Ice Core

Matt Thacker

February 27, 2018

Bayesian analysis of heavy isotope (deuterium) as a function of depth in Antarctic ice core. See write up for technical details of each model and nicer looking plots.

```
library(rjags)
```

```
## Loading required package: coda
```

```
## Linked to JAGS 4.3.0
```

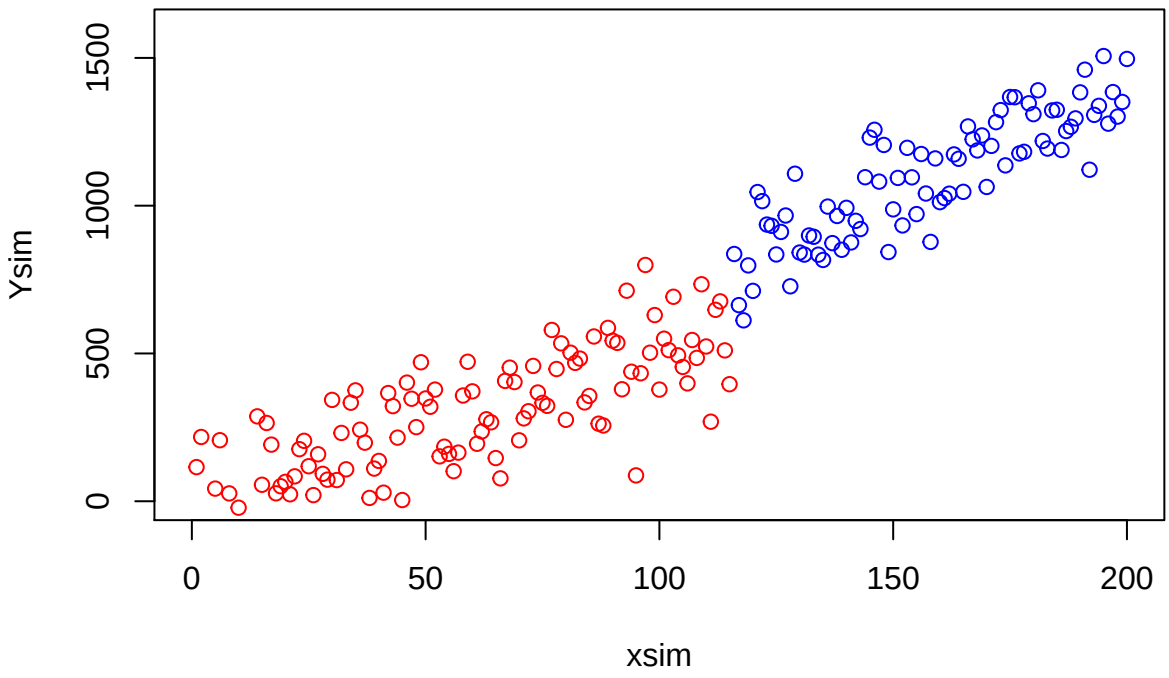
```
## Loaded modules: basemod,bugs
```

```
library(coda)

#read in data
dat<- na.omit(read.csv("Eggers_stats_no_av.csv", header=T))
#sort by depth values
dat<- dat[order(dat$depth),]
```

First we'll fit a singular breakpoint model to simulated data to ensure we've specified it well.

```
#breakpoint model
breakpoint_mod <- "
model{

  ##priors
  #betas
  beta1 ~ dmnorm(b0,Vb)
  beta2 ~ dmnorm(b0, Vb)
  #precisions
  prec1 ~ dgamma(s1,s2)
  prec2 ~ dgamma(s1,s2)
  #discrete uniform index for prior break point value
  K ~ dcat(pi)

  for (i in 1:n){
    #prebreak mu process model
    mu1[i] <- beta1[1] + beta1[2]*depth[i]

    #postbreak mu process model
      mu2[i] <- beta2[1] + beta2[2]*depth[i]
```

```
      #process model
      mu[i] <- ifelse(i>K,mu2[i],mu1[i])
    prec[i]<- ifelse(i>K, prec2, prec1)

    # data model
    Isotope[i]  ~ dnorm(mu[i],prec[i])
  }
}
"

##simulate 1 breakpoint data to test model
#simulated data
#x values
Xsim <- 1:200
#breakpoints
Ksim <- 115
#betas
Bsim1 <- c(1,5)
Bsim2<- c(3, 7)
#std. deviation
SDsim <- c(150,100)
SDsim <- c(rep(SDsim[1], Ksim),rep(SDsim[2], length(Xsim)-Ksim))


#expected y values
Yexpec <- c(Bsim1[1] + Bsim1[2]*Xsim[1:Ksim] , Bsim2[1] + Bsim2[2]*Xsim[(Ksim+1):length(Xsim)])
#simulated Y values
Ysim <- rnorm(200, Yexpec, SDsim)

#plot simulated data
plot(Xsim[1:Ksim],Ysim[1:Ksim],
     col = "red",
     ylim = c(0, 1600),
     xlim = c(0,200),
     ylab = "Ysim",
     xlab = "xsim"
     )
points(Xsim[(Ksim+1):length(Xsim)], Ysim[(Ksim+1):length(Ysim)], col = "blue")
```




```
#define data list
data.sim<- list(Isotope = Ysim, depth = Xsim, n = length(Xsim))

##priors
# regression beta means
data.sim$b0 <- as.vector(c(0,0))
# regression beta precisions
data.sim$Vb <- solve(diag(10000,2))
#uninformative error priors
data.sim$s1 <- .001
data.sim$s2 <- .001
#prior probs for discrete uniform
data.sim$pi <- rep(1/length(Xsim), length(Xsim))

#initial conditions
nchain = 3
inits.sim <- list()
for(i in 1:nchain){
 inits.sim[[i]] <- list(beta1 = rnorm(2,0,5), beta2 = rnorm(2,0,5), prec1 = runif(1,1/100,1/20), prec2 =
}


##initialize and sample from model
#jags object
singleBP.sim   <- jags.model(file = textConnection(breakpoint_mod),
                             data = data.sim,
```

