NDVI Prediction Project

James Lee ( jlee73@ncsu.edu)

**Statistical Model:**

**In this section I will separate the method into two procedures, data preprocessing and the Bayesian modeling.**

**Based on the description of the three datasets Y1, Y2, Y3, Y1 is unbiased but Y2 and Y3 are not. Therefore, I try to “eliminate” the bias by calculate the difference between the means. Note that Y2 only measure pixel 1, thus its mean is compared only with Y1[,1], the first column of Y1. I then construct a Y dataset by simply averaging from all unbiased Y1, Y2 and Y3. Lastly, I filled rest of the missing values (which is less than 3%) with grand mean to finish the data preprocessing part.**

**The Bayesian model I use is 2 way random effects model. Denote αi∼Normal(0,σ^2) is random effect between time, γj∼Normal(0,σ^2) is random effect between pixel, and σe^2 is the error variance. All unknown prior are denote as uninformative prior. (mu~N(0,100) and tau~Gamma(0.1,0.1))**

**The Code below should allow anyone to reproduce the results.**

**JAGS Code:**

**###################################################################**

**# 2018 ST540 Take Home Exam Code**

**# Author: James Lee**

**# ID: 200203022**

**# Abstract: The goal of this question is to estimate the NDVI**

**# in 6 pixels over the course of 365 days.**

**# Three datasets from different satellites is given (Y1.Y2.Y3)**

**# Y1 is gold standard, unbiased, 80% missing value**

**# Y2 is potentially biased, single pixel(pixel1), 10% missing value**

**# Y3 is potentially biased, entire spatial avg, 10% missing value**

**###################################################################**

**#load dataset**

**#change directory if needed**

**load("D:/E2.Rdata")**

**###data preprocessing**

**##adjust Y2 Y3 based on Y1 by subtract the difference in mean**

**#Y2 only have observation on pixel1, compare it with Y1[,1]**

**Y2\_mean<-mean(Y2, na.rm=TRUE)**

**Y1p1\_mean<-mean(Y1[,1], na.rm=TRUE)**

**bias\_Y2<-Y2\_mean-Y1p1\_mean**

**#Y3 is the average of pixels, compare it with the whole Y1 dataset**

**Y3\_mean<-mean(Y3, na.rm=TRUE)**

**Y1\_mean<-mean(Y1, na.rm=TRUE)**

