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

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

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 57
##    Unobserved stochastic nodes: 8
##    Total graph size: 900
##
## Initializing model
```

```
#sample from model
bp2.out<- coda.samples (model = bp2.model,
                        variable.names = c("K", "beta", "prec"),
                            n.iter = 250000)

#convergence?
```

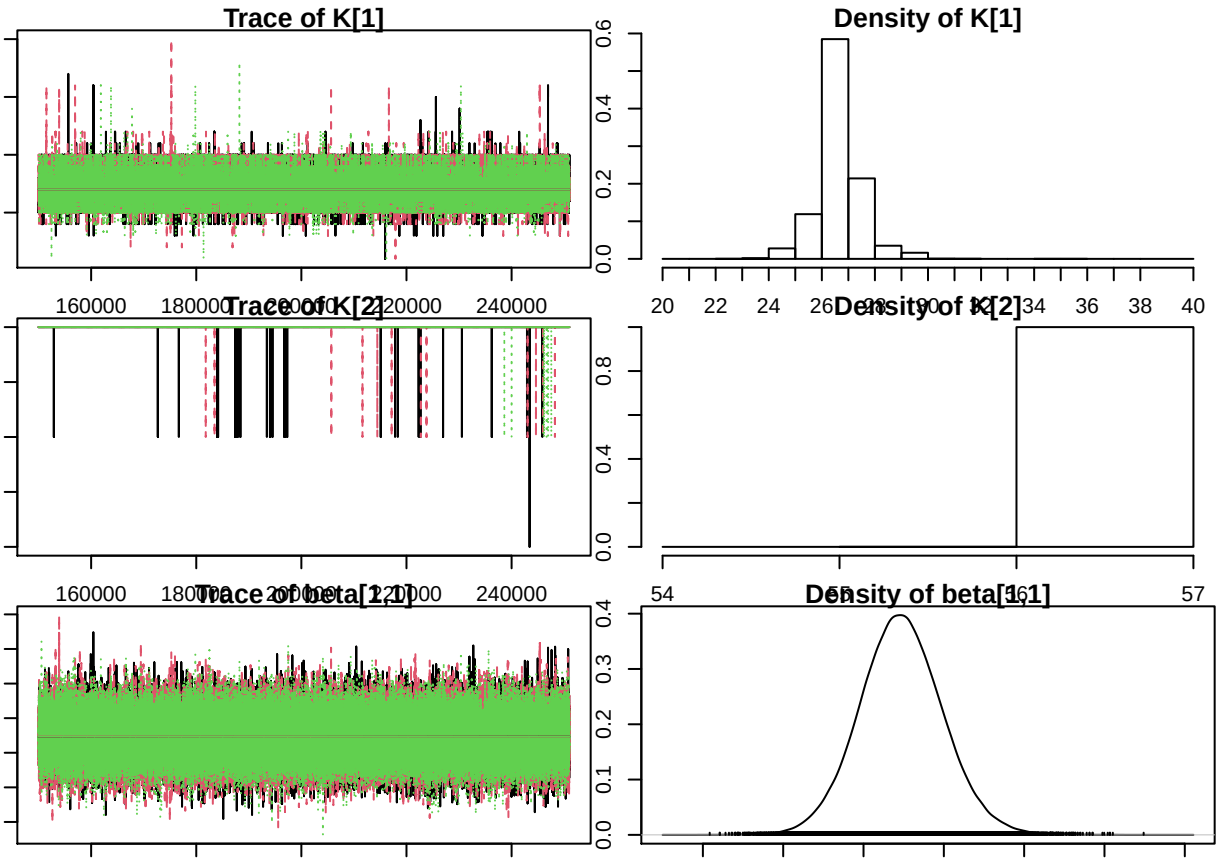
```
gelman.diag(bp2.out)
```

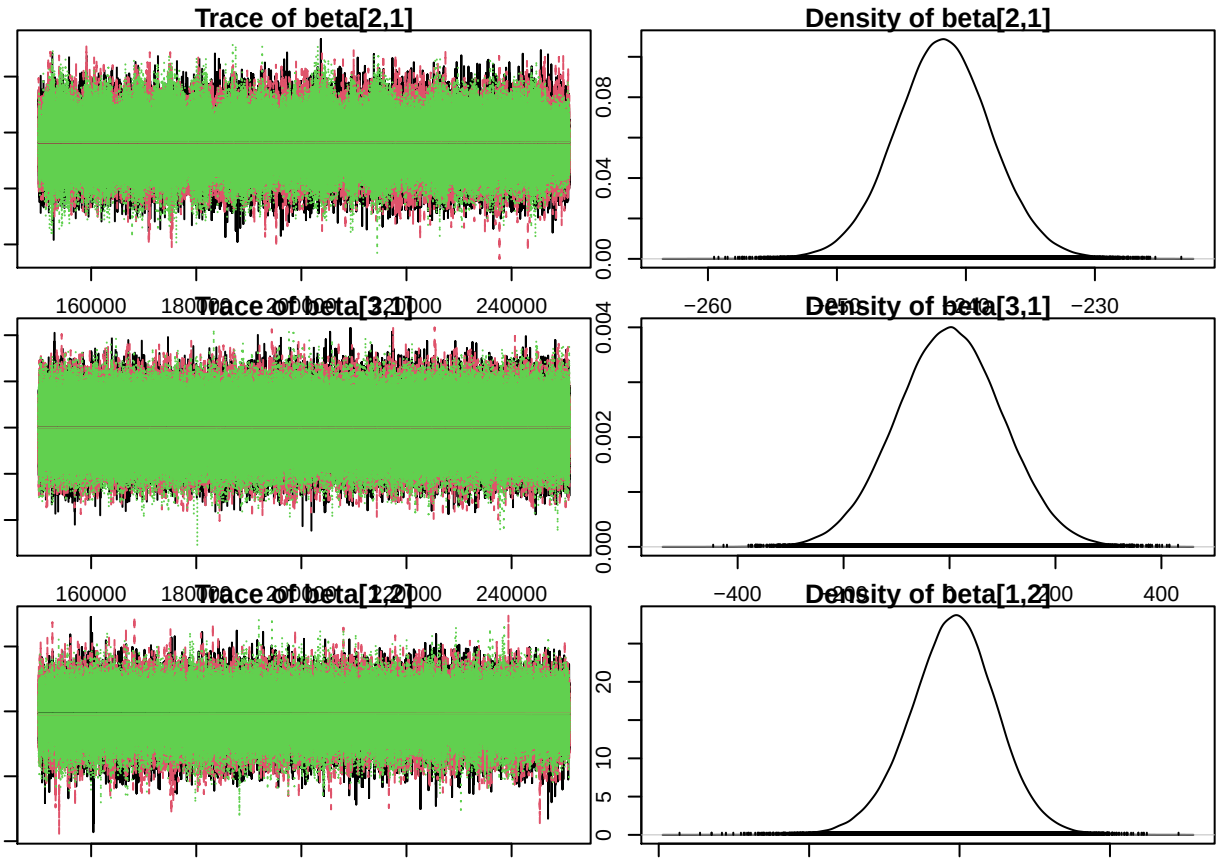
```
## Potential scale reduction factors:
##
##           Point est. Upper C.I.
## K[1]            1.00       1.00
## K[2]            1.08       1.08
## beta[1,1]       1.00       1.00
## beta[2,1]       1.00       1.00
## beta[3,1]       1.00       1.00
## beta[1,2]       1.00       1.00
## beta[2,2]       1.00       1.00
## beta[3,2]       1.00       1.00
## prec[1]         1.00       1.00
## prec[2]         1.00       1.00
## prec[3]         1.30       1.70
##
## Multivariate psrf
##
## 1.01
```

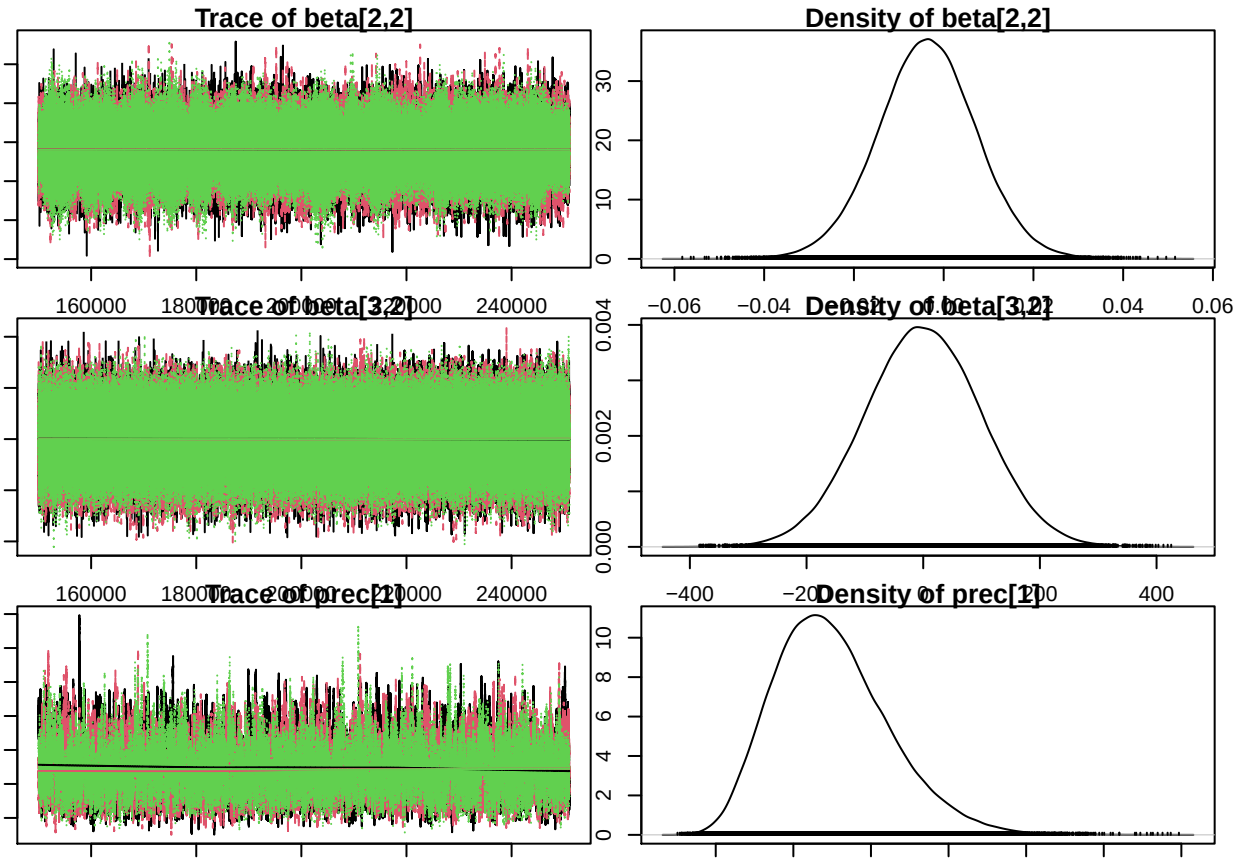
```
#burnin
burnin <- 150000
bp2.burn <- window(bp2.out, start=burnin)

