New functions for exploring multiple-QTL models

Karl W Broman, 7 February 2008

(minor revisions 28 May 2008; further revised 18 July 2008 to discuss `stepwiseqtl`; added color 26 Oct 2010)

R/qtl version 1.08 included a number of new functions to simplify the exploration of multiple-QTL models. In R/qtl version 1.09, the function `stepwiseqtl` was added to perform a stepwise selection to identify the multiple-QTL model optimizing a penalized LOD score.

The "Brief tour of R/qtl" document contains a brief description of these functions (see Example 5 in that document); here, we provide a more thorough explanation.

Introduction

Let us begin with a brief overview of the changes. The basic functions remain `makeqtl` (for creating a QTL object), `fitqtl` (for fitting a defined multiple-QTL model) and `scanqtl` (for multi-dimensional scans with a multiple-QTL model).

Previously, `fitqtl` and `scanqtl` used multiple imputation exclusively. We have now implemented Haley-Knott regression as well. To use Haley-Knott regression, use the argument `what="prob"` in a call to `makeqtl` and then `method="hk"` in calls to `fitqtl` and `scanqtl` (and the other functions, to be described shortly).

The `scanqtl` function, while completely flexible and so suitable for most purposes, is rather cumbersome to use. Our most important additions are the functions `addqtl`, to scan for a single QTL to be added to a multiple-QTL model, and `addpair`, to scan for a pair of QTL to be added to a multiple-QTL model. The output of these functions is of the forms produced by `scanone` and `scantwo`, respectively, and so one may use the corresponding plot and summary functions, making the results easier to deal with. For most purposes, these functions will be sufficient and direct calls to `scanqtl` will no longer be needed. Thus, we will not discuss the use of `scanqtl` further here.

The next important addition is `refineqtl`, which uses an iterative algorithm to refine the locations of QTL in a multiple-QTL model, with the aim of obtaining the maximum likelihood estimates of the QTL positions. If the function is called with `keeplodprofile=TRUE` (which is the default), one may use another new function, `plotLodProfile`, to plot the LOD profiles for each QTL, in the context of the multiple-QTL model, as is commonly used in multiple interval mapping.

The function `addint` may be used to test all possible pairwise interactions among the QTL in a multiple-QTL model.

We added several functions for manipulating a QTL object (created by `makeqtl`). The function `addtoqtl` is used to add additional QTL to an object, `dropfromqtl` is used to remove QTL from an object, `replaceqtl` is used to move a QTL to a new position, and `reorderqtl` is used to change the order the loci within the QTL object.

Finally, the function `stepwiseqtl` may be used to perform a stepwise search for the multiple-QTL model optimizing a penalized LOD score criterion. The function `calc.penalties` can

be used to calculate the penalties for `stepwise`, using the output of a permutation test with `scantwo`.

`makeqtl` and `fitqtl`

We'll look at the `hyper` data as an example. These data are from Sugiyama et al. (Genomics 71:70–71, 2001), and concern blood pressure in 250 male backcross mice. We'll use multiple imputation (the default), as Haley-Knott regression performs poorly in the case of selectively genotyping, which was used for the `hyper` data.

First we need to load the package and the data.

```
> library(qtl)
> data(hyper)
```

We will use the multiple imputation approach, and so we first run `sim.geno` to perform the imputations. We'll use 128 imputations; this is insufficient for the current data, which has extensive missing genotype information, but suffices to illustrate the methods. In practice, it is a good idea to repeat the analysis with independent imputations. If the results are much changed, increase the number of imputations.

```
> hyper <- sim.geno(hyper, step=2, n.draws=128, err=0.001)
```

Results of `scanone` and `scantwo`, which we won't revisit, indicated QTL on chromosomes 1, 4, 6 and 15, with an interaction between the QTL on chr 6 and 15, and the possibility of a second QTL on chr 1. We will begin by fitting this 4-QTL model. The function `makeqtl` is used to create a QTL object; it pulls out the imputated genotypes at the selected locations.

```
> qtl <- makeqtl(hyper, chr=c(1, 4, 6, 15), pos=c(67.3, 30, 60, 17.5))
```

Note that if you type the name of the QTL object, you get a brief summary.

```
> qtl

  QTL object containing imputed genotypes, with 128 imputations.

      name chr  pos n.gen
Q1  1@67.3   1 67.3     2
Q2  4@30.0   4 30.0     2
Q3  6@60.0   6 60.0     2
Q4 15@17.5  15 17.5     2
```

Also, there is a plot function for displaying the locations of the QTL on the genetic map. See Fig. 1.

```
> plot(qtl)
```

We may now use `fitqtl` to fit the model (with QTL in fixed positions). (In previous versions of R/qtl, `fitqtl` required a column of phenotypes, rather than a cross object, but this has been revised to make calls to `fitqtl` more like those to `scanone`, `scantwo`, and `scanqtl`.) Note that we use `Q3*Q4` to indicate that QTL 3 and 4 should interact. We could also have written the formula as `y~Q1+Q2+Q3+Q4+Q3:Q4`. See the help file for `formula` for more information.

