A Continuous Bayesian Networks Model for Water Quality Modeling

Song Qian and Robert Miltner

December 12, 2014

Considerations in Developing Empirical Models

Instead of following the Occam's razor to make a model as simple as possible (but not simpler), our exploration followed the guidelines in Gelman and Hill (2007) and Qian (2010):

- Including predictors that are substantively relevant,
- Item Considering compound variables, i.e., creating a variable by combining several variables (e.g., length times width to form area),
- Consider log-transformation and other transformations to normality (based on Box-Cox),
- Including a statistically insignificant predictor if the estimated slope is of the correct sign,
- Considering interactions of predictors with strong effects.

Including a statistically insignificant predictor is often considered a bad practice. But in the context of this study, we find this rule to be important. For example, when developing the model for predicting benthic chlorophyll a concentration, TP is consistently a weak and insignificant predictor. The estimated TP slope is small but positive (the correct sign). Including TP in the model will not make much difference in predicting chlorophyll a concentration. But conceptually, TP is a nutrient and it should have a positive effect on benthic chlorophyll a. A potential reason of an insignificant TP slope is the use of a cross-sectional data, and the effect of TP varies from stream to stream. When updating our model using stream- or region-specific data, we may find that TP is an important factor in some streams or regions.

In this document, we report only the final model.

R Code

The R code used in this work are divided into three groups – functions for specific tasks, regression model fitting, and MCMC using JAGS.

R functions

The following functions were used in various places:

```
## functions
ini <- function(mat, val) {                          # Initializing matrices
  ina <- is.na(mat); mat[ina] <- val; mat[!ina] <- NA; mat;}

## load/install packages
packages<-function(x, repos="http://cran.r-project.org", ...){
  x<-as.character(match.call()[[2]])
  if (!require(x,character.only=TRUE)){
    install.packages(pkgs=x, repos=repos, ...)
```

```
    require(x,character.only=TRUE)
  }
}

hockey <- function(x,alpha1,beta1,beta2,brk,eps=diff(range(x))/100,
                   delta=T) {
  ## alpha1 is the intercept of the left line segment
  ## beta1 is the slope of the left line segment
  ## beta2 is the slope of the right line segment
  ## brk is location of the break point
  ## 2*eps is the length of the connecting quadratic piece

  ## reference: Bacon & Watts "Estimating the Transition Between
  ## Two Intersecting Straight Lines", Biometrika, 1971
  x <- x-brk
  if (delta) beta2 <- beta1+beta2
  x1 <- -eps
  x2 <- +eps
  b <- (x2*beta1-x1*beta2)/(x2-x1)
  cc <- (beta2-b)/(2*x2)
  a <- alpha1+beta1*x1-b*x1-cc*x1^2
  alpha2 <- - beta2*x2 +(a + b*x2 + cc*x2^2)

  lebrk <- (x <= -eps)
  gebrk <- (x >= eps)
  eqbrk <- (x > -eps & x < eps)

  result <- rep(0,length(x))
  result[lebrk] <- alpha1 + beta1*x[lebrk]
  result[eqbrk] <- a + b*x[eqbrk] + cc*x[eqbrk]^2
  result[gebrk] <- alpha2 + beta2*x[gebrk]
  result
}

sim.nls <- function (object, n.sims=100){
  # sim.nls:  get posterior simulations of sigma and beta from a nls object
  #
  # Arguments:
  #
  #     object:  the output of a call to "nls"
  #              with n data points and k predictors
  #     n.sims:  number of independent simulation draws to create
  #
  # Output is a list (sigma.sim, beta.sim):
  #
  #     sigma.sim:  vector of n.sims random draws of sigma
  #       (for glm's, this just returns a vector of 1's or else of the
  #       square root of the overdispersion parameter if that is in the model)
  #     beta.sim:  matrix (dimensions n.sims x k) of n.sims random draws of beta
  #

  object.class <- class(object)[[1]]
  if (object.class!="nls") stop("not a nls object")
```

```
  summ <- summary (object)
  coef <- summ$coef[,1:2,drop=FALSE]
  dimnames(coef)[[2]] <- c("coef.est","coef.sd")
  sigma.hat <- summ$sigma
  beta.hat <- coef[,1]
  V.beta <- summ$cov.unscaled
  n <- summ$df[1] + summ$df[2]
  k <- summ$df[1]
  sigma <- rep (NA, n.sims)
  beta <- array (NA, c(n.sims,k))
  dimnames(beta) <- list (NULL, names(beta.hat))
  for (s in 1:n.sims){
    sigma[s] <- sigma.hat*sqrt((n-k)/rchisq(1,n-k))
    beta[s,] <- mvrnorm (1, beta.hat, V.beta*sigma[s]^2)
  }
  return (list (beta=beta, sigma=sigma))
}
```

Packages and general parameters

```
packages(arm)
packages(rjags)
```

```
## Warning: package 'rjags' was built under R version 3.1.2
```

```
packages(lattice)
packages(dclone)
packages(rv)
setnsims(10000)
```

```
## [1] 4000
```

```
packages(tikzDevice)
packages(knitr)
```

```
## Warning: package 'knitr' was built under R version 3.1.2
```

```
packages(car)
options(width=72)
```

Setting some parameters:

```
nchains <- 3
niters <- 100000
nkeep <- 2500
set.seed(111)

base <- "~/copy/OhioBN"  ## change to base <- "c:/my dir"
```

```
  #base <- "c:/users/song/copy/OhioBN"
  setwd(base)
  dataDIR <- paste(base, "Data", sep="/")
  plotDIR <- paste(base, "manuscript", sep="/")
```

Data and data processing

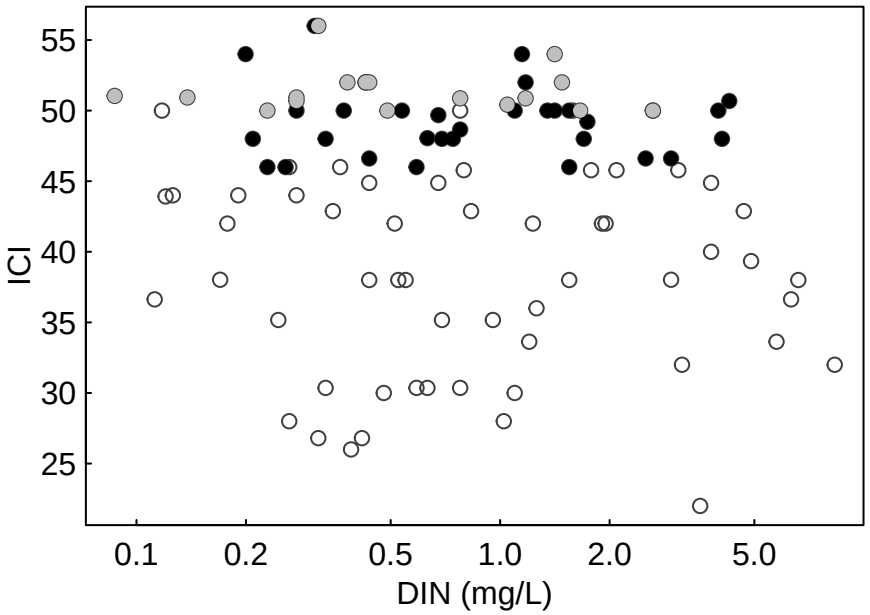
```
  OhioData <- read.csv(paste(dataDIR, "ohioepadata.csv", sep="/"), header=T)
  OhioData$basinsize <- "large"
  OhioData$basinsize [OhioData$LOGDRAIN <log10(1300)]<- "small"
  ## OhioData$DO.MAX <- OhioData$DO.MIN+OhioData$DO.RANGE
  OhioData$logitR<-log(OhioData$DO.MIN/OhioData$DO.RANGE)
  OhioData$DIN <- (10^OhioData$DIN) / 1000  ## data were log10(ug/L)
  OhioData$TP <- (10^OhioData$TP) / 1000
  OhioData$Chla <- 10^OhioData$LOG.P.CHLA
```

