The qtlnet package: a tutorial

Elias Chaibub Neto[*] and Brian S Yandell[†]

July 8, 2012

1 Overview

This tutorial illustrates the QDG (Chaibub Neto et al. 2008) and QTLNet (Chaibub Neto et al. 2010) routines of the qtlnet R package. We illustrate the basic functionality with a few simulated toy examples, and with real data analyzed in the above publications. The qtlnet package builds over the R/qtl package (Broman et al. 2003), and we assume the reader is familiar with it.

2 Basic functionality

We illustrate the package's basic functionality with simulated data.

2.1 Simulation of experimental cross data with R/qtl

Load the R/qtlnet package. The R/qtl package is loaded automatically.

```
library(qtlnet)
```

We start by simulating data from a F_2 cross object composed of 5 phenotypes, 500 individuals, and 5 chromosomes of length 100 cM, containing 11 equally spaced markers per chromosome. We simulated one QTL per phenotype. The QTLs, Q_t, $t = 1, 2, 3, 4, 5$, were unlinked and placed at the middle marker on chromosome t. We set additive and dominance QTL effects to 1 and 0, respectively. The phenotype data was simulated according to the network structure in Figure 1, using regression equations with regression coefficients set to 1.

```
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)
```

[*]Department of Computational Biology, Sage Bionetworks, Seattle WA
[†]Department of Statistics, University of Wisconsin-Madison, Madison WI

```
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]]
}
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 = "")
```

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

Figure 1: True model from which the simulated data was generated.

We compute conditional genotype probabilities, and perform a permutation test (Churchill and Doerge 1994) to determine the LOD threshold that control GWER $< 5\%$ (3.04).

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

2.2 QDG routines

We perform QTL mapping with Haley-Knott regression (Haley and Knott 1992) for all 5 phenotypes.

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

Before we can fit the QDG algorithm, we need first to determine the QTLs that will be used to help out determine the directions of edges in the skeleton of the causal model.

We use the function `sim.geno` to simulate QTL genotypes given the marker data. [This function produces multiple imputations of the genotypes. However, because we do not have missing data and we use the genotypes at the markers positions as the QTL anchors, a single imputation (`n.draws = 1`) is enougth]. The imputed genotypes are needed by the `makeqtl` function, that generates an object of class `qtl`. The `qtl` objects are required by the function `fitqtl` that is used internally in the computation of the model LOD scores.

```
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])
}
```

The function `qdg` implements the QDG algorithm described in Chaibub Neto et al. (2008). It creates and scores QTL dependency graphs (QDGs). The algorithm goes as follows:

1. Construct an undirected dependency graph (UDG). [Setting `skel.method = "pcskel"` uses the PC algorithm (Spirtes et al. 2000) to construct the UDG. `skel.method = "udg"` calls the UDG algorithm (Shipley 2002).]

2. Use QTL genotypes to direct each edge in the UDG. Call it DG.

3. Randomly choose an ordering of all edges in the DG. Call this list ODG.

4. For an edge in the ODG recompute the direction LOD score using all other causative phenotypes to either or both nodes, according to the DG, in addition to the QTL genotypes. If the direction changes, update the DG and move to the next edge in the ODG.

5. Repeat step 4 until no more edges change direction. The corresponding directed graph is one solution. If step 4 keeps oscillating between different graphs without converging to a solution in 30 steps, we include all distinct graphs as solutions.

6. Repeat steps 3, 4, and 5, n times and store all different solutions. [The argument `n.qdg.random.starts` sets the number of repeats.]

7. Score all solutions in step 6, using a likelihood-based measure of fit of the whole graph, and choose the graph with the highest score.