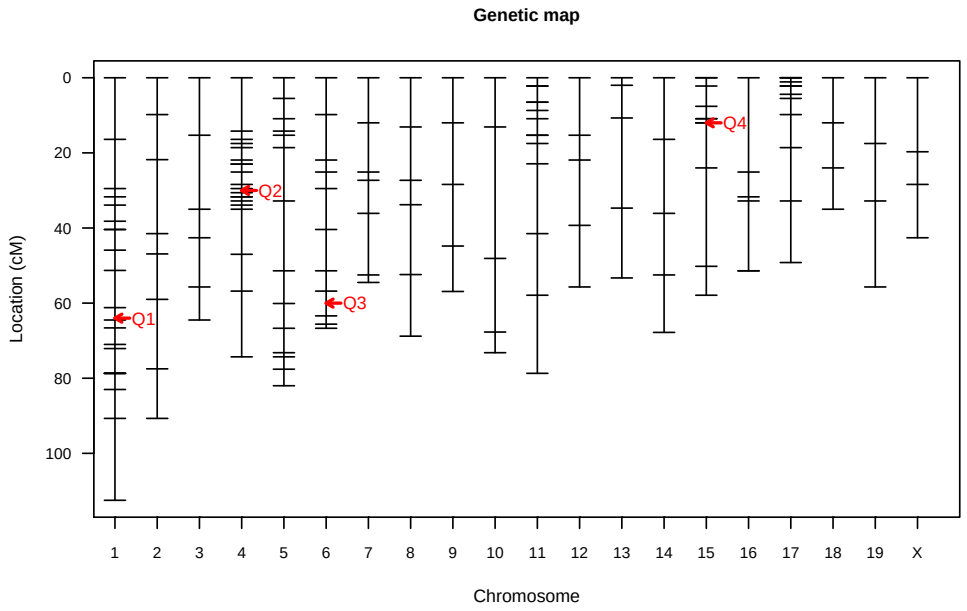

Figure 1: Locations of the QTL object `qtl` on the genetic map for the `hyper` data.

```
> out.fq <- fitqtl(hyper, qtl=qtl, formula=y~Q1+Q2+Q3*Q4)
> summary(out.fq)

                fitqtl summary

Method: multiple imputation
Model:  normal phenotype
Number of observations : 250

Full model result
----------------------------------
Model formula: y ~ Q1 + Q2 + Q3 + Q4 + Q3:Q4

       df    SS     MS  LOD %var Pvalue(Chi2) Pvalue(F)
Model   5  5842 1168.5 21.8 33.1            0         0
Error 244 11827   48.5
Total 249 17669


Drop one QTL at a time ANOVA table:
---------------------------------
                df Type III SS    LOD   %var F value Pvalue(Chi2) Pvalue(F)
1@67.3           1        1292  5.628  7.312   26.65            0  5.05e-07
4@30.0           1        2949 12.086 16.690   60.84            0  1.80e-13
6@60.0           2        1635  7.031  9.255   16.87            0  1.37e-07
15@17.5          2        1474  6.376  8.342   15.21            0  5.98e-07
6@60.0:15@17.5   1        1188  5.196  6.723   24.51            0  1.38e-06
```

The initial table indicates the overall fit of the model; the LOD score of 21.8 is relative to the null model (with no QTL). In the second table, each locus is dropped from the model, one at a time, and a comparison is made between the full model and the model with the term omitted. If a QTL is dropped, any interactions it is involved in are also dropped, and so the loci on chr 6 and 15 are associated with 2 degrees of freedom, as the 6×15 interaction is dropped when either of these QTL is dropped. The results indicate strong evidence for all of these loci as well as the interaction.

refineqtl and plotLodProfile

Let us explore the use of `refineqtl`, which allows us to get improved estimates of the locations of the QTL. We use `verbose=FALSE` to suppress the display of tracing information.

```
> rqtl <- refineqtl(hyper, qtl=qtl, formula=y~Q1+Q2+Q3*Q4, verbose=FALSE)
```

The output is a modified QTL object, with loci in new positions. We can type the name of the new QTL object to see the new locations.

```
> rqtl


  QTL object containing imputed genotypes, with 128 imputations.

      name chr  pos n.gen
Q1  1@67.8   1 67.8     2
Q2    4@30   4 30.0     2
Q3  6@66.7   6 66.7     2
Q4 15@17.5  15 17.5     2
```

The loci on chr 1 and 4 changed position very slightly. Let us use `fitqtl` to assess the improvement in fit; we'll skip the drop-one analysis.

```
> out.fq2 <- fitqtl(hyper, qtl=rqtl, formula=y~Q1+Q2+Q3*Q4, dropone=FALSE)
> summary(out.fq2)


                fitqtl summary

Method: multiple imputation
Model:  normal phenotype
Number of observations : 250

Full model result
----------------------------------
Model formula: y ~ Q1 + Q2 + Q3 + Q4 + Q3:Q4

       df    SS     MS  LOD %var Pvalue(Chi2) Pvalue(F)
Model   5  5893 1178.6 22.0 33.4            0         0
Error 244 11776   48.3
Total 249 17669
```

The LOD score comparing the full model to the null model has increased by 0.2, from 21.8 to 22.0.

If `refineqtl` is run with the argument `keeplodprofile=TRUE` (which is the default), the LOD traces at that last iteration are saved, which can then be plotted with `plotLodProfile`, as follows. See Fig. 2.

```
> plotLodProfile(rqtl)
```

The LOD profiles in Fig. 2 are similar to the usual LOD curves, but instead of comparing a model with a single QTL at a particular position to the null model, we compare, at each position for a given QTL, the model with the QTL of interest at that particular position (and with the positions of all other QTL fixed at their maximum likelihood estimates) to the model with the QTL of interest omitted (and, again, with the positions of all other QTL fixed at their maximum likelihood estimates). For the loci on chromosomes 6 and 15, the

