The R/qtlnet package

Elias Chaibub Neto and Brian S Yandell

SISG 2012

July 8, 2012

Simulate data

We simulate data from a F_2 cross with 500 ind, and 5 chr of len 100 cM, containing 11 equally spaced markers per chr. We simulated one QTL per pheno. The QTLs, Q_t, $t = 1, 2, 3, 4, 5$, were placed at the middle marker on chr t. We set additive and dominance QTL effects to 1 and 0, respectively.

```
> library(qtlnet)
> set.seed(12345)
> Map <- sim.map(len = rep(100, 5), n.mar = 11, eq.spacing = TRUE,
+                include.x = FALSE)
> Cross <- sim.cross(map = Map, n.ind = 500, type = "f2")
> crosses <- vector(mode = "list", length = 5)
> add.effects <- c(1, 1, 1, 1, 1)
> for (i in 1:5) {
+   map <- sim.map(len = rep(100, i), n.mar = 11, eq.spacing = TRUE,
+                  include.x = FALSE)
+   crosses[[i]] <- sim.cross(map = map, n.ind = 500, type = "f2",
+                             model = c(i, 50, add.effects[i], 0))
+   Cross$geno[[i]] <- crosses[[i]]$geno[[i]]
+ }
```

Simulate data

The pheno data was simulated according to the network below, using regr equations with regr coeffs set to 1.

```
> beta <- 1
> Cross$pheno[, 1] <- crosses[[1]]$pheno
> Cross$pheno[, 2] <- crosses[[2]]$pheno + beta * Cross$pheno[, 1]
> Cross$pheno[, 3] <- crosses[[3]]$pheno + beta * Cross$pheno[, 2]
> Cross$pheno[, 4] <- crosses[[4]]$pheno + beta * Cross$pheno[, 2]
> Cross$pheno[, 5] <- crosses[[5]]$pheno + beta * Cross$pheno[, 3] +
+                     beta * Cross$pheno[,4]
> names(Cross$pheno) <- paste("y", 1:5, sep = "")
```

$$\begin{tikzcd}
& & & Q_3 \arrow[d] & & \\
& & & Y_3 \arrow[dr] & & \\
Q_1 \arrow[r] & Y_1 \arrow[r] & Y_2 \arrow[ur] \arrow[dr] & & Y_5 & Q_5 \arrow[l] \\
& & Q_2 \arrow[u] & Y_4 \arrow[ur] & & \\
& & & Q_4 \arrow[u] & &
\end{tikzcd}$$

Permutation test threshold

We determine the QTL mapping LOD threshold via permutation test.

```
> Cross <- calc.genoprob(Cross, step = 1)
> set.seed(12345)
> perm.test <- scanone(Cross, n.perm = 1000, method = "hk")
Doing permutation in batch mode ...
> summary(perm.test)
LOD thresholds (1000 permutations)
     lod
5%  3.04
10% 2.70
```

We adopt a LOD threshold of 3.04, that aims to control GWER $< 5\%$.

QDG routines

We perform QTL mapping with Haley-Knott regression for all 5 phenotypes.

```
> Scan <- scanone(Cross, pheno.col = 1:5, method = "hk")
```

Next we determine the QTLs for each phenotype, and create a list with objects of class **qtl** that is needed as impute for the **qdg** function.

```
> Cross <- sim.geno(Cross, n.draws = 1)
> marker.nms <- allqtls <- vector(mode = "list", length = 5)
> names(marker.nms) <- names(allqtls) <- paste("y", 1:5, sep = "")
> for (i in 1:5) {
+   aux <- summary(Scan[, c(1, 2, i + 2)], thr = 3.04)
+   marker.nms[[i]] <- find.marker(Cross, chr = aux[, 1], pos = aux[, 2])
+   allqtls[[i]] <- makeqtl(Cross, chr = aux[, 1], pos = aux[, 2])
+ }
```

QDG routines

Fit the QDG algorithm.

```
> out1 <- qdg(cross = Cross,
+             phenotype.names = paste("y", 1:5, sep = ""),
+             marker.names = marker.nms,
+             QTL = allqtls,
+             alpha = 0.005,
+             n.qdg.random.starts = 10,
+             addcov = NULL,
+             intcov = NULL,
+             skel.method = "pcskel")
>
> out1$UDG
  node1 node2 edge
1    y1    y2    1
3    y2    y3    1
4    y2    y4    1
6    y3    y5    1
8    y4    y5    1
```