**bias\_Y3<-Y3\_mean-Y1\_mean**

**#Create new datasets for Y2.Y3**

**Y2\_revise<-Y2-bias\_Y2**

**Y3\_revise<-Y3-bias\_Y3**

**#construct the full dataset Y\_proc: average of {Y1.Y2.Y3}**

**Y\_proc<-matrix(, nrow = 365, ncol = 6)**

**Y\_proc[,1]<-rowMeans(cbind(Y1[,1],Y2\_revise,Y3\_revise),na.rm=TRUE)**

**Y\_proc[,2]<-rowMeans(cbind(Y1[,2],Y3\_revise),na.rm=TRUE)**

**Y\_proc[,3]<-rowMeans(cbind(Y1[,3],Y3\_revise),na.rm=TRUE)**

**Y\_proc[,4]<-rowMeans(cbind(Y1[,4],Y3\_revise),na.rm=TRUE)**

**Y\_proc[,5]<-rowMeans(cbind(Y1[,5],Y3\_revise),na.rm=TRUE)**

**Y\_proc[,6]<-rowMeans(cbind(Y1[,6],Y3\_revise),na.rm=TRUE)**

**#fill in grand mean to the rest of the NaNs**

**Y\_proc[is.nan(Y\_proc)]<-mean(Y\_proc,na.rm = TRUE)**

**#store grand mean.col.row for further usage**

**Y\_bar=mean(Y\_proc,na.rm = TRUE)**

**ns<-nrow(Y\_proc)**

**nt<-ncol(Y\_proc)**

**###Bayesian modeling- 2 way random effects model**

**##Yij~Normal(mu+alphai+gammaj,taue^2)**

**#JAGs code**

**NDVI\_model <- "model{**

**# Likelihood**

**for(i in 1:ns){for(j in 1:nt){**

**Y[i,j] ~ dnorm(mean[i,j],taue)**

**mean[i,j] <- mu + alpha[i] + gamma[j]**

**}}**

**# Random effects**

**for(i in 1:ns){**

**alpha[i] ~ dnorm(0,taus)**

**}**

**for(j in 1:nt){**

**gamma[j] ~ dnorm(0,taut)**

**}**

**# Priors**

**mu ~ dnorm(0,0.01)**

**taue ~ dgamma(0.1,0.1)**

**taus ~ dgamma(0.1,0.1)**

**taut ~ dgamma(0.1,0.1)**

**# Output the parameters of interest**

**sigma2[1] <- 1/taue**

**sigma2[2] <- 1/taus**

**sigma2[3] <- 1/taut**

**sigma[1] <- 1/sqrt(taue)**

**sigma[2] <- 1/sqrt(taus)**

**sigma[3] <- 1/sqrt(taut)**

**pct[1] <- sigma2[1]/sum(sigma2[])**

**pct[2] <- sigma2[2]/sum(sigma2[])**

**pct[3] <- sigma2[3]/sum(sigma2[])**

**}"**

**#run rjags package###convergence diagnostics**

**library(rjags)**

**dat <- list(Y=Y\_proc,ns=ns,nt=nt)**

**init <- list(mu=Y\_bar)**

**model1 <- jags.model(textConnection(NDVI\_model),inits=init,data = dat, n.chains=1)**

**update(model1, 10000, progress.bar="none")**

**samp <- coda.samples(model1,**

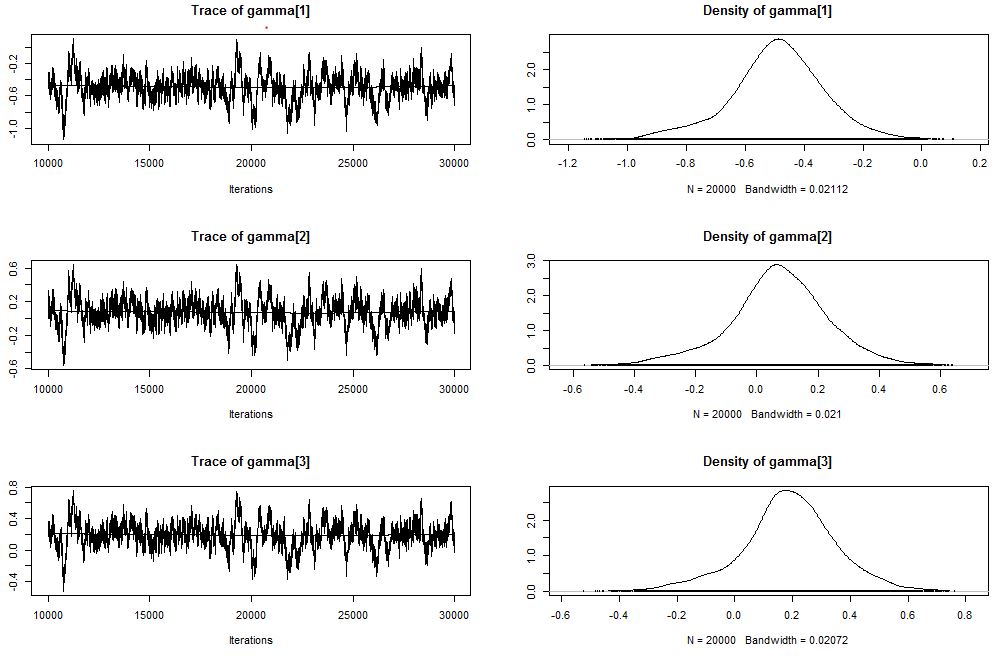
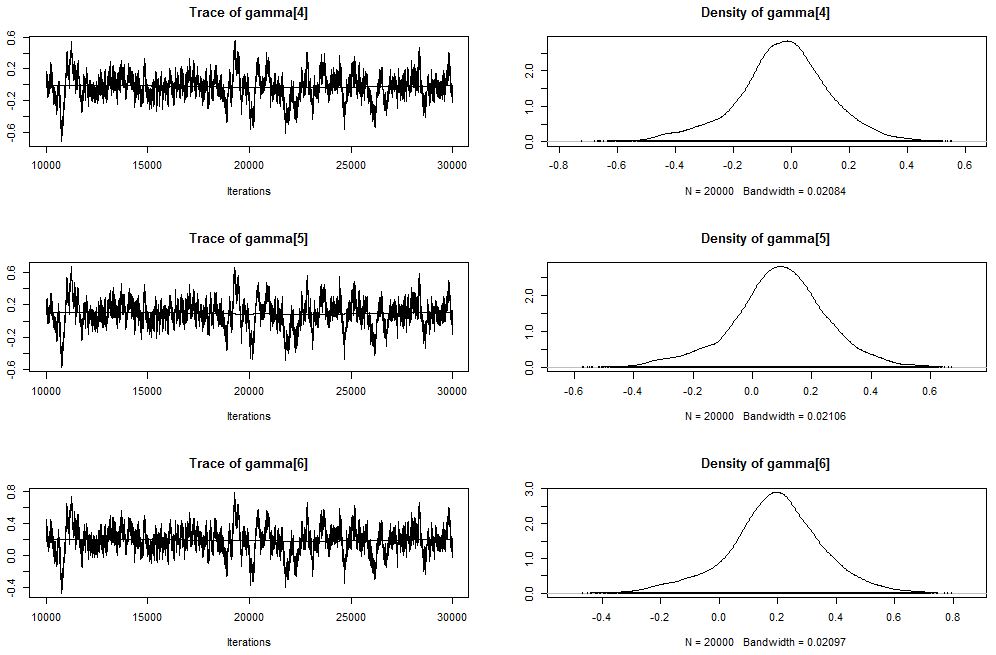
**variable.names=c("sigma","pct","gamma"),**