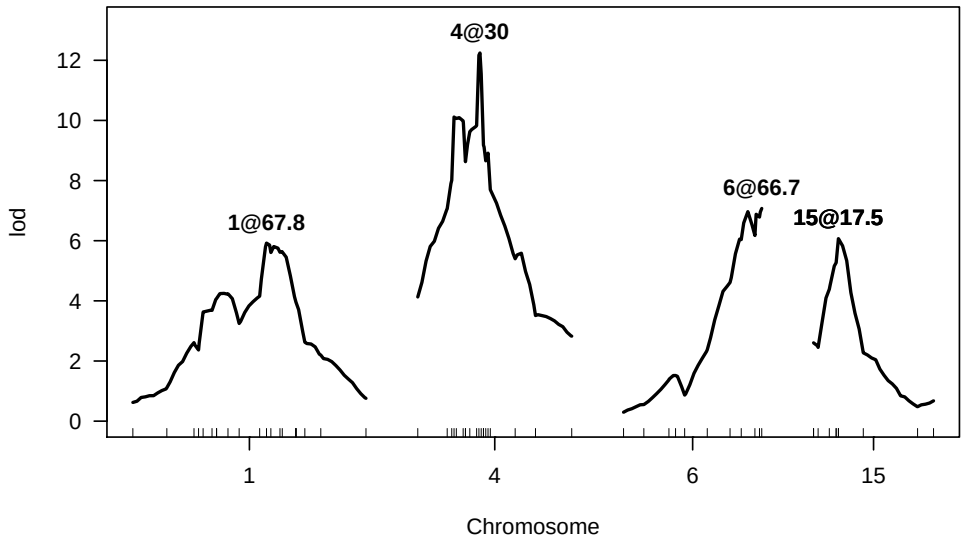

Figure 2: LOD profiles for a 4-QTL model with the `hyper` data.

6×15 interaction is omitted when either of the two loci is omitted. Note that maximum LOD for each of the LOD profiles should be the value observed in the drop-one analysis from `fitqtl`.

These profile LOD curves are useful for the assessment of both the evidence for the individual QTL and the precision of localization of each QTL, but note that they fail to take account of the uncertainty in the location of the other QTL in the model.

addint

The function `addint` is used to test, one at a time, all possible pairwise interactions between QTL that are not already included in a model. For our model with loci on chr 1, 4, 6 and 15, and with a 6×15 interaction, we will consider each of the 5 other possible pairwise interactions, and will compare the base model (with the four QTL and just the 6×15 interaction) to the model with the additional interaction included.

The syntax of the function is similar to that of `fitqtl`. The output is a table of results similar to that provided by the drop-one analysis of `fitqtl`.

```
> addint(hyper, qtl=rqtl, formula=y~Q1+Q2+Q3*Q4)


Method: multiple imputation
Model:  normal phenotype
Model formula: y ~ Q1 + Q2 + Q3 + Q4 + Q3:Q4

Add one pairwise interaction at a time table:
-----------------------------------------
               df Type III SS      LOD    %var F value Pvalue(Chi2) Pvalue(F)
1@67.8:4@30     1      61.381 0.283709 0.34739 1.27327        0.253     0.260
1@67.8:6@66.7   1       1.403 0.006468 0.00794 0.02896        0.863     0.865
```

```
1@67.8:15@17.5  1      71.145 0.328975 0.40265 1.47704        0.218     0.225
4@30:6@66.7     1      64.916 0.300094 0.36740 1.34701        0.240     0.247
4@30:15@17.5    1      16.226 0.074853 0.09183 0.33529        0.557     0.563
```

None of the interactions is particularly interesting.

Note the difference in the results if we use as the formula `y Q1+Q2+Q3+Q4` (that is, omitting the 6×15 interaction).

```
> addint(hyper, qtl=rqtl, formula=y~Q1+Q2+Q3+Q4)
```

```
Method: multiple imputation
Model:  normal phenotype
Model formula: y ~ Q1 + Q2 + Q3 + Q4

Add one pairwise interaction at a time table:
-----------------------------------------
               df Type III SS     LOD   %var F value Pvalue(Chi2)  Pvalue(F)
1@67.8:4@30     1       60.31 0.25455 0.3413  1.1468        0.279      0.285
1@67.8:6@66.7   1       11.63 0.04898 0.0658  0.2202        0.635      0.639
1@67.8:15@17.5  1       86.59 0.36589 0.4901  1.6501        0.194      0.200
4@30:6@66.7     1       39.82 0.16793 0.2253  0.7560        0.379      0.385
4@30:15@17.5    1       58.92 0.24870 0.3335  1.1204        0.285      0.291
6@66.7:15@17.5  1     1115.46 4.91316 6.3131 23.1130        0.000 0.00000264
```

The 6×15 interaction is also tested, and the LOD scores for the other interactions are somewhat different, as they concern comparisons between the 4-locus additive model and the model with that one interaction added.

addqtl

We may use the `addqtl` function to scan for an additional QTL, to be added to the model. By default, the new QTL is strictly additive.

```
> out.aq <- addqtl(hyper, qtl=rqtl, formula=y~Q1+Q2+Q3*Q4)
```

The output of `addqtl` has the same form as that from `scanone`, and so we may use the same summary and plot functions. For example, we can identify the genome-wide maximum LOD score with `max.scanone`.

```
> max(out.aq)
```

```
        chr  pos  lod
D5Mit31   5 66.7 1.58
```

And we may plot the results with `plot.scanone`; see Fig. 3.

```
> plot(out.aq)
```

We may also use `addqtl` to scan for an additional QTL, interacting with one of the current loci. This is done by including the additional QTL in the model formula, with the relevant interaction term. For example, let's scan for an additional QTL interacting with the chr 15 locus.

```
> out.aqi <- addqtl(hyper, qtl=rqtl, formula=y~Q1+Q2+Q3*Q4+Q4*Q5)
```