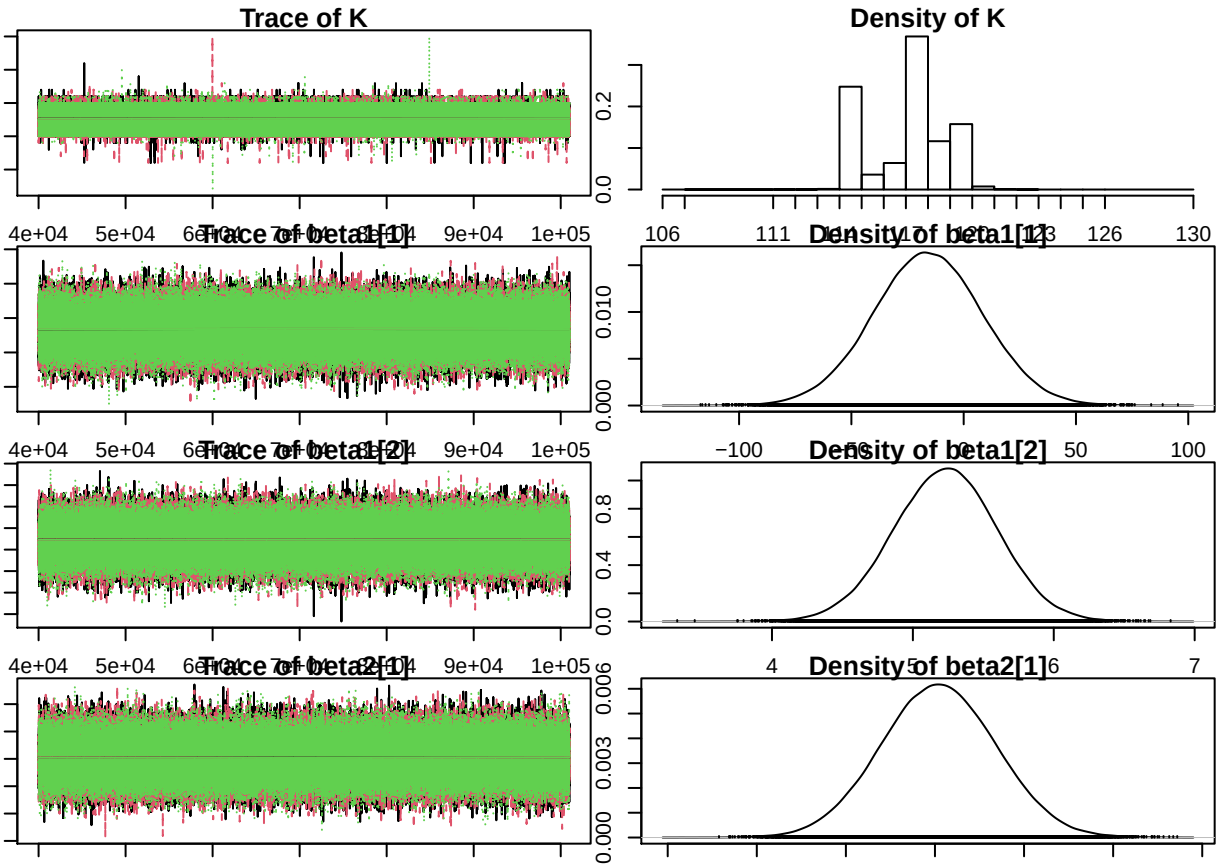
```
                              inits = inits.sim,
                              n.chains = nchain)
```

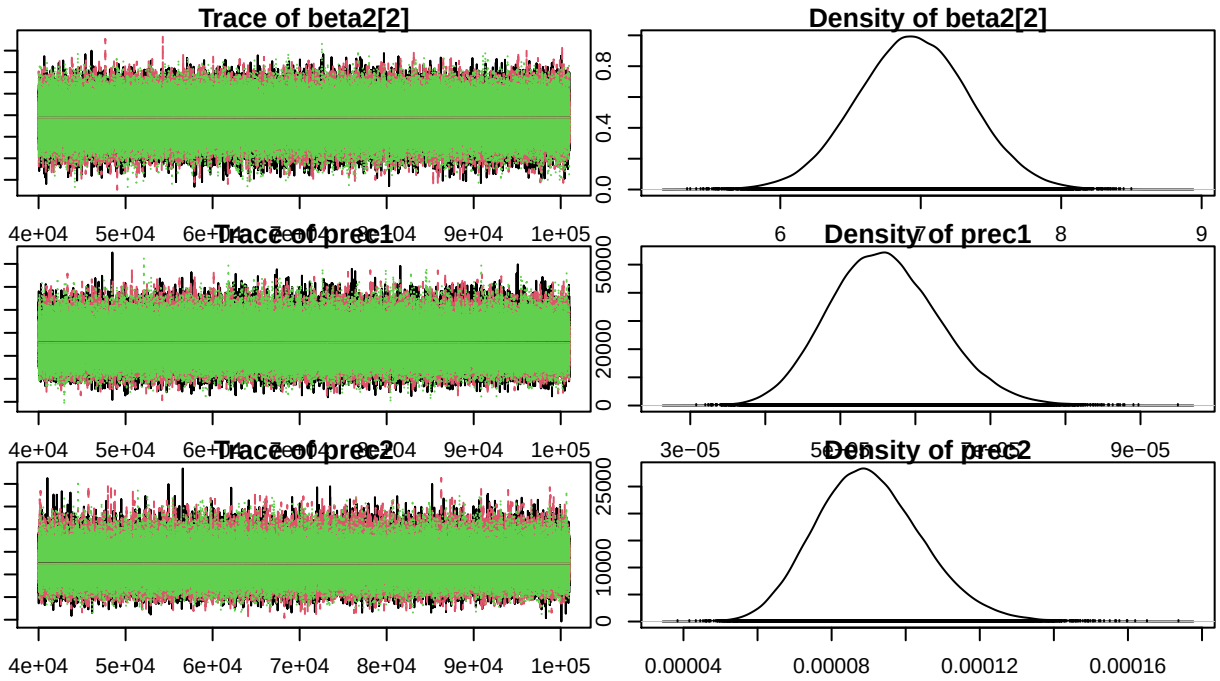
```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 200
##    Unobserved stochastic nodes: 5
##    Total graph size: 2221
##
## Initializing model
```

```
#sample from model
singleBP.sim.out<- coda.samples (model = singleBP.sim,
                            variable.names = c("K", "beta1", "beta2", "prec1", "prec2"),
                                n.iter = 100000)

#check for convergence
gelman.diag(singleBP.sim.out)
```

```
## Potential scale reduction factors:
##
##          Point est. Upper C.I.
## K                 1          1
## beta1[1]          1          1
## beta1[2]          1          1
## beta2[1]          1          1
## beta2[2]          1          1
## prec1             1          1
## prec2             1          1
##
## Multivariate psrf
##
## 1
```

```
#burnin
burnin <- 40000
singleBP.sim.burn <- window(singleBP.sim.out, start = burnin)

#plot outputs
#deal with graphics issue
par(mar=c(1,1,1,1))
plot(singleBP.sim.burn)
```


