Genetic map construction with R/qtl

Karl W. Broman

University of Wisconsin-Madison
Department of Biostatistics & Medical Informatics

Technical Report # 214

4 November 2010

Abstract: Genetic map construction remains an important prerequisite for quantitative trait loci analysis in organisms for which genomic sequence is not available. Surprisingly little has been written about the construction of genetic maps, particularly regarding the many practical issues involved. We provide an overview of the process, including a description of the key facilities in the R/qtl software. The process is illustrated through simulated data on an F_2 intercross derived from two inbred strains.

email: `kbroman@biostat.wisc.edu`

Introduction

Mouse geneticists no longer have to be concerned with genetic map construction, as the available genomic sequence provides the physical locations and order of genetic markers. Scientists interested in mapping quantitative trait loci (QTL) in other organisms, however, often must first spend considerable time identifying polymorphic markers and then constructing a genetic map specifying the chromosomal locations of the markers.

Surprisingly little has been written about the construction of genetic maps. Certainly, papers describing seminal genetic maps include some description of the methods used, but students desire a general overview of the process along with a discussion of key issues that arise and how to overcome them. I attempt to provide such an overview here, along with a description of the key facilities in R/qtl (Broman et al., *Bioinformatics* 18:889–890, 2003; see `http://www.rqtl.org`) for genetic map construction. While I describe the use of R/qtl for this purpose in considerable detail, I hope that my general comments are useful for, and that the R code is not an obstacle to, readers interested in the process generally or in other software.

R/qtl is principally aimed at the analysis of simple crosses (particularly backcrosses and intercrosses) derived from a pair of inbred strains. R/qtl includes the ability to analyze doubled haploids or a panel of recombinant inbred lines or even a phase-known four-way cross, but here I will focus on the simple case of an intercross between two inbred strains.

Experimental design issues

Before proceeding with our discussion of genetic map construction, let us first describe some of the design issues that a scientist should consider before embarking on such a project.

One must first ask: what sort of cross should be performed? As I have in mind a QTL mapping application, I would suggest using the same cross (that is, the same material) for both constructing a genetic map and for the subsequent QTL mapping. It is likely the case that one will need a larger cross for the QTL mapping than for map construction, and also one probably should use a larger set of markers for map construction than for QTL mapping. Thus the ideal strategy may be to produce a large cross, for QTL mapping, but genotype only a portion of that cross on the full set of markers. Once the genetic map is constructed, one may then identify a smaller set of markers, which cover the genome as evenly as possible, to genotype on the remainder of the cross.

In spite of the fact that map construction often requires fewer individuals than QTL mapping, as a general rule one should use more than the minimal number of individuals for map construction. Issues such as genotyping errors and segregation distortion are more easily overcome with a larger cross population.

Similarly, one should aim for a much larger set of genetic markers than simple calculations about genome size might indicate to be sufficient. In organisms with a mature complement of genomic resources, one may choose an ideal set of markers that evenly cover the genome and that are all well behaved and easily genotyped. In the *de novo* construction of a genetic map, however, markers will be developed at random, and so some regions will be densely covered with markers and other regions will be quite thin with markers. Variation in recombination rate (as well as in marker density and polymorphism) will exacerbate this issue. Moreover, one often finds that some markers are very difficult to genotype, and rather than spend considerable time optimizing such markers (such as by redesigning primers or changing PCR conditions), it would be most expedient if such poorly behaved markers could simply be omitted. If one has at hand more markers than are really necessary, it is less of a concern to be dropping 10% of them.

In the actual genotyping, one should always include DNA from the parental lines and F_1 individuals as controls, preferably on each plate. Ideally, include the *actual* parents and F_1 individuals; if residual heterozygosity is identified, the genotypes of these progenitors will be extremely useful for making sense of the data. Additional blind duplicates are useful to give some sense of the overall genotyping error rate.

Load the data

I will consider, as an illustration, a set of simulated data. Real data might be preferred, but by considering simulated data, I can be sure that they feature all of the various issues that I wish to illustrate. Moreover, these features can be made quite striking, so that there will be little question here of what to do. In practice, aberrant features in the data will often be less than clear and may require additional genotyping, or at least a reconsideration of the raw genotyping results, before the appropriate action is clear.

The data set is called `mapthis` and is included in R/qtl version 1.19-10 and above. This is an intercross between two inbred lines, with 300 individuals genotyped at 100 markers. There is just one "phenotype," which contains individual identifiers. The 100 markers were chosen at random from five chromosomes of length 200, 150, 100, 75 and 50 cM, respectively. The chromosomal locations of the markers are to be treated as unknown, though the marker names do contain this information: marker `C2M4` is the fourth marker on chromosome 2. (Note that the chromosomes are all autosomes. I'm not considering the X chromosome in this tutorial.)

To load the data, in R, one first needs to load the R/qtl package, using the function `library()`. The data set is then loaded via `data()`.

```
> library(qtl)
> data(mapthis)
```

In practice, one would use the function `read.cross()` to load a data set into R. See the help file (type `?read.cross`), look at the sample data sets at `http://www.rqtl.org/sampledata`, and consider Chapter 2 of Broman and Sen (2009) *A Guide to QTL Mapping with R/qtl* (`http://www.rqtl.org/book`).

In assembling the sort of comma-delimited file that R/qtl can read, one should assign all markers to the same chromosome (which can be named whatever is convenient, "`1`" or "`un`" or whatever). There is no need to assign map positions to markers.

A data file for the `mapthis` data is at `http://www.rqtl.org/tutorials/mapthis.csv`. It can be read into R as follows. (Note the use of `estimate.map=FALSE`. Markers will be assigned dummy locations, with no attempt to estimate inter-marker distances.)

```
> mapthis <- read.cross("csv", "http://www.rqtl.org/tutorials", "mapthis.csv",
+                       estimate.map=FALSE)
```

Omit individuals and markers with lots of missing data

After importing a new data set, the first thing I look at is a summary of the cross.

```
> summary(mapthis)
```

```
    F2 intercross

    No. individuals:    300

    No. phenotypes:     1
    Percent phenotyped: 100

    No. chromosomes:    1
        Autosomes:      1

    Total markers:      100
    No. markers:        100
    Percent genotyped:  95.4
    Genotypes (%):      AA:26.2  AB:48.2  BB:25.6  not BB:0  not AA:0
```

I look to make sure that the cross type (in this case, an F_2 intercross) was determined correctly and that the numbers of individuals and markers is as expected. The function `summary.cross()` also performs a variety of basic checks of the integrity of the data, and so I'd pay attention to any warning or error messages.

Next, I look at the pattern of missing data, through the function `plot.missing()`.

```
> plot.missing(mapthis)
```

The result (in Fig. 1) indicates several individuals with a great deal of missing data (horizontal lines), as well as several markers with a great deal of missing data (vertical lines).

The function `ntyped()` provides the numbers of genotyped markers for each individual (or the number of genotyped individuals for each marker). Let us plot these. (And note that there is a related function, `nmissing()`, which provides the number of missing genotypes for each individual or marker.)

```
> par(mfrow=c(1,2), las=1)
> plot(ntyped(mapthis), ylab="No. typed markers", main="No. genotypes by individual")
> plot(ntyped(mapthis, "mar"), ylab="No. typed individuals",
+      main="No. genotypes by marker")
```

As seen in Fig. 2, there are six individuals missing almost all genotypes, and there are four markers that are missing genotypes at about half of the individuals. Such appreciable missing data often indicates a problem (either bad DNAs or difficult-to-call markers) and can lead to difficulties in the genetic map construction, and so it is best, at this stage, to omit them, though one might consider adding them back in later.

To omit the individuals with lots of missing genotype data, we may use the function `subset.cross()`, as follows.

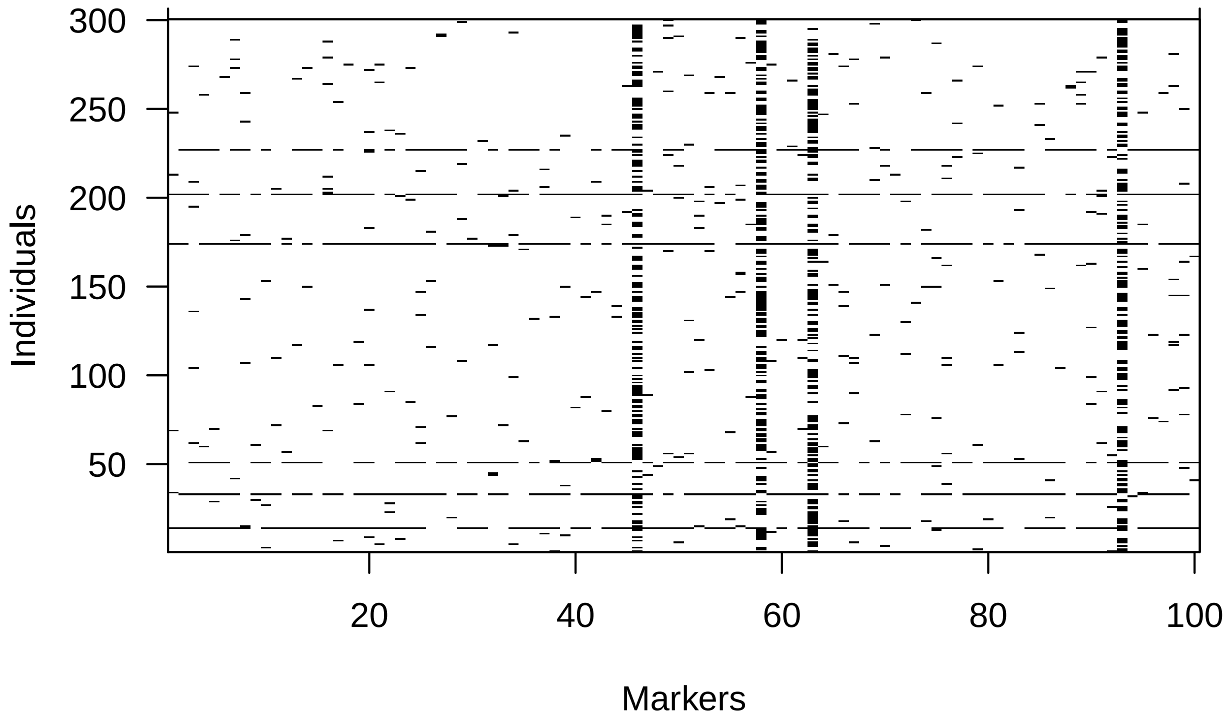

Figure 1: Pattern of missing genotype data in the `mapthis` dataset. Black pixels indicate missing genotypes.

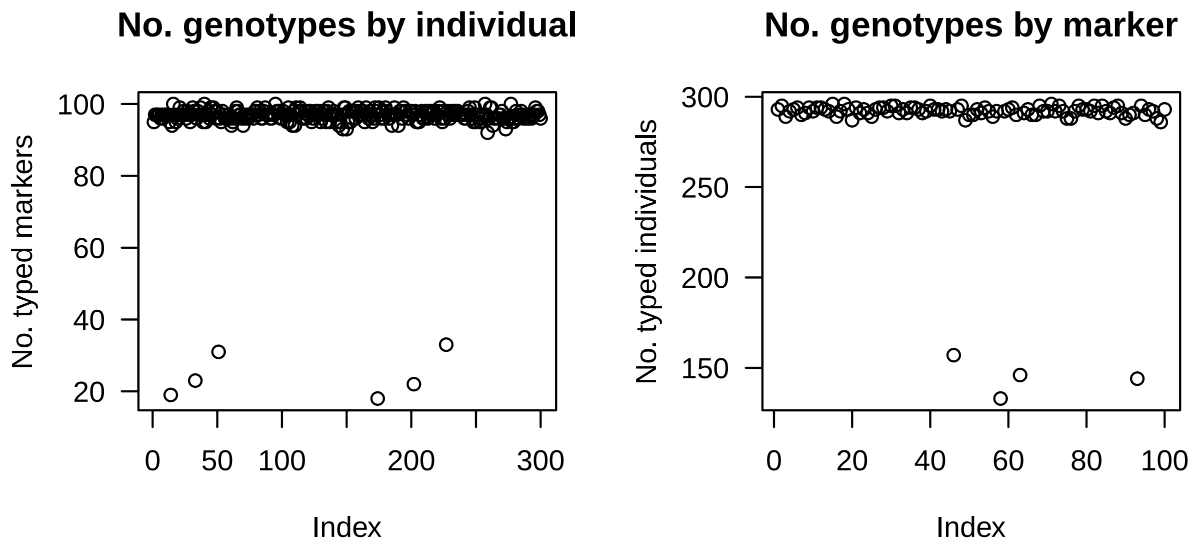

Figure 2: Plot of number of genotyped markers for each individual (left panel) and number of genotyped individuals for each marker (right panel).

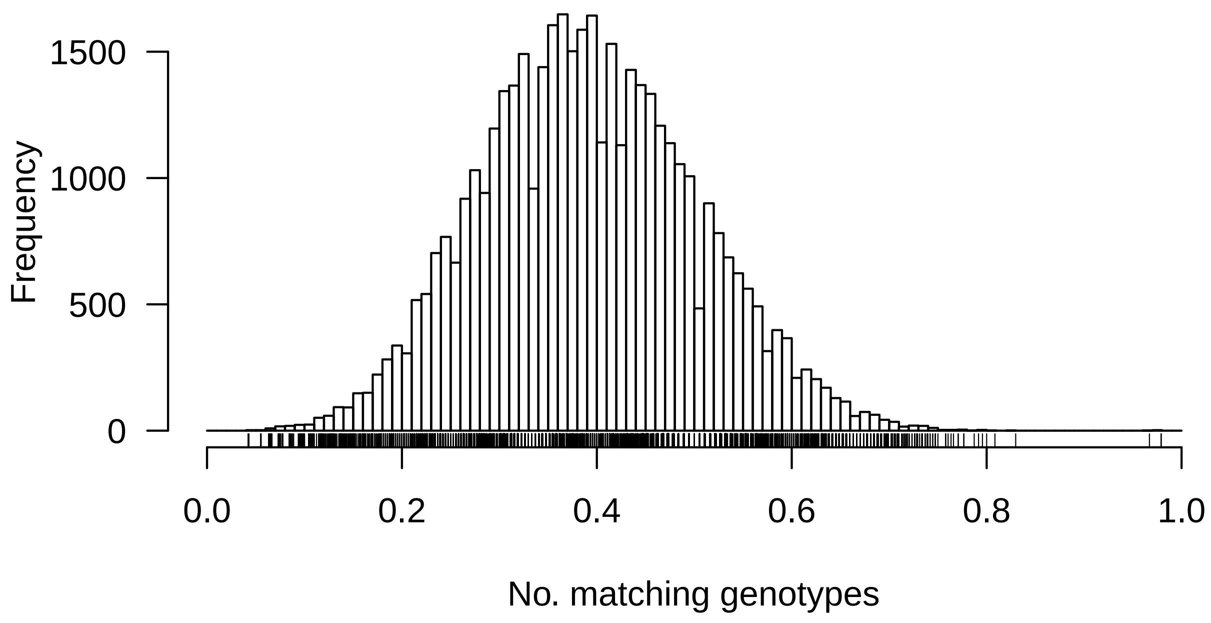

Figure 3: Histogram of the proportion of markers for which pairs of individuals have matching genotypes.

```
> mapthis <- subset(mapthis, ind=(ntyped(mapthis)>50))
```

To omit the markers with lots of missing data, we first need to identify the names of the markers. We then use `drop.markers()`.

```
> nt.bymar <- ntyped(mapthis, "mar")
> todrop <- names(nt.bymar[nt.bymar < 200])
> mapthis <- drop.markers(mapthis, todrop)
```

Identify duplicate individuals

I find it useful to compare the genotypes between all pairs of individuals, in order to reveal pairs with unusually similar genotypes, which may indicate sample duplications or monozygotic twins. In either case, we will want to remove one individual from each pair. Such duplicates are not common, but they are also not rare.

