A tour of R/qtl

Karl W Broman

Department of Biostatistics, Johns Hopkins University
`http://biosun01.biostat.jhsph.edu/˜kbroman/software/qtl.html`

24 September 2001

Overview of R/qtl

The aim of R/qtl is to provide an extensible, interactive QTL mapping environment for the sophisticated user. We hope that this environment will allow the user to focus on modelling and exploratory data analysis, and be relieved of tedious data manipulation. A key component of R/qtl is code for the underlying hidden Markov model (HMM) engine, which is required for the proper allowance of missing genotype data in interval mapping and its extensions.

We chose to implement this QTL mapping environment as an add-on package for R, an open-source implementation of the S language (described below), for two reasons. First, we may save considerable programming effort by taking advantage of the extensive statistical and graphical capabilities of R. Second, the user will benefit by performing QTL analyses within a general statistical package, in that QTL mapping and data exploration may take place side-by-side. We also considered the use of Matlab, but chose R because (a) it is free, (b) we hope to eventually take advantage of its powerful modelling language, and (c) the structure and delivery of add-on packages, with documentation, is somewhat more slick.

R/qtl is very early in its development. Code for the HMM engine, with possible allowance for genotyping errors, is available for the backcross, intercross and four-way cross; extensions for other types of crosses should not be difficult. R/qtl also includes functions for the estimation of a genetic map for a given marker order, estimation of the recombination fractions between all pairs of markers, and calculation of the error LOD scores introduced by Lincoln and Lander (1992).

Only two QTL mapping procedures are currently available: standard interval mapping (Lander and Botstein 1989) and a variation suitable for a quantitative phenotype for which some individuals' trait values are undefined (such as the survival time following an infection, with some individuals failing to die). In the near future, we will implement the use of covariates in interval mapping, multiple interval mapping (Kao et al. 1999), and the pseudomarker algorithms (Sen and Churchill 2001). We expect to release a relatively complete version of R/qtl by June, 2002.

Overview of R

R is an open-source implementation of the S language. It is much like the commercial product S-PLUS, but is entirely free. (Access to the source code is valuable not just for developing extensions to R, but also for improving one's skills in statistical programming through the study of well-written code.) There are some minor differences in syntax and graphics, and some major differences in memory usage, between R and S-PLUS. The following description of R is taken from the R Project homepage (`http://www.R-project.org`). See also the Comprehensive R Archive, where one may obtain R and its extensions (`http://cran.r-project.org`).

> R is 'GNU S'—A freely available language and environment for statistical computing and graphics. R is similar to the award-winning S system, which was developed at Bell Laboratories by John Chambers et al. It provides a wide variety of statistical and graphical techniques (linear and nonlinear modelling, statistical tests, time series analysis, classification, clustering, . . .).
>
> R is designed as a true computer language with control-flow constructions for iteration and alternation, and it allows users to add additional functionality by defining new functions. For computationally intensive tasks, C, C++ and Fortran code can be linked and called at run time.

License

R/qtl (and R) are released under the GNU Public License. This means they are entirely free, their source code is freely available, and any software which incorporates any portion of them must also be subject to these terms.

Current R/qtl functions

Sample data	fake.4way	Simulated data for a 4-way cross experiment
	fake.bc	Simulated data for a backcross experiment
	fake.f2	Simulated data for an F_2 intercross experiment
	listeria	Intercross data on Listeria monocytogenes susceptibility
	hyper	Backcross data on salt-induced hypertension
Input/output	read.cross	Read data for a QTL experiment
	read.cross.karl	Read data for a QTL experiment in Karl Broman's format
	read.cross.gary	Read data for a QTL experiment in Gary Churchill's format
	read.cross.csv	Read data for a QTL experiment in comma-delimited format
	read.cross.mm	Read data for a QTL experiment in Mapmaker format
	write.cross	Write data for a QTL experiment to a file
	write.cross.csv	Write data in comma-delimited format
	write.cross.mm	Write data in Mapmaker format
Simulation	sim.cross	Simulate a QTL experiment
	sim.map	Simulate a genetic map
	drop.qtlgeno	Remove QTL genotypes from simulated data
Summaries	summary.cross	Print summary of QTL experiment
	plot.cross	Plot various features of a cross object
	plot.missing	Plot grid of missing genotypes
	nind	Number of individuals
	nchr	Number of chromosomes
	nphe	Number of phenotypes
	nmar	Number of markers per chromosome
	totmar	Total number of markers
	geno.table	Create table of genotype distributions
Data manipulation	clean	Remove intermediate calculations from a cross
	pull.chr	Pull out a subset of chromosomes from a cross
	pull.map	Pull out the genetic map from a cross
	drop.markers	Remove a list of markers
	drop.nullmarkers	Remove markers without data
	replace.map	Replace the genetic map of a cross
HMM engine	calc.genoprob	Calculate conditional genotype probabilities
	sim.geno	Simulate genotypes given observed marker data
	argmax.geno	Reconstruct underlying genotypes
Interval mapping	scanone	Genome scan with single QTL model
	scanone.perm	Permutation test for genome scan with single QTL model
	summary.scanone	Print summary of the scanone output
	plot.scanone	Plot output for a single QTL scan
	vbscan	Genome scan for trait with some undefined values
	vbscan.perm	Permutation test for trait with some undefined values
Genetic mapping	est.map	Estimate genetic maps
	est.rf	Estimate pairwise recombination fractions
	plot.map	Plot genetic map(s)
	plot.rf	Plot recombination fractions
Genotyping errors	calc.errorlod	Calculate Lincoln & Lander (1992) error LOD scores
	find.errors	Compare observed genotypes to results of argmax.geno
	plot.errorlod	Plot grid of error LOD values
	plot.errors	Plot the results of find.errors
	top.errorlod	List genotypes with highest error LOD values
	plot.geno	Plot genotypes with indication of likely errors