Exploratory Data Analysis (EDA)

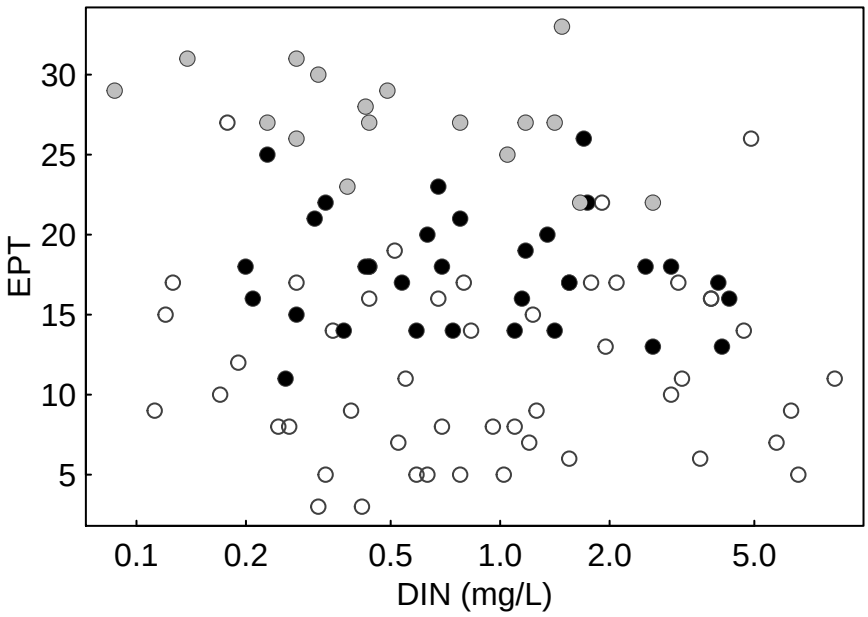
We explored multiple models for each link in the hypothesized relationship among the variables. We summarize a few plots to illustrate our process.

Initially, we explored the connection between ICI (EPT) and DIN.

```
## ICI against DIN
  subset1 <- OhioData$ICI>=50&OhioData$QUALEPT>21
  subset2 <- OhioData$ICI>=46&OhioData$QUALEPT>10
##  tikz(file="ICIvDIN.tex", width=4.5, height=3.35)
  par(mar=c(3,3,1,0.5), mgp=c(1.25,0.25,0), las=1, tck=0.01)
  plot(ICI~DIN, data=OhioData, xlab="DIN (mg/L)", ylab="ICI", log="x", col=grey(0.25))
  points(OhioData$DIN[subset2], OhioData$ICI[subset2], pch=16)
  points(OhioData$DIN[subset1], OhioData$ICI[subset1], col=grey(0.75), pch=16)
```



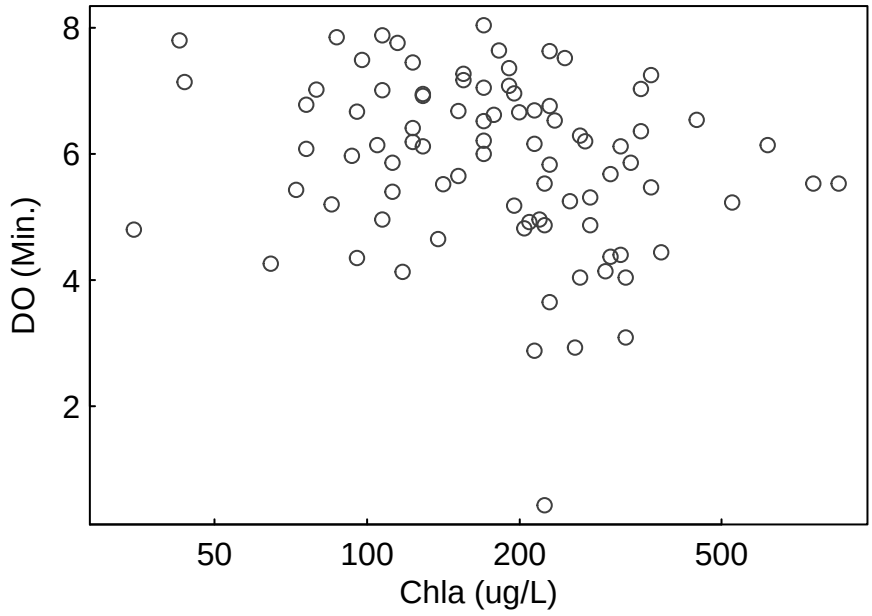
```
##  dev.off()

## EPT against DIN
  par(mar=c(3,3,1,0.5), mgp=c(1.25,0.25,0), las=1, tck=0.01)
  plot(QUALEPT~DIN, data=OhioData, xlab="DIN (mg/L)", ylab="EPT", log="x", col=grey(0.25))
  points(OhioData$DIN[subset2], OhioData$QUALEPT[subset2], pch=16)
  points(OhioData$DIN[subset1], OhioData$QUALEPT[subset1], col=grey(0.75), pch=16)
```



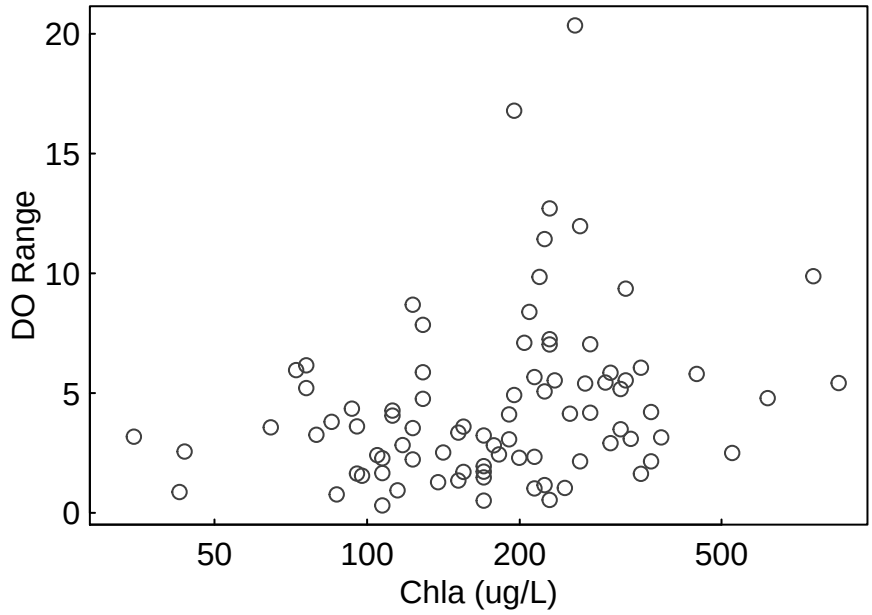
These plots suggest that directly link the response to nutrient concentrations will be unlikely to achieve the goal. We then selected to model chla first. We selected the best model forms based on both our understanding of the subject matter and plots of chla against various potential predictors.

We then explored how to model the DO – chla relation. We used the general principles of developing a predictive model in Gelman and Hill (2007) and Qian (2010). Based on plots of minimum DO and DO range against chla, we decided to explore compounding variables. Just as we measure of daily DO fluctuation, we explored the logarithmic of the ratio of DO minimum over DO range.

