An example Knitr/R Markdown document

Karl W Broman

This is a portion of the "A shorter tour of R/qtl" tutorial, developed here in multiple formats to illustrate the use of knitr. This particular document is written with LaTeX.

Preliminaries

To install R/qtl, you need to first install the package. Type (within R) `install.packages("qtl")` (This needs to be done just once.)

You then load the R/qtl package using the `library` function:

```
library(qtl)
```

This needs to be done every time you start R. (There is a way to have the package loaded automatically every time, but we won't discuss that here.)

To get help on the functions and data sets in R (and in R/qtl), use `help()` or `?`. For example, to view the help file for the `read.cross` function, type one of the following:

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

Data import

We will consider data from Sugiyama et al., Physiol Genomics 10:5–12, 2002. Load the data into R/qtl as follows.

```
sug <- read.cross("csv", "http://www.rqtl.org", "sug.csv",
                  genotypes=c("CC", "CB", "BB"),
                  alleles=c("C", "B"))

##  --Read the following data:
##   163  individuals
##   93  markers
##   6  phenotypes
##  --Cross type: f2
```

The function `read.cross` is for importing data into R/qtl. `"sug.csv"` is the name of the file, which we import directly from the R/qtl website. `genotypes` indicates the codes used for the genotypes; `alleles` indicates single-character codes to be used in plots and such.

`read.cross` loads the data from the file and formats it into a special cross object, which is then assigned to `sug` via the assignment operator `<-`.

The data are from an intercross between BALB/cJ and CBA/CaJ; only male offspring were considered. There are four phenotypes: blood pressure, heart rate, body weight, and heart weight. We will focus on the blood pressure phenotype, will consider just the 163 individuals with genotype data and, for simplicity, will focus on the autosomes.

Summaries

The data object `sug` is complex; it contains the genotype data, phenotype data and genetic map. R has a certain amount of "object oriented" facilities, so that calls to functions like `summary` and `plot` are interpreted appropriately for the object considered.

The object `sug` has "class" `"cross"`, and so calls to `summary` and `plot` are actually sent to the functions `summary.cross` and `plot.cross`.

Use `summary()` to get a quick summary of the data. (This also performs a variety of checks of the integrity of the data.)

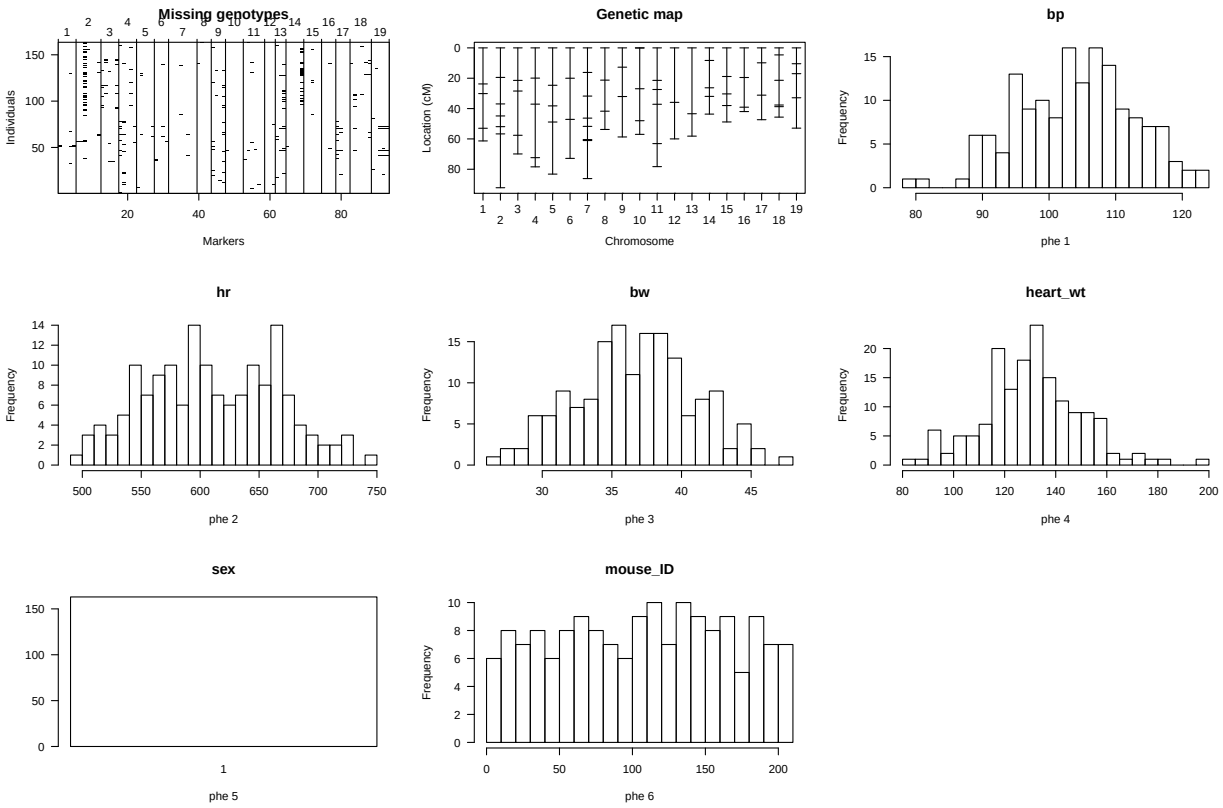
```
summary(sug)
```

```
##     F2 intercross
##
##     No. individuals:    163
##
##     No. phenotypes:     6
##     Percent phenotyped: 95.1 95.7 99.4 99.4 100 100
##
##     No. chromosomes:    19
##         Autosomes:      1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
##                         16 17 18 19
##
##     Total markers:      93
##     No. markers:        5 7 5 5 5 4 8 4 4 5 6 3 3 5 5 4 4 6
##                         5
##     Percent genotyped:  98.3
##     Genotypes (%):      CC:23.9  CB:50.2  BB:26  not BB:0
##                         not CC:0
```