#check for convergence
par(mar=c(1,1,1,1))
gelman.diag(bp2.burn)
```

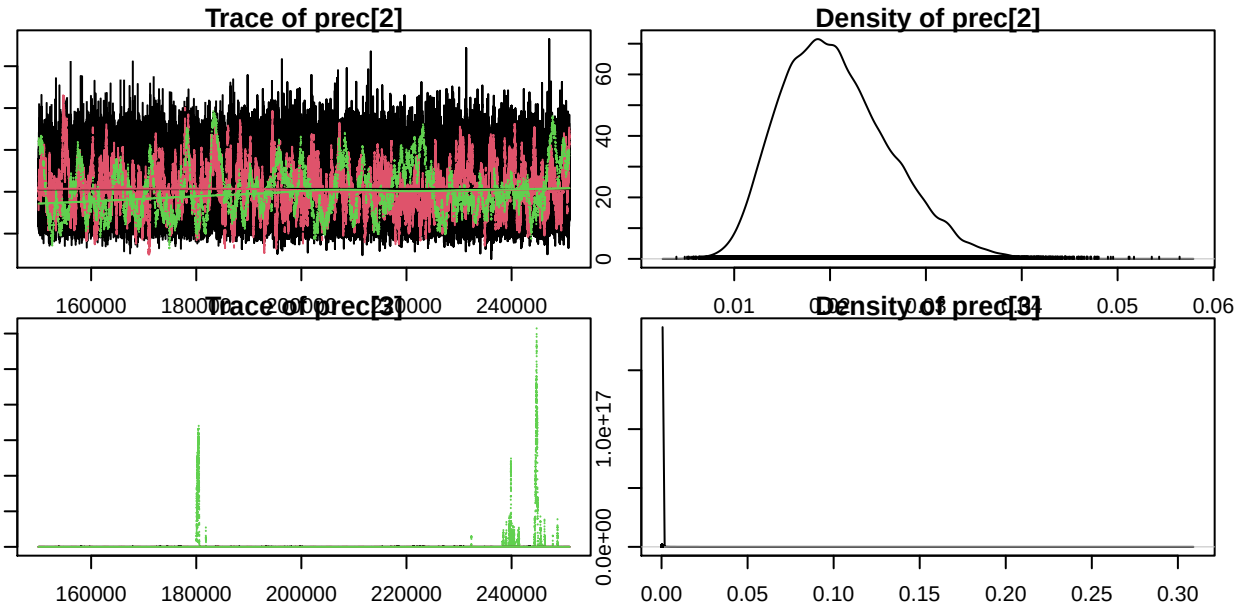
```
## Potential scale reduction factors:
##
##           Point est. Upper C.I.
## K[1]            1.00       1.00
## K[2]            1.10       1.10
## beta[1,1]       1.00       1.00
## beta[2,1]       1.00       1.00
## beta[3,1]       1.00       1.00
## beta[1,2]       1.00       1.00
## beta[2,2]       1.00       1.00
## beta[3,2]       1.00       1.00
## prec[1]         1.00       1.01
## prec[2]         1.01       1.02
## prec[3]         1.30       1.59
##
## Multivariate psrf
##
## 1.01
```

