4     124
```

```
## 2        4 3 2 1 5     134
## 3        2 3 4 1 5     146
## 4        1 5 4 3 2     146
## 5        1 5 3 2 4     152
## ... [ 16  additional rows] ...
```

The first row is the original marker order. Other marker orders are sorted by the number of obligate crossovers. We can adopt the second order (with the minimal number of obligate crossovers):

```
ch3c <- switch.order(ch3c, chr = 1, order = rip[2,])
```

We will recheck again but with a smaller window size, to see if the likelihood approach is inconsistent with the approximate method. We assume a genotyping error rate of 0.001:

```
rip <- ripple(
  ch3c,
  chr = 1,
  window = 3,
  method = "likelihood",
  error.prob = 0.001,
  verbose = FALSE
  )
summary(rip)
```

```
##                        LOD chrlen
## Initial  1 2 3 4 5     0.0  117.3
## 1        2 1 3 4 5    -1.2  173.8
```

The LOD score (log10 likelihood ratio) compares the original order to the alternative order: a negative value (as here) indicates that the original order has higher likelihood. The last column gives the estimated genetic length of the chromosome with the different marker orders; the best marker order is generally that giving the shortest chromosome length. We see that no further change is needed.

We would now use the same approach with all other chromosomes:

```
# Create a list for the output
rip <- vector("list", nchr(ch3c))

# Assign it the names of the chromosomes
names(rip) <- names(ch3c$geno)

# Check the order of markers
for (i in names(ch3c$geno)) {
  rip[[i]] <- ripple(
    ch3c,
    chr = i,
    window = 7,
    method = "countxo",
    verbose = FALSE
    )
}

# Extract for each chr, the difference in the number of obligate crossovers
# between the initial order and the best of the other orders
dif.nxo <- sapply(rip, function(a) {
  a[1, ncol(a)] - a[2, ncol(a)]
})
```

```
# Switch the order of markers
for(i in names(ch3c$geno)) {
  if (dif.nxo[i] > 0) {
    ch3c <- switch.order(ch3c, i, rip[[i]][2, ])
  }
}

# Repeat the process to see if any further improvement may be found
for(i in names(ch3c$geno)) {
  rip[[i]] <- ripple(
    ch3c,
    chr = i,
    window = 7,
    method = "countxo",
    verbose = FALSE
    )
}

dif.nxo <- sapply(rip, function(a) {
  a[1, ncol(a)] - a[2, ncol(a)]
})
any(dif.nxo > 0)
```

```
## [1] FALSE
```

Finally, we go back through all of the chromosomes with ripple, this time using likelihood method and a window size of 3 markers:

```
# Create a list for the output
rip <- vector("list", nchr(ch3c))

# Assign it the names of the chromosomes
names(rip) <- names(ch3c$geno)

# Check the order of markers
for(i in names(ch3c$geno)){
  rip[[i]] <- ripple(
  ch3c,
  chr = i,
  window = 3,
  method = "likelihood",
  error.prob = 0.001,
  verbose = FALSE
  )
}

lod <- sapply(rip, function(a) {
  a[2, ncol(a) - 1]
})
lod[lod > 0]
```

```
##       X
## 1.65515
```

The X chromosome shows some improvement, and so we look at those results more closely:

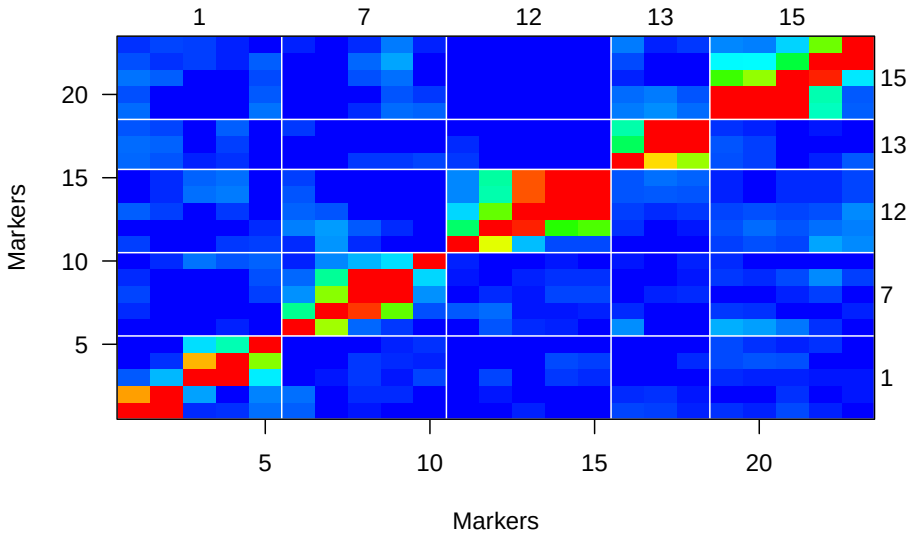
```
summary(rip[["X"]])
```

```
##                                  LOD chrlen
## Initial  1 2 3 4 5 6 7 8 9 10   0.0   96.9
## 1        1 2 3 5 4 6 7 8 9 10   1.7  102.2
## 2        1 2 3 5 6 4 7 8 9 10   1.7  102.2
## 3        1 2 3 4 5 7 6 8 9 10   0.0   96.9
```

The second order increases the likelihood by a factor of 10 ^ 1.7 = 50, but leads to a longer chromosome. As a LOD score of 1.7 is not exceptionally strong, we do not need to switch the orders.

Finally, we may take another look at the pairwise recombination fractions, at least for chromosomes that were shown in the previous section:

```
ch3c <- est.rf(ch3c)
plotRF(
  ch3c,
  chr = c(1, 7, 12, 13, 15),
  main = "",
  col.scheme = "redblue"
)
```

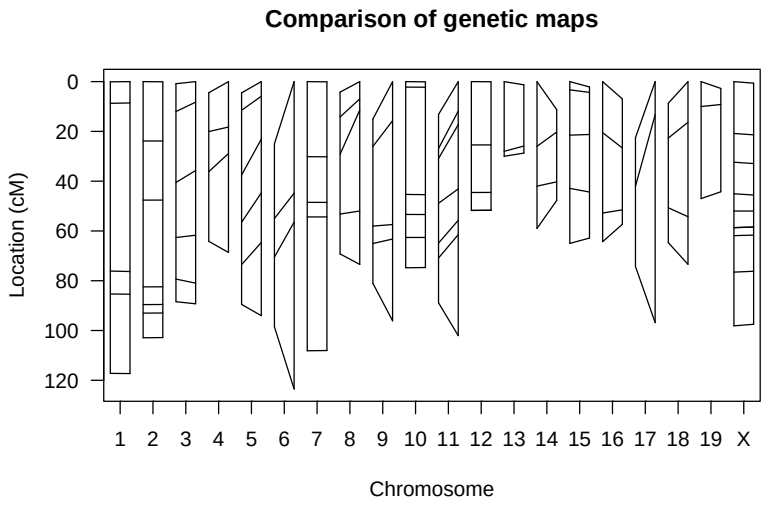



The results are just what we want: red along the diagonal, fading to blue off the diagonal.

### 5.4.3 Estimate genetic map

Now we estimate the intermarker distances from the observed data and compare the results to the map that was included with the data:

```
nm <- est.map(ch3c, error.prob = 0.001, verbose = FALSE)
plot.map(ch3c, nm)
```

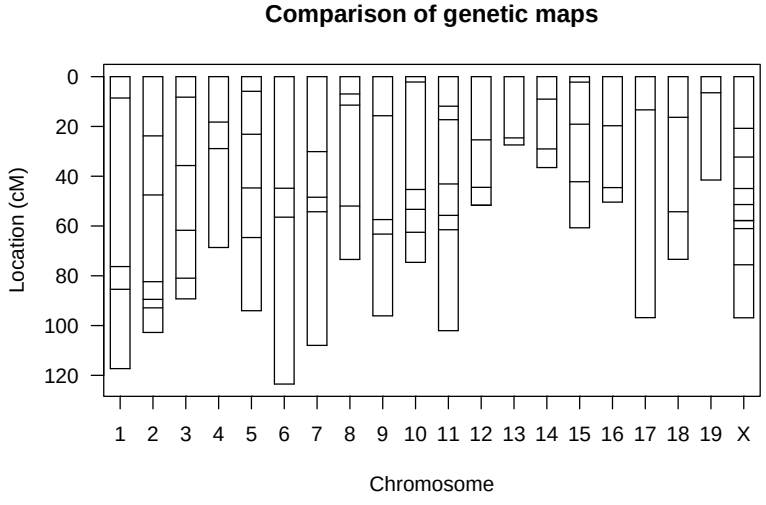



For many chromosomes, the estimated map is identical to the one within the dataset. Several chromosomes exhibit considerable map expansion (e.g., chromosome 6): the estimated map is quite a bit longer than the map in the data. This may indicate the presence of genotyping errors.

One may wish, at this point, to replace the map within the dataset with that estimated from the data. Reference genetic maps are often based on a rather small number of individuals. One's own data often contains many more individuals, and so may produce a more accurate map. The only caveat is that reference genetic maps generally contain a much more dense set of markers, which provides greater ability to detect genotyping errors. Thus reference genetic maps may be based on cleaner genotype data.

Let's replace the genetic map in the dataset with that estimated from the data and recheck the plot:

```
ch3c <- replace.map(ch3c, nm)
plot.map(ch3c, nm)
```




## 5.5 Identifying genotyping errors

Genotyping errors may appear as apparent tight double crossovers. Meiosis generally exhibits strong crossover interference, and so crossovers will not occur too close together. Thus, if the genotype at a single marker is

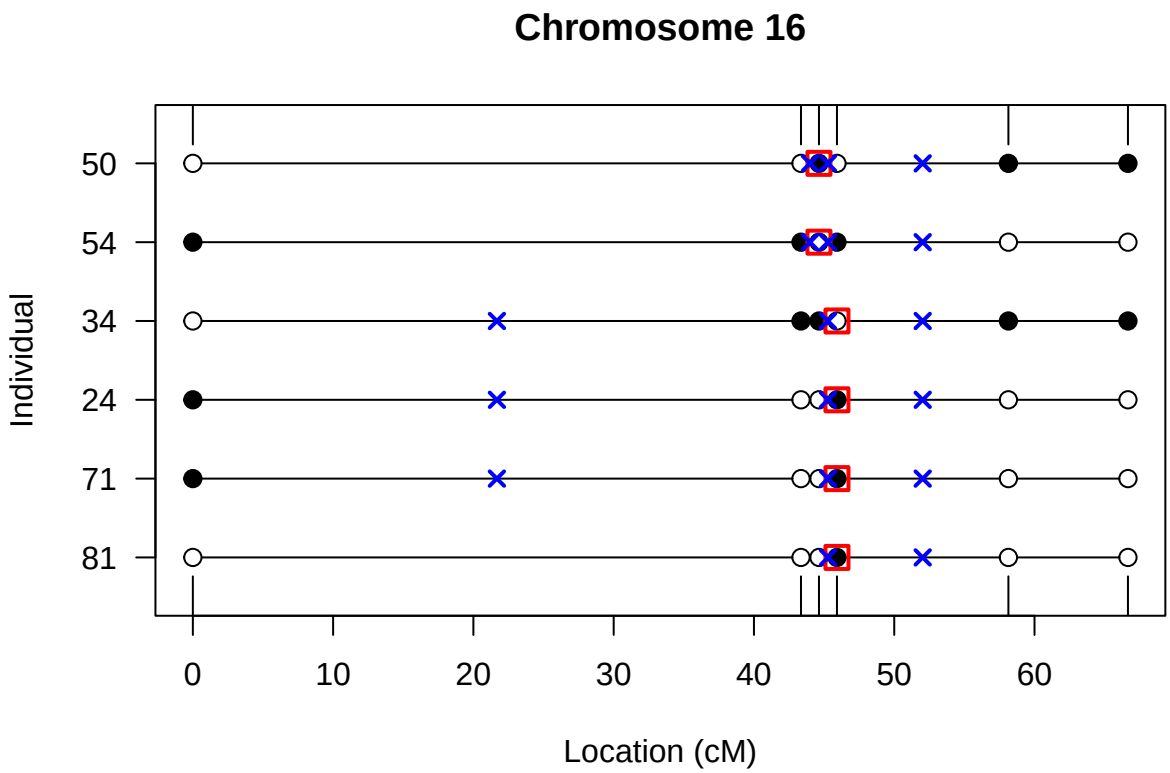
out of phase with the surrounding markers, it is likely in error. Let's calculate the genotyping error LOD scores, but first, we may first wish to replace the map in the data with that estimated from the data:

```
data(hyper)
newmap <- est.map(hyper, error.prob = 0.01)
hyper <- replace.map(hyper, newmap)
hyper <- calc.errorlod(hyper)
top <- top.errorlod(hyper, cutoff = 5)
top
```

```
##   chr id    marker  errorlod
## 1  16 50 D16Mit171 17.996266
## 2  16 54 D16Mit171 17.996266
## 3  16 34   D16Mit5  9.914991
## 4  16 24   D16Mit5  9.914991
## 5  16 71   D16Mit5  9.914991
## 6  16 81   D16Mit5  9.914991
## 7  13 42  D13Mit78  8.999670
## 8  13 42 D13Mit148  8.880574
```

Let's plot the results for chr 16:

```
plotGeno(hyper, chr = 16, top$id[top$chr == 16], cutoff = 5)
```

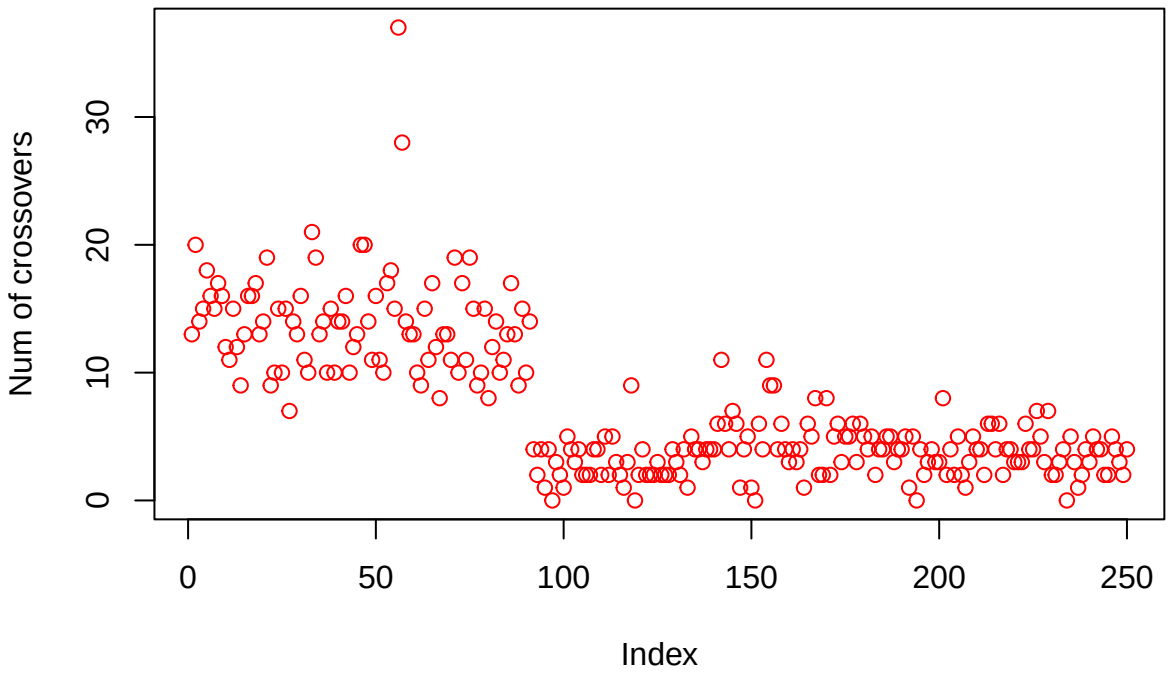



A small number of genotyping errors will not have much influence on the results.

## 5.6 Counting crossovers

Another useful diagnostic is to count the number of crossovers implied by the genotype data in each individual. Individuals with an unusually small or large number of crossovers should be viewed with suspicion:

```
nxo <- countXO(hyper)
plot(nxo, ylab = "Num of crossovers", col = "red")
```




We see a large shift in the distribution between the first 92 individuals and the remaining 158 individuals. Particularly interesting are the two individuals with $> 25$ crossovers:

```
nxo[nxo > 25]
```

```
## 56 57
## 37 28
```

The 56th individual exhibited 37 crossovers. If we pull out the crossover counts for each chromosome for individual 56, we can identify the chromosomes that are particularly problematic:

```
countXO(hyper, bychr = TRUE)[56, ]
```

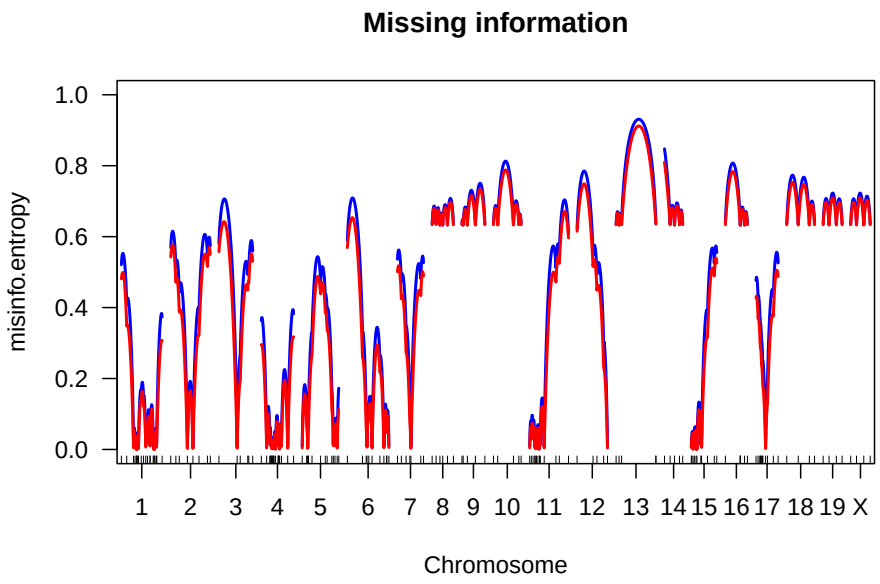
```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19  X
##  2  0  1  1  5  7  2  2  1  2  4  1  0  1  4  1  1  1  1  0
```

The genotype data for chromosome 6 are particularly suspicious, and deserve further investigation.

## 5.7 Missing genotype information

Now we compute the proportion of missing genotype information at positions along the genome, given the available marker data. This can help us to identify regions where further markers might be added:

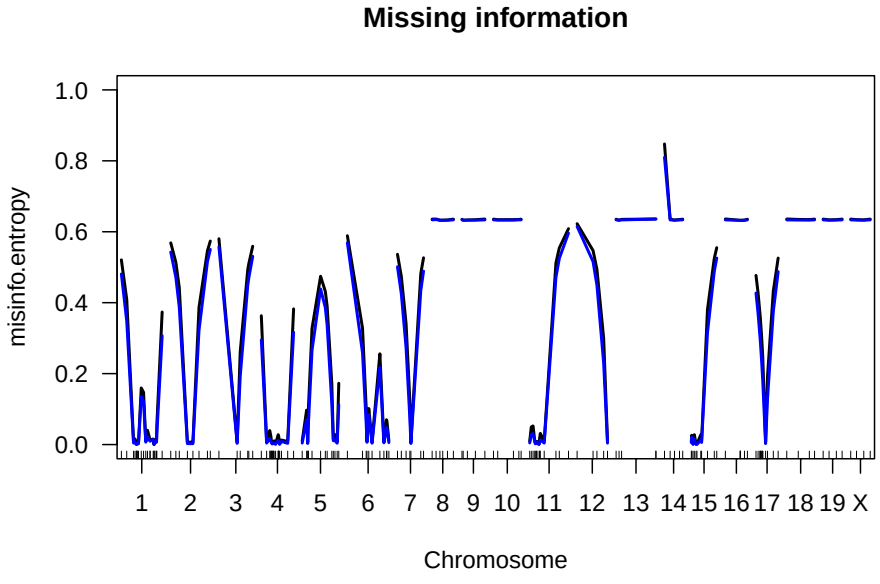
```
plotInfo(hyper, method = "both", col = c("blue", "red"))
```




The entropy and variance versions of the results are plotted in blue and red, respectively. The proportion of missing genotype information is effectively 0 at the fully typed markers. For several chromosomes, the minimal missing information is about 63%, as only the 92 individuals (out of 250) with extreme phenotypes were genotyped.

We can get the results just at the markers as follows:

```
z <- plotInfo(hyper, method = "both", step = 0)
```

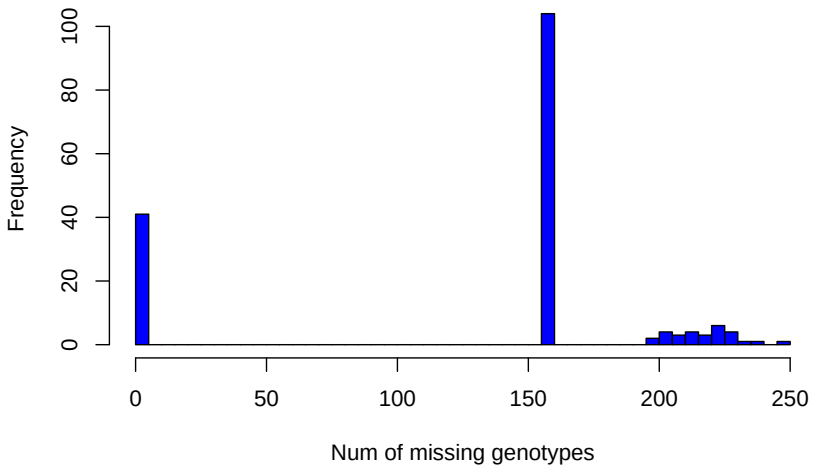



```
z[z[, 1] == 14, ]
```

```
##           chr      pos misinfo.entropy misinfo.variance
## D14Mit48   14  0.00000       0.8475491        0.8098361
## D14Mit14   14 16.40000       0.6355141        0.6335411
## D14Mit37   14 29.05081       0.6331401        0.6323897
## D14Mit7    14 43.67865       0.6343898        0.6333494
## D14Mit266  14 52.97287       0.6355100        0.6336334
```

We can get a histogram of the number of missing genotypes at the markers:

```
hist(
  nmissing(hyper, what = "mar"),
  breaks = 50,
  main = "",
  xlab = "Num of missing genotypes",
   col = "blue"
)
```




About 40 markers were typed on essentially everyone; over 100 were typed on only the 92 individuals with extreme phenotypes. The remaining 29 mark- ers were typed only on a few individuals.

# 6 Single-QTL analysis

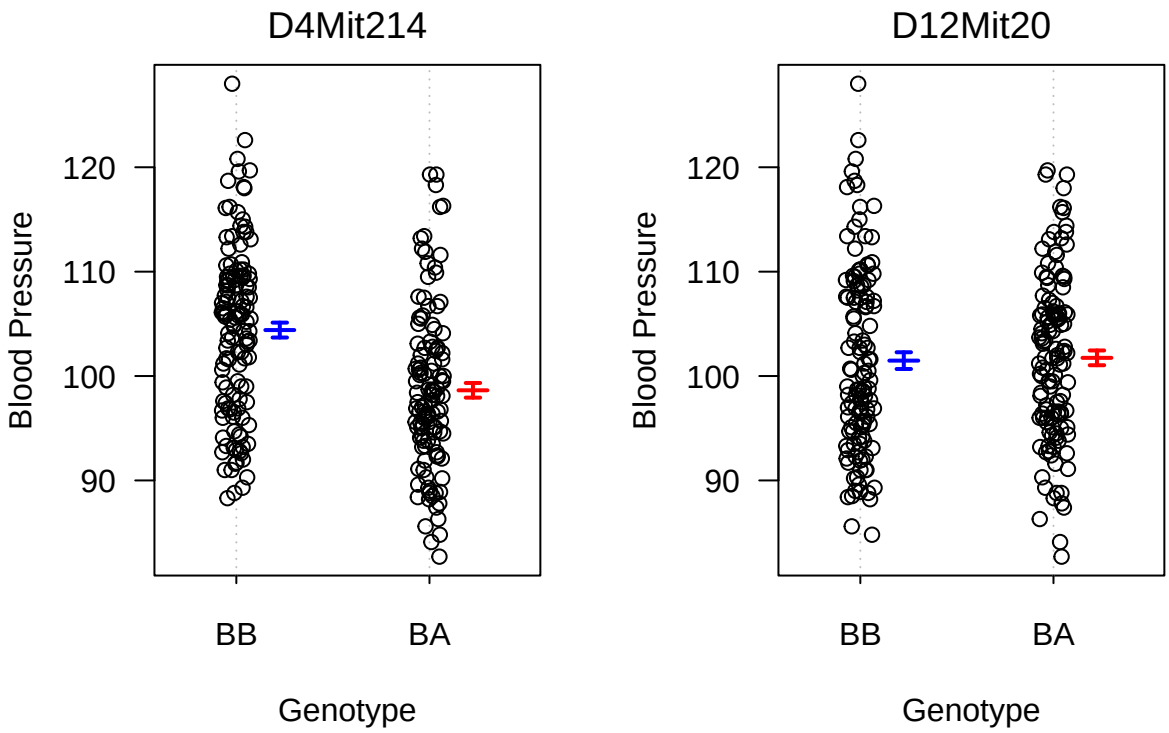
The most commonly used method for QTL analysis is interval mapping, in which one posits the presence of a single QTL and considers each point on a dense grid across the genome, one at a time, as the location of the putative QTL. But first, we describe an even simpler method, sometimes called marker regression.

## 6.1 Marker regression

It consists of considering each marker individually, splitting the individuals into groups, according to their genotypes at the marker, and comparing the groups' phenotype averages. Consider, for example, the *hyper* data. The blood pressure phenotype is plotted against the genotype at markers "D4Mit214" and "D12Mit20":