We use the function `comparegeno` to compare the genotypes for all pairs of individuals. The output is a matrix, whose contents are the proportions of markers at which the pairs have matching genotypes.

```
> cg <- comparegeno(mapthis)
> hist(cg[lower.tri(cg)], breaks=seq(0, 1, len=101), xlab="No. matching genotypes")
> rug(cg[lower.tri(cg)])
```

As seen in Fig. 3, a pair of F_2 siblings typically share genotypes at 40% of the markers. But there are some pairs with well over 90% matching genotypes. We may identify these pairs as follows.

```
> wh <- which(cg > 0.9, arr=TRUE)
> wh <- wh[wh[,1] < wh[,2],]
> wh

     row col
[1,] 214 216
[2,] 238 288
[3,] 144 292
```

We may inspect the genotype matches for these pairs with the following.

```
> g <- pull.geno(mapthis)
> table(g[144,], g[292,])

     1  2  3
  1 26  0  0
  2  2 44  0
  3  0  1 18

> table(g[214,], g[216,])

     1  2  3
  1 21  0  0
  2  0 35  1
  3  1  0 37

> table(g[238,], g[288,])

     1  2  3
  1 36  1  0
  2  0 46  0
  3  1  0 12
```

As seen above, in each case, the pairs have matching genotypes at all but 2 or 3 markers. Ideally, one would go back to the records to try to assess the source of these problems (e.g., are the pairs from the same litters?). Here, we will simply omit one individual from each pair. But first we will omit the genotypes that mismatch, as these are indicated to be errors in one or the other individual (or both). The R code is a bit complicated.

```
> for(i in 1:nrow(wh)) {
+   tozero <- !is.na(g[wh[i,1],]) & !is.na(g[wh[i,2],]) & g[wh[i,1],] != g[wh[i,2],]
+   mapthis$geno[[1]]$data[wh[i,1],tozero] <- NA
+ }
```

Now, we omit one individual from each pair.

```
> mapthis <- subset(mapthis, ind=-wh[,2])
```

It's also useful to look for duplicate markers (that is, markers with identical genotypes). This is particularly true for the case of very large sets of markers; multiple markers with identical genotypes will invariably map to the same location, and so one might as well thin out the markers so that there are no such duplicates, as the extra markers simply slow down all of our analyses. The function `findDupMarkers()` may be used to identify markers with matching genotypes. (Note that the function `drop.dupmarkers()` is for dropping markers with matching *names*, and considers the genotypes only in order to establish consensus genotypes across the multiple markers with the same name). Here, though, there are no markers with matching genotypes.

```
> print(dup <- findDupMarkers(mapthis, exact.only=FALSE))

NULL
```

Look for markers with distorted segregation patterns

We next study the segregation patterns of the markers. We expect the genotypes to appear with the frequencies 1:2:1. Moderate departures from these frequencies are not unusual and may indicate the presence of partially lethal alleles. Gross departures from these frequencies often indicate problematic markers that should be omitted, at least initially: monomorphic markers (that is, where the two parental lines actually have the same allele), or markers that are especially difficult to call (e.g., AA are often called as AB). We use the function `geno.table` to inspect the segregation patterns. It calculates the genotype frequencies and also a P-value for a test of departure from the 1:2:1 expected ratios. We will focus on those markers that show significant distortion at the 5% level, after a Bonferroni correction for the multiple tests.

```
> gt <- geno.table(mapthis)
> gt[gt$P.value < 0.05/totmar(mapthis),]

      chr missing  AA  AB  BB not.BB not.AA   P.value
C4M2    1       1  97 143  50      0      0  4.79e-04
C1M4    1       4   8 207  72      0      0  3.97e-19
C2M9    1       2 284   3   2      0      0 2.08e-180
C1M21   1       5 196  10  80      0      0  7.00e-75
C2M15   1       3   0   1 287      0      0 1.31e-186
C2M27   1       5   2 214  70      0      0  4.66e-23
```

The first of these markers, `C4M2`, is not terrible. The others appear to be monomorphic with a few errors (`C2M9` and `C2M15`) or have one genotype that is quite rare, which likely indicates difficulties in genotyping. It would be best to omit the worst of these markers.

```
> todrop <- rownames(gt[gt$P.value < 1e-10,])
> mapthis <- drop.markers(mapthis, todrop)
```

Study individuals' genotype frequencies

Just as we expect the markers to segregate 1:2:1, we expect the individuals to have genotype frequencies that are in approximately those proportions. Studying such genotype frequencies may help to identify individuals with high genotyping error rates or some other labeling or breeding mistake.

There's no R/qtl function to to pull out the genotype frequencies by individual, but we can write a bit of R code to do so. It is a bit nasty, with calls to `apply()`, `table()`, `factor()`, `colSums()` and `t()`, but hopefully the reader can figure this out after a bit of exploration and introspection.

```
> g <- pull.geno(mapthis)
> gfreq <- apply(g, 1, function(a) table(factor(a, levels=1:3)))
> gfreq <- t(t(gfreq) / colSums(gfreq))
> par(mfrow=c(1,3), las=1)
> for(i in 1:3)
+   plot(gfreq[i,], ylab="Genotype frequency", main=c("AA", "AB", "BB")[i],
+        ylim=c(0,1))
```

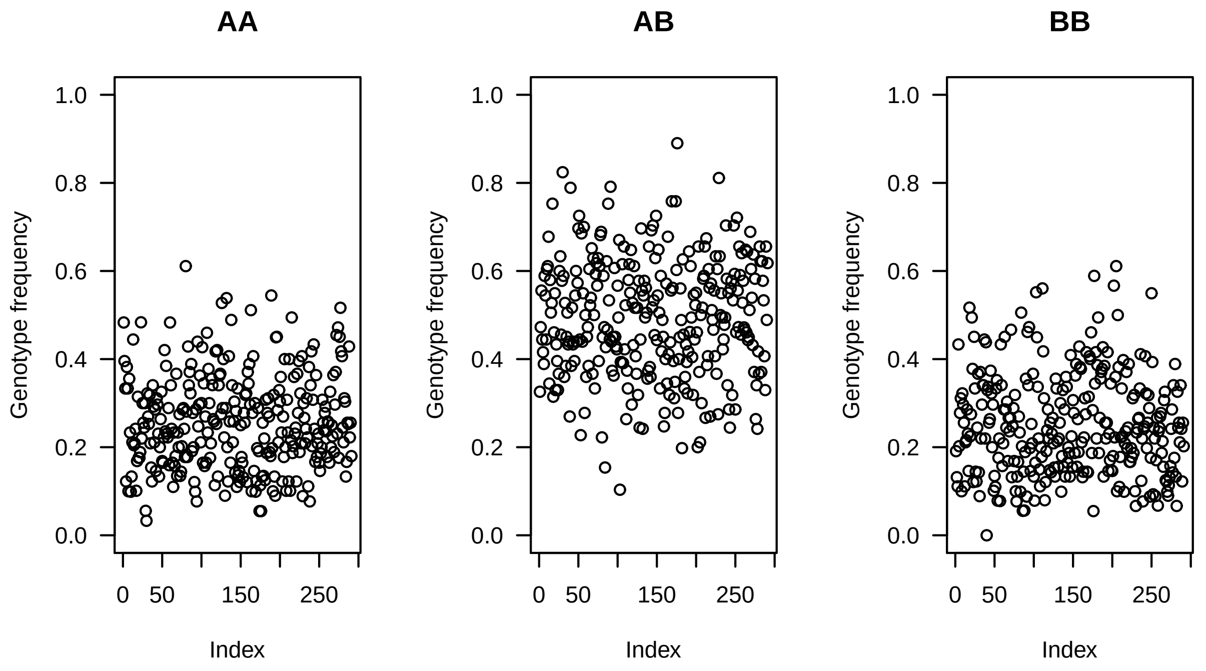

Figure 4: Genotype frequencies by individual.

The results in Fig. 4 do not indicate any particular problems, though the small number of short chromosomes result in considerable variability, including one individual with no BB genotypes, and the frequencies of AB genotypes varies from 10–89%. However, if there were an individual with ∼ 100% AA or BB genotypes (like one of the parental strains), we would see it here.

A more fancy, and potentially more clear view of these genotype frequencies is obtained by representing them as points in an equilateral triangle (see Fig. 5). For any point within an equilateral triangle, the sum of the distances to the three sides is constant, and so one may represent a trinomial distribution as a point within the triangle. I won't show the code, but note that the red X in the center of the figure corresponds to the expected genotype frequencies (1/4, 1/2, 1/4). Each black dot corresponds to an individual's genotype frequencies. There's one point along the right edge; this corresponds to the individual with no BB genotypes. Again, the small size of the genome results in enormous variation, and so no one individual stands out as unreasonable.

Study pairwise marker linkages; look for switched alleles

We are now in a position to begin the genetic map construction. We start by assessing the linkage between all pairs of markers. The function `est.rf()` is used to estimate the recombination fraction between each pair and to calculate a LOD score for a test of rf = 1/2. But first note that, in the presence of appreciable segregation distortion (which is not the case for these data), unlinked markers can appear to be linked. For example, consider

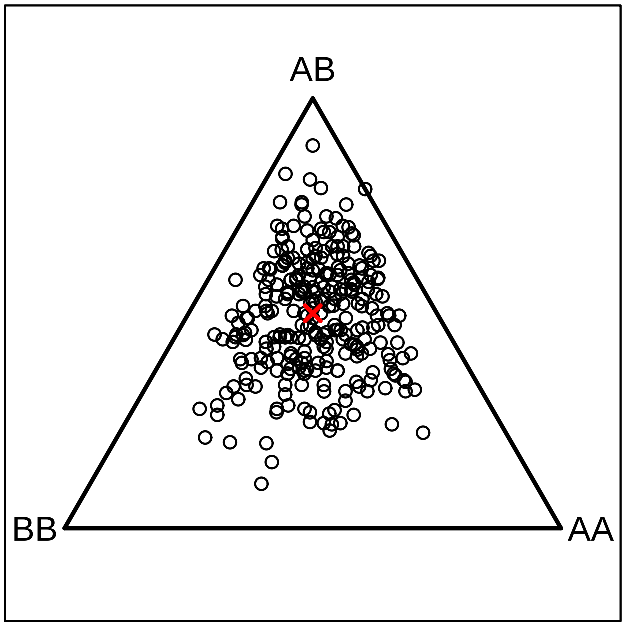

Figure 5: Genotype frequencies by individual, represented on an equilateral triangle. The red X indicates the expected frequencies of 1:2:1.

the following 2×2 table of two-locus genotypes in a backcross.

	Marker 2		
Marker 1	AA	AB	overall
AA	243	27	270
AB	27	3	30
overall	270	30	300

In this case, the two markers are segregating independently but show severe segregation distortion. (They segregate 9:1 rather than 1:1.) The usual estimate of the recombination fraction is $(27+27)/300 = 0.18$, and the LOD score for the test of rf $= 1/2$ is ~ 28.9.

If segregation distortion is rampant in a dataset (and such things do happen), the usual tests of pairwise linkage will give distorted results, and so it is best to instead use a simple chi-square or likelihood ratio test, to assess the association between markers. The function `markerlrt()` behaves just like `est.rf()`, but uses a general likelihood ratio test in place of the usual test of pairwise linkage.

Now back to the data, which are not so badly behaved.

```
> mapthis <- est.rf(mapthis)
```

```
Warning message:
In est.rf(mapthis) : Alleles potentially switched at markers
  C3M16 C2M16 C1M2 C3M9 C2M14 C1M24 C1M1 C2M12 C1M36 C3M1 C2M25 C1M22 C5M5 C5M7 C1M17 C5M1
  C3M5 C1M15 C2M24 C2M17 C1M23 C5M6 C1M16 C3M2 C3M10 C3M6 C2M13
```

Note the warning message, which indicates that there are numerous markers with likely switched alleles (A $\leftrightarrow$ B). This is indicated through pairs of markers that are strongly indicated to be associated but have estimated recombination fractions $\gg 1/2$. The `checkAlleles()` function gives more detailed information on this issue.

```
> checkAlleles(mapthis, threshold=5)
```

```
   marker chr index diff.in.max.LOD
4   C3M16   1     4           31.19
5   C2M16   1     5           55.02
8    C1M2   1     8            5.52
9    C3M9   1     9           39.57
12  C2M14   1    12           32.54
17  C1M24   1    17            5.76
18   C1M1   1    18          102.00
19  C2M12   1    19            6.22
35  C1M36   1    35           92.99
38   C3M1   1    38           49.19
41  C2M25   1    41           38.53
42  C1M22   1    42            5.77
45   C5M5   1    45           17.34
49   C5M7   1    49           24.09
51  C1M17   1    51            9.80
54   C5M1   1    54           25.76
55   C3M5   1    55           53.61
58  C1M15   1    58           21.98
60  C2M24   1    60          104.63
65  C2M17   1    65           43.55
69  C1M23   1    69           88.31
70   C5M6   1    70           15.60
76  C1M16   1    76           86.79
83   C3M2   1    83           24.93
84  C3M10   1    84           48.88
89   C3M6   1    89           10.65
91  C2M13   1    91           44.01
```

The final column in the output for a marker, `diff.in.max.LOD`, is the difference between the maximum LOD score for the cases where the estimated recombination fraction is $> 1/2$ and the maximum LOD score for the cases where the estimated recombination fraction is $< 1/2$. There are a large number of markers that are tightly associated with other markers, but with recombination fractions well above 1/2, which indicates that some markers likely have their alleles switched.

A plot of the LOD scores against the estimated recombination fractions for all marker pairs will give another good view of this problem. We use the function `pull.rf()` to pull out the recombination fractions and LOD scores as matrices.

```
> rf <- pull.rf(mapthis)
> lod <- pull.rf(mapthis, what="lod")
> plot(as.numeric(rf), as.numeric(lod), xlab="Recombination fraction", ylab="LOD score")
```

As seen in Fig. 6, there are many marker pairs with large LOD scores but estimated recombination fractions $\gg 1/2$.

One solution to this problem is to form initial linkage groups, ensuring that markers with rf $> 1/2$ are placed in different groups. If all goes well, each chromosome will come out as a pair of linkage groups: one containing markers with correct alleles and another containing markers with switched alleles.