We see that this is an intercross with 163 individuals. There are 6 phenotypes, and genotype data at 93 markers across the 19 autosomes. The genotype data is quite complete.

Use `plot()` to get a summary plot of the data.

```
plot(sug)
```



The plot in the upper-left shows the pattern of missing genotype data, with black pixels corresponding to missing genotypes. The next plot shows the genetic map of the typed markers. The following plots are histograms or bar plots for the six phenotypes. The last two "phenotypes" are sex (with 1 corresponding to males) and mouse ID.

Single-QTL analysis

Let's now proceed to QTL mapping via a single-QTL model.

We first calculate the QTL genotype probabilities, given the observed marker data, via the function `calc.genoprob`. This is done at the markers and at a grid along the chromosomes. The argument `step` is the density of the grid (in cM), and defines the density of later QTL analyses.

```
sug <- calc.genoprob(sug, step=1)
```

The output of `calc.genoprob` is the same cross object as input, with additional information (the QTL genotype probabilities) inserted. We assign this back to the original object (writing over the previous data), though it could have also been assigned to a new object.

To perform a single-QTL genome scan, we use the function `scanone`. By default, it performs standard interval mapping (that is, maximum likelihood via the EM algorithm). Also, by default, it considers the first phenotype in the input cross object (in this case, blood pressure).

```
out.em <- scanone(sug)
```

The output has "class" "scanone". The summary function is passed to the function summary.scanone, and gives the maximum LOD score on each chromosome.

```
summary(out.em)
```

```
##           chr   pos   lod
## D1MIT36     1 76.73 1.449
## c2.loc77    2 82.80 1.901
## c3.loc42    3 52.82 1.393
## c4.loc43    4 47.23 0.795
## D5MIT223    5 86.57 1.312
## c6.loc26    6 27.81 0.638
## c7.loc45    7 47.71 6.109
## c8.loc34    8 54.90 1.598
## D9MIT71     9 27.07 0.769
## c10.loc51  10 60.75 0.959
## c11.loc34  11 38.70 2.157
## D12MIT145  12  2.23 1.472
## c13.loc20  13 27.26 1.119
## D14MIT138  14 12.52 1.119
## c15.loc8   15 11.96 5.257
## c16.loc31  16 45.69 0.647
## D17MIT16   17 17.98 1.241
## D18MIT22   18 13.41 1.739
## D19MIT71   19 56.28 0.402
```