The argument `marker.nms` is a list of length equal to the number of phenotypes that stores the names of the markers closest to phenotypes' QTLs. `allqtls` stores the respective `qtl` objects. `alpha` represents the significance level threshold for PC or UDG algorithms (for the inference of the graph skeleton. See step 1 of the QDG algorithm). The `addcov` and `intcov` arguments represent the names of the additive and interactive covariates (must be valid phenotypes in the `cross` object).

```
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")
```

The object `out1` has class `qdg`. The component `UDG`, represents the undirected dependency graph (step 1 of the algorithm).

```
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
```

The component `DG` represents the initial QTL directed graph (step 2). The lod score provides a measure of the strength of the direction (positive values when the causal direction is from node 1 to node 2, negative values otherwise).

```
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
```

The component `best.lm` indicates the solution with best fit to the data, after recheck step (steps 3, 4 and 5 of the QDG algorithm). In this case, the `n.qdg.random.starts = 10` initial orderings converged to the same solution after recheck.

```
out1$best.lm
[1] 1
```

The component `Solutions` shows the solutions, after recheck step (single one in this case). The lod score now takes into consideration other phenotypes as covariates. Log-likelihood and BIC scores are also provided.

```
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

$BIC
[1] 7432.697
```

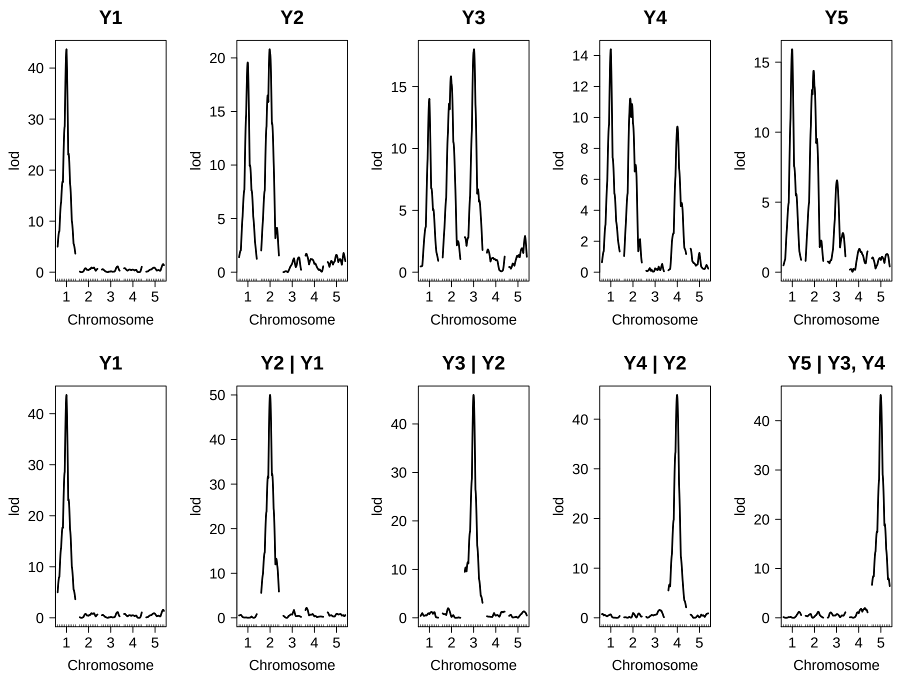
The function `graph.qdg` creates an igraph object than can be plotted. Although the structure of the phenotype network is correct, the genetic architecture is not (compare to Figure 1).

