Code

Dimitrios Ligas

Estimate Models M1,M2,M3 and compute DIC, WAIC

```
library(coda)
library(rjags)
#Load the data:
df=read.csv('paleo_data.csv',header=T,stringsAsFactors = F)
X=df$Paleo.Lat
Y=df$Temperature.C
time_slice=df$Paleocoordinate.Age
#Keep the Quantitative Data only:
miss=is.na(Y)
X=X[!miss]
Y=Y[!miss]
time_slice=time_slice[!miss]
# Find the Clusters of the data:
clusters=unique(df$Paleocoordinate.Age)
m=length(clusters)
n=length(Y)
cluster=matrix(NA,nrow=length(Y),ncol=1)
for (i in 1:m) {
  cluster[time_slice==clusters[i]]=i

}
# Plot the data:
for (i in 1:m) {
  plot(X[time_slice==clusters[i]],Y[time_slice==clusters[i]],pch=i,xlim=range(X),main=clusters[i],ylim=
  abline(h=mean(Y[time_slice==clusters[i]]),col="green",lwd=2)
  }

#Fit Model 1: Varying slopes as fixed effects
burn=10000
iter=25000
Y=as.vector(Y)
X=as.vector(X)
cluster=as.vector(cluster)
x=seq(min(X),max(X),length=100)
data=list(Y=Y,X=X,x=x,n=n,m=m,cluster=cluster)
model_string_1=textConnection("model{
 #Likelihood:
 for (i in 1:n){
 mu[i]=a1[cluster[i]]+a2[cluster[i]]*X[i]+a3[cluster[i]]*pow(X[i],2)
 Y[i]~dnorm(mu[i],tau1)
 #WAIC Computation
 like[i]=dnorm(Y[i],mu[i],tau1)
```

```
 }


 #Slopes as fixed effects:
 for (j in 1:m){
 a1[j]~dnorm(0,0.001)
 a2[j]~dnorm(0,0.001)
 a3[j]~dnorm(0,0.001)
 }


 #Priors:
 tau1~dgamma(0.1,0.1)

}")
parameters=c("a1","a2","a3","like")
model_1=jags.model(model_string_1,data=data,n.chains=2,quiet=TRUE)
update(model_1,burn,progress.bar="none")
samples_1=coda.samples(model_1,variable.names=parameters,n.iter=iter,thin=2,progress.bar="none")
post_samples_1=rbind(samples_1[[1]],samples_1[[2]])

#WAIC Computation:
like=post_samples_1[,28:7980]
fbar=colMeans(like)
Pw=sum(apply(log(like),2,var))
WAIC_1=2*sum(log(fbar))+2*Pw

#DIC:
dic1=dic.samples(model_1,n.iter=iter,progress.bar="none")

# Fit Model 2: Random Slopes with spatial heteroskedasticity:
d=diff(X,lag=1)
d=append(d,0)
d=abs(d)
model_string_2=textConnection("model{
  #Likelihood:
  for (i in 1:n){
  Y[i]~dnorm(mu[i],tau[i])
  mu[i]=a1[cluster[i]]+a2[cluster[i]]*X[i]+a3[cluster[i]]*pow(X[i],2)
  tau[i]=tao*exp(-d[i]/c)
  #WAIC
  like[i]=dnorm(Y[i],mu[i],tau[i])
  }


 #Random Slopes:
 for (j in 1:m){
 a1[j]~dnorm(beta[1],tau2)
 a2[j]~dnorm(beta[2],tau3)
 a3[j]~dnorm(beta[3],tau4)
 }

 #Priors:
```

```
 for(j in 1:3){
 beta[j]~dnorm(0,0.001)
 }
 c~dunif(0,134)
 tao~dgamma(0.1,0.1)
 tau2~dgamma(0.1,0.1)
 tau3~dgamma(0.1,0.1)
 tau4~dgamma(0.1,0.1)
 tau5~dgamma(0.1,0.1)
 tau6~dgamma(0.1,0.1)


}")
parameters=c("a1","a2","a3","like")
model_2=jags.model(model_string_2,data=data,n.chains=2,quiet=TRUE)
update(model_2,burn,progress.bar="none")
samples_2=coda.samples(model_2,variable.names=parameters,n.iter=iter,thin=2,progress.bar="none")
post_samples_2=rbind(samples_2[[1]],samples_2[[2]])

#WAIC Computation:
like=post_samples_2[,28:7980]
fbar=colMeans(like)
Pw=sum(apply(log(like),2,var))
WAIC_2=2*sum(log(fbar))+2*Pw

#DIC Computation
dic2=dic.samples(model_2,n.iter=iter,thin=2, progress.bar="none")

# Fit Model 3: Random Slopes with polynomial heteroskedasticity:

model_string_3=textConnection("model{
 #Likelihood:
 for (i in 1:n){
 Y[i]~dnorm(mu[i],inv_var[i])
 mu[i]=a1[cluster[i]]+a2[cluster[i]]*X[i]+a3[cluster[i]]*pow(X[i],2)
 inv_var[i]=1/sig2[i]
 log(sig2[i])=mu2+a4[cluster[i]]*pow(X[i],2)+a5[cluster[i]]*pow(X[i],3)
 #WAIC
 like[i]=dnorm(Y[i],mu[i],inv_var[i])
 }

 #Random Slopes:
 for (j in 1:m){
 a1[j]~dnorm(beta[1],tau2)
 a2[j]~dnorm(beta[2],tau3)
 a3[j]~dnorm(beta[3],tau4)
 a4[j]~dnorm(beta[4],tau5)
 a5[j]~dnorm(beta[5],tau6)
  }


 #Priors:
 mu2~dnorm(0,0.001)
```

```
 for(j in 1:5){
 beta[j]~dnorm(0,0.001)
 }
 tau2~dgamma(0.1,0.1)
 tau3~dgamma(0.1,0.1)
 tau4~dgamma(0.1,0.1)
 tau5~dgamma(0.1,0.1)
 tau6~dgamma(0.1,0.1)



 #Fit the model:
 for (i in 1:100){for(j in 1:9){
 fitted[i,j]=a1[j]+a2[j]*x[i]+a3[j]*(x[i]^2)
 }}


}")
parameters=c("a1","a2","a3","fitted")
model_3=jags.model(model_string_3,data=data,n.chains=2,quiet=TRUE)
update(model_3,burn,progress.bar="none")
samples_3=coda.samples(model_3,variable.names=parameters,n.iter=iter,thin=2,progress.bar="none")
post_samples_3=rbind(samples_3[[1]],samples_3[[2]])
#WAIC Computation:
like=post_samples_3[,928:8880]
fbar=colMeans(like)
Pw=sum(apply(log(like),2,var))
WAIC_3=2*sum(log(fbar))+2*Pw
#DIC Computation
dic3=dic.samples(model_3,n.iter=iter,thin=2, progress.bar="none")
```