Trace of K
Density of K
Trace of beta1[1]
Density of beta1[1]
Trace of beta1[2]
Density of beta1[2]
Trace of beta2[1]
Density of beta2[1]
4e+04
5e+04
9e+04
1e+05
0.0
0.2
0.000
0.010
0.4
0.8
0.003
0.006
106
111
126
130
−100
50
100
4
5
6
7




```
summary(singleBP.sim.burn)
```

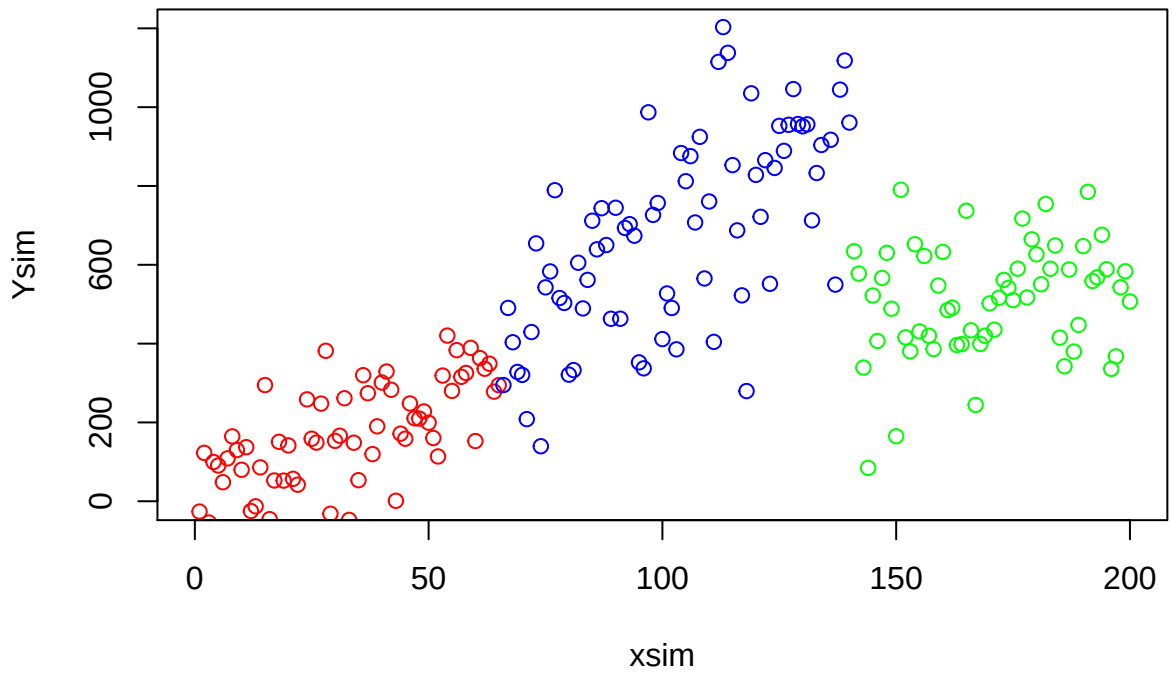
```
##
## Iterations = 40000:101000
## Thinning interval = 1
## Number of chains = 3
## Sample size per chain = 61001
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##                Mean        SD  Naive SE Time-series SE
## K          1.176e+02 1.767e+00 4.131e-03      4.872e-03
## beta1[1] -1.588e+01 2.450e+01 5.727e-02      5.840e-02
## beta1[2]  5.230e+00 3.691e-01 8.628e-04      9.181e-04
## beta2[1]  4.287e+00 6.443e+01 1.506e-01      1.635e-01
## beta2[2]  6.947e+00 3.998e-01 9.345e-04      9.967e-04
## prec1     5.594e-05 7.347e-06 1.717e-08      1.934e-08
## prec2     9.022e-05 1.435e-05 3.354e-08      4.404e-08
##
## 2. Quantiles for each variable:
##
##                2.5%        25%        50%        75%     97.5%
## K          1.150e+02  1.160e+02  1.180e+02 1.190e+02 1.200e+02
## beta1[1] -6.414e+01 -3.225e+01 -1.592e+01 4.982e-01 3.212e+01
```

```
## beta1[2]  4.499e+00  4.983e+00  5.233e+00 5.478e+00 5.952e+00
## beta2[1] -1.212e+02 -3.922e+01  4.067e+00 4.773e+01 1.306e+02
## beta2[2]  6.163e+00  6.678e+00  6.948e+00 7.217e+00 7.729e+00
## prec1     4.250e-05  5.080e-05  5.563e-05 6.072e-05 7.122e-05
## prec2     6.433e-05  8.020e-05  8.946e-05 9.942e-05 1.205e-04
```