QDG routines

```
> out1$DG
  node1 direction node2 lod score
1    y1      ---->    y2 24.135325
2    y2      ---->    y3 23.990280
3    y2      ---->    y4 32.013798
4    y3      ---->    y5  3.119176
5    y4      ---->    y5  9.726617
>
>out1$Solutions
$solutions
$solutions[[1]]
  node1 direction node2      lod
1    y1      ---->    y2 24.13533
2    y2      ---->    y3 61.02425
3    y2      ---->    y4 69.14849
4    y3      ---->    y5 54.17467
5    y4      ---->    y5 69.25563

$loglikelihood
[1] -3595.164
```

QDG routines

Plot the QDGs

```
> gr1 <- graph.qdg(out1, include.qtl = FALSE)
> plot(gr1)
> gr2 <- graph.qdg(out1, include.qtl = TRUE)
> plot(gr2)
```

(cannot export eps from R. pdf has no margins)

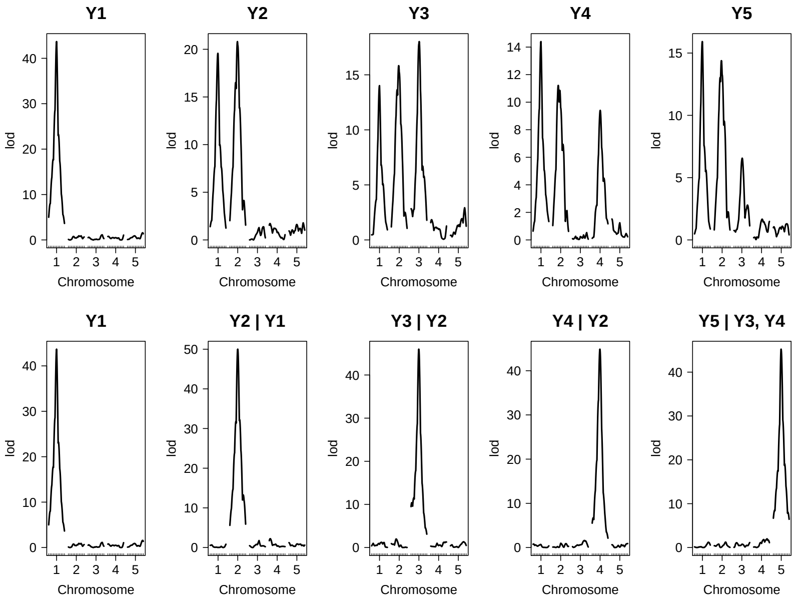
Although the structure of the phenotype network is correct, the genetic architecture is not.

Unconditional versus conditional QTL mapping

Here we plot the LOD profiles for all phenotypes using both unconditional mapping analysis, and conditional mapping (where the parents of each phenotype are used as additive covariates in the QTL mapping).

```
> par(mfrow = c(2, 5), cex.lab = 1.5, cex.axis = 1.5, cex.main = 2)
> uncond.nms <- paste("Y", 1:5, sep = "")
> for (i in 1:5) {
+   plot(Scan, lodcolumn = i, main = uncond.nms[i], ylab = "lod")
+ }
> plot(Scan, lodcolumn = 1, main = uncond.nms[1], ylab = "lod")
> cond.nms <- c("Y1", "Y2 | Y1", "Y3 | Y2", "Y4 | Y2", "Y5 | Y3, Y4")
> pheno.parents <- list(NULL, 1, 2, 2, c(3, 4))
> for (i in 2:5) {
+   CondScan <- scanone(Cross, pheno.col = i, method = "hk",
+                       addcov = Cross$pheno[, pheno.parents[[i]]])
+   plot(CondScan, main = cond.nms[i], ylab = "lod")
+ }
```

Unconditional versus conditional QTL mapping



QTLnet routines - basic functionality

Fit the QTLnet algorithm.

```
> out2 <- mcmc.qtlnet(cross = Cross,
+                     pheno.col = 1:5,
+                     threshold = 3.04,
+                     addcov = NULL,
+                     intcov = NULL,
+                     nSamples = 1000,
+                     thinning = 3,
+                     max.parents = 4,
+                     M0 = NULL,
+                     burnin = 0.2,
+                     method = "hk",
+                     random.seed = 987654321,
+                     init.edges = 0,
+                     saved.scores = NULL,
+                     rev.method = "nbhd",
+                     verbose = TRUE)
```