3-fold Cross-Validation for Model Selection:

In the following CV, model M3 is fitted 2nd and model M2 is fitted 3rd.

```
K=3
obs=n/K
fold=rep(1:K,obs)
fold=sample(fold)
#Matrices to store the predictions for each of the K model fits:
Y_mean=matrix(NA,n,3)
Y_low=matrix(NA,n,3)
Y_high=matrix(NA,n,3)
#Initiate CV: Fit the model K times
for (f in 1:K) {
  # f is the index for the current testing group
  # Select data only from the training group
  ind=fold!=f
  data=list(Y=Y[ind],X=X[ind],n=sum(ind),m=m,cluster=cluster[ind],d=d[ind])
  #Fit model M1:
  model_string_1=textConnection("model{
   #Likelihood:
   for (i in 1:n){
   mu[i]=a1[cluster[i]]+a2[cluster[i]]*X[i]+a3[cluster[i]]*pow(X[i],2)
```

```
 Y[i]~dnorm(mu[i],tau1)
 }


 #Slopes as fixed effects:
 for (j in 1:m){
 a1[j]~dnorm(0,0.001)
 a2[j]~dnorm(0,0.001)
 a3[j]~dnorm(0,0.001)
 }


 #Priors:
 tau1~dgamma(0.1,0.1)
 }")
parameters=c("a1","a2","a3")
model_1=jags.model(model_string_1,data=data,n.chains=2,quiet=TRUE)
burn=10000
iter=25000
update(model_1,burn,progress.bar="none")
samples1=coda.samples(model_1,variable.names=parameters,n.iter=iter,thin=2,progress.bar="none")
a_model1=rbind(samples1[[1]],samples1[[2]])
# Fit Model M3:
model_string_2=textConnection("model{
 #Likelihood:
 for (i in 1:n){
 mu[i]=a1[cluster[i]]+a2[cluster[i]]*X[i]+a3[cluster[i]]*pow(X[i],2)
 Y[i]~dnorm(mu[i],inv_var[i])
 inv_var[i]=1/sig2[i]
 log(sig2[i])=mu2+a4[cluster[i]]*pow(X[i],2)+a5[cluster[i]]*pow(X[i],3)
 }

 #Random Slopes:
 for (j in 1:m){
 a1[j]~dnorm(beta[1],tau2)
 a2[j]~dnorm(beta[2],tau3)
 a3[j]~dnorm(beta[3],tau4)
 a4[j]~dnorm(beta[4],tau5)
 a5[j]~dnorm(beta[5],tau6)
                          }


 #Priors:
 mu2~dnorm(0,0.001)
 for(j in 1:5){
 beta[j]~dnorm(0,0.001)
 }
 tau2~dgamma(0.1,0.1)
 tau3~dgamma(0.1,0.1)
 tau4~dgamma(0.1,0.1)
 tau5~dgamma(0.1,0.1)
 tau6~dgamma(0.1,0.1)
```

```
 }")
parameters=c("a1","a2","a3")
model_2=jags.model(model_string_2,data=data,n.chains=2,quiet=TRUE)
update(model_2,burn,progress.bar="none")
samples2=coda.samples(model_2,variable.names=parameters,n.iter=iter,thin=2,progress.bar="none")
a_model2=rbind(samples2[[1]],samples2[[2]])
# Fit model M2:
model_string_3=textConnection("model{
 #Likelihood:
 for (i in 1:n){
 Y[i]~dnorm(mu[i],tau[i])
 mu[i]=a1[cluster[i]]+a2[cluster[i]]*X[i]+a3[cluster[i]]*pow(X[i],2)
 tau[i]=tao*exp(-d[i]/c)
 }


 #Random Slopes:
 for (j in 1:m){
 a1[j]~dnorm(beta[1],tau2)
 a2[j]~dnorm(beta[2],tau3)
 a3[j]~dnorm(beta[3],tau4)
 }

 #Priors:
 for(j in 1:3){
 beta[j]~dnorm(0,0.001)
 }
 c~dunif(0,134)
 tao~dgamma(0.1,0.1)
 tau2~dgamma(0.1,0.1)
 tau3~dgamma(0.1,0.1)
 tau4~dgamma(0.1,0.1)
 tau5~dgamma(0.1,0.1)
 tau6~dgamma(0.1,0.1)


 }")
parameters=c("a1","a2","a3")
model_3=jags.model(model_string_3,data=data,n.chains=2,quiet=TRUE)
update(model_3,burn,progress.bar="none")
samples3=coda.samples(model_3,variable.names=parameters,n.iter=iter,thin=2,progress.bar="none")
a_model3=rbind(samples3[[1]],samples3[[2]])


#Extract the posterior parameters for each model:
#Model M1:
a1_model1=a_model1[,1:9]
a2_model1=a_model1[,10:18]
a3_model1=a_model1[,19:27]

#Model M3:
a1_model2=a_model2[,1:9]
a2_model2=a_model2[,10:18]
```

```
  a3_model2=a_model2[,19:27]

  #Model M2:
  a1_model3=a_model3[,1:9]
  a2_model3=a_model3[,10:18]
  a3_model3=a_model3[,19:27]

  # Make predictions for the Test Data:
  for (i in 1:total) {if(fold[i]==f){

    Y_model1=a1_model1[,cluster[i]]+a2_model1[,cluster[i]]*X[i]+a3_model1[,cluster[i]]*(X[i]^2)
    Y_mean[i,1]=mean(Y_model1)
    Y_low[i,1]=quantile(Y_model1,0.025)
    Y_high[i,1]=quantile(Y_model1,0.0975)

    Y_model2=a1_model2[,cluster[i]]+a2_model2[,cluster[i]]*X[i]+a3_model2[,cluster[i]]*(X[i]^2)
    Y_mean[i,2]=mean(Y_model2)
    Y_low[i,2]=quantile(Y_model2,0.025)
    Y_high[i,2]=quantile(Y_model2,0.0975)

    Y_model3=a1_model3[,cluster[i]]+a2_model3[,cluster[i]]*X[i]+a3_model3[,cluster[i]]*(X[i]^2)
    Y_mean[i,3]=mean(Y_model3)
    Y_low[i,3]=quantile(Y_model3,0.025)
    Y_high[i,3]=quantile(Y_model3,0.0975)


     }}


  rm(model_1)
  rm(model_2)
  rm(model_3)

}


#Evaluate the models:
y=cbind(Y,Y,Y)
Bias=colMeans(Y_mean-y)
MSE=colMeans((Y_mean-y)^2)
MAD=colMeans(abs(Y_mean-y))
Coverage=colMeans((Y_low <= y) & (y <= Y_high))
CV=cbind(Bias,MSE,MAD,Coverage)
colnames(CV)=c("BIAS","MSE","MAD","COVERAGE")
rownames(CV)=c("Model M1","Model M3","Model M2")
kable(CV)
```

