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.

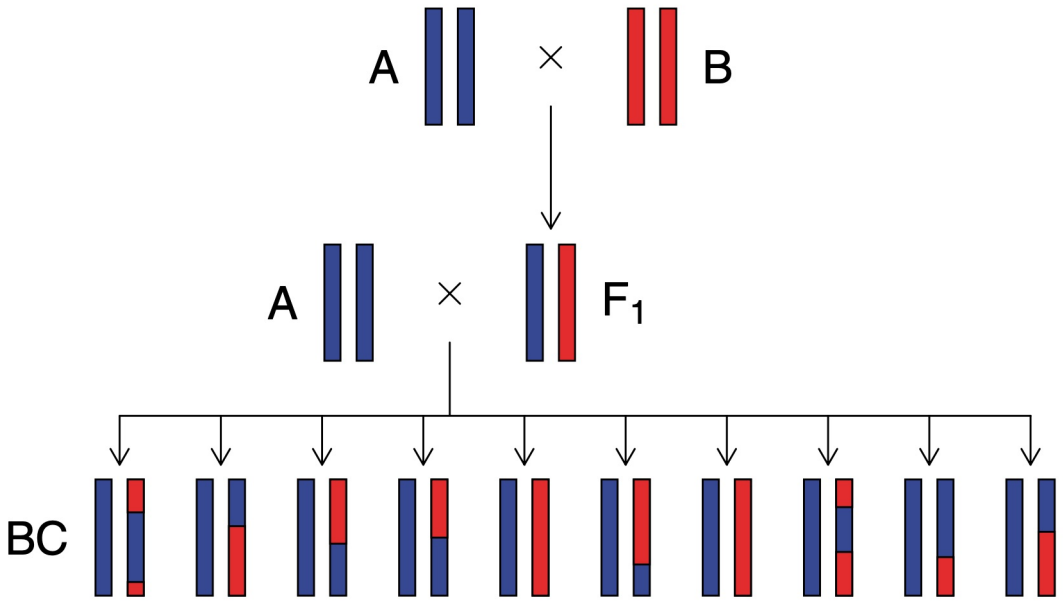


Figure 1: 11-1; Backcrossing

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.

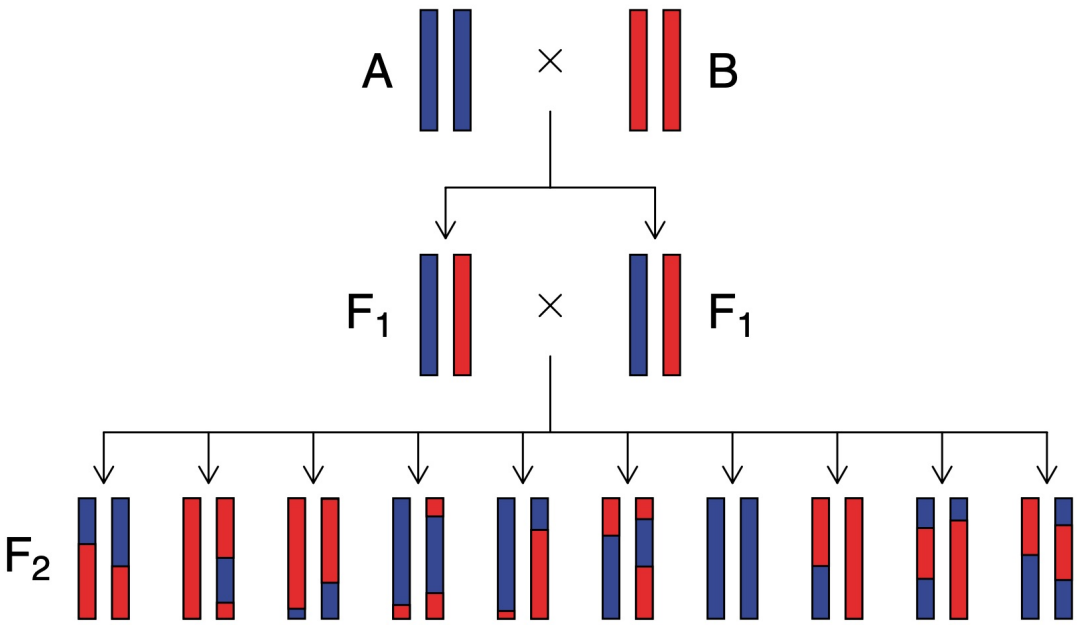


Figure 2: 11-2; Intercrossing

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.