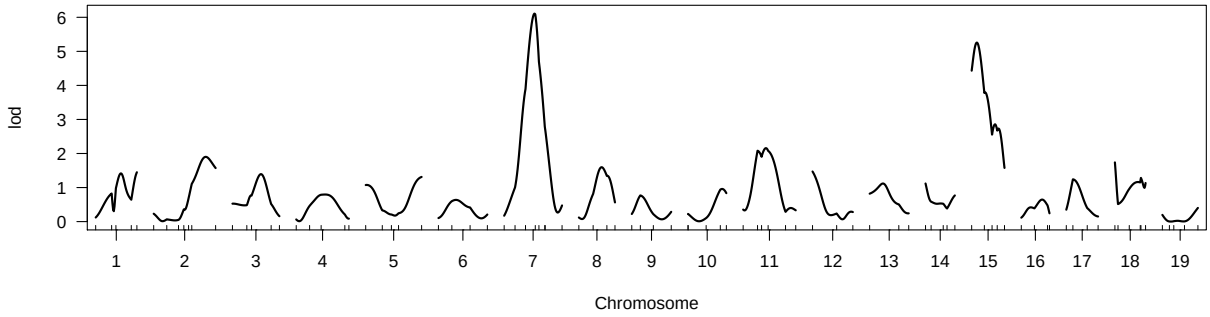
Alternatively, we can give a threshold, e.g., to only see those chromosomes with LOD > 3.

```
summary(out.em, threshold=3)
```

```
##          chr  pos  lod
## c7.loc45   7 47.7 6.11
## c15.loc8  15 12.0 5.26
```

We can plot the results as follows.

```
plot(out.em)
```

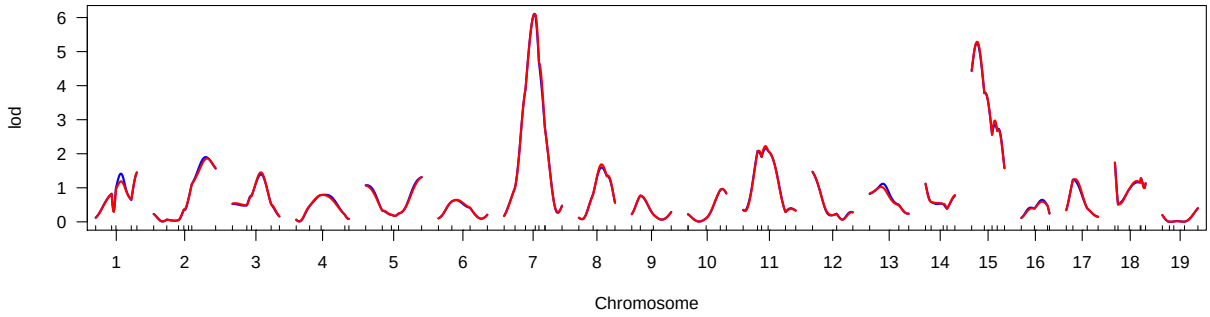


We can do the genome scan via Haley-Knott regression by calling `scanone` with the argument `method="hk"`.

```
out.hk <- scanone(sug, method="hk")
```

We may plot the two sets of LOD curves together in a single call to `plot`.

```
plot(out.em, out.hk, col=c("blue", "red"))
```

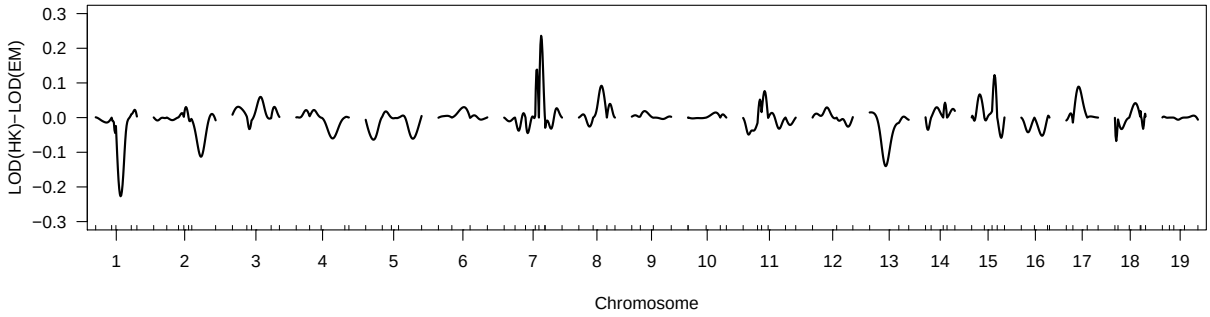


Alternatively, we could do the following (figure not included, for brevity):

```
plot(out.em, col="blue")
plot(out.hk, col="red", add=TRUE)
```

It's perhaps more informative to plot the differences:

```
plot(out.hk - out.em, ylim=c(-0.3, 0.3),
     ylab="LOD(HK)-LOD(EM)")
```