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

2.3 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")
}
```



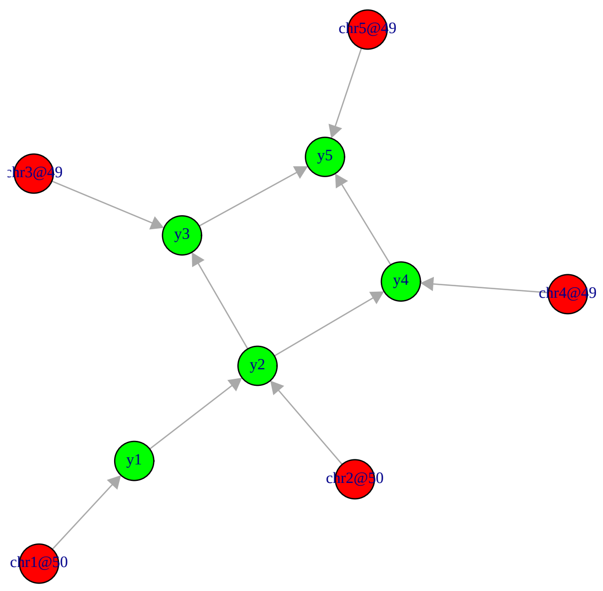
2.4 QTLnet routines

2.4.1 Basic functionality

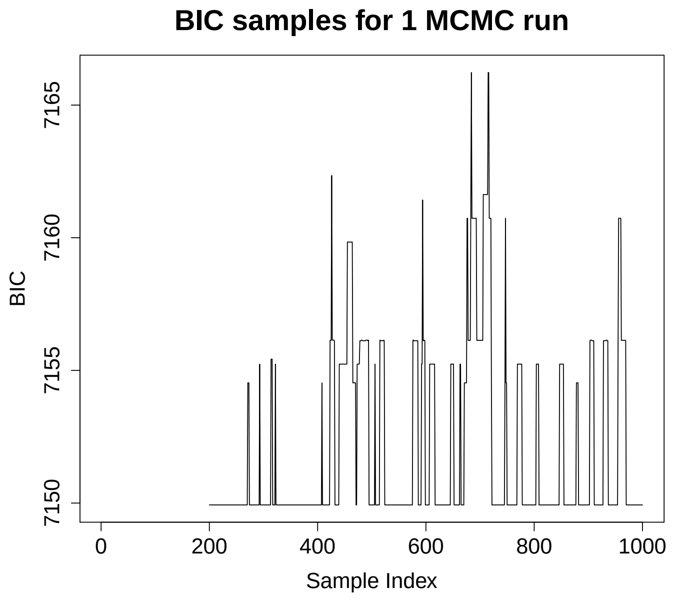
```
out2 <- mcmc.qtlnet(cross = Cross,
                    pheno.col = 1:5,
                    threshold = 3.04,
                    addcov = NULL,
                    intcov = NULL,
                    nSamples = 1000,
                    thinning = 3,
                    max.parents = 4,
        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

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
6     y2    y4 1.000 0.000 0.000
7     y2    y5 0.094 0.000 0.906
8     y3    y4 0.054 0.029 0.917
9     y3    y5 1.000 0.000 0.000
10    y4    y5 1.000 0.000 0.000

Acceptance frequency for MCMC: 0.9996667

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
(1)(2|1)(3|2)(4|1,2)(5|2,3,4)   0.012484395 7159.838
(1)(2|1)(3|1,2)(4|2)(5|1,3,4)   0.012484395 7161.628
(1)(2|1)(3|1,2)(4|2)(5|3,4)     0.003745318 7155.425
(1)(2|1)(3|1,2)(4|1,2)(5|1,3,4) 0.003745318 7166.228
(1)(2|1)(3|2)(4|2,3)(5|2,3,4)   0.001248439 7161.425
```

```
(1)(2|1)(3|2,4)(4|2)(5|1,3,4)   0.001248439 7162.345
Warning message:
In max(pp$prob) : no non-missing arguments to max; returning -Inf

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

$y2
[1] "chr2@50"

$y3
[1] "chr3@49"

$y4
[1] "chr4@49"

$y5
[1] "chr5@49"

plot(out2)
```



```
write.qtlnet(out2, "out2.txt")

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



2.4.2 Parallel implementation

Step 1

```
pheno.col <- 1:5
max.parents <- 4
```

The `size.qtlnet` function 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
```

The `parents.qtlnet` routine 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` calculations. The possible baseline models for QTL analysis with phenotypes as covariates are

$$\begin{aligned} \text{child} &\sim 1\ , \\ \text{child} &\sim \text{parent}_1\ , \\ \text{child} &\sim \text{parent}_1 + \text{parent}_2\ , \\ \text{child} &\sim \text{parent}_1 + \text{parent}_2 + \ldots + \text{parent}_k\ , \end{aligned}$$

where `max.parents = k`. The phenotype names are coded as numbers, according to their order in the `cross` object. The `parents` column shows the possible parent sets (as comma-separated string of parents), and the `n.child` column represents the number of possible child nodes to the respective parent set. For instance, with no parents we perform 5 `scanone` calculations ($y_1 \sim 1$, $y_2 \sim 1$, $y_3 \sim 1$, $y_4 \sim 1$, and $y_5 \sim 1$). With y_1 as a parent we perform 4 `scanone` calculations ($y_2 \sim y_1$, $y_3 \sim y_1$, $y_4 \sim y_1$, and $y_5 \sim y_1$). With y_1 and y_2 as parents we perform 3 `scanone` calculations ($y_3 \sim y_1 + y_2$, $y_4 \sim y_1 + y_2$, and $y_5 \sim y_1 + y_2$), and so on. Note that the sum of the `n.child` column gives the total number of `scanone` computations (80 in this example).

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

The `group.qtlnet` function ...