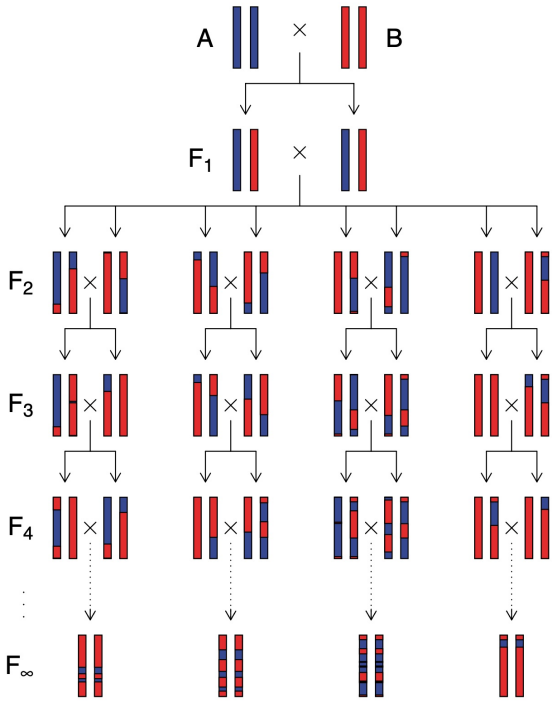


Figure 3: 11-3; Recombinant inbred lines

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:

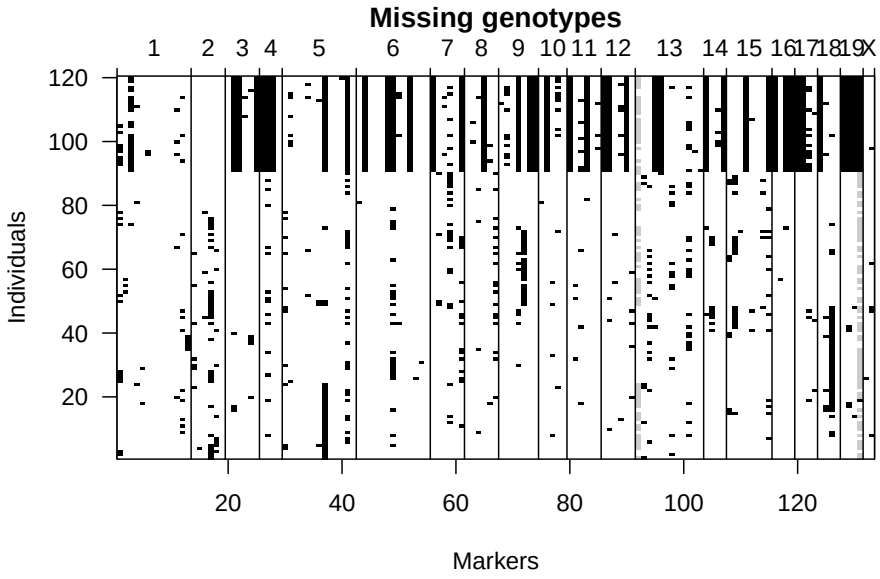
```
# Import data
library(qtl)
data("listeria")

# Check data summary
summary(listeria)
```

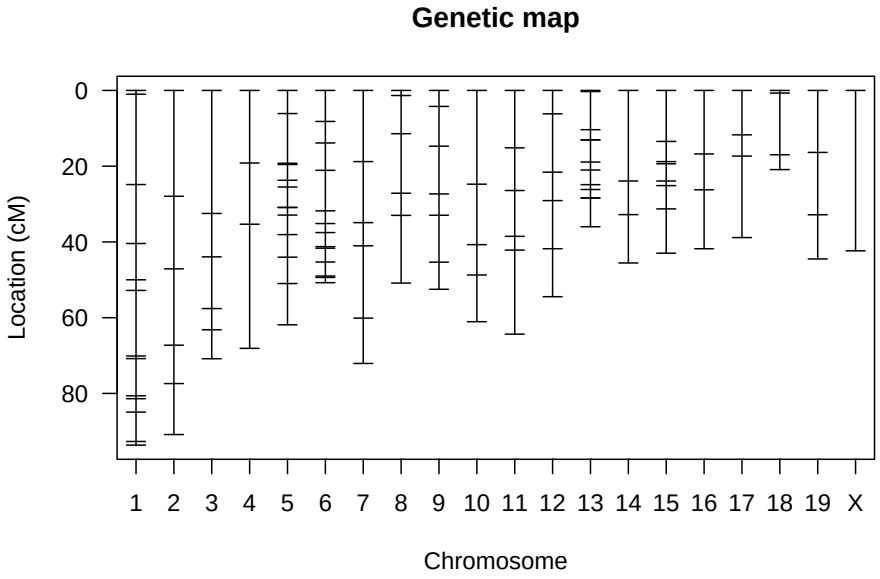
```
##     F2 intercross
##
##     No. individuals:    120
##
##     No. phenotypes:     2
##     Percent phenotyped: 96.7 100
##
##     No. chromosomes:    20
##         Autosomes:      1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
##         X chr:          X
##
##     Total markers:      133
##     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:

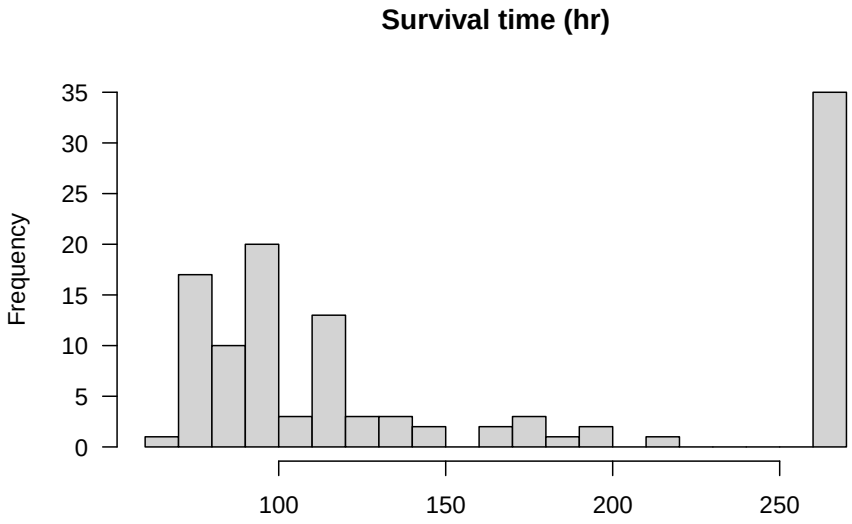
```
# The pattern of missing genotype data
plotMissing(listeria)
```



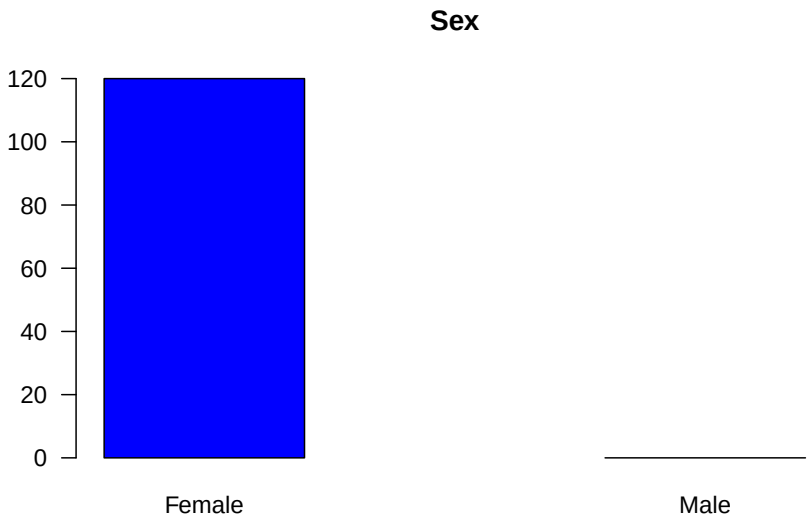