Next we'll do the same with our two breakpoint model.

```
##two breakpoint model
breakpoint2_mod <- "
model{

  ##priors
  #betas
  for (i in 1:3){
    beta[i,1:2] ~ dmnorm(b0,Vb)
  }

  #precisions
  for (i in 1:3){
    prec[i] ~ dgamma(s1,s2)
  }

  #index for prior break point value
  for (i in 1:2){
    k0[i] ~ dcat(pi)
  }
  K[1:2] <- sort(k0)

  ##loop through each observation
  for (i in 1:Nobs){

    #calc mu at each set of betas for depth i
    for (j in 1:3){
      mu[i,j] <- beta[j,1] + beta[j,2]*depth[i]
    }

    ##produce final process models
    Ex_int[i] <- ifelse(i>K[1], mu[i,2], mu[i,1])
    Ex[i] <- ifelse(i>K[2], mu[i,3], Ex_int[i])


    #same with precision
    prec_int[i]<- ifelse(i>K[1], prec[2], prec[1])
    prec_ex[i] <- ifelse(i>K[2], prec[3], prec_int[i])  #final process model output

    # data model
    Isotope[i]  ~ dnorm(Ex[i],prec_ex[i])
  }
}
"

##simulate 2 breakpoint data to test model
#simulated data
```

```r
#x values
Xsim <- 1:200
#breakpoints
Ksim1 <- 65
Ksim2 <- 140
#betas
Bsim1 <- c(1,5)
Bsim2<- c(0, 7)
Bsim3<- c(2,3)
#std. deviation
SDsim <- c(100,200,150)
SDsim <- c(rep(SDsim[1], Ksim1), rep(SDsim[2], Ksim2-Ksim1), rep(SDsim[3], length(Xsim)-Ksim2))


#expected y values
Yexpec <- c(Bsim1[1] + Bsim1[2]*Xsim[1:Ksim1] , Bsim2[1] + Bsim2[2]*Xsim[(Ksim1+1):Ksim2], Bsim3[1] + Bs
#simulated Y values
Ysim <- rnorm(length(Xsim), Yexpec, SDsim)

#plot simulated data
plot(Xsim[1:Ksim1],Ysim[1:Ksim1],
     col = "red",
     ylim = c(0, 1200),
     xlim = c(0,200),
     ylab = "Ysim",
     xlab = "xsim"
     )
points(Xsim[(Ksim1+1):Ksim2], Ysim[(Ksim1+1):Ksim2], col = "blue")
points(Xsim[(Ksim2+1):length(Xsim)], Ysim[(Ksim2+1):length(Xsim)], col = "green")
```




```
#define data list
data.sim<- list(Isotope = Ysim, depth = Xsim, Nobs = length(Xsim))

##prior params
# regression beta means
data.sim$b0 <- as.vector(c(0,0))
# regression beta precisions
data.sim$Vb <- solve(diag(1000,2))
#uninformative error priors
data.sim$s1 <- .001
data.sim$s2 <- .001
#prior probs for discrete uniform
data.sim$pi <- rep(1/length(Xsim), length(Xsim))

#initial conditions
nchain = 3

#sample initial beta vals
beta.init <- list()

for (i in 1:nchain){
  beta.mat <- matrix(nrow=3,ncol=2)

  for (j in 1:3){
    beta.mat[j,] <- rnorm(2,0,5)
  }
```

```
  beta.init[[i]] <- beta.mat
}


#sample initial prec vals
prec.init <- list()

for (i in 1:nchain){
  prec.mat <- rep(NA, 3)

  for (j in 1:3){
    prec.mat[j] <- runif(1,1/1000,1/100)
  }

  prec.init[[i]] <- prec.mat
}


inits.bp2.sim <- list()
for(i in 1:nchain){
 inits.bp2.sim[[i]] <- list(
   beta = beta.init[[i]],
   prec = prec.init[[i]], k0 = sort(sample(1:length(Xsim), 2)))
}

##initialize and sample from model
#jags object
twoBP.sim   <- jags.model(file = textConnection(breakpoint2_mod),
                          data = data.sim,
                          inits = inits.bp2.sim,
                          n.chains = nchain)
```

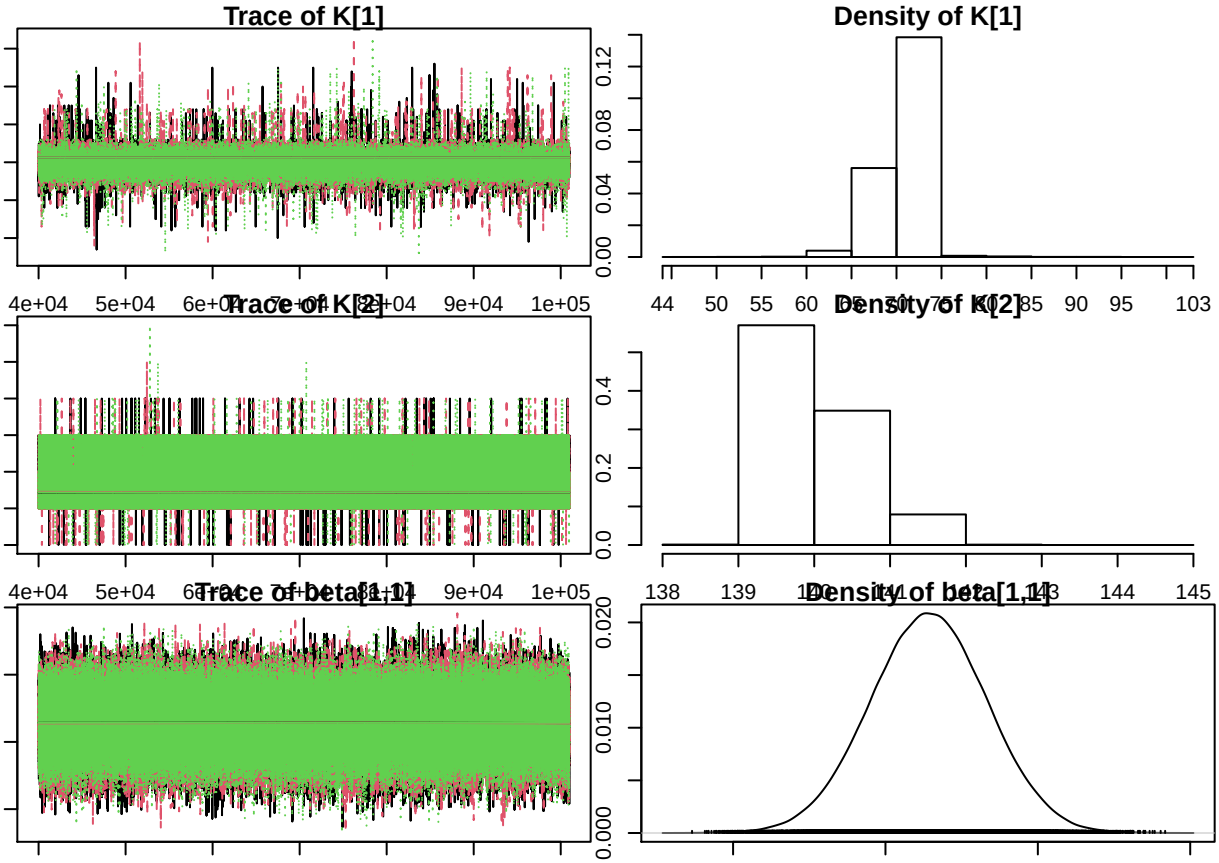
```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 200
##    Unobserved stochastic nodes: 8
##    Total graph size: 3230
##
## Initializing model
```