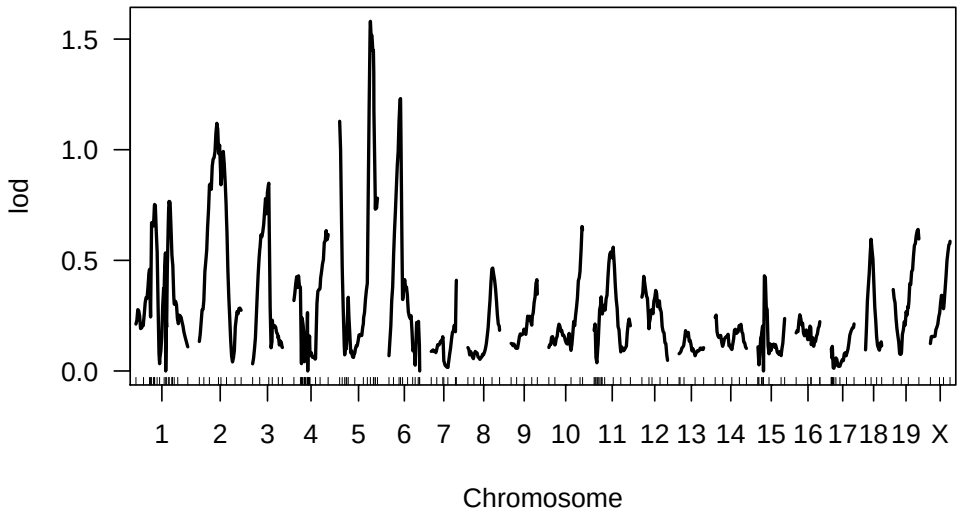

Figure 3: LOD curves for adding one QTL to the 4-QTL model, with the `hyper` data.

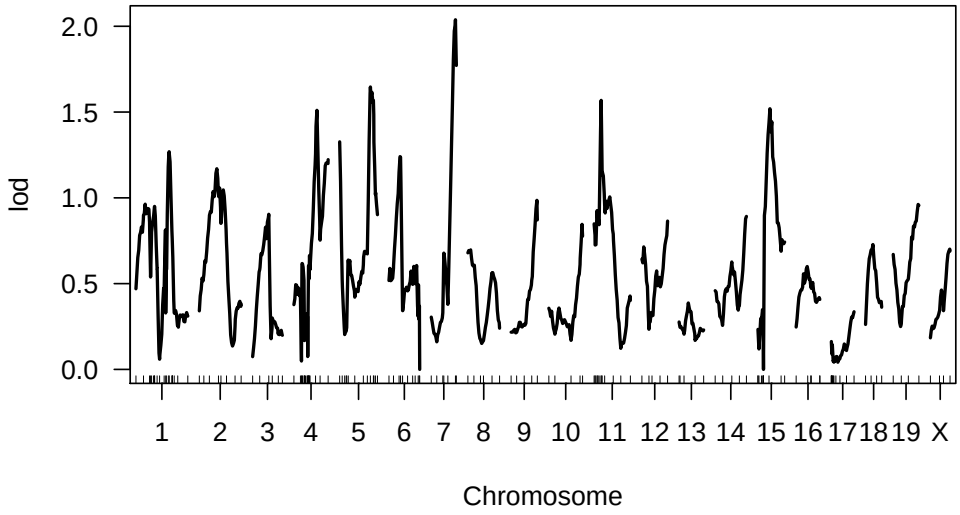

Figure 4: LOD curves for adding one QTL, interacting with the chromosome 15 locus, to the 4-QTL model, with the `hyper` data.

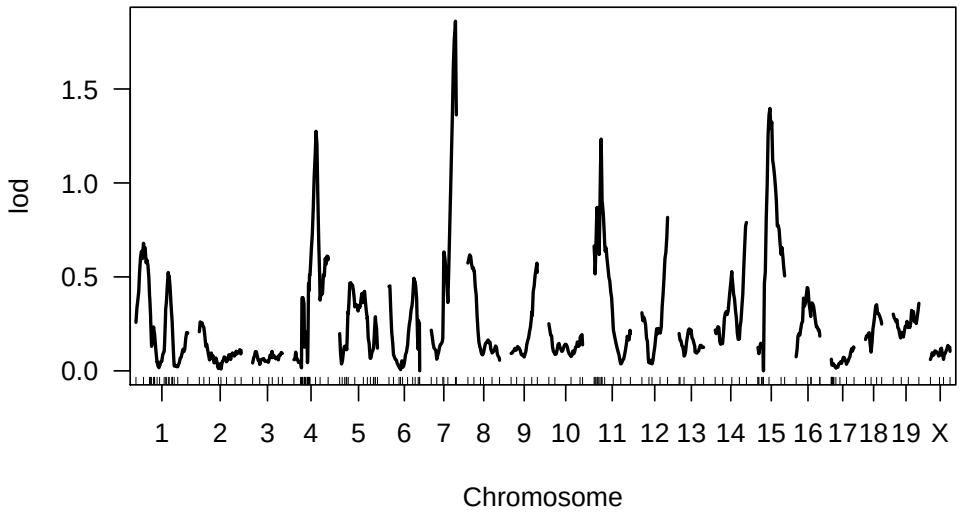

Figure 5: Interaction LOD curves in the scan for an additional QTL, interacting with the chromosome 15 locus, to the 4-QTL model, with the `hyper` data.

We plot the results as follows; see Fig. 4.

```
> plot(out.aqi)
```

Also of interest are the LOD scores for the interaction between the chr 15 locus and the new locus being scanned, which are the differences between the LOD scores in `out.aqi` and `out.aq`. See Fig. 5.

```
> plot(out.aqi - out.aq)
```

There is nothing particularly exciting in either of these plots.

addpair

The function `addpair` is similar to `addqtl`, but performs a two-dimensional scan to seek a pair of QTL to add. By default, `addpair` performs a two-dimensional scan analogous to that of `scantwo`: for each pair of positions for the two putative QTL, we fit both an additive model and a model including an interaction between the two QTL.