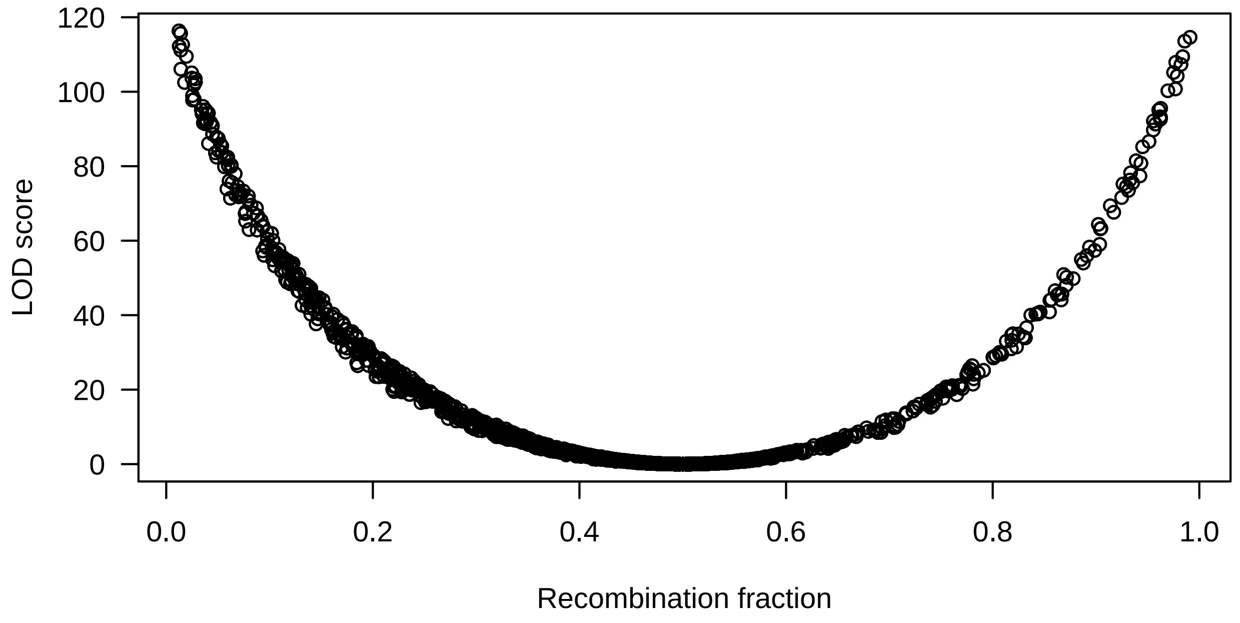

Figure 6: Plot of LOD scores versus estimated recombination fractions for all marker pairs.

We use the function `formLinkageGroups()` to infer linkage groups. It uses the pairwise linkage results from `est.rf()` (or, pairwise association information from `markerlrt()`). The function has two main arguments: `max.rf` and `min.lod`. Two markers will be placed in the same linkage groups if they have estimated recombination fraction $\leq$ `max.rf` and LOD score $\geq$ `min.lod`. The linkage groups are closed via the transitive property. That is, if markers a and b are linked and markers b and c are linked, then all three are placed in the same linkage group. I generally start with `min.lod` relatively large (say 6 or 8 or even 10). The appropriate value depends on the number of markers and chromosomes (and individuals). The aim is to get as many truly linked markers together, but avoid completely putting unlinked markers in the same linkage group. It is usually easier to combine linkage groups after the fact rather than to have to split linkage groups apart.

```
> lg <- formLinkageGroups(mapthis, max.rf=0.35, min.lod=6)
> table(lg[,2])
```

```
 1  2  3  4  5  6  7  8  9 10 11
29 21 10  9  6  5  4  3  2  1  1
```

The output of `formLinkageGroups()` is a matrix with two columns: the initial linkage group or chromosome for each marker, and then the assigned linkage group, as inferred from the pairwise linkage information. The inferred linkage groups are numbered in decreasing order of size (so that linkage group 1 has the largest number of markers). Here we see that 11 linkage groups were inferred, with the last 2 groups having just one marker. One may play with `max.rf` and `min.lod` until the result is about as expected. Here, I was hoping for about 10 linkage groups (since there are five chromosomes but each will likely be split in two due to the allele switching problem), and so the results seem okay.

Since we are happy with the results, we can reorganize the markers into these inferred linkage groups. We do so with the same function, `formLinkageGroups()`, via the argument `reorgMarkers`.

```
> mapthis <- formLinkageGroups(mapthis, max.rf=0.35, min.lod=6, reorgMarkers=TRUE)
```

A plot of the pairwise recombination fractions and LOD scores may indicate how well this worked.

```
> plot.rf(mapthis, alternate.chrid=TRUE)
```

The result, in Fig. 7, looks about as expected. The markers within linkage group 1 are linked to each other. The pattern within the group is a bit random, but that is because we have not yet ordered the markers in any way.

Note that linkage groups 4 and 5 are associated with each other, in that they have large LOD scores (the lower-right rectangle for groups 4 and 5 is largely red), but their recombination fractions are not small (the upper-left rectangle for groups 4 and 5 is strongly blue). I would infer from this that these markers belong to the same chromosome, but the alleles in one or the other group are switched. We can't know which is correct, and ideally one would have parental genotype data to help fix these problems, but it is not unreasonable, for now, to assume that the larger group of markers corresponds to the ones with the correct alleles.

Similarly, groups 6 and 7 belong together, group 8 belongs to group 2, and groups 9–11 belong to group 1.

We can look at this more clearly by picking out a marker from one group and studying the recombination fractions and LOD scores for that marker against all others. Let us pick the third marker in linkage group 4. We create the objects `rf` and `lod` again, using `pull.rf()`. These objects have class `"rfmatrix"`, and so `plot` below actually goes to the function `plot.rfmatrix()`, which plots the values for a single marker in a display similar to a set of LOD curves.

```
> rf <- pull.rf(mapthis)
> lod <- pull.rf(mapthis, what="lod")
> mn4 <- markernames(mapthis, chr=4)
> par(mfrow=c(2,1))
> plot(rf, mn4[3], bandcol="gray70", ylim=c(0,1), alternate.chrid=TRUE)
> abline(h=0.5, lty=2)
> plot(lod, mn4[3], bandcol="gray70", alternate.chrid=TRUE)
```

As seen in Fig. 8, the marker `C3M13` is strongly associated with markers in both linkage groups 4 and 5, but the estimated recombination fractions between `C3M13` and the markers on linkage group 4 are all $< 1/2$, while the estimated recombination fractions between `C3M13` and the markers on linkage group 5 are all $> 1/2$. We can see the problem even more clearly by inspecting a few tables of two-locus genotypes, produced by `geno.crosstab()`.

```
> geno.crosstab(mapthis, mn4[3], mn4[1])
```

```
     C3M11
C3M13 - AA  AB BB
   -  0  2   3  2
   AA 3 50   9  0
   AB 1 15 101 16
   BB 1  1  13 74
```

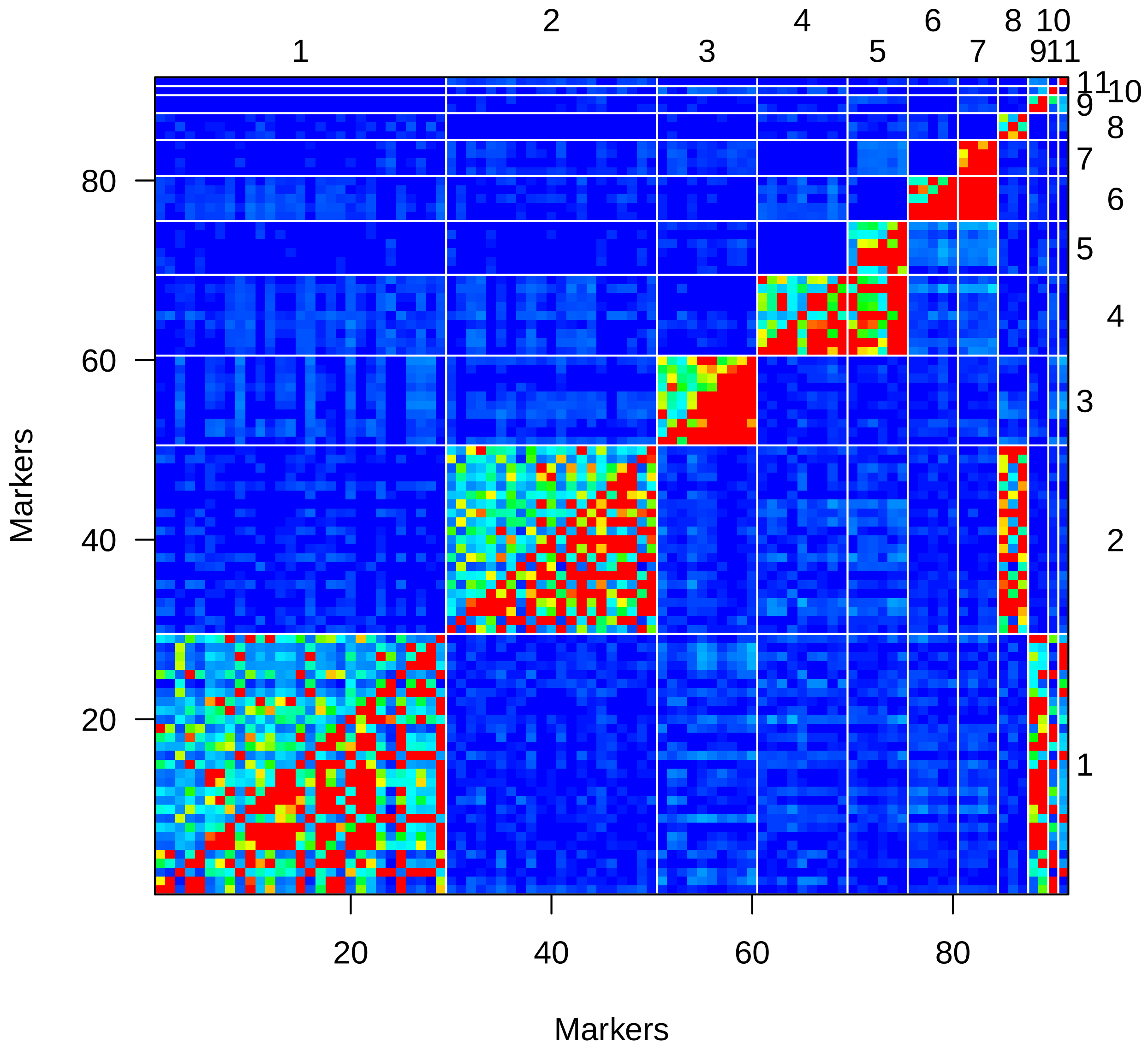

Figure 7: Plot of estimated recombination fractions (upper-left triangle) and LOD scores (lower-right triangle) for all pairs of markers. Red indicates linked (large LOD score or small recombination fraction) and blue indicates not linked (small LOD score or large recombination fraction).

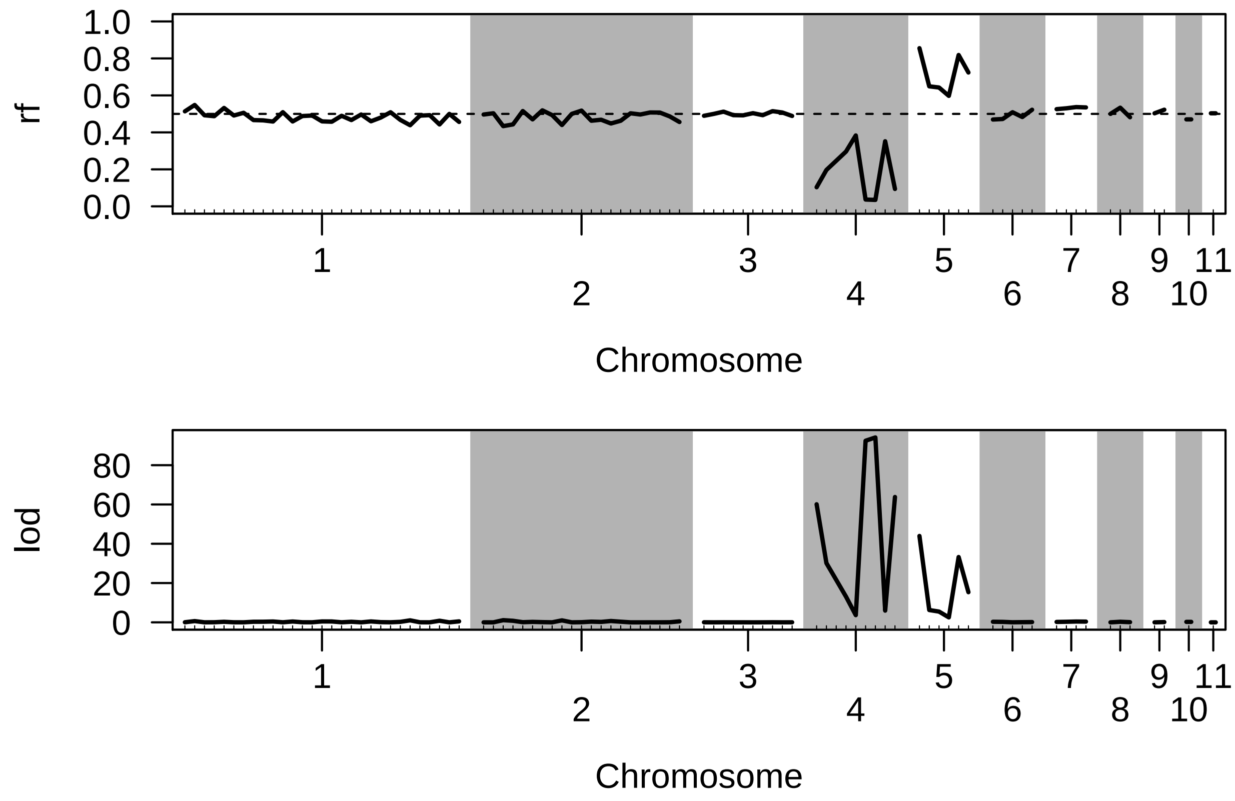

Figure 8: Plot of estimated recombination fractions (top panel) and LOD scores (bottom panel) for the marker `C3M13` against all other markers.

```
> mn5 <- markernames(mapthis, chr=5)
> geno.crosstab(mapthis, mn4[3], mn5[1])
```

```
     C3M16
C3M13 - AA  AB BB
   -  0  3   3  1
   AA 1  0  17 44
   AB 0 16 100 17
   BB 1 62  26  0
```

It is clear that, if the genotypes for `C3M13` are correct, then the A and B alleles for `C3M16` are switched.

In practice, I will look at enormous numbers of these sorts of two-locus genotype tables in order to figure out what is going on. But these data are (by design) quite clean, and so we will simply trust that the alleles need to be switched for all of the markers on linkage groups 5 and 7–11. The function `switchAlleles()` is convenient for performing the allele switching.

```
> toswitch <- markernames(mapthis, chr=c(5, 7:11))
> mapthis <- switchAlleles(mapthis, toswitch)
```

Now when we revisit the plot of recombination fractions and LOD scores, we will see a quite different picture. (Note that we need to re-run `est.rf()` after having run `switchAlleles()`.)

```
> mapthis <- est.rf(mapthis)
> plot.rf(mapthis, alternate.chrid=TRUE)
```

As seen in Fig. 9, the LOD scores between marker pairs are unchanged, but now the recombination fractions are small (indicated in red) for marker pairs with evidence of association (large LOD scores, also in red).

It is useful to revisit the plot of LOD scores versus recombination fractions for all pairs.

```
> rf <- pull.rf(mapthis)
> lod <- pull.rf(mapthis, what="lod")
> plot(as.numeric(rf), as.numeric(lod), xlab="Recombination fraction", ylab="LOD score")
```

Now we see (in Fig. 10) no estimated recombination fractions that are much above $1/2$, and certainly no large recombination fractions with large LOD scores. In fact, the largest LOD score for marker pairs with estimated recombination fraction $> 1/2$ is 1.38.