Fitting Model M3 and estimate the curves:

```
#Extract the Posterior Samples of the random slopes:
alpha1=post_samples_3[,1:9]
alpha2=post_samples_3[,10:18]
alpha3=post_samples_3[,19:27]
```

```
boxplot(alpha1,main="Posteriors of Intercepts across time-slice",outline=FALSE,xlab="Time Slice")
abline(h=0,col="red",lwd=1)
boxplot(alpha2,main="Posteriors of linear slopes across time-slice",outline=FALSE,xlab="Time Slice")
abline(h=0,col="red",lwd=1)
boxplot(alpha3,main="Posteriors of quadratic slopes across time-slice",outline=FALSE,xlab="Time Slice")
abline(h=0,col="red",lwd=1)

#Extract the fitted model and estimate the curves using the posterior median:
fit_3=summary(samples_3)$quantiles[1:900+27,3]
fit_3_low=summary(samples_3)$quantiles[1:900+27,1]
fit_3_high=summary(samples_3)$quantiles[1:900+27,5]
id_cluster=rep(1:9,each=100)
#Plot the fitted model:
for (j in 1:9) {
  plot(NA,xlim=range(X),ylim=range(Y),xlab="Paleo Latitude",ylab="MAT",main=clusters[j])
  points(X[cluster==j],Y[cluster==j],pch=1,cex=1)
  lines(x,fit_3[id_cluster==j],col="green",lty=1,lwd=2)
  lines(x,fit_3_low[id_cluster==j],lty=2,lwd=2)
  lines(x,fit_3_high[id_cluster==j],lty=2,lwd=2)
  legend("topright",c("Data","Posterior Median","95% interval"),
         pch=c(19,NA,NA),lty=c(NA,1,2),cex=1,bty="n",col=c("black","green","black"))
}


#Examine how MAT varies across the time-slices: Posteriors of the the 9 fitted curves
post_fit=post_samples_3[,1:900+27]
slice=seq(100,900,by=100)
m=NULL
for (i in 1:length(slice)) {
  temp1=post_fit[,(slice[i]-100+1):(slice[i])]
  temp2=rowMeans(temp1)
  m=cbind(m,temp2)
}
colnames(m)=clusters
boxplot(m,outline=FALSE,names=label,main="Posteriors of fitted curves for MAT across Time-Slices",ylab=
```

Posterior Predictive Checks/ Goodness of fit statistics for Model 3:

```
model_string_ppd=textConnection("model{
  #Likelihood:
   for (i in 1:n){
   mu[i]=a1[cluster[i]]+a2[cluster[i]]*X[i]+a3[cluster[i]]*pow(X[i],2)
   Y[i]~dnorm(mu[i],inv_var[i])
   inv_var[i]=1/sig2[i]
   log(sig2[i])=mu2+a4[cluster[i]]*pow(X[i],2)+a5[cluster[i]]*pow(X[i],3)
   }

  #Random Slopes:
   for (j in 1:m){
   a1[j]~dnorm(beta[1],tau2)
   a2[j]~dnorm(beta[2],tau3)
   a3[j]~dnorm(beta[3],tau4)
   a4[j]~dnorm(beta[4],tau5)
```

```
   a5[j]~dnorm(beta[5],tau6)
   }


  #Priors:
   mu2~dnorm(0,0.001)
   for(j in 1:5){
   beta[j]~dnorm(0,0.0001)
   }
   tau2~dgamma(0.1,0.1)
   tau3~dgamma(0.1,0.1)
   tau4~dgamma(0.1,0.1)
   tau5~dgamma(0.1,0.1)
   tau6~dgamma(0.1,0.1)


  #Posterior Predicitve Checks across all clusters
  #Sample 1 obs from the PPD of each variate:

   for (i in 1:n){
   mu_p[i]=a1[cluster[i]]+a2[cluster[i]]*X[i]+a3[cluster[i]]*pow(X[i],2)
   log(sig2_p[i])=mu2+a4[cluster[i]]*pow(X[i],2)+a5[cluster[i]]*pow(X[i],3)
   inv_var_p[i]=1/sig2_p[i]
   Y2[i]~dnorm(mu_p[i],inv_var_p[i])
   }

   D[1]=mean(Y2[])
   D[2]=sd(Y2[])
   D[3]=min(Y2[])
   D[4]=max(Y2[])
   D[5]=max(Y2[])-min(Y2[])


}")
parameters=c("D")
model_ppd=jags.model(model_string_ppd,data=data,n.chains=2,quiet=TRUE)
update(model_ppd,burn,progress.bar="none")
samples_ppd=coda.samples(model_ppd,variable.names=parameters,n.iter=iter,thin=2,progress.bar="none")

#True Statistics:
D0=c(mean(Y),sd(Y),min(Y),max(Y),max(Y)-min(Y))
D0_names=c("Mean Y","Standard Deviation Y","Min Y","Max Y","Range Y")
m=length(D0)
p=rep(0,m)
D=rbind(samples_ppd[[1]],samples_ppd[[2]])

#Posterior Predictive Check Summary:
for (i in 1:m) {
  #Evaluate the p-values:
  p[i]=mean(D[,i]>D0[i])
  hist(D[,i],breaks=40,col="blue",ylab="Posterior Density",main=paste(D0_names[i],"p-value",p[i]))
  abline(v=D0[i],col="green",lwd=3)
  legend("topright",c("Posterior Predictive","True Value"),col=c("blue","green"),bty="n",lty=1,cex=0.9)
```

```
}
```