```
library(qtl)
data(hyper)
par(mfrow = c(1, 2))
```

```
plotPXG(hyper, "D4Mit214", ylab = "Blood Pressure")
plotPXG(hyper, "D12Mit20", ylab = "Blood Pressure")
```




At "D4Mit214", the homozygous individuals exhibit a larger average phenotype than the heterozygotes, indicating that this marker is linked to a QTL.

In a backcross, we test for linkage of a marker to a QTL by a t test; in an intercross, we would use analysis of variance (ANOVA), which gives an F statistic. Traditionally, evidence for linkage to a QTL is measured by a LOD score: the log10 likelihood ratio comparing the hypothesis that there is a QTL at the marker to the hypothesis that there is no QTL anywhere in the genome:

```
# Fit of single-QTL models
out.mr <- scanone(hyper, method = "mr")

# Take a peak at the results
out.mr[out.mr$chr == 12, ]
```

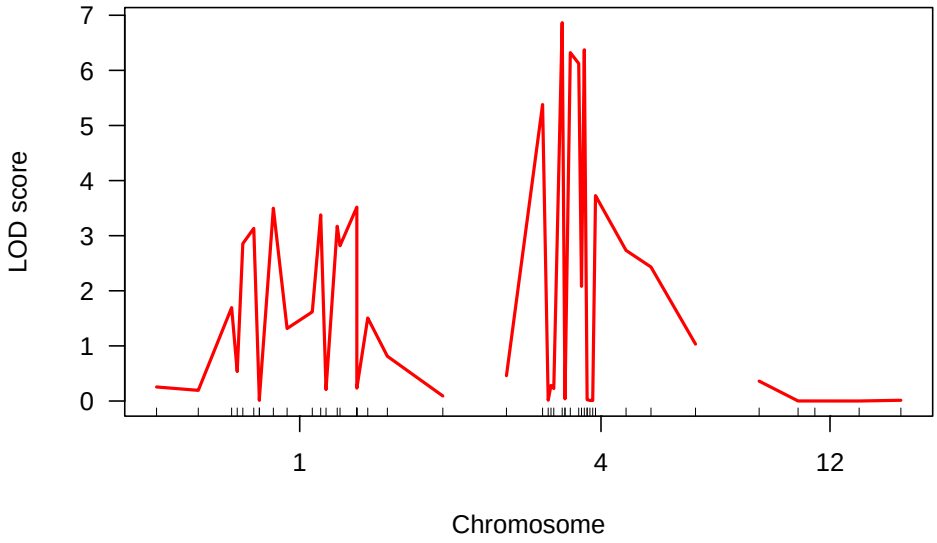
```
##           chr  pos          lod
## D12Mit37   12  1.1 0.3610905438
## D12Mit110  12 16.4 0.0009558728
## D12Mit34   12 23.0 0.0005334957
## D12Mit118  12 40.4 0.0003868014
## D12Mit20   12 56.8 0.0136116027
```

```
# Show just those with a LOD > 3
summary(out.mr, threshold = 3)
```

```
##          chr  pos  lod
```

```
## D1Mit14    1 82.0 3.52
## D4Mit214   4 21.9 6.86
```

```
# Plot of the LOD scores at chr 1, 4, and 12
plot(out.mr, chr = c(1, 4, 12), ylab = "LOD score", col = "red")
```




The jagged appearance of the LOD curve for chromosome 4 is due to the pattern of missing marker genotype data.

## 6.2 Interval mapping

Interval mapping improves on the marker regression method by taking account of missing genotype data at a putative QTL. In this section, we consider several variants on interval mapping.

### 6.2.1 Standard interval mapping

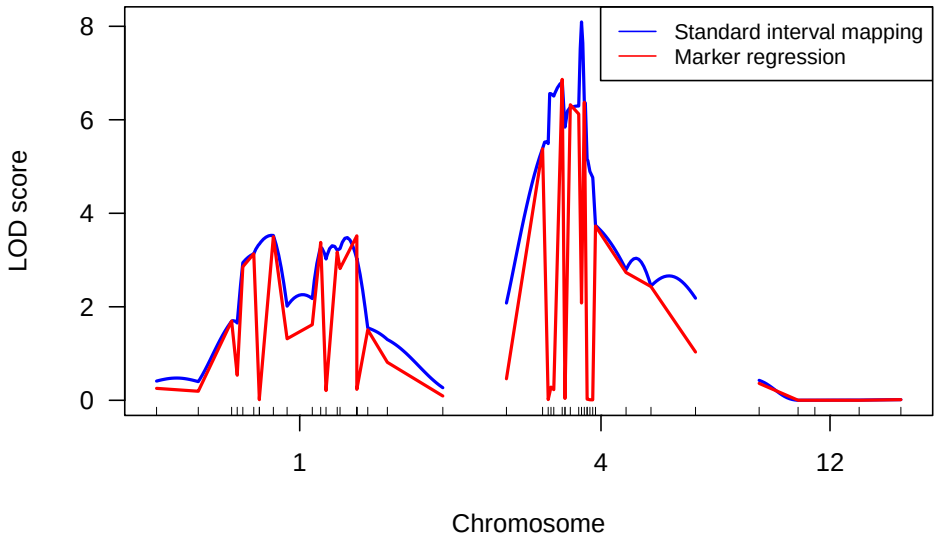
Standard interval mapping uses maximum likelihood estimation under a mixture model. The most important disadvantage of interval mapping is that we are still considering only a single-QTL model, and so we have limited ability to separate linked QTL and no ability to assess possible interactions among QTL. Note that no two markers should be placed at precisely the same position. Marker positions may be moved apart slightly as follows:

```
hyper <- jittermap(hyper)
```

Now to perform standard interval mapping:

```
# Calculate the conditional genotype probabilities
hyper <- calc.genoprob(hyper, step = 1, error.prob = 0.001)

# Perform interval mapping
out.em <- scanone(hyper, method = "em")

# Plot the results and compare it with marker regression
plot(
```

```
  out.em,
  out.mr,
  chr = c(1, 4, 12),
  col = c("blue", "red"),
  ylab = "LOD score"
)
legend(
  x = "topright",
  legend = c("Standard interval mapping", "Marker regression"),
  col = c("blue", "red"),
  lty = 1,
  cex = 0.8
)
```

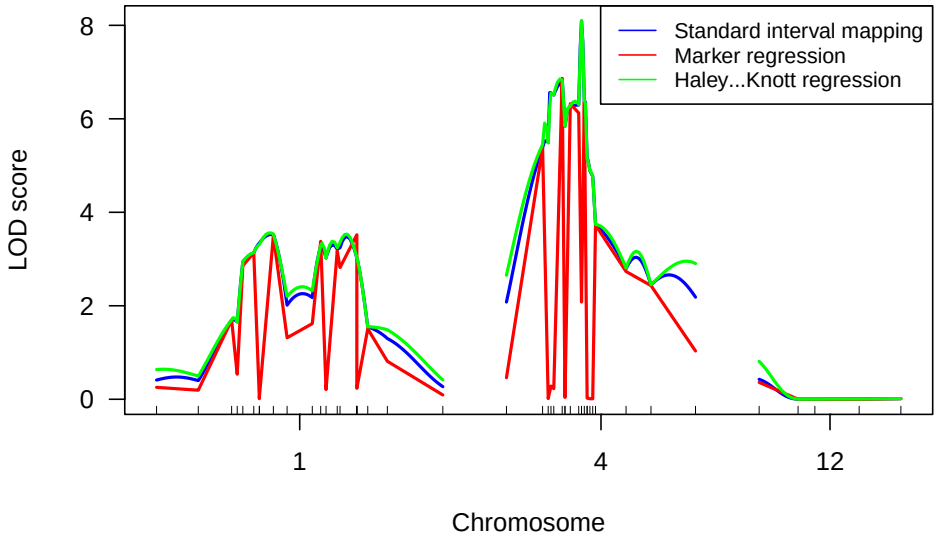



### 6.2.2 Haley–Knott regression

Haley–Knott regression provides a fast approximation of the results of standard interval mapping. Haley–Knott regression provides a fast approximation of the results of standard interval mapping:

```
# Calculate the conditional genotype probabilities
hyper <- calc.genoprob(hyper, step = 1, error.prob = 0.001)

# Perform Haley-Knott regression
out.hk <- scanone(hyper, method = "hk")

# Plot the results and compare it with the previous ones
plot(
  out.em,
  out.mr,
  out.hk,
  chr = c(1, 4, 12),
  col = c("blue", "red", "green"),
```

```
  ylab = "LOD score"
)
legend(
  x = "topright",
  legend = c(
    "Standard interval mapping",
    "Marker regression",
    "Haley-Knott regression"
  ),
  col = c("blue", "red", "green"),
  lty = 1,
  cex = 0.8
)
```

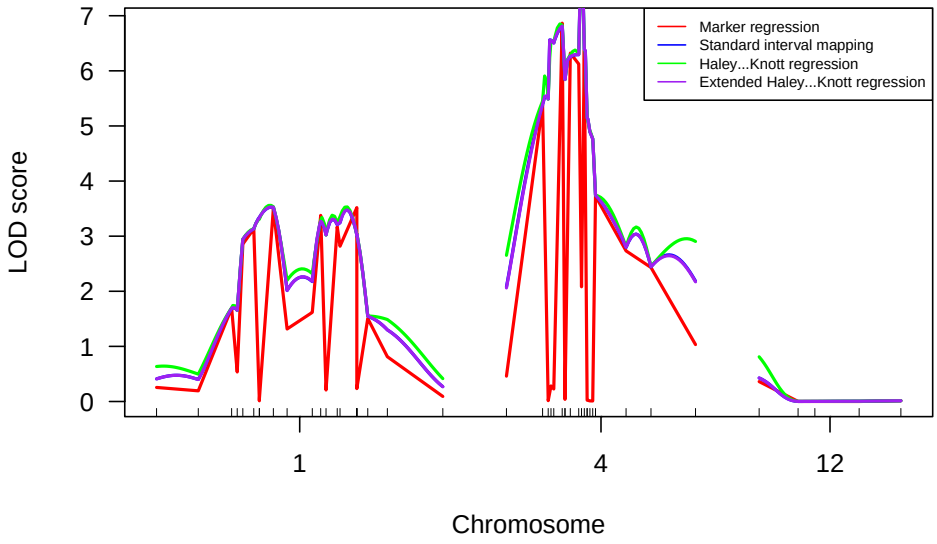



### 6.2.3 Extended Haley–Knott regression

The extended Haley–Knott method is not as fast as Haley–Knott regression, but it provides an improved approximation and is still somewhat faster than standard interval mapping. Most importantly, the extended Haley–Knott method is more robust than standard interval mapping:

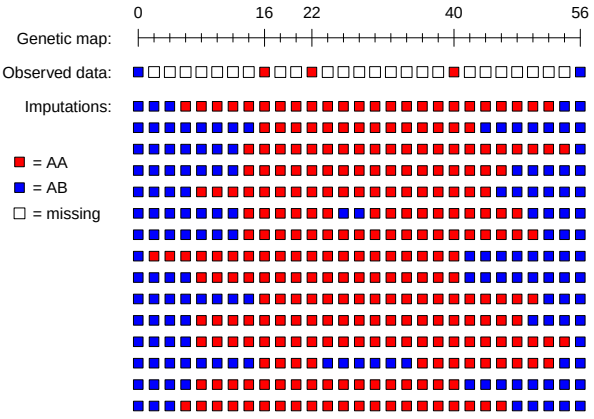
```
# Calculate the conditional genotype probabilities
hyper <- calc.genoprob(hyper, step = 1, error.prob = 0.001)

