Bayesian **M**ultiple **QTL** Mapping

Samprit Banerjee, Brian S. Yandell, Nengjun Yi

April 28, 2006

1 Overview

Bayesian multiple mapping of QTL library R/bmqtl provides Bayesian analysis of multiple quantitative trait loci (QTL) models. This includes posterior estimates of the number and location of QTL, and of their main and epistatic effects. This document assumes some familiarity with QTL and with Bayesian methods. In addition it provides graphical diagnostics that can help investigate several "better" models. It also provides a 1-D and 2-D genome scan. Library R/bmqtl requires R/qtl.

```
> library(bmqtl)
```

```
Loading required package: qtl
```

2 Bayesian QTL Mapping

This document tries to walk you through the `R/bmqtl` package by demonstrating the following major functions.

2.1 Data Simulation

R/bmqtl has an inbuilt function `sim.data` to simulated a backcross or F2 data set of class `cross` (see R/qtl help pages for details). The following chunk of code would generate a data set of 100 individuals of F2 mating design. These individuals are genotyped for 11 not equally spaced markers on 20 chromosomes. There are 7 QTLs, two on chromosome 1 and one each on chromosomes 3,5,7,10 and 19. QTL numbers 1,3 and 4 have additive main effects of 0.5, -0.5 and 0.5 and numbers 2 and 4 have dominant main effects of 0.5 and -0.5. QTL numbers 4 and 5 have an additive-additive interaction of -0.7 and numbers 6 and 7 have an additive-dominant interaction of 1.2. Two covariates, a binary fixed covariate and an ordinal random are generated with their corresponding coefficients as 0.5 and 0.07. G x E (gene x environemt) interaction is also considered with the fixed covariate. A normal phenotype and an ordinal phenotype with 3 categories are measured. 7% of the genotypes are randomly missing.

```
> cross <- sim.data(len = rep(100, 20), n.mar = 11, eq.spacing = F,
+     n.ind = 100, type = "f2", ordinal = c(0.3, 0.3, 0.2, 0.2),
+     missing.geno = 0.03, missing.pheno = 0.07, qtl.pos = rbind(c(1,
+         15), c(1, 45), c(3, 12), c(5, 15), c(7, 15), c(10, 15),
+         c(12, 35), c(19, 15)), qtl.main = rbind(c(1, 0.5, 0),
+         c(2, 0, 0.7), c(3, -0.5, 0), c(4, 0.5, -0.5)), qtl.epis = rbind(c(4,
+         5, -0.7, 0, 0, 0), c(6, 8, 0, 1.2, 0, 0)), covariate = c(0.5,
+         0.07), gbye = rbind(c(7, 0.8, 0)))
```

By using the function `sim.data` a list is attached to cross object named ”qtl”.

```
> names(cross)
```

```
[1] "geno"  "pheno" "qtl"
```

The `cross$plot` contains information about the true values which can be compared to after the analysis.

```
> str(cross$qtl)
```

```
List of 6
 $ geno      : int [1:100, 1:8] 3 2 1 2 3 2 2 3 3 1 ...
 $ pos       : num [1:8, 1:2] 1 1 3 5 7 10 12 19 15 45 ...
 $ herit.main: num [1:4, 1:3] 1.000 2.000 3.000 4.000 0.062 ...
 $ herit.epis: num [1:2, 1:6] 4.0000 6.0000 5.0000 8.0000 0.0607 ...
 $ herit.cov : num [1:2] 0.0310 0.0347
 $ herit.gbye: num [1, 1:3] 7.0000 0.0397 0.0000
```

The summary of the cross object looks like the following.

```
> summary(cross)
```