Form linkage groups

We now should have the genotype data in good shape and can finally proceed to the actual map construction. First, we again attempt to infer the linkage groups, with the hope that we'll come away with exactly five.

```
> lg <- formLinkageGroups(mapthis, max.rf=0.35, min.lod=6)
> table(lg[,2])
```

```
 1  2  3  4  5 
33 24 15 10  9 
```

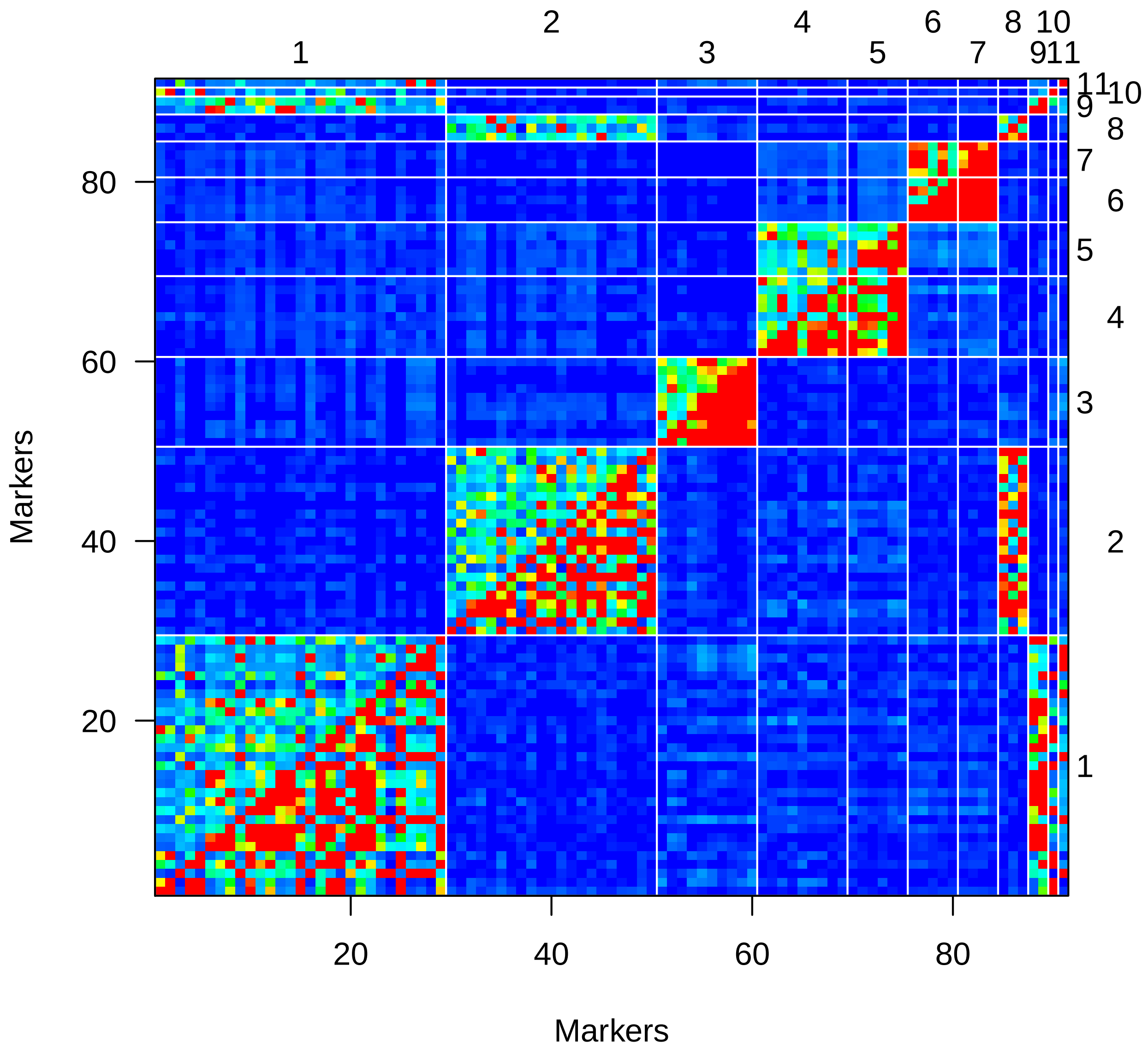

Figure 9: Plot of estimated recombination fractions (upper-left triangle) and LOD scores (lower-right triangle) for all pairs of markers, after having switched alleles for many markers. Red indicates linked (large LOD score or small recombination fraction) and blue indicates not linked (small LOD score or large recombination fraction).

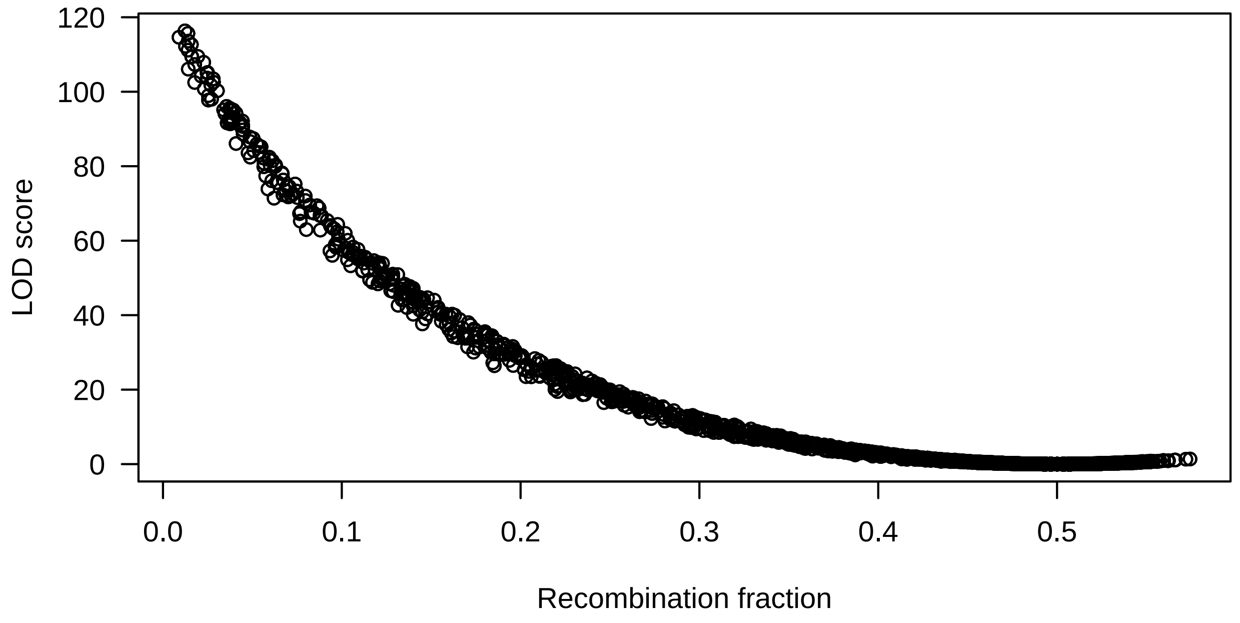

Figure 10: Plot of LOD scores versus estimated recombination fractions for all marker pairs, after alleles at many markers have been switched.

Right; we've got five groups. Now we reorganize the markers again.

```
> mapthis <- formLinkageGroups(mapthis, max.rf=0.35, min.lod=6, reorgMarkers=TRUE)
```

It is useful to plot the pairwise recombination fractions and LOD scores again.

```
> plot.rf(mapthis)
```

As seen in Fig. 11, we have five clear linkage groups, with markers within a group linked to one another, and markers in distinct groups showing no evidence of linkage. The results couldn't possibly be cleaner (and they wouldn't be, if these were real data).

Order markers on chromosome 5

We now turn to the problem of ordering markers within each linkage group. (We could go ahead and call them chromosomes at this point.) I always start with the chromosome with the fewest markers, as there are fewer possible orders and so the process is quicker. I like to see some progress before I move to the larger chromosomes.

The function `orderMarkers()` may be used to establish an initial order. It picks an arbitrary pair of markers, and then adds an additional marker (chosen at random), one at a time, in the best supported position. With the argument `use.ripple=TRUE`, the function `ripple()` is used after the addition of each marker, to consider all possible orders in a sliding window of markers, to attempt to improve marker order. The argument `window` defines the length of the window. Larger values will explore more possible orders but will require considerably more computation time. The value `window=7` is usually about the largest one would ever consider;

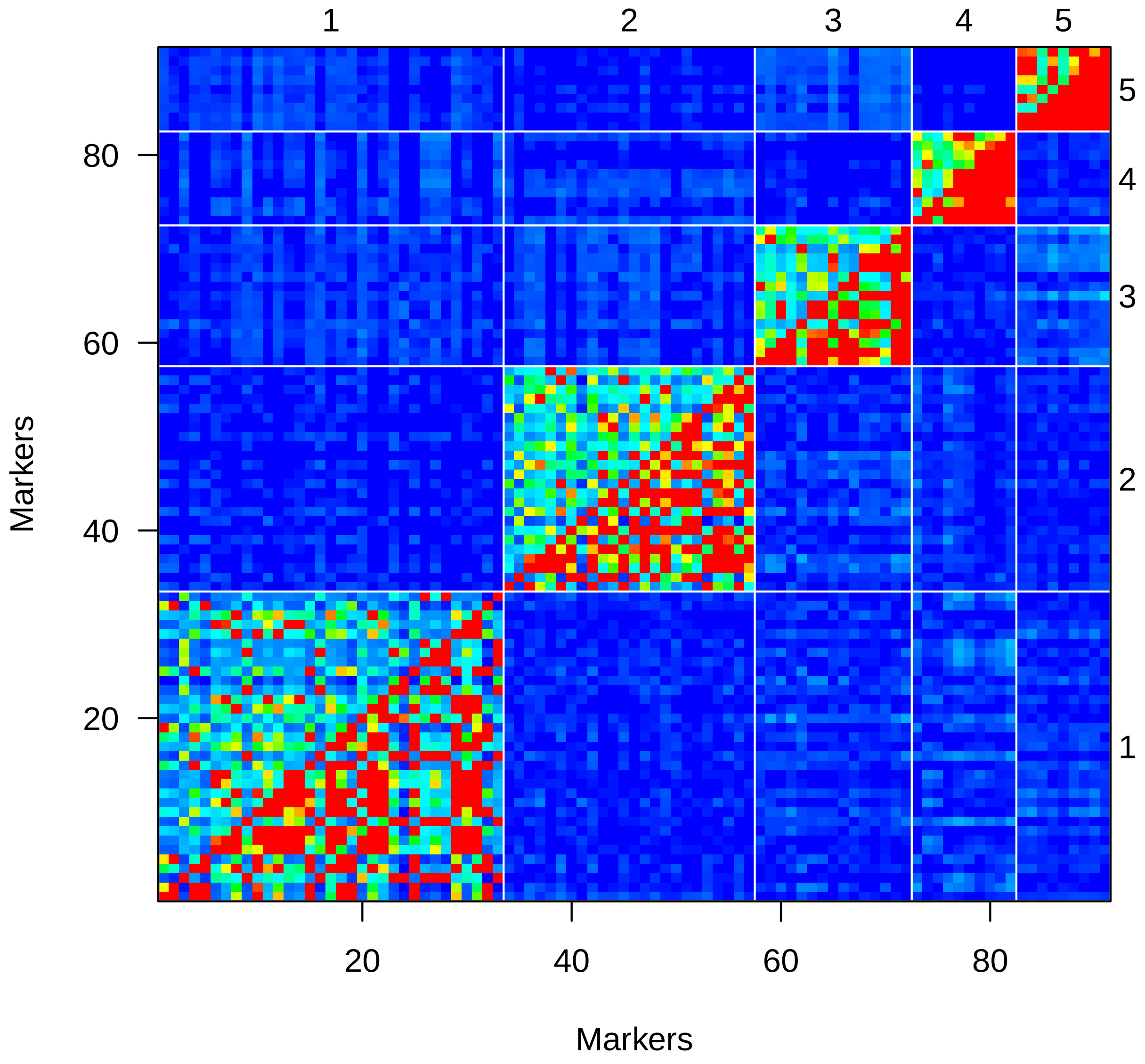

Figure 11: Plot of estimated recombination fractions (upper-left triangle) and LOD scores (lower-right triangle) for all pairs of markers, after markers have been placed in their final linkage groups. Red indicates linked (large LOD score or small recombination fraction) and blue indicates not linked (small LOD score or large recombination fraction).

use of window=8 will take so long that one would not want to sit and wait. Note that the other arguments to orderMarkers (e.g., error.prob and map.function) are passed to the function est.map() for estimation of the genetic map with the final order that is chosen. Also note that ripple(), in this case, chooses among orders in order to minimize the number of obligate crossovers. More on this below.

```
> mapthis <- orderMarkers(mapthis, chr=5)
```

We may use pull.map() to inspect the result.

```
> pull.map(mapthis, chr=5)
```

```
 C5M1  C5M3  C5M4  C5M5  C5M6  C5M7  C5M8  C5M9 C5M10
 0.00  1.40  3.30  5.37 10.00 11.03 14.23 39.31 42.11
```

Since the marker names were chosen to indicate the true marker order, we can see that we got the order exactly right. (Though note that the second marker, C5M2 was omitted at some point during the course of our analysis.) Of course, we wouldn't know this, and so we should make an attempt to explore alternate orders, in case another order might be seen to be an improvement.

We use ripple() to explore alternate orders. As mentioned above, it considers all possible orders of markers in a sliding window of fixed length, with the argument window defining the width of the window. If window is equal to the number of markers on the chromosome, than all possible orders are considered.

The quickest approach is to count the number of obligate crossovers for each order. The good orders generally are those that result in the smallest numbers of crossovers. The more refined, but considerably slower approach, is to compare the likelihoods of the different orders. (The likelihood for a given marker order is the probability of the data assuming that order is correct and plugging in estimates of the inter-marker distances.) We do this with method="likelihood" in ripple(). We may also indicate a genotyping error probability (through error.prob).

While we could start by comparing orders to minimize the number of obligate crossovers (with method="countxo", the default, in ripple()), this was already done when we called orderMarkers(), and it is not necessary to run it again. Nevertheless, the results will indicate how close, in terms of number of crossovers, the next-best marker order is to the inferred one.

```
> rip5 <- ripple(mapthis, chr=5, window=7)
```

```
   13680 total orders
```

```
> summary(rip5)
```

```
                           obligXO
Initial   1 2 3 4 5 6 7 8 9     215
1         1 2 3 4 5 6 7 9 8     216
2         1 2 3 4 6 5 7 8 9     217
```

As seen above, 13,680 marker orders were considered. (There are 9!/2 = 181,440 total marker orders.) On my computer, the code above took about 1.5 seconds. If we'd looked at all possible orders (that is, with `window=9`), it would take about 14 seconds, but the results—I checked—are the same.) Switching markers 8 and 9 results in one additional obligate crossover. If, instead, one switches markers 5 and 6, there are two additional obligate crossovers.

