Niyati Srivastava ID:200491035
Applied Bayesian Analysis ST540 Exam 2

Quantitative Data Analysis

1a. The data comes from the Cretaceous Climates research paper and aims at estimating the mean annual temperature (Y_{ik}) across 9 ages of the species. This is regressed against the Paleo Latitude covariate(X) with a nonlinear quadratic relation $g_k(x)$ $k \in 1,2,3,4,5,6,7,8,9$ as evidenced by data:

$$Y_{ik} = \beta_{0k} + (\beta_{1k}) * X_{ik} + \beta_{2k} * \left(X_{ik}{}^2\right)$$

In this analysis, we choose 3 models for comparison and find the best fit:

1. Random slopes with uninformed priors: $Y_{ik} \sim Normal(g_k(x), \sigma_k^2)$ with uninformed priors $\beta_{0k}, \beta_{1k}, \beta_{2k} \sim Normal(0,0.001) and\ \sigma_k^2 \sim InvGamma(0.1,0.1)$
2. Random slopes with informed priors and heteroscedastic variance: Since the variance across the latitude increases and then decreases, testing a nonlinear variance function was reasonable. $Y_{ik} \sim Normal(g_k(x), \sigma_k^2(X_i))$, $log\left(\sigma_k^2(X_i)\right) = \alpha_{0k} + \alpha_{1k} * X_{ik} + \alpha_{2k} * (X_{2k})^2$, $\beta_{1k}, \beta_{2k} \sim Normal(0, sigmaB), \alpha_{1k}, \alpha_{2k} \sim Normal(0, sigmaA),\ \beta_{0k}, \alpha_{0k} \sim Normal(0,0.001)$ $sigmaA, sigmaB \sim InvGamma(0.1,0.1)$
3. Random Slopes with LASSO Priors: $Y_{ik} \sim Normal(g_k(x), \sigma 1^2)$, $\beta_{0k}, \beta_{1k}, \beta_{2k} \sim DoubleExponential(0, (\sigma 1 * \sigma 2)^2)$ with $\sigma 1, \sigma 2 \sim InvGamma(0.1,0.1)$

1b. Using model comparison methods such as DIC and WAIC, Model 2 was chosen as the best fit with the lowest DIC and WAIC values of 45986 in comparison to others which were more than 47000. Hence, further analysis is completed using Model2

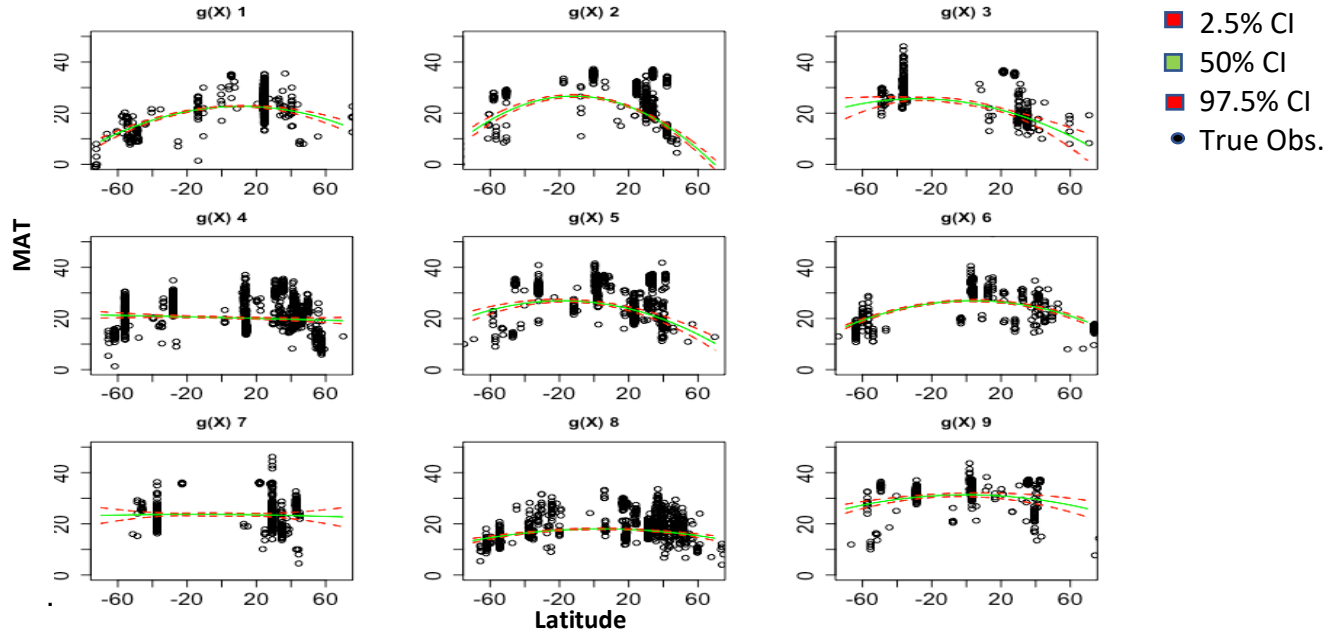
1c. The Effective Sample Size for parameters in Model2 is high (>1000) and the Gelman statistic for 2 chains has a score of 1 which is less than 1.1 hence, the chains have converged