```
plot(bp2.burn)
```


Trace of K[1]
Density of K[1]
Trace of K[2]
Density of K[2]
Trace of beta[1,1]
Density of beta[1,1]


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]
160000
240000
0.00
0.04
0.08
0.000
0.002
0.004
0
5
10
20
−260
−230
−400
400


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]
160000
240000
0
10
20
30
0.000
0.002
0.004
0
2
4
6
8
10
−0.06
−0.04
0.04
0.06
−400
400



Now lets fit our polynomial regression models starting with quadratic (2nd order)

```
#bayesian fitting of polynomial (2nd order) regression (linear with respect to parameters)
quad_regression<- "
model{

  ##priors
  #betas
  beta ~ dmnorm(b0,Vb)
  #precision
  prec ~ dgamma(s1,s2)

  ##process and data model
  for(i in 1:n){
    # process model
      mu[i] <- beta[1] + beta[2]*x[i,1] + beta[3]*x[i,2]
      # data model
      Isotope[i]  ~ dnorm(mu[i],prec)
  }
}
"

##fit model
#specify data values
data.quad<- list(Isotope = dat$deuturium, x = cbind(dat$depth, dat$depth^2), n = nrow(dat))
```

```r
##priors
# regression beta means
data.quad$b0 <- as.vector(c(0,0,0))
# regression beta precisions
data.quad$Vb <- solve(diag(10000,3))
# uninformative error priors
data.quad$s1 <- .1
data.quad$s2 <- .1

#initial conditions
inits.quad<- list()
for (i in 1:nchain){
  inits.quad[[i]]<- list(beta = rnorm(3,0,5), prec = runif(1,1/1000,1/100))
}

##estimate and sample from model
#JAGS object
quad.model   <- jags.model(file = textConnection(quad_regression),
                           data = data.quad,
                           inits = inits.quad,
                           n.chains = nchain)
```

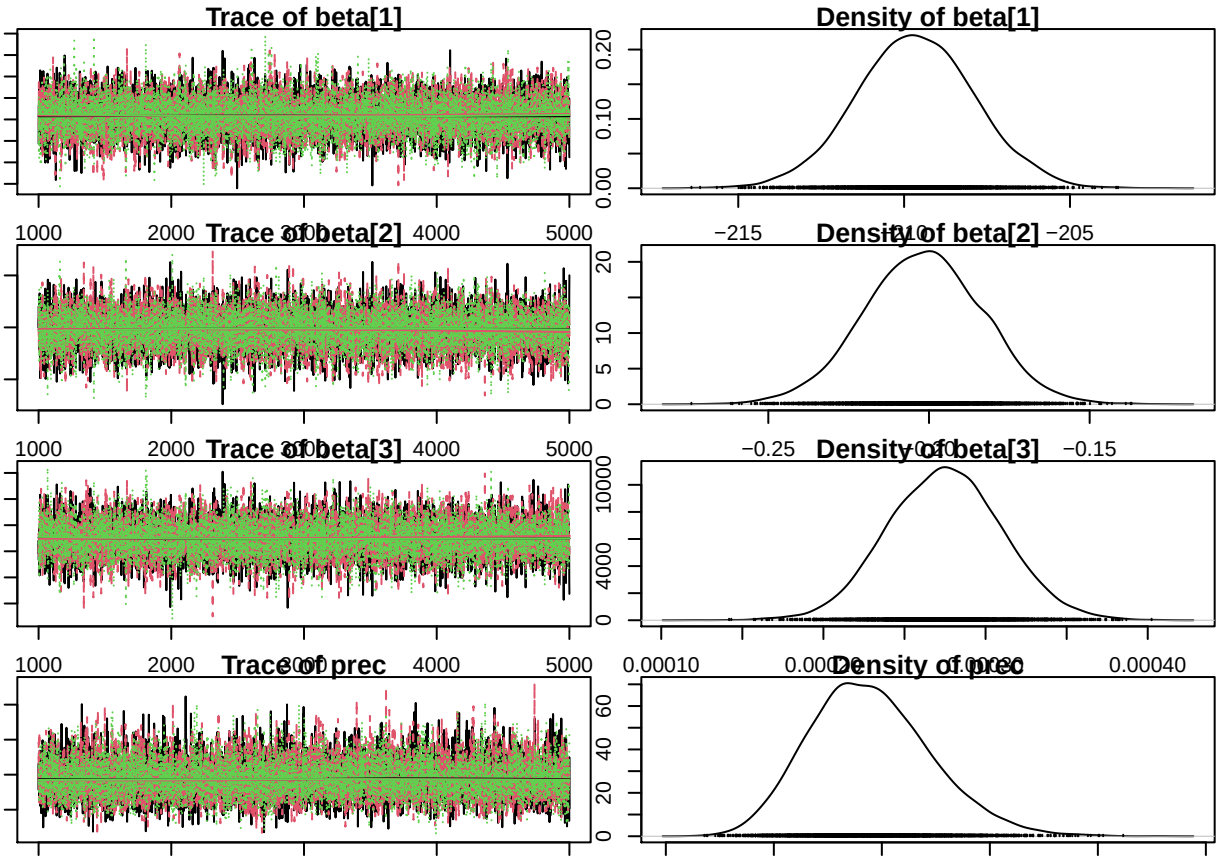
```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 57
##    Unobserved stochastic nodes: 2
##    Total graph size: 343
##
## Initializing model
```