Example 1

Use of the R/qtl package requires considerable knowledge of the R language/environment. We hope that the examples presented here will be understandable without any prior knowledge of R or Splus, especially because we neglect to explain the syntax of R. Several books, as well as some free documents, are available to assist the user in learning R; see the R project websites cited above. We assume here that the user is running a version of the Windows operating system.

1. To start R, double-click its icon.

2. To exit, type:

   ```
   q()
   ```

   Click yes or no to save or discard your work.

3. View the objects in your workspace:

   ```
   ls()
   ```

4. View the available libraries/packages:

   ```
   library()
   ```

5. View the functions/data in the R/qtl package:

   ```
   library(help=qtl)
   ```

6. Load the R/qtl package:

   ```
   library(qtl)
   ```

7. One of the most difficult steps in the use of any statistical software is the import of data. In R/qtl, one may use the function `read.cross` to import data on a QTL mapping experiment. Several different data formats are allowed. We'll skip over this here, but let us see how to view the documentation on the `read.cross` function.

   One way to view the help file: click (in the menu bar) Help → R language (html) → Packages → qtl → read.cross.

   Alternatively, type one of the following:

   ```
   help(read.cross)
   ?read.cross
   ```

   The html version of the help files are somewhat easier to read, and allow use of hotlinks between different functions. Typing the following will make the above text commands open the html version of the help files.

   ```
   options(htmlhelp=TRUE)
   ```

8. Get access to some sample data. This data is taken from Boyartchuk et al. (Nat Genet 27:259-260, 2001), and is kindly provided by Victor Boyartchuk and Bill Dietrich.

   ```
   data(listeria)
   ```

9. Check that it is now in your workspace:

   ```
   ls()
   ```

10. Get summary information on this cross:

    ```
    summary(listeria)
    nind(listeria)
    nphe(listeria)
    nchr(listeria)
    totmar(listeria)
    nmar(listeria)
    ```

11. Plot a summary of these data.

    ```
    plot(listeria)
    ```

    In the upper left, black pixels indicate missing genotype data; gray pixels are partially missing genotypes (e.g., a genotype may be known to be either AA or AB, but not which). Note that some markers have no genotype data. In the upper right, the genetic map of the markers is shown. In the lower left, a histogram of the phenotype is shown. The phenotype here is the survival time of a mouse (in hours) following infection with *Listeria monocytogenes*. Individuals with a survival time of 264 hours are those that recovered from the infection.

12. The Windows version of R has a slick method for recording graphs, so that one may page up and down through a series of plots. To initiate this, click (on the menu bar) History → Recording.

13. View the components of the above plot individually.

    ```
    plot.missing(listeria)
    plot.map(listeria)
    hist(listeria$pheno,breaks=30)
    ```

14. Let us briefly discuss the rather complex data structure that R/qtl uses for QTL mapping experiments. This may be rather dull or confusing; you might want to skip to the next item.

    First, these objects have a "class," which indicates that it corresponds to data for an experimental cross, and gives the cross type. By having class `cross`, the function `plot` knows to plot the data using the function `plot.cross`.

    ```
    class(listeria)
    ```

    Every `cross` object has two components, one containing the genotype data and genetic maps and the other containing the phenotype data.

    ```
    names(listeria)
    ```

    The phenotype data is simply a matrix with rows corresponding to individuals and columns corresponding to phenotypes.

    ```
    listeria$pheno
    ```

    The genotype data is split into a number of components, one per chromosome. Each component contains the genotype data and the genetic map for the corresponding chromosome.

    ```
    names(listeria$geno)
    names(listeria$geno[[3]])
    ```

15. Recall that some markers had no genotype data. It might be best to simply remove these:

    ```
    listeria <- drop.nullmarkers(listeria)
    plot.missing(listeria)
    ```

16. Re-estimate the genetic map (keeping the order of markers fixed), and plot the original map against the newly estimated one.