# Perform Haley-Knott regression
out.ehk <- scanone(hyper, method = "ehk")

# Plot the results and compare it with the previous ones
plot(out.mr, chr = c(1, 4, 12), col = "red", ylab = "LOD score")
plot(out.em, chr = c(1, 4, 12), col = "blue", add = TRUE)
plot(out.hk, chr = c(1, 4, 12), col = "green", add = TRUE)
plot(out.ehk, chr = c(1, 4, 12), col = "purple", add = TRUE)
legend(
```

```
  x = "topright",
  legend = c(
    "Marker regression",
    "Standard interval mapping",
    "Haley-Knott regression",
    "Extended Haley-Knott regression"
  ),
  col = c("red", "blue", "green", "purple"),
  lty = 1,
  cex = 0.6
)
```




### 6.2.4 Multiple imputation

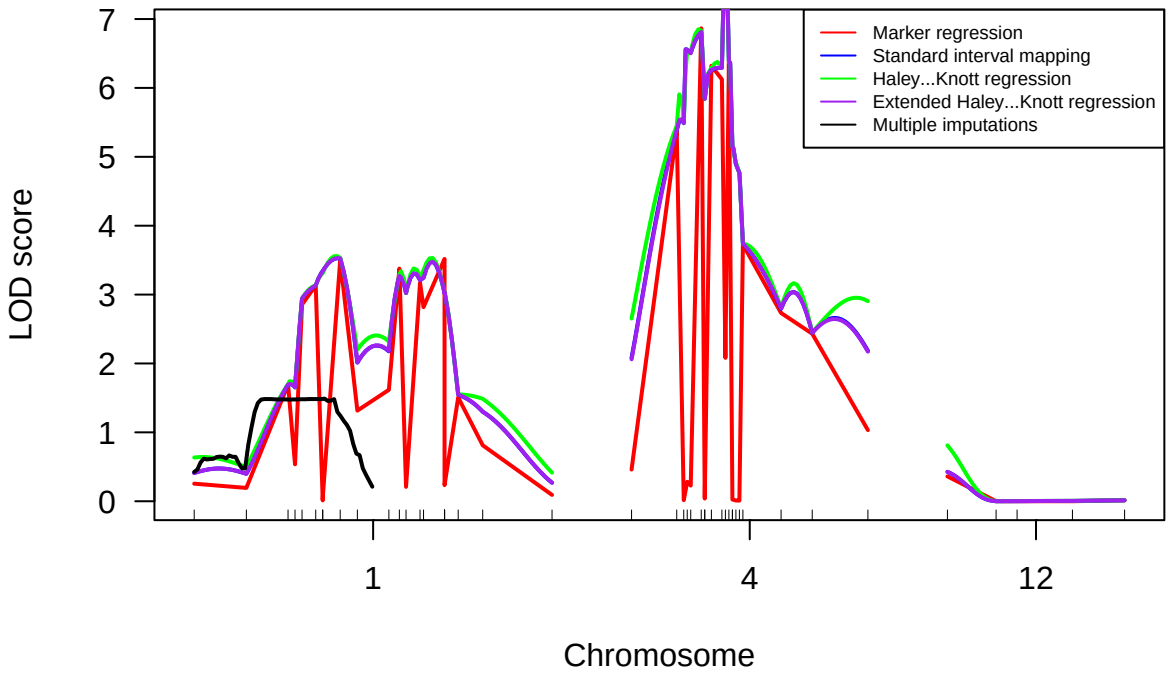
The multiple imputation approach dispenses with the missing data problem by filling in all missing genotype data, even at sites between markers. For example, the following figure illustrates the imputation of a single backcross individual's genotype data:



The observed genotype data at five genetic markers is shown at the top, followed by the imputed genotypes for 15 different imputations. Note that the positions of the recombination events vary among the imputations, and a couple of imputations exhibit double-crossovers between markers.

The multiple imputation approach has greatest value for the fit and exploration of multiple-QTL models:

```
# Perform the imputations
hyper <- sim.geno(hyper,
                  step = 1,
                  n.draws = 64,
                  error.prob = 0.001)

# Perform multiple imputation
out.imp <- scanone(hyper, method = "imp")

# Plot the results and compare it with the previous ones
plot(out.mr, chr = c(1, 4, 12), col = "red", ylab = "LOD score")
plot(out.em, chr = c(1, 4, 12), col = "blue", add = TRUE)
plot(out.hk, chr = c(1, 4, 12), col = "green", add = TRUE)
plot(out.ehk, chr = c(1, 4, 12), col = "purple", add = TRUE)
plot(out.imp, chr = c(1, 4, 12), col = "black", add = TRUE)
legend(
  x = "topright",
  legend = c(
    "Marker regression",
    "Standard interval mapping",
    "Haley-Knott regression",
    "Extended Haley-Knott regression",
    "Multiple imputations"
  ),
  col = c("red", "blue", "green", "purple", "black"),
  lty = 1,
  cex = 0.6
)
```




### 6.2.5 Comparison of methods

Relative advantages and disadvantages of the four interval mapping methods are shown below:

| Method | Use of genotype information | Robustness | Selective genotyping | Speed |
|---|---|---|---|---|
| Standard interval mapping | ++ | - | + | - |
| Haley–Knott | - | + | - | + |
| Extended Haley–Knott | + | + | + | - |
| Multiple imputation | ++ | + | + | – |

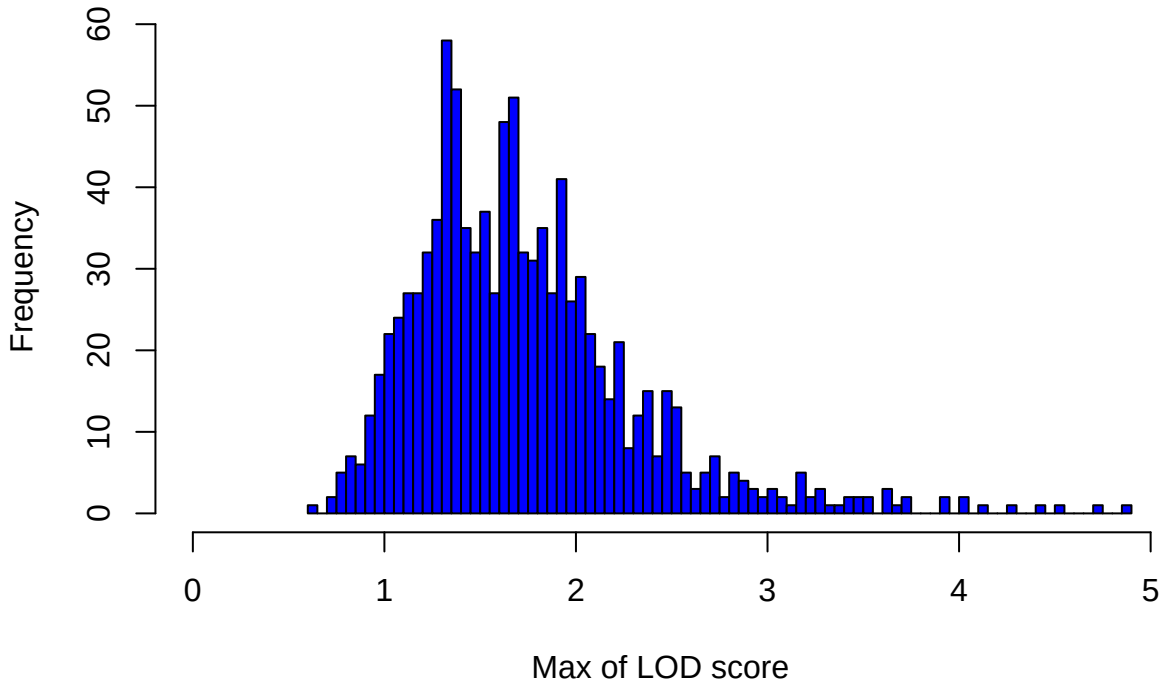
## 6.3 Significance thresholds

A LOD score indicates evidence for the presence of a QTL, with larger LOD scores corresponding to greater evidence. We compare our observed LOD scores to the distribution of the genome-wide maximum LOD score, in the case that there were no QTL anywhere. The 95th percentile of this distribution may be used as a genome-wide LOD threshold. One may derive this null distribution by permutation test. While we generally use 1000 permutation replicates initially, we may go up to 10,000 or even 100,000 replicates in order to achieve greater precision. Alternatively, one may calculate a genome-scan-adjusted p-value corresponding to an observed LOD score.

Let us illustrate the permutation test:

```
# Import data
data(hyper)
```

```
# Obtain the QTL genotype probabilities
hyper <- calc.genoprob(hyper, step = 1, error.prob = 0.001)

# Do the permutation test
operm <- scanone(hyper, n.perm = 1000, verbose = FALSE)

# Plot a histogram of the permutation results
plot(operm, col = "blue", xlab = "Max of LOD score")
```




```
# Obtain genome-wide LOD thresholds for significance levels 20% and 5%
summary(operm, alpha = c(0.20, 0.05))
```

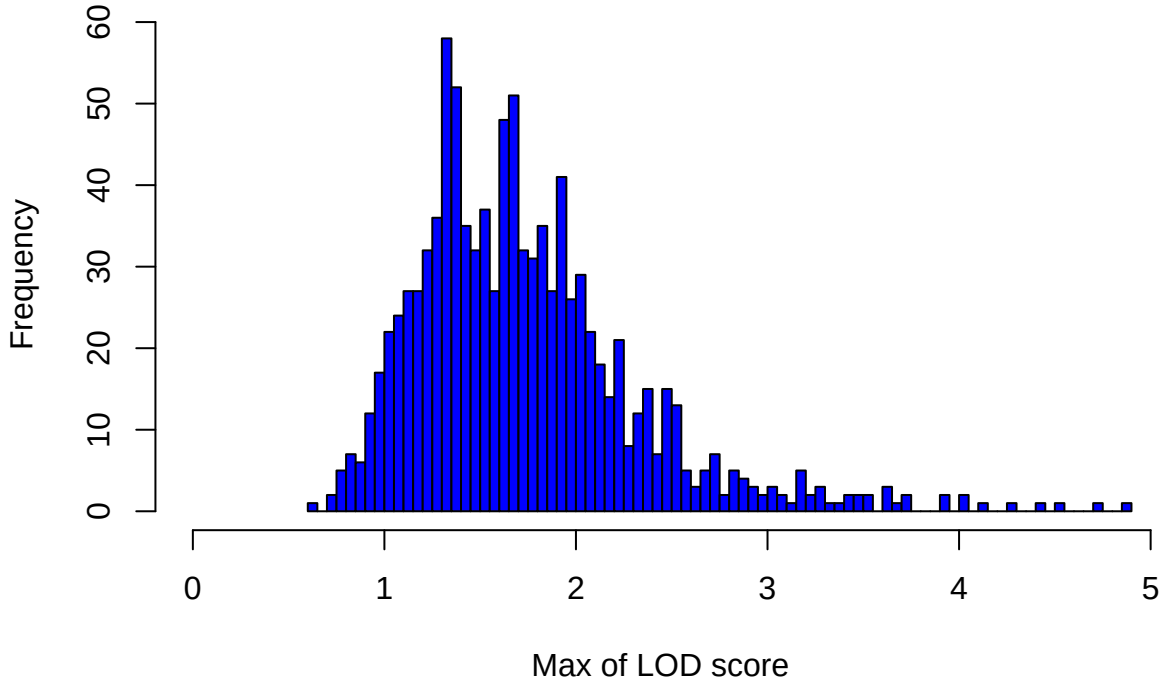
```
## LOD thresholds (1000 permutations)
##      lod
## 20% 2.10
## 5%  2.81
```

In the above, we used the traditional permutation test. However, for the *hyper* data, a selective genotyping strategy was used, and so it is best to use a stratified permutation test, permuting individuals' phenotypes separately within strata defined by the extent of genotyping. We must first define a vector that indicates the strata. This may be done as follows. We place individuals who were genotyped at more than 100 markers in one group and the other individuals in a second group:

```
strat <- (ntyped(hyper) > 100)
```

We then rerun the permutation test:

```
# Do the permutation test
operms <-
  scanone(hyper,
          n.perm = 1000,
          perm.strata = strat,
          verbose = FALSE)

# Plot a histogram of the permutation results
plot(operm, col = "blue", xlab = "Max of LOD score")
```




```
# Obtain genome-wide LOD thresholds for significance levels 20% and 5%
summary(operm, alpha = c(0.20, 0.05))
```

```
## LOD thresholds (1000 permutations)
##      lod
## 20% 2.10
## 5%  2.81
```

In this particular case, we see little difference in the significance threshold when using the stratified permutation test.

Turning now to the use of the permutation results, we can pick out the LOD peaks (no more than one per chromosome) that meet the 10% significance level with the genome-scan-adjusted p-value for the results of single-QTL analysis using the following commands:

```
summary(out.em, perms = operms, alpha = 0.1, pvalues = TRUE)
```

```
##          chr  pos  lod  pval
```

```
## c1.loc45   1 48.3 3.53 0.007
## D4Mit164   4 29.5 8.09 0.000
```

For the QTL on chromosome 4, our estimated p-value is 0. Citing a p-value of 0 doesn't seem right, but we can get an upper confidence limit on the true p-value as follows:

```
binom.test(0, 1000)$conf.int
```

```
## [1] 0.000000000 0.003682084
## attr(,"conf.level")
## [1] 0.95
```

Thus, we might report $p < 0.004$.

## 6.4 The X chromosome

The X chromosome exhibits special behavior and must be treated differently from the autosomes in QTL mapping.

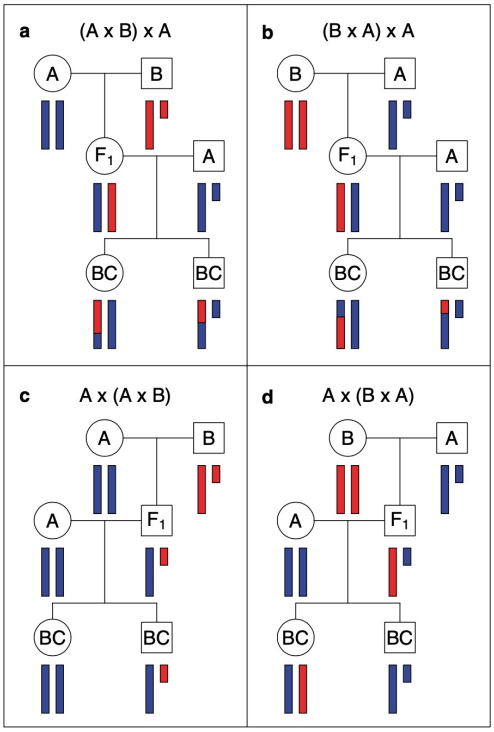



Figure 4: The behavior of the X chromosome in a backcross. Circles and squares correspond to females and males, respectively. The small bar is the Y chromosome.

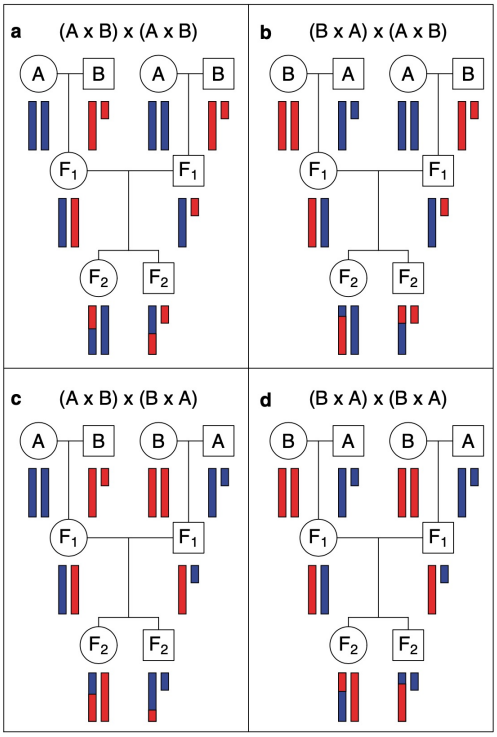



Figure 5: The behavior of the X chromosome in a intercross Circles and squares correspond to females and males, respectively. The small bar is the Y chromosome.

If not taken into account, such systematic differences can lead to large LOD scores on the X chromosome even in the absence of X chromosome linkage. Finally, to account for the fact that the number of degrees of freedom for the linkage test on the X chromosome may be different from that on the autosomes, an X-chromosome-specific significance threshold is required.

### 6.4.1 Analysis

For all interval mapping methods, the actual analysis is essentially the same for the X chromosome as for the autosomes. As each backcross or intercross individual has a single X chromosome that was subject to recombination,the calculation of the genotype probabilities given the available multipoint marker genotype data is identical to those for an autosome in a backcross, and so nothing new is needed there.

### 6.4.2 Significance thresholds

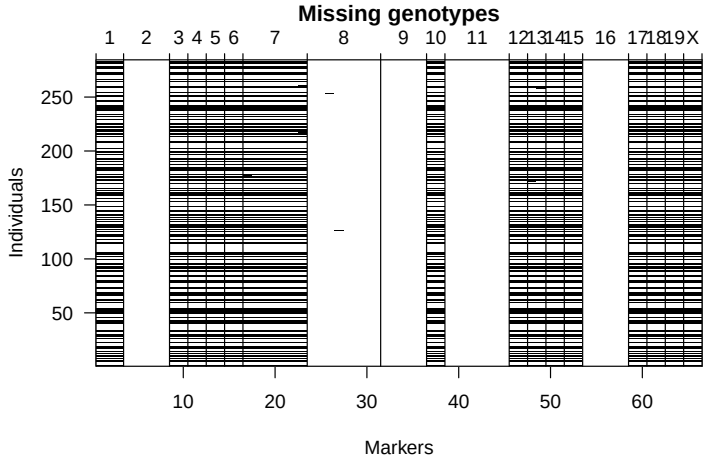
We use a constant LOD threshold across the autosomes and a separate threshold for the X chromosome. The precise estimation of the X-chromosome-specific LOD threshold will require considerably more permutation replicates.

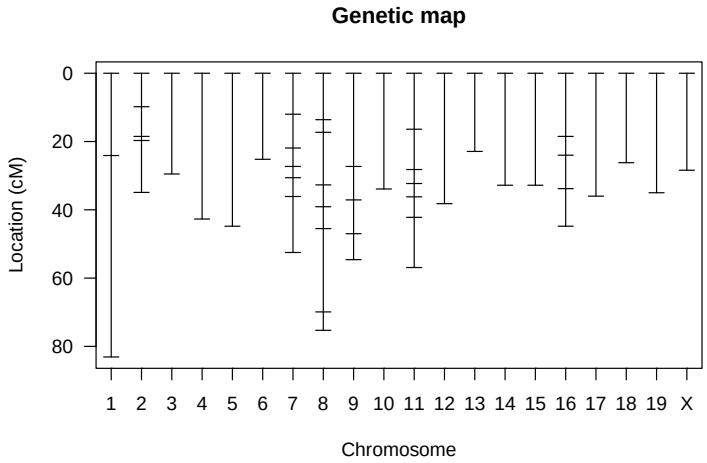
### 6.4.3 Example

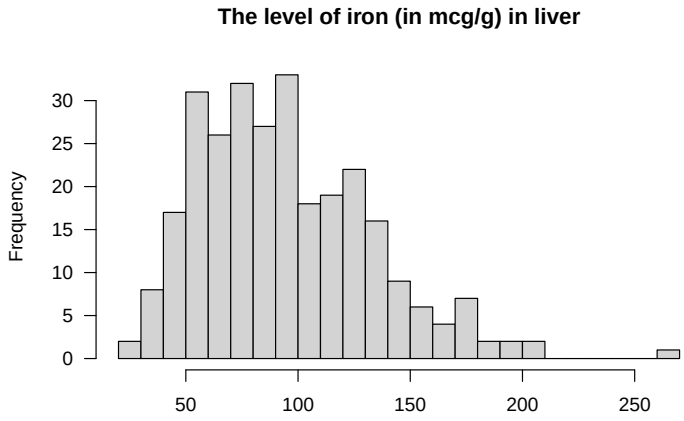
As an example, we consider the data of Grant et al. (2006), which concerns the basal iron levels in the liver and spleen of intercross mice. Both sexes from reciprocal intercrosses with the "C57BL/6J/Ola" and "SWR/Ola" strains were used; there are 284 individuals in total. There are two phenotypes: the level of iron (in mcg/g) in the liver and spleen. There are approximately equal proportions of males and females and of mice from each cross direction:

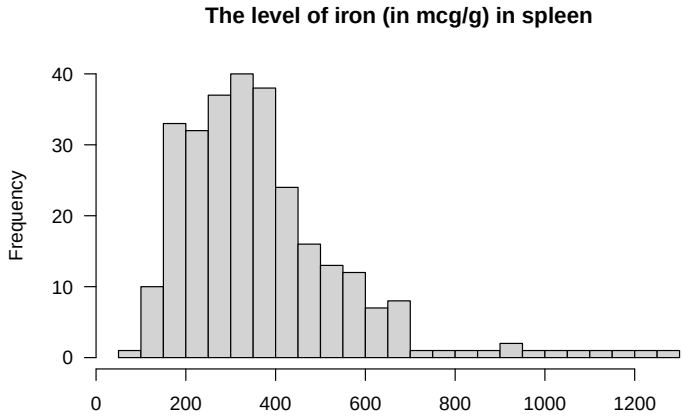
```
# Import data
library(qtlbook)
```

```r
data(iron)

# The pattern of missing genotype data
plotMissing(iron)

# The genetic map of the typed markers
plot.map(iron)

# A histogram of the first phenotype
plotPheno(iron,
          pheno.col = 1,
          xlab = "",
          main = "The level of iron (in mcg/g) in liver")

# A histogram of the second phenotype
plotPheno(iron,
          pheno.col = 2,
          xlab = "",
          main = "The level of iron (in mcg/g) in spleen")
```



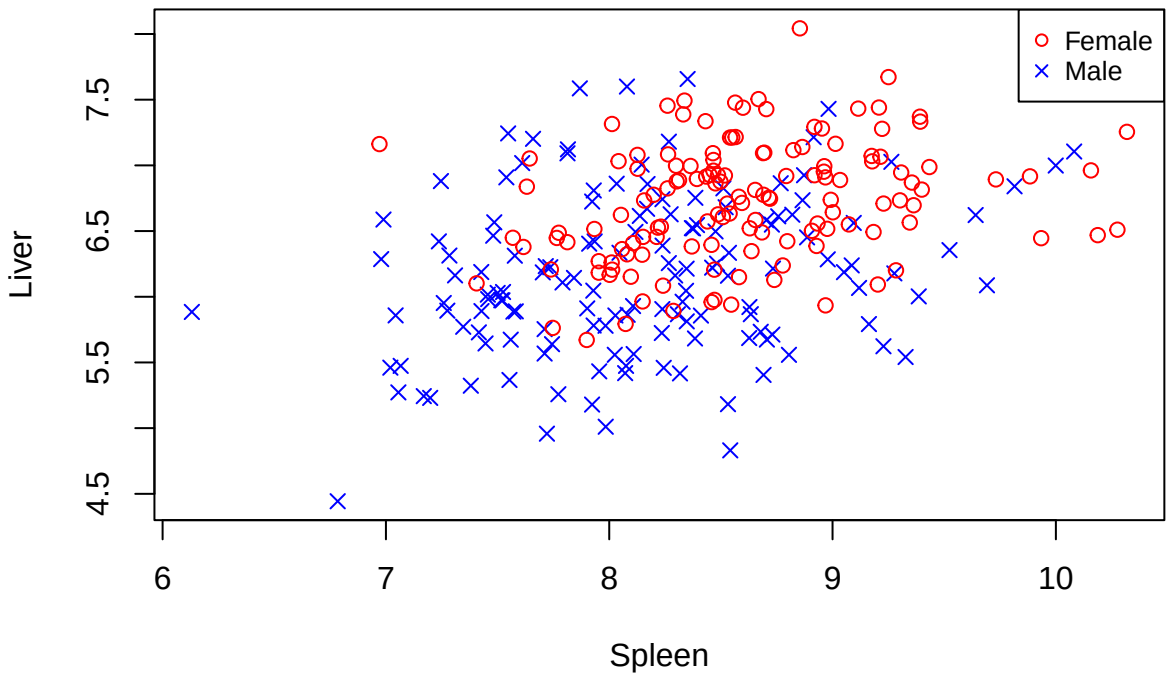






Now plot a scatterplot of the log2(liver) and log2(spleen) phenotypes:

```
plot(
  log2(liver) ~ log2(spleen),
  data = iron$pheno,
  col = c("red", "blue")[iron$pheno$sex],
  pch = c(1, 4)[iron$pheno$sex],
  xlab = "Spleen",
  ylab = "Liver"
)
legend(
  x = "topright",
  legend = c("Female", "Male"),
  col = c("red", "blue"),
  pch = c(1, 4),
  cex = 0.8
)
```



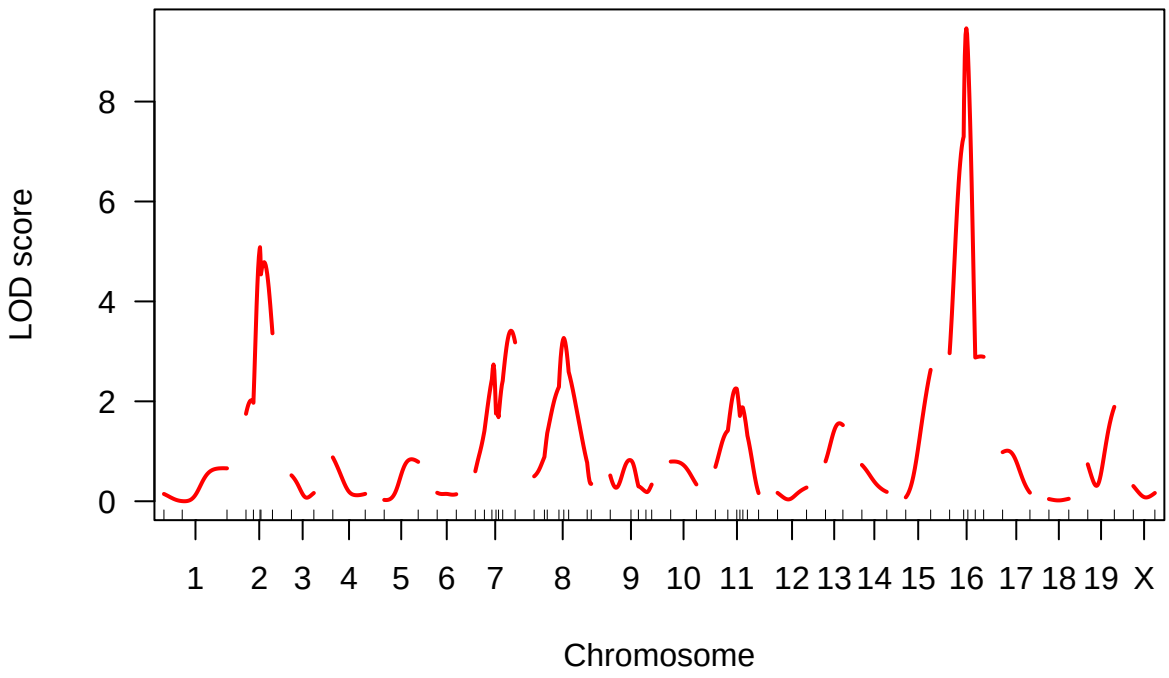

Note that both phenotypes show a large sex difference, with females having larger iron levels than males. We will focus on the liver phenotype now. We transform the phenotype, and then perform a genome scan by standard interval mapping:

```
iron$pheno[, 1] <- log2(iron$pheno[, 1])
iron <- calc.genoprob(iron, step = 1, error.prob = 0.001)
out.liver <- scanone(iron)

# Check results with with LOD > 3
summary(out.liver, 3)
```

```
##           chr  pos  lod
## D2Mit17     2 56.8 5.09
## c7.loc47    7 48.1 3.41
## D8Mit294    8 39.1 3.27
## c16.loc22  16 28.6 9.47
```

Let's plot the results:

```
plot(
  out.liver,
  col = "red",
  ylab = "LOD score"
)
```




Now let's perform X-chromosome-specific permutations:

```
operm.liver <-
  scanone(iron,
          n.perm = 1000,
          perm.Xsp = TRUE,
          verbose = FALSE)

# The LOD thresholds for a 5% significance level
summary(operm.liver, alpha = 0.05)
```

```
## Autosome LOD thresholds (1000 permutations)
##     lod
## 5% 3.36
##
## X chromosome LOD thresholds (28243 permutations)
##     lod
## 5% 4.79
```

The threshold for the X chromosome is much larger than that for the autosomes. Now calculate the relevant LOD thresholds corresponding to a particular significance level and to obtain genome-scan-adjusted p-values:

```
summary(out.liver,
        perms = operm.liver,
        alpha = 0.05,
        pvalues = TRUE)
```

```
##           chr  pos  lod    pval
```

```
## D2Mit17     2 56.8 5.09 0.00311
## c7.loc47    7 48.1 3.41 0.04449
## c16.loc22  16 28.6 9.47 0.00000
```

## 6.5 Interval estimates of QTL location

Once one has obtained evidence for a QTL, one may seek an interval estimate of the location of the QTL. There are two major methods for calculating an interval estimate of QTL location: LOD support intervals and Bayes credible intervals.

The 1.5-LOD support interval is the interval in which the LOD score is within 1.5 units of its maximum. We prefer to use 1.5-LOD support intervals for an backcross, and 1.8-LOD support intervals for an intercross:

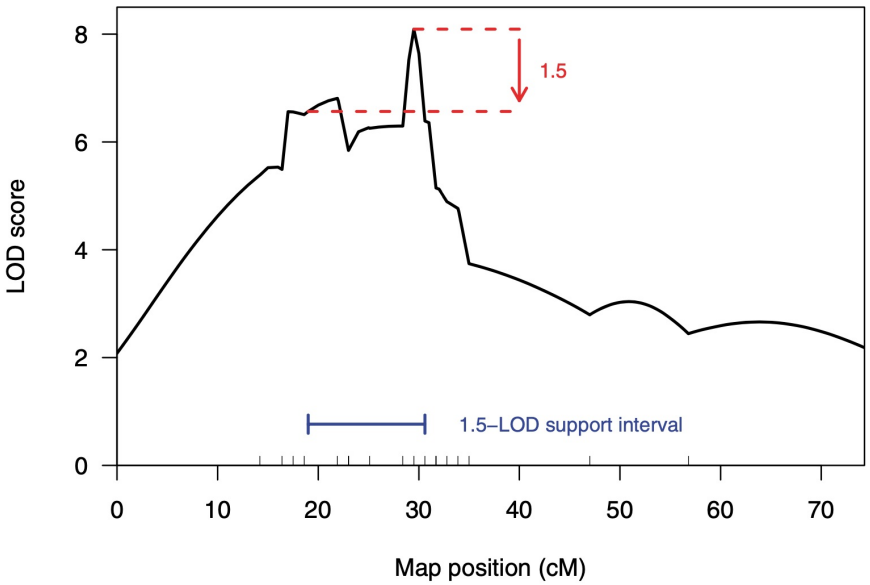
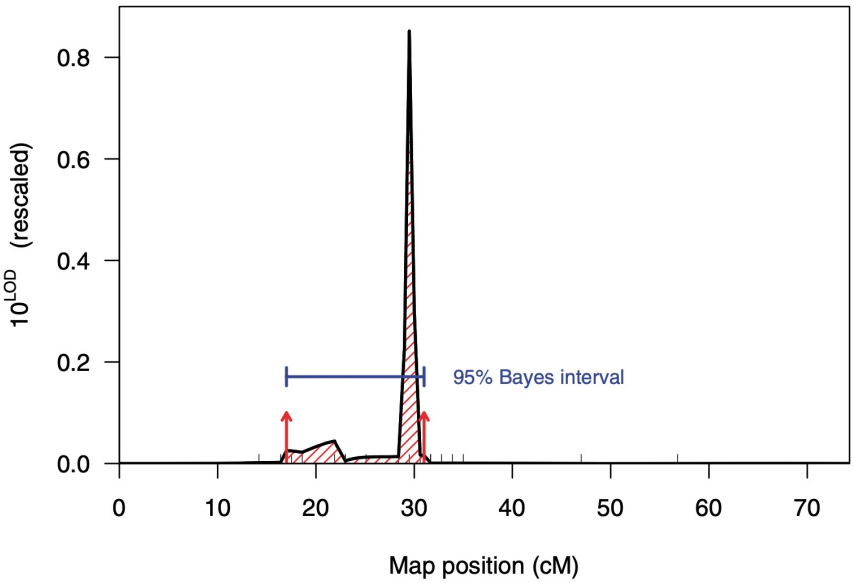



Figure 6: Illustration of the 1.5-LOD support interval.

```
# The 1.5-LOD support for chromosome 4 in the "hyper" data
lodint(results = out.em, chr = 4, qtl.index = 1.5)
```

```
##          chr      pos      lod
## c4.loc19   4 19.00000 6.565531
## D4Mit164   4 29.50001 8.092266
## D4Mit178   4 30.60001 6.387156
```

An approximate Bayes credible interval is obtained by viewing 10 ^ LOD as a real likelihood function for QTL location: \begin{figure}[H]




{

}

\caption{Illustration of the approximate 95% Bayes credible interval.} \end{figure}

```
# The 95% Bayes credible intervals for chromosome 4 in the "hyper" data
bayesint(results = out.em, chr = 4, prob = 0.95)
```

```
##          chr      pos      lod
## c4.loc17   4 17.00000 6.561763
## D4Mit164   4 29.50001 8.092266
## c4.loc31   4 31.00000 6.358530
```

The first and last rows in the results indicate the ends of the intervals; the middle row is the maximum likelihood estimate of QTL location. Experience has show that Bayes intervals have remarkably consistent coverage, and so may be preferred.

## 6.6 QTL effects

The effect of a QTL is characterized by the difference in the phenotype averages among the QTL genotype groups. It has important influence on one's ability to fine-map the QTL and ultimately identify the causal gene. Among those experiments in which the QTL is detected, the estimated QTL effect will be, on average, larger than its true effect. This is selection bias. Note that the selection bias is largest for QTL with small effects.

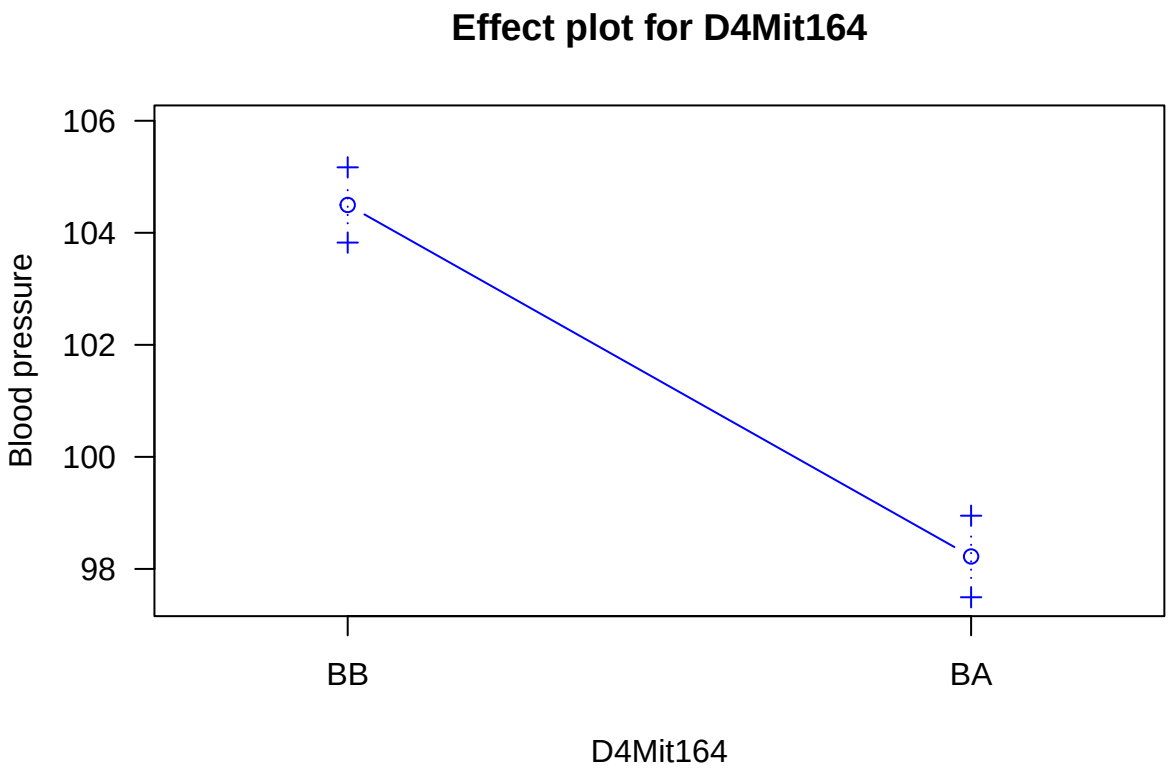
The function *effectplot* uses the multiple imputation method to obtain estimates of the genotype-specific phenotype averages. For the "hyper" data, the largest LOD score was obtained at marker *D4Mit164* (chr 4, 29.5 cM). If we had known only the position, we could find the name of the nearest marker as follows:

```
find.marker(hyper, chr = 4, pos = 29.5)
```

```
## [1] "D4Mit164"
```

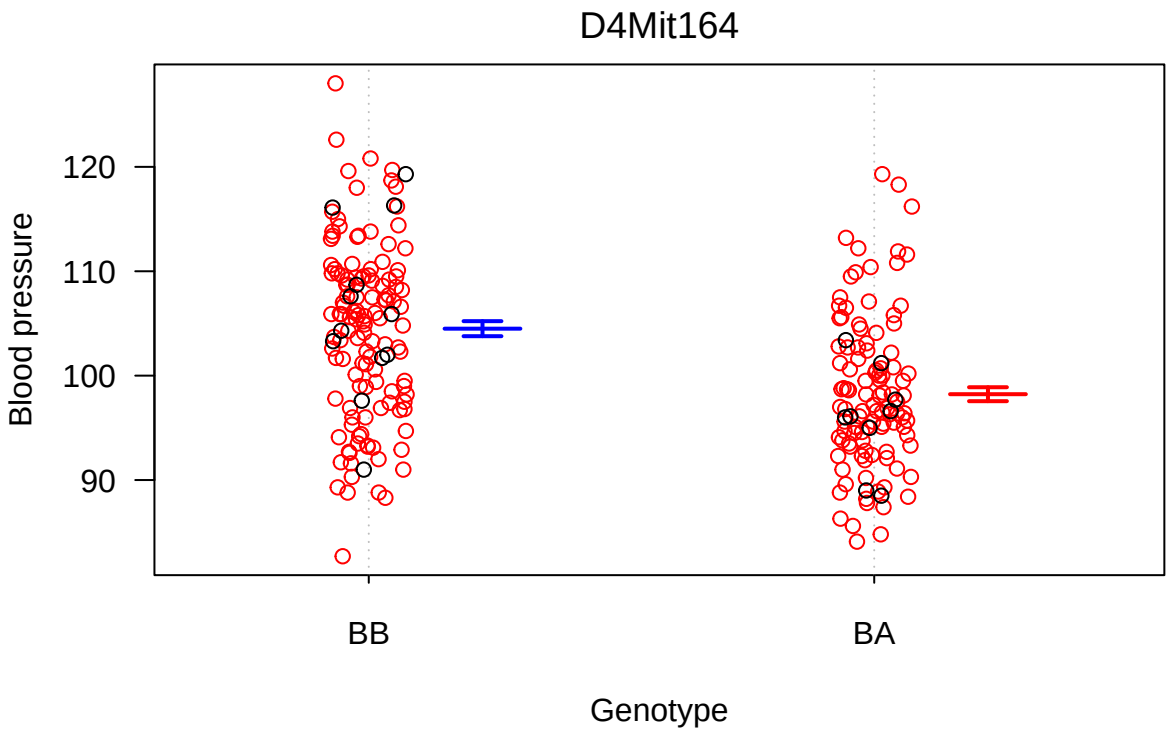
The *effectplot* for marker *D4Mit164* in the "hyper" data may be obtained as follows:

```
hyper <- sim.geno(hyper, n.draws = 16, error.prob = 0.001)
effectplot(hyper,
           mname1 = "D4Mit164",
           ylab = "Blood pressure",
           col = "blue")
```




The function *plotPXG* can be used to create a dot plot of a phenotype against the genotypes at a marker:

```
plotPXG(hyper, marker="D4Mit164", ylab = "Blood pressure")
```




Genotypes that were imputed are plotted in red. Error bars are ± 1 SE.

## 6.7 Multiple phenotypes

In a QTL experiment, one often measures multiple related phenotypes. The joint analysis of multiple phenotypes can increase the power for QTL detection and the precision of QTL localization, and can allow one to test for pleiotropy (that a single QTL influences multiple phenotypes) versus tight linkage of distinct QTL:

```
# Import data
data(iron)
iron$pheno <-
  cbind(
    iron$pheno[, 1:2],
    log2liver = log2(iron$pheno$liver),
    log2spleen = log2(iron$pheno$spleen),
    iron$pheno[, 3:4]
  )

# Obtain the QTL genotype probabilities
iron <- calc.genoprob(iron, step = 1, error.prob = 0.001)

# Analyze all four phenotypes
out.all <- scanone(iron, pheno.col = 1:4)
```

The *summary* function displays (by default) the peak positions for the first phenotype, but also shows LOD scores at those positions for the other phenotypes:

```
summary(out.all)[1:4, ]
```

```
##          chr   pos liver spleen log2liver log2spleen
## D1Mit17    1 110.4 0.900 0.0482     0.659     0.0866
## D2Mit17    2  56.8 4.907 1.9169     5.086     2.2792
## D3Mit22    3  25.1 0.448 0.0753     0.519     0.0372
## D4Mit2     4  10.9 1.011 0.3351     0.878     0.3044
```

We can look at the peaks for another phenotype with the *lodcolumn* argument:

```
summary(out.all, threshold = 3, lodcolumn = 4)
```

```
##          chr  pos liver spleen log2liver log2spleen
## D8Mit4     8 13.6 0.739    4.3     0.887        3.9
## c9.loc50   9 56.6 0.183   10.9     0.199       12.6
```

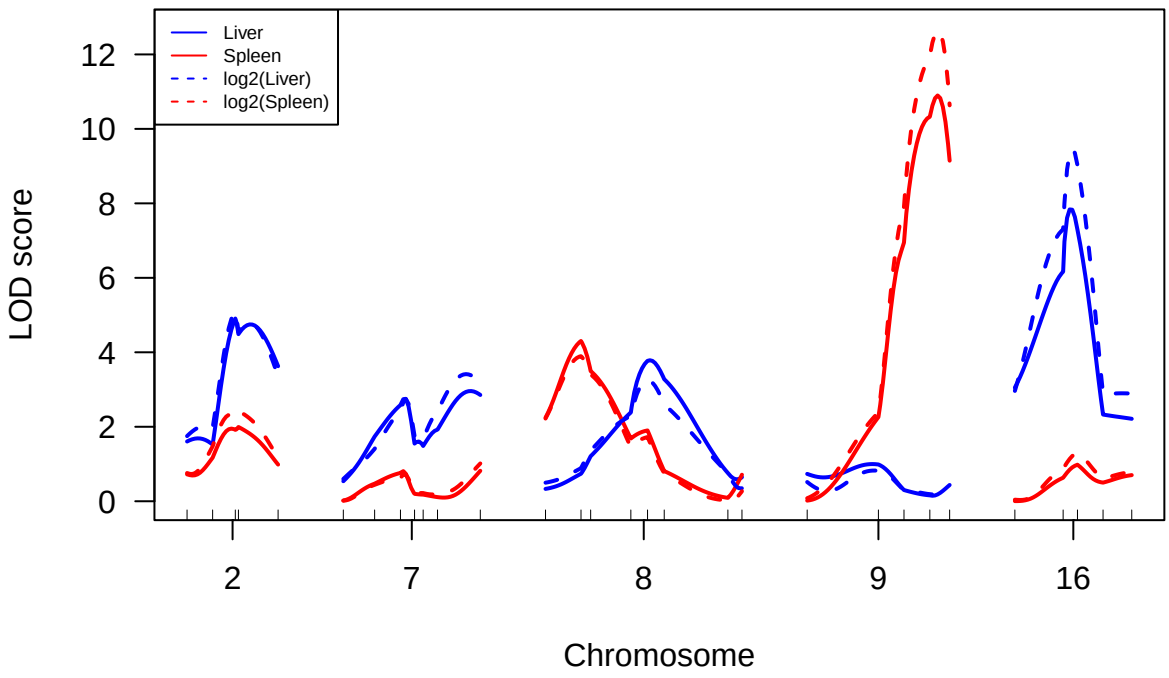
To display the peak LOD scores for all phenotypes, provided that maximum LOD score exceeded 3:

```
summary(out.all, threshold = 3, format = "allpheno")[1:4, ]
```

```
##          chr  pos liver spleen log2liver log2spleen
## D2Mit17    2 56.8 4.907  1.917     5.086      2.279
## c7.loc47   7 48.1 2.935  0.395     3.413      0.596
## D8Mit4     8 13.6 0.739  4.303     0.887      3.895
## D8Mit294   8 39.1 3.769  1.902     3.268      1.724
```

If we want to look at the results for all four phenotypes, we might do the following:

```
plot(
  out.all,
  lodcolumn = 1:2,
  col = c("blue", "red"),
  chr = c(2, 7, 8, 9, 16),
  lty = 1,
  ylim = c(0, 12.7),
  ylab = "LOD score"
)
plot(
  out.all,
  lodcolumn = 3:4,
  col = c("blue", "red"),
  lty = 2,
  chr = c(2, 7, 8, 9, 16),
  add = TRUE
)
legend(
  x = "topleft",
  legend = c(
    "Liver",
    "Spleen",
    "log2(Liver)",
    "log2(Spleen)"
  ),
  col = c("blue", "red", "blue", "red"),
  lty = c(1, 1, 2, 2),
  cex = 0.6
)
```




# 7 Non-normal phenotypes

The methods discussed above all rely on the assumption that, given QTL genotype, the phenotype follows a normal distribution. While the normality assumption is often reasonable, departures from normality are not uncommon.

## 7.1 Nonparametric interval mapping

In the case of complete genotype data at a putative QTL, standard interval mapping is equivalent to using a t test (for a backcross) or analysis of variance (for an intercross). The nonparametric analogs of these methods are the Wilcoxon rank-sum test and the Kruskal–Wallis test.

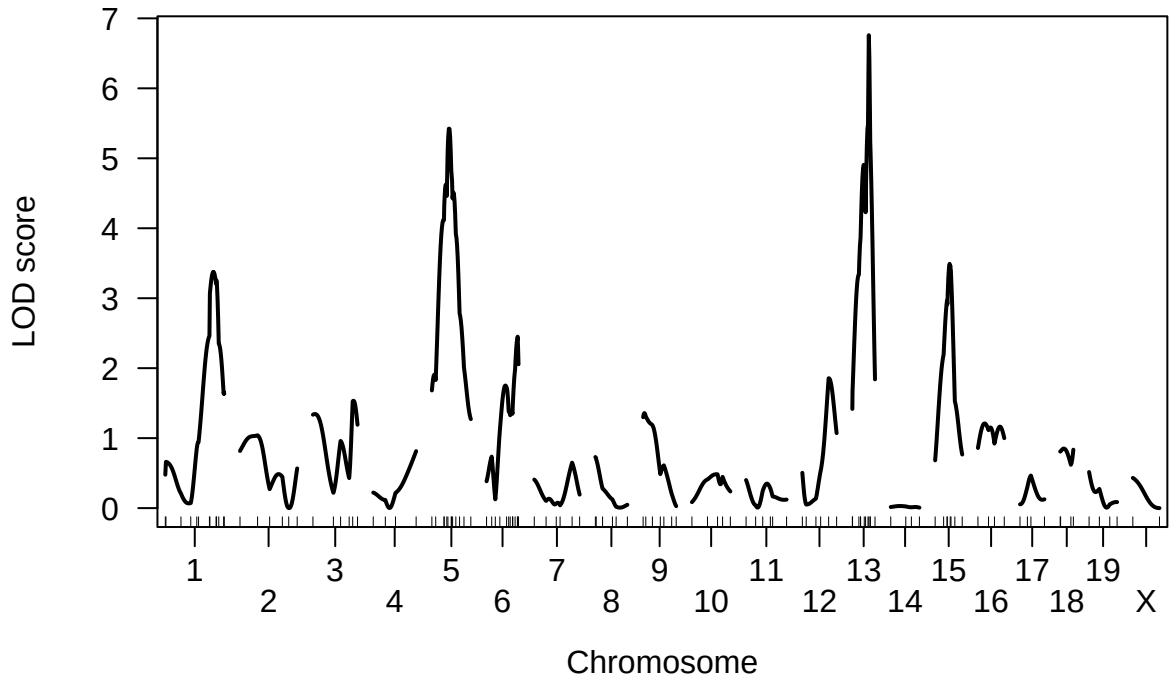
To illustrate these nonparametric methods, we consider the *listeria* data. This is a mouse intercross; the phenotype concerns survival time following infection with *Listeria monocytogenes*, and exhibits a spike at 264 hours:

```
# Import data
library(qtl)
data(listeria)

# Calculate the QTL genotype probabilities
listeria <- calc.genoprob(listeria, step = 1, error.prob = 0.001)
```

Now to perform nonparametric interval mapping:

```
out.np <- scanone(listeria, method = "em", model="np")
```

```r
# Plot the results
plot(out.np, ylab = "LOD score", alternate.chrid = TRUE)
```




A permutation test may aslo be performed as follows:

```r
operm.np <-
  scanone(
    listeria,
    method = "em",
    model = "np",
    n.perm = 1000,
    perm.Xsp = TRUE,
    verbose = FALSE
  )
```

The 5% LOD thresholds:

```r
summary(operm.np, alpha = 0.05)
```

```
## Autosome LOD thresholds (1000 permutations)
##     lod
## 5% 3.27
##
## X chromosome LOD thresholds (25078 permutations)
##    lod
## 5% 2.5
```

Significant evidence for a QTL is seen on chromosomes 1, 5, 13, and 15:

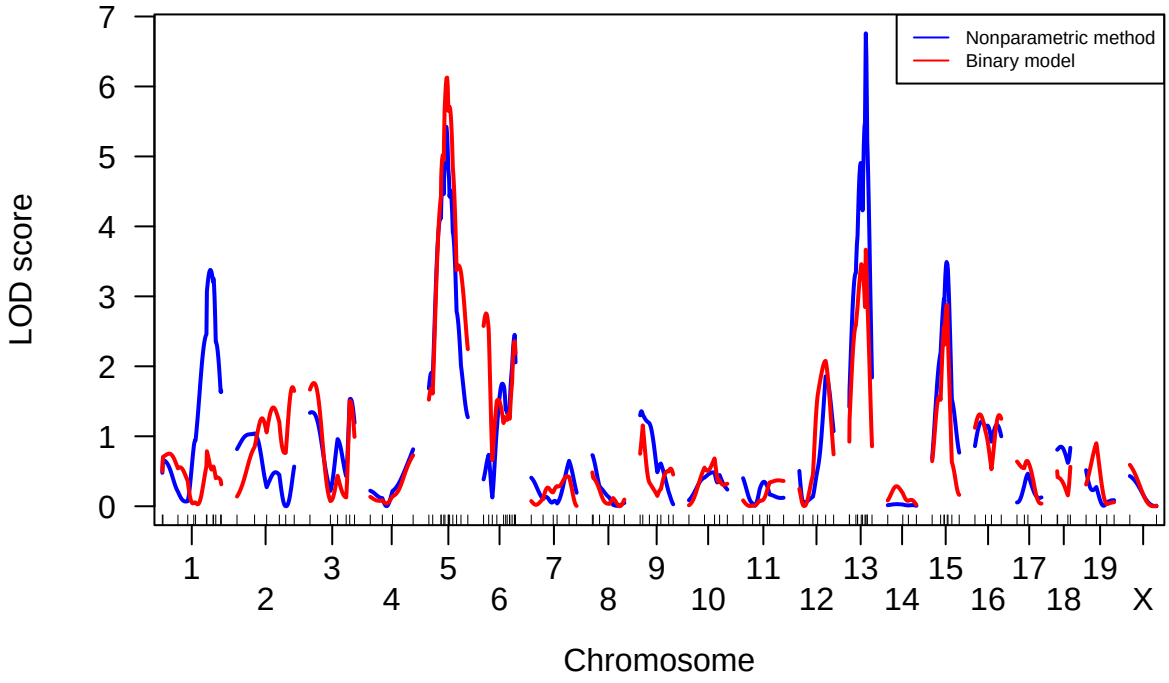
```
summary(out.np,
        perms = operm.np,
        alpha = 0.05,
        pvalues = TRUE)
```

```
##           chr  pos  lod   pval
## c1.loc76    1 76.0 3.38 0.0426
## c5.loc27    5 27.0 5.42 0.0000
## D13M147    13 26.2 6.76 0.0000
## c15.loc23  15 23.0 3.49 0.0353
```

## 7.2 Binary traits

Interval mapping for binary traits is no more difficult than for quantitative traits. Let us apply the binary trait method to the *listeria* data, taking as the binary trait whether the individuals' survived the infection or not (survived for $> 250$ hours).

```
binphe <- as.numeric(pull.pheno(listeria, 1) > 250)
listeria$pheno <- cbind(listeria$pheno, binary = binphe)
out.bin <- scanone(listeria, pheno.col = "binary", model = "binary")
plot(
  out.np,
  out.bin,
  col = c("blue", "red"),
  ylab = "LOD score",
  alternate.chrid = TRUE
)
legend(
  x = "topright",
  legend = c(
    "Nonparametric method",
    "Binary model"
  ),
  col = c("blue", "red"),
  lty = 1,
  cex = 0.6
)
```



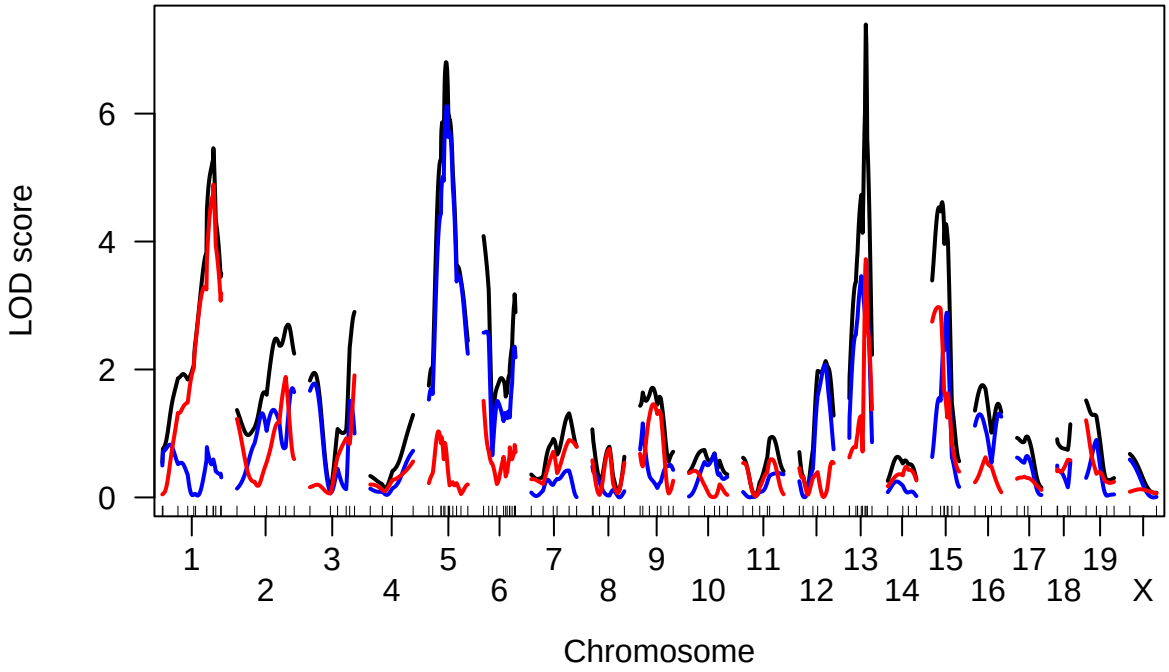

### 7.3 Two-part model

One often observes a spike in the phenotype distribution. In this section, we describe an analysis method particular for this situation. We calculate two additional sets of LOD scores. The LOD score for the two-part model is simply the sum of the LOD scores from the two separate analyses.

As an illustration, we again consider the *listeria* data. There is a spike for the survival phenotype. The spike in the phenotype is assumed to be either the largest or the smallest observed phenotype. We must use the argument "upper = TRUE" to indicate that it is the largest observed phenotype (264 hr) that is to be treated as the spike:

```
# Get log survival time
y <- log(pull.pheno(listeria, 1))
listeria$pheno <- cbind(listeria$pheno, logsurv = y)

# Calculate the LOD curves
out.2p <-
  scanone(listeria,
          model = "2part",
          upper = TRUE,
          pheno.col = "logsurv")

# Plot the results
plot(
  out.2p,
  lodcolumn = 1:3,
  ylab = "LOD score",
```

```
  alternate.chrid = TRUE
)
```




The results indicate that the locus on chromosome 1 largely affects time-to-death, given that an individual has died (LOD in red, is large while LOD in blue, is small). The locus on chromosome 5 largely affects the chance of survival (LOD in blue, is large while LOD in red, is small). The loci on chromosomes 13 and 15 affect both aspects of the phenotype (both LODs are large).

# 8 Experimental design and power

## 8.1 Phenotypes and covariates

In addition to the primary phenotype of interest, it is useful to record all covariates that may affect the phenotype. Sex and cross direction should always be recorded. Whenever possible date/time of experiment, experiment batch, technician name, cage number, litter number, and parent IDs should be recorded.

## 8.2 Strains and strain surveys

We would generally want to cross two strains that exhibit a consistent difference in the phenotype of interest. An experimenter may perform a strain survey by comparing the phenotype of interest in a number of available strains. Once there is reliable data on the phenotypes of strains, if we see differences in a phenotype between two strains, then we can be confident that the difference is genetic. Our next step would be to cross the two strains creating genetic variation, so that we can study the association between genetic markers and the phenotype.

## 8.3 Theory

Having decided which strains to cross, an investigator will have to decide what type of cross to use (e.g., backcross, intercross, or recombinant inbred lines), how many progeny to raise and phenotype, and the genotyping and phenotyping strategies.

Barring specific knowledge about the mode of action of the QTL, the intercross is often the best cross choice. It segregates all possible genotypes, and therefore permits detection of QTL with any mode of action (dominant, recessive, additive, or overdominant). If the phenotype is noisy, with a lot of environmental variation, then RILs (provided that they are available) are the best choice, as we can use replicate individuals to decrease noise. If we are confident of the nature of the effect, or suspect substantial genetic variance due to epistasis, then performing a backcross would be a good choice.

## 8.4 Example

Let us explore what effects we can detect using a backcross or intercross population with 100 individuals. We assume that we desire 80% power in a mouse cross. We first estimate the 5% genome-wide LOD threshold that we will use for the mouse genome (of size 1440 cM), assuming infinitely dense markers:

```
library(qtlDesign)

# Threshold for backcross
thresh(G = 1440, cross = "bc", p = 0.05)
```

```
## [1] 3.189658
```

```
# Threshold for intercross
thresh(G = 1440, cross = "f2", p = 0.05)
```

```
## [1] 4.182964
```

We can now calculate the minimum detectable effect sizes:

```
# For backcross
detectable(
  cross = "bc",
  n = 100,
  sigma2 = 1,
  thresh = 3.2
)
```

```
##         effect percent.var.explained
## [1,] 0.9360905              17.97001
```

```
# For intercross
detectable(
  cross = "f2",
  n = 100,
  sigma2 = 1,
  thresh = 4.2
)
```

```
##      additive.effect dominance.effect percent.var.explained
## [1,]       0.7259998                0              20.85714
```