Package 'qtlnet'

May 9, 2026

Version 1.5.4

Date 2020-04-12

Author Elias Chaibub Neto <echaibub@hotmail.com> and Brian S. yandell <brian.yandell@wisc.edu>

Title Causal Inference of QTL Networks

Description Functions to Simultaneously Infer Causal Graphs and Genetic Architecture. Includes acyclic and cyclic graphs for data from an experimental cross with a modest number (<10) of phenotypes driven by a few genetic loci (QTL). Chaibub Neto E, Keller MP, Attie AD, Yandell BS (2010) Causal Graphical Models in Systems Genetics: a unified framework for joint inference of causal network and genetic architecture for correlated phenotypes. Annals of Applied Statistics 4: 320-339. <doi:10.1214/09-AOAS288>.

Maintainer Brian S. Yandell <brian.yandell@wisc.edu>

Depends qtl,igraph,sem,graph,pcalg, R (>= 2.10)

LazyLoad yes

LazyData yes

License GPL (>= 2)

URL http://www.stat.wisc.edu/~yandell/sysgen

RoxygenNote 6.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2020-04-13 04:30:02 UTC

Contents

`acyclic` *Acyclic graph example*

Description

We generate synthetic data (sample size 300) according to a DAG composed by 100 nodes and 107 edges (exactly as in Figure 1). Each phenotype node is affected by three QTLs, and we allow only additive genetic effects. The QTLs for each phenotype are randomly selected among 200 markers, with 10 markers unevenly distributed on each of 20 autosomes. We allowed different phenotypes to potentially share common QTLs. For each phenotype, the regression coefficients with other phenotypes are chosen uniformly between 0.5 and 1; QTL effects are chosen between 0.2 to 0.6; and residual standard deviations are chosen from 0.1 to 0.5. For each realization we apply the QDG algorithm to infer causal directions for the edges of the skeleton obtained by the PC-skeleton algorithm.

Usage

```
data(acyclic)
```

Details

For cyclic graphs, the output of the qdg function computes the log-likelihood up to the normalization constant (un-normalized log-likelihood). We can use the un-normalized log-likelihood to compare cyclic graphs with reversed directions (since they have the same normalization constant). However we cannot compare cyclic and acyclic graphs.

References

Chaibub Neto et al. (2008) Inferring causal phenotype networks from segregating populations. Genetics 179: 1089-1100.

See Also

sim.cross, sim.geno, sim.map, skeleton, qdg, graph.qdg, generate.qtl.pheno

Examples

```
## Not run:
## This reproduces Figure 1 exactly.
set.seed(3456789)

tmp <- options(warn=-1)
acyclic.DG <- randomDAG(n = 100, prob = 2 / 99)

options(tmp)

## Simulate cross object using R/qtl routines.
n.ind <- 300
mymap <- sim.map(len=rep(100,20), n.mar=10, eq.spacing=FALSE, include.x=FALSE)
mycross <- sim.cross(map=mymap, n.ind=n.ind, type="f2")
summary(mycross)
mycross <- sim.geno(mycross,n.draws=1)


## Produce 100 QTL at three markers apiece.
acyclic.qtl <- generate.qtl.markers(cross=mycross,n.phe=100)

## Generate data from directed graph.
bp <- runif(100,0.5,1)
stdev <- runif(100,0.1,0.5)
bq <- matrix(0,100,3)
bq[,1] <- runif(100,0.2,0.4)
bq[,2] <- bq[,1]+0.1
bq[,3] <- bq[,2]+0.1
## Generate phenotypes.
acyclic.data <- generate.qtl.pheno("acyclic", cross = mycross,
  bp = bp, bq = bq, stdev = stdev, allqtl = acyclic.qtl$allqtl)

acyclic.qdg <- qdg(cross=acyclic.data,
phenotype.names=paste("y",1:100,sep=""),
marker.names=acyclic.qtl$markers,
QTL=acyclic.qtl$allqtl,
alpha=0.005,
n.qdg.random.starts=1,
skel.method="pcskel")
save(acyclic.DG, acyclic.qtl, acyclic.data, acyclic.qdg,
  file = "acyclic.RData", compress = TRUE)

data(acyclic)

dims <- dim(acyclic.data$pheno)
SuffStat <- list(C = cor(acyclic.data$pheno), n = dims[1])
pc <- skeleton(SuffStat, gaussCItest, p = dims[2], alpha = 0.005)
summary(pc)
```

```
summary(graph.qdg(acyclic.qdg))
gr <- graph.qdg(acyclic.qdg, include.qtl = FALSE)
plot(gr)

## End(Not run)
```

`bic.qtlnet` *Pre-compute BIC values for qtlnet sampling.*

Description

Pre-compute BIC values for qtlnet sampling to speed up MCMC sampling.

Usage

```
bic.qtlnet(cross, pheno.col, threshold, addcov = NULL, intcov = NULL,
  max.parents = 3, parents, verbose = TRUE, ...)
bic.join(cross, pheno.col, ..., max.parents = 3)
data(Pscdbp.bic)
```

Arguments

`cross`	Object of class `cross`. See `read.cross`.
`pheno.col`	Phenotype identifiers from `cross` object. May be numeric, logical or character.
`threshold`	Scalar or list of thresholds, one per each node.
`addcov`	Additive covariates for each phenotype (`NULL` if not used). If entered as scalar or vector (same format as `pheno.col`), then the same `addcov` is used for all phenotypes. Altenatively, may be a list of additive covariate identifiers.
`intcov`	Interactive covariates, entered in the same manner as `addcov`.
`max.parents`	Maximum number of parents per node. This reduces the complexity of graphs and shortens run time. Probably best to consider values of 3-5.
`parents`	List containing all possible parents up to `max.parents` in size. May be a subset
`verbose`	Print iteration and number of models fit.
`...`	Additional arguments passed to internal routines. In the case of `bic.join`, these are a list of objects produced by `bic.qtlnet` (see example below).

Details

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

We divide the job into three 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. See the example for details.

Value

Matrix with columns:

code	Binary code as decimal for the parents of a phenotype node, excluding the phenotype. Value is between 0 (no parents) and 2 ^ (length(pheno.col) - 1).
pheno.col	Phenotype column in reduced set, in range 1:length(pheno.col).
bic	BIC score for phenotype conditional on parents (and covariates).