It is good to also study the likelihood of different orders, though we will want to greatly reduce the `window` argument, so that it can be accomplished in a reasonable amount of time.

```
> rip5lik <- ripple(mapthis, chr=5, window=4, method="likelihood",
+                   error.prob=0.005)


   114 total orders


> summary(rip5lik)


                                LOD chrlen
Initial  1 2 3 4 5 6 7 8 9      0.0   38.2
1        1 2 3 4 5 6 7 9 8      0.1   38.5
```

The result (on my computer) took about two minutes. We see that switching markers 8 and 9 has a LOD score (that is, $\log_{10}$ likelihood, relative to the initial order) of 0.1, which indicates that it is slightly preferred. However, the estimated chromosome length is slightly longer (38.5 versus 38.2 cM).

We know, from the marker names, that the initial order was the true order, but in practice we would not have such information, and we would probably want to switch markers 8 and 9. Usually we are looking to have the estimated chromosome length be as short as possible, but if we trust the likelihood calculation, we should go with the alternate order. Of course, the two orders are not really distinguishable; we can't really say, on the basis of these data, whether the correct order is the first or the second.

Note that the results here can be sensitive to the assumed genotyping error rate. (I chose `error.prob=0.005` above, because the data were simulated with this error rate.) The function `compareorder()` can be used to compare an initial order to a fixed alternative order. We can use this to quickly inspect how sensitive the results are to the assumed error rate.

```
> compareorder(mapthis, chr=5, c(1:7,9,8), error.prob=0.01)


        LOD length
orig 0.0000   36.6
new  0.0354   36.9


> compareorder(mapthis, chr=5, c(1:7,9,8), error.prob=0.001)


       LOD length
orig 0.000   40.6
new  0.181   40.8
```

```
> compareorder(mapthis, chr=5, c(1:7,9,8), error.prob=0)

       LOD length
orig 0.000   45.4
new  0.244   45.6
```

These results indicate that with smaller assumed genotyping error rates, the evidence in favor of switching markers 8 and 9 increases somewhat. Note also that the map length increases quite a bit.

If we were looking at these data blindly, I would likely go with switching markers 8 and 9, so let's go ahead and do that here. We use `switch.order()` to do so. It takes the same sort of arguments as `est.map()`, as after the marker order is switched, `est.map()` is called to revise the estimated map for the chromosome.

```
> mapthis <- switch.order(mapthis, chr=5, c(1:7,9,8), error.prob=0.005)
> pull.map(mapthis, chr=5)

 C5M1  C5M3  C5M4  C5M5  C5M6  C5M7  C5M8 C5M10  C5M9
 0.00  1.09  2.64  4.24  8.31  8.90 11.55 36.11 38.45
```

Note that the map is slightly smaller than what we had seen above, after running `orderMarkers()`, as we had used the default value for `error.prob` in that function, and now we are using `error.prob=0.005`. Also note that markers `C5M10` and `C5M9` are quite close together and are separated from the next marker by 24.6 cM. This explains why it is difficult to assess the appropriate order for these two markers.

Order markers on chromosome 4

We now turn to chromosome 4. First, we run `orderMarkers()` and print out the estimated map.

```
> mapthis <- orderMarkers(mapthis, chr=4)
> pull.map(mapthis, chr=4)

 C4M1  C4M2  C4M3  C4M4  C4M5  C4M6  C4M7  C4M8 C4M10  C4M9
  0.0  17.0  23.7  28.7  35.9  44.3  47.7  50.0  61.4  63.9
```

The marker names tell us the true order, and so we see that the inferred order is correct except that markers `C4M10` and `C4M9` are switched. Let us run `ripple()`, to see which orders have similar numbers of obligate crossovers.

```
> rip4 <- ripple(mapthis, chr=4, window=7)

   18000 total orders

> summary(rip4)

                             obligXO
Initial  1 2 3 4 5 6 7 8  9 10     326
1        1 2 3 4 5 6 7 8 10  9     326
```

The order with markers 9 and 10 switched gives the same number of obligate crossovers, and so we can not distinguish between these two marker orders. We turn to the likelihood comparison.

```
> rip4lik <- ripple(mapthis, chr=4, window=4, method="likelihood",
+                   error.prob=0.005)
```

```
   132 total orders
```

```
> summary(rip4lik)
```

```
                                     LOD chrlen
Initial   1 2 3 4 5 6 7 8  9 10     0.0   57.7
1         1 2 3 4 5 6 7 8 10  9     1.2   57.6
```

There is reasonably good evidence to switch markers 9 and 10 (which is nice, as this results in the true marker order).

```
> mapthis <- switch.order(mapthis, chr=4, c(1:8,10,9), error.prob=0.005)
> pull.map(mapthis, chr=4)
```

```
 C4M1  C4M2  C4M3  C4M4  C4M5  C4M6  C4M7  C4M8  C4M9 C4M10
  0.0  16.2  22.3  27.1  33.7  41.3  44.0  45.8  56.0  57.6
```

Order markers on chromosome 3

Turning to chromosome 3, we again run `orderMarkers()`. These calculations are starting to take quite a bit of time. It would all be faster if we reduced the argument `window` to a smaller value (say 4 rather than 7), but then not as many alternate orders will be explored and so we may not identify the best order.

```
> mapthis <- orderMarkers(mapthis, chr=3)
> pull.map(mapthis, chr=3)
```

```
C3M16 C3M15 C3M14 C3M13 C3M12 C3M11 C3M10  C3M9  C3M8  C3M6  C3M5  C3M4  C3M3  C3M2
  0.0  14.7  15.9  19.5  29.3  32.6  43.9  47.5  64.1  85.3 101.8 107.3 109.4 121.3
 C3M1
127.4
```

Note, from the marker names, that the marker order is the true order. The whole chromosome is flipped, but we have no information, from the genotype data, to orient the chromosome. Let us again use `ripple()` to study alternate orders.

```
> rip3 <- ripple(mapthis, chr=3, window=7)
```

```
   39600 total orders
```

```
> summary(rip3)
```

```
                                            obligXO
Initial  1 2 3 4 5 6 7 8 9 10 11 12 13 14 15     635
1        1 3 2 4 5 6 7 8 9 10 11 12 13 14 15     641
```

The next-best order, with markers 2 and 3 switched, results in an additional 6 obligate crossovers. We turn to the likelihood comparison.

```
> rip3lik <- ripple(mapthis, chr=3, window=4, method="likelihood",
+                   error.prob=0.005)


   222 total orders


> summary(rip3lik)


                                                LOD chrlen
Initial  1 2 3 4 5 6 7 8 9 10 11 12 13 14 15     0.0    111
1        1 3 2 4 5 6 7 8 9 10 11 12 13 14 15    -2.4    110
```

The next-best order (with markers 2 and 3 switched) is considerably worse than our initial order.

Order markers on chromosome 2

We turn to chromosome 2, beginning with `orderMarkers()`.

```
> mapthis <- orderMarkers(mapthis, chr=2)
> pull.map(mapthis, chr=2)


 C2M1  C2M2  C2M3  C2M4  C2M5  C2M6  C2M8 C2M10 C2M11 C2M12 C2M13 C2M14 C2M16 C2M17
  0.0  14.8  26.6  28.0  34.0  35.8  58.1  74.4  78.4  86.6  94.0  98.6 109.0 110.6
C2M18 C2M19 C2M20 C2M21 C2M22 C2M23 C2M24 C2M25 C2M26 C2M28
124.2 126.3 131.7 138.5 150.1 157.4 163.3 164.2 174.2 186.9
```

Again, as seen from the marker names, the inferred order is the true one. Let us run `ripple()` to inspect alternate orders.

```
> rip2 <- ripple(mapthis, chr=2, window=7)


   78480 total orders


> summary(rip2)


                                                                       obligXO
Initial  1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24     942
1        1 2 4 3 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24     948
```

The next-best order, with markers 3 and 4 switched, has 6 additional obligate crossovers. We turn to the likelihood comparison.

```
> rip2lik <- ripple(mapthis, chr=2, window=4, method="likelihood",
+                   error.prob=0.005)


   384 total orders
```

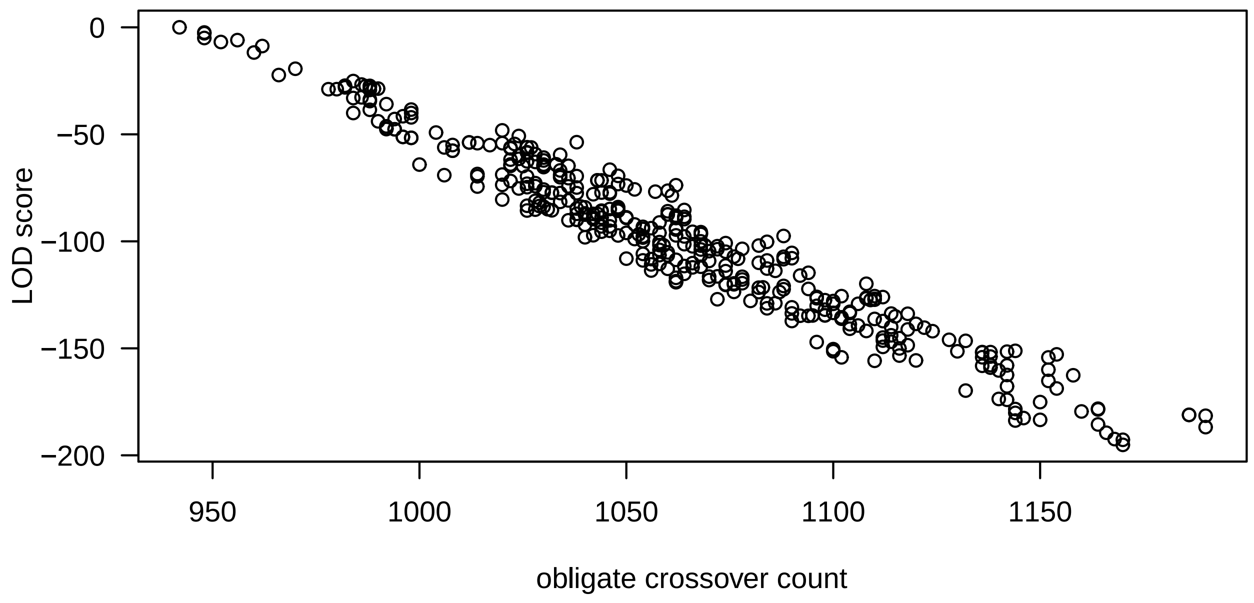

Figure 12: Comparison of the number of obligate crossovers to the LOD score (relative to our inferred order), for chromosome 2 marker orders explored via `ripple()`.

```
> summary(rip2lik)
```

```
                                                                   LOD chrlen
Initial   1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24     0.0    161
1         1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 22 21 23 24    -2.4    161
```

The next-best order, in terms of likelihood, has markers 21 and 22 switched, and is considerably worse than our initial order.

It is interesting to compare the numbers of obligate crossovers for different orders to the LOD scores. We have LOD scores for a much smaller number of orders (384 versus 78,480), since we had used a smaller value for `window`, but we can pull out the obligate crossover counts for those orders that we evaluated in terms of likelihood. It is a bit tricky to line up the orders.

```
> pat2 <- apply(rip2[,1:24], 1, paste, collapse=":")
> pat2lik <- apply(rip2lik[,1:24], 1, paste, collapse=":")
> rip2 <- rip2[match(pat2lik, pat2),]
> plot(rip2[,"obligXO"], rip2lik[,"LOD"], xlab="obligate crossover count",
+      ylab="LOD score")
```

As seen in Fig. 12, there is a clear negative relationship between the crossover counts and the likelihoods, though the relationship is not perfect, particularly for the orders that are not well supported.

Order markers on chromosome 1

Finally, we turn to chromosome 1, first running `orderMarkers()`.

```
> mapthis <- orderMarkers(mapthis, chr=1)
> pull.map(mapthis, chr=1)


  C1M1   C1M2   C1M3   C1M5   C1M6   C1M7   C1M8   C1M9  C1M10  C1M11  C1M12  C1M13
  0.00   1.68   4.50  15.30  20.43  34.09  38.67  41.16  46.48  54.37  60.16  61.20
 C1M14  C1M15  C1M16  C1M17  C1M18  C1M19  C1M20  C1M22  C1M23  C1M24  C1M25  C1M26
 65.21  70.57  72.75  80.87  94.85  99.00 100.21 106.14 107.35 108.18 111.37 135.85
 C1M27  C1M28  C1M29  C1M30  C1M31  C1M33  C1M34  C1M35  C1M36
181.40 200.20 202.83 205.46 206.47 220.43 234.60 235.50 239.29
```

Again, the inferred order is precisely the true order. I was quite surprised to see how well `orderMarkers()` performed with these data. Of course, the data are quite clean (by design) and comprise quite a large number of individuals. In practice, one can't expect `orderMarkers()` to perform so well, and it is worthwhile to study the estimated map and the pairwise linkage information closely. We will do so in a moment, but first let's complete our analysis of chromosome 1 by studying the results of `ripple`.

```
> rip1 <- ripple(mapthis, chr=1, window=7)


   117360 total orders


> summary(rip1)


Initial   1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29
1         1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 29 28
2         1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29
                              obligXO
Initial   30 31 32 33    1155
1         30 31 32 33    1157
2         30 32 31 33    1157
```

Two alternate orders (switching markers 28 and 29, or switching markers 31 and 32) have 2 additional obligate crossovers, compared to our initial order. We turn to the likelihood comparison.

```
> rip1lik <- ripple(mapthis, chr=1, window=4, method="likelihood",
+                   error.prob=0.005)


   546 total orders


> summary(rip1lik)


Initial   1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29
1         1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 29 28
                                  LOD chrlen
Initial   30 31 32 33     0.0    213
1         30 31 32 33    -1.7    212
```

The next best order (switching markers 28 and 29) results in a moderately large decrease in likelihood, and so we stick with the initial order produced by `orderMarkers()`.

Let us return to the question of how to assess the quality of the results of `orderMarkers()`, aside from our investigations with `ripple()`. The `ripple()` analysis is excellent for comparing nearby marker orders. And for chromosomes with a modest number of markers (like chromosome 5, here), one can consider all possible orders exhaustively. But with many markers (as with chromosome 1), we can investigate only a small proportion of the possible orders. For example, with 33 markers (as on chromosome 1), there are $33!/2 \approx 10^{36}$ possible marker orders, and we investigated only 117,360 of them using `ripple` with `window=7`.

There are several things to look it, in assessing whether gross changes in marker order are necessary. (Ideally, R/qtl would include a function providing a more complete investigation of marker order, perhaps via simulated annealing or another randomized optimization algorithm, and we do hope to implement such a feature in the future.) First, we should look at the actual map: are there large gaps between markers, indicating adjacent markers that are only weakly linked? Studying the map locations with `pull.map()`, as above, is hard when there are lots of markers. The function `summary.map()` provides a summary of the average inter-marker distance and the largest gap on each chromosome.

```
> summary.map(mapthis)
```

```
        n.mar length ave.spacing max.spacing
1          33  239.3         7.5        45.6
2          24  186.9         8.1        22.3
3          15  127.4         9.1        21.2
4          10   57.6         6.4        16.2
5           9   38.5         4.8        24.6
overall    91  649.6         7.6        45.6
```

We see that chromosome 1 has a 45.6 cM gap; the other chromosomes have gaps no larger than 25 cM. The large gap on chromosome 1 is suspicious, but it is not terrible.