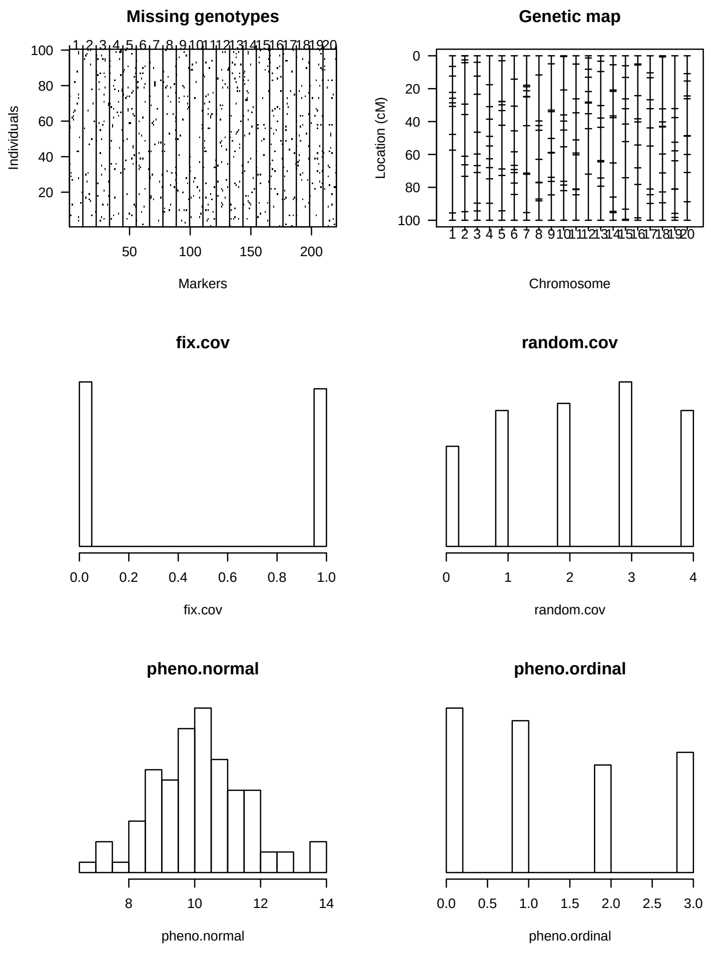
```
    F2 intercross

    No. individuals:  100

    No. phenotypes:   4
    Percent phenotyped:  94 95 93 86

    No. chromosomes:  20
    Total markers:    220
    No. markers:      11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11 11
    Percent genotyped:  97
    Genotypes (%):      AA:25.4  AB:49.5  BB:25  not BB:0  not AA:0
```

```
> plot(cross)
```



2.2 Genotype Probabilites

The function `bmq.genoprob` would create a grid of putative QTL positions throughout the entire genome and compute the genotypic probabilities of the same. It would also compute genotypic probabilities for missing markers. In general, the genotypes at different loci are unobserved except at completely informative markers but the probability distribution can be inferred from the observed marker data using the multipoint method (Jiang and Zeng 1997).

```
> cross <- bmq.genoprob(cross, map.func = "Haldane", step = 5)
```

2.3 Specifying data

The function `bmq.data` specifies the traits to be analyzed, their underlying distribution, the random and/or fixed covariates and whether to standardize or to use a boxcox transformation. Note that, the cross object can have several phenotypes and some of which could be used as covariates.

```
> data <- bmq.data(cross, pheno.col = 3, trait = "normal", rancov = 2,
+     fixcov = 1, boxcox = F, standardize = F)
```

2.4 Defining the model

The function `bmq.model` defines the model which is a Cockerham epistatic model. For mapping a *K+1* genotypes per loci, there are K main effects and K^2 epistatic effects. For a backcross population with two segregating genotypes $b_q b_q$ and $B_q b_q$ at locus q the coefficients are

$x_{iq1} = z_{iq} - 0.5$ and $x_{iqq'1} = x_{iq1} x_{iq'1}$

where z_{iq} denotes the number of b_q alleles. For an intercross derived from two inbred lines, there are three segregating genotypes $b_q b_q$, $b_q B_q$ and $B_q B_q$. The coefficients of the Cockerham model are as follows:

$$x_{iq1} = z_{iq}+1,\ x_{iq2} = (1-x_{iq1})(1+x_{iq1})-0.5 \text{ and } x_{iqq'1} = \begin{cases} x_{iq1}x_{iq'1} & k=1 \\ x_{iq1}x_{iq'2} & k=2 \\ x_{iq2}x_{iq'1} & k=3 \\ x_{iq2}x_{iq'2} & k=4 \end{cases}$$