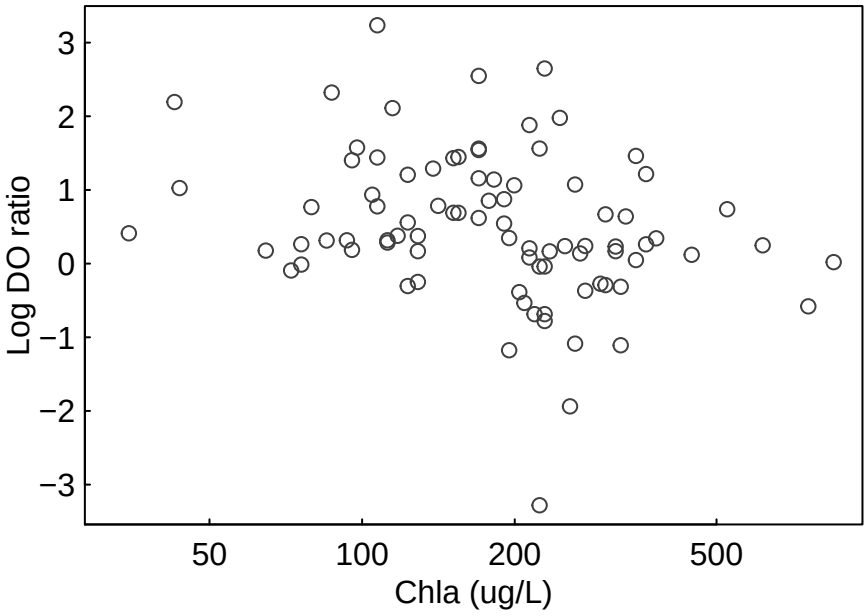
```
## DO
  par(mar=c(3,3,1,0.5), mgp=c(1.25,0.25,0), las=1, tck=0.01)
  plot(DO.MIN~Chla, data=OhioData, xlab="Chla (ug/L)", ylab="DO (Min.)", log="x", col=grey(0.25))
```




```
  par(mar=c(3,3,1,0.5), mgp=c(1.25,0.25,0), las=1, tck=0.01)
plot(DO.RANGE ~ Chla, data=OhioData, xlab="Chla (ug/L)", ylab="DO Range", log="x", col=grey(0.25))
```




```
  par(mar=c(3,3,1,0.5), mgp=c(1.25,0.25,0), las=1, tck=0.01)
plot(logitR ~ Chla, data=OhioData, xlab="Chla (ug/L)", ylab="Log DO ratio", log="x", col=grey(0.25))
```



It seem that the effect of chla can be better described by the difference of log DO minumum and log DO maximum, or the range of log DO. A small DO ratio suggesting a large fluctuation, while a large ratio suggesting a small fluctuation.

Linear and nonlinear regression models

In Miltner (2010), four regression models were used. Through exhaustive search, these four groups of models were refit and their connections exploited to form the continuous Bayesian networks model. The first model links nutrient input to benthic algal concentration measured by chlorophyll a:

```
chla.m1 <- lm(log(Chla)~log(DIN)+log(TP)+car::logit(AG.PCT)+LOGARC,
              data=OhioData, subset=RIVCODE!="02-500")
summary(chla.m1)
```

```
##
## Call:
## lm(formula = log(Chla) ~ log(DIN) + log(TP) + car::logit(AG.PCT) +
##     LOGARC, data = OhioData, subset = RIVCODE != "02-500")
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.93754 -0.21110  0.04733  0.26547  1.32319
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)         3.73115    0.37606   9.922  < 2e-16 ***
## log(DIN)            0.21170    0.06549   3.233  0.00166 **
## log(TP)             0.02188    0.05141   0.426  0.67134
## car::logit(AG.PCT)  0.08936    0.03726   2.399  0.01831 *
## LOGARC              0.84395    0.19476   4.333  3.5e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 0.5232 on 100 degrees of freedom
## Multiple R-squared:  0.3772, Adjusted R-squared:  0.3523
## F-statistic: 15.14 on 4 and 100 DF,  p-value: 1.037e-09
```

```
  sims.m1 <- sim(chla.m1, getnsims())
  betas.m1 <- rvsims(sims.m1@coef)
  sigma.m1 <- rvsims(sims.m1@sigma)
```

All models were fit with data with the river (02-500).

We have two DO related variables – minimum DO and DO range. We tried these two and several derived variables and find that the log ratio of minimum DO over the range of DO (DO ratio) is probably the best form and a hockey stick model was used to model the relationship between log DO ratio and log benthic chlorophyll a:

```
  DO.m2 <- nls(logitR~hockey(log(Chla), beta0, beta1=0,
                             delta, phi),
               start=list(beta0=1,
               delta=-2., phi=2*log(10)), data=OhioData,
               na.action=na.omit, control=list(maxiter = 500),
               subset=OhioData$RIVCODE!="02-500")
  summary(DO.m2)
```

```
##
## Formula: logitR ~ hockey(log(Chla), beta0, beta1 = 0, delta, phi)
##
## Parameters:
##       Estimate Std. Error t value Pr(>|t|)
## beta0   0.8776     0.2610   3.362  0.00118 **
## delta  -0.5884     0.2590  -2.272  0.02573 *
## phi     4.6611     0.5970   7.807  1.8e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9643 on 81 degrees of freedom
##
## Number of iterations to convergence: 45
## Achieved convergence tolerance: 7.036e-06
##   (21 observations deleted due to missingness)
```

```
  sims.m2 <- sim.nls(DO.m2, getnsims())
```

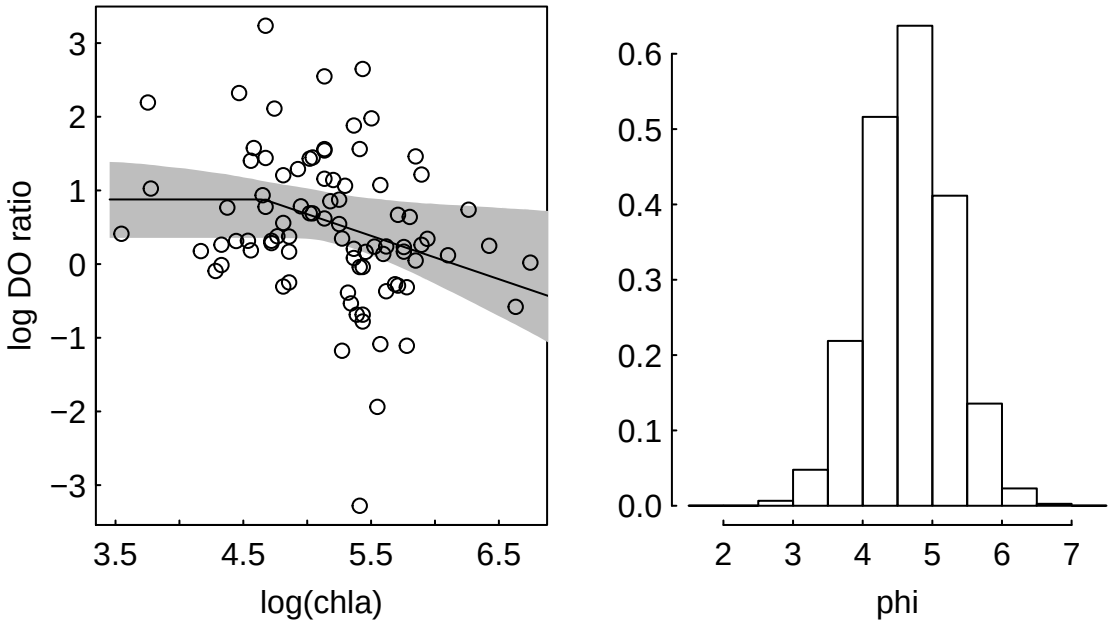
Details of the hockestick model is in Qian (2010). The fitted model is plotted (with uncertainty).

```
  betas <- sims.m2$beta
  ind <- function(x)ifelse(x>=0, 1, 0)
  x <- log(10)*seq(1.5,3,,100)
  y <- matrix(0, ncol=100, nrow=getnsims())
  for (i in 1:100) y[,i] <- betas[,1]+betas[, 2]*ind(x[i]-betas[,3])*(x[i]-betas[,3])
  y.int <- apply(y, 2, quantile, prob=c(0.025,0.975))
  tikz(file="DOhockeystk.tex", width=6, height=3.5, standAlone=F)
```

```
  par(mfrow=c(1,2), mar=c(3,3,1,0.25), mgp=c(1.5,0.25,0), tck= 0.01, las=1)
  plot(c(0,1),c(0,1),
       xlim=log(range(OhioData$Chla, na.rm=T)),
       ylim=range(OhioData$logitR, na.rm=T),
       xlab="$\\log(chla)$",type="n",
       ylab="$\\log\\left (\\frac{DO_{min}}{DO_{range}}\\right )$")
  polygon(x=c(x, rev(x)),y=c(y.int[1,], rev(y.int[2,])), density=-1,col="gray",
          border=F)
  lines(log(10)*seq(1.5, 3,,100),
        hockey(log(10)*seq(1.5,3,,100), coef(DO.m2)[1], 0,
               coef(DO.m2)[2], coef(DO.m2)[3]))
  points(log(OhioData$Chla), OhioData$logitR)
  hist(betas[,3], xlab="$\\phi$", main="", ylab="", prob=T)
  dev.off()
```

```
## pdf
##   2
```

```
  par(mfrow=c(1,2), mar=c(3,3,1,0.25), mgp=c(1.5,0.25,0), tck= 0.01, las=1)
  plot(c(0,1),c(0,1),
       xlim=log(range(OhioData$Chla, na.rm=T)),
       ylim=range(OhioData$logitR, na.rm=T),
       xlab="log(chla)",type="n",
       ylab="log DO ratio")
  polygon(x=c(x, rev(x)),y=c(y.int[1,], rev(y.int[2,])), density=-1,col="gray",
          border=F)
  lines(log(10)*seq(1.5, 3,,100),
        hockey(log(10)*seq(1.5,3,,100), coef(DO.m2)[1], 0,
               coef(DO.m2)[2], coef(DO.m2)[3]))
  points(log(OhioData$Chla), OhioData$logitR)
  hist(betas[,3], xlab="phi", main="", ylab="", prob=T)
```