**n.iter=20000, progress.bar="none")**

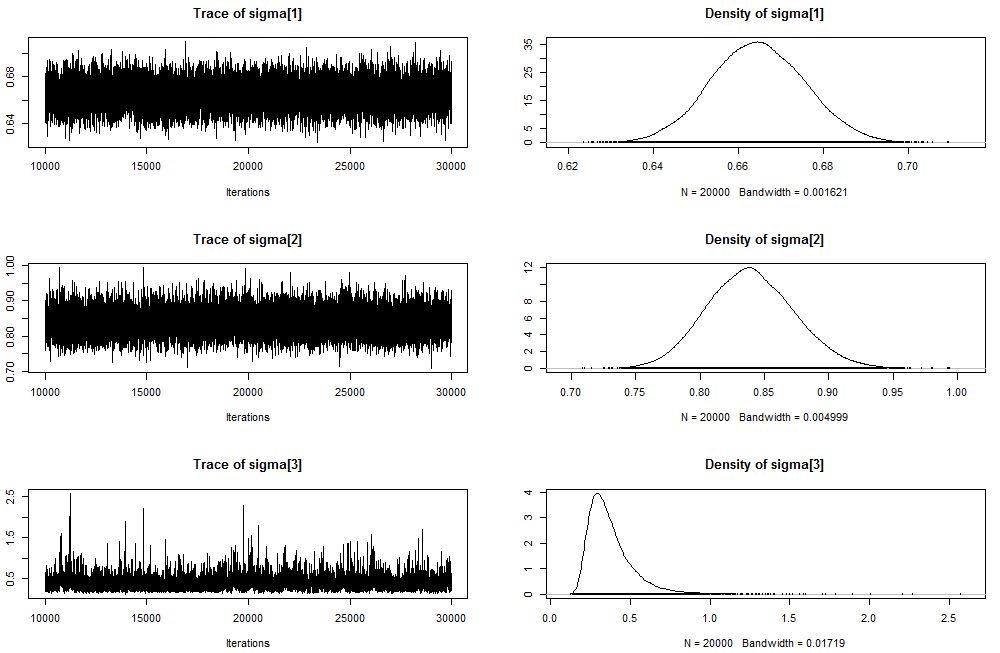
**plot(samp)**

**Convergence Diagnostics:**

**As the trace plot shows, both coefficients of random effects converges after 20000 iterations (and 10000 burn-outs).**



**Final Results:**



**By fitting coefficients, we can have predictions for all 365\*6 entries. It is interesting to see how data preprocessing will affect final results. Y\_proc, the dataset after data preprocessing is actually very familiar to the final results already. And NaNs fill with grand mean does somewhat affect certain unfortunate columns where there isn’t any observations on all three satellites. One improvement may be done is take average around the missing value to fill in.**

**Another important point that needs to be mentioned is the model I was using are not how it was generated. The data was generated with lag-1 time dependence, where pixel in time t is related to time t-1. Where I used 2-way random effects did wasted that precious information.**