```r
#sample from model
quad.out<- coda.samples (model = quad.model,
                         variable.names = c("beta", "prec"),
                             n.iter = 5000)

#calc DIC for future model selection
quad.DIC<- dic.samples(model = quad.model, n.iter = 5000)


#burnin
burnin <- 1000
quad.burn <- window(quad.out, start = burnin)

#outputs
summary(quad.burn)
```

```
##
## Iterations = 1000:5000
## Thinning interval = 1
## Number of chains = 3
## Sample size per chain = 4001
```

```
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##               Mean        SD  Naive SE Time-series SE
## beta[1] -2.097e+02 1.793e+00 1.636e-02      1.636e-02
## beta[2] -2.018e-01 1.845e-02 1.684e-04      1.669e-04
## beta[3]  2.749e-04 3.526e-05 3.219e-07      3.198e-07
## prec     2.892e-02 5.614e-03 5.124e-05      5.505e-05
##
## 2. Quantiles for each variable:
##
##               2.5%        25%        50%        75%      97.5%
## beta[1] -2.132e+02 -2.109e+02 -2.097e+02 -2.085e+02 -2.061e+02
## beta[2] -2.386e-01 -2.142e-01 -2.017e-01 -1.894e-01 -1.661e-01
## beta[3]  2.067e-04  2.508e-04  2.748e-04  2.982e-04  3.442e-04
## prec     1.905e-02  2.498e-02  2.859e-02  3.247e-02  4.100e-02
```

```
par(mar=c(1,1,1,1))
plot(quad.burn)
```

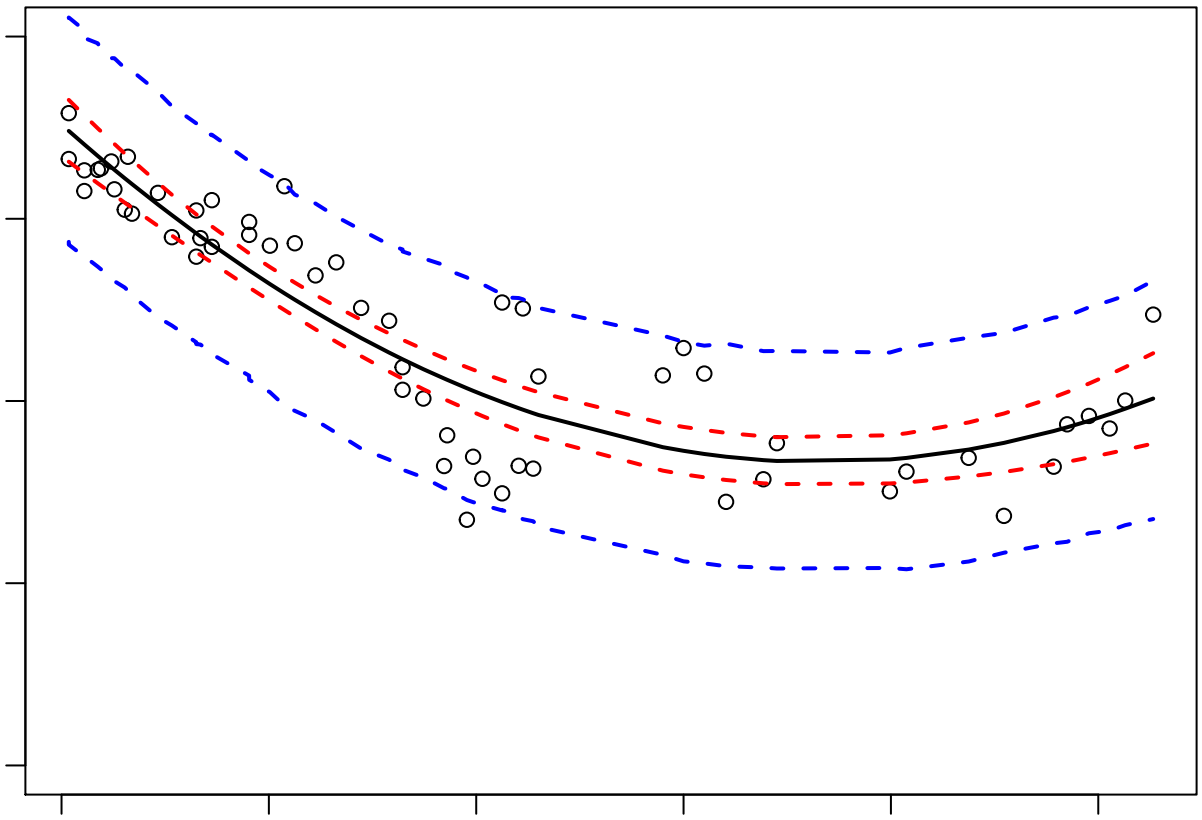



```
gelman.diag(quad.burn)
```

```
## Potential scale reduction factors:
##
```

```
##         Point est. Upper C.I.
## beta[1]          1       1.00
## beta[2]          1       1.01
## beta[3]          1       1.01
## prec             1       1.00
##
## Multivariate psrf
##
## 1
```

```r
##predictive and credible intervals
#initial values
quad.mat <- as.matrix(quad.burn)
nsamp <- 10000
samp <- sample.int(nrow(quad.mat),nsamp)
xpred <- dat$depth
npred <- length(xpred)
ycred <- matrix(NA,nrow=nsamp,ncol=npred)
ypred <- matrix(NA,nrow=nsamp,ncol=npred)