Author(s)

Brian S. Yandell and Elias Chaibub Neto

References

Chaibub Neto E, Keller MP, Attie AD, Yandell BS (2010) Causal Graphical Models in Systems Genetics: a unified framework for joint inference of causal network and genetic architecture for correlated phenotypes. Ann Appl Statist 4: 320-339. http://dx.doi.org/10.1214/09-AOAS288

See Also

mcmc.qtlnet, parents.qtlnet

Examples

```
pheno.col <- 1:13
max.parents <- 12
size.qtlnet(pheno.col, max.parents)

## Not run:
## Compute all phenotype/parent combinations.
## This shows how to break up into many smaller jobs.

########################################
## STEP 1: Preparation. Fast. Needed in steps 2 and 3.

pheno.col <- 1:13
max.parents <- 12
threshold <- 3.83

## Load cross object. Here we use internal object.
data(Pscdbp)
## or: load("Pscdbp.RData")
cross <- Pscdbp
## or: cross <- read.cross("Pscdbp.csv", "csv")

## Break up into groups to run on several machines.
## ~53 groups of ~1000, for a total of 53248 scanone runs.
parents <- parents.qtlnet(pheno.col, max.parents)
groups <- group.qtlnet(parents = parents, group.size = 1000)

## Save all relevant objects for later steps.
```

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

########################################
## STEP 2: Compute BIC scores. Parallelize.

## NB: Configuration of parallelization determined using Step 1 results.
## Load Step 1 computations.
load("Step1.RData")

## Parallelize this:
for(i in seq(nrow(groups)))
{
  ## Pre-compute BIC scores for selected parents.
  bic <- bic.qtlnet(cross, pheno.col, threshold,
    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: Sample Markov chain (MCMC). Parallelize.

## NB: n.runs sets the number of parallel runs.
n.runs <- 100

## Load Step 1 computations.
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
}
saved.scores <- bic.join(cross, pheno.col, bic.group)


## Parallelize this:
for(i in seq(n.runs))
{
  ## Run MCMC with randomized initial network.
  mcmc <- mcmc.qtlnet(cross, pheno.col, threshold = threshold,
    max.parents = max.parents, saved.scores = saved.scores, init.edges = NULL)

  save(mcmc, file = paste("mcmc", i, ".RData", sep = ""), compress = TRUE)
}

########################################
## STEP 4: Combine results for post-processing.
```

```
## NB: n.runs needed from Step 3.
n.runs <- 100

## Combine outputs together.
outs.qtlnet <- list()
for(i in seq(n.runs))
{
  load(paste("mcmc", i, ".RData", sep = ""))
  outs.qtlnet[[i]] <- mcmc
}
out.qtlnet <- c.qtlnet(outs.qtlnet)
summary(out.qtlnet)
print(out.qtlnet)

## End of parallel example.
########################################

## End(Not run)

dim(Pscdbp.bic)
```

`cyclica` *Cyclic graph (a) example*

Description

We use a Gibbs sampling scheme to generate a data-set with 200 individuals (according with cyclic graph (a)). Each phenotype is affected by 3 QTLs. We fixed the regression coefficients at 0.5, error variances at 0.025 and the QTL effects at 0.2, 0.3 and 0.4 for the three F2 genotypes. We used a burn-in of 2000 for the Gibbs sampler.

Usage

```
data(cyclica)
```

Details

For cyclic graphs, the output of the qdg function computes the log-likelihood up to the normalization constant (un-normalized log-likelihood). We can use the un-normalized log-likelihood to compare cyclic graphs with reversed directions (they have the same normalization constant). However we cannot compare cyclic and acyclic graphs.

References

Chaibub Neto et al. (2008) Inferring causal phenotype networks from segregating populations. Genetics 179: 1089-1100.

See Also

`sim.cross`, `sim.geno`, `sim.map`, `skeleton`, `qdg`, `graph.qdg`, `generate.qtl.pheno`

Examples

```
## Not run:
bp <- matrix(0, 6, 6)
bp[2,1] <- bp[4,2] <- bp[4,3] <- bp[5,4] <- bp[2,5] <- bp[6,5] <- 0.5
stdev <- rep(0.025, 6)

## Use R/qtl routines to simulate.
set.seed(3456789)
mymap <- sim.map(len = rep(100,20), n.mar = 10, eq.spacing = FALSE,
  include.x = FALSE)
mycross <- sim.cross(map = mymap, n.ind = 200, type = "f2")
mycross <- sim.geno(mycross, n.draws = 1)

cyclica.qtl <- generate.qtl.markers(cross = mycross, n.phe = 6)
mygeno <- pull.geno(mycross)[, unlist(cyclica.qtl$markers)]

cyclica.data <- generate.qtl.pheno("cyclica", cross = mycross, burnin = 2000,
  bq = c(0.2,0.3,0.4), bp = bp, stdev = stdev, geno = mygeno)
save(cyclica.qtl, cyclica.data, file = "cyclica.RData", compress = TRUE)

data(cyclica)
out <- qdg(cross=cyclica.data,
phenotype.names=paste("y",1:6,sep=""),
marker.names=cyclica.qtl$markers,
QTL=cyclica.qtl$allqtl,
alpha=0.005,
n.qdg.random.starts=10,
skel.method="pcskel")


gr <- graph.qdg(out)
gr
plot(gr)

## End(Not run)
```

`cyclicb` *Cyclic graph (b) example*

Description

We use a Gibbs sampling scheme to generate a data-set with 200 individuals (according with cyclic graph (b)). Each phenotype is affected by 3 QTLs. We fixed the regression coefficients at 0.5, error variances at 0.025 and the QTL effects at 0.2, 0.3 and 0.4 for the three F2 genotypes. We used a burn-in of 2000 for the Gibbs sampler.

Usage

```
data(cyclicb)
```

Details

For cyclic graphs, the output of the qdg function computes the log-likelihood up to the normalization constant (un-normalized log-likelihood). We can use the un-normalized log-likelihood to compare cyclic graphs with reversed directions (since they have the same normalization constant). However we cannot compare cyclic and acyclic graphs.

References

Chaibub Neto et al. (2008) Inferring causal phenotype networks from segregating populations. Genetics 179: 1089-1100.

See Also

sim.cross, sim.geno, sim.map, skeleton, qdg, graph.qdg, generate.qtl.pheno

Examples