Recall that in the single-QTL analysis with the `hyper` data, there were two peaks in the LOD curve on chromosome 1, which indicates that there may be two QTL on that chromosome. In the context of our multiple-QTL model, the LOD profile on chromosome 1 (see Fig. 2) still shows two peaks, though the distal one is more prominent.

We may use `addpair` to investigate the possibility of a second QTL on chromosome 1. To do so, we omit the chr 1 locus from our formula, and perform a two-dimensional scan just on chromosome 1.

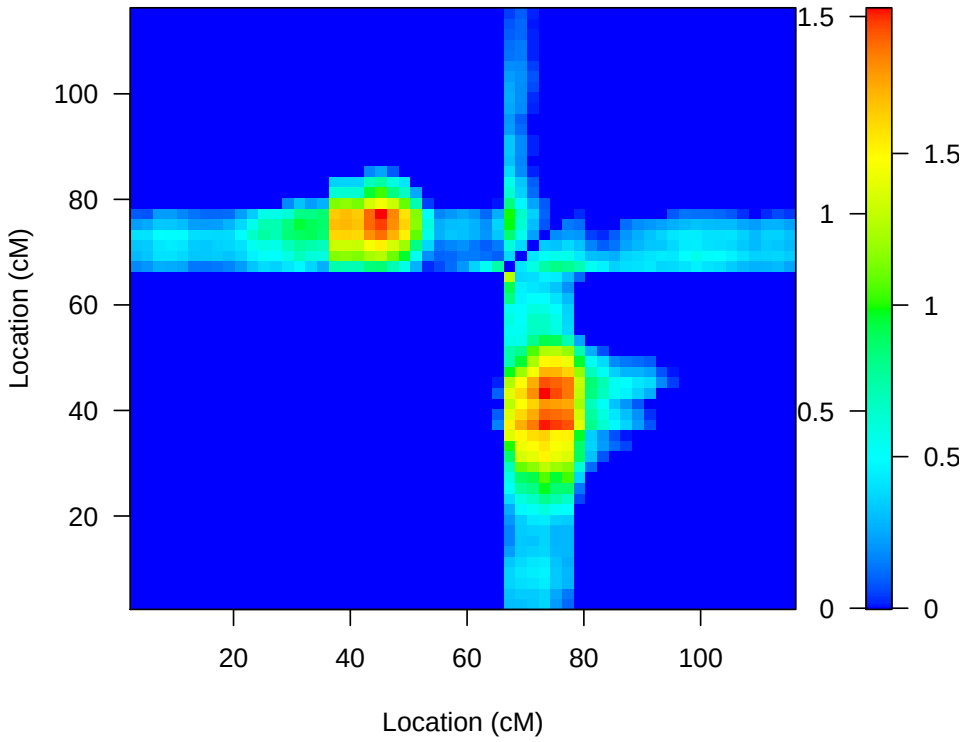

Figure 6: Results of a two-dimensional, two-QTL scan on chromosome 1, in the context of a model with additional QTL on chr 4, 6 and 15, and a 6×15 interaction, with the `hyper` data. LOD_{fv1} is in the lower-right triangle, and LOD_{av1} is in the upper-left triangle.

```
> out.ap <- addpair(hyper, qtl=rqtl, chr=1, formula=y~Q2+Q3*Q4, verbose=FALSE)
```

The output is of the same form as that produced by `scantwo`, and so we may use the same summary and plot functions.

```
> summary(out.ap)
```

```
      pos1f pos2f lod.full lod.fv1 lod.int     pos1a pos2a lod.add lod.av1
c1:c1  43.3  73.3     7.83    1.98   0.458      45.3  77.3    7.37    1.52
```

There is little evidence for an interaction, and the LOD score comparing the model with two additive QTL on chr 1 to that with a single QTL on chr 1 is 1.52, indicating relatively weak evidence for a second QTL on chr 1.

Let us also plot the results. We'll focus on the evidence for a second QTL on the chromosome, displaying LOD_{fv1} (evidence for a second QTL, allowing for an interaction) and LOD_{av1} (evidence for a second QTL, assuming to two are additive). See Fig 6.

```
> plot(out.ap, lower="cond-int", upper="cond-add")
```

There is a good deal of flexibility in the way that `addpair` may be used. As in `addqtl`, where one can scan for loci that interact with a particular locus in the model, we can use `addpair` to scan for an additional pair, with any prespecified set of interactions.

For example, we may retain the loci on chromosomes 1, 4 and 6, and scan for an additional pair of interacting loci, one of which also interacts with chromosome 6. This would be useful for assessing evidence for an additional QTL interacting with the chromosome 15 locus, but allowing the location of the locus on chromosome 15 to vary. We use the formula `y~Q1+Q2+Q3+Q5*Q6+Q3:Q5`, as we will omit the chr 15 locus (`Q4`), scan for an additional interacting pair (`Q5*Q6`), and allow the first QTL in the additional pair to interact with the chr 6 locus (`Q3`). Note that the positions of the chr 1, 4 and 6 loci are assumed known. A three-dimensional scan could be performed with the `scanqtl` function, but we won't try that here.

To save time, we will focus just on chromosomes 7 and 15.

```
> out.ap2 <- addpair(hyper, qtl=rqtl, formula=y~Q1+Q2+Q3+Q5*Q6+Q3:Q5, chr=c(7,15),
+                    verbose=FALSE)
```

Because we are using a special formula here, with one of the new QTL interacting with the chr 6 locus, the results are similar to, but not quite the same as, those from `scantwo`. Rather than fitting an additive and an interactive model at each pair of positions, we fit just the single model specified in the formula. And note that as the formula is not symmetric in `Q5` and `Q6`, we must do a full 2-dimensional scan, and not just on the triangle. (That is, we need `Q5` and `Q6` assigned to chromosomes (7,15) as well as (15,7).)