```
# The genetic map of the typed markers
plot.map(listeria)
```




```
# 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

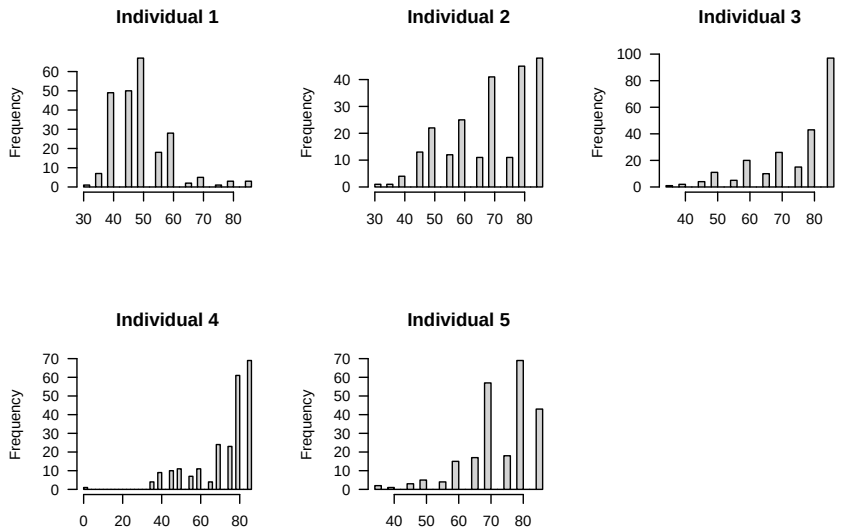
5.1 Phenotypes

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:

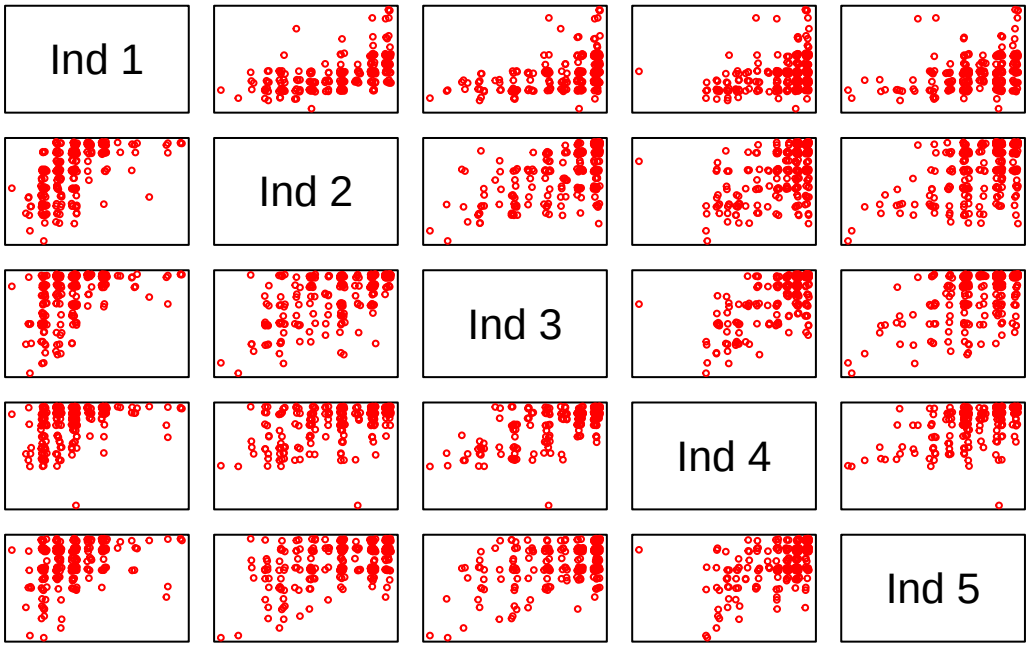
```
par(mfrow = c(2, 3))
for (i in 1:5) {
  plotPheno(ch3a, pheno.col = i, main = paste("Individual", i), xlab = "")
}
```