```
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
```

```
save(Cross, pheno.col, max.parents, threshold, parents, groups,
     file = "Step1.RData", compress = TRUE)
```

Step 2

```
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)
}
```

Step 3

```
n.runs <- 3
load("Step1.RData")

## Read in saved BIC scores and combine into one object.
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)
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

set.seed(54321)
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)
}
```

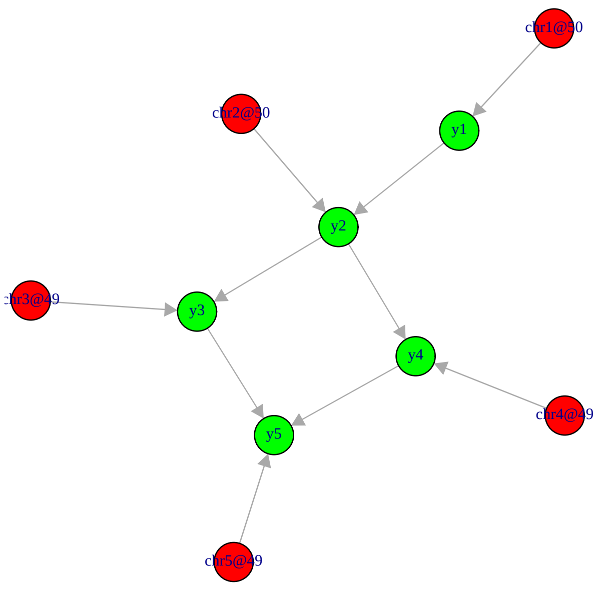
Step 4

```
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)

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
```

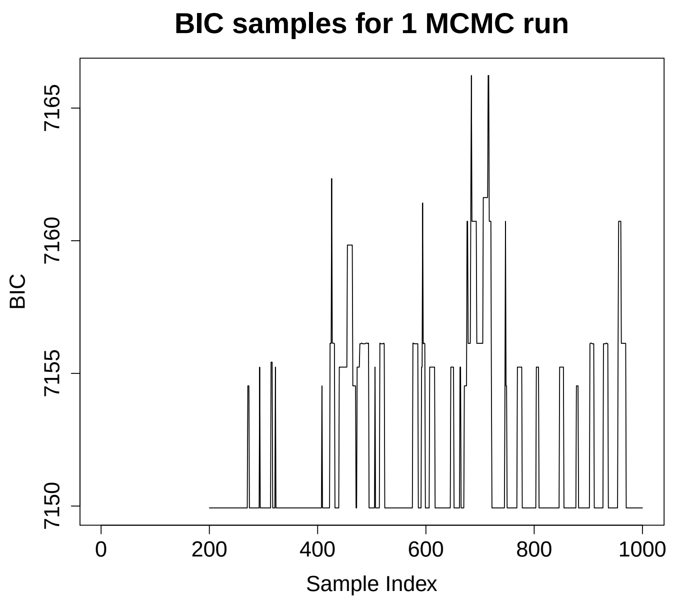
```
4    y3     y5 0.9085185
5    y4     y5 0.9103704

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
6     y2    y4 0.913 0.087 0.000
7     y2    y5 0.138 0.000 0.862
8     y3    y4 0.096 0.132 0.772
9     y3    y5 0.909 0.091 0.000
10    y4    y5 0.910 0.090 0.000

Acceptance frequency for MCMC: 0.999

print(out3)
Model averaged probabilities for edge direction (row -> col):
      [,1]  [,2]  [,3]  [,4]  [,5]
[1,] 0.000 0.916 0.019 0.033 0.028
[2,] 0.084 0.000 0.926 0.913 0.138
[3,] 0.015 0.074 0.000 0.096 0.909
[4,] 0.020 0.087 0.132 0.000 0.910
[5,] 0.006 0.000 0.091 0.090 0.000

