

## **Supplement 12.1. Fitting the Bayesian change point models in JAGS and R**

*Song S. Qian, Nicholas School of the Environment, Duke University, Durham, North Carolina, USA.(Current address: Center for Ecological Sciences, Tetra Tech, Inc., Research Triangle Park, North Carolina, USA.)*

This supplement provides instructions for fitting the Bayesian change point models described in Chapter 12. The three linear threshold models discussed in this chapter are implemented in JAGS through R. R is an open source statistical software package and programming environment (R Development Core Team 2012). R can interact with many other specialized programs, including, Bayesian computation programs such as BUGS (Lunn et al. 2000) and JAGS (available online at <http://www-ice.iarc.fr/~martyn/software/jags/>). JAGS is used in this chapter because it can be run in all operating systems. The programs included in this appendix have been successfully run under Windows XP, Mac OS X, and Linux (Ubuntu). No special changes are needed. To run the change point models, we first save the BUGS models into text files and call them from within R.

The BUGS codes for the three threshold models are as follow.

1. The general model

```
## the general model, saved as "threshG.txt"
model{
  for (i in 1:n){
    y[i] ~ dnorm(y.hat[i], prec)
    y.hat[i] <- (beta[1]+delta[1]*step(x[i]-phi)) +
               (beta[2]+delta[2]*step(x[i]-phi))*(x[i]-phi)
  }
  for (i in 1:2){
    delta[i] ~ dnorm(0, 0.01)
    beta[i] ~ dnorm(0, 0.001)
  }
  phi ~ dunif(1,u)
  prec <- pow(sigma, -2)
  sigma ~ dunif(0, 2)
}
```

2. The hockey stick model

```
## hockey stick model, saved as "threshH.txt"
model{
  for (i in 1:n){
    y[i] ~ dnorm(y.hat[i], prec)
    y.hat[i] <- beta[1] +
               (beta[2]+delta*step(x[i]-phi))*(x[i]-phi)
  }
  for (i in 1:2){
    beta[i] ~ dnorm(0, 0.0001)
  }
  delta ~ dnorm(0, 0.0001)
  phi ~ dunif(1,u)
  prec <- pow(sigma, -2)
  sigma ~ dunif(0, 2)
}
```

3. The Step function model

```
## the step function, saved as "threshS.txt"
model{
  for (i in 1:n){
    y[i] ~ dnorm(y.hat[i], prec[J[i]])
    y.hat[i] <- beta + delta*step(x[i]-phi)
    J[i] <- 1+step(x[i]-phi)
  }
}
```

```

    }
    delta ~ dnorm(0, 0.0001)
    beta ~ dnorm(0, 0.0001)
    phi ~ dunif(1,u)
    for (j in 1:2){
      prec[j] <- pow(sigma[j], -2)
      sigma[j] ~ dunif(0, 1)
    }
  }
}

```

All three models require input of the response variable data  $y$ , the predictor variable data  $x$ , and the range of the predictor variable  $l, u$ . This model requires input of the response variable data  $y$ , the predictor variable data  $x$ , and the lower and upper bounds of the threshold  $(l, u)$ , usually set to be the range of the predictor variable.

Once installed, JAGS can be run entirely inside R. The following R scripts are used to provide necessary input data and initial values to JAGS.

## 1. The general model

```

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

bugs.in <- function(infile, y.col, x.col,
                    y.trans="none", x.trans="none",
                    subset=NULL){
  if(!is.null(subset)) infile<-infile[subset,]
  if (y.trans=="logit") y <- logit(infile[,y.col])
  else if (y.trans=="log") y <- log(infile[,y.col])
  else y <- infile[,y.col]
  n <- length(y)
  if (x.trans=="logit") x <- logit(infile[,x.col])
  else if (x.trans=="log") x <- log(infile[,x.col])
  else x <- infile[,x.col]
  bugs.dat <- list(n=n, y=y, x=x, l=min(x), u=max(x))
  init1 <- list(delta=rnorm(2),
                beta=rnorm(2), phi=runif(1, min(x),max(x)),
                sigma=runif(1), y=ini(y, rnorm(1, 6, 1.4)))
  init2 <- list(delta=rnorm(2),
                beta=rnorm(2), phi=runif(1, min(x),max(x)),
                sigma=runif(1), y=ini(y, rnorm(1, 6, 1.4)))
}