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

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

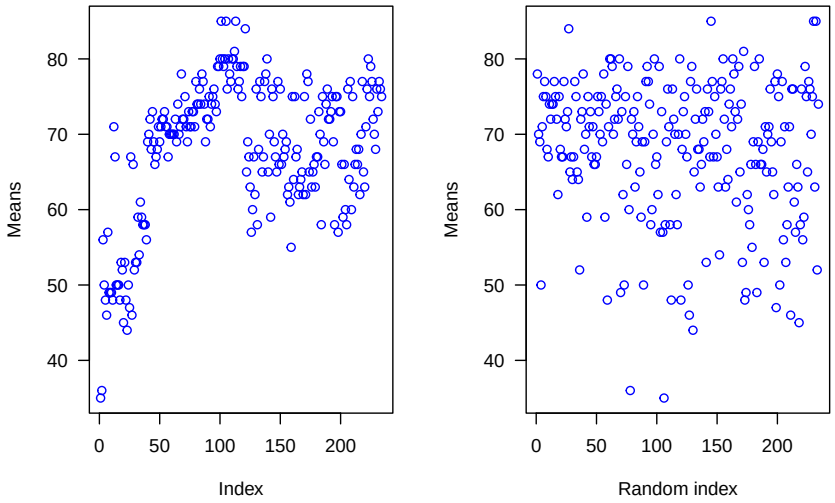
```
  las = 1,
  col = "red",
  labels = c("Ind 1", "Ind 2", "Ind 3", "Ind 4", "Ind 5"),
  xaxt = 'n',
  yaxt = 'n'
)
```



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]
```

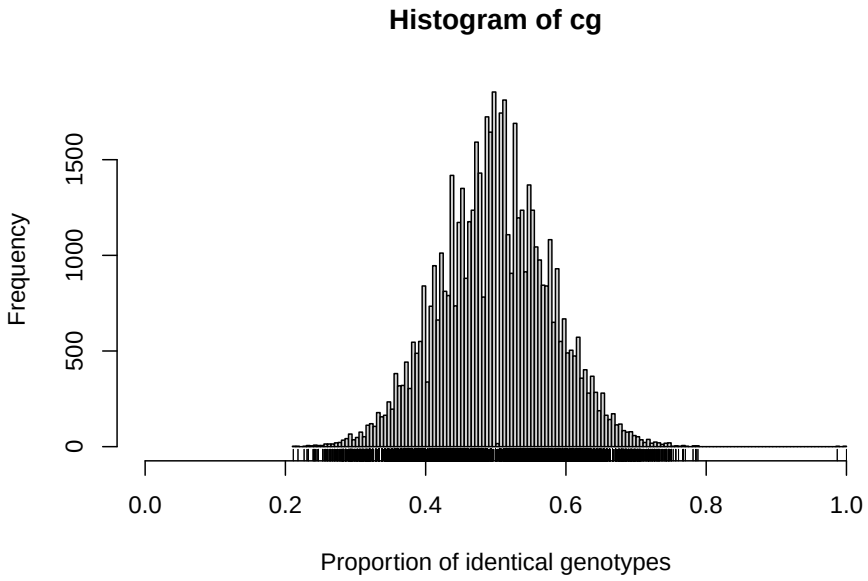
```
##       AA AB  BB not.BB not.AA
## c4m7  31  0 113      0      0
## c6m9  28  0 116      0      0
## c7m3  62  2  40      0      0
## c10m8 36  4  40      0      0
```

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
```