##calculate interval values
#loop through and fill each row
for(g in seq_len(nsamp)){
  #sampled parameters
  theta = quad.mat[samp[g],]

  #credible interval
  ycred[g,] <- theta["beta[1]"] + theta["beta[2]"]*xpred + theta["beta[3]"]*xpred^2

  #predictive interval
  ypred[g,] <- rnorm(npred,ycred[g,],1/sqrt(theta["prec"]))
}

##intervals themselves
# credible interval and median
ci.quad <- apply(ycred,2,quantile,c(0.025,0.5,0.975))
# prediction interval
pi.quad <- apply(ypred,2,quantile,c(0.025,0.975))

#plot it
plot(dat$depth, dat$deuturium, ylim= c(-280, -200))
lines(xpred, ci.quad[2,], type="l", col = 1, lwd= 2)    #median model
lines(xpred, ci.quad[1,], type="l", lty = 2, col = "red", lwd = 2)  #CI
lines(xpred, ci.quad[3,], type="l", lty = 2, col = "red", lwd = 2)
lines(xpred, pi.quad[1,], type="l", lty = 2, col = "blue", lwd = 2) #PI
lines(xpred, pi.quad[2,], type="l", lty = 2, col = "blue", lwd = 2)
```



Now lets fit a cubic (3rd order regression)

```
#bayesian fitting of polynomial (3rd order) regression (linear with respect to parameters)
cubic_regression<- "
model{

  ##priors
  #betas
  beta ~ dmnorm(b0,Vb)
  #precision
  prec ~ dgamma(s1,s2)

  ##process and data model
  for(i in 1:n){
    # process model
      mu[i] <- beta[1] + beta[2]*x[i,1] + beta[3]*x[i,2] + beta[4]*x[i,3]
      # data model
      Isotope[i]  ~ dnorm(mu[i],prec)
  }
}
"

#specify data values
data.cubic<- list(Isotope = dat$deuturium, x = cbind(dat$depth, dat$depth^2, dat$depth^3), n = nrow(dat)

##priors
```

```r
# regression beta means
data.cubic$b0 <- as.vector(c(0,0,0,0))
# regression beta precisions
data.cubic$Vb <- solve(diag(10000,4))
# uninformative error priors
data.cubic$s1 <- .1
data.cubic$s2 <- .1

#initial conditions
inits.cubic<- list()
for (i in 1:nchain){
  inits.cubic[[i]]<- list(beta = rnorm(4,0,5), prec = runif(1,1/1000,1/100))
}

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

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 57
##    Unobserved stochastic nodes: 2
##    Total graph size: 459
##
## Initializing model
```

```r
cubic.out<- coda.samples (model = cubic.model,
                          variable.names = c("beta", "prec"), n.iter = 5000)
cubic.DIC<- dic.samples(model = cubic.model, n.iter = 5000)


#burnin
burnin <- 1000
cubic.burn <- window(cubic.out, start = burnin)

#outputs
summary(cubic.burn)
```

```
##
## Iterations = 1000:5000
## Thinning interval = 1
## Number of chains = 3
## Sample size per chain = 4001
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##               Mean        SD  Naive SE Time-series SE
## beta[1] -2.111e+02 2.202e+00 2.010e-02      2.005e-02
```

```
## beta[2] -1.571e-01 4.409e-02 4.025e-04      3.993e-04
## beta[3]  3.224e-05 2.197e-04 2.006e-06      1.964e-06
## beta[4]  3.244e-07 2.895e-07 2.642e-09      2.628e-09
## prec     2.916e-02 5.642e-03 5.150e-05      5.393e-05
##
## 2. Quantiles for each variable:
##
##               2.5%        25%        50%        75%      97.5%
## beta[1] -2.155e+02 -2.126e+02 -2.112e+02 -2.097e+02 -2.067e+02
## beta[2] -2.436e-01 -1.867e-01 -1.569e-01 -1.280e-01 -6.883e-02
## beta[3] -4.036e-04 -1.135e-04  3.375e-05  1.800e-04  4.618e-04
## beta[4] -2.355e-07  1.312e-07  3.252e-07  5.168e-07  8.993e-07
## prec     1.937e-02  2.512e-02  2.886e-02  3.267e-02  4.114e-02
```

```
effectiveSize(cubic.burn)
```

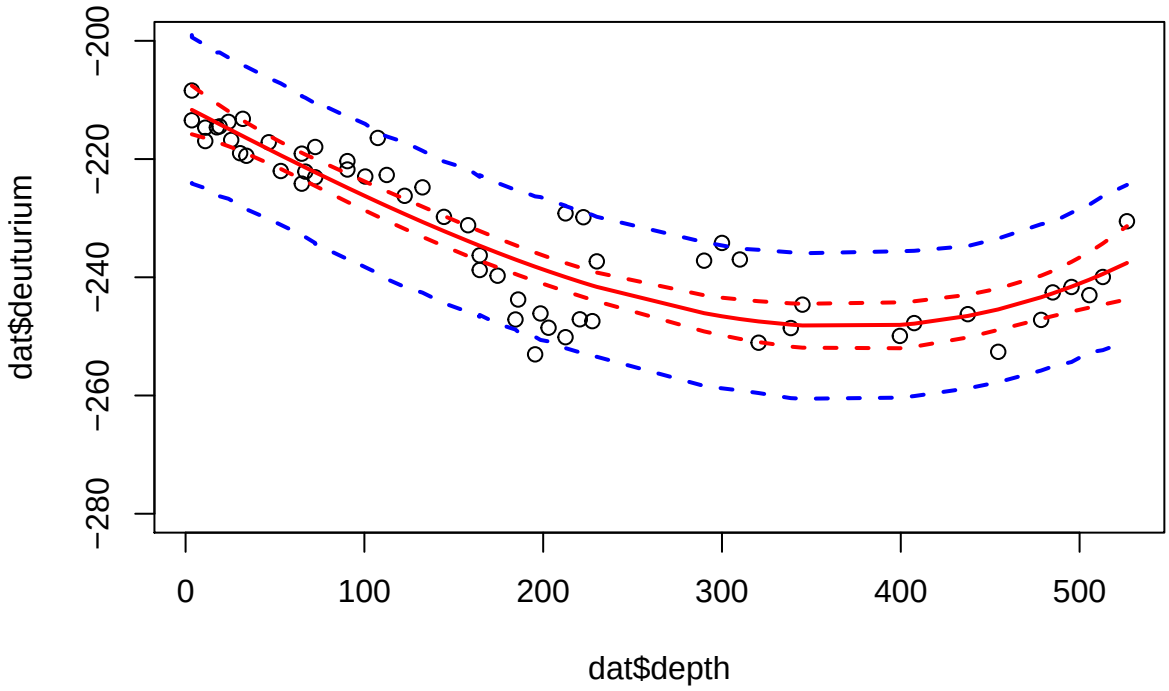
```
##  beta[1]  beta[2]  beta[3]  beta[4]     prec
## 12095.29 12216.77 12527.04 12136.51 10994.64
```