It also specifies if epistasis is considered, the expected number of main effect QTLs (`main.qtl`), the prior number of total QTLs on all chromosome which includes QTLs with only epistatic effect, the maximum number of QTLs allowed per chromosome. The maximum number of QTLs allowed per chromosome has a default of $l_0+3\sqrt{l_0}$ where l_0 is `main.qtl` in a non-epistatic and `mean.qtl` in an epistatic model. The interval between two flanking QTLs for each chromosome and the fixed covariate(s) interacting with QTLs are also specified. Typically a real data set has several traits which can be considered as covariates. By setting the `max.qtl=0` (the maximum number of QTLs allowed) a traditional Bayesian model selection can be performed. This would provide a more relevant and shorter set of covariates. This set of covariates can then be used to for QTL mapping. For our case as we have only a couple of covariates we would go ahead and include both in the model. The `main.qtl` argument of the `bmq.model` function refers to the prior number of main effect QTL. This is obtained generally from a previous rudimentary non-Bayesian analysis. For example, R/qtl can be used to analyze this data without epistasis and the number of QTLs detected could be used as the prior number of QTLs.

```
> model <- bmq.model(cross, epistasis = TRUE, main.nqtl = 3, interval = rep(5,
+     nchr(cross)), chr.nqtl = rep(2, nchr(cross)), depen = FALSE,
+     prop = c(0.5, 0.1, 0.05), intcov = 1)
```

2.5 Running MCMC

The function `bmq.mcmc` would run MCMC on the data and model specified. The results can be saved in a specified directory ”C:/package”. The `genoupdate` parameter of the `bmq.mcmc` function can be switched on when there are too many missing marker genotypes. For our case, we have only 7% of the marker genotypes missing so we are not updating the genotypes. There are two choices of the MCMC algorithm namely, Metropolis-Hastings algorithm and the Gibbs sampler. The M-H algorithm samples from a proposal density approximating the conditional posterior distributions whereas Gibbs sampler samples directly from the conditional posterior distribution. The M-H algorithm is a lot faster compared to the Gibbs sampler.

```
> result <- bmq.mcmc(cross, data, model, main.qtl = 3, n.iter = 2000,
+     n.thin = 10, algorithm = "M-H", genoupdate = F, )
```

```
Bayesian MCMC run in progress. The current saved iterations:
200
400
600
800
1000
1200
1400
1600
1800
2000
MCMC sample has been saved to: ./pheno.normal_Apr-28-151554.
Bayesian MCMC took 0.38 minutes.
```

2.6 Creating MCMC object for R/CODA

The following function would convert the object of class `bim` to an object of class `mcmc` for posterior analysis using `R/CODA` .

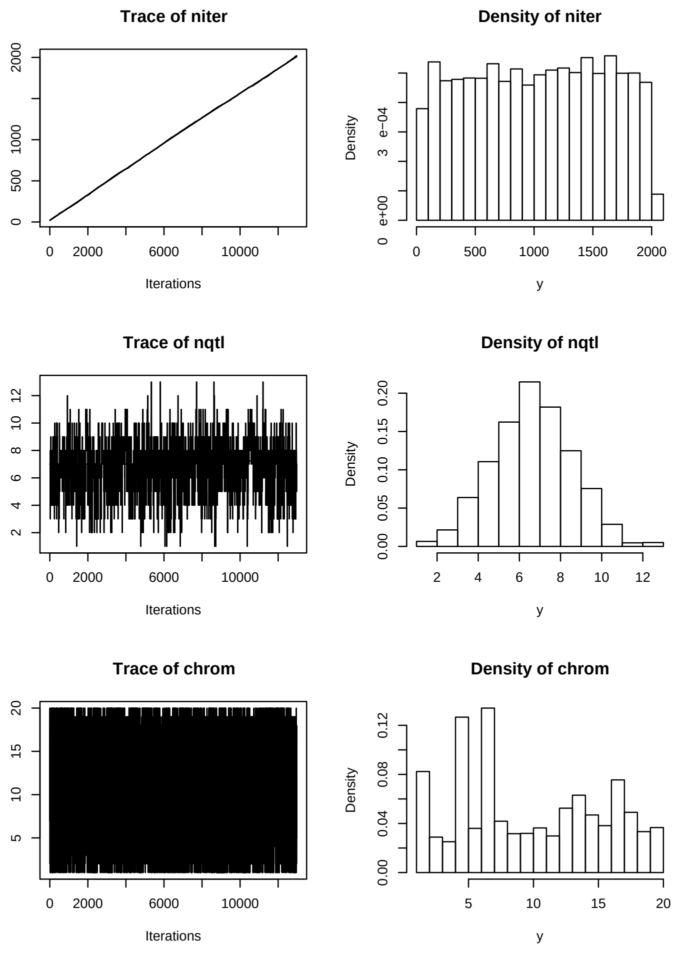
```
> x <- bmq.coda(result, "mainloci", c(1, 2, 3))
```

```
Loading required package: coda
Loading required package: lattice
```

```
> is.mcmc(x)
```

```
[1] TRUE
```

```
> plot(x)
```



Trace of niter
Iterations
Density of niter
Density
y
Trace of nqtl
Iterations
Density of nqtl
Density
y
Trace of chrom
Iterations
Density of chrom
Density
y

R/bmqtl Plots

William Whipple Neely and Brian S. Yandell

2006-04-11

Abstract

The R/bmqtl package provides plotting facilities for results generated by the analytical tools in the R/bmqtl package. These plotting facilities include time series plots of QTL model charactacteristics as basic MCMC diagnostic plots, visual tools for comparison of putative QTL models and exploratory plots whose purpose is the aid in the identification of likely QTL.