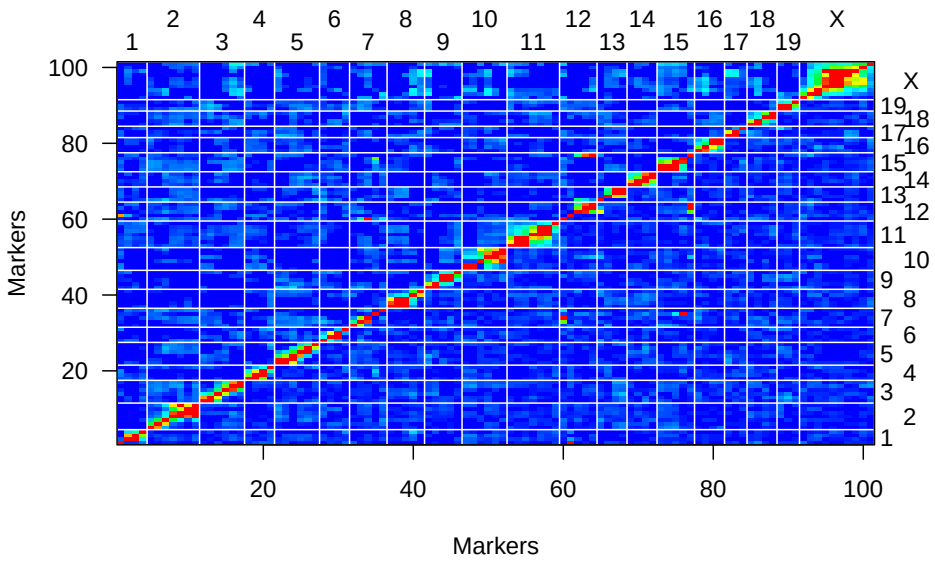
Now, let's recheck:

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

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

Now plot the pairwise recombination fractions and LOD scores:

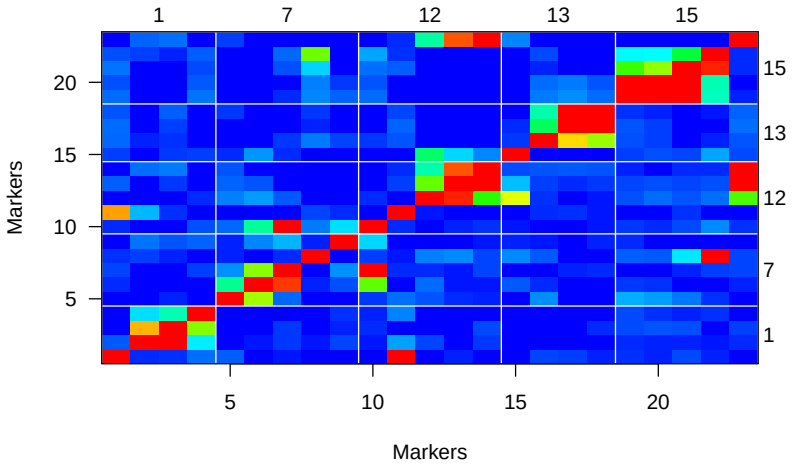
```
plotRF(
  ch3c,
  alternate.chrid = TRUE,
  main = "",
  col.scheme = "redblue"
)
```



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,
```

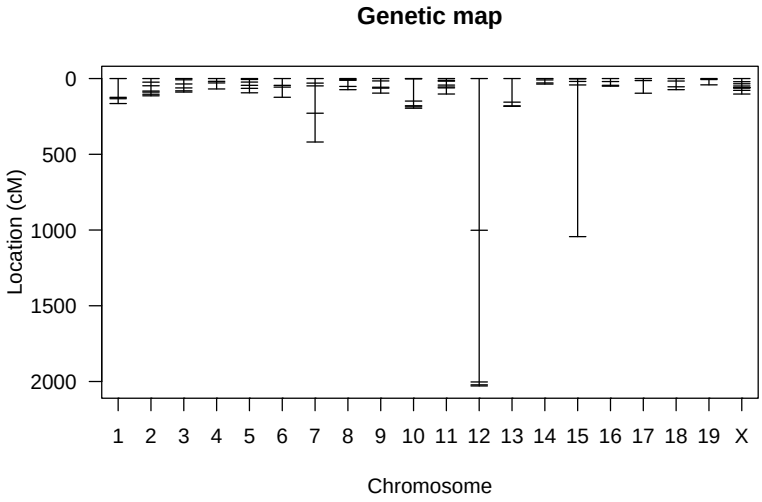
```
  chr = c(1, 7, 12, 13, 15),
  main = "",
  col.scheme = "redblue"
)
```



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)
```



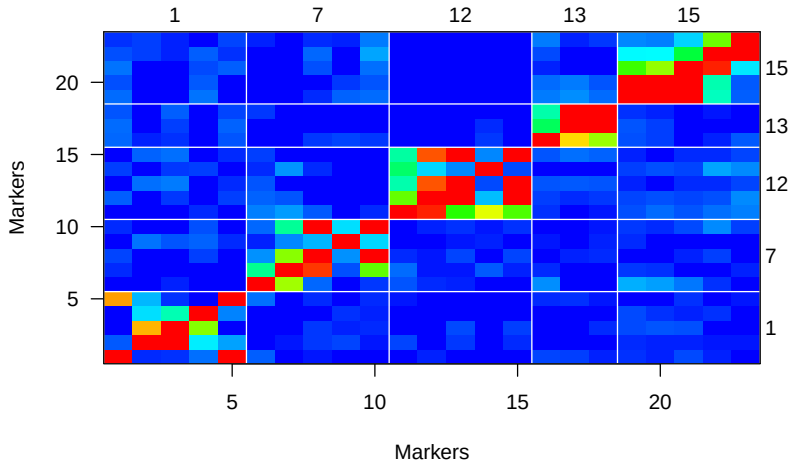
The estimated map indicates clear problems on chr 7, 12 and 15: enormous map expansion occurs as a result of markers that do not belong on those chromosomes.