QTLnet routines - basic functionality

```
> summary(out2)

Model-averaged network: (min.prob = 0.5)
  cause effect prob
1    y1     y2    1
2    y2     y3    1
3    y2     y4    1
4    y3     y5    1
5    y4     y5    1

Posterior probabilities by direction:
   node1 node2   -->   <--    no
1     y1    y2 1.000 0.000 0.000
2     y1    y3 0.019 0.000 0.981
3     y1    y4 0.073 0.000 0.927
4     y1    y5 0.080 0.000 0.920
5     y2    y3 1.000 0.000 0.000
...

Acceptance frequency for MCMC: 0.9996667
```

QTLnet routines - basic functionality

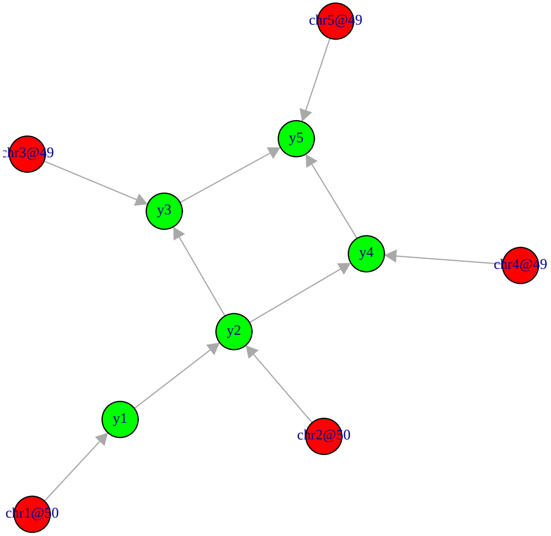
```
> print(out2)

Model averaged probabilities for edge direction (row -> col):
     [,1] [,2]  [,3]  [,4]  [,5]
[1,]    0    1 0.019 0.073 0.080
[2,]    0    0 1.000 1.000 0.094
[3,]    0    0 0.000 0.054 1.000
[4,]    0    0 0.029 0.000 1.000
[5,]    0    0 0.000 0.000 0.000

Posterior probabilities by causal model:
                                  post.prob      BIC
(1)(2|1)(3|2)(4|2)(5|3,4)       0.714107366 7149.930
(1)(2|1)(3|2)(4|2)(5|2,3,4)     0.081148564 7155.238
(1)(2|1)(3|2)(4|2,3)(5|3,4)     0.049937578 7156.117
(1)(2|1)(3|2)(4|2)(5|1,3,4)     0.037453184 7156.134
(1)(2|1)(3|2)(4|1,2)(5|3,4)     0.028714107 7154.531
(1)(2|1)(3|2,4)(4|2)(5|3,4)     0.027465668 7156.141
(1)(2|1)(3|2)(4|1,2)(5|1,3,4)   0.026217228 7160.734
...
```

QTLnet routines - basic functionality

```
> loci.qtlnet(out2)
$y1
[1] "chr1@50"

$y2
[1] "chr2@50"

$y3
[1] "chr3@49"

$y4
[1] "chr4@49"

$y5
[1] "chr5@49"
```

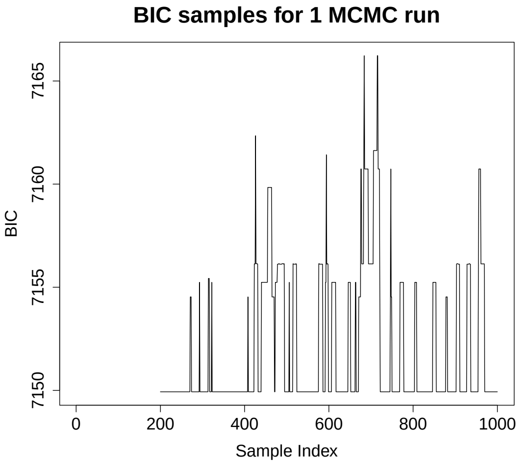
QTLnet routines - basic functionality

```
> plot(out2)
```



QTLnet routines - basic functionality

```
> par(mfrow = c(1, 1))
> plotbic.qtlnet(out2, smooth = FALSE)
```



QTLnet routines - parallel implementation

The most expensive part of calculations is running **scanone** on each phenotype with parent phenotypes as covariates. Our strategy is to pre-compute the BIC contributions using a cluster and save them for later use.

