LAB_3

Name Huiyu Hu

```
# Set working directory
setwd("/Users/huiyuhu/Desktop/Study/UCLA_Biostat/BIOSTAT234/lab/Lab 3")
getwd()

## [1] "/Users/huiyuhu/Desktop/Study/UCLA_Biostat/BIOSTAT234/lab/Lab 3"

#LOAD NECESSARY PACKAGES
library(R2jags)

## Loading required package: rjags

## Loading required package: coda

## Linked to JAGS 4.3.0

## Loaded modules: basemod,bugs

##
## Attaching package: 'R2jags'

## The following object is masked from 'package:coda':
##
##     traceplot

library(lattice)
#Function to help with the burn in
load("AddBurnin.RData")
TraumaData = read.table("bcj97data.txt")

#Give the columns useful names
colnames(TraumaData) <- c("death", "n", "intercept",    "iss",  "rts", "age",
 "ti", "age*ti")

#first 6 observations from proto-typical cases
head(TraumaData, n=6)

##   death    n intercept iss  rts age ti age*ti
## 1   1.1  8.6         1  25 7.84  60  0      0
## 2   3.0 13.0         1  25 3.34  10  0      0
## 3   4.9  6.6         1  41 3.34  60  1     60
## 4   1.3 12.3         1  41 7.84  10  1     10
## 5   1.1  5.0         1  33 5.74  35  0      0
## 6   1.5  6.0         1  33 5.74  35  1     35

#For the 6 proto-typical cases define Xp the design matrix, Yp the outcomes (
death=1), and np the number of trials
```

```
Xp = as.matrix(TraumaData[1:6,3:8])
Yp = TraumaData[1:6,1]
np = TraumaData[1:6,2]
#define the inverse of the Xp matrix to be used to convert the prior distribu
tions on pi to distributions on the regression coefficients
invXp = solve(Xp)
#For the observed data define the design matrix, outcomes, and number of tria
ls
Xobs = as.matrix(TraumaData[7:306,3:8])
Yobs = TraumaData[7:306,1]
nobs = TraumaData[7:306,2]
```

- Store the model in the file LAB3.Priors.txt

```
# sink("LAB3.Priors.txt")
# cat("
# model{
#
#   betas<-invXp %*% logitp[]
#
#   for(j in 1:6){
#       logitp[j]<-logit(pie[j])
#   }
#   pie[1]~dbeta(1.1,8.5)
#   pie[2]~dbeta(3.0,11.0)
#   pie[3]~dbeta(5.9,1.7)
#   pie[4]~dbeta(1.3,12.9)
#   pie[5]~dbeta(1.1,4.9)
#   pie[6]~dbeta(1.5,5.5)
#
# }
#
#   ",fill = TRUE)
# sink()

#Now we incorporate the data into the model

# sink("Lab3.Posteriors.txt")
# cat("
# model{
#
#   betas<-invXp %*% logitp[]
#
#   for(j in 1:6){
#       logitp[j]<-logit(pie[j])
#   }
#   pie[1]~dbeta(1.1,8.5)
#   pie[2]~dbeta(3.0,11.0)
#   pie[3]~dbeta(5.9,1.7)
#   pie[4]~dbeta(1.3,12.9)
#   pie[5]~dbeta(1.1,4.9)
#   pie[6]~dbeta(1.5,5.5)
#
#
#       for(i in 1:T){
#       y[i] ~ dbern(p[i])
#       p[i]<-ilogit(inprod(x[i,],betas[]))
#       }
#
#
# }
#   ",fill = TRUE)
# sink()
```

To-Do 1

```
ex1.data=list(invXp=invXp)
ex1.inits=rep(list(list(pie=c(0.5,0.5,0.5,0.5,0.5,0.5))),5)
ex1.parameters = c("betas", "pie[1:6]")

#Run the JAGS model
hw1.out = jags(ex1.data, ex1.inits, ex1.parameters, "LAB3.Priors.txt",
    n.chains=5, n.iter=51000, n.burnin=0, n.thin=2, DIC=F) # the first 1000 i
terations as a burn in

## module glm loaded

## module dic loaded

## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 0
##    Unobserved stochastic nodes: 6
##    Total graph size: 62
##
## Initializing model

#Treat the first 1000 iterations as a burn in
Output.hw1 = AddBurnin(hw1.out$BUGSoutput$sims.array,burnin=1000,n.thin=2)

ex2.data = list(x=Xobs, y=Yobs, T=300, invXp=invXp)
ex2.inits = rep(list(list(pie=c(0.5,0.5,0.5,0.5,0.5,0.5))),5)
ex2.parameters = c("betas", "pie[1:6]")


hw2.out = jags(ex2.data, ex2.inits, ex2.parameters, "Lab3.Posteriors.txt",
    n.chains=5, n.iter=51000, n.burnin=0, n.thin=2, DIC=F)

## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 300
##    Unobserved stochastic nodes: 6
##    Total graph size: 3025
##
## Initializing model
```