1 Getting Started With Plotting

This vignette describes the plotting facililities available through the R/bmqtl package. The purpose of these plots is to provide graphical tools for

1. exploring putative single and multiple QTL,
2. producing interpretable graphics of the relative evidence in favor of a set putative QTL,
3. visual diagnostics of the MCMC model selection algorithm.

Before exploring the plotting functions in R/bmqtl you must have an example of a bim object produced by a run of the MCMC algorithm. The hyper demo produces a bim object called bmqResult2. Consequently, one way to produce a bim object is to run the hyper demo by typing

```
demo(bmq.hyper.tour)
```

at the R prompt. Alternatively a bim object can be created by the following sequence of commands.

```
# First load the library
library(bmqtl)

# Now load the hyper data set.
data(hyper)

# Select just the columns in the hyper data set corresponding to
# chromosomes one through 19.  This means that we have dropped
# the X-chromosome from our analysis.
hyper = subset(hyper, chr=1:19)
```

```
# Calculate genotype  probabilities.
hyper = bmq.genoprob(hyper, type="bc", map.func="Haldane", step=2)

# Now run the MCMC model selection algorithm. Running this command
# may take as much as 15 minutes on slower computers.
bmqResult = bmq.mcmc(hyper, pheno.col = 1,genoupdate=TRUE)
```

The comments in the code above describe the meaning of each step in the sequence of commands used to produce `bmqResult`. Notice that the required package `R/qtl` is automatically loaded when `R/bmqtl` is loaded. The final command above creates a `bim` object called `bmqResult` that will be used throughout this vignette. If you choose to run the `hyper` demo, the rest of the vignette may be easier to follow if you copy the `bmqResults2` object created by the `hyper` demo to a new object called `bmqResult` with the command

```
bmqResult <- bmqResult2
```

Changing the name of the `bim` object is required for the plotting demo to work and allows the sample code in this vignette to be used without modification.

2 Plotting MCMC History

The final command above produced the `bmqResult` object by running the MCMC model selection algorithm:

```
bmqResult = bmq.mcmc(hyper, pheno.col = 1, genoupdate=TRUE)
```

Consequently a logical first step in exploring the results of the analysis is to examine this MCMC chain. The plotting tools in `R/bmqtl` provide a method for visually inspecting the history of the MCMC run. The command

```
plot.bim.mcmc(bmqResult)
```

shows a default view of the MCMC chain as a time series. The results of this command are shown in Figure 1. Each iteration of the MCMC chain represents a single model; therefore, we can explore the history of the MCMC chain by plotting time series for relevant model features. The default time series plotted by `plot.bim.mcmc` show the sampling histories for

1. number of QTL in each model model (`nqtl`),
2. mean phenotype according to each model (`mean`),
3. environmental variability under each model (`envvar`),
4. variance explained under each model (`var`) and
5. heritability under each model (`herit`).

It is possible to plot a subset of the model characteristics above, by using the optional argument `items` in the `plot.bim.mcmc` function. For example, in order to view just the number of QTL (`nqtl`) and the model means, use the following command.

```
plot.bim.mcmc(bmqResult, items = c("nqtl","mean"))
```