The use of a hockeystick model is hardly justifiable based on the plot. A simpler log-log linear model is as effective. Because a linear model is a special case of the hockeystick model, using a hockeystick model will not exclude the linear model. Using the hockeystick model will provide some flexibility, especially when updating the model using stream- or region-specific data. As a result, we chose the more complicated model over the equally well fit simple model, an apparent violation of the principle of Occam's razor (a model should be as simple as possible but not simpler).

Ohio uses ICI and EPT taxa richness as measures of biological condition.

```
ICI.m3 <- lm(ICI~log(Chla)+logitR+QHEI, data=OhioData,
             subset=RIVCODE!="02-500")
summary(ICI.m3)
```

```
##
## Call:
## lm(formula = ICI ~ log(Chla) + logitR + QHEI, data = OhioData,
##     subset = RIVCODE != "02-500")
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.8180  -3.9437   0.9501   4.3498  12.0479
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  37.1187     7.3689   5.037 2.92e-06 ***
## log(Chla)    -1.6673     1.1727  -1.422 0.159048
## logitR        2.8349     0.7144   3.968 0.000158 ***
## QHEI          0.2016     0.0484   4.166 7.86e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.018 on 79 degrees of freedom
##   (22 observations deleted due to missingness)
## Multiple R-squared:  0.4172, Adjusted R-squared:  0.3951
## F-statistic: 18.85 on 3 and 79 DF,  p-value: 2.573e-09
```

```
sims.m3 <- sim(ICI.m3, 2502)
betas.m3 <- rvsims(sims.m3@coef)
sigma.m3 <- rvsims(sims.m3@sigma)

EPT.m3 <- lm(log(QUALEPT)~log(Chla)+logitR+QHEI, data=OhioData,
             subset=RIVCODE!="02-500")
summary(EPT.m3)
```

```
##
## Call:
## lm(formula = log(QUALEPT) ~ log(Chla) + logitR + QHEI, data = OhioData,
##     subset = RIVCODE != "02-500")
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.00017 -0.23544 -0.00298  0.27021  0.82481
##
```

```
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.310605   0.511270   4.519 2.34e-05 ***
## log(Chla)   -0.124033   0.081395  -1.524 0.131868
## logitR       0.129709   0.048827   2.657 0.009691 **
## QHEI         0.013546   0.003306   4.097 0.000107 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4056 on 73 degrees of freedom
##   (28 observations deleted due to missingness)
## Multiple R-squared:  0.3674, Adjusted R-squared:  0.3414
## F-statistic: 14.13 on 3 and 73 DF,  p-value: 2.351e-07
```

```
  sims.m4 <- sim(EPT.m3, 2502)
  betas.m4 <- rvsims(sims.m4@coef)
  sigma.m4 <- rvsims(sims.m4@sigma)
```

Again, benthic chlorophyll a is not a statistically significant predictor, but the slopes of chlorophyll a in both models are of the correct sign.

The cBN Model

The cBN model combines empirical models developed in previous section and refit the model using flat prior distributions.

The JAGS model

The JAGS model is written in a text file:

```
cat("
  ## x1: logDIN, x2: logTP, x3: logit(AG),
  ## x4: log(ARC), x5: QHEI

  model{
    sigma.C <- sigma.hatC*sqrt(dfC/chisqC)
    chisqC ~ dchisqr(dfC)

    sigma.D <- sigma.hatD*sqrt(dfD/chisqD)
    chisqD ~ dchisqr(dfD)

    sigma.E <- sigma.hatE*sqrt(dfE/chisqE)
    chisqE ~ dchisqr(dfE)

    sigma.I <- sigma.hatI*sqrt(dfI/chisqI)
    chisqI ~ dchisqr(dfI)

    for (i in 1:kC){
      for (j in 1:kC){
        SigmaC[i,j] <- pow(sigma.C, 2) * V.C[i,j]
      }
```

```
 }
 Oc <- inverse(SigmaC[,])
 TauC[1:kC,1:kC] <- 0.5*(Oc+t(Oc))
 betaC[1:kC]~dmnorm(beta.hatC[],TauC[,])
 tau.C <- pow(sigma.C, -2)

for (i in 1:kD){
   for (j in 1:kD){
     SigmaD[i,j] <- pow(sigma.D, 2) * V.D[i,j]
   }
 }
 Od <- inverse(SigmaD[,])
 TauD[1:kD,1:kD] <- 0.5*(Od+t(Od))
 betaD[1:kD]~dmnorm(beta.hatD[],TauD[,])
 tau.D <- pow(sigma.D, -2)

 for (i in 1:kE){
   for (j in 1:kE){
     SigmaE[i,j] <- pow(sigma.E, 2) * V.E[i,j]
   }
 }
 Oe <- inverse(SigmaE[,])
 TauE[1:kE,1:kE] <- 0.5*(Oe+t(Oe))
 betaE[1:kE]~dmnorm(beta.hatE[],TauE[,])
 tau.E <- pow(sigma.E, -2)

for (i in 1:kI){
   for (j in 1:kI){
     SigmaI[i,j] <- pow(sigma.I, 2) * V.I[i,j]
   }
 }
 Oi <- inverse(SigmaI[,])
 TauI[1:kI,1:kI] <- 0.5*(Oi+t(Oi))
 betaI[1:kI]~dmnorm(beta.hatI[],TauI[,])
 tau.I <- pow(sigma.I, -2)

 for (j in 1:J) {
     logchla[j] ~ dnorm(mu.C[j], tau.C)
     mu.C[j] <- betaC[1] + betaC[2]*X1[j] +
                betaC[3]*X2[j] + betaC[4]*X3[j] +
                betaC[5]*X4[j]

     DO.logitR[j] ~ dnorm(mu.D[j], tau.D)
     mu.D[j] <- betaD[1] +
      betaD[2]*step(mu.C[j]-betaD[3])*(mu.C[j]-betaD[3])

     logEPT[j] ~ dnorm(mu.E[j], tau.E)
     mu.E[j] <- betaE[1] + betaE[2]*mu.C[j] +
                betaE[3]*mu.D[j] + betaE[4]* X5[j]

     ICI[j] ~ dnorm(mu.I[j], tau.I)
     mu.I[j] <- betaI[1] + betaI[2]*mu.C[j] +
                betaI[3]*mu.D[j] + betaI[4]* X5[j]
```

```
  }
}

", file="OhioBN1.txt")
```