Permutation tests

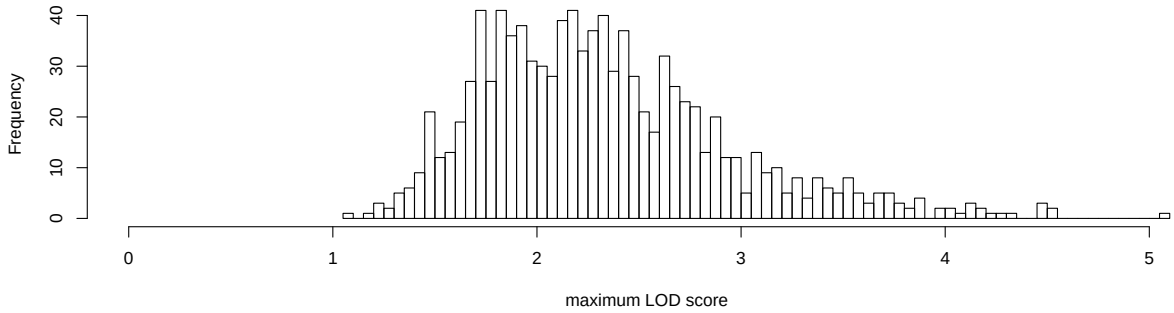
To perform a permutation test, to get a genome-wide significance threshold or genome-scan-adjusted p-values, we use `scanone` just as before, but with an additional argument, `n.perm`, indicating the number of permutation replicates. It's quickest to use Haley-Knott regression.

```
operm <- scanone(sug, method="hk", n.perm=1000)

## Doing permutation in batch mode ...
```

A histogram of the results (the 1000 genome-wide maximum LOD scores) is obtained as follows:

```
plot(operm)
```



Significance thresholds may be obtained via the `summary` function:

```
summary(operm)

## LOD thresholds (1000 permutations)
```

```
##      lod
## 5%  3.52
## 10% 3.15
```

```
summary(operm, alpha=c(0.05, 0.2))
```

```
## LOD thresholds (1000 permutations)
##      lod
## 5%  3.52
## 20% 2.76
```

The permutation results may be used along with the `scanone` results to have significance thresholds and p-values calculated automatically:

```
summary(out.hk, perms=operm, alpha=0.2, pvalues=TRUE)
```

```
##          chr  pos  lod pval
## c7.loc45   7 47.7 6.11    0
## c15.loc8  15 12.0 5.29    0
```

Interval estimates of QTL location

For the blood pressure phenotype, we've seen good evidence for QTL on chromosomes 7 and 15. Interval estimates of the location of QTL are commonly obtained via 1.5-LOD support intervals, which may be calculated via the function `lodint`. Alternatively, an approximate Bayes credible interval may be obtained with `bayesint`.

To obtain the 1.5-LOD support interval and 95% Bayes interval for the QTL on chromosome 7, type the following. The first and last rows define the ends of the intervals; the middle row is the estimated QTL location.

```
lodint(out.hk, chr=7)
```

```
##          chr   pos   lod
## c7.loc34   7 36.71 4.404
## c7.loc45   7 47.71 6.107
## c7.loc54   7 56.71 4.505
```

```
bayesint(out.hk, chr=7)
```

```
##          chr   pos   lod
## c7.loc37   7 39.71 5.086
## c7.loc45   7 47.71 6.107
## c7.loc50   7 52.71 5.379
```

It is sometimes useful to identify the closest flanking markers; use `expandtomarkers=TRUE`:

```
lodint(out.hk, chr=7, expandtomarkers=TRUE)
```

```
##          chr   pos   lod
## D7MIT176   7 34.48 3.894
## c7.loc45   7 47.71 6.107
## D7MIT7     7 63.14 2.800
```

```
bayesint(out.hk, chr=7, expandtomarkers=TRUE)
```

```
##          chr   pos   lod
## D7MIT176   7 34.48 3.894
## c7.loc45   7 47.71 6.107
## D7MIT323   7 54.45 4.691
```

We can calculate the 2-LOD support interval and the 99% Bayes interval as follows.

```
lodint(out.hk, chr=7, drop=2)
```

```
##          chr   pos   lod
## c7.loc32   7 34.71 3.946
## c7.loc45   7 47.71 6.107
## c7.loc57   7 59.71 3.850
```

```
bayesint(out.hk, chr=7, prob=0.99)
```

```
##          chr   pos   lod
## c7.loc34   7 36.71 4.404
## c7.loc45   7 47.71 6.107
## c7.loc54   7 56.71 4.505
```

The intervals for the chr 15 locus may be calculated as follows.

```
lodint(out.hk, chr=15)
```

```
##           chr   pos   lod
## D15MIT175  15  3.96 4.433
## c15.loc8   15 11.96 5.290
## D15MIT184  15 22.82 3.778
```

```
bayesint(out.hk, chr=15)
```

```
##           chr   pos   lod
## D15MIT175  15  3.96 4.433
## c15.loc8   15 11.96 5.290
## c15.loc16  15 19.96 4.374
```