In addition to this summary, it is good to also plot the map. We use the argument `show.marker.names=TRUE` so that marker names are included. Most of them are unreadable, because the markers are densely spaced, but at least we learn the identity of markers that are surrounded by large gaps.

```
> plot.map(mapthis, show.marker.names=TRUE)
```

As seen in Fig. 13, there are some large gaps on chromosome 1 and some smaller gaps on the other chromosomes. They deserve further investigation (and we will do so in the next section), but they aren't particularly troubling. With gross mistakes in marker order, one will often see much larger gaps, say > 100 cM.

Finally, we inspect the pairwise linkage information again.

```
> plot.rf(mapthis)
```

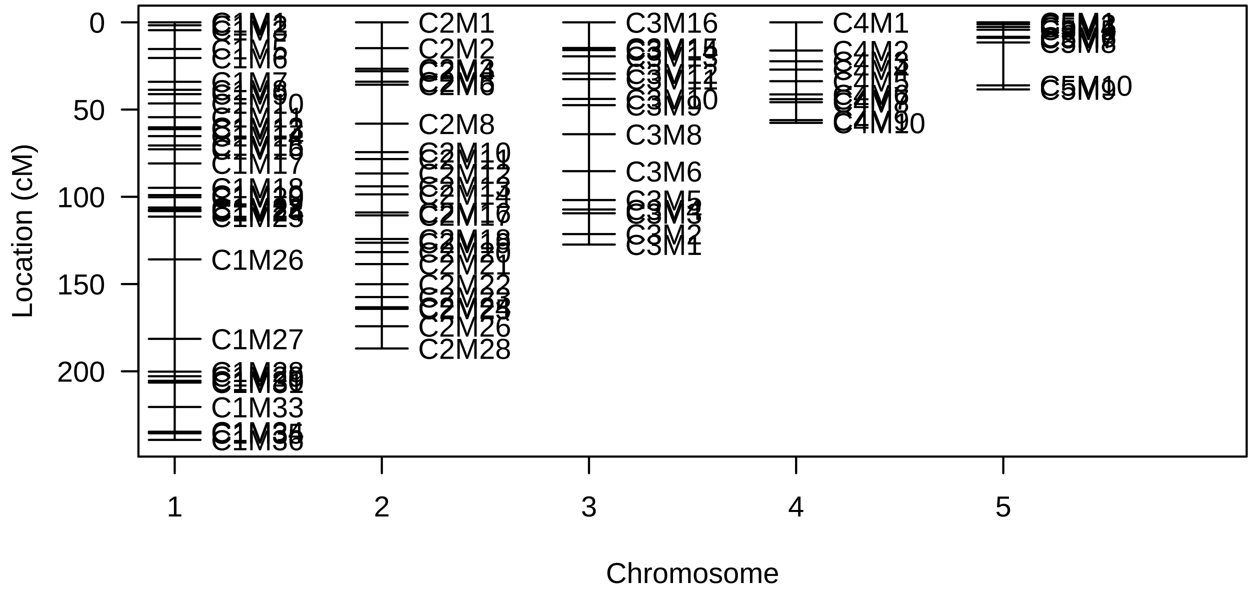

Figure 13: Plot of the estimated genetic map, after the markers on each chromosome have been ordered.

The pairwise linkage information in Fig. 14 is close to what we want to see: nearby markers show clear association and no distant markers show any association: red along the diagonal, dissipating to blue away from the diagonal.

If there were gross problems with marker order, we might see groups of distantly placed markers that are more highly associated than more closely placed markers. Just as an example, suppose we split chromosome 1 into three pieces and moved the latter piece into the middle. Let's look at the pairwise linkage information following such a re-ordering.

```
> messedup <- switch.order(mapthis, chr=1, c(1:11,23:33,12:22),
+                          error.prob=0.005)
> plot.rf(messedup, chr=1)
```

Fig. 15 (on page 30) displays the sort of pattern one should expect with gross problems in marker order: the markers in the first and last segments are more tightly associated to each other than they are to the markers in the middle segment.

A plot of the genetic map (see Fig. 16, page 30) shows a 50 cM gap between the first and middle segments and an 80 cM gap between the middle and last segments.

```
> plot.map(messedup, show.marker.names=TRUE)
```

The map is not as telling as the pairwise linkage information, but these are the sorts of things to look at, in trying to decide whether there are gross problems that need to be corrected.

If features such as those in Fig. 15 and 16 were seen, one should identify the segments of markers that need to be moved around, and then use `switch.order()` to reorganize

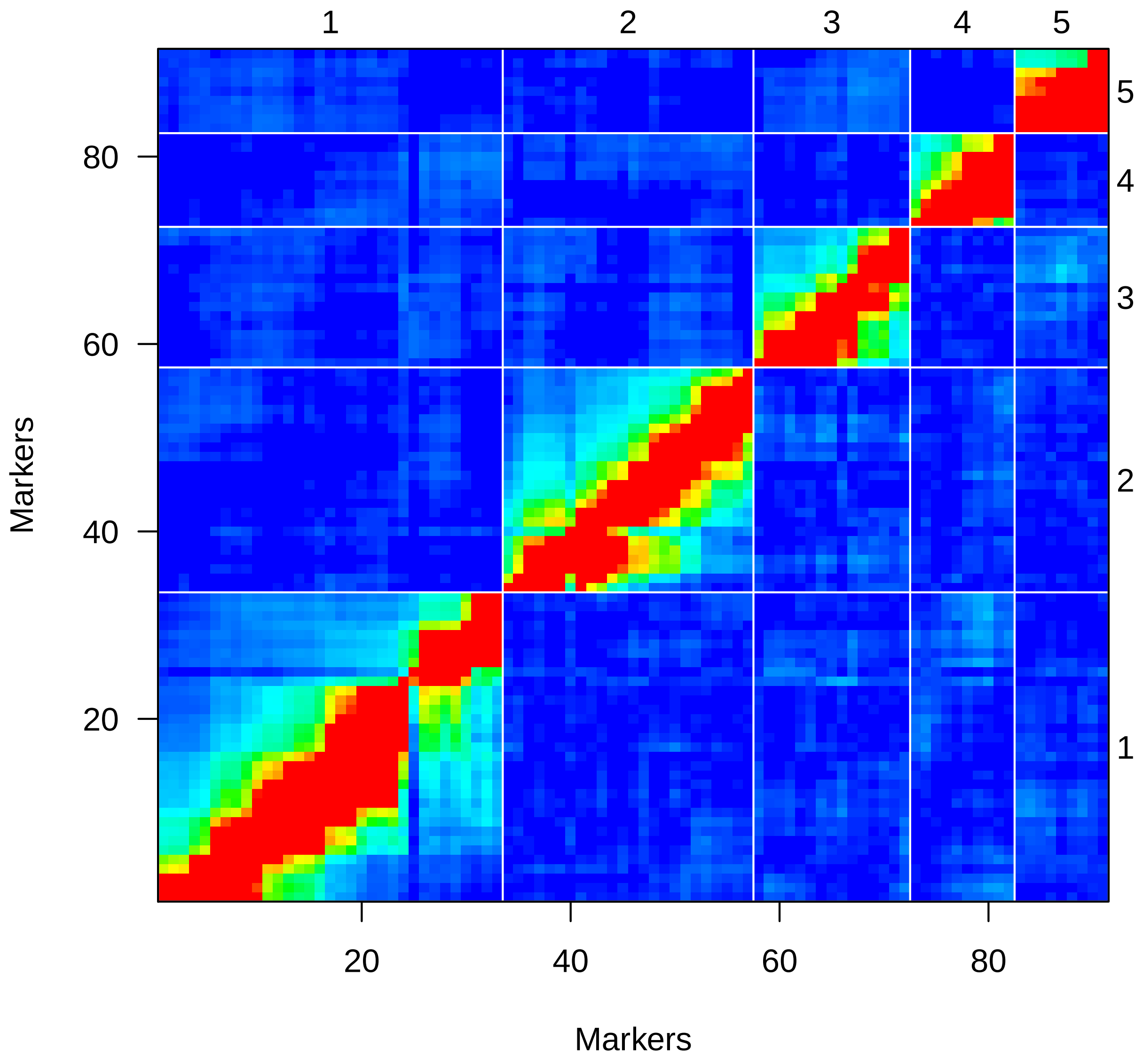

Figure 14: Plot of estimated recombination fractions (upper-left triangle) and LOD scores (lower-right triangle) for all pairs of markers, after ordering the markers on each chromosome. Red indicates linked (large LOD score or small recombination fraction) and blue indicates not linked (small LOD score or large recombination fraction).

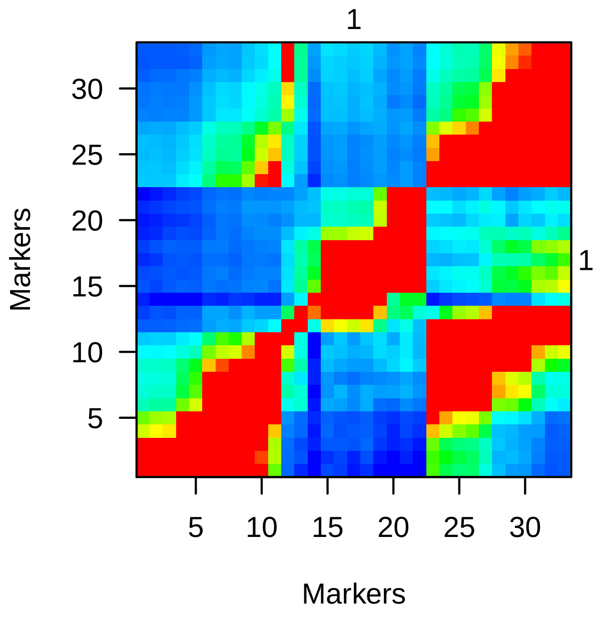

Figure 15: Plot of estimated recombination fractions (upper-left triangle) and LOD scores (lower-right triangle) for all pairs of markers on chromosome 1, after messing up the order of the markers. Red indicates linked (large LOD score or small recombination fraction) and blue indicates not linked (small LOD score or large recombination fraction).

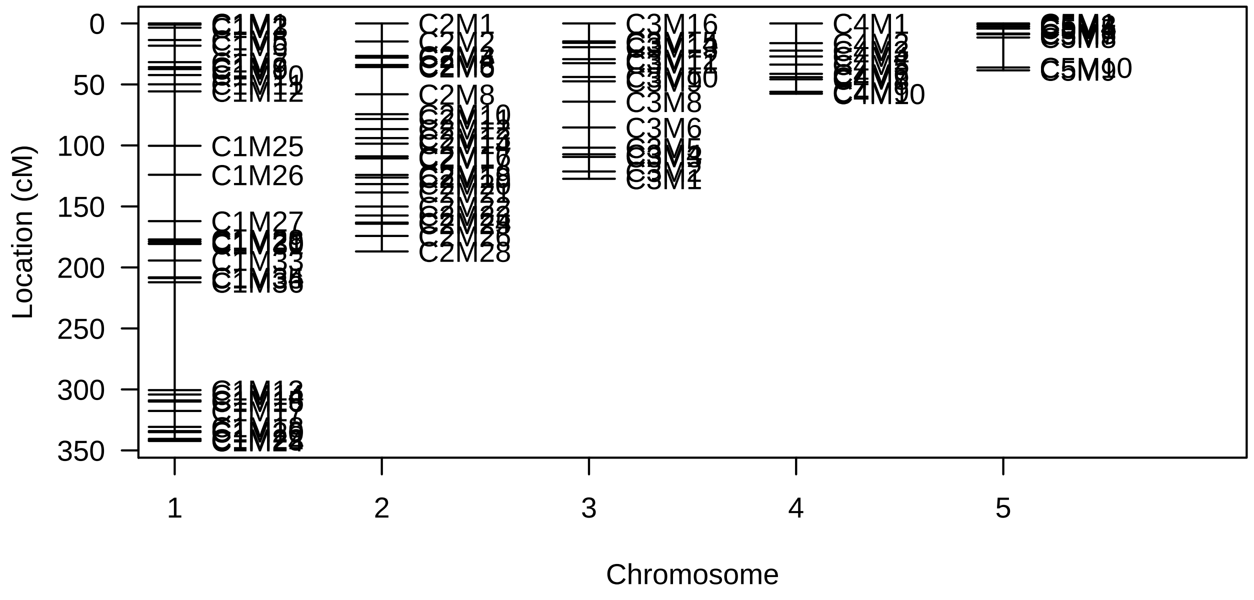

Figure 16: Plot of the estimated genetic map, after messing up the order of markers on chromosome 1.

the markers. One might first use compareorder() to compare the current order to the reorganized one, to see that the new order gave a clear improvement in likelihood. In addition, one will often need to alternate between ripple() and switch.order() until the final marker order is established.

Drop one marker at a time

The large gaps in the genetic map on chromosome 1 remain a concern. While such gaps may indicate problems with the order of markers, they might also indicate a high genotyping error rate at certain markers. If an individual marker is more prone to genotyping errors than others, it would often (in the sort of analyses performed above) be placed at one end of the chromosome or the other, but in some cases (particularly if the genotyping error rate is not high and there are a large number of individuals in the cross) it may be placed at approximately the correct position but result in reasonably large gaps surrounding the marker.

One approach for identifying such problematic markers is to drop one marker at a time and investigate the change in chromosome length and the change in log likelihood. This analysis may be accomplished with the function droponemarker().

```
> dropone <- droponemarker(mapthis, error.prob=0.005)
```

The results are of the same form as a genome scan by QTL mapping. (In particular, they have class "scanone", like an object produced by the scanone() function.) We may plot the results as follows.

```
> par(mfrow=c(2,1))
> plot(dropone, lod=1, ylim=c(-100,0))
> plot(dropone, lod=2, ylab="Change in chromosome length")
```

As seen in top panel of Fig. 17, there is no one marker for which its omission results in an increase in likelihood, but as seen in the bottom panel, there are a number of markers that give an appreciable decrease in chromosome length, of 15–25 cM, when omitted. Markers at the ends of chromosomes will often result in a smaller estimated chromosome length, but that is just because such terminal markers hang off some distance from the rest of the markers, and so these changes can often be discounted. Interior markers that result in a big change, and there appears to be one on each of chromosomes 1, 2 and 3, might indicate error-prone markers that are best omitted.