```
## Not run:
bp <- matrix(0, 6, 6)
bp[2,1] <- bp[1,5] <- bp[3,1] <- bp[4,2] <- bp[5,4] <- bp[5,6] <- bp[6,3] <- 0.5
stdev <- rep(0.025, 6)

## Use R/qtl routines to simulate.
set.seed(3456789)
mymap <- sim.map(len = rep(100,20), n.mar = 10, eq.spacing = FALSE,
  include.x = FALSE)
mycross <- sim.cross(map = mymap, n.ind = 200, type = "f2")
mycross <- sim.geno(mycross, n.draws = 1)

cyclicb.qtl <- generate.qtl.markers(cross = mycross, n.phe = 6)
mygeno <- pull.geno(mycross)[, unlist(cyclicb.qtl$markers)]

cyclicb.data <- generate.qtl.pheno("cyclicb", cross = mycross, burnin = 2000,
  bq = c(0.2,0.3,0.4), bp = bp, stdev = stdev, geno = mygeno)
save(cyclicb.qtl, cyclicb.data, file = "cyclicb.RData", compress = TRUE)

data(cyclicb)
out <- qdg(cross=cyclicb.data,
phenotype.names=paste("y",1:6,sep=""),
marker.names=cyclicb.qtl$markers,
QTL=cyclicb.qtl$allqtl,
alpha=0.005,
n.qdg.random.starts=10,
skel.method="pcskel")

gr <- graph.qdg(out)
gr
plot(gr)

## End(Not run)
```

cyclicc *Cyclic graph (c) example*

Description

We use a Gibbs sampling scheme to generate a data-set with 200 individuals (according with cyclic graph (c)). Each phenotype is affected by 3 QTLs. We fixed the regression coefficients at 0.5, (except for beta[5,2]=0.8) error variances at 0.025 and the QTL effects at 0.2, 0.3 and 0.4 for the three F2 genotypes. We used a burn-in of 2000 for the Gibbs sampler. This example illustrates that even though our method cannot detect reciprocal interactions (e.g. between phenotypes 2 and 5 in cyclic graph (c)), it can still infer the stronger direction, that is, the direction corresponding to the higher regression coefficient. Since beta[5,2] is greater than beta[2,5], the QDG method should infer the direction from 2 to 5.

Usage

```
data(cyclicc)
```

Details

For cyclic graphs, the output of the qdg function computes the log-likelihood up to the normalization constant (un-normalized log-likelihood). We can use the un-normalized log-likelihood to compare cyclic graphs with reversed directions (since they have the same normalization constant). However we cannot compare cyclic and acyclic graphs.

References

Chaibub Neto et al. (2008) Inferring causal phenotype networks from segregating populations. Genetics 179: 1089-1100.

See Also

sim.cross, sim.geno, sim.map, skeleton, qdg, graph.qdg, generate.qtl.pheno

Examples

```
## Not run:
bp <- matrix(0, 6, 6)
bp[2,5] <- 0.5
bp[5,2] <- 0.8
bp[2,1] <- bp[3,2] <- bp[5,4] <- bp[6,5] <- 0.5
stdev <- rep(0.025, 6)

## Use R/qtl routines to simulate map and genotypes.
set.seed(34567899)
mymap <- sim.map(len = rep(100,20), n.mar = 10, eq.spacing = FALSE,
  include.x = FALSE)
mycross <- sim.cross(map = mymap, n.ind = 200, type = "f2")
mycross <- sim.geno(mycross, n.draws = 1)
```

```
## Use R/qdg routines to produce QTL sample and generate phenotypes.
cyclicc.qtl <- generate.qtl.markers(cross = mycross, n.phe = 6)
mygeno <- pull.geno(mycross)[, unlist(cyclicc.qtl$markers)]

cyclicc.data <- generate.qtl.pheno("cyclicc", cross = mycross, burnin = 2000,
  bq = c(0.2,0.3,0.4), bp = bp, stdev = stdev, geno = mygeno)
save(cyclicc.qtl, cyclicc.data, file = "cyclicc.RData", compress = TRUE)

data(cyclicc)
out <- qdg(cross=cyclicc.data,
phenotype.names=paste("y",1:6,sep=""),
marker.names=cyclicc.qtl$markers,
QTL=cyclicc.qtl$allqtl,
alpha=0.005,
n.qdg.random.starts=1,
skel.method="pcskel")

gr <- graph.qdg(out)
plot(gr)

## End(Not run)
```

`dist.qtlnet` *QTL network diagnostic routines*

Description

Various QTLnet diagnostic routines.

Usage

```
dist.qtlnet(qtlnet.object, min.prob = 0.9, method = "manhattan", cex = 5)
edgematch.qtlnet(qtlnet.object, min.prob = 0.9, method = "manhattan", cex = 5)
mds.qtlnet(qtlnet.object, min.prob = 0.9, method = "manhattan", cex = 5)
plotbic.qtlnet(x, ..., smooth = TRUE)
```

Arguments

`qtlnet.object`, `x`	Object of class `qtlnet`.
`min.prob`	Minimum probability to include edge in network.
`method`	Distance method to be used between columns of connection matrix. Used by `dist`. (Only used for `mds.qtlnet`.)
`cex`	Character expansion. (Only used for `mds.qtlnet`, scaled by range of BIC values.)
`smooth`	Use `lowess` smoother if `TRUE`.
`...`	Additional unused arguments.

Value

List containing, for each phenotype in the network, a character vector of the QTL names as `chr@pos`, or pseudomarker name if `chr.pos` is `FALSE`.

Author(s)

Brian S. Yandell and Elias Chaibub Neto

References

Chaibub Neto E, Keller MP, Attie AD, Yandell BS (2010) Causal Graphical Models in Systems Genetics: a unified framework for joint inference of causal network and genetic architecture for correlated phenotypes. Ann Appl Statist 4: 320-339. http://www.stat.wisc.edu/~yandell/doc/2010/92.AnnApplStat.pdf

See Also

`mcmc.qtlnet`

Examples

```
loci.qtlnet(Pscdbp.qtlnet)
```

`generate.qtl` *Generate QTLs ane phenotypes from cross object*

Description

Generate QTLs ane phenotype data for individual examples from cross. These are utility routines to illustrate the examples. They are not meant for users per se.

Usage