Fitting Model 3 to the mixed-data-Interval Censoring in JAGS:

```
#Quantitative Data:
miss=is.na(Y)
Y_quant=Y[!miss]
#Censored Data:
min=df$Min.Temp
max=df$Max.Temp
Y_cens=Y[miss]
#Mixed Data:
Z=c(Y_quant,Y_cens)
X_quant=X[!miss]
X_cens=X[miss]
time_slice=time_slice[!miss]
# Find the Clusters of the data:
clusters=unique(df$Paleocoordinate.Age)
m=length(clusters)
n=length(Y_quant)
cluster=matrix(NA,nrow=length(Y),ncol=1)
for (i in 1:m) {
  cluster[time_slice==clusters[i]]=i

}

#Censored Data:
X_cens=X[miss]
low=min[miss]
high=max[miss]
time_slice_cens=df$Paleocoordinate.Age[miss]
J=length(X_cens)
cluster_cens=matrix(NA,nrow=J,ncol=1)
for (i in 1:m) {
  cluster_cens[time_slice_cens==clusters[i]]=i

}
lim=cbind(low,high)
colnames(lim)=NULL
#I indicator random variable for interval censoring:
I=rep(1,nrow(lim))

#Fit Model 3 with both quantitative and interval data:
Y=as.vector(Y)
X_quant=as.vector(X_quant)
X_cens=as.vector(X_cens)
cluster=as.vector(cluster)
cluster_cens=as.vector(cluster_cens)
x=seq(min(X),max(X),length=100)
data=list(Z=Z,X_quant=X_quant,X_cens=X_cens,x=x,n=n,J=J,I=I,lim=lim,m=m,cluster=cluster,cluster_cens=clu
model_string=textConnection("model{
 #Likelihood:
```

```
 for (i in 1:n){
 Z[i]~dnorm(mu[i],inv_var[i])
 mu[i]=a1[cluster[i]]+a2[cluster[i]]*X_quant[i]+a3[cluster[i]]*pow(X_quant[i],2)
 log(sig2[i])=mu2+a4[cluster[i]]*pow(X_quant[i],2)+a5[cluster[i]]*pow(X_quant[i],3)
 inv_var[i]=1/sig2[i]
 }

#Censored Data
 for (j in 1:J){
#Indicator Random Variable I:
 I[j]~dinterval(Z[n+j],lim[j,])
 mu[n+j]=a1[cluster_cens[j]]+a2[cluster_cens[j]]*X_cens[j]+a3[cluster_cens[j]]*pow(X_cens[j],2)
 log(sig2[n+j])=mu2+a4[cluster_cens[j]]*pow(X_cens[j],2)+a5[cluster_cens[J]]*pow(X_cens[j],3)
 inv_var[n+j]=1/sig2[n+j]
#Likelihood for the censored response:
 Z[n+j]~dnorm(mu[n+j],inv_var[n+j])
 }

  #Random Slopes:
  for (j in 1:m){
  a1[j]~dnorm(beta[1],tau2)
  a2[j]~dnorm(beta[2],tau3)
  a3[j]~dnorm(beta[3],tau4)
  a4[j]~dnorm(beta[4],tau5)
  a5[j]~dnorm(beta[5],tau6)
  }


 #Priors:
  mu2~dnorm(0,0.001)
  for(j in 1:5){
  beta[j]~dnorm(0,0.0001)
  }
  tau2~dgamma(0.1,0.1)
  tau3~dgamma(0.1,0.1)
  tau4~dgamma(0.1,0.1)
  tau5~dgamma(0.1,0.1)
  tau6~dgamma(0.1,0.1)



 #Fit the model:
 for (i in 1:100){for(j in 1:9){
 fitted[i,j]=a1[j]+a2[j]*x[i]+a3[j]*(x[i]^2)
 }}



 #Posterior Predicitve Checks across all clusters
 #Sample 1 obs from the PPD for the Quantitative responses:


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