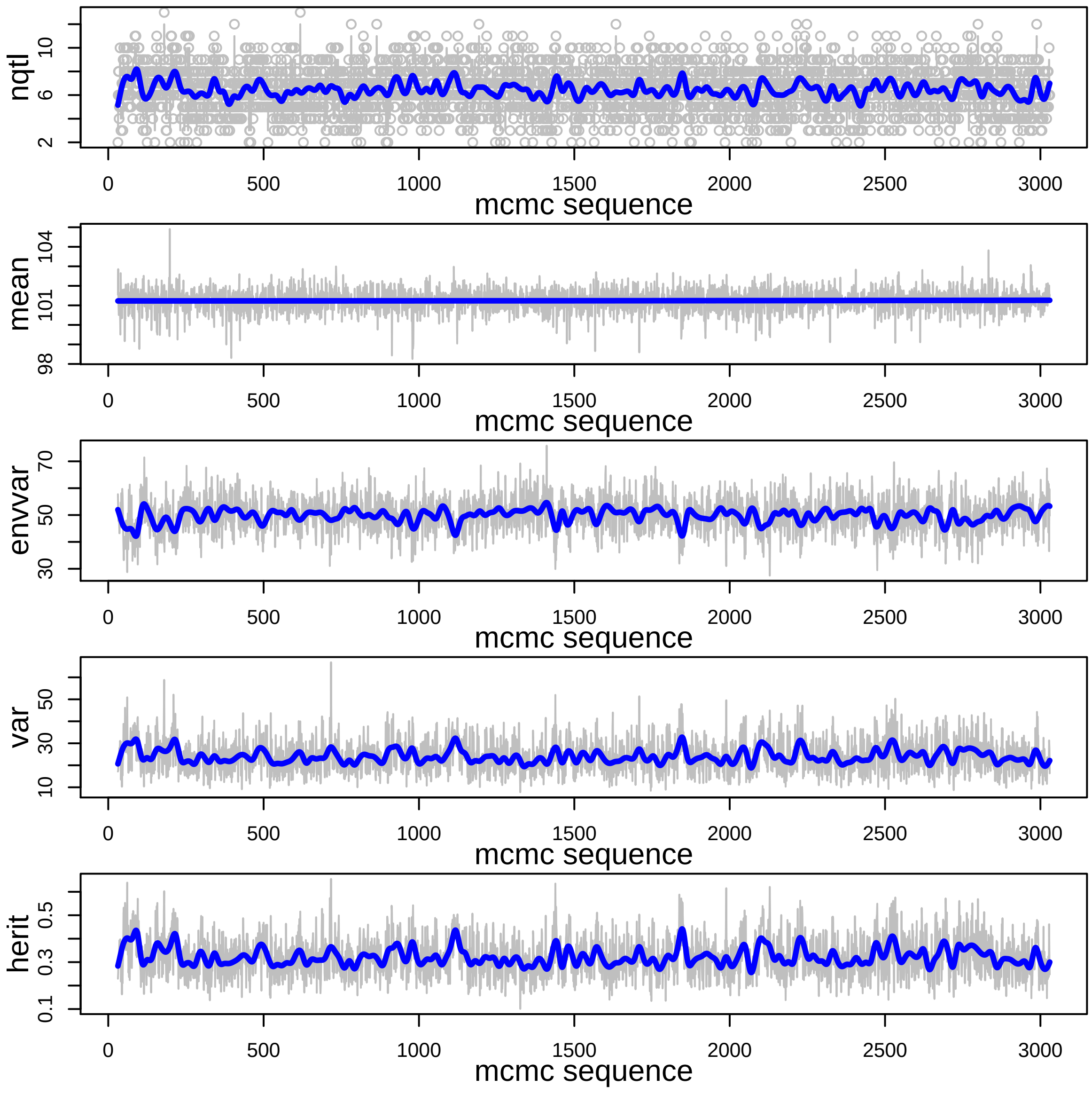


Figure 1: Diagnostic Plot for a MCMC run.

3 A Plot of QTL by chromosome

From a biological perspective it may be interesting to view the location of possible QTL along the chromosome. The function `plot.bim.loci` shows a plot of quantitative trait loci for each chromosome. The QTL are from single QTL models appearing as samples in the MCMC chain. In the plot, the actual locations of possible QTL are jittered slightly in order to give a sense of the density of putative QTL in the vicinity of each marker. The code

```
plot.bim.loci(bmqResult)
```

will produce the plot shown in Figure 2.

In order to view a subset of the chromosomes, the parameter `chr` can be used to limit the plot to a selected set of chromosomes. The horizontal (blue) lines in the plot show the locations of markers. The markers themselves can be labelled by using the parameters `markers` in the function.

```
plot.bim.loci(bmqResult,chr=c(3,4), markers=TRUE)
```

Figure 3 shows the result of this command.

4 Summary Plots for Sampled Models

The function `plot.bim.model` produces a composite (4-by-4) summary plot of the models sampled by the MCMC chain. These plots are useful as an initial tool for examining the evidence in favor of multiple QTL models and in determining the locations of QTL. Figure 4 shows the plot produced by the command `plot.bim.model(bmqResult)`. The function of each of these plots is described below.

