Causal Inference for QTL Networks with R/qtlnet Package

Elias Chaibub Neto and Brian S. Yandell

April 12, 2020

This vignette briefly describes the R/qtlnet package. This contains the legacy R/qdg package, and thus has code for Chaibub Neto et al. (2008) and Chaibub Neto et al. (2010) papers. Not all routines are described here. Further, the package has code for parallel processing using Condor that is not yet documented adequately.

R/qtlnet depends on R/pcalg, which in turn depends on RBGL from bioconductor. To ease your pain in installing, you can install as follows from within R:

```
> source("http://bioconductor.org/biocLite.R")
> biocLite("RBGL")
> install.packages("qtlnet")
```

This should work on any platform. It is possible to set up R so that it always checks Bioconductor or other repositories, using pull-down menu (Windows) or `.Rprofile` (see text below). See R package documentation for more information.

```
## .Rprofile using multiple repositories.
repos <- structure(c(CRAN="http://streaming.stat.iastate.edu/CRAN",
                     CRANextra="http://www.stats.ox.ac.uk/pub/RWin",
                     BioCsoft="http://www.bioconductor.org/packages/release/bioc",
                     Rforge="http://r-forge.r-project.org"))
options(repos=repos)
```

1 QTLNET routines

Examples not run due to some incompatibility with igraph and new R4.0.

```
> library(qtlnet)
```

Acyclic example:

```
> example(acyclic)
```

Cyclic A example:

```
> example(cyclica)
```

Cyclic B example:

```
> example(cyclicb)
```

Cyclic C example:

```
> example(cyclicc)
```

GLX network example (from Chaibub Neto et al. (2008)):

```
> example(glxnet)
```

2 QDG routines

The QDG routines are now incorporated into R/qtlnet. This document shows how to generate data, fit a QDG model and plot the inferred graph. We focus on a simple graph, y1 -> y3, y2 -> y3 and y3 -> y4, with QTLs that affect each of the three phenotypes.

```
> library(qtlnet)
```

Simulate a genetic map (20 autosomes, 10 not equaly spaced markers per chromosome).

```
> mymap <- sim.map(len=rep(100,20), n.mar=10, eq.spacing=FALSE, include.x=FALSE)
```

Simulate an F2 cross object with n.ind (number of individuals).

```
> n.ind <- 200
> mycross <- sim.cross(map=mymap, n.ind=n.ind, type="f2")
```

Produce multiple imputations of genotypes using the sim.geno function. The makeqtl function requires it, even though we are doing only one imputation (since we don't have missing data and we are using the genotypes in the markers, one imputation is enough).

```
> mycross <- sim.geno(mycross,n.draws=1)
```

Use 2 markers per phenotype, samples from the cross.

```
> genotypes <- pull.geno(mycross)
> geno.names <- dimnames(genotypes)[[2]]
> m1 <- sample(geno.names,2,replace=FALSE)
> m2 <- sample(geno.names,2,replace=FALSE)
> m3 <- sample(geno.names,2,replace=FALSE)
> m4 <- sample(geno.names,2,replace=FALSE)
> ## get marker genotypes
> g11 <- genotypes[,m1[1]]; g12 <- genotypes[,m1[2]]
> g21 <- genotypes[,m2[1]]; g22 <- genotypes[,m2[2]]
> g31 <- genotypes[,m3[1]]; g32 <- genotypes[,m3[2]]
> g41 <- genotypes[,m4[1]]; g42 <- genotypes[,m4[2]]
> ## generate phenotypes
> y1 <- runif(3,0.5,1)[g11] + runif(3,0.5,1)[g12] + rnorm(n.ind)
> y2 <- runif(3,0.5,1)[g21] + runif(3,0.5,1)[g22] + rnorm(n.ind)
> y3 <- runif(1,0.5,1) * y1 +  runif(1,0.5,1) * y2 + runif(3,0.5,1)[g31] + runif(3,0.5,1)[g32] + rnorm(
> y4 <- runif(1,0.5,1) * y3 + runif(3,0.5,1)[g41] + runif(3,0.5,1)[g42] + rnorm(n.ind)
```

Incorporate phenotypes into cross object.

```
> mycross$pheno <- data.frame(y1,y2,y3,y4)
```

Create markers list.

```
> markers <- list(m1,m2,m3,m4)
> names(markers) <- c("y1","y2","y3","y4")
```

Create qtl object.

```
> allqtls <- list()
> m1.pos <- find.markerpos(mycross, m1)
> allqtls[[1]] <- makeqtl(mycross, chr = m1.pos[,"chr"], pos = m1.pos[,"pos"])
> m2.pos <- find.markerpos(mycross, m2)
> allqtls[[2]] <- makeqtl(mycross, chr = m2.pos[,"chr"], pos = m2.pos[,"pos"])
```

```
> m3.pos <- find.markerpos(mycross, m3)
> allqtls[[3]] <- makeqtl(mycross, chr = m3.pos[,"chr"], pos = m3.pos[,"pos"])
> m4.pos <- find.markerpos(mycross, m4)
> allqtls[[4]] <- makeqtl(mycross, chr = m4.pos[,"chr"], pos = m4.pos[,"pos"])
> names(allqtls) <- c("y1","y2","y3","y4")
```

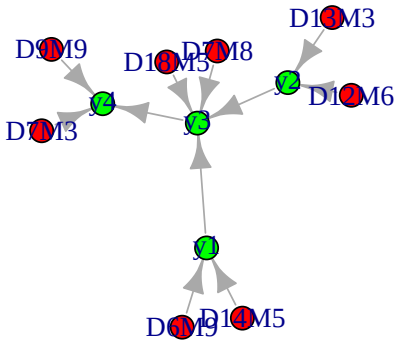
Infer QDG object.

```
> out <- qdg(cross=mycross,
+            phenotype.names = c("y1","y2","y3","y4"),
+            marker.names = markers,
+            QTL = allqtls,
+            alpha = 0.005,
+            n.qdg.random.starts=10,
+            skel.method="pcskel")
> out

$UDG
  node1 node2 edge
1    y1    y3    1
2    y2    y3    1
5    y3    y4    1

$DG
  node1 direction node2 lod score
1    y1     ---->    y3 0.3283315
2    y2     ---->    y3 2.6184065
3    y3     ---->    y4 1.6685702

$best.lm
[1] 1

$Solutions
$Solutions$solutions
$Solutions$solutions[[1]]
  node1 direction node2       lod
1    y1     ---->    y3  4.114187
2    y2     ---->    y3  6.404262
3    y3     ---->    y4 12.344708


$Solutions$loglikelihood
[1] -1127.196

$Solutions$BIC
[1] 2397.446


$marker.names
$marker.names$y1
[1] "D14M5" "D6M9"

$marker.names$y2
[1] "D12M6" "D13M3"
```

```
$marker.names$y3
[1] "D18M5" "D7M8"

$marker.names$y4
[1] "D9M9" "D7M3"


$phenotype.names
[1] "y1" "y2" "y3" "y4"

attr(,"class")
[1] "qdg"  "list"
```

Plot object. The graph is an object of class igraph, which can be plotted using the igraph package.

```
> graph <- graph.qdg(out)
> plot(graph)
```



You can use tkplot() for an interactive plot.