Inputs to JAGS

The R code include an R function for organizing input data, generating initial values, and listing model coefficients for updating:

```
bugs.inAll2 <- function(infile=OhioData, nchains=3, subset=NULL, flat=F){
    if (!is.null(subset)) infile <- infile[subset,]
    summ <- summary(chla.m1)
    coefC <- summ$coef[,1:2,drop=FALSE]
    dimnames(coefC)[[2]] <- c("coef.est","coef.sd")
    sigma.hatC <- summ$sigma
    beta.hatC <- coefC[,1]
    V.betaC <- summ$cov.unscaled
    nC <- summ$df[1] + summ$df[2]
    kC <- summ$df[1]

    summ <- summary(DO.m2)
    coefD <- summ$coef[,1:2,drop=FALSE]
    dimnames(coefD)[[2]] <- c("coef.est","coef.sd")
    sigma.hatD <- summ$sigma
    beta.hatD <- coefD[,1]
    V.betaD <- summ$cov.unscaled
    nD <- summ$df[1] + summ$df[2]
    kD <- summ$df[1]

    summ <- summary(ICI.m3)
    coefI <- summ$coef[,1:2,drop=FALSE]
    dimnames(coefI)[[2]] <- c("coef.est","coef.sd")
    sigma.hatI <- summ$sigma
    beta.hatI <- coefI[,1]
    V.betaI <- summ$cov.unscaled
    nI <- summ$df[1] + summ$df[2]
    kI <- summ$df[1]

    summ <- summary(EPT.m3)
    coefE <- summ$coef[,1:2,drop=FALSE]
    dimnames(coefE)[[2]] <- c("coef.est","coef.sd")
    sigma.hatE <- summ$sigma
    beta.hatE <- coefE[,1]
    V.betaE <- summ$cov.unscaled
    nE <- summ$df[1] + summ$df[2]
    kE <- summ$df[1]
    inits <- list()
    J <- dim(infile)[1]
    if (flat){
        dfC <- dfD <- dfI <- dfE <- 4
    } else {
```

```
        dfC <- nC-kC
        dfD <- nD-kD
        dfI <- nI-kI
        dfE <- nE-kE
    }
    bugs.dat <- list(J=J, kC=kC, kD=kD, kI=kI,kE=kE,
                     dfC=dfC, dfD=dfD, dfE=dfE, dfI=dfI,
                     beta.hatC=beta.hatC,
                     sigma.hatC=sigma.hatC,
                     V.C=V.betaC,
                     beta.hatD=beta.hatD,
                     sigma.hatD=sigma.hatD,
                     V.D=V.betaD,
                     beta.hatI=beta.hatI,
                     sigma.hatI=sigma.hatI,
                     V.I=V.betaI,
                     beta.hatE=beta.hatE,
                     sigma.hatE=sigma.hatE,V.E=V.betaE,
                     X1=log(infile$DIN),X2=log(infile$TP),
                     X3=car::logit(infile$AG.PCT),
                     X4=infile$LOGARC, X5=infile$QHEI,
                     ICI=infile$ICI, DO.logitR=infile$logitR,
                     logchla=log(infile$Chla),logEPT=log(infile$QUALEPT))
    for (k in 1:nchains) inits[[k]] <- list(ICI=ini(infile$ICI, runif(1,20,60)),
                                            logchla=ini(log(infile$Chla),
                                                runif(1, 1.5,3)),
                                            logEPT=ini(log(infile$QUALEPT),
                                                log(rpois(1,5)+1)),
                                            DO.logitR=ini(infile$logitR, rnorm(1)),
                                            chisqC=rchisq(1, df=nC-kC),
                                            betaC= mvrnorm(1, beta.hatC,
                                                V.betaC*sigma.hatC^2),
                                            chisqD=rchisq(1, df=nD-kD),
                                            betaD= mvrnorm(1, beta.hatD,
                                                V.betaD*sigma.hatD^2),
                                            chisqE=rchisq(1, df=nE-kE),
                betaE= mvrnorm(1, beta.hatE,V.betaE*sigma.hatE^2),
                                            chisqI=rchisq(1, df=nI-kI),
                                            betaI= mvrnorm(1, beta.hatI,
                                                V.betaI*sigma.hatI^2))
    para <- c("betaC", "betaD","betaE", "betaI","sigma.C",
              "sigma.D","sigma.I","sigma.E#")
    return(list(data=bugs.dat, inits=inits, para=para))
}
```

Fitting the cBN model

When fitting using the entire data set, we used a flat prior (setting model residual variance to have an inverse χ^2 distribution with $df = 4$).

```
input.to.bugs <- bugs.inAll2(flat=T)
################ JAGS ##############
#m <- jags.model("OhioBN1.txt",
```

```
#                 input.to.bugs$data,
#                 input.to.bugs$inits,
#                 n.chain=nchains, n.adapt=5000)

#update(m, niters*5)
#x <- coda.samples(m, input.to.bugs$para,
#                   n.iter=niters,
#               thin=round(niters*nchains/nkeep))

#simCodaAll <- NULL
#for (i in 1:nchains)
#    simCodaAll <- rbind(simCodaAll, x[[i]])
#print(summary(simCodaAll))
```

When updating the model for the set-aside river, we use the posterior from the full model as the prior:

```
  ## Updating for River 02-500
#input.to.bugs <-
#    bugs.inAll2(subset=OhioData$RIVCODE=="02-500",
#                flat=F)

############### JAGS #############
#m <- jags.model("OhioBN1.txt",
#                input.to.bugs$data,
#                input.to.bugs$inits,
#                n.chain=nchains, n.adapt=5000)

#update(m, niters*5)
#x2 <- coda.samples(m, input.to.bugs$para,
#                   n.iter=niters,
#                   thin=round(niters*nchains/nkeep))

#simCodaAll1 <- NULL
#for (i in 1:nchains)
#    simCodaAll1 <- rbind(simCodaAll1, x2[[i]])
#print(summary(simCodaAll1))
```

The JAGS model output are saved (to avoid repeatedly running the model):

```
##save(simCodaAll, simCodaAll1, file=paste(base, "bugsOut.RData", sep="/"))
## can be later loaded
 load("bugsOut.RData")
```

Processing MCMC output

Monte Carlo simulations are done using random variable objects (R package "rv"). First, random variates of input variables are drawn:

```
## random samples of drivers (using uniform, except TP DIN)
sims.np<-rvnorm(1, mean=apply(cbind(log(OhioData$DIN),log(OhioData$TP)), 2, mean),
                var= var(cbind(log(OhioData$DIN),log(OhioData$TP))))
```

```
  sims.ag <- rvunif(1, car::logit(0.075),car::logit(0.99))
  sims.arc <- rvunif(1, 0.95,2.5)
  sims.qhei <- rvunif(1, 23, 91)
```

Then, random variates of model coefficients are taken from the existing MCMC output.