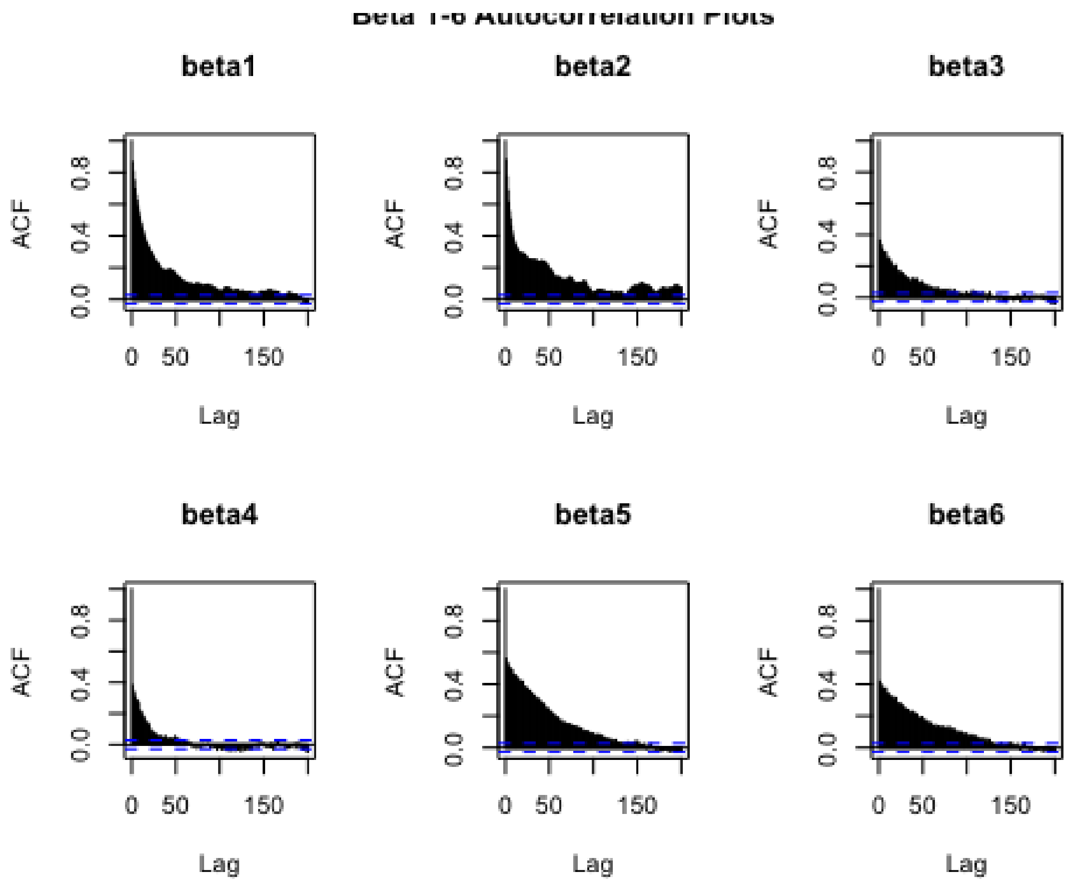
```
#Treat the first 1000 iterations as a burn in
Output.hw2 = AddBurnin(hw2.out$BUGSoutput$sims.array,burnin=1000,n.thin=2)

# Partial data T = 100
ex3.data = list(x=Xobs, y=Yobs, T=100, invXp=invXp)
ex3.inits = rep(list(list(pie=c(0.5,0.5,0.5,0.5,0.5,0.5))),5)
ex3.parameters = c("betas", "pie[1:6]")



hw3.out = jags(ex2.data, ex2.inits, ex2.parameters, "Lab3.Posteriors.txt",
    n.chains=5, n.iter=51000, n.burnin=0, n.thin=2, DIC=F)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 300
##    Unobserved stochastic nodes: 6
##    Total graph size: 3025
##
## Initializing model
```

```
#Treat the first 1000 iterations as a burn in
Output.hw3 = AddBurnin(hw3.out$BUGSoutput$sims.array,burnin=1000,n.thin=2)
```

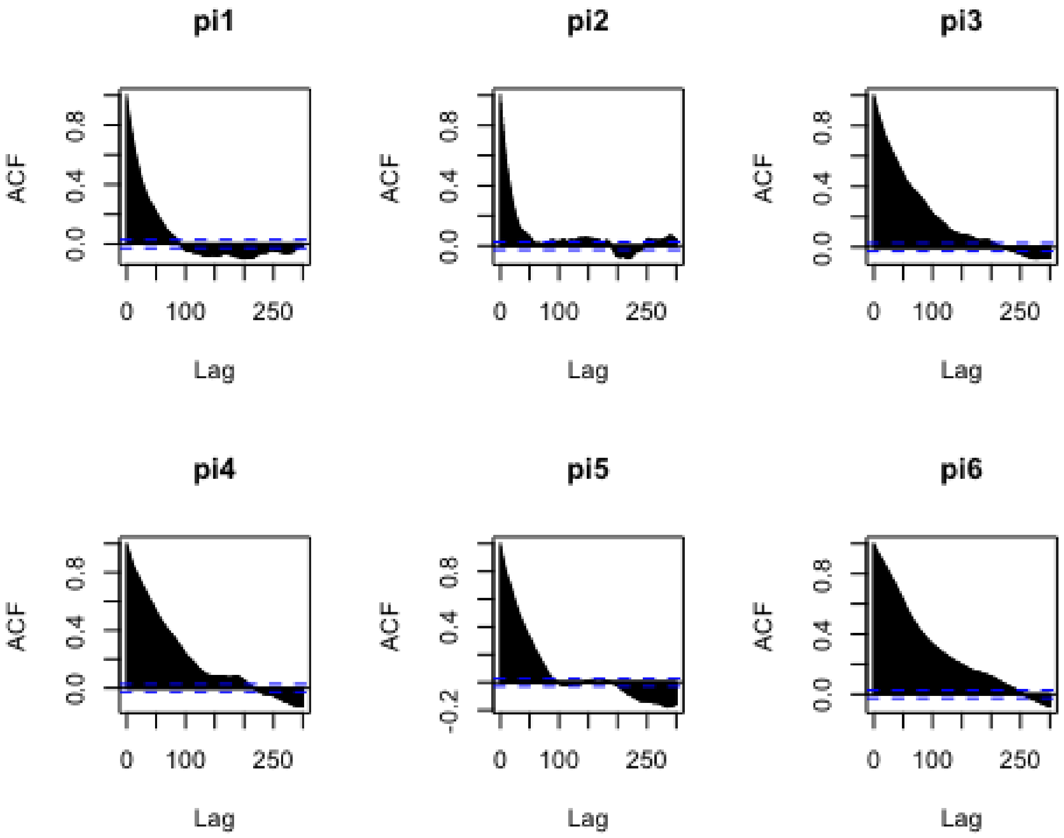
1. At what lags do the autocorrelations hit zero for the 6 regression coefficients? Are the beta autocorrelations better or worse than the 6 pi's?

```
par(mfrow=c(2,3))
acf(hw2.out$BUGSoutput$sims.array[1:5000,1,1], main="beta1", lag.max = 200)
acf(hw2.out$BUGSoutput$sims.array[1:5000,1,2], main="beta2", lag.max = 200)
acf(hw2.out$BUGSoutput$sims.array[1:5000,1,3], main="beta3", lag.max = 200)
acf(hw2.out$BUGSoutput$sims.array[1:5000,1,4], main="beta4", lag.max = 200)
acf(hw2.out$BUGSoutput$sims.array[1:5000,1,5], main="beta5", lag.max = 200)
acf(hw2.out$BUGSoutput$sims.array[1:5000,1,6], main="beta6", lag.max = 200)
title(outer = T, "Beta 1-6 Autocorrelation Plots")
```



Autocorrelation of β_1 hits 0 at Lag ≈ 100
Autocorrelation of β_2 hits 0 at Lag ≈ 120
Autocorrelation of β_3 hits 0 at Lag ≈ 100
Autocorrelation of β_4 hits 0 at Lag ≈ 100
Autocorrelation of β_5 hits 0 at Lag ≈ 70
Autocorrelation of β_6 hits 0 at Lag ≈ 100

```r
par(mfrow=c(2,3))
acf(hw2.out$BUGSoutput$sims.array[1:5000,1,7], main="pi1", lag.max = 300)
acf(hw2.out$BUGSoutput$sims.array[1:5000,1,8], main="pi2", lag.max = 300)
acf(hw2.out$BUGSoutput$sims.array[1:5000,1,9], main="pi3", lag.max = 300)
acf(hw2.out$BUGSoutput$sims.array[1:5000,1,10], main="pi4", lag.max = 300)
acf(hw2.out$BUGSoutput$sims.array[1:5000,1,11], main="pi5", lag.max = 300)
acf(hw2.out$BUGSoutput$sims.array[1:5000,1,12], main="pi6", lag.max = 300)
```



Autocorrelation of π_1 hits 0 at Lag ≈ 80
Autocorrelation of π_2 hits 0 at Lag ≈ 100
Autocorrelation of π_3 hits 0 at Lag ≈ 200
Autocorrelation of π_4 hits 0 at Lag ≈ 90
Autocorrelation of π_5 hits 0 at Lag ≈ 100
Autocorrelation of π_6 hits 0 at Lag ≈ 250

2. Turn in your properly formatted table of output for the full data set, and turn in a set of the 6 plots of the prior and posterior for the betas.

- Table of output

```
names <- paste0("betas[", 1:6, "]")
Parameter <- c("Intercept", "ISS", "RTS", "Age", "ti", "Age * ti")
table1 <- cbind(Parameter, round(Output.hw2$Burnin.Summary[names, ], 4))
knitr::kable(table1, caption = "Posterior of Betas from Full Dataset Run")
```