1d. The predictive checks were completed using 3 statistics: mean, range, and standard deviation. The simulated plots contained true values. The Bayesian P values for the statistic were:

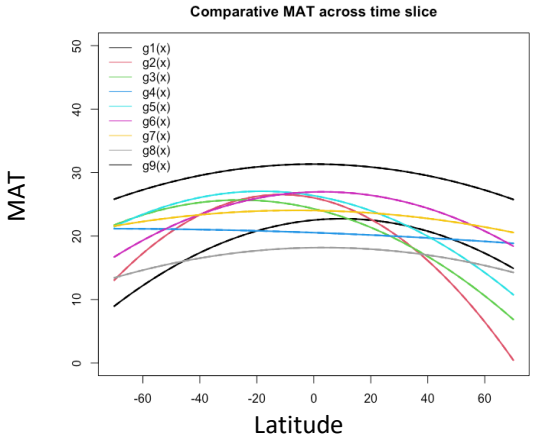
Model2	Mean	Range	Standard Deviation
Bayesian P Value	0.35016	0.9989	09952

Since, most p values are close to 1 it implies the models suggest a lack of fit. An alternate model can be using spline functions to capture the nonlinear trend of change in MAT across time slices as currently the trend is loosely captured by the quadratic function

1e.



1f. Clearly, the MAT changes across time slices. This is also captured in the credible intervals across slices some of which do not overlap