```

```

init3 <- list(delta=rnorm(2),
              beta=rnorm(2), phi=runif(1, min(x),max(x)),
              sigma=runif(1), y=ini(y, rnorm(1, 6, 1.4)))
inits <- list(init1, init2, init3)
para <- c("beta","delta","phi", "sigma", "deviance")
return(list(data=bugs.dat, inits=inits, para=para))
}

```

## 2. The hockey stick model

```

bugs.in2 <- function(infile, y.col, x.col,
                    y.trans="none", x.trans="none",
                    subset=NULL){
  if(!is.null(subset)) infile<-infile[subset,]
  if (y.trans=="logit") y <- logit(infile[,y.col])
  else if (y.trans=="log") y <- log(infile[,y.col])
  else y <- infile[,y.col]
  n <- length(y)
  if (x.trans=="logit") x <- logit(infile[,x.col])
  else if (x.trans=="log") x <- log(infile[,x.col])
  else x <- infile[,x.col]
  bugs.dat <- list(n=n, y=y, x=x, l=min(x), u=max(x))
  init1 <- list(delta=rnorm(1),
                beta=rnorm(2), phi=runif(1, min(x),max(x)),
                sigma=runif(1), y=ini(y, rnorm(1, 6, 1.4)))
  init2 <- list(delta=rnorm(1),
                beta=rnorm(2), phi=runif(1, min(x),max(x)),
                sigma=runif(1), y=ini(y, rnorm(1, 6, 1.4)))
  init3 <- list(delta=rnorm(1),
                beta=rnorm(2), phi=runif(1, min(x),max(x)),
                sigma=runif(1), y=ini(y, rnorm(1, 6, 1.4)))
  inits <- list(init1, init2, init3)
  para <- c("beta","delta","phi", "sigma", "deviance")
  return(list(data=bugs.dat, inits=inits, para=para))
}

```

## 3. The step function model

```

bugs.in3 <- function(infile, y.col, x.col,
                    y.trans="none", x.trans="none",
                    subset=NULL){
  if(!is.null(subset)) infile<-infile[subset,]
  if (y.trans=="logit") y <- logit(infile[,y.col])
  else if (y.trans=="log") y <- log(infile[,y.col])
  else y <- infile[,y.col]
  n <- length(y)
  if (x.trans=="logit") x <- logit(infile[,x.col])
  else if (x.trans=="log") x <- log(infile[,x.col])
  else x <- infile[,x.col]

```

```

bugs.dat <- list(n=n, y=y, x=x,
               l=min(x, na.rm=T), u=max(x, na.rm=T))
init1 <- list(delta=rnorm(1),
              beta=rnorm(1), phi=runif(1, min(x),max(x)),
              sigma=runif(2), y=ini(y, rnorm(1, 6, 1.4)))
init2 <- list(delta=rnorm(1),
              beta=rnorm(1), phi=runif(1, min(x),max(x)),
              sigma=runif(2), y=ini(y, rnorm(1, 6, 1.4)))
init3 <- list(delta=rnorm(1),
              beta=rnorm(1), phi=runif(1, min(x),max(x)),
              sigma=runif(2), y=ini(y, rnorm(1, 6, 1.4)))
inits <- list(init1, init2, init3)
para <- c("beta","delta","phi", "sigma","deviance")
return(list(data=bugs.dat, inits=inits, para=para))
}

```

These three functions will return a list consisting of three elements for input data, initial values, and a list of parameters to monitor.