We divide the job into four steps:

1. Determine parents and divide into reasonable sized groups.
2. Compute BIC scores using scanone on a grid of computers.
3. Compute multiple MCMC runs on a grid of computers.
4. Catenate the outputs of the multiple MCMC runs into a single output object.

We illustrate this approach with a simple example of "parallel" analysis.

QTLnet routines - parallel implementation - step 1

STEP 1: defines how the computations are going to break up (that are carried out on steps 2 and 3).

```
> pheno.col <- 1:5
> max.parents <- 4
> size.qtlnet(pheno.col, max.parents)
[1] 80
> parents <- parents.qtlnet(pheno.col, max.parents)
> groups <- group.qtlnet(parents = parents, group.size = 10)
>
> save(Cross, pheno.col, max.parents, parents, groups,
+      file = "Step1.RData", compress = TRUE)
```

The function **size.qtlnet** determines the number of **scanone** calculations possible for a network with nodes **pheno.col** and maximum parent size **max.parents**.

```
> size.qtlnet(pheno.col, max.parents)
[1] 80
```

QTLnet routines - parallel implementation - step 1

The **parents.qtlnet** function creates a list of all possible parent sets (up to **max.parents** in size) to be used as covariates of the child phenotypes in the **scanone** computations.

The `parents` column shows the possible parent sets. The `n.child` column represents the number of possible child nodes to the parent set.

```
> parents <- parents.qtlnet(pheno.col, max.parents)
> parents
        parents n.child
                      5
1             1       4
2             2       4
...
1,2         1,2       3
...
```

No parents (5 `scanones`): $y_1 \sim 1$, $y_2 \sim 1$, $y_3 \sim 1$, $y_4 \sim 1$, and $y_5 \sim 1$.

With y_1 as a parent (4 `scanones`): $y_2 \sim y_1$, $y_3 \sim y_1$, $y_3 \sim y_1$, and $y_4 \sim y_1$.

With y_1 and y_2 as parents (3 `scanones`): $y_3 \sim y_1 + y_2$, $y_4 \sim y_1 + y_2$, and $y_4 \sim y_1 + y_2$.

QTLnet routines - parallel implementation - step 1

The function **group.qtlnet** groups the parent sets into roughly equal size groups for parallel computations.

```
> groups <- group.qtlnet(parents = parents, group.size = 10)
> groups
  begin end
1     1   2
2     3   4
3     5   7
4     8  10
5    11  14
6    15  18
7    19  23
8    24  30
9    31  31
> pa <- summary(parents)
> N <- rep(NA, nrow(groups))
> for (i in 1:nrow(groups))
+   N[i] <- sum(pa[seq(groups[i, 1], groups[i, 2]), 2])
> N
[1]  9  8 11  9 12 10 10 10  1
```

QTLnet routines - parallel implementation - step 2

STEP 2: Pre-compute BIC scores for selected parents.

```
> load("Step1.RData")
> for (i in seq(nrow(groups))) {
+   bic <- bic.qtlnet(Cross,
+                     pheno.col,
+                     threshold = 3.04,
+                     max.parents = max.parents,
+                     parents = parents[seq(groups[i,1], groups[i,2])])
+
+   save(bic, file = paste("bic", i, ".RData", sep = ""), compress = TRUE)
+ }
```

QTLnet routines - parallel implementation - step 2

Read in saved BIC scores and combine into one object.

```
> load("Step1.RData")
> bic.group <- list()
> for (i in seq(nrow(groups))) {
+   load(paste("bic", i, ".RData", sep = ""))
+   bic.group[[i]] <- bic
+   cat("group =", i, "\n")
+ }
> saved.scores <- bic.join(Cross, pheno.col, bic.group, max.parents = 4)
```

QTLnet routines - parallel implementation - step 2

```
> saved.scores
              y1        y2       y3       y4       y5
        1480.704 1785.9987 1982.565 2014.538 2677.213
1       1132.647 1414.8944 1776.324 1780.912 2437.802
2       1304.698 1222.2440 1394.943 1434.985 2005.474
3       1291.897 1242.0799 1682.636 1687.116 1953.474
4       1299.858 1156.7878 1273.851 1246.465 1917.227
1,2     1137.728 1059.9072 1400.437 1439.585 2011.442
1,3     1138.785 1089.4257 1627.265 1631.393 1917.990
1,4     1138.526 1023.7947 1276.038 1241.347 1885.897
2,3     1263.316 1002.2426 1401.154 1441.172 1800.790
2,4     1287.025 1110.6142 1221.812 1218.454 1759.518
3,4     1279.065 1128.8087 1210.769 1186.155 1424.405
1,2,3   1143.837  896.6137 1406.650 1445.789 1805.105
1,2,4   1143.942  984.3935 1225.789 1222.706 1765.651
1,3,4   1144.734 1000.8086 1202.754 1171.667 1430.608
2,3,4   1269.522 1008.2210 1091.324 1087.177 1429.712
1,2,3,4 1149.933  902.5707 1096.584 1093.126 1435.644
```