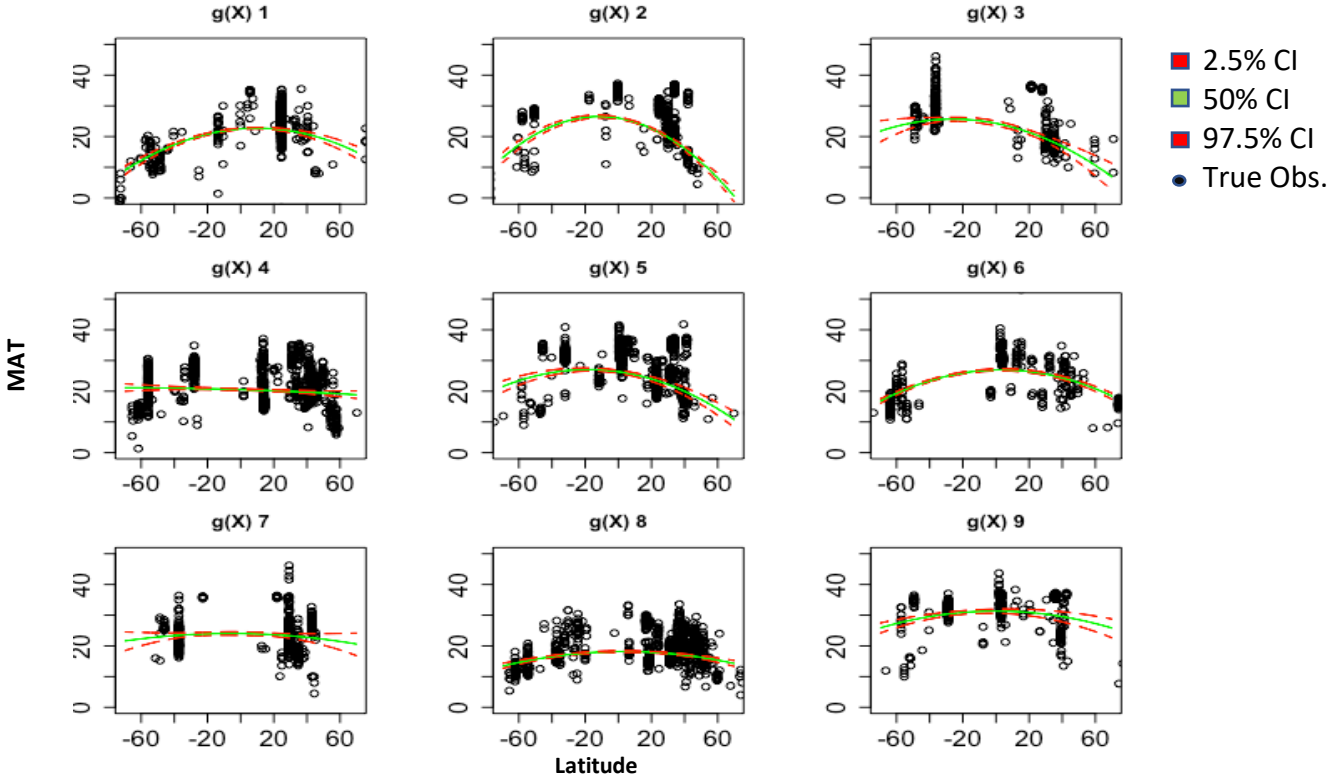


Mixed Data Analysis

2a. The data is divided into 2 sets: Quantitative and Interval. Since intervals only provide a region where the MAT can be, a probability likelihood is used to estimate the parameters. Model 2 is used for both sets with following assumptions:

Quantitative	Interval
$Y_{ik} \sim Normal(g_k(x), \sigma_k^2(X_i))$, $log\left(\sigma_k^2(X_i)\right) = \alpha_{0k} + \alpha_{1k} * X_{ik} + \alpha_{2k} * (X_{2k})^2$, $\beta_{1k}, \beta_{2k} \sim Normal(0, sigmaB)$ $\alpha_{1k}, \alpha_{2k} \sim Normal(0, sigmaA)$ $\beta_{0k}, \alpha_{0k} \sim Normal(0,0.001)$ $sigmaA, sigmaB \sim InvGamma(0.1,0.1)$	$P(u_j<Y_{jk}<l_j)=F(u_j, g_k(x), \sigma_k^2(X_j)) - F(l_j, g_k(x), \sigma_k^2(X_j))$ F(X) is normally distributed CDF function with mean $g_k(x), and\ variance\ \sigma_k^2(X_j)$. U_j: Max Temperature L_j: Min Temperature Priors for $g_k(x), \sigma_k^2(X_j)\ remain\ same\ as\ quantitative$,

2b.



2c. There is no visual change in the intervals from the plots. However, some β coefficients which were insignificant in part 1 were now significant. The mean value of coefficients increases. The credible intervals slightly widen. Additionally, the Bayesian p-values for mean, range, and standard deviation also showed minor improvement implying a slightly better fit. Though, the model does not completely capture the MAT trend.

APPENDIX and CODE

1. Summary from Model 2 in part 1
   Iterations = 11001:36000
   Thinning interval = 1
   Number of chains = 2
   Sample size per chain = 25000