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:

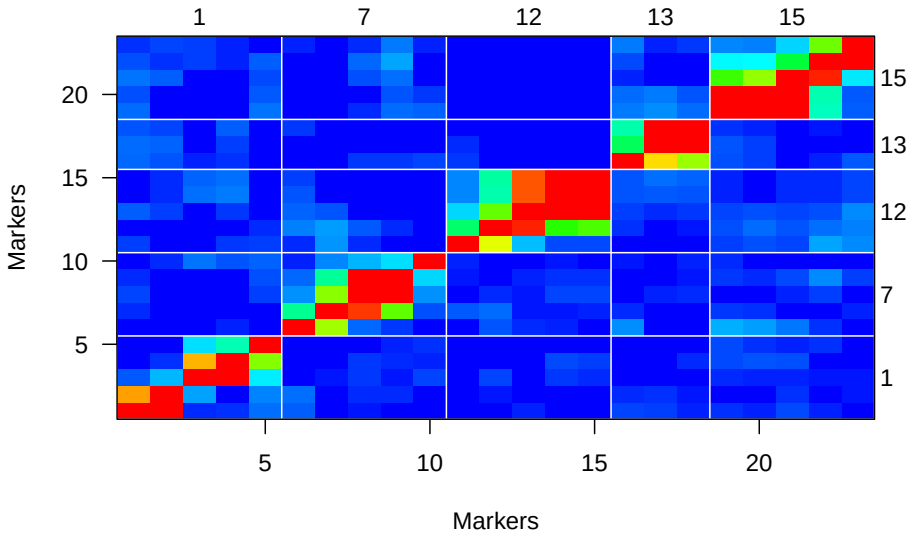
```
summary(rip[["X"]])
```

```
##                                  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"
)
```

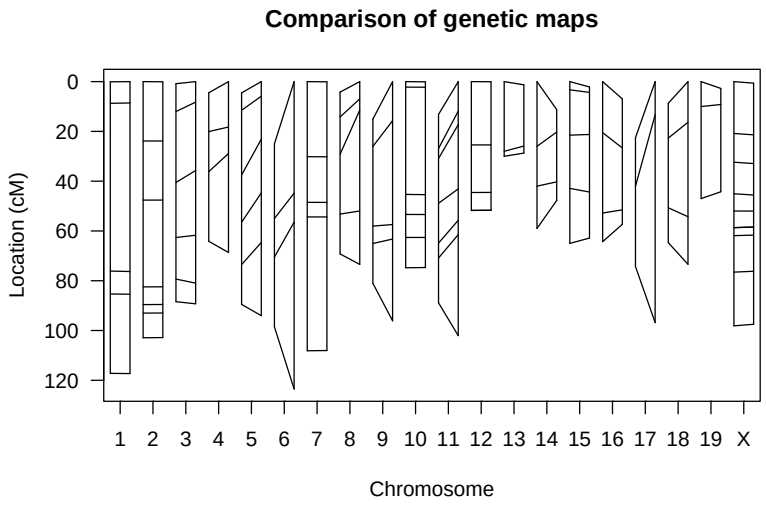


The results are just what we want: red along the diagonal, fading to blue off the diagonal.