QTLnet routines - parallel implementation - step 3

STEP 3: Sample Markov chain (MCMC).

```
> set.seed(54321)
> n.runs <- 3
> for (i in seq(n.runs)) {
+   cat("run =", i, "\n")
+   ## Run MCMC with randomized initial network.
+   mcmc <- mcmc.qtlnet(Cross,
+                       pheno.col,
+                       threshold = 3.04,
+                       thinning = 1,
+                       max.parents = max.parents,
+                       saved.scores = saved.scores,
+                       init.edges = NULL)
+
+   save(mcmc, file = paste("mcmc", i, ".RData", sep = ""),
+        compress = TRUE)
+ }
```

QTLnet routines - parallel implementation - step 4

STEP 4: Combine results for post-processing.

```
> n.runs <- 3
> outs.qtlnet <- list()
> for (i in seq(n.runs)) {
+   load(paste("mcmc", i, ".RData", sep = ""))
+   outs.qtlnet[[i]] <- mcmc
+ }
> out3 <- c.qtlnet(outs.qtlnet)
```

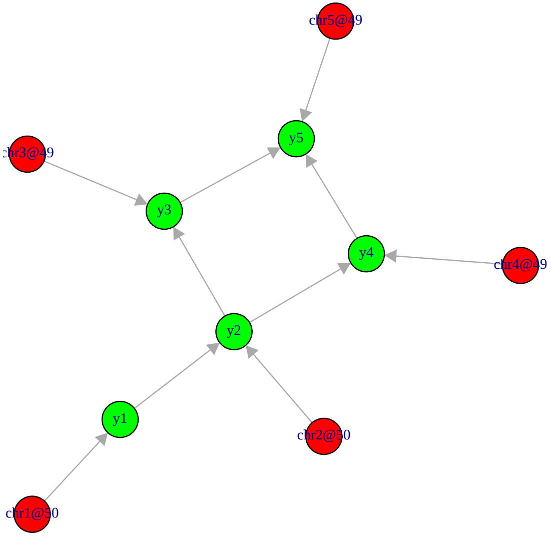
The function **c.qtlnet** catenates the outputs of the 3 separate runs together.

QTLnet routines - parallel implementation - outputs

```
> summary(out3)

Model-averaged network: (min.prob = 0.5)
  cause effect      prob
1    y1     y2 0.9155556
2    y2     y3 0.9255556
3    y2     y4 0.9129630
4    y3     y5 0.9085185
5    y4     y5 0.9103704

Posterior probabilities by direction:
   node1 node2   -->   <--    no
1     y1    y2 0.916 0.084 0.000
2     y1    y3 0.019 0.015 0.966
3     y1    y4 0.033 0.020 0.947
4     y1    y5 0.028 0.006 0.966
5     y2    y3 0.926 0.074 0.000
...

Acceptance frequency for MCMC: 0.999
```

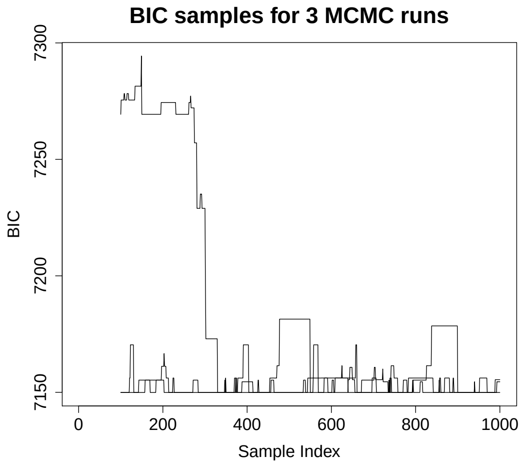
QTLnet routines - parallel implementation - outputs

```
> plot(out3)
```



QTLnet routines - parallel implementation - outputs

```
> plotbic.qtlnet(out3, smooth = FALSE)
```


BIC samples for 3 MCMC runs
BIC
Sample Index