```
gelman.diag(cubic.burn)
```

```
## Potential scale reduction factors:
##
##         Point est. Upper C.I.
## beta[1]          1          1
## beta[2]          1          1
## beta[3]          1          1
## beta[4]          1          1
## prec             1          1
##
## Multivariate psrf
##
## 1
```

```
#predictive and credible intervals
#initial values
cubic.mat <- as.matrix(cubic.burn)
nsamp <- 10000
samp <- sample.int(nrow(cubic.mat),nsamp)
xpred <- dat$depth
npred <- length(xpred)
ycred <- matrix(NA,nrow=nsamp,ncol=npred)
ypred <- matrix(NA,nrow=nsamp,ncol=npred)

#calculate intervals
for(g in seq_len(nsamp)){
  theta = cubic.mat[samp[g],]
  ycred[g,] <- theta["beta[1]"] + theta["beta[2]"]*xpred + theta["beta[3]"]*xpred^2 + theta["beta[4]"]*x
  ypred[g,] <- rnorm(npred,ycred[g,],1/sqrt(theta["prec"]))
}
ci.cubic <- apply(ycred,2,quantile,c(0.025,0.5,0.975))  ## credible interval and median
pi.cubic <- apply(ypred,2,quantile,c(0.025,0.975))        ## prediction interval
```

```r
#plot it
plot(dat$depth, dat$deuturium, ylim= c(-280, -200))
lines(xpred, ci.cubic[2,], type="l", col = "red", lwd= 2) #median model
lines(xpred, ci.cubic[1,], type="l", lty = 2, col = "red", lwd = 2)   #CI
lines(xpred, ci.cubic[3,], type="l", lty = 2, col = "red", lwd = 2)
lines(xpred, pi.cubic[1,], type="l", lty = 2, col = "blue", lwd = 2)  #PI
lines(xpred, pi.cubic[2,], type="l", lty = 2, col = "blue", lwd = 2)
```



And a quartic. JAGS crashes when trying to sample this model, the MCMC has been hardcoded in the next chunk

```r
##quad reg model
quartic_regression<- "
model{

  #priors
  beta ~ dmnorm(b0,Vb)      ## prior regression params
  prec ~ dgamma(s1,s2)  ## prior precision

  #process and data model
  for(i in 1:n){
      mu[i] <- beta[1] + beta[2]*x[i,1] + beta[3]*x[i,2] + beta[4]*x[i,3] +beta[5]*x[i,4]   ## process m
      Isotope[i]  ~ dnorm(mu[i],prec)       ## data model
  }
```

```
}
"

#specify data values
data.quartic<- list(Isotope = dat$deuturium, x = cbind(dat$depth, dat$depth^2, dat$depth^3, dat$depth^4)

##priors
# regression beta means
data.quartic$b0 <- as.vector(c(0,0,0,0,0))
# regression beta precisions
data.quartic$Vb <- solve(diag(10000,5))

#uninformative error prior
data.quartic$s1 <- .1
data.quartic$s2 <- .1

#initial conditions
inits.quartic<- list()
for (i in 1:nchain){
  inits.quartic[[i]]<- list(beta = rnorm(5,0,5), prec = runif(1,1/1000,1/100))
}