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)
```

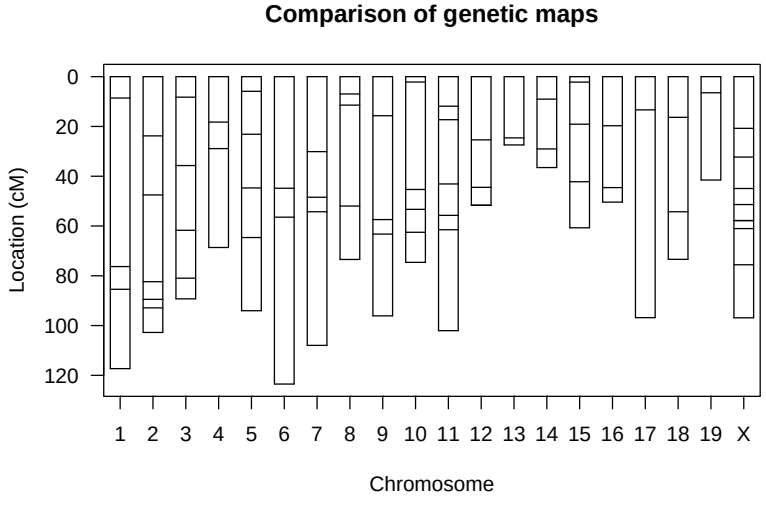


For many chromosomes, the estimated map is identical to the one within the dataset. Several chromosomes exhibit considerable map expansion (e.g., chromosome 6): the estimated map is quite a bit longer than the map in the data. This may indicate the presence of genotyping errors.

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

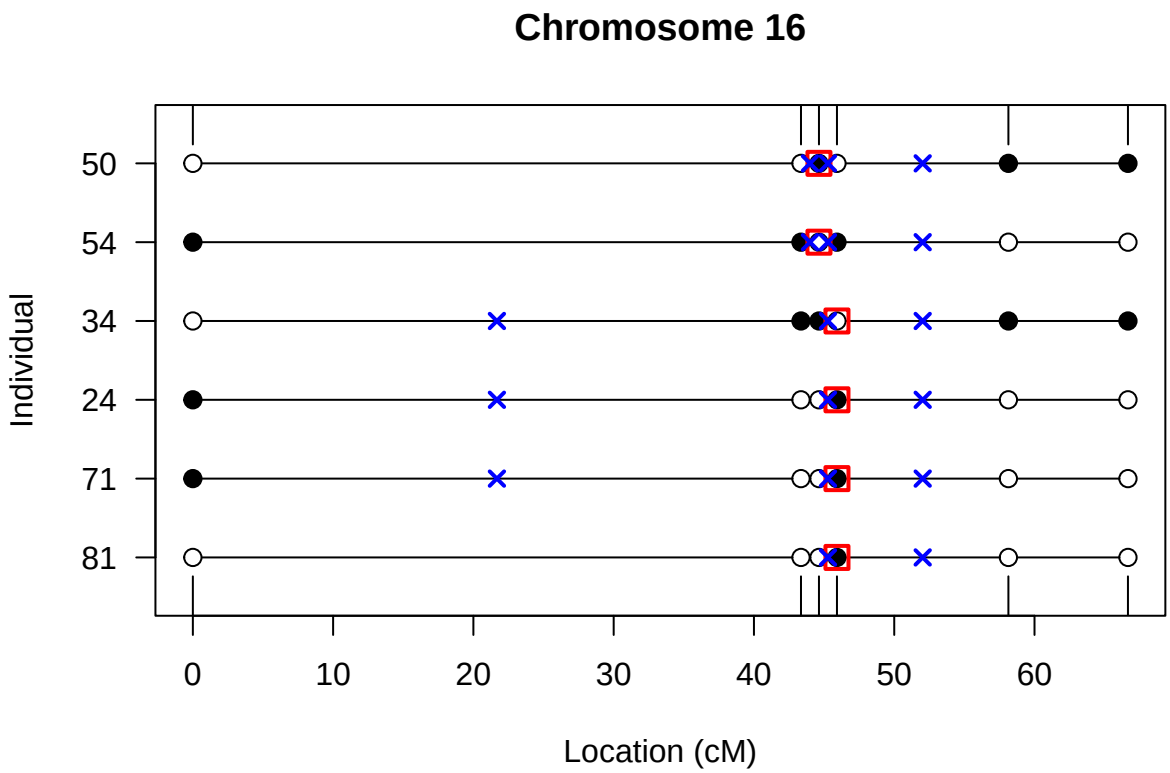
out of phase with the surrounding markers, it is likely in error. Let's calculate the genotyping error LOD scores, but first, we may first wish to replace the map in the data with that estimated from the data:

```
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)
```

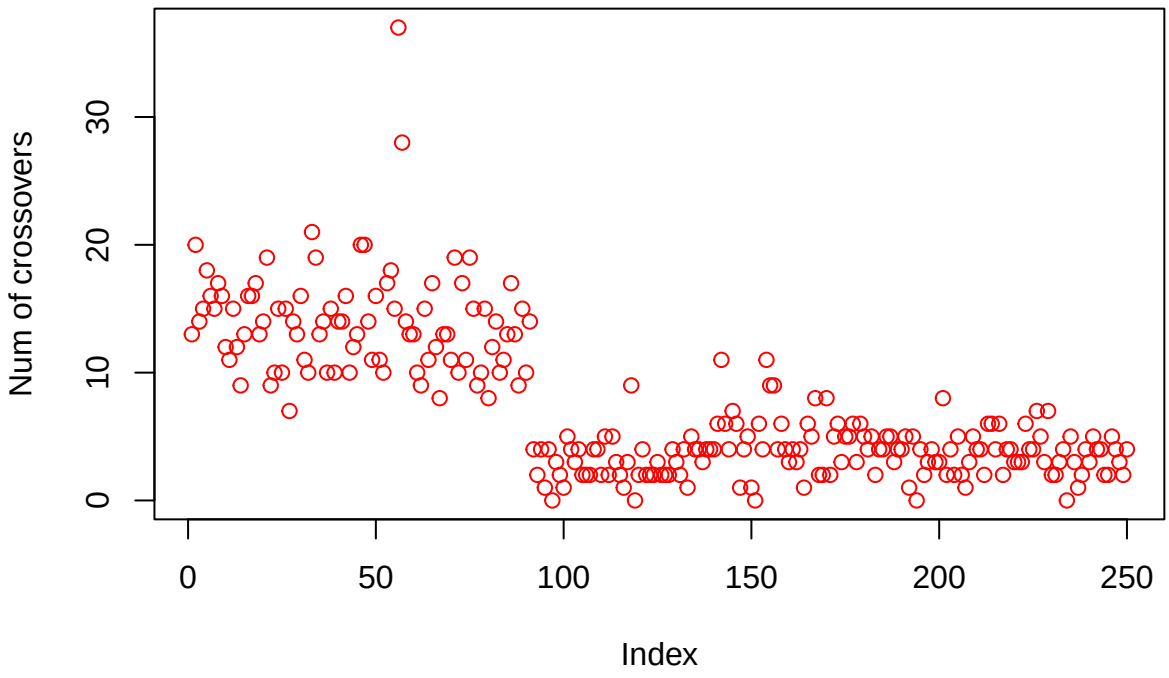


A small number of genotyping errors will not have much influence on the results.

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, ]
```