   1. Empirical mean and standard deviation for each variable,
      plus standard error of the mean:

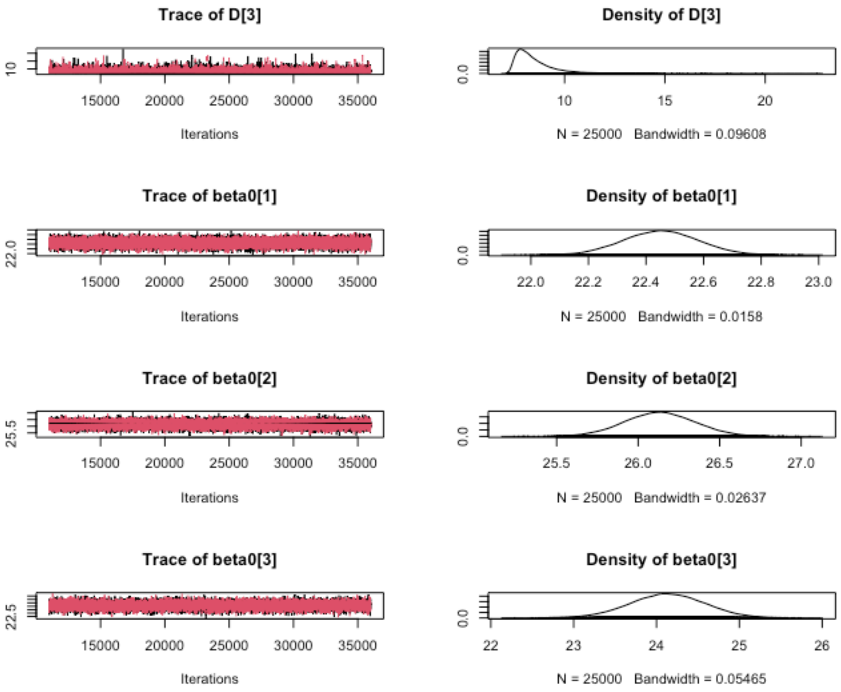
	Mean	SD	Naive SE	Time-series SE
D[1]	23.97144	0.09231	0.0004128	0.0005224
D[2]	409.38652	191.16767	0.8549278	1.3224559
D[3]	8.48568	1.01472	0.0045380	0.0074617
beta0[1]	22.45131	0.12973	0.0005802	0.0008358
beta0[2]	26.13260	0.21657	0.0009686	0.0015308
beta0[3]	24.12971	0.44885	0.0020073	0.0049770
beta0[4]	20.39302	0.16520	0.0007388	0.0007547
beta0[5]	26.25404	0.27097	0.0012118	0.0035089
beta0[6]	26.95716	0.19147	0.0008563	0.0009352
beta0[7]	23.67370	0.31920	0.0014275	0.0023760
beta0[8]	18.03218	0.13359	0.0005974	0.0007366
beta0[9]	31.33965	0.35646	0.0015941	0.0104337
beta1[1]	1.87400	0.23064	0.0010315	0.0017057
beta1[2]	-3.83166	0.26854	0.0012010	0.0019321
beta1[3]	-4.28680	0.32056	0.0014336	0.0035416
beta1[4]	-0.63682	0.15987	0.0007150	0.0007540
beta1[5]	-3.17566	0.22850	0.0010219	0.0022010
beta1[6]	0.54527	0.15147	0.0006774	0.0007749
beta1[7]	-0.18014	0.22362	0.0010001	0.0013178
beta1[8]	0.26174	0.11920	0.0005331	0.0005932
beta1[9]	-0.01449	0.24878	0.0011126	0.0068721
beta2[1]	-3.43585	0.23635	0.0010570	0.0017774
beta2[2]	-6.62836	0.28695	0.0012833	0.0015471
beta2[3]	-3.09066	0.90369	0.0040414	0.0138612
beta2[4]	-0.05123	0.18987	0.0008491	0.0013799
beta2[5]	-3.52628	0.28045	0.0012542	0.0034547
beta2[6]	-3.10623	0.10670	0.0004772	0.0005924
beta2[7]	-0.23876	0.63702	0.0028489	0.0054983
beta2[8]	-1.38526	0.14497	0.0006483	0.0011400
beta2[9]	-1.85171	0.29949	0.0013394	0.0083372

   2. Quantiles for each variable:

	2.5%	25%	50%	75%	97.5%
D[1]	23.79015	23.9093	23.97127	24.03299	24.1516
D[2]	146.11621	272.4092	373.56212	507.58945	874.1595
D[3]	7.40334	7.8019	8.21126	8.85932	11.1480
beta0[1]	22.19872	22.3636	22.45154	22.53897	22.7046
beta0[2]	25.70785	25.9859	26.13214	26.27825	26.5577

beta0[3]	23.24011	23.8286	24.13058	24.43437	25.0020
beta0[4]	20.07158	20.2810	20.39275	20.50379	20.7176
beta0[5]	25.71568	26.0736	26.25455	26.43610	26.7828
beta0[6]	26.57993	26.8289	26.95643	27.08659	27.3310
beta0[7]	23.04803	23.4601	23.67511	23.88839	24.2970
beta0[8]	17.77015	17.9416	18.03252	18.12307	18.2941
beta0[9]	30.64262	31.1018	31.33705	31.57837	32.0448
beta1[1]	1.42060	1.7191	1.87460	2.03107	2.3218
beta1[2]	-4.35454	-4.0132	-3.83123	-3.65095	-3.3027
beta1[3]	-4.93591	-4.4990	-4.27859	-4.06470	-3.6841
beta1[4]	-0.95219	-0.7451	-0.63617	-0.52914	-0.3260
beta1[5]	-3.62647	-3.3307	-3.17483	-3.02104	-2.7286
beta1[6]	0.24474	0.4447	0.54695	0.64735	0.8401
beta1[7]	-0.62469	-0.3292	-0.17930	-0.02779	0.2559
beta1[8]	0.02804	0.1820	0.26200	0.34150	0.4942
beta1[9]	-0.50385	-0.1820	-0.01486	0.15267	0.4778
beta2[1]	-3.91057	-3.5926	-3.43194	-3.27396	-2.9872
beta2[2]	-7.19312	-6.8204	-6.62903	-6.43640	-6.0675
beta2[3]	-4.99498	-3.6887	-3.02819	-2.43227	-1.5086
beta2[4]	-0.41515	-0.1812	-0.05201	0.07617	0.3258
beta2[5]	-4.07400	-3.7154	-3.52519	-3.33775	-2.9733
beta2[6]	-3.32037	-3.1769	-3.10423	-3.03323	-2.9035
beta2[7]	-1.51693	-0.6604	-0.23130	0.18999	0.9954
beta2[8]	-1.66203	-1.4841	-1.38809	-1.28932	-1.0927
beta2[9]	-2.43680	-2.0534	-1.85491	-1.65175	-1.2552