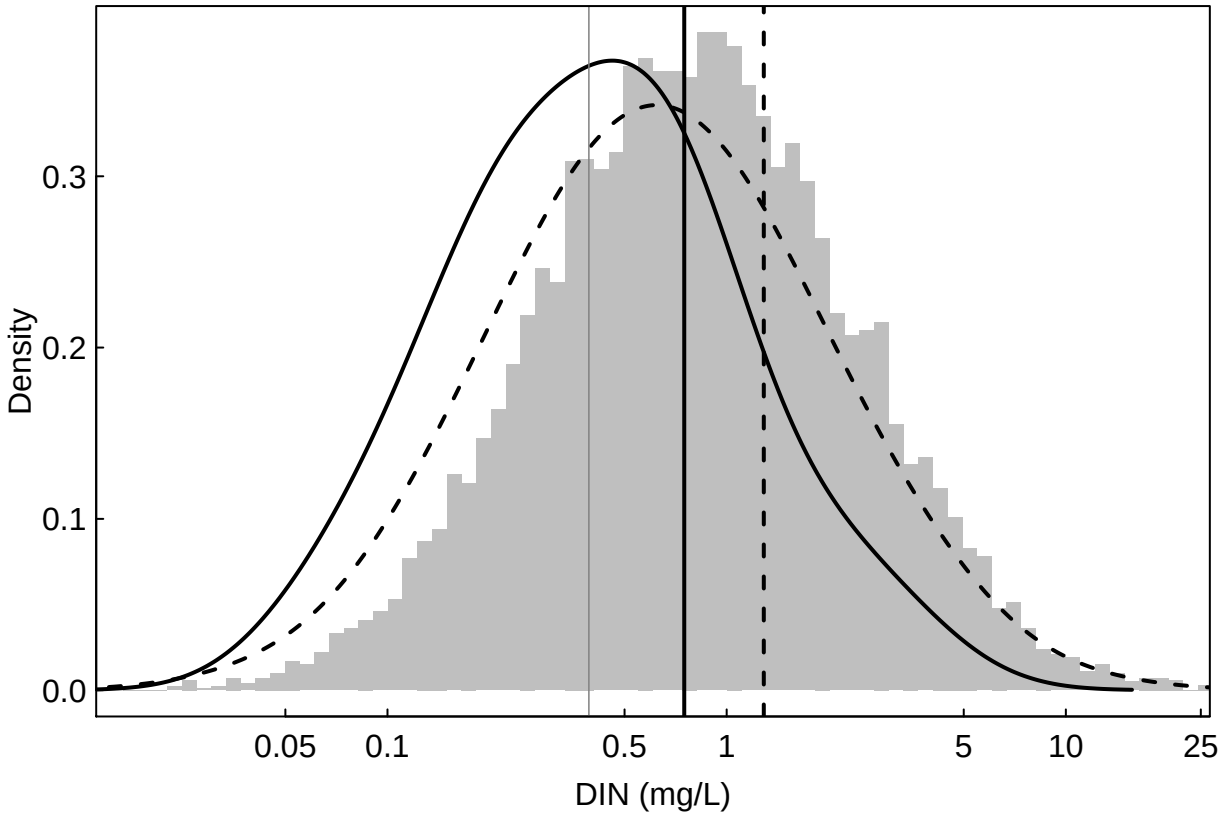
```
## 1. draw model coefficients
  simCodarv <- rvsims(simCodaAll)
  names(simCodarv) <- labels(simCodaAll)[[2]]

  simCodarv1 <- rvsims(simCodaAll1)
  names(simCodarv1) <- labels(simCodaAll1)[[2]]

  ## 2. MC of chla
  ## prior model
  betas.mcmc1 <-
    simCodarv[substring(names(simCodarv), 1, 5)=="betaC"]
  sigma.mcmc1 <- simCodarv[names(simCodarv)=="sigma.C"]
  mu.chlaAll <- betas.mcmc1[1]+betas.mcmc1[2]*sims.np[1]+
      betas.mcmc1[3]*sims.np[2]+ betas.mcmc1[4]*sims.ag+
      betas.mcmc1[5]*sims.arc

  ## updated for River 02-500
  betas.mcmc1S <-
    simCodarv1[substring(names(simCodarv1), 1, 5)=="betaC"]
  sigma.mcmc1S <- simCodarv1[names(simCodarv1)=="sigma.C"]
  mu.chlaAllS <- betas.mcmc1S[1]+betas.mcmc1S[2]*sims.np[1]+
      betas.mcmc1S[3]*sims.np[2]+ betas.mcmc1S[4]*sims.ag+
      betas.mcmc1S[5]*sims.arc

  ## 3. MC of DO
  betas.mcmc2 <-
     simCodarv[substring(names(simCodarv), 1, 5)=="betaD"]
  sigma.mcmc2 <- simCodarv[names(simCodarv)=="sigma.D"]
  mu.doAll <-
     betas.mcmc2[1]+betas.mcmc2[2]*(mu.chlaAll>betas.mcmc2[3])*
    (mu.chlaAll-betas.mcmc2[3])

  betas.mcmc2S <-
     simCodarv1[substring(names(simCodarv1), 1, 5)=="betaD"]
  sigma.mcmc2S <- simCodarv1[names(simCodarv1)=="sigma.D"]
  mu.doAllS <-
    betas.mcmc2[1]+betas.mcmc2[2]*(mu.chlaAllS>betas.mcmc2S[3])*
    (mu.chlaAllS-betas.mcmc2S[3])

  ## 4. MC of ICI & EPT
  betas.mcmc3 <-
     simCodarv[substring(names(simCodarv), 1, 5)=="betaI"]
  sigma.mcmc3 <- simCodarv[names(simCodarv)=="sigma.I"]
  betas.mcmc4 <-
     simCodarv[substring(names(simCodarv), 1, 5)=="betaE"]
  sigma.mcmc4 <- simCodarv[names(simCodarv)=="sigma.E"]
```

```
mu.iciAll <- betas.mcmc3[1] + betas.mcmc3[2]*mu.chlaAll+
    betas.mcmc3[3]*mu.doAll+betas.mcmc3[4]*sims.qhei
mu.eptAll <- betas.mcmc4[1] + betas.mcmc4[2]*mu.chlaAll+
    betas.mcmc4[3]*mu.doAll+betas.mcmc4[4]*sims.qhei

betas.mcmc3S <-
    simCodarv1[substring(names(simCodarv1), 1, 5)=="betaI"]
sigma.mcmc3S <- simCodarv1[names(simCodarv1)=="sigma.I"]
betas.mcmc4S <-
    simCodarv1[substring(names(simCodarv1), 1, 5)=="betaE"]
sigma.mcmc4S <- simCodarv1[names(simCodarv1)=="sigma.E"]

mu.iciAllS <- betas.mcmc3S[1] +
    betas.mcmc3S[2]*mu.chlaAllS+betas.mcmc3S[3]*mu.doAllS+
    betas.mcmc3S[4]*sims.qhei
mu.eptAllS <- betas.mcmc4S[1] +
    betas.mcmc4S[2]*mu.chlaAllS+betas.mcmc4S[3]*mu.doAllS+
    betas.mcmc4S[4]*sims.qhei

QQs <- quantile(OhioData$QHEI, prob=c(0.25,0.5,0.75,0.95))
## set QHEI at 25%-tile (65)
mu25.iciAll <- betas.mcmc3[1] + betas.mcmc3[2]*mu.chlaAll+
  betas.mcmc3[3]*mu.doAll+betas.mcmc3[4]*QQs[1]
mu25.eptAll <- betas.mcmc4[1] + betas.mcmc4[2]*mu.chlaAll+
  betas.mcmc4[3]*mu.doAll+betas.mcmc4[4]*QQs[1]

## 50%-tile
mu50.iciAll <- betas.mcmc3[1] + betas.mcmc3[2]*mu.chlaAll+
  betas.mcmc3[3]*mu.doAll+betas.mcmc3[4]*QQs[2]
mu50.eptAll <- betas.mcmc4[1] + betas.mcmc4[2]*mu.chlaAll+
  betas.mcmc4[3]*mu.doAll+betas.mcmc4[4]*QQs[2]

## 75%-tile
mu75.iciAll <- betas.mcmc3[1] + betas.mcmc3[2]*mu.chlaAll+
  betas.mcmc3[3]*mu.doAll+betas.mcmc3[4]*QQs[3]
mu75.eptAll <- betas.mcmc4[1] + betas.mcmc4[2]*mu.chlaAll+
  betas.mcmc4[3]*mu.doAll+betas.mcmc4[4]*QQs[3]

## 95%-tile
mu95.iciAll <- betas.mcmc3[1] + betas.mcmc3[2]*mu.chlaAll+
  betas.mcmc3[3]*mu.doAll+betas.mcmc3[4]*QQs[4]
mu95.eptAll <- betas.mcmc4[1] + betas.mcmc4[2]*mu.chlaAll+
  betas.mcmc4[3]*mu.doAll+betas.mcmc4[4]*QQs[4]

## 5. conditional DIN for ICI > 50 or 46 & EPT > 21 or 10
din.sims1 <-
  rvsims(sims(sims.np[1])[sims(mu.iciAll)>50&
                          sims(mu.eptAll)>log(21)])
din.sims2 <-
  rvsims(sims(sims.np[1])[sims(mu.iciAll)>46&
                          sims(mu.eptAll)>log(10)])

## 6. set QHEI at 50, 75, 95 %-tile
```