```
generate.qtl.markers(cross, n.phe, nqtl = 3)
generate.qtl.pheno(name, cross, bp, bq, stdev, allqtl,
    burnin = 2000, geno)
```

Arguments

`cross`	object of class `cross`; see `read.cross`
`name`	character string for example name
`bp`	vector or matrix of coefficients for dependencies between phenotypes; see cyclic and acyclic examples
`bq`	vector or matrix of coefficients for QTL effects on phenotypes; see cyclic and acyclic examples
`stdev`	vector of standard deviations per phenotype

allqtl	list of objects of class qtl produced by generate.qtl.sample
burnin	number of burnin cycles for MCMC; default is 2000
geno	genotypes at markers, typically extracted with pull.geno
n.phe	number of phenotypes
nqtl	number of QTL

See Also

acyclic, cyclica, cyclicb, cyclicc

Examples

```
## Not run:
example(acyclic)
example(cyclica)
example(cyclicb)
example(cyclicc)

## End(Not run)
```

---

glxnet *Generate and graph Glx network*

Description

This is the Glx network reported in Chaibub Neto et al 2008 and in Ferrara et al 2008. Age was used as an additive covariate and we allowed for sex by genotype interaction. The network differs slightly from the published network due to improved code.

References

Chaibub Neto et al. 2008 Inferring causal phenotype networks from segregating populations. Genetics 179: 1089-1100.

Ferrara et al. 2008 Genetic networks of liver metabolism revealed by integration of metabolomic and transcriptomic profiling. PLoS Genetics 4: e1000034.

See Also

qdg

Examples

```
data(glxnet)
glxnet.cross <- calc.genoprob(glxnet.cross)
set.seed(1234)
glxnet.cross <- sim.geno(glxnet.cross)

n.node <- nphe(glxnet.cross) - 2 ## Last two are age and sex.
markers <- glxnet.qtl <- vector("list", n.node)
for(i in 1:n.node) {
    ac <- model.matrix(~ age + sex, glxnet.cross$pheno)[, -1]
    ss <- summary(scanone(glxnet.cross, pheno.col = i,
                          addcovar = ac, intcovar = ac[,2]),
                  threshold = 2.999)
    glxnet.qtl[[i]] <- makeqtl(glxnet.cross, chr = ss$chr, pos = ss$pos)
    markers[[i]] <- find.marker(glxnet.cross, chr = ss$chr, pos = ss$pos)
}
names(glxnet.qtl) <- names(markers) <- names(glxnet.cross$pheno)[seq(n.node)]

glxnet.qdg <- qdg(cross=glxnet.cross,
phenotype.names = names(glxnet.cross$pheno[,seq(n.node)]),
marker.names = markers,
QTL = glxnet.qtl,
alpha = 0.05,
n.qdg.random.starts=10,
addcov="age",
intcov="sex",
skel.method="udgskel",
udg.order=6)
glxnet.qdg

## Not run:
gr <- graph.qdg(glxnet.qdg)
plot(gr)

## Or use tkplot().
glxnet.cross <- clean(glxnet.cross)
save(glxnet.cross, glxnet.qdg, glxnet.qtl, file = "glxnet.RData", compress = TRUE)

## End(Not run)
```

---

igraph.qtlnet *qtlnet plot using igraph*

Description

Plot inferred causal network using igraph package.

Usage

```
igraph.qtlnet(x, edges, loci.list,
  pheno.color = "green", qtl.color = "red", vertex.color,
  include.qtl = TRUE, ...)
## S3 method for class 'qtlnet'
plot(x, ...)
```

Arguments

x	Object of class qtlnet.
edges	Data frame with first two columns being cause and effect directed phenotype pairs. Typically determined as averaged.net element from call to summary.qtlnet.
loci.list	List of character names of loci by phenotype. Typically determined by call to loci.qtlnet.
pheno.color, qtl.color	Name of color to use for phenotypes and QTLs, respectively.
vertex.color	Vertex colors in order of pheno-pheno edged augmented by loci.list, by default determined by pheno.color and qtl.color.
include.qtl	Include QTL in graph if TRUE and loci.list is not NULL.
...	Additional arguments passed to called routines.

Details

Uses the igraph package to create graph objects. These can be exported to a variety of other modern graphics packages. graph.qtlnet is synonymous with igraph.qtlnet.

Value

Object of class graph created using graph.data.frame.

Author(s)

Brian S. Yandell and Elias Chaibub Neto

References

Chaibub Neto E, Keller MP, Attie AD, Yandell BS (2010) Causal Graphical Models in Systems Genetics: a unified framework for joint inference of causal network and genetic architecture for correlated phenotypes. Ann Appl Statist 4: 320-339. http://www.stat.wisc.edu/~yandell/doc/2010/92.AnnApplStat.pdf

See Also

summary.qtlnet, loci.qtlnet, graph.data.frame, tkplot

Examples

```
Pscdbp.graph <- igraph.qtlnet(Pscdbp.qtlnet)
Pscdbp.graph
## Not run:
tkplot(Pscdbp.graph)

## End(Not run)
```

`loci.qtlnet` *QTL architecture per node as list*

Description

Determines QTL that affect each phenotype conditional on the model-averaged network and on covariates.

Usage

```
loci.qtlnet(qtlnet.object, chr.pos = TRUE, merge.qtl = 10, ...)
est.qtlnet(qtlnet.object, ..., verbose = TRUE)
```

Arguments

`qtlnet.object`	Object of class `qtlnet`.
`chr.pos`	Include chromsome and position if `TRUE`.
`merge.qtl`	Merge QTL within `merge.qtl` cM of the mean QTL per chromosome across all nodes. No merge if 0 or less. This can reduce the number of QTL nodes to one per chr.
`...`	Additional unused arguments.
`verbose`	verbose output if `TRUE`.

Value

List containing, for each phenotype in the network, a character vector of the QTL names as `chr@pos`, or pseudomarker name if `chr.pos` is `FALSE`.

Author(s)

Brian S. Yandell and Elias Chaibub Neto

References

Chaibub Neto E, Keller MP, Attie AD, Yandell BS (2010) Causal Graphical Models in Systems Genetics: a unified framework for joint inference of causal network and genetic architecture for correlated phenotypes. Ann Appl Statist 4: 320-339. http://www.stat.wisc.edu/~yandell/doc/2010/92.AnnApplStat.pdf

See Also

mcmc.qtlnet

Examples