```
 mu_p[i]=a1[cluster[i]]+a2[cluster[i]]*X_quant[i]+a3[cluster[i]]*pow(X_quant[i],2)
 log(sig2_p[i])=mu2+a4[cluster[i]]*pow(X_quant[i],2)+a5[cluster[i]]*pow(X_quant[i],3)
 inv_var_p[i]=1/sig2_p[i]
 Z2[i]~dnorm(mu_p[i],inv_var_p[i])
 }

 D[1]=mean(Z2[])
 D[2]=sd(Z2[])
 D[3]=min(Z2[])
 D[4]=max(Z2[])
 D[5]=max(Z2[])-min(Z2[])


}")
parameters=c("a1","a2","a3","fitted","D")
model=jags.model(model_string,data=data,n.chains=2,quiet=TRUE)
burn=10000
iter=25000
update(model,burn,progress.bar="none")
samples=coda.samples(model,variable.names=parameters,n.iter=iter,thin=2,progress.bar="none")
post_samples=rbind(samples[[1]],samples[[2]])

#Extract the Posterior Samples
alpha1=post_samples[,1:9+5]
alpha2=post_samples[,10:18+5]
alpha3=post_samples[,19:27+5]
boxplot(alpha1,main="Random Intercepts across clusters",outline=FALSE,xlab="Cluster")
abline(h=0,col="red",lwd=1)
boxplot(alpha2,main="Random linear slope across clusters",outline=FALSE,xlab="Cluster")
abline(h=0,col="red",lwd=1)
boxplot(alpha3,main="Random quadratic slope across clusters",outline=FALSE,xlab="Cluster")
abline(h=0,col="red",lwd=1)

#Extract the fitted model and estimate the curves using the posterior median:
fit=summary(samples)$quantiles[1:900+27+5,3]
fit_low=summary(samples)$quantiles[1:900+27+5,1]
fit_high=summary(samples)$quantiles[1:900+27+5,5]
id_cluster=rep(1:9,each=100)
#Plot the fitted model:
for (j in 1:9) {
  plot(NA,xlim=range(X),ylim=range(Y_quant),xlab="Paleo Latitude",ylab="MAT",main=clusters[j])
  points(X_quant[cluster==j],Y_quant[cluster==j],pch=1,cex=1)
  lines(x,fit[id_cluster==j],col="green",lty=1,lwd=2)
  lines(x,fit_low[id_cluster==j],lty=2,lwd=2)
  lines(x,fit_high[id_cluster==j],lty=2,lwd=2)
  legend("topright",c("Data","Posterior Median","95% interval"),
         pch=c(19,NA,NA),lty=c(NA,1,2),cex=1,bty="n",col=c("black","green","black"))
}


#Posterior Predictive Checks:
D=post_samples[,1:5]
#True Statistics:
```

```
D0=c(mean(Y_quant),sd(Y_quant),min(Y_quant),max(Y_quant),max(Y_quant)-min(Y_quant))
D0_names=c("Mean Y","Standard Deviation Y","Min Y","Max Y","Range Y")
m=length(D0)
p=rep(0,m)
for (i in 1:m) {
  p[i]=mean(D[,i]>D0[i])
  hist(D[,i],breaks=40,col="blue",ylab="Posterior Density",main=paste(D0_names[i],"p-value",p[i]))
  abline(v=D0[i],col="green",lwd=3)
  legend("topright",c("Posterior Predictive","True Value"),col=c("blue","green"),bty="n",lty=1,cex=0.9)

}
```