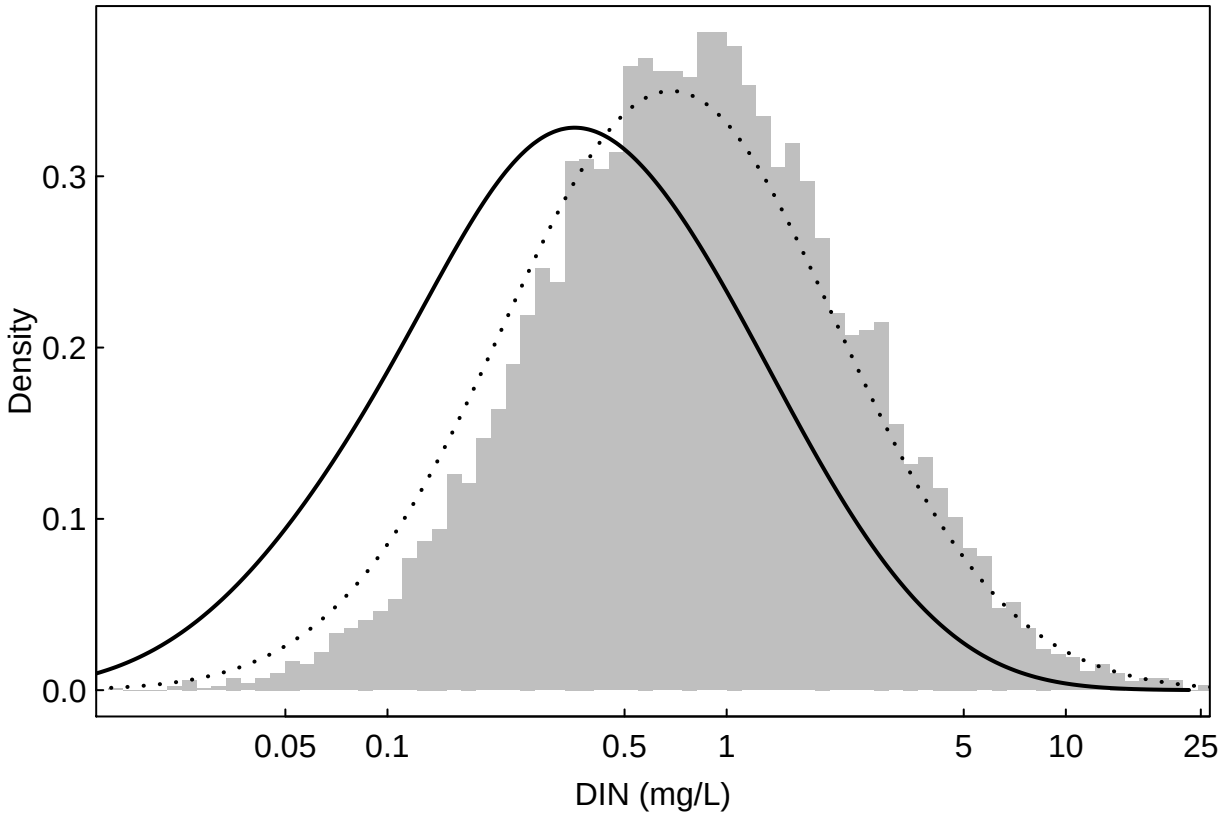
```
din.sims25 <-
  rvsims((sims(sims.np[1])[sims(mu25.iciAll)>46&
                           sims(mu25.eptAll)>log(10)]))
din.sims50 <-
  rvsims((sims(sims.np[1])[sims(mu50.iciAll)>46&
                           sims(mu50.eptAll)>log(10)]))
din.sims75 <-
  rvsims((sims(sims.np[1])[sims(mu75.iciAll)>46&
                           sims(mu75.eptAll)>log(10)]))
din.sims95 <-
  rvsims((sims(sims.np[1])[sims(mu95.iciAll)>46&
                           sims(mu95.eptAll)>log(10)]))
```

The results are plotted:

```
par(mgp=c(1.5,.25,0), mar=c(3,3,1,0.5), tck= 0.01,las=1)
rvhist((sims.np[1]), border=F, col=grey(0.75), main="",
       xlab="DIN (mg/L)", xlim=c(-4,3), axes=F)
axis(2)
axis(1, at=log(c(0.05, 0.1,0.5,1,5, 10,25,50)),
     label=c(0.05,0.1,0.5,1,5,10,25,50))
##rvhist(din.sims1, add=T, col=rgb(0.5,0.5,0.5,0.5), border=grey(0.7))
##rvhist(din.sims2, add=T, col=rgb(0.25,0.25,0.25,0.25), border=grey(0.7))
lines(density(sims(din.sims1), width=2), lwd=2, lty=1)
lines(density(sims(din.sims2), width=2), lwd=2, lty=2)
abline(v=rvquantile(din.sims1, prob=c(0.75)), lwd=2)
abline(v=rvquantile(din.sims2, prob=c(0.75)), lty=2, lwd=2)
abline(v=rvquantile(sims.np[1], prob=0.25), col=grey(0.5), lwd=0.75)
box()
```




```
par(mgp=c(1.5,.25,0), mar=c(3,3,1,0.5), tck= 0.01,las=1)
rvhist((sims.np[1]), border=F, col=grey(0.75), main="",
       xlab="DIN (mg/L)", xlim=c(-4,3), axes=F)
axis(2)
axis(1, at=log(c(0.05, 0.1,0.5,1,5, 10,25,50)),
     label=c(0.05,0.1,0.5,1,5,10,25,50))
lines(density(sims(din.sims25), width=2.5), lwd=2, lty=1)
lines(density(sims(din.sims95), width=2), lwd=2, lty=3)
box()
```



The estimated 75 percentiles for the two conditional DIN distributions are

```
print(exp(rvquantile(din.sims1, prob=c(0.75))))
```

```
##           [,1]
## [1,] 0.7498113
```

```
print(exp(rvquantile(din.sims2, prob=c(0.75))))
```