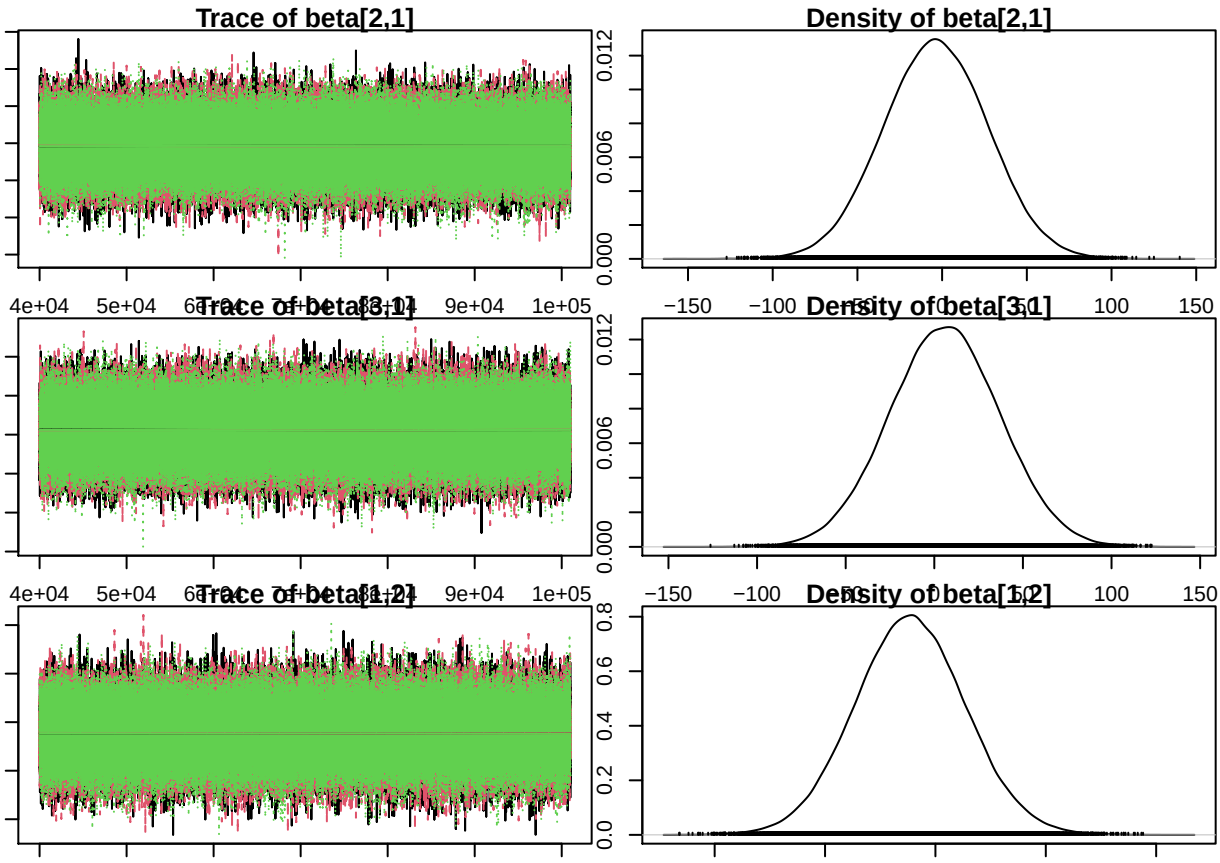
```
#sample from model
twoBP.sim.out2<- coda.samples (model = twoBP.sim,
                           variable.names = c("K", "beta", "prec"),
                               n.iter = 100000)

#check for convergence
gelman.diag(twoBP.sim.out2)
```