    ```
    newmap <- est.map(listeria)
    plot.map(listeria,newmap)
    plot.map(listeria,newmap,horiz=TRUE)
    ```

    At this point we require that the marker order be known. Eventually we would like to include facilities for determining marker order, or at least for investigating changes in marker order, but we consider this to be a low priority item.

17. We may also estimate the map allowing for the presence of genotyping errors.

    ```
    newmap2 <- est.map(listeria,error.prob=0.01)
    plot(newmap,newmap2)
    ```

18. If we wish, we may replace the genetic map within the `listeria` object with the newly estimated one:

    ```
    listeria <- replace.map(listeria,newmap)
    ```

19. Estimate recombination fractions between all pairs of loci, and plot them. This also calculates a LOD score for the test of H_0: $r = 1/2$. The plot of the recombination fractions can be either with recombination fractions in the upper part and LOD scores below, or with just recombination fractions or just LOD scores. Note that white indicates a small recombination fraction or a big LOD score, while red indicates a large recombination fraction or a small LOD score.

    ```
    listeria <- est.rf(listeria)
    names(listeria)
    plot.rf(listeria)
    plot.rf(listeria,c(5,13))
    plot.rf(listeria,c(5,13),"rf")
    plot.rf(listeria,c(5,13),"lod")
    ```

    This is used largely as a diagnostic, to identify markers that have been misplaced (such as being placed on the wrong chromosome).

20. We now turn to the identification of genotyping errors. In the following, we calculate the error LOD scores of Lincoln and Lander (1992). A LOD score is calculated for each individual at each marker; large scores indicate likely genotyping errors.

    The core of R/qtl is a set of functions which make use of the hidden Markov model (HMM) technology to calculate QTL genotype probabilities conditional on the observed marker data, to calculate the most likely sequence of genotypes given the observed marker data, and to simulate from the joint genotype distribution, conditional on the observed marker data. This is done in a quite general way, with possible allowance for the presence of genotyping errors. Of course, we assume no crossover interference. The function `calc.genoprob` performs the first of these; the values are used in the calculation of the error LOD scores.

    ```
    listeria <- calc.genoprob(listeria,error.prob=0.01)
    names(listeria$geno[[1]])
    listeria <- calc.errorlod(listeria,error.prob=0.01)
    names(listeria$geno[[1]])
    ```

    We may now make various plots and receive other summary information regarding which genotypes are likely in error.

    ```
    plot.errorlod(listeria)
    plot.errorlod(listeria,c(5,13))
    top.errorlod(listeria)
    ```

    Nothing here is particularly compelling.

21. The following display the actual genotype data, one chromosome at a time, with possible genotyping errors flagged by red squares. Of course, it's difficult to look at too many individuals at once.

    ```
    plot.geno(listeria,13)
    plot.geno(listeria,13,41:70)
    ```

    Note that white = AA, gray = AB, black = BB, green = AA or AB, and orange = AB or BB.

22. We now, finally, get to interval mapping. We re-calculate the QTL genotype probabilities (at 1 cM steps, going 5 cM past the terminal markers), and then do a genome scan by interval mapping.

    ```
    listeria <- calc.genoprob(listeria,step=1,off.end=5)
    out <- scanone(listeria)
    ```

    We get a warning message; at one position in the genome, the EM algorithm did not converge to within the specified tolerance within the specified maximum number of iterations. Use the function `args` to see the arguments to the `scanone` function. (This is useful to get quick though extremely brief information about a function.) Run `scanone` again with a larger value for `maxit`, and note that the warning message no longer appears.

    ```
    args(scanone)
    out <- scanone(listeria,maxit=5000)
    ```

The following gives plots of the LOD curves and summaries of the peak LOD on each chromosome.

```
plot(out)
plot(out,chr=3)
summary(out)
summary(out,threshold=3)
```

Note the NAs in the BB means for the X chromosome; in an intercross, the X chromosome is either not segregating, or is segregating like a backcross.

More importantly, note that something very strange is going on with chromosome 3, in a large gap between markers. Recall the distribution of this phenotype; around 30% of the mice had a survival time of 264 hours, indicating that they had recovered from the infection and failed to die. Interval mapping, in the presence of limited genotype information, is choking.

23. The paper describing these data (Boyartchuk et al. 2001) used a modified form of interval mapping, appropriate for this type of phenotype. A two-part model was considered, assuming a single QTL. A mouse with QTL genotype g has probability p_g of surviving the infection. If the mouse dies, its survival time (or perhaps its log survival time) follows a normal distribution with mean μ_g and SD σ (independent of genotype). A LOD score is formed to test the hypothesis H_0: $p_g \equiv p$ and $\mu_g \equiv \mu$.