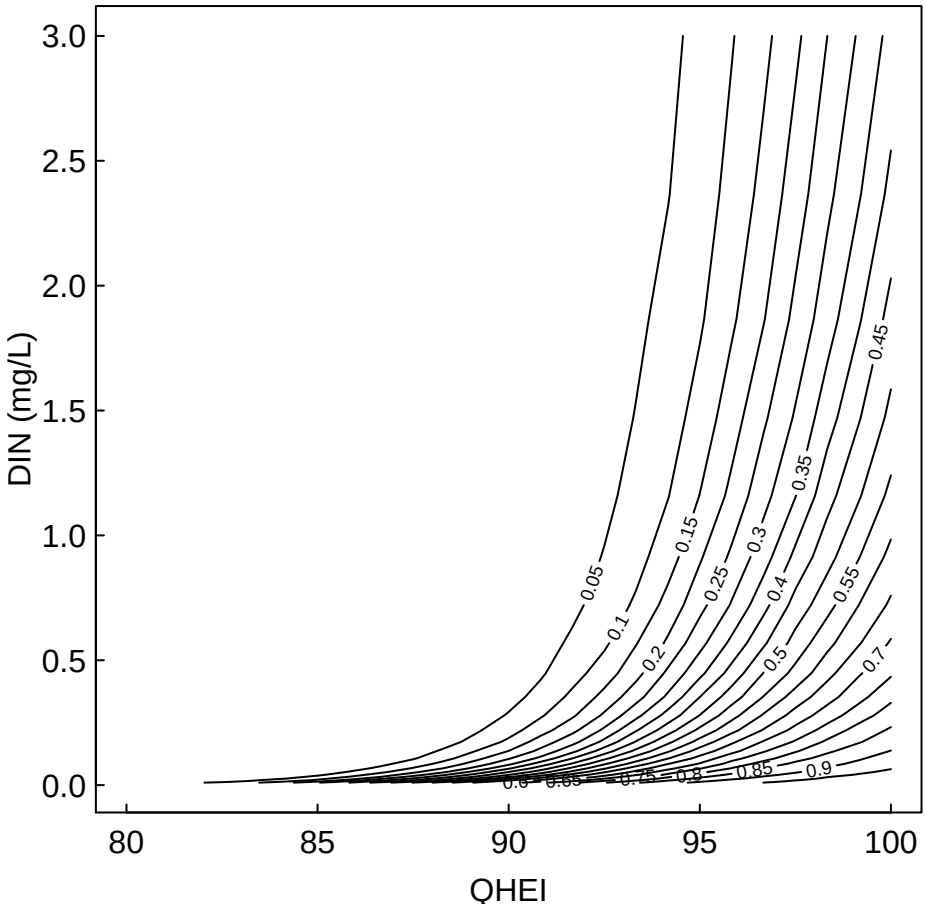
```
##          [,1]
## [1,] 1.286557
```

Additional simulations

In addition to deriving the conditional distribution of DIN given EPT$\geq$ 21 and ICI $\geq$ 50, the output can also be used to derive the conditional probability of EPT$\geq$ 21 and ICI $\geq$ 50 given a specific value of DIN. Because ICI and EPT are affected by other factors, this probability should be estimated for a given site. We use the set-aside river as an example to illustrate the process.

```
## river 02-500
r02500 <- OhioData[OhioData$RIVCODE=="02-500",
                   c("Chla","DIN","TP","AG.PCT","LOGARC",
                     "logitR","ICI","QUALEPT","QHEI")]

sims.TP <- mean(log(r02500$TP))
sims.Ag <- min(car::logit(r02500$AG.PCT))
```

```
sims.Cn <- max(r02500$LOGARC)

logDIN <- seq(log(0.01),log(3),,25)
sim.muchla <- betas.mcmc1[1]+betas.mcmc1[2]*logDIN+
    betas.mcmc1[3]*sims.TP+ betas.mcmc1[4]*sims.Ag+
        betas.mcmc1[5]*sims.Cn

sim.mudor <-
    betas.mcmc2[1]+betas.mcmc2[2]*(sim.muchla>betas.mcmc2[3])*
        (sim.muchla-betas.mcmc2[3])

sim.muici <- sim.muept <- rvmatrix(NA, 25,25)
qhri02500 <- seq(80,100,,25)
for (i in 1:25){
    sim.muici[i,] <- betas.mcmc3[1] + betas.mcmc3[2]*sim.muchla+
        betas.mcmc3[3]*sim.mudor+betas.mcmc3[4]*qhri02500[i]
    sim.muept[i,] <- betas.mcmc4[1] + betas.mcmc4[2]*sim.muchla+
        betas.mcmc4[3]*sim.mudor+betas.mcmc4[4]*qhri02500[i]
}

#sim.ici <- rvnorm(1,sim.muici, sigma.mcmc3S)
#sim.ept <- rvnorm(1,sim.muept, sigma.mcmc4S)

threeD <- matrix(Pr(sim.muici > 50 & sim.muept>log(20)), 25, 25)
#matrix(Pr(sim.ici > 50 & sim.ept>log(20)), 25, 25)
par(mgp=c(1.5,.25,0), mar=c(3,3,1,0.5), tck= 0.01,las=1)
contour(y=exp(logDIN), x=qhri02500, z=threeD, xlab="QHEI", ylab="DIN (mg/L)",
        nlevels=20)
```

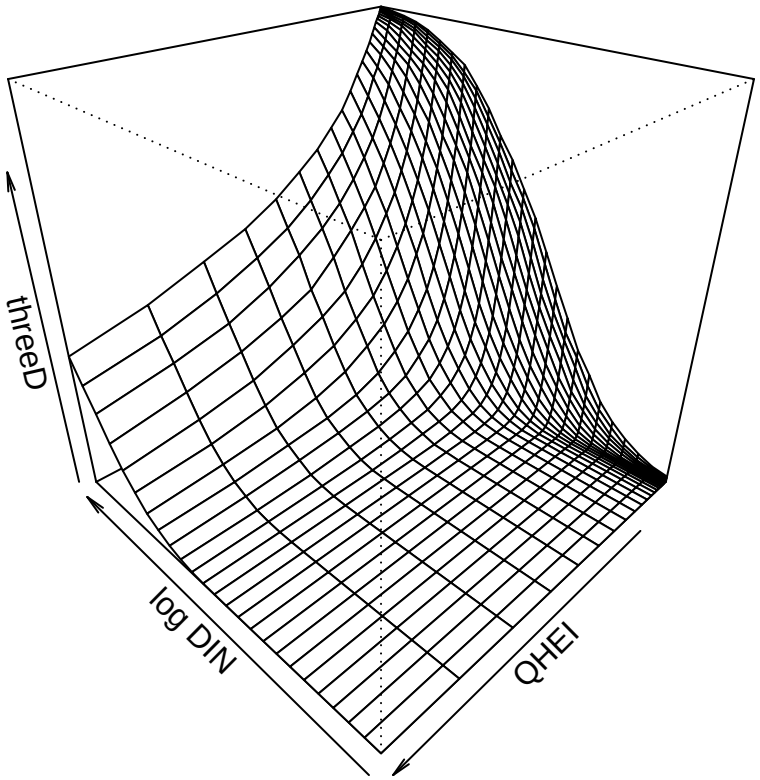



```
tikz(file=paste(plotDIR, "QNcont.tex", sep="/"), height=4, width=4.5, standAlone=F)
par(mgp=c(1.5,.25,0), mar=c(3,3,1,0.5), tck= 0.01,las=1)
contour(y=exp(logDIN), x=qhri02500, z=threeD, xlab="QHEI", ylab="DIN (mg/L)",
        nlevels=15)
dev.off()
```

```
## pdf
##   2
```

Using 3D perspective plot

```
par(mgp=c(1.5,.25,0), mar=c(3,3,1,0.5), tck= 0.01,las=1)
persp(y=exp(logDIN), x=qhri02500, z=threeD, phi=30, theta=225,
      xlab="log DIN", ylab="QHEI")
```


threeD
log DIN
QHEI