##fit and sample from model
#jags object
quartic.model   <- jags.model(file = textConnection(quartic_regression),
                              data = data.quartic,
                              inits = inits.quartic,
                              n.chains = nchain)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 57
##    Unobserved stochastic nodes: 2
##    Total graph size: 577
##
## Initializing model
```

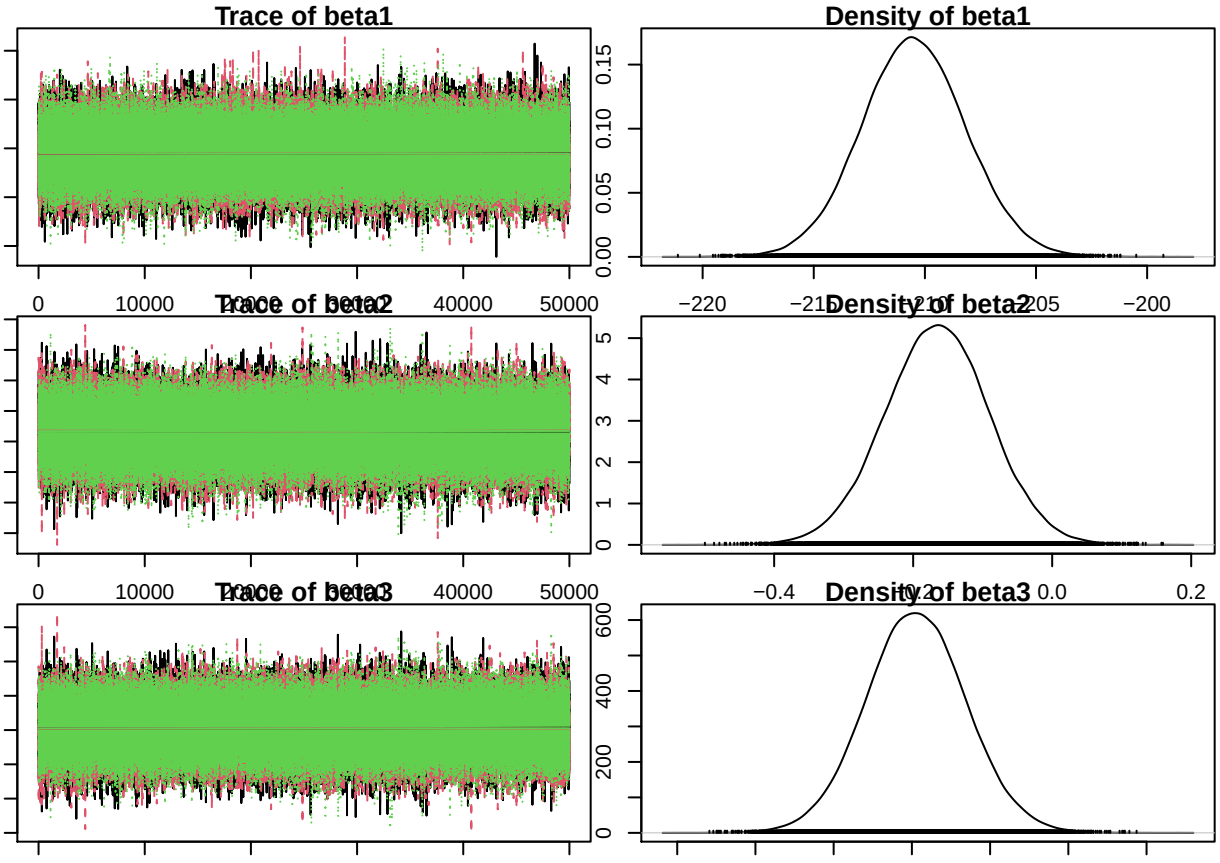
```
#sample from model will crash JAGS
#quartic.out<- coda.samples (model = quartic.model, variable.names = c("beta", "prec"), n.iter = 5000)
```

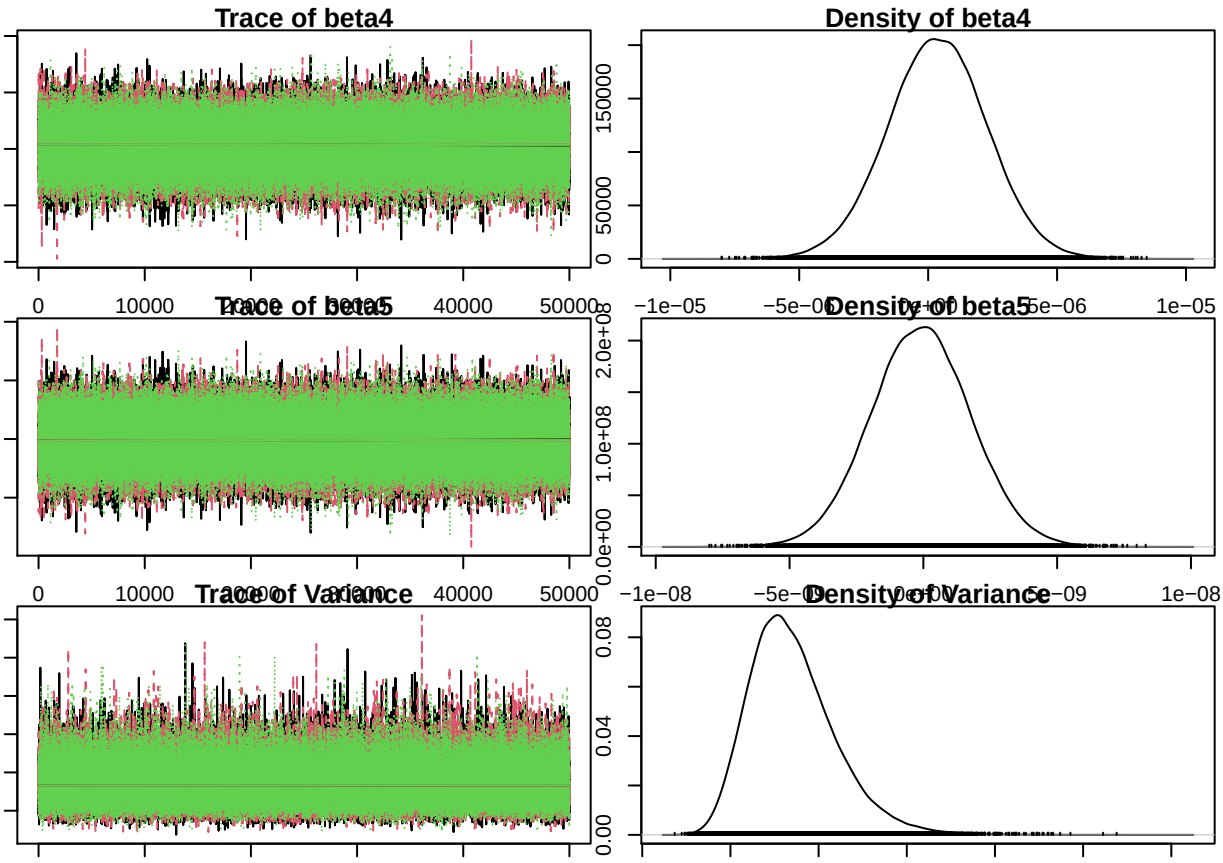
Now lets do the MCMC the hard way... Beta[5] appears to be non identifiable

```
##manual sampling of quartic reg
#define data
x<- cbind(rep(1,nrow(dat)),dat$depth, dat$depth^2, dat$depth^3, dat$depth^4)
y<- dat$deuturium

## specify priors
#number of itterations
n.g<- 50000
```

```r
#prior on the mean
bprior <- as.vector(c(0,0,0,0,0))
#inverse of prior variance
vinvert <- solve(diag(1000,5))
#inverse gamma prior on variance so 1/s1 from the precision prior
s1 <- 10
s2 <- 10
n <- nrow(dat)  #sample size, should this be my sample or prior sample size? right now its sample

##precompute frequently used quantities
XX <- t(x) %*% x
XY <- t(x) %*% y
VbB <- vinvert %*% bprior

##load libraries
library(coda)
library(mvtnorm)

## Gibbs loop
gibbs_loop<- function(itt){
  ## initial conditions
  sg <- 50
  sinv <- 1/sg

  #mcmc storage
  ngibbs<- length(itt)
  bgibbs <- matrix(0.0,nrow=ngibbs,ncol=5)    ## storage for beta
  sgibbs <- numeric(ngibbs)           ## storage for sigma2
  dgibbs <- numeric(ngibbs)         ## storage for deviance at each itteration

  for(g in itt){

  ## sample regression parameters
  bigV    <- solve(sinv*XX + vinvert, tol=1e-25)  ## Covariance matrix
  littlev <- sinv*XY + VbB
  b = t(rmvnorm(1,bigV %*% littlev,bigV))   ## Vu is the mean vector