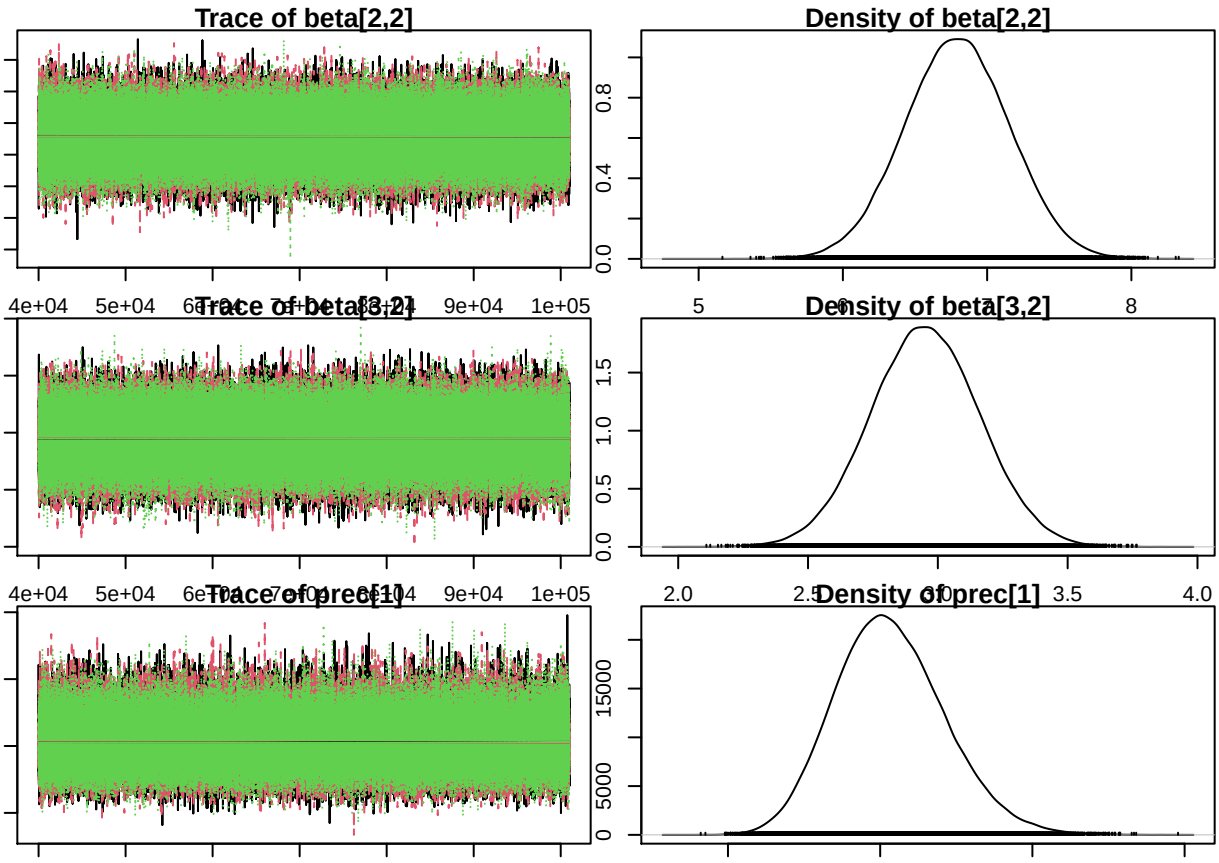
```
## Potential scale reduction factors:
##
##           Point est. Upper C.I.
```

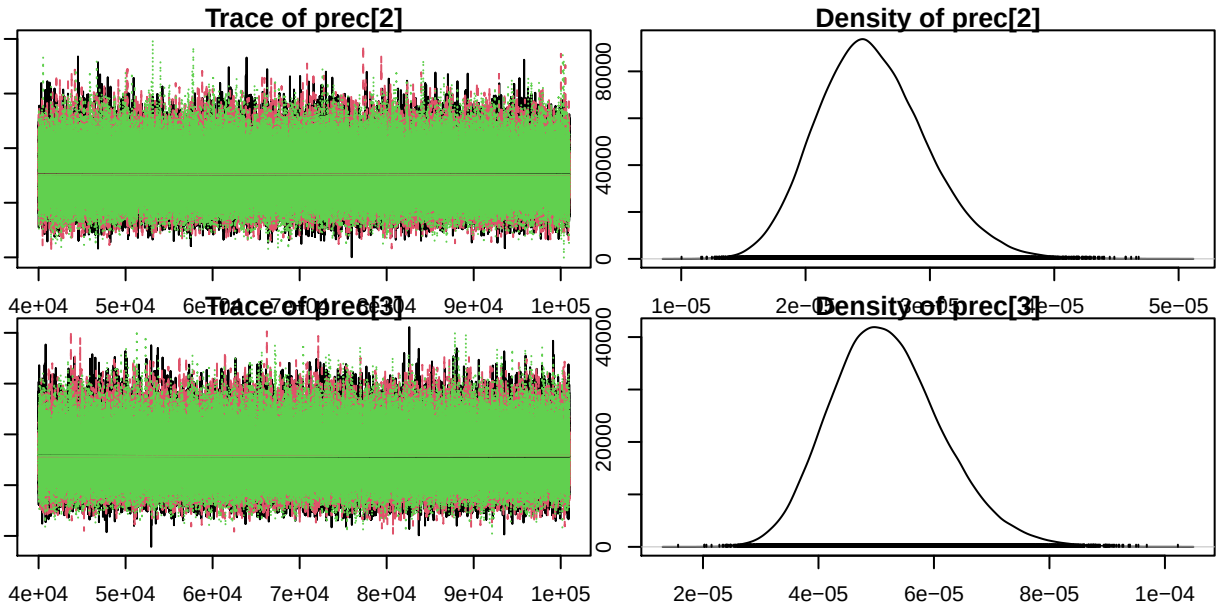
```
## K[1]                1          1
## K[2]                1          1
## beta[1,1]           1          1
## beta[2,1]           1          1
## beta[3,1]           1          1
## beta[1,2]           1          1
## beta[2,2]           1          1
## beta[3,2]           1          1
## prec[1]             1          1
## prec[2]             1          1
## prec[3]             1          1
##
## Multivariate psrf
##
## 1
```

```
#burn
burnin <- 40000
twoBP.sim.burn <- window(twoBP.sim.out2, start = burnin)

#outputs
par(mar=c(1,1,1,1))
plot(twoBP.sim.burn)
```




Trace of beta[2,1]
Density of beta[2,1]
Trace of beta[3,1]
Density of beta[3,1]
Trace of beta[1,2]
Density of beta[1,2]
4e+04
5e+04
9e+04
1e+05
0.000
0.006
0.012
0.0
0.2
0.4
0.6
0.8
−150
−100
100
150