1. The plot appearing in the upper-left of the figure represents a plot of the prior distribution for the number of QTL involved in models (shown as a broken blue line) against the corresponding posterior probabilities (shown as a histogram).
2. The plot in the upper-right shows Bayes factor ratios. These are the ratios of posterior probabilities to prior probabilities. For pairs of values along the horizontal axis of this plot, the member of the pair with a larger Bayes factor ratio should be interpretted as more likely. The vertical arrows give an idication of the strength of evidence.
3. The plot in the lower-left shows the same information as the plot in the upper left in terms of the pattern of chromosomes involved in the models.
4. The plot in the lower-right gives the Bayes factor ratios, as does the plot in the upper right, arranged by the pattern of chromosomes involved in the models.

As with other plot functions in the `R/bmqtl` package, it is possible to limit attention of a subset of chromosomes. The argument `pattern` can be used to limit the models plotted to those involving a specified list of chromosomes. For example the command `plot.bim.model(bmqResult,pattern=c(2,3,17))`

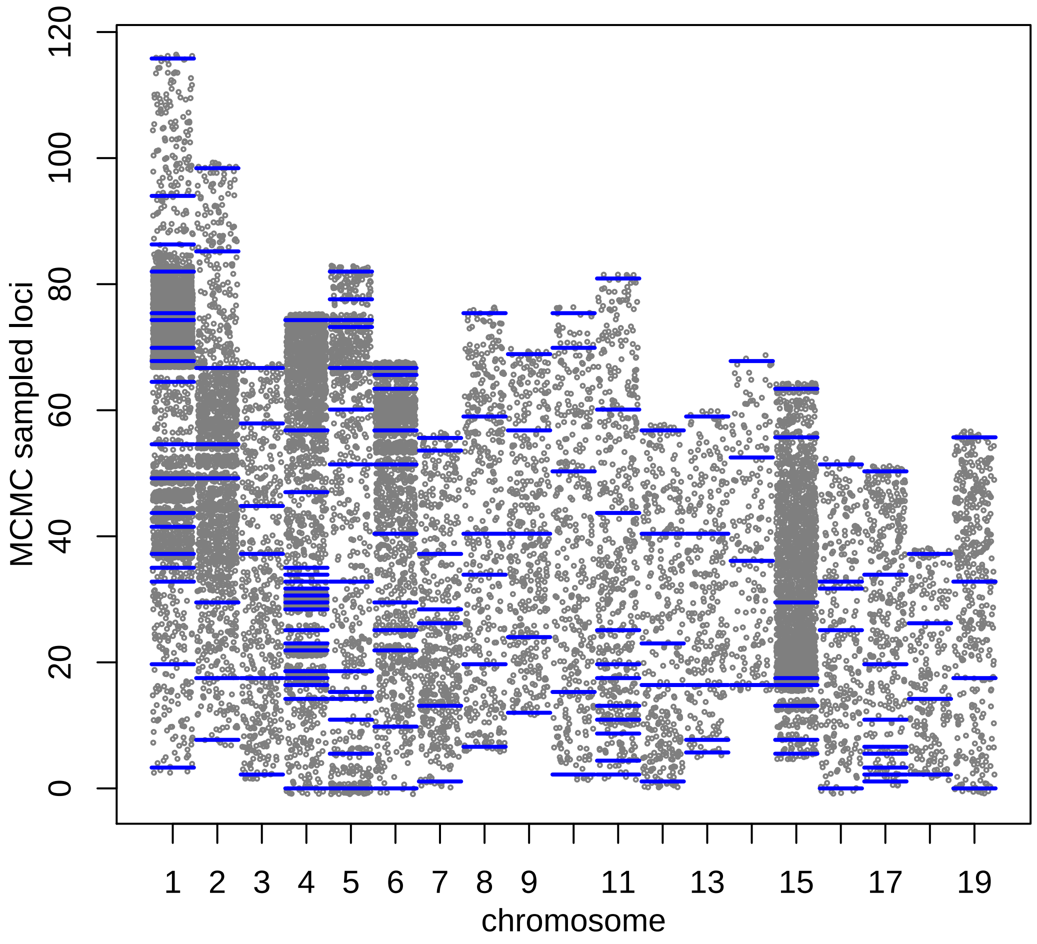


Figure 2: jittered plot of quantitative trait loci by chromosome

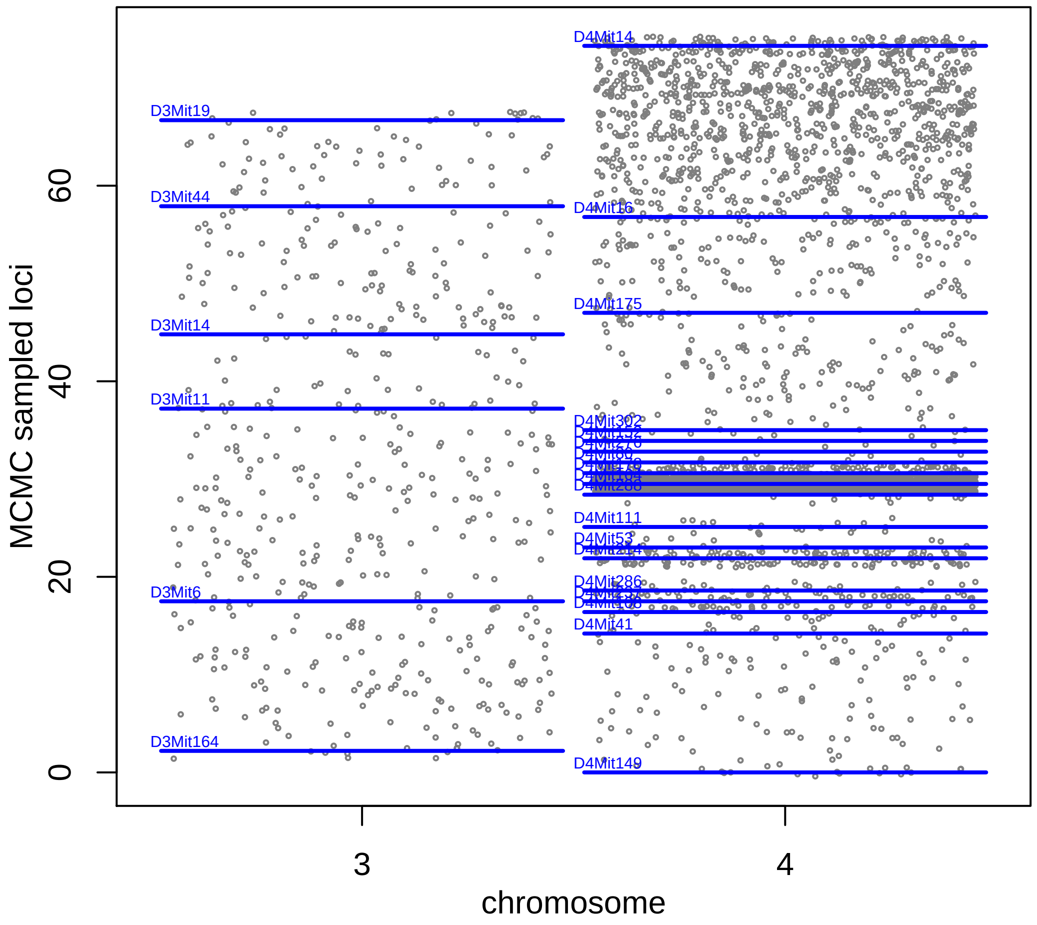


Figure 3: A jittered plot of quantitative trait loci showing only only chromosomes 3 and 4 that includes locations and ids of markers.