```
loci.qtlnet(Pscdbp.qtlnet)
```

---

mcmc.qtlnet *Sample genetic architecture and QTL network*

Description

Use MCMC to alternatively sample genetic architecture and QTL network as directed acyclic graphs (DAGs).

Usage

```
mcmc.qtlnet(cross, pheno.col, threshold, addcov = NULL, intcov = NULL,
  nSamples = 1000, thinning = 1, max.parents = 3, M0 = NULL,
  burnin = 0.1, method = "hk", random.seed = NULL, init.edges = 0,
  saved.scores = NULL, rev.method = c("nbhd", "node.edge", "single"),
  verbose = FALSE, ...)
init.qtlnet(pheno.col, max.parents, init.edges)
```

Arguments

cross	Object of class cross. See read.cross.
pheno.col	Phenotype identifiers from cross object. May be numeric, logical or character.
threshold	Scalar or list of thresholds, one per each node.
addcov	Additive covariates for each phenotype (NULL if not used). If entered as scalar or vector (same format as pheno.col), then the same addcov is used for all phenotypes. Altenatively, may be a list of additive covariate identifiers.
intcov	Interactive covariates, entered in the same manner as addcov.
nSamples	Number of samples to record.
thinning	Thinning rate. Number of MCMC samples is nSamples*thinning.
max.parents	Maximum number of parents to a node. This reduces the complexity of graphs and shortens run time. Probably best to consider values of 3-5.
M0	Matrix of 0s and 1s with initial directed graph of row->col if (row,col) entry is 1. Cycles are forbidden (e.g. 1s on diagonal or symmetric 1s across diagonal). Default (if NULL) is sampled by a call to init.qtlnet; all 0s if init.edges = 0 (default).
burnin	Proportion of MCMC samples to use as burnin. Default is 0.1 if burnin is TRUE. Must be between 0 and 1.
method	Model fitting method for scanone.

random.seed	Initialization seed for random number generator. Must be NULL (no reset) or positive numeric. Used in Random.
init.edges	Initial number of edges for M0, to be sampled using {init.qtlnet}. Chosen uniformly from 0 to the number of possible edges if set to NULL.
saved.scores	Updated scores, typically pre-computed by bic.qtlnet.
rev.method	Method to use for reversing edges. See details.
verbose	Print iteration and number of models fit.
...	Additional arguments. Advanced users may want to supply pre-computed saved.scores to speed up calculations.

Details

Models are coded compactly as (1)(2|1)(3|1,2,4,5)(4|2)(5|2). Each parenthetical entry is a of form (node|parents); these each require a model fit, for now with scanone.

The scanone routine is run on multiple phenotypes in the network that could all have the same parents. For instance, for 5 phenotypes, if (1|2,4) is sampled, then do scanone of this model as well as (3|2,4) and (5|2,4). Setting the hidden parameter scan.parents to a value smaller than length(pheno.col) - 1 (default) disallows multiple trait scanning with more than that number of parents.

The saved.scores parameter can greatly reduce MCMC run time, by supplying pre-computed BIC scores. See bic.qtlnet. Another option is to capture saved.scores from a previous mcmc.qtlnet run with the same phenotypes (and covariates). Caution is advised as only a modest amount of checking can be done.

The init.qtlnet routine can be used to randomly find an initial causal network M0 with up to init.edges edges.

MCMC updates include delete, add or reverse edge direction. The early version of this method only considered the edge on its own (rev.method = "single"), while the neighborhood method (rev.method = "nbhd") uses the update

Value

List of class qtlnet

post.model	Model code (see details).
post.bic	Posterior BIC
Mav	Model average of M across MCMC samples.
freq.accept	Frequency of acceptance M-H proposals.
saved.scores	Saved LOD score for each phenotype and all possible sets of the other phenotypes as parent nodes.
all.bic	
cross	The cross object with calculated genotype probabilities.

In addition, a number of attributes are recorded:

M0	Initial network matrix.

threshold	threshold list
nSamples	Number of samples saved
thinning	Thinning rate
pheno.col	Phenotype columns.
pheno.names	Phenotype names
addcov	Additive covariate columns.
intcov	Interactive covariate columns.
burnin	Burnin proportion
method	Method used for scanone.
random.seed	Initial random number generator seed.
random.kind	Random number generator kind from Random.

Author(s)

Brian S. Yandell and Elias Chaibub Neto

References

Chiabub Neto E, Keller MP, Attie AD, Yandell BS (2010) Causal graphical models in systems genetics: a unified framework for joint inference of causal network and genetic archicecture for correlated phenotypes. Ann Appl Statist 4: 320-339. http://dx.doi.org/10.1214/09-AOAS288

Grzegorczyk and Husmeier (2008) Improving the structure MCMC sampler for Bayesian networks by introducing a new edge reversal move. Mach Learn 71: 265-305. http://dx.doi.org/10.1007/s10994-008-5057-7

See Also

read.cross, scanone, Random, bic.qtlnet.

Examples

```
data(Pscdbp)
## Not run:
  ## Run of subset of traits. Still takes some time.
  Pscdbp.qtlnet <- mcmc.qtlnet(Pscdbp, pheno.col = c(1,2,4,5,6),
                               threshold = 3.83,
                               nSamples = 1000, thinning = 20,
                               random.seed = 92387475, verbose = TRUE)
  save(Pscdbp.qtlnet, file = "Pscdbp.qtlnet.RData", compress = TRUE)

## End(Not run)
data(Pscdbp.qtlnet)

## Not run:
  out.qtlnet <- mcmc.qtlnet(Pscdbp, pheno.col = 1:13,
                            threshold = 3.83,
                            nSamples = 1000, thinning = 20,
```

```
                        random.seed = 92387475, verbose = TRUE,
                        saved.scores = Pscdbp.bic)

## End(Not run)
```

`parallel.qtlnet` *Code to parallelize use of qtlnet*

Description

This routine calls one of four phases in a parallelized version of qtlnet.

Usage