The summary of the results are somewhat different here. For each pair of chromosomes, a set of three LOD scores are presented. `lod.2v0` compares the full model to the model with neither of the two new QTL included, `lod.2v1b` compares the full model to the model with the first of the two new QTL omitted, and `lod.2v1a` compares the full model to the model with the second QTL omitted. When a QTL is omitted, any interactions involving that QTL is also omitted.

```
> summary(out.ap2)
```

```
        pos1a pos2a lod.2v0 lod.2v1b lod.2v1a
c7 :c7   29.1  25.1    2.89     2.54     1.55
c7 :c15  51.1  15.5    3.84     2.51     2.51
c15:c7   17.5  53.1    8.08     7.72     2.01
c15:c15  17.5  31.5    7.59     6.26     1.52
```

Note that, because of the lack of symmetry in the formula we used in `addpair`, separate results are provided for the two cases `c7:c15` (in which the chr 7 locus interacts with the chr 6 locus) and `c15:c7` (in which the chr 15 locus interacts with the chr 6 locus). The `c15:c7` row is most interesting, but `lod.2v1a` is 2.01, indicating little evidence for a chr 7 locus. (Note that `lod.2v1a` here concerns both the chr 7 locus and the 7×15 interaction.)

With this sort of `addpair` output, the `thresholds` argument should have length just 1 or 2 (which is different from the usual case for `summary.scantwo`). Rows will be retained if `lod.2v0` is greater than `thresholds[1]` and either of `lod.2v1a` or `lod.2v1b` is greater than `thresholds[2]`. (If a single thresholds is given, we assume that `thresholds[2]==0`.) Note that, of the other arguments to `summary.scantwo`, all but `allpairs` is ignored.

The plot of the output from `addpair`, in the case of a special formula, is also different from the usual `scantwo` plot.

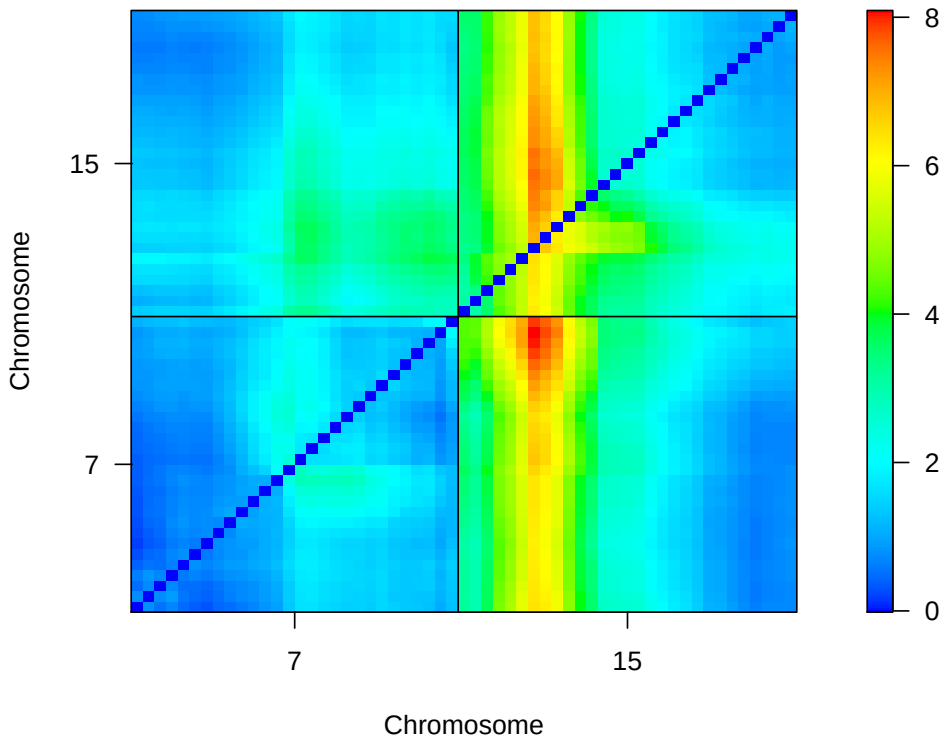

Figure 7: Results of a two-dimensional, two-QTL scan on chr 7 and 15, in the context of a model with additional QTL on chr 1, 4, and 6, with the `hyper` data. The two QTL being scanned were allowed to interact, and the first of them interacts with the chr 6 locus. The LOD scores displayed are for the 5-QTL model relative to the 3-QTL model. The x-axis corresponds to the first of the new QTL (which interacts with the chr 6 locus); the y-axis corresponds to the second of the new QTL.

```
> plot(out.ap2)
```

The plot, shown in Fig. 7, contains LOD scores comparing the full 5-QTL model to the 3-QTL model (having loci on chr 1, 4 and 6). The x-axis corresponds to the first of the new QTL (`Q5`), which is the one that interacts with the chr 6 locus. The y-axis corresponds to the second of the new QTL (`Q6`). Clearly, the QTL interacting with the chr 6 locus wants to be on chr 15.

Note that the `lower` and `upper` arguments to `plot.scantwo` are ignored in this case.

addtoqtl, dropfromqtl, and replaceqtl

Our analysis of the `hyper` data, above, did not indicate much evidence for any further QTL. If we had seen evidence for additional loci, we would want to add them to the QTL object and repeat our explorations with `fitqtl`, `addint`, `addqtl`, and `addpair`.