    The function `vbscan` performs this analysis. ("vb" stands for Victor Boyartchuk; a more appropriate name didn't come to mind, and so for now it is named for the scientist who presented us with the data.) The argument `upper` indicates that the maximum phenotype is to be considered as the undefined value.

    ```
    out2 <- vbscan(listeria,upper=TRUE)
    plot(out2)
    plot(out2,chr=c(1,5,13))
    plot(out2,out,chr=c(1,5,13))
    summary(out2)
    summary(out2,threshold=5)
    ```

24. For this model, we also calculate separate LOD scores testing the hypotheses H_1: $p_g \equiv p$ and H_2: $\mu_g \equiv \mu$. We can plot all three LOD curves plotted together.

    ```
    plot(out2,out2[,-3],out2[,-(3:4)],chr=c(1,5,13))
    ```

    Note that the peak on chromosome 1 is due largely to differences in the means, the peak on chromsome 5 is due largely to differences in the proportions, and the peak on chromosome 13 is due to differences in both.

25. Note that because this model fits three additional parameters, the 5% genome-wide LOD threshold will be considerably higher. Permutation tests are useful for estimating an appropriate LOD threshold, but you probably won't want to sit and wait for 1000 permutation replicates.

    ```
    perm2 <- vbscan.perm(listeria,upper=TRUE,n.perm=2)
    perm2
    ```

26. Repeat the analysis using the log survival time as the phenotype.

    ```
    listeria$pheno <- cbind(listeria$pheno,log.surv=log(listeria$pheno[,1]))
    out3 <- vbscan(listeria, pheno.col=2, upper=TRUE)
    summary(out3)
    summary(out3,threshold=4)
    plot(out2,out3,chr=c(1,5,13))
    ```

27. You may wish to remove all intermediate calculations from the `listeria` data:

    ```
    listeria <- clean(listeria)
    ```

28. You may also wish to clean out your workspace.

    ```
    ls()
    rm(list=ls())
    ```

Example 2

1. Get access to a second set of sample data. This data is taken from Sugiyama *et al.* (Genomics 71:70-77, 2001), and is kindly provided by Bev Paigen and Gary Churchill.

   ```
   data(hyper)
   summary(hyper)
   plot(hyper)
   ```

2. Note the odd patter of missing data; we may make this missing data plot with the individuals ordered according to the value of their phenotype.

   ```
   plot.missing(hyper)
   plot.missing(hyper,reorder=TRUE)
   ```

   We see that, for most markers, only individuals with extreme phenotypes were genotyped. At many markers (in regions of interest), markers were typed only on recombinant individuals.

3. Re-estimate the genetic map.

   ```
   newmap <- est.map(hyper,error.prob=0.01)
   plot.map(hyper,newmap)
   hyper <- replace.map(hyper,newmap)
   ```

   We see some map expansion, especially on chromosomes 6, 13 and 18. It is questionable whether we should replace the map or not. Keep in mind that the previous map locations are from the MIT maps, based on a very limited number of meioses.

4. Estimate all pairwise recombination fractions.

   ```
   hyper <- est.rf(hyper)
   plot.rf(hyper)
   plot.rf(hyper,c(1,4))
   ```

   Note that the gray values are missing; one marker is not typed at any individuals. There are some very strange patterns in the recombination fractions, but this is due to the fact that some markers were typed only on recombinant individuals.

   For example, on chr 6, the 9th marker shows a high recombination fraction with all other markers on the chromosome, but a plot of the missing data shows that this marker was typed only on a selected number of individuals (those showing recombination events across the interval).

   ```
   plot.rf(hyper,6)
   plot.missing(hyper,6)
   ```

5. Calculate the error LOD scores.

   ```
   hyper <- calc.genoprob(hyper,error.prob=0.01)
   hyper <- calc.errorlod(hyper,error.prob=0.01)
   top.errorlod(hyper)
   ```

   Note likely errors in individuals 102, 107 and 216 on chr 4.

   ```
   plot.errorlod(hyper)
   plot.errorlod(hyper,4)
   ```

6. Plot the genotypes for these possible errors.

   ```
   plot.geno(hyper,4)
   plot.geno(hyper,4,c(101:110,211:220),min.sep=3)
   ```

   We haven't figured out how best to indicate the individual numbers in this plot yet. The first ten individuals here are 101-110; the second ten are 211-220. Note the tight apparent double crossovers.

7. Perform interval mapping.

```
hyper <- calc.genoprob(hyper,step=1,off.end=10)
out <- scanone(hyper)
summary(out,threshold=2)
plot(out)
plot(out,chr=c(1,4,15))
```

8. Perform a few reps of a permutation test.

```
perm <- scanone.perm(hyper,n.perm=2)
perm
```