```
parallel.qtlnet(phase, index = 1, ..., dirpath = ".")
```

Arguments

`phase`	Phase of parallelization as number 1 through 4. See details.
`index`	Index for phase. Used in phases 2 and 4, and for error codes saved in `RESULT.phase.index` file.
`...`	Additional arguments for phases. See details.
`dirpath`	Character string for directory were user can read and write files. When submitting to a cluster, this should remain the default.

Details

See http://www.stat.wisc.edu/~yandell/sysgen/qtlnet for details of implementation in progress. The plan is to run qtlnet via Condor (https://research.cs.wisc.edu/htcondor/) to scale up to larger networks, say up to 100 nodes. Most important information is passed in files. Phase 1 imports arguments from the `params.txt` file, which must have parse-able assignments to the arguments of `qtlnet:::qtlnet.phase1`. This first phase produces file `Phase1.RData`, which included objects used by all other phases.

Phase 1 also creates file `groups.txt`, which for each line has begin and end indices for the `parents` that would result from a call to `parents.qtlnet`. Phase 2 should be run the same number of times as the number of lines in file `groups.txt`. Each run produces a `bicN.RData` file containing BIC computations. These computations are aggregated in Phase 3 to create `Phase3.RData`, which contains the `saved.scores` used for `mcmc.qtlnet` runs in Phase 4, which each produce an `mcmcN.RData` file. The number of runs of Phase 4 is an argument `nruns` stored in the `params.txt` file processed in Phase 1. Finally, Phase 5 aggregates the MCMC results from multiple independent runs into one `qtlnet` object.

Author(s)

Brian S. Yandell and Elias Chaibub Neto

References

Chaibub Neto E, Keller MP, Attie AD, Yandell BS (2010) Causal Graphical Models in Systems Genetics: a unified framework for joint inference of causal network and genetic architecture for correlated phenotypes. Ann Appl Statist 4: 320-339. http://www.stat.wisc.edu/~yandell/doc/2010/92.AnnApplStat.pdf

http://www.stat.wisc.edu/~yandell/sysgen/qtlnet

See Also

mcmc.qtlnet, bic.qtlnet

Examples

```
## Not run:
  parallel.qtlnet("/u/y/a/yandell/public/html/sysgen/qtlnet/condor", 1)

## End(Not run)
```

---

parents.qtlnet *Determine and group node-parent combinations.*

Description

Routines useful for examining the size of node-parent combinations.

Usage

```
parents.qtlnet(pheno.col, max.parents = 3, codes.only = FALSE)
## S3 method for class 'parents.qtlnet'
summary(object, ...)
size.qtlnet(pheno.col, max.parents = 3)
group.qtlnet(pheno.col, max.parents = 3, n.groups = NULL,
  group.size = 50000, parents = parents.qtlnet(pheno.col, max.parents))
```

Arguments

pheno.col	Phenotype identifiers from cross object. May be numeric, logical or character.
max.parents	Maximum number of parents per node. This reduces the complexity of graphs and shortens run time. Probably best to consider values of 3-5.
parents	List containing all possible parents up to max.parents in size. May be a subset
codes.only	Return only codes of parents if TRUE.
n.groups	Number of groups for parallel computation. Determined from group.size if missing.
group.size	Size of groups for parallel computation. See details.
object	Object of class parent.qtlnet.
...	Additional arguments ignored.

Details

The most expensive part of calculations is running scanone on each phenotype with parent phenotypes as covariates. One strategy is to pre-compute the BIC contributions using a cluster and save them for later use. The parents.qtlnet routine creates a list of all possible parent sets (up to max.parents in size). The size.qtlnet determines the number of scanone calculations possible for a network with nodes pheno.col and maximum parent size max.parents. The group.qtlnet groups the parent sets into roughly equal size groups for parallel computations. See bic.qtlnet for further details.

Value

The size.qtlnet returns the number of possible scanone computations needed for BIC scores.

The group.qtlnet produces and index into the parents list created by parents.qtlnet. See details.

The parents.qtlnet creates a list object with names being the code.

The summary method for such an object is a data frame with row.names being the code, a binary code as decimal for the parents of a phenotype node, excluding the phenotype. Value is between 0 (no parents) and 2 ^ (length(pheno.col) - 1). The columns are

parents	Comma-separated string of parents to potential child node.
n.child	Number of possible child nodes to this parent set.

Author(s)

Brian S. Yandell and Elias Chaibub Neto

References

Chaibub Neto E, Keller MP, Attie AD, Yandell BS (2010) Causal Graphical Models in Systems Genetics: a unified framework for joint inference of causal network and genetic architecture for correlated phenotypes. Ann Appl Statist 4: 320-339. http://dx.doi.org/10.1214/09-AOAS288

See Also

bic.qtlnet

Examples

```
## Restrict to at most 3 parents per node.
pheno.col <- 1:6
max.parents <- 3
size.qtlnet(pheno.col, max.parents)
parents <- parents.qtlnet(pheno.col, max.parents)
summary(parents)

## Allow an arbitrary number (up to 12) of parents per node.
pheno.col <- 1:13
max.parents <- 12
size.qtlnet(pheno.col, max.parents)
```

```
## Make ~53 groups of ~1000, for a total of 53248 scanone runs.
parents <- parents.qtlnet(pheno.col, max.parents)
n.child <- summary(parents)$n.child
table(n.child)
groups <- group.qtlnet(parents = parents, group.size = 1000)
apply(groups, 1,
      function(group, parents) sapply(parents[seq(group[1], group[2])], length),
      parents)
```

Pscdbp *Cross and qtlnet objects with Ghazalpour et al. (2006) data. Only 13 phenotypes are included.*

Description

The R/qtl `cross` object was created from data at source. The `qtlnet` object was created using `mcmc.qtlnet`.

Usage

```
data(Pscdbp)
data(Pscdbp.qtlnet)
```

Source

https://horvath.genetics.ucla.edu/coexpressionnetwork/

References

Ghazalpour A, Doss S, Zhang B, Wang S, Plaisier C, Castellanos R, Brozell A, Schadt EE, Drake TA, Lusis AJ, Horvath S (2006) Integrating genetic and network analysis to characterize genes related to mouse weight. PLoS Genetics 2: e130-NA. http://dx.doi.org/10.1371/journal.pgen.0020130

Chaibub Neto E, Keller MP, Attie AD, Yandell BS (2010) Causal Graphical Models in Systems Genetics: a unified framework for joint inference of causal network and genetic architecture for correlated phenotypes. Ann Appl Statist 4: 320-339. http://dx.doi.org/10.1214/09-AOAS288

See Also

`read.cross`, `mcmc.qtlnet`

Examples