The functions `addtoqtl`, `dropfromqtl` and `replaceqtl` can be used to facilitate such analyses. Rather than re-creating the QTL object from scratch with `makeqtl`, one can use `addtoqtl` to add an additional locus to a QTL object that was previously created. For example, if we were satisfied with the evidence for an additional QTL on chr 1, it could be added to the QTL object `rqtl` as follows.

```
> rqtl <- addtoqtl(hyper, rqtl, 1, 43.3)
> rqtl

  QTL object containing imputed genotypes, with 128 imputations.

      name chr  pos n.gen
Q1  1@67.8   1 67.8     2
Q2    4@30   4 30.0     2
Q3  6@66.7   6 66.7     2
Q4 15@17.5  15 17.5     2
Q5  1@43.3   1 43.3     2
```

The syntax of `addtoqtl` is much like that of `makeqtl`, though one also provides the QTL object to which additional QTL are to be added.

If we want to move the first QTL on chromosome 1 to a different position (say to 73.3 cM rather than 67.8 cM), we may use `replaceqtl`, as follows.

```
> rqtl <- replaceqtl(hyper, rqtl, 1, 1, 73.3)
> rqtl

  QTL object containing imputed genotypes, with 128 imputations.

      name chr  pos n.gen
Q1  1@73.3   1 73.3     2
Q2    4@30   4 30.0     2
Q3  6@66.7   6 66.7     2
Q4 15@17.5  15 17.5     2
Q5  1@43.3   1 43.3     2
```

If we wish to reorder the QTL (e.g., according to their map positions), we may use `reorderqtl`, as follows. The argument `neworder` is to indicate the new order for the QTL. If missing, the QTL will be ordered by chromosome and position within a chromosome.

```
> rqtl <- reorderqtl(rqtl, c(5,1:4))
> rqtl


  QTL object containing imputed genotypes, with 128 imputations.

      name chr  pos n.gen
Q1  1@43.3   1 43.3     2
Q2  1@73.3   1 73.3     2
Q3    4@30   4 30.0     2
Q4  6@66.7   6 66.7     2
Q5 15@17.5  15 17.5     2
```

Finally, `dropfromqtl` is used to drop a locus from a QTL object.

```
> rqtl <- dropfromqtl(rqtl, 2)
> rqtl


  QTL object containing imputed genotypes, with 128 imputations.

      name chr  pos n.gen
Q1  1@43.3   1 43.3     2
Q2    4@30   4 30.0     2
Q3  6@66.7   6 66.7     2
Q4 15@17.5  15 17.5     2
```

stepwiseqtl

With the function `stepwiseqtl`, one may perform a forward/backward stepwise search algorithm find the multiple-QTL model optimizing a penalized LOD score criterion. The penalized LOD score for a model is the LOD score comparing the model to the null model (with no QTL), with a penalty subtracted for each main effect and separate penalties subtracted for each pairwise interactions among QTL.

We consider models with possible pairwise interactions among QTL but no higher-order interactions allowed. A hierarchy is enforced in which the inclusion of an interaction requires the inclusion of each of the corresponding main effects. Such a model may be represented by a graph in which vertices (dots) represent QTL and edges (line segments between the dots) represent interactions between QTL.

In the penalized LOD score considered by `stepwiseqtl`, we allow two penalties on interactions, a light penalty and a heavy penalty. Each disconnected component of a model is allowed one light interaction penalty; all other interactions are assigned the heavy penalty.

The three penalties may be calculed from permutation results with `scantwo`, using the function `calc.penalties`. We will use default penalties derived by computer simulation: (2.69, 2.62, 1.19) for a mouse backcross, or (3.52, 4.28, 2.69) for a mouse intercross. (The penalties are in the order (main, heavy interaction, light interaction).)

First, let us apply `stepwiseqtl`, considering only additive QTL models (with `additive.only=TRUE`. The algorithm performs forward selection up to a model with a given number of QTL (specified by the argument `max.qtl`; we'll use 6), followed by backward elimination.

```
> stepout1 <- stepwiseqtl(hyper, additive.only=TRUE, max.qtl=6,
+                         verbose=FALSE)
```

The output is a QTL object; type the name to view the chosen model.

```
> stepout1


  QTL object containing imputed genotypes, with 128 imputations.

     name chr  pos n.gen
Q1 1@67.8   1 67.8     2
Q2 4@29.5   4 29.5     2

  Formula: y ~ Q1 + Q2

  pLOD:  8.62
```

We obtain a model with two QTL, with one on each of chr 1 and 4.

Now let's re-run the analysis, allowing for the possibility of interactions among the QTL. If `stepwiseqtl` is called with `keeptrace=TRUE`, the sequence of models from the stepwise selection is retained as an attribute.

```
> stepout2 <- stepwiseqtl(hyper, max.qtl=6, keeptrace=TRUE,
+                         verbose=FALSE)
```

The chosen model contains QTL on chr 1, 4, 6 and 15, with the QTL on 6 and 15 interacting.

```
> stepout2


  QTL object containing imputed genotypes, with 128 imputations.

      name chr  pos n.gen
Q1  1@67.8   1 67.8     2
Q2    4@30   4 30.0     2
Q3  6@66.7   6 66.7     2
Q4 15@17.5  15 17.5     2

  Formula: y ~ Q1 + Q2 + Q3 + Q4 + Q3:Q4

  pLOD:  10.1
```

Since we called `stepwiseqtl` with the argument `keeptrace=TRUE`, the sequence of models visited by `stepwiseqtl` are retained as an *attribute* (called `"trace"`) of the output, `stepout2`. Attributes are a way of hiding additional information within an object. The entire set of attributes for an object may be obtained with the `attributes` function. It is often useful to just look at the names of the attributes.

```
> names(attributes(stepout2))


[1] "names"   "class"   "map"     "formula" "pLOD"    "trace"
```