One may identify the marker on each chromosome whose omission results in the largest decreases in chromosome length as follows.

```
> summary(dropone, lod.column=2)
```

```
      chr   pos   LOD Ldiff
C1M27   1 162.4 -26.3 25.78
C2M8    2  47.3 -13.7 15.79
C3M8    3  55.4 -31.0 14.86
C4M1    4   0.0 -41.5 16.13
C5M9    5  38.5 -97.6  2.73
```

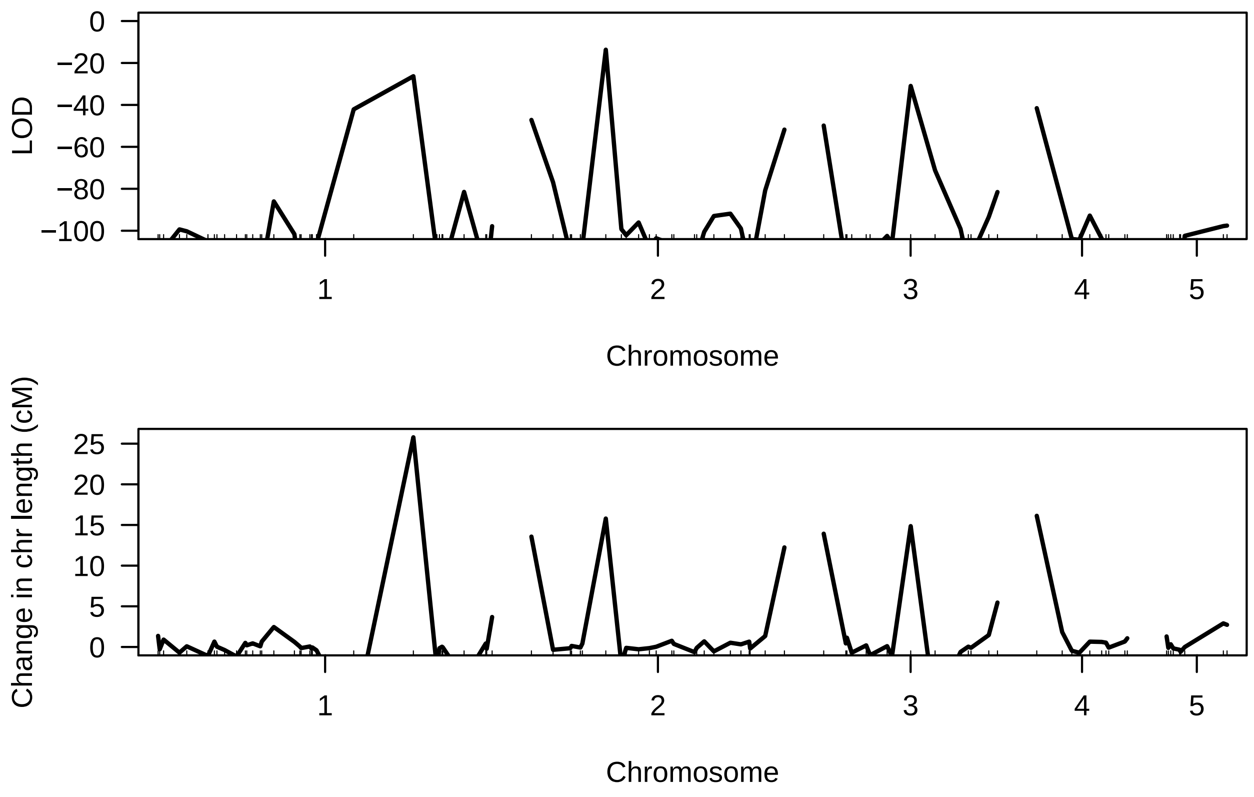

Figure 17: Results of dropping one marker at a time. The top panel contains LOD scores; positive values would indicate that dropping a marker improves the likelihood. The bottom panel indicates the decrease in estimated chromosome length (in cM) following dropping a marker.

For chromosomes 4 and 5, these are terminal markers. The markers on chromosomes 1, 2 and 3 are all interior markers. One should probably study the pairwise linkage between these markers and surrounding markers before proceeding, but we will go ahead and remove these markers without any further investigations.

```
> badmar <- rownames(summary(dropone, lod.column=2))[1:3]
> mapthis <- drop.markers(mapthis, badmar)
```

One should re-estimate the genetic map. We use `replace.map()` to insert it into the cross object.

```
> newmap <- est.map(mapthis, error.prob=0.005)
> mapthis <- replace.map(mapthis, newmap)
> summary.map(mapthis)


        n.mar length ave.spacing max.spacing
1          32  186.8         6.0        27.1
2          23  145.2         6.6        13.8
3          14   95.7         7.4        16.1
4          10   57.6         6.4        16.2
5           9   38.5         4.8        24.6
overall    88  523.7         6.3        27.1
```

Removing those three markers resulted in a decrease in the overall map length from 650 cM to 524 cM.

Look for problem individuals

Now that we have the markers in their appropriate order, it is a good idea to return to the question of whether there are particular individuals whose data are problematic. One should ask: if a particular individual showed considerable genotyping errors or did not actually belong to the cross under investigation (e.g., through some sort of labeling or breeding error), what sort of aberrations might be seen in the data? Above, we studied the amount of missing genotype data for each individual as well as the individuals' genotype frequencies. Another feature to investigate is the observed number of crossovers in each individual. These may be counted with the function `countXO()`. (A related function `locateXO()` will return the estimated locations of all crossovers.)

```
> plot(countXO(mapthis), ylab="Number of crossovers")
```

The crossover counts in Fig. 18 clearly indicate two problematic individuals, with 73 and 86 crossovers; the other individuals have 3–20 crossovers. We should remove these individuals.

```
> mapthis <- subset(mapthis, ind=(countXO(mapthis) < 50))
```

Ideally, we would now revisit the entire process again; in particular, after removing these problematic individuals, is there evidence that marker order needs to be changed? Let us at least look at chromosome 5.

```
> summary(rip <- ripple(mapthis, chr=5, window=7))
```

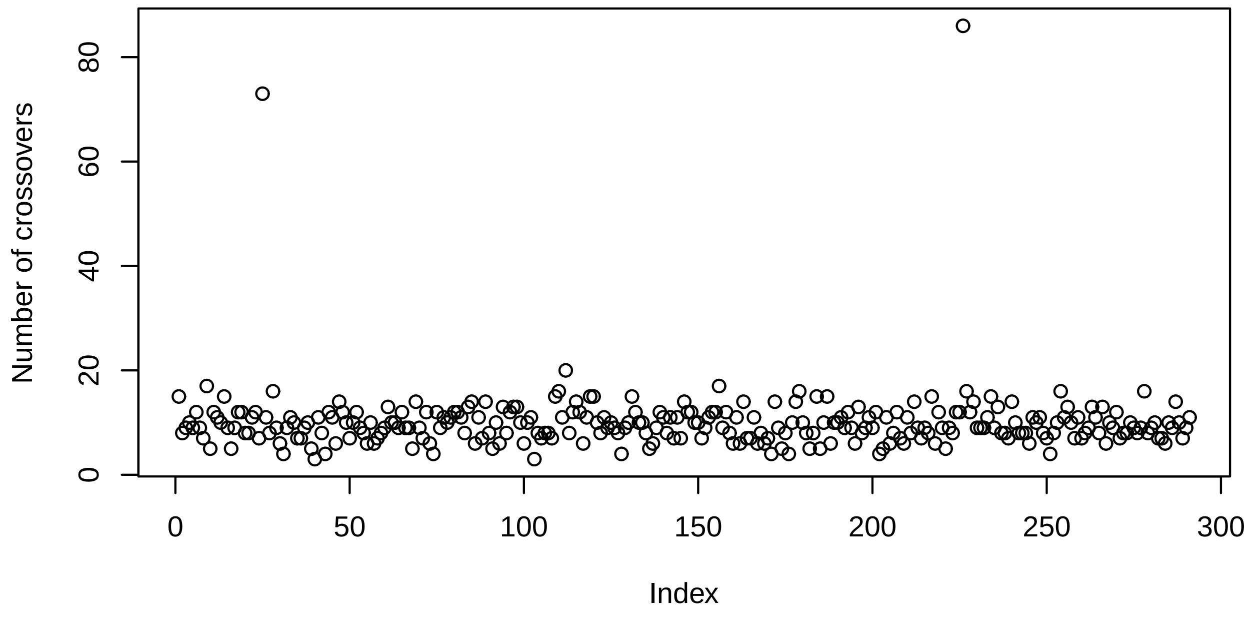

Figure 18: Numbers of observed crossovers in each individual.

```
   13680 total orders
                            obligXO
Initial   1 2 3 4 5 6 7 8 9     198
1         1 2 3 4 5 6 7 9 8     195


> summary(rip <- ripple(mapthis, chr=5, window=2, method="likelihood",
+                        error.prob=0.005))


   9 total orders
    --Order 2
    --Order 4
    --Order 6
    --Order 8
                                LOD chrlen
Initial   1 2 3 4 5 6 7 8 9     0.0   37.8
1         1 2 3 4 5 6 7 9 8     0.9   37.1
```

There is good evidence for switching markers 8 and 9, which actually brings us back to the true order of the markers.

```
> mapthis <- switch.order(mapthis, chr=5, c(1:7,9,8), error.prob=0.005)
> pull.map(mapthis, chr=5)


 C5M1  C5M3  C5M4  C5M5  C5M6  C5M7  C5M8  C5M9 C5M10
 0.00  1.06  2.50  3.93  7.88  8.48 11.09 35.07 37.07
```

I investigated the other four chromosomes similarly, and there was no evidence for further changes in marker order, so I won't present the results here. We should, finally, re-estimate the genetic map.

```
> newmap <- est.map(mapthis, error.prob=0.005)
> mapthis <- replace.map(mapthis, newmap)
> summary.map(mapthis)
```

```
        n.mar length ave.spacing max.spacing
1          32  181.6         5.9        26.7
2          23  141.5         6.4        13.1
3          14   94.0         7.2        16.3
4          10   55.4         6.2        15.6
5           9   37.1         4.6        24.0
overall    88  509.6         6.1        26.7
```

The overall map length has decreased further, from 524 cM to 510 cM.

Estimate genotyping error rate

Above, I had generally assumed a genotyping error rate of 5/1000 (in estimating map distances and in likelihood calculations comparing different marker orders); but I cheated somewhat in using this value, as the genotype data were simulated with this error rate.

One can actually estimate the genotyping error rate from the data, as the function `est.map()` not only estimates the inter-marker distances, but also calculates the log likelihood for each chromosome. Thus, if we run `est.map()` with different assumed values for the genotyping error rate (specified with the `error.prob` argument), one can identify the maximum likelihood estimate of the error rate.

```
> loglik <- err <- c(0.001, 0.0025, 0.005, 0.0075, 0.01, 0.0125, 0.015, 0.0175, 0.02)
> for(i in seq(along=err)) {
+   cat(i, "of", length(err), "\n")
+   tempmap <- est.map(mapthis, error.prob=err[i])
+   loglik[i] <- sum(sapply(tempmap, attr, "loglik"))
+ }
> lod <- (loglik - max(loglik))/log(10)
```

The code is a bit tricky, mostly because the log likelihoods (and note that they are on the natural log scale) are included as "attributes" to each chromosome component in the output from `est.map()`. We use `sapply()` to pull those out, and then we add them up. We finally convert them to the $\log_{10}$ scale and re-center them so that the maximum is 0.

We may plot the $\log_{10}$ likelihood as follows.

```
> plot(err, lod, xlab="Genotyping error rate", xlim=c(0,0.02),
+      ylab=expression(paste(log[10], " likelihood")))
```

The $\log_{10}$ likelihood in Fig. 19 indicates that the MLE is approximately 0.005. Error rates of 0.0025 and 0.0075 have $\log_{10}$ likelihoods that are 3 less than that of the MLE. We might investigate additional assumed error rates, or even use the R function `optimize()` to refine our estimate, but we won't pursue that here.

Look for genotyping errors

While the methods for estimating inter-marker distances allow for the presence of genotyping errors at a fixed rate, it is nevertheless worthwhile to look for, and ideally correct, potential genotyping errors in the data. Such errors may be identified through apparent tight double-crossovers, with a single marker being out of phase with its adjacent markers.

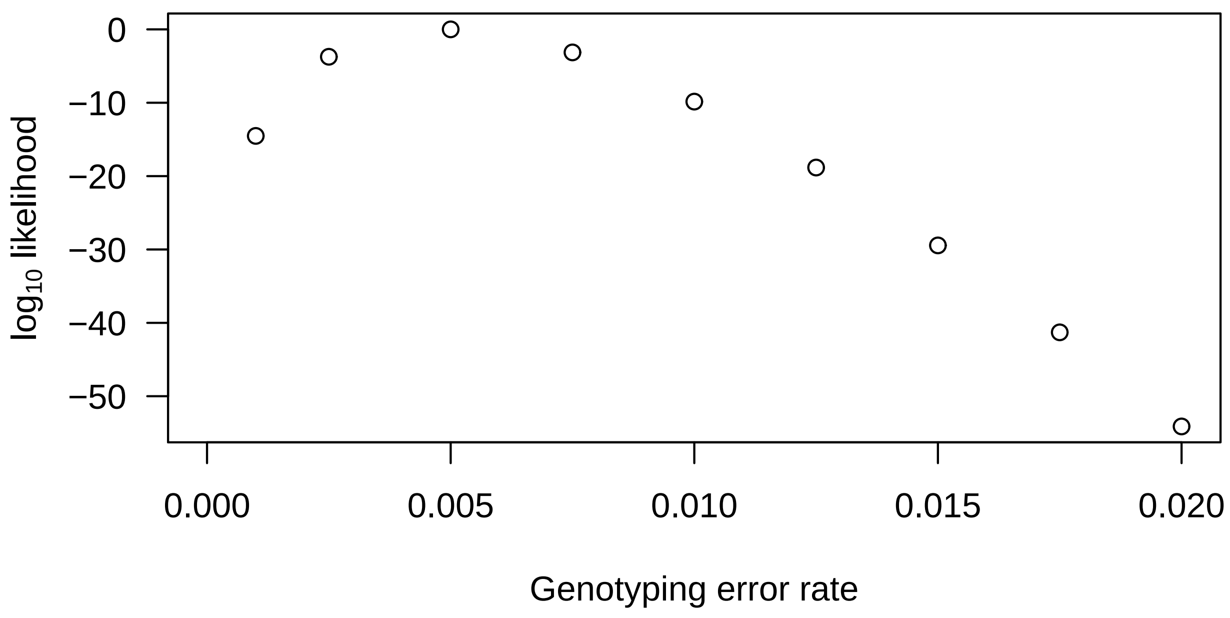

Figure 19: The $\log_{10}$ likelihood for the genotyping error rate.

The most convenient approach for identifying such double-crossovers is to calculate genotyping error LOD scores, first developed by Lincoln and Lander (*Genomics* 14:604–610, 1992). The LOD score compares the likelihood for a genotype being in error versus it not being in error. R/qtl uses a modified calculation of such genotyping error LOD scores, with all genotypes except that being considered assumed to be strictly correct.