```
summary(Pscdbp)
## Not run:
summary(Pscdbp.qtlnet)

## End(Not run)
```

qdg *Produces a directed graph using QDG algorithm*

Description

This function implements the QDG algorithm described in Chaibub Neto et al 2008. It creates and scores QDGs. The computed scores (log-likelihood and BIC) are only valid for acyclic graphs. For cyclic networks qdgSEM should be used to compute the scores.

Usage

```
qdg(cross, phenotype.names, marker.names, QTL, alpha,
    n.qdg.random.starts, addcov = NULL, intcov = NULL,
    skel.method = c("pcskel","udgskel"), udg.order = 2)
graph.qdg(x, ...)
## S3 method for class 'qdg'
print(x, ...)
## S3 method for class 'qdg'
summary(object, ...)
```

Arguments

cross	object of class cross (see read.cross).
phenotype.names	character string with names of phenotype nodes corresponding to phenotypes in cross.
marker.names	list of character strings, one for each of phenotype.names. Each character string has the marker names for that phenotype.
QTL	object of class qtl (see makeqtl).
alpha	significance level threshold for PC or UDG algorithms (for the inference of the graph skeleton. See step 1 of the QDG algorithm). Must be between 0 and 1.
n.qdg.random.starts	number of random starts for the QDG algorithm (see step 3 of the QDG algorithm).
addcov	names of additive covariates. Must be valid phenotype names in cross. Expanded to include all intcov names.
intcov	names of additive covariates. Must be valid phenotype names in cross.
skel.method	Either "pcskel" for the PC skeleton algorithm (skeleton) or "udgskel" for the UDG algorithm (approximate.UDG routine defined internal to qdg).
udg.order	maximum allowed order of the UDG algorithm. Must be between zero and the number of variables minus 2.
x, object	object of class qdg.
...	additional arguments (ignored).

Details

The log-likelihood and BIC scores are computed based in the factorization of the joint distribution, and hence are only valid for acyclic networks. For cyclic networks these scores are relative to the unnormalized likelihoods. Models include phenotypes and QTLs. The 'udgskel' method for the computation of the skeleton of the causal model should be used for small networks only (the UDG algorithm quickly becomes computationally infeasible as the number of nodes increases).

Value

List object that inherits class "qdg" and "qdg" with components:

`UDG`	Undirected dependency graph from PC skeleton or UDG algorithms.
`DG`	Directed dependency graph before recheck step (output of the step 2 of the QDG algorithm).
`best.lm`	Solution with lowest BIC (best fit to the data).
`Solutions`	Solutions of dependency graph after recheck step (output of steps 3, 4 and 5 of the QDG algorithm.)
`marker.names`	List of character strings, one for each of `phenotype.names`. Each character string has the marker names for that phenotype.
`phenotype.names`	Character string with names of phenotype nodes corresponding to phenotypes in `cross`.

References

Chaibub Neto et al. (2008) Inferring causal phenotype networks from segregating populations. Genetics 179: 1089-1100.

See Also

`skeleton`

Examples

```
## 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)
```

```
## sample markers (2 per phenotype)
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(n.ind)
y4 <- runif(1,0.5,1) * y3 + runif(3,0.5,1)[g41] + runif(3,0.5,1)[g42] +
      rnorm(n.ind)

## incorporate phenotypes to 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
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
## Not run:
gr <- graph.qdg(out)
```

```
gr
plot(gr)

## End(Not run)
```

---

qdg.perm.test *Conduct permutation test for LOD score of edge direction on directed graph*

Description

Conduct permutation test for LOD score of edge direction on directed graph.

Usage

```
qdg.perm.test(cross, nperm, node1, node2, common.cov = NULL,
  DG, QTLs, addcov = NULL, intcov = NULL)
## S3 method for class 'qdg.perm.test'
summary(object, ...)
## S3 method for class 'qdg.perm.test'
print(x, ...)
```

Arguments

cross	Object of class cross (see read.cross).
nperm	Number of permutations.
node1	Character string with name of a phenotype nodes.
node2	Character string with name of a phenotype nodes.
common.cov	Character string with name of common phenotype covariates.
DG	Directed graph of class QDG
QTLs	List of objects of class qtl.
addcov	Names of additive covariates. Must be valid phenotype names in cross. Expanded to include all intcov names.
intcov	Names of additive covariates. Must be valid phenotype names in cross.
x, object	Object of class qdg.perm.test.
...	Additional arguments ignored.

Details

qdg.perm.test performs nperm permutation-based test of LOD score for an edge of a directed graph.

Value

List composed by:

pvalue	Permutation p-value.
obs.lod	Observed LOD score.
PermSample	Permutation LOD scores sample.
node1	Character string with name of a phenotype nodes.
node2	Character string with name of a phenotype nodes.

References

Chaibub Neto et al. (2008) Inferring causal phenotype networks from segregating populations. Genetics 179: 1089-1100.

Examples

```
data(glxnet)
glxnet.cross <- calc.genoprob(glxnet.cross)
set.seed(1234)
glxnet.cross <- sim.geno(glxnet.cross)
## Should really use nperm = 1000 here.
qdg.perm.test(glxnet.cross, nperm = 10, "Glx", "Slc1a2",
   DG = glxnet.qdg$DG, QTLs = glxnet.qtl)
```

---

qdg.sem *Score directed graphs outputed by qdg using structural equation models (SEM)*

Description

Score directed graphs (cyclic or acyclic) outputed by qdg function using the sem R package.

Usage