Convergence plots for some parameters:



CODE

2023-04-16

```
#ALL MODEL FIT IN JAGS
df=read.csv("paleo_dat.csv")
summary(df)
df_quant=df[!is.na(df$Temperature.C),]
df_quant=df_quant %>% group_by(Paleocoordinate.Age) %>%
  dplyr::mutate(ID = cur_group_id())
df_quant=df_quant[order(df_quant$ID),]
df_quant

#Analysing Data
plot(subset(df_quant,ID=='2')$Paleo.Lat,subset(df_quant,ID=='2')$Temperature.C)
plot(subset(df_quant,ID=='3')$Paleo.Lat,subset(df_quant,ID=='3')$Temperature.C)
plot(subset(df_quant,ID=='1')$Paleo.Lat,subset(df_quant,ID=='1')$Temperature.C)


#Creating Y and X
Y_quant=cbind(df_quant$ID,df_quant$Temperature.C)
colnames(Y_quant)=c("ID","Y")
X_lat=scale(df_quant$Paleo.Lat,TRUE,TRUE)
X2_lat=scale((df_quant$Paleo.Lat)^2,TRUE,TRUE)
X_quant_scale=cbind(df_quant$ID, X_lat,X2_lat)
X_quant=cbind(df_quant$ID,df_quant$Paleo.Lat,(df_quant$Paleo.Lat)^2)
colnames(X_quant)=c("ID","X","X^2")



library(rjags)
#Varying slopes with normal uninformed priors
#creating model specification
model_string1=textConnection("model{

                             #Likelihood
                              for (i in 1:n){
                              Y[i,2]~dnorm(mn[i],taue[gk[i]])
                              mn[i]=beta0[gk[i]] + beta1[gk[i]]*X[i,2]
                                + beta2[gk[i]]*X[i,3]
                              }


                             #Priors
                             for (j in 1:9){
                             beta0[j]~dnorm(0,0.001)
                             beta1[j]~dnorm(0,0.001)
                             beta2[j]~dnorm(0,0.001)
```

```
                          taue[j]~dgamma(0.1,0.1)
                          sigmae[j]=1/sqrt(taue[j])
                          }


                          # WAIC calculations
                         for(i in 1:n){
                             like[i]    <- dnorm(Y[i,2],mu[i],taue[gk[i]])
                             mu[i] <- beta0[gk[i]] + beta1[gk[i]]*X[i,2] + beta2[gk[i]]*X[i,3]
                         }

                         #Diagnostic Checks
                         for (i in 1:n){
                         Y2[i]~dnorm(mn[i],taue[gk[i]])
                         }

                           D[1]<- mean(Y2[])
                           D[2]<- max(Y2[])-min(Y2[])
                           D[3]<-sd(Y2[])



}")


#compiling model
model=jags.model(model_string1,data=list(n=nrow(X_quant),
                                         Y=Y_quant,X=X_quant_scale,
                                         gk=df_quant$ID),n.chain=2,quiet=TRUE)
#burn in for 5000 samples
update(model, 5000, progress.bar="none")
params  <- c("beta0","beta1","beta2","sigmae",'D')

#Generate post burn out samples
samples1 <- coda.samples(model,
                        variable.names=params,
                        n.iter=10000, progress.bar="none")

 #summarise ouputs
summary(samples1)

#Calculate ESS
effectiveSize(samples1)
#Calculate Gelman Diagnostic
gelman.diag(samples1)

# Compute DIC
dic1   <- dic.samples(model,n.iter=10000,progress.bar="none")

# Compute WAIC
waic1   <- coda.samples(model,
                        variable.names=c("like"),
                        n.iter=10000, progress.bar="none")
```

```
like1   <- waic1[[1]]
fbar1   <- colMeans(like1)
P1      <- sum(apply(log(like1),2,var))
WAIC1   <- -2*sum(log(fbar1))+2*P1



#Varying slope with normal informed priors and heteroscadastic variance
#creating model specification
model_string2=textConnection("model{

                            #Likelihood
                             for (i in 1:n){
                             Y[i,2]~dnorm(mn[i],taue[i])
                             taue[i]=1/sig2[i]
                             log(sig2[i])=alpha0[gk[i]] +
                               alpha1[gk[i]]*X[i,2]+alpha2[gk[i]]*X[i,3]

                             mn[i]=beta0[gk[i]] + beta1[gk[i]]*X[i,2] +