The error LOD scores are calculated with `calc.errorlod()`. One must assume a genotyping error rate, but the results are almost identical for a wide range of values.

```
> mapthis <- calc.errorlod(mapthis, error.prob=0.005)
```

The function `top.errorlod()` produces a list of the genotypes with the largest error LOD scores. One may generally focus on those with quite large values, say at least 4–5. Here will we look at just those genotypes with error LOD ≥ 6; this is indicated with the argument `cutoff`.

```
> print(toperr <- top.errorlod(mapthis, cutoff=6))
```

```
   chr    id marker errorlod
1    1 id200  C1M23     8.25
2    1  id36  C1M30     8.13
3    1 id236  C1M24     7.32
4    1 id217  C1M35     7.25
5    1  id35  C1M29     7.10
6    1 id261  C1M29     7.10
7    5 id112   C5M6     6.96
8    4 id113   C4M7     6.58
9    1  id87   C1M9     6.22
10   5 id122   C5M5     6.19
11   1  id72  C1M34     6.03
12   2 id115  C2M19     6.02
```

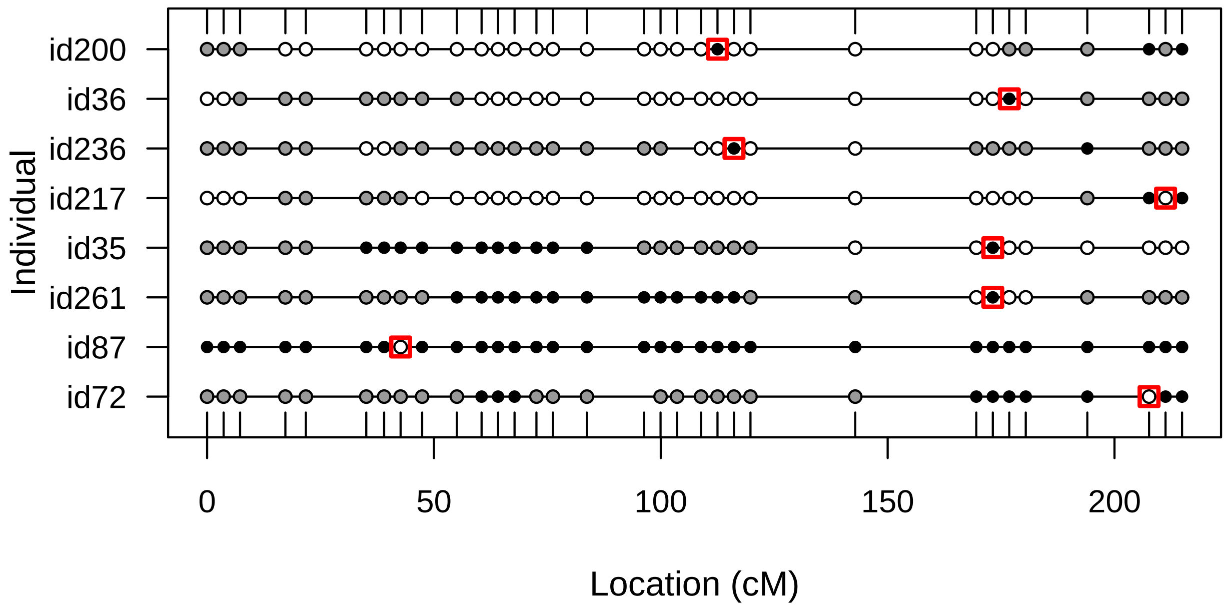

Figure 20: Genotypes on chromosome 1 for individuals with some potential errors flagged by red squares. White, gray and black circles correspond to AA, AB and BB genotypes, respectively.

There are 12 genotypes that meet this criterion. Let us look at the genotypes that were flagged on chromosome 1, using the function `plot.geno()`. The argument `cutoff` indicates a threshold for flagging genotypes as potential errors, based on their error LOD scores. If we had used the argument `include.xo=TRUE` (which is the default), inferred locations of crossovers would be displayed; we suppress that here to get a more clean figure.

```
> plot.geno(mapthis, chr=1, ind=toperr$id[toperr$chr==1],
+           cutoff=6, include.xo=FALSE)
```

The results are in Fig. 20. All of the flagged genotypes are cases with an exchange from one homozygote to the other and then back again (thus, two crossovers in each interval flanking a single marker).

One might zero out these suspicious genotypes (that is, make them missing). Even better would be to revisit the raw genotyping information, or even re-genotype these instances. But we are talking about just 12 genotypes out of 25,147, and with our allowance for genotyping errors in the map estimation, they have little influence on the results.

If we did wish to delete these genotypes, we could do so as follows.

```
> mapthis.clean <- mapthis
> for(i in 1:nrow(toperr)) {
+   chr <- toperr$chr[i]
+   id <- toperr$id[i]
+   mar <- toperr$marker[i]
+   mapthis.clean$geno[[chr]]$data[mapthis$pheno$id==id, mar] <- NA
+ }
```

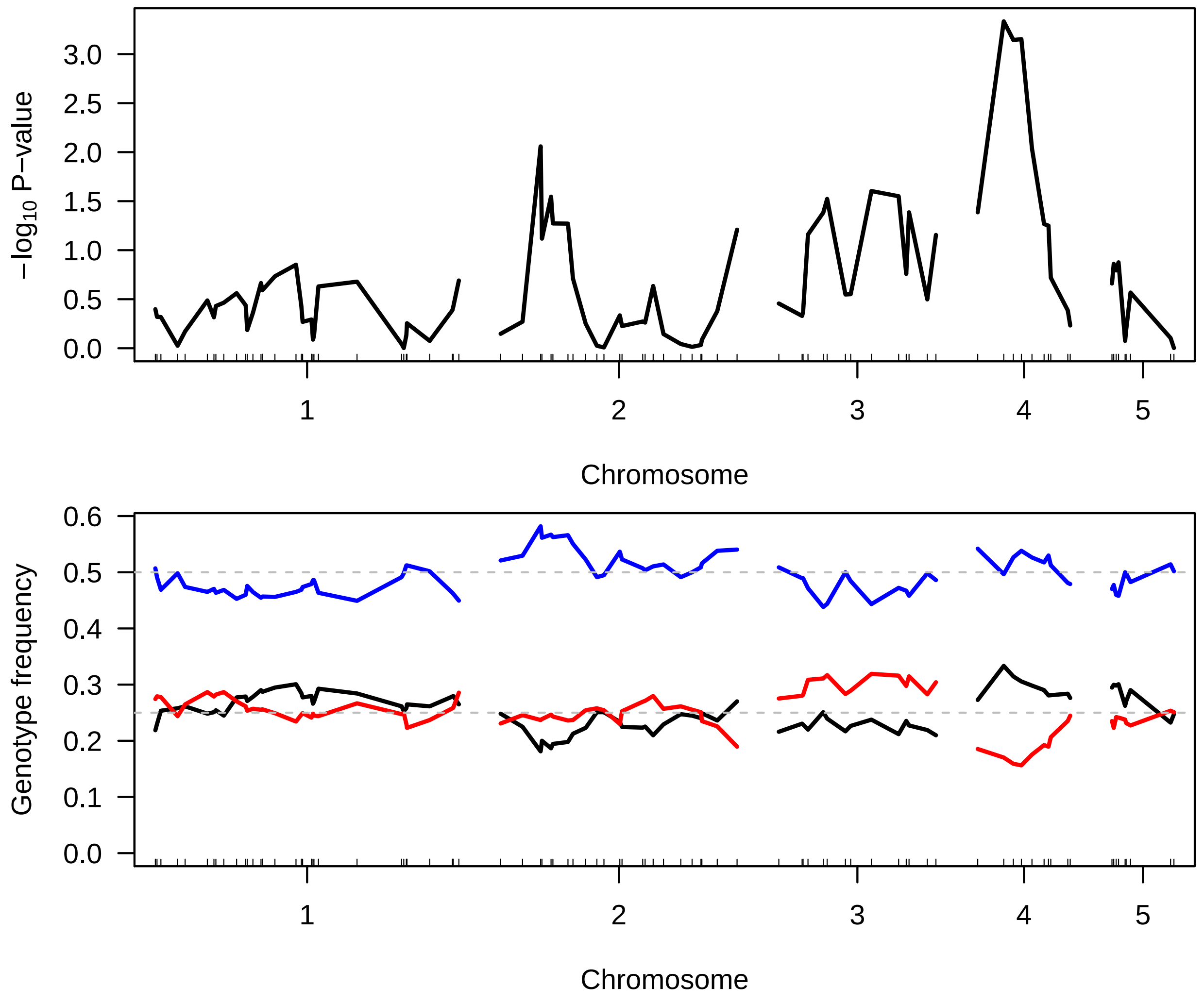

Figure 21: Evidence for segregation distortion: $-\log_{10}$ P-values from tests of 1:2:1 segregation at each marker (top panel) and the genotype frequencies at each marker (bottom panel, with black, blue and red denoting AA, AB and BB genotypes, respectively).

Revisit segregation distortion

Finally, let us return to an investigation of segregation distortion in these data.

```
> gt <- geno.table(mapthis, scanone.output=TRUE)
> par(mfrow=c(2,1))
> plot(gt, ylab=expression(paste(-log[10], " P-value")))
> plot(gt, lod=3:5, ylab="Genotype frequency")
> abline(h=c(0.25, 0.5), lty=2, col="gray")
```

The top panel of Fig. 21 contains $-\log_{10}$ P-values from tests of 1:2:1 segregation at each marker. The bottom panel in Fig. 21 contains the observed genotype frequencies at each marker (with black, blue and red corresponding to AA, AB and BB genotypes, respectively).

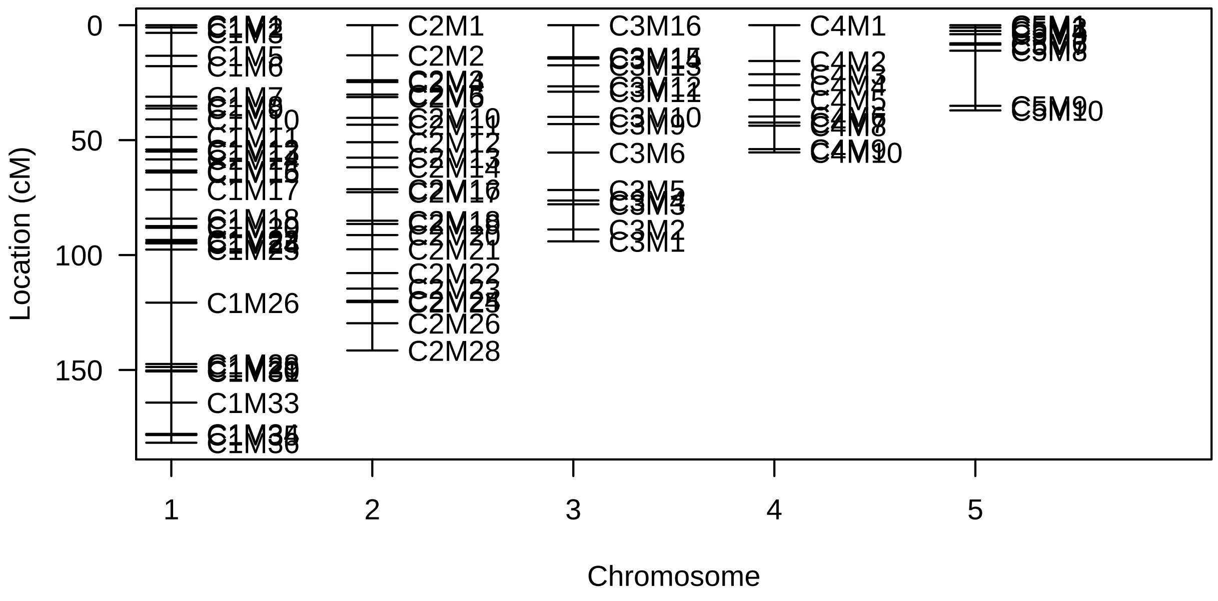

Figure 22: Plot of the final estimated genetic map

The greatest departure from 1:2:1 segregation is on chromosome 4, with somewhat more AA genotypes and somewhat fewer BB genotypes. If we apply a Bonferroni correction for the 88 tests (88 is the total number of markers we have retained in the data), we would look for $P \geq 0.05/88$ which corresponds to $-\log_{10} P \geq 3.25$, and there is one marker on chromosome 4 that exceeds this.

There are also some departures from 1:2:1 segregation on chromosomes 2 and 3, but these appear to be within the range of what would be expected by chance; the evidence for a real departure from normal segregation is not strong.

The aberrant segregation pattern on chromosome 4 is not too worrisome. Multipoint estimates of genetic map distances are little affected by segregation distortion, and the pattern of distortion on the chromosome indicates rather smooth changes in genotype frequency. More worrisome would be a single distorted marker in the midst of other markers with normal segregation, which would indicate genotyping errors rather than, for example, the presence of partially lethal alleles.

And so, finally, we're done. Let us plot the final map.

```
> plot.map(mapthis, show.marker.names=TRUE)
```

The map in Fig. 22 is not pretty, as most of the marker names are obscured. R/qtl does not produce production-quality figures automatically; I always go through quite a few extra contortions within R to produce a figure suitable for a paper.

Discussion

The process of genetic map construction seems to be at least 90% data diagnostics. While many might find that frustrating, for me that is a large part of what makes it fun: it is interesting detective work. I have been involved in the construction of genetic maps for humans, mice, dogs, zebrafish, mosquitoes, and sea squirts. Each project was different, with its own special issues that needed to be overcome, and such issues can seldom be anticipated in advance.

The general strategy is to think about what sorts of things might be going wrong with data, and then what sorts of features of the data (summary statistics or plots) might indicate the presence of such problems. There are a variety of things to check routinely, and note that the particular order in which these checks are performed is often important.

In the procedures described above, and in forming these simulated data, I attempted to cover most of the possible problems that might be expected to arise. With the simple intercross considered here, and particularly as the data were simulated so that the problems would generally be quite clear, the decisions about how to proceed were easy to make. In practice, potential problems in data will often be more murky, and careful judgment calls will need to be made and frequently revisited, ideally with careful consideration of raw genotyping data and other records, and perhaps even with some additional rounds of genotyping or even the redesign of genotyping assays. Moreover, with more complex experiments, such as an outcross or the combined analysis of multiple crosses, additional issues will arise that we have not touched on here. But the overall strategy, laid out above, can be applied quite generally.

R/qtl functions useful for genetic map construction

plot.missing	Plot pattern of missing genotypes
ntyped	Count number of typed markers for each individual
nmissing	Count number of missing genotypes for each individual
subset.cross	Pull out a specified set of chromosomes and/or individuals from a cross
drop.markers	Remove a list of markers
pull.markers	Drop all but a selected set of markers
drop.nullmarkers	Remove markers without data
comparegeno	Count proportion of matching genotypes between all pairs of individuals
findDupMarkers	Find markers with identical genotype data
drop.dupmarkers	Drop duplicate markers
geno.table	Create table of genotype distributions
est.rf	Estimate pairwise recombination fractions
markerlrt	General likelihood ratio test for association between marker pairs
checkAlleles	Identify markers with potentially switched alleles
pull.rf	Pull out the pairwise recombination fractions or LOD scores as a matrix
formLinkageGroups	Partition markers into linkage groups
plot.rf	Plot recombination fractions
markernames	Pull out the marker names from a cross
plot.rfmatrix	Plot a slice through the pairwise recombination fractions or LOD scores
geno.crosstab	Create cross-tabulation of genotypes at two markers
switchAlleles	Switch alleles at selected markers
orderMarkers	Find an initial order for markers within chromosomes
ripple	Assess marker order by permuting groups of adjacent markers
summary.ripple	Print summary of ripple output
est.map	Estimate genetic map
pull.map	Pull out the genetic map from a cross
compareorder	Compare two orderings of markers on a chromosome
switch.order	Switch the order of markers on a chromosome
summary.map	Print summary of a genetic map
plot.map	Plot genetic map(s)
droponemarker	Drop one marker at a time from a genetic map
replace.map	Replace the genetic map of a cross
countXO	Count number of obligate crossovers for each individual
locateXO	Estimate locations of crossovers
calc.errorlod	Calculate Lincoln & Lander (1992) error LOD scores
top.errorlod	List genotypes with highest error LOD values
plot.geno	Plot genotypes on a particular chromosomes for selected individuals
tryallpositions	Test all possible positions for a marker
allchrsplits	Test all possible splits of a chromosome into two pieces
movemarker	Move a marker from one chromosome to another
convert.map	Change map function for a genetic map
shiftmap	Shift starting points in genetic maps
rescalemap	Rescale genetic map