  ## sample variance
  u1 <- s1 + n/2
  u2 <- s2 + 0.5*crossprod(y-x%*%b)
  sinv <- rgamma(1,u1,u2)
  sg <- 1/sinv

  ## storage
  bgibbs[g,] <- b  ## store the current value of beta vector
  sgibbs[g]  <- sg  ## store the current value of the variance

  ##deviance
  l_lik <- sum(dnorm(y, x%*%b, sg*1, log=T))   #log liklihood
  dgibbs[g] <- -2*l_lik                    #deviance


  }
```

```r
  #create mcmc outputs
  allgibbs<- cbind(bgibbs,sgibbs)
  colnames(allgibbs)<- c("beta1", "beta2", "beta3", "beta4", "beta5", "Variance")
  all_mcmc<- mcmc(allgibbs)

  return(list(all_mcmc, dgibbs))

}

#run mcmc, extract outputs and deviance
gibbs_list<- lapply(list(1:n.g, 1:n.g, 1:n.g), gibbs_loop)   #run three chains
gibbs_mcmc <- mcmc.list(list(gibbs_list[[1]][[1]], gibbs_list[[2]][[1]], gibbs_list[[3]][[1]]))    #mcm
gibbs_dev <- list(gibbs_list[[1]][[2]], gibbs_list[[2]][[2]], gibbs_list[[3]][[2]])     #deviance vecto

#calculate DIC
gibbs_sum<- summary(gibbs_mcmc)
beta_bar<- gibbs_sum$statistics[1:5,1]     #mean beta
var_bar<- gibbs_sum$statistics[6,1]        #mean variance
D_thetabar <- -2*sum(dnorm(y, x%*%beta_bar, var_bar*1, log=T))
Dtheta_bar <- mean(c(gibbs_dev[[1]], gibbs_dev[[2]], gibbs_dev[[3]]))
quartic.DIC <- 2*Dtheta_bar - D_thetabar

#outputs
par(mar=c(1,1,1,1))
plot(gibbs_mcmc)
```






```
effectiveSize(gibbs_mcmc)
```

```
##    beta1    beta2    beta3    beta4    beta5 Variance
## 149018.2 150000.0 150000.0 150000.0      0.0 125462.9
```

```
gelman.diag(gibbs_mcmc)
```

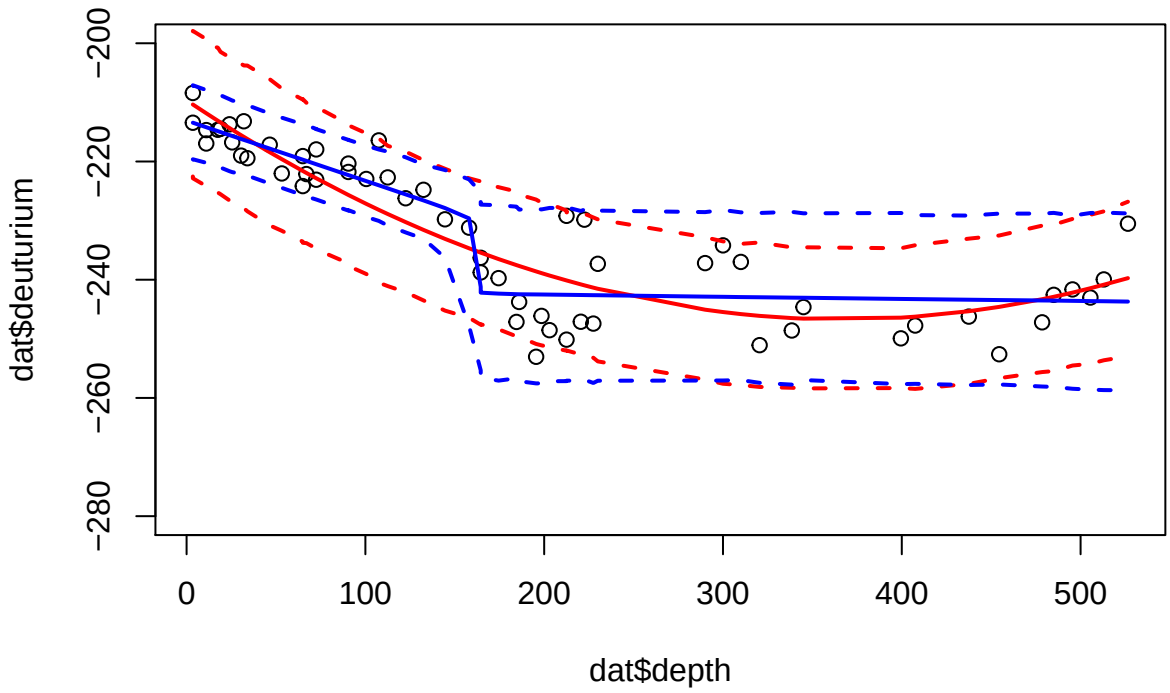
```
## Potential scale reduction factors:
##
##          Point est. Upper C.I.
## beta1             1          1
## beta2             1          1
## beta3             1          1
## beta4             1          1
## beta5             1          1
## Variance          1          1
##
## Multivariate psrf
##
## 1
```

```
#compare DICs
#quad.DIC
#cubic.DIC
quartic.DIC
```

```
## [1] 480.6342
```

lets compare the predictive intervals between best quad and best BP

```
#plot to compare model intervals between quad and breakpoint
plot(dat$depth, dat$deuturium, ylim=c(-280, -200))
lines(xpred, ci.quad[2,], type="l", col = "red", lwd= 2)
lines(xpred, pi.quad[1,], type="l", lty = 2, col = "red", lwd = 2)
lines(xpred, pi.quad[2,], type="l", lty = 2, col = "red", lwd = 2)
lines(xpred, ci.bp[2,], type="l", col = "blue", lwd= 2)  #median model
lines(xpred, pi.bp[1,], type="l", lty = 2, col = "blue", lwd = 2)   #PI
lines(xpred, pi.bp[2,], type="l", lty = 2, col = "blue", lwd = 2)
```




```
#hahahahahaha looks like shit with too many lines
```