Individual attributes may be obtained with the `attr` function. So we can pull out the trace of models with the following. This is a long list, with each component being a compact representation of a QTL model, and so we will print just the first of them.

```
> thetrace <- attr(stepout2, "trace")
> thetrace[[1]]
```

```
   chr  pos
Q1   4 29.5
  Formula: y ~ Q1
  pLOD:  5.4
```

It is nicer to look at a sequence of pictures rather than a long list of models. The function `plotModel` may be used to plot a graphical representation of a model, with nodes (i.e., dots) representing QTL and edges (i.e., line segments connecting two nodes) representing pairwise interactions among QTL. The argument `chronly` is used to print just the chromosome ID for each QTL (rather than chromosome and position). The penalized LOD score for each model is saved as an attribute, `"pLOD"`; we include them in the title of each subplot, but this requires another call to `attr`.

```
> par(mfrow=c(4,3))
> for(i in seq(along=thetrace))
+   plotModel(thetrace[[i]], chronly=TRUE,
+             main=paste(i, ": pLOD =",
+               round(attr(thetrace[[i]], "pLOD"), 2)))
```

As seen in Fig. 8, our chosen model is identified immediately (at step 4). Note that the model at step 3 (with additive QTL on chr 1, 4 and 6) has a lower penalized LOD score than the model at step 2 (with just the chr 1 and 4 QTL), but then the inclusion of the chr 15 QTL and the 6×15 interaction gave the largest penalized LOD score, among all models visited. With the addition of a QTL on chr 5 (at step 5), the pLOD decreased somewhat; the LOD score for the model increased, but not as much as the main effect penalty.

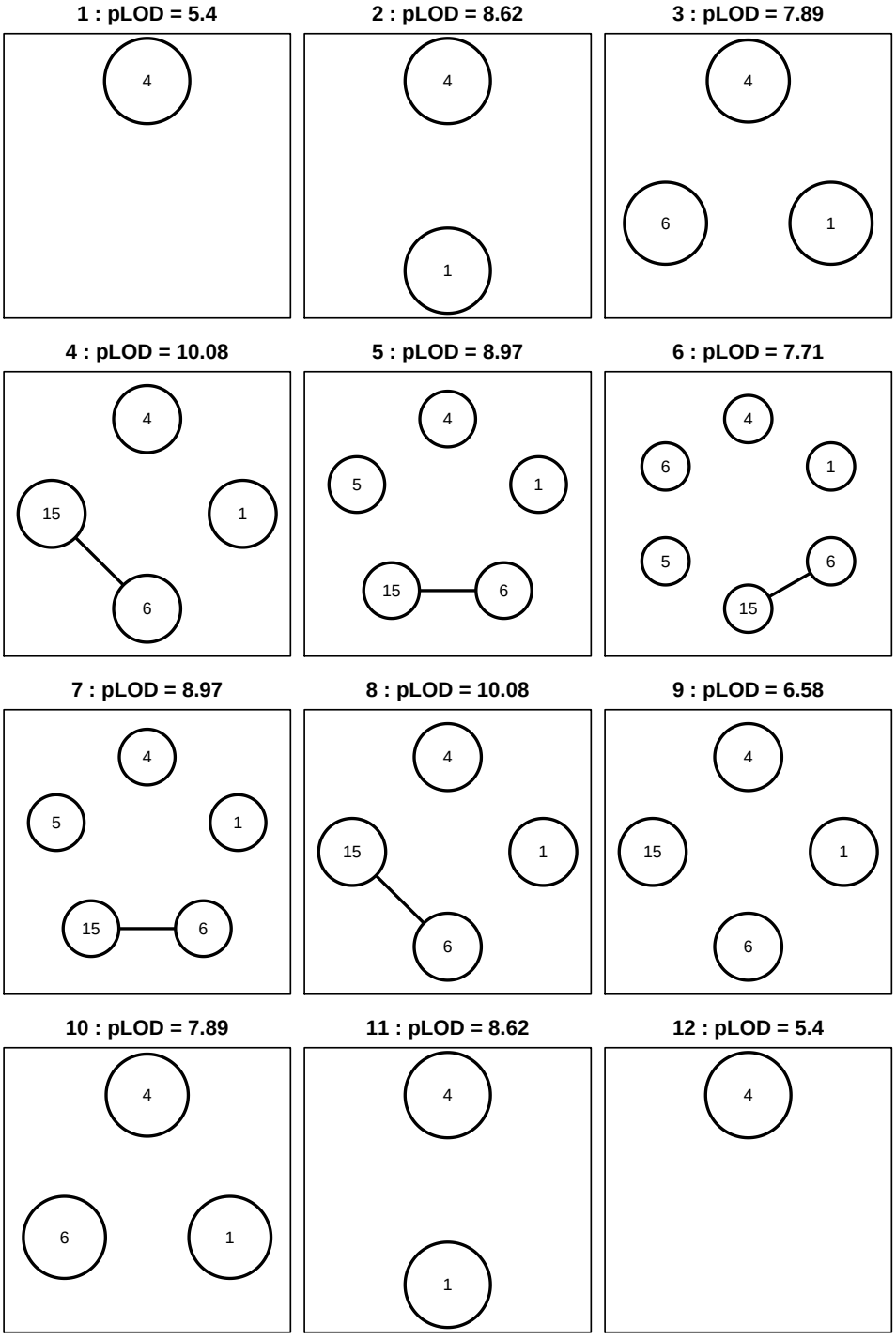

Figure 8: The sequence of models visited by the forward/backward search of `stepwiseqtl`, with the `hyper` data.