As a diagnostic step, all three models are run initially to identify the likely true model. The following R function will take the input data (y and x) and run the three models:

```

model.runs <- function(infile, y.col, x.col,
                      y.trans="none", x.trans="none",
                      subset=NULL, rm.na=F, xlab="",
                      main="", n.iters=25000,
                      n.keep=1000, n.chains=3){
  print("+++++")
  print(paste("y = ", y.col, "; ", "x = ", x.col))
  infile<-infile[!is.na(infile[,x.col]), ]
  if (rm.na) infile <- infile[!is.na(infile[,y.col]),]
  input.to.bugs <- bugs.in(infile=infile,
                          y.col=y.col, x.col=x.col,
                          subset=subset,
                          x.trans=x.trans, y.trans=y.trans)

  m <- jags.model("threshG.txt",
                 input.to.bugs$data,
                 input.to.bugs$inits,
                 n.chains=n.chains)
  update(m, n.iters)
  x <- coda.samples(m, input.to.bugs$para,
                   n.iter=n.iters,

```

```

        thin=max(c(1, floor(n.iters*n.chains/n.keep))))
simCoda1 <- NULL
for (i in 1:n.chains)
  simCoda1 <- rbind(simCoda1, x[[i]])

ycol <- labels(simCoda1)[[2]]=="phi"
hist(simCoda1[,ycol], xlab=xlab, main=paste(main, "(general)"),
      xlim=range(input.to.bugs$data$x, na.rm=T), col="gray",
      axes=F, prob=T)
axis(1)
# lines(density(simCoda1[,3], from=min(input.to.bugs$data$x),
# to=max(input.to.bugs$data$x), bw = "sj"))

### ii. the hockey stick model

input.to.bugs <- bugs.in2(infile=infile,y.col=y.col, x.col=x.col,
                          subset=subset,
                          x.trans=x.trans, y.trans=y.trans)
m <- jags.model("threshH.txt",
               input.to.bugs$data,
               input.to.bugs$inits,
               n.chains=n.chains)
update(m, n.iters)
x <- coda.samples(m, input.to.bugs$para,
                  n.iter=n.iters,
                  thin=max(c(1, floor(n.iters*n.chains/n.keep))))

simCoda2 <- NULL
for (i in 1:n.chains)
  simCoda2 <- rbind(simCoda2, x[[i]])

ycol <- labels(simCoda2)[[2]]=="phi"
hist(simCoda2[,ycol], xlab=xlab, main=paste(main, "(hockey)"),
      xlim=range(input.to.bugs$data$x, na.rm=T), col="gray",
      axes=F, prob=T)
axis(1)

### iii the step function model

input.to.bugs <- bugs.in3(infile=infile,y.col=y.col, x.col=x.col,
                          subset=subset,
                          x.trans=x.trans, y.trans=y.trans)
m <- jags.model("threshS.txt",
               input.to.bugs$data,
               input.to.bugs$inits,
               n.chains=n.chains)
update(m, n.iters)
x <- coda.samples(m, input.to.bugs$para,
                  n.iter=n.iters,
                  thin=max(c(1, floor(n.iters*n.chains/n.keep))))

```

```

simCoda3 <- NULL
for (i in 1:n.chains)
  simCoda3 <- rbind(simCoda3, x[[i]])

ycol<-labels(simCoda3)[[2]]=="phi"
hist(simCoda3[,ycol], xlab=xlab, main=paste(main, "(step)"),
      xlim=range(input.to.bugs$data$x, na.rm=T), col="gray",
      axes=F, prob=T)
axis(1)
# lines(density(simCoda3[,3], from=min(input.to.bugs$data$x),
#           to=max(input.to.bugs$data$x), bw = "sj"))
invisible(list("gen"=simCoda1,"hockey"=simCoda2,"step"=simCoda3))
}

```

This function takes a data frame as the required input, as well as column indicators for the response and predictor variables. For example, the following lines take x1 as the predictor and y1 and the response, run the three models, and produce three plots (histograms of the posterior distribution of the threshold).

```

par(mfrow=c(1,3), mar=c(3,0.75,3,0.75), mgp=c(1.5,.5,0))
templ <- model.runs (infile=data.frame(x=x1, y=y1),
                    y.col="y", x.col="x",
                    y.trans="none", x.trans="none",
                    subset=NULL,
                    xlab="x", main="dataset 1", n.iters=10000)

```