Posterior probabilities by causal model:
                                   post.prob      BIC
(1)(2|1)(3|2)(4|2)(5|3,4)       0.6108028117 7149.930
(1)(2|1)(3|2)(4|2)(5|2,3,4)     0.0710321865 7155.238
(1)(2|1)(3|2)(4|2,3)(5|3,4)     0.0506844247 7156.117
(1)(2|1)(3|2,4)(4|2)(5|3,4)     0.0440251572 7156.141
(1|2)(2|3,4)(3|4,5)(4|5)(5)     0.0288568257 7269.336
(1)(2|1)(3|2,4,5)(4|2)(5|2,4)   0.0270070292 7181.425
(1)(2|1)(3|2)(4|2)(5|1,3,4)     0.0240473548 7156.134
(1)(2|1)(3|2)(4|2,3,5)(5|2,3)   0.0229374769 7178.508
(1)(2|1)(3|2)(4|1,2)(5|3,4)     0.0196078431 7154.531
(1)(2|1)(3|1,2)(4|2)(5|3,4)     0.0144284129 7155.425
(1|2,3)(2|3,4)(3|4,5)(4|5)(5)   0.0144284129 7274.417
(1)(2|1,4)(3|2)(4|1)(5|3,4)     0.0129485757 7170.389
(1|2)(2)(3|2)(4|2)(5|3,4)       0.0103588605 7172.978
(1|2,4)(2|3,4)(3|4,5)(4|5)(5)   0.0099889012 7275.475
(1)(2|1)(3|2)(4|2,3)(5|2,3,4)   0.0073991861 7161.425
```

```
(1|2)(2|3,4)(3)(4|3)(5|3,4)      0.0059193489 7228.975
(1|2,4,5)(2|3,4)(3|4,5)(4|5)(5) 0.0055493896 7281.424
(1)(2|1)(3|2)(4|2)(5|1,2,3,4)   0.0036995930 7161.170
(1)(2|1)(3|1,2)(4|2)(5|2,3,4)   0.0033296337 7160.732
(1|2)(2|3,4)(3|5)(4|3,5)(5)     0.0029596744 7272.108
(1)(2|1)(3|2,4)(4|2)(5|2,3,4)   0.0022197558 7161.449
(1|2)(2|3,4)(3)(4|3,5)(5|3)     0.0022197558 7257.083
(1|2,4)(2|3,4)(3|5)(4|3,5)(5)   0.0022197558 7278.246
(1|2,4)(2|3,4)(3)(4|3)(5|3,4)   0.0014798372 7235.113
(1)(2|1)(3|1,2)(4|1,2)(5|3,4)   0.0003699593 7160.025
(1)(2|1)(3|1,2)(4|2)(5|1,2,3,4) 0.0003699593 7166.664
(1|2)(2|3)(3)(4|2)(5|3,4)       0.0003699593 7196.845
(1|2,3)(2|3,4)(3|5)(4|3,5)(5)   0.0003699593 7277.189
(1|3,4)(2|1,3,4)(3|4,5)(4|5)(5) 0.0003699593 7294.377
Warning message:
In max(pp$prob) : no non-missing arguments to max; returning -Inf
```

```
plot(out3)
```



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



3 More examples: both simulated and real data

3.1 QDG routines

Include here the simulated examples of the old tutorial (100 nodes and cyclic nets examples).

3.2 QTLnet routines

Include here the 13 node Pscdb real data example.

4 References

1. Broman et al. (2003)
2. Chaibub Neto et al. (2008) Inferring causal phenotype networks from segregating populations. Genetics 179: 1089-1100.
3. Chaibub Neto et al. (2010) Causal graphical models in systems genetics: a unified framework for joint inference of causal network and genetic archicecture for correlated phenotypes. Annals of Applied Statistics 4: 320-339.
4. Churchill and Doerge
5. Haley amd Knott (1992)
6. Shipley (2002) Cause and Correlation in Biology. Cambridge University Press, Cambridge/London/New York.

7. Spirtes et al. (2000) Causation, Prediction, and Search, Ed. 2. MIT Press, Cambridge, MA.