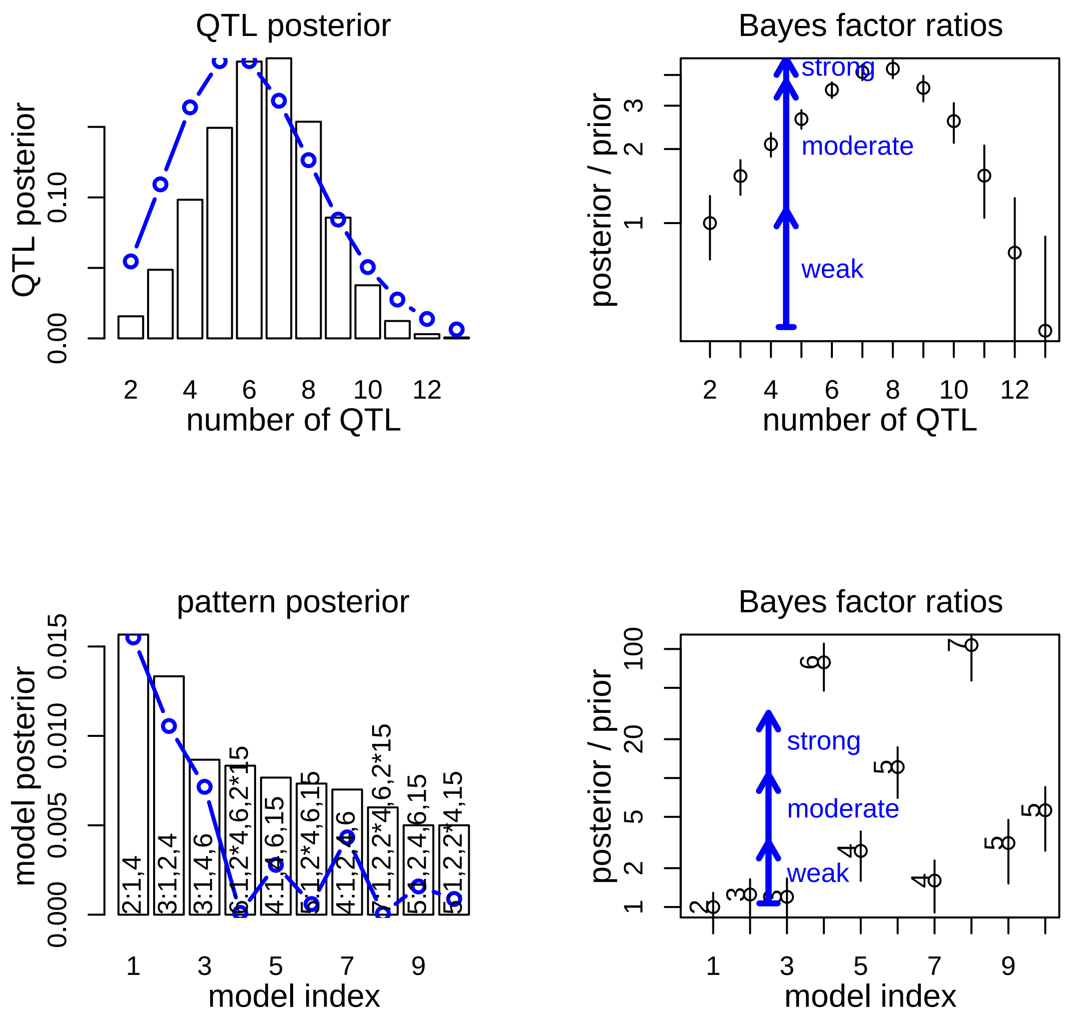


Figure 4: jittered plot of quantitative trait loci by chromosome

plots only those models involving chromosomes 2,3 and 17. Repeats in the pattern sequence indicate multiple QTL on the same chromosome.

5 The Plot Demo

The plot demo (`demo(bmq.plot.tour)` gives a sample of the plots available in the `R/bmqtl` package. Running the plot demo requires, the `lattice` package, this pacakge is part of the standard `R` distribution so it is probably already installed. To start the plot demo, use the command

```
demo(bmq.plot.tour)
```

The plot demo begins by giving a generic plot for the `bim` object `bmqResult`. The `R/bmqtl` generic `bim` plot is analogous to the generic `R` plot for linear model objects. Where the generic plot for a linear model object shows a sequence of graphics whose purpose is to aid in the initial results of model fitting, the generic plot function for `bim` objects shows a sequence of graphics whose purpose is to give an initial assessment of the results produced by the MCMC algortihm. The generic plot for the `bim` object `bmqResult` created above is shown with the command

```
plot(bmqResult)
```

. The generic plot function shows a sequence of plots that include time series plots of the mcmc chain, jittered plots of QTL by chromosome and others. The sequence of plots appearing in the plot demo is listed below.

The list of plots shown by the generic plot function.

1. A time series plot of the mcmc chain runs. This can also be created by the command `plot.mcmc.bim(bmqResult)`.
2. A jittered plot of QTL by chromosome. This plot is identical to the plot produced by the command `plot.bim.loci(bmqResult)`.
3. A summary plot the posterior distribution generated by the mcmc chain.
4. A model selection plot by chromosome. This plot can be produced as follows.

   ```
   model <- bim.model(bmqResult)
   plot(model)
   ```

5. Histograms of QTL loci plus scatter plots and smooth estimates of QTL effects. This plot is the same as the plot generated by `plot.bim.effects(bmqResult)`.
6. A plot of epistatic effects if such effects are allowed. This plot is generated by `plot.bim.epistasis(bmqResult)`.
7. Summary diagnostics as histograms and boxplots by number of QTL. This final diagnostic plot can be generated separately by the command `plot.bim.diag(bmqResult)`.