                               beta2[gk[i]]*X[i,3]
                             }


                            #Priors
                            for (j in 1:9){
                            beta0[j]~dnorm(0,0.001)
                            beta1[j]~dnorm(0,taub)
                            beta2[j]~dnorm(0,taub)
                            alpha0[j]~dnorm(0,0.001)
                            alpha1[j]~dnorm(0,taua)
                            alpha2[j]~dnorm(0,taua)
                            }

                            taub~dgamma(0.1,0.1)
                            taua~dgamma(0.1,0.1)


                             # WAIC calculations
                           for(i in 1:n){
                               like[i]    <- dnorm(Y[i,2],mn[i],taue[i])
                           }

                             #Diagnostic Checks
                           for (i in 1:n){
                           Y2[i]~dnorm(mn[i],taue[i])
                           }

                             D[1]<- mean(Y2[])
                             D[2]<- max(Y2[])-min(Y2[])
                             D[3]<-sd(Y2[])

}")
```

```
#compiling model
model2=jags.model(model_string2,data=list(n=nrow(X_quant_scale),Y=Y_quant,
                                          X=X_quant_scale,gk=df_quant$ID),n.chain=2,quiet=TRUE)

#burn in for 5000 samples
update(model2, 10000, progress.bar="none")
params  <- c("beta0","beta1","beta2","D")

#Generate post burn out samples
samples2 <- coda.samples(model2,
                         variable.names=params,
                         n.iter=25000, progress.bar="none")

#summarise ouputs
summary(samples2)


#Calculate ESS
effectiveSize(samples2)
#Calculate Gelman Diagnostic
gelman.diag(samples2)

# Compute DIC
dic2  <- dic.samples(model2,n.iter=10000,progress.bar="none")

# Compute WAIC
waic2   <- coda.samples(model2,
                        variable.names=c("like"),
                        n.iter=10000, progress.bar="none")
like2   <- waic2[[1]]
fbar2   <- colMeans(like2)
P2      <- sum(apply(log(like2),2,var))
WAIC2   <- -2*sum(log(fbar2))+2*P2

#varying regression coeffcient with LASSO priors and constant variance
model_string3=textConnection("model{

                            #Likelihood
                             for (i in 1:n){
                             Y[i,2]~dnorm(mn[i],taue)
                             mn[i]=beta0 + beta1[gk[i]]*X[i,2] +
                               beta2[gk[i]]*X[i,3]
                             }


                            #Priors
                            beta0~dnorm(0,0.001)
                            for (j in 1:9){
                            beta1[j]~ddexp(0,taue*taub)
                            beta2[j]~ddexp(0,taue*taub)
                            }
                            taue~dgamma(0.1,0.1)
                            taub~dgamma(0.1,0.1)
```

```
                         # WAIC calculations
                        for(i in 1:n){
                            like[i]    <- dnorm(Y[i,2],mn[i],taue)
                        }



}")

#compiling model
model3=jags.model(model_string3,data=list(n=nrow(X_quant_scale),Y=Y_quant,
                                          X=X_quant_scale,
                                          gk=df_quant$ID),n.chain=2,quiet=TRUE)
#burn in for 5000 samples
update(model3, 5000, progress.bar="none")
params  <- c("beta0","beta1","beta2")

#Generate post burn out samples
samples3 <- coda.samples(model3,
                         variable.names=params,
                         n.iter=10000, progress.bar="none")

#summarise ouputs
summary(samples3)


#Calculate ESS
effectiveSize(samples3)
#Calculate Gelman Diagnostic
gelman.diag(samples3)

# Compute DIC
dic3    <- dic.samples(model3,n.iter=10000,progress.bar="none")

# Compute WAIC
waic3   <- coda.samples(model3,
                        variable.names=c("like"),