```
qdg.sem(qdgObject, cross)
## S3 method for class 'qdg.sem'
print(x, ...)
## S3 method for class 'qdg.sem'
summary(object, ...)
```

Arguments

qdgObject	list containing the output of qdg.
cross	object of class cross (see read.cross).
x, object	object of class qdg.
...	extra arguments to print or summary (ignored).

Details

Fits a SEM to the phenotypes network. QTLs are not included as variables in the model. When additive covariates are used in qdg, qdg.sem fits a SEM model to the residuals of the variables after adjustment of the additive covariates.

Value

List object that inherits class "qdg.sem" and "qdg" composed by:

`best.SEM`	Solution with lowest SEM BIC (best fit to the data).
`BIC.SEM`	Vector with the BIC values of all solutions from qdg.
`path.coeffs`	Path coefficients associated with the best SEM solution.
`Solutions`	Solutions of dependency graph after recheck step (output of steps 3, 4 and 5 of the QDG algorithm.)
`marker.names`	List of character strings, one for each of `phenotype.names`. Each character string has the marker names for that phenotype.
`phenotype.names`	Character string with names of phenotype nodes corresponding to phenotypes in `cross`.
`dropped`	Indexes of solutions that were dropped (`NULL` if none dropped).

See Also

qdg sem

Examples

```
## 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)

## sample markers (2 per phenotype)
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(n.ind)
y4 <- runif(1,0.5,1) * y3 + runif(3,0.5,1)[g41] + runif(3,0.5,1)[g42] +
      rnorm(n.ind)

## incorporate phenotypes to 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
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")

## Not run:
gr <- graph.qdg(out)
plot(gr)

## Following does not work. Not sure why.
out2 <- qdg.sem(out, cross=mycross)
out2
gr2 <- graph.qdg(out2)
plot(gr2)
```

```
## End(Not run)
```

`subset.qtlnet` *Catenate or subset qtlnet object(s).*

Description

Multiple qtlnet objects can be catenated together or subsetted by run.

Usage

```
## S3 method for class 'qtlnet'
subset(x, run, ...)
## S3 method for class 'qtlnet'
c(...)
best.qtlnet(x, burnin = attr(x, "burnin"), wh = which.min(meanbic(x, burnin)))
```

Arguments

`x`	Object of class `qtlnet`. See `mcmc.qtlnet`.
`run`	Numeric index to desired run. Must be between 0 and number of runs.
`burnin`	Proportion of MCMC samples to be considered as burnin. Taken from `qtlnet` object usually.
`wh`	Number identifying which model is best.
`...`	For `c.qtlnet`, objects of class `qtlnet` to be joined. Ignored for `subset.qtlnet`.

Details

The catenation is used by `parallel.qtlnet` in phase 5 to join together multiple independent MCMC runs. Note that the averaged network and the frequency of acceptance for a derived subset are only based on the saved samples, while the original `qtlnet` objects used all samples. Thus catenation and subset are not strictly reversible functions.

The `best.qtlnet` routine picks the run with the best (lowest) BIC score on average and returns that run as a qtlnet object. It also produces a trace plot of BIC for all the runs.

Value

Both return an object of class `qtlnet`.

Author(s)

Brian Yandell

See Also

`mcmc.qtlnet`

Examples

```
## Not run:
joined <- c(qtlnet1, qtlnet2)
sub1 <- subset(joined, 1)
best <- best.qtlnet(joined)
## qtlnet1 and sub1 should be nearly identical.

## End(Not run)
```

`summary.qtlnet` *summary of model average network and posterior table*

Description

Print and summary methods for qtlnet objects.

Usage

```
## S3 method for class 'qtlnet'
print(x, cutoff = 0.01, digits = 3, ...)
## S3 method for class 'qtlnet'
summary(object, parent.patterns = FALSE, ...)
## S3 method for class 'summary.qtlnet'
print(x, ...)
check.qtlnet(object, min.prob = 0.9, correct = TRUE, verbose = FALSE, ...)
```

Arguments

`x, object`	Object of class `qtlnet`.
`cutoff`	Frequency cutoff for model patterns to be displayed. Always shows at least the most common pattern.
`digits`	Number of digits to display for posterior probabilities on directed edges.
`parent.patterns`	Include summary of parent patterns if `TRUE`.
`min.prob`	Set the minimum posterior probability for inclusion of an edge.
`correct`	Correct `min.prob` if `TRUE`.
`verbose`	Print forbidden edges in model-averaged solution if `TRUE`.
`...`	Other hidden arguments. These include `min.prob`, which can also be passed to other `igraph.qtlnet` and `plot.qtlnet` routines.

Author(s)

Brian S. Yandell and Elias Chaibub Neto

References

Chaibub Neto E, Keller MP, Attie AD, Yandell BS (2010) Causal Graphical Models in Systems Genetics: a unified framework for joint inference of causal network and genetic architecture for correlated phenotypes. Ann Appl Statist 4: 320-339. http://www.stat.wisc.edu/~yandell/doc/2010/92.AnnApplStat.pdf

See Also

mcmc.qtlnet

Examples

```
data(Pscdbp.qtlnet)
print(Pscdbp.qtlnet)
summary(Pscdbp.qtlnet)
```

---

write.qtlnet *write qtlnet as text file*

Description

Write resulting graph as text file

Usage

```
write.qtlnet(x, filename, edges, loci.list, include.qtl = TRUE,
    est.list, include.est = TRUE,
    digits = 3, col.names = TRUE, ...)
```

Arguments

x	Object of class qtlnet.
filename	Character string with name of text file (usually ends in .txt).
edges	Data frame with first two columns being cause and effect directed phenotype pairs. Typically determined as averaged.net element from call to summary.qtlnet.
loci.list	List of character names of loci by phenotype. Typically determined by call to loci.qtlnet.
include.qtl	Include QTL in graph if TRUE and loci.list is not NULL.
est.list	List of estimates from internal est.qtlnet?
include.est	Include estimate if TRUE.
digits	Number of significant digits for width.
col.names	Character vector of column names.
...	Additional arguments passed to called routines.

Details

Simple write of causal network, for instance to use with Cytoscape.

Value

Invisibly returns data frame that corresponds to saved file.

Author(s)

Brian S. Yandell and Elias Chaibub Neto

References

Chaibub Neto E, Keller MP, Attie AD, Yandell BS (2010) Causal Graphical Models in Systems Genetics: a unified framework for joint inference of causal network and genetic architecture for correlated phenotypes. Ann Appl Statist 4: 320-339. http://www.stat.wisc.edu/~yandell/doc/2010/92.AnnApplStat.pdf

See Also

igraph.qtlnet

Examples

```
## Not run:
write.qtlnet(Pscdbp.qtlnet, "Pscdbp.txt")

## End(Not run)
```

Index