Trace of beta[2,2]
Density of beta[2,2]
Trace of beta[3,2]
Density of beta[3,2]
Trace of prec[1]
Density of prec[1]
4e+04
5e+04
9e+04
1e+05
0.0
0.4
0.8
0.5
1.0
1.5
0
5000
15000
2.0
3.5
4.0
5
8




```
summary(twoBP.sim.burn)
```

```
##
## Iterations = 40000:101000
## Thinning interval = 1
## Number of chains = 3
## Sample size per chain = 61001
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##                 Mean        SD  Naive SE Time-series SE
## K[1]       7.086e+01 2.036e+00 4.760e-03      5.108e-03
## K[2]       1.405e+02 6.451e-01 1.508e-03      1.590e-03
## beta[1,1]  1.414e+01 1.913e+01 4.472e-02      4.532e-02
## beta[2,1] -3.755e+00 3.082e+01 7.205e-02      7.323e-02
## beta[3,1]  5.908e+00 3.119e+01 7.292e-02      7.238e-02
## beta[1,2]  4.772e+00 5.022e-01 1.174e-03      1.219e-03
## beta[2,2]  6.792e+00 3.617e-01 8.456e-04      8.677e-04
## beta[3,2]  2.951e+00 2.099e-01 4.906e-04      4.906e-04
## prec[1]    1.033e-04 1.792e-05 4.190e-08      5.674e-08
## prec[2]    2.535e-05 4.353e-06 1.018e-08      1.244e-08
## prec[3]    5.189e-05 9.612e-06 2.247e-08      2.775e-08
##
## 2. Quantiles for each variable:
```

```
##
##                 2.5%        25%        50%       75%     97.5%
## K[1]       6.600e+01  7.000e+01  7.100e+01 7.200e+01 7.400e+01
## K[2]       1.400e+02  1.400e+02  1.400e+02 1.410e+02 1.420e+02
## beta[1,1] -2.353e+01  1.314e+00  1.417e+01 2.701e+01 5.150e+01
## beta[2,1] -6.414e+01 -2.454e+01 -3.744e+00 1.713e+01 5.660e+01
## beta[3,1] -5.530e+01 -1.509e+01  5.938e+00 2.689e+01 6.713e+01
## beta[1,2]  3.791e+00  4.436e+00  4.771e+00 5.107e+00 5.767e+00
## beta[2,2]  6.082e+00  6.548e+00  6.791e+00 7.036e+00 7.498e+00
## beta[3,2]  2.540e+00  2.809e+00  2.950e+00 3.092e+00 3.363e+00
## prec[1]    7.094e-05  9.077e-05  1.024e-04 1.149e-04 1.410e-04
## prec[2]    1.755e-05  2.230e-05  2.509e-05 2.814e-05 3.460e-05
## prec[3]    3.485e-05  4.515e-05  5.129e-05 5.795e-05 7.251e-05
```

Now lets fit our one breakpoint model to the ice core data

```
#define data list
data.bp<- list(Isotope = dat$deuturium, depth = dat$depth, n = nrow(dat))

##priors
# regression beta means
data.bp$b0 <- as.vector(c(0,0))
# regression beta precisions
data.bp$Vb <- solve(diag(10000,2))
# uninformative error prior
data.bp$s1 <- .1
data.bp$s2 <- .1
#prior probs for discrete uniform
data.bp$pi <- rep(1/nrow(dat), nrow(dat))

#initial conditions
nchain = 3
inits.bp <- list()
for(i in 1:nchain){   #set different values for each chain
 inits.bp[[i]] <- list(beta1 = rnorm(2,0,5), beta2 = rnorm(2,0,5), prec1 = runif(1,1/1000,1/100), prec2
}

#estimate model and sample from it
bp.model   <- jags.model(file = textConnection(breakpoint_mod),
                         data = data.bp,
                         inits = inits.bp,
                         n.chains = nchain)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 57
##    Unobserved stochastic nodes: 5
##    Total graph size: 620
##
## Initializing model
```

```
bp.out<- coda.samples (model = bp.model,
                           variable.names = c("K", "beta1", "beta2", "prec1","prec2"),
                               n.iter = 100000)

#convergence?
gelman.diag(bp.out)
```

```
## Potential scale reduction factors:
##
##          Point est. Upper C.I.
## K                 1          1
## beta1[1]          1          1
## beta1[2]          1          1
## beta2[1]          1          1
## beta2[2]          1          1
## prec1             1          1
## prec2             1          1
##
## Multivariate psrf
##
## 1
```

```
#burnin
burnin <- 40000
bp.burn <- window(bp.out, start=burnin)

#outputs
summary(bp.burn)
```

```
##
## Iterations = 40000:101000
## Thinning interval = 1
## Number of chains = 3
## Sample size per chain = 61001
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##                Mean      SD  Naive SE Time-series SE
## K         2.716e+01 0.86105 2.013e-03      3.000e-03
## beta1[1] -2.131e+02 1.03298 2.415e-03      2.773e-03
## beta1[2] -1.021e-01 0.01452 3.393e-05      4.594e-05
## beta2[1] -2.418e+02 3.71917 8.694e-03      9.458e-03
## beta2[2] -3.691e-03 0.01090 2.548e-05      2.684e-05
## prec1     1.254e-01 0.03676 8.592e-05      1.305e-04
## prec2     2.119e-02 0.00568 1.328e-05      1.884e-05
##
## 2. Quantiles for each variable:
##
##                2.5%        25%        50%        75%      97.5%
## K          25.00000   27.00000  2.700e+01   28.00000   29.00000
## beta1[1] -215.07249 -213.74528 -2.131e+02 -212.39656 -210.98746
```

```
## beta1[2]   -0.13168   -0.11137 -1.018e-01   -0.09257   -0.07416
## beta2[1] -249.11164 -244.21936 -2.418e+02 -239.30270 -234.43040
## beta2[2]   -0.02522   -0.01089 -3.672e-03    0.00348    0.01779
## prec1       0.06413    0.09896  1.218e-01    0.14786    0.20723
## prec2       0.01154    0.01712  2.071e-02    0.02468    0.03371
```

```
effectiveSize(bp.burn)
```