                        n.iter=10000, progress.bar="none")
like3   <- waic3[[1]]
fbar3   <- colMeans(like3)
P3      <- sum(apply(log(like3),2,var))
WAIC3   <- -2*sum(log(fbar3))+2*P3


#Comparing DIC and WAIC for best model
dic1
dic2
dic3

WAIC1
WAIC2
WAIC3
```

```
#Analysis using MODEL2
#Second Model has lower DIC and WAIC so that is the better model for the fit
D=samples2[[1]][,1:3]


#Diagnostics Checks and Bayesian P value
bayesian_p_val=rep(0,3)
D0   <- c(   mean(Y_quant[,2]),   max(Y_quant[,2])-min(Y_quant[,2]),
             sd(Y_quant[,2]))
Dnames <- c("Mean Y", "Range Y","SD Y")

par(mfrow=c(3,2))
for(j in 1:3){
  plot(density(D[,j]),xlim=range(c(D0[j],D[,j])),
       xlab="D",ylab="Posterior probability",
       main=Dnames[j])
  abline(v=D0[j],col=2,lwd=2)
  legend("topright",c("Model 2","True Data"),lty=1,col=1:2,bty="n")

  bayesian_p_val[j] <- mean(D[,j]>D0[j])
}

names(bayesian_p_val)=Dnames
bayesian_p_val


#Plotting curves with uncertainty
sum=summary(samples2)
samp=samples2[[1]]
beta0=samp[,4:12]
beta1=samp[,13:21]
beta2=samp[,22:30]



X_quant=data.frame(X_quant)
Y_quant=data.frame(Y_quant)
pred=seq(-70,70,length=1133)
pred_scale=scale(pred,TRUE,TRUE)

par(mar=c(2,2,3,4),mfrow=c(3,3))
for (i in 1:9){
  X=X_quant[X_quant$ID==i,]
  Y=Y_quant[Y_quant$ID==i,]

  # Plot the posterior of the mean alpha1+age[j]*alpha2

  fit <- NULL
  for(j in 1:length(pred)){
    fit <- cbind(fit,beta0[,i]+pred_scale[j]*beta1[,i]+
                   (pred_scale[j]^2)*beta2[,i])
  }
  q  <- apply(fit,2,quantile,c(0.025,0.5,0.975))
```

```
  plot(X[,2],Y[,2],xlab="PaleoLatitude",ylab="Temperature",
       cex.lab=1.5,cex.axis=1.5,xlim=c(-70,70),ylim=c(0,50),
       main=paste("g(X)",i))
  lines(pred,q[1,],lty=2,col='red')
  lines(pred,q[2,],lty=1,col='green')
  lines(pred,q[3,],lty=2,col='red')
}

#Plotting MAT
par(mfrow=c(1,1))
plot(NA,NA,xlab='Paleo Latitude',ylab='MAT',xlim=c(-70,70),ylim=c(0,50)
     ,main='Comparative MAT across time slice')

for (i in 1:9){
  X=X_quant[X_quant$ID==i,]
  Y=Y_quant[Y_quant$ID==i,]

  # Plot the posterior of the mean alpha1+age[j]*alpha2

  fit <- NULL
  for(j in 1:length(pred)){
    fit <- cbind(fit,mean(beta0[,i])+pred_scale[j]*mean(beta1[,i])+
                   (pred_scale[j]^2)*mean(beta2[,i]))
  }
  q=apply(fit,2,quantile,c(.025,0.5,0.75))
  lines(pred,q[1,],lty=2,col=i)
  lines(pred,q[2,],lty=1,col=i)
  lines(pred,q[3,],lty=2,col=i)
}
legend("topleft",
       c('g1(x)','g2(x)','g3(x)','g4(x)','g5(x)','g6(x)','g7(x)',
         'g8(x)','g9(x)')
       ,col=1:9,lty=1,bty='n')
```

```
 #PART 2
#Part2

#Use model 1 from before
#Create data
df=read.csv("paleo_dat.csv")
summary(df)
df_quant=df[!is.na(df$Temperature.C),]
df_quant=df_quant %>% group_by(Paleocoordinate.Age) %>%
  dplyr::mutate(ID = cur_group_id())
df_quant=df_quant[order(df_quant$ID),]
df_quant


#Creating Y and X
Y_quant=cbind(df_quant$ID,df_quant$Temperature.C)
colnames(Y_quant)=c("ID","Y")
X_lat=scale(df_quant$Paleo.Lat,TRUE,TRUE)
X2_lat=scale((df_quant$Paleo.Lat)^2,TRUE,TRUE)
```

```
X_quant_scale=cbind(df_quant$ID, X_lat,X2_lat)
X_quant=cbind(df_quant$ID,df_quant$Paleo.Lat,(df_quant$Paleo.Lat)^2)
colnames(X_quant)=c("ID","X","X^2")

df_mixed=df[!is.na(df$Min.Temp),]
df_mixed=df_mixed %>% group_by(Paleocoordinate.Age) %>%
  dplyr::mutate(ID = cur_group_id())
df_mixed=df_mixed[order(df_mixed$ID),]
Y_mixed=cbind(df_mixed$ID, df_mixed$Min.Temp,df_mixed$Max.Temp)
X_lat=scale(df_mixed$Paleo.Lat)
X2_lat=scale((df_mixed$Paleo.Lat)^2)
X_mixed_scale=cbind(df_mixed$ID,X_lat,X2_lat)
X_mixed=cbind(df_mixed$ID,df_mixed$Paleo.Lat,(df_mixed$Paleo.Lat)^2)
Y_mixed=data.frame(Y_mixed)
X_mixed=data.frame(X_mixed)
one=rep(1,2120)

model_string=textConnection("model{

                            #likelihood for quantitative observations
                            for (i in 1:n1){
                                Y1[i,2]~dnorm(mn[i],taue[i])