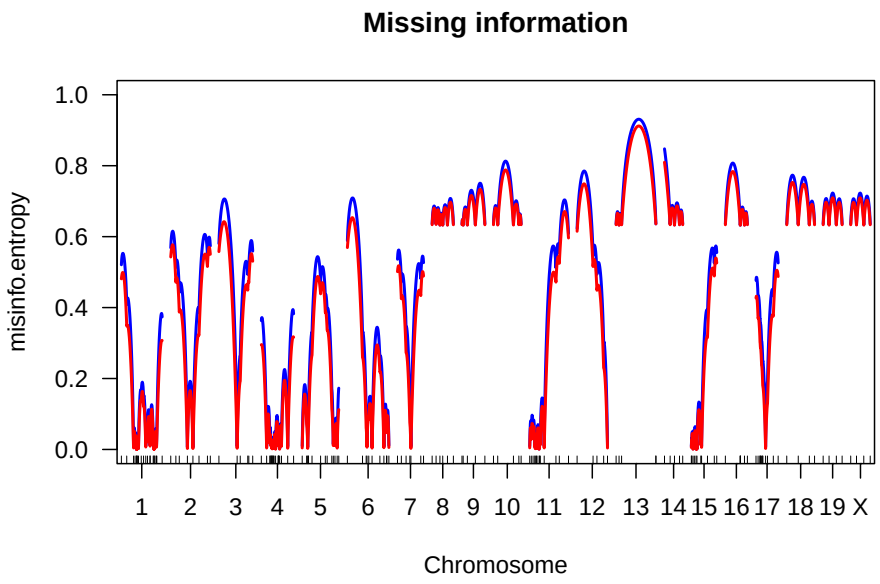
```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19  X
##  2  0  1  1  5  7  2  2  1  2  4  1  0  1  4  1  1  1  1  0
```

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:

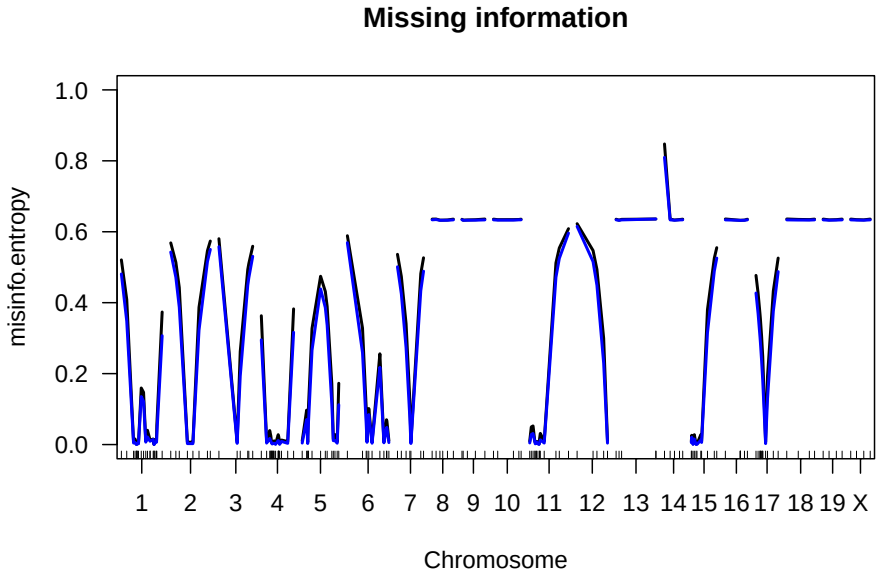
```
plotInfo(hyper, method = "both", col = c("blue", "red"))
```



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)
```

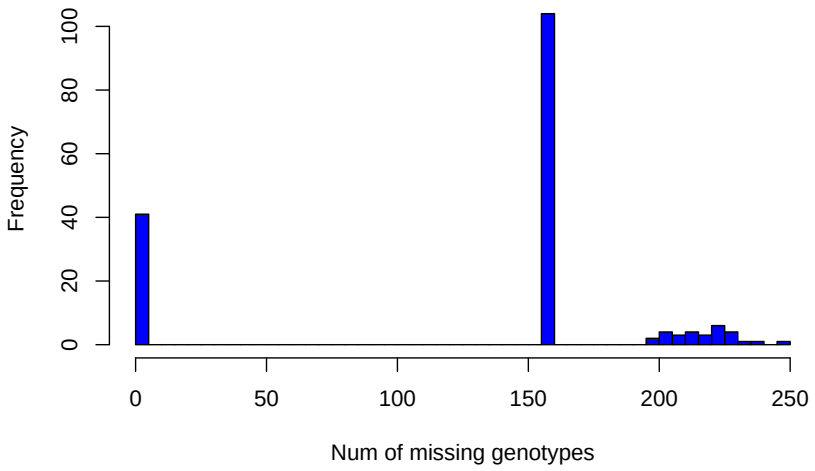



```
z[z[, 1] == 14, ]
```

```
##           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