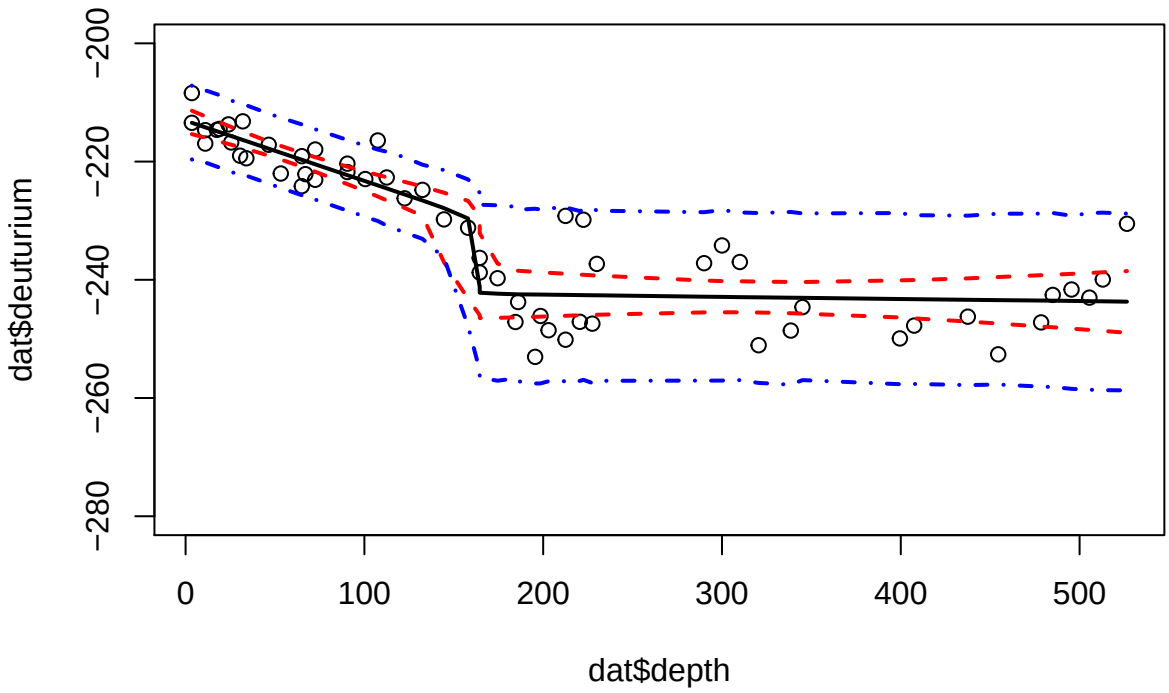
```
##         K  beta1[1]  beta1[2]  beta2[1]  beta2[2]     prec1     prec2
##  82701.47 138768.66  99900.99 154644.36 164902.55  79271.29  90934.70
```

```
gelman.diag(bp.burn)
```

```
## Potential scale reduction factors:
##
##          Point est. Upper C.I.
## K                 1          1
## beta1[1]          1          1
## beta1[2]          1          1
## beta2[1]          1          1
## beta2[2]          1          1
## prec1             1          1
## prec2             1          1
##
## Multivariate psrf
##
## 1
```

```
##predictive and credible intervals
#define variables and empty storage
bp.mat <- as.matrix(bp.burn)
#number of samples for generating intervals
nsamp <- 10000
#sample row indices
samp <- sample.int(nrow(bp.mat),nsamp)
#x values to predict over
xpred <- dat$depth
#number of predictions to make
npred <- length(xpred)

##storage for vals
# storage for predictive interval
prebreak.ypred <- matrix(NA,nrow=nsamp,ncol=npred)
postbreak.ypred <- matrix(NA,nrow=nsamp,ncol=npred)
#storage for credible interval
prebreak.ycred <- matrix(NA,nrow=nsamp,ncol=npred)
postbreak.ycred <- matrix(NA,nrow=nsamp,ncol=npred)

#loop through to calculate intervals
for(g in seq_len(nsamp)){
  #sample from parameters by row
  theta = bp.mat[samp[g],]
```

```
  #loop through each row by column in ycred and ypred
  for (j in 1:npred){
    #post breakpoint model
    if (j > theta["K"]){
      postbreak.ycred[g,j] <- theta["beta2[1]"] + theta["beta2[2]"]*xpred[j]
      postbreak.ypred[g,j] <- rnorm(1,postbreak.ycred[g,j],1/sqrt(theta["prec2"]))
    } else {                #pre breakpoint model
      prebreak.ycred[g,j] <- theta["beta1[1]"] + theta["beta1[2]"]*xpred[j]
      prebreak.ypred[g,j] <- rnorm(1,prebreak.ycred[g,j],1/sqrt(theta["prec1"]))
    }
  }
}

#create 1 ycred and ypred matrix
ycred <- matrix(NA,nrow=nsamp,ncol=npred)
ypred <- matrix(NA,nrow=nsamp,ncol=npred)

#loop through and combine pre and post break values
for (i in 1:nsamp){
  ycred[i,] <- na.omit(c(prebreak.ycred[i,], postbreak.ycred[i,]))
  ypred[i,] <- na.omit(c(prebreak.ypred[i,], postbreak.ypred[i,]))
}

##calc intervals
# credible interval and median
ci.bp <- apply(ycred,2,quantile,c(0.025,0.5,0.975))
# prediction interval
pi.bp <- apply(ypred,2,quantile,c(0.025,0.975))
#plot em
plot(dat$depth, dat$deuturium, ylim= c(-280, -200))
lines(xpred, ci.bp[2,], type="l", col = 1, lwd= 2)  #median model
lines(xpred, ci.bp[1,], type="l", lty = 2, col = "red", lwd = 2)  #CI
lines(xpred, ci.bp[3,], type="l", lty = 2, col = "red", lwd = 2)
lines(xpred, pi.bp[1,], type="l", lty = 4, col = "blue", lwd = 2)   #PI
lines(xpred, pi.bp[2,], type="l", lty = 4, col = "blue", lwd = 2)
```



Next we'll try fitting a model for two breakpoints which will fail to converge on values for K, or converge to a value for K[2] which is the final index available. In either case the data do no support the adoption of this model.

```
#define data list
data.bp2<- list(Isotope = dat$deuturium, depth = dat$depth, Nobs = nrow(dat))

##priors
# regression beta means
data.bp2$b0 <- as.vector(c(0,0))
# regression beta precisions
data.bp2$Vb <- solve(diag(10000,2))
# uninformative error prior
data.bp2$s1 <- .01
data.bp2$s2 <- .01
#prior probs for discrete uniform
data.bp2$pi <- rep(1/nrow(dat), nrow(dat))


#initial conditions
nchain = 3

#sample initial beta vals
beta.init <- list()

for (i in 1:nchain){
```