                                taue[i]=1/sig2[i]
                                log(sig2[i])=alpha0[gk[i]] +
                                alpha1[gk[i]]*X[i,2]+alpha2[gk[i]]*X[i,3]

                                mn[i]=beta0[gk[i]] + beta1[gk[i]]*X[i,2] + beta2[gk[i]]*X[i,3]
                            }

                            #likelihood for interval observations
                            for (j in 1:n2){
                                one[j]~dbern(p[j])
                                logit(p[j])=pnorm(Y2[j,3],m[j],tauf[j])-
                                  pnorm(Y2[j,2],m[j],taue[j])

                                tauf[j]=1/sig2f[j]
                                log(sig2f[j])=alpha0[gk[j]]+ alpha1[gk[j]]*X2[j,2]+alpha2[gk[j]]*X2[j,3

                                m[j]=beta0[gk2[j]] + beta1[gk2[j]]*X2[j,2] + beta2[gk2[j]]*X2[j,3]
                              }

                           #Priors
                           for (j in 1:9){
                            beta0[j]~dnorm(0,0.001)
                           beta1[j]~dnorm(0,taub)
                           beta2[j]~dnorm(0,taub)
                           alpha0[j]~dnorm(0,0.001)
                           alpha1[j]~dnorm(0,taua)
                           alpha2[j]~dnorm(0,taua)
                            }
                           taua~dgamma(0.1,0.1)
                           taub~dgamma(0.1,0.1)
```

```
                            #Diagnostic Checks
                            for (i in 1:n1){
                            Ypred[i]~dnorm(mn[i],taue[i])
                            }

                            D[1]=mean(Ypred[])
                            D[2]=max(Ypred[])-min(Ypred[])
                            D[3]=sd(Ypred[])


                        }")

#compiling model
model=jags.model(model_string,data=list(one=one,n1=nrow(Y_quant),
                                        n2=nrow(Y_mixed),
                                        Y1=Y_quant,Y2=Y_mixed,X=X_quant_scale,
                                        X2=X_mixed_scale,gk=df_quant$ID,
                                        gk2=df_mixed$ID),n.chain=2,quiet=TRUE)
#burn in for 5000 samples
update(model, 5000, progress.bar="none")
params  <- c("beta0","beta1","beta2","D")

#Generate post burn out samples
samples1 <- coda.samples(model,
                         variable.names=params,
                         n.iter=5000, progress.bar="none")

#summarise ouputs
summary(samples1)
D=samples1[[1]][,1:3]

#Bayesian P Values
bayesian_p_val=rep(0,3)
D0   <- c(   mean(Y_quant[,2]),   max(Y_quant[,2])-min(Y_quant[,2]),
             sd(Y_quant[,2]))
Dnames <- c("MeanY", "Range Y","SD Y")

par(mfrow=c(3,2))
for(j in 1:3){
  plot(density(D[,j]),xlim=range(c(D0[j],D[,j])),
       xlab="D",ylab="Posterior probability",
       main=Dnames[j])
  abline(v=D0[j],col=2,lwd=2)
  legend("topright",c("Model Output","True Data"),lty=1,col=1:2,bty="n")

  bayesian_p_val[j] <- mean(D[,j]>D0[j])
}

names(bayesian_p_val)=Dnames
bayesian_p_val

#plots
sum=summary(samples1)
```

```
samp=rbind(samples1[[1]],samples1[[2]])
beta0=samp[,1:9]
beta1=samp[,10:18]
beta2=samp[,19:27]

X_quant=data.frame(X_quant)
Y_quant=data.frame(Y_quant)
pred=seq(-70,70,length=1133)
pred_scale=scale(pred,TRUE,TRUE)

par(mar=c(2,2,3,4),mfrow=c(3,3))
for (i in 1:9){
  X=X_quant[X_quant$ID==i,]
  Y=Y_quant[Y_quant$ID==i,]

  # Plot the posterior of the mean alpha1+age[j]*alpha2

  fit <- NULL
  for(j in 1:length(pred)){
    fit <- cbind(fit,beta0[,i]+pred_scale[j]*beta1[,i]+
                 (pred_scale[j]^2)*beta2[,i])
  }
  q  <- apply(fit,2,quantile,c(0.025,0.5,0.975))
  plot(X[,2],Y[,2],xlab="PaleoLatitude",ylab="Temperature",
       cex.lab=1.5,cex.axis=1.5,xlim=c(-70,70),ylim=c(0,50),
       main=paste("g(X)",i))
  lines(pred,q[1,],lty=2,col='red')
  lines(pred,q[2,],lty=1,col='green')
  lines(pred,q[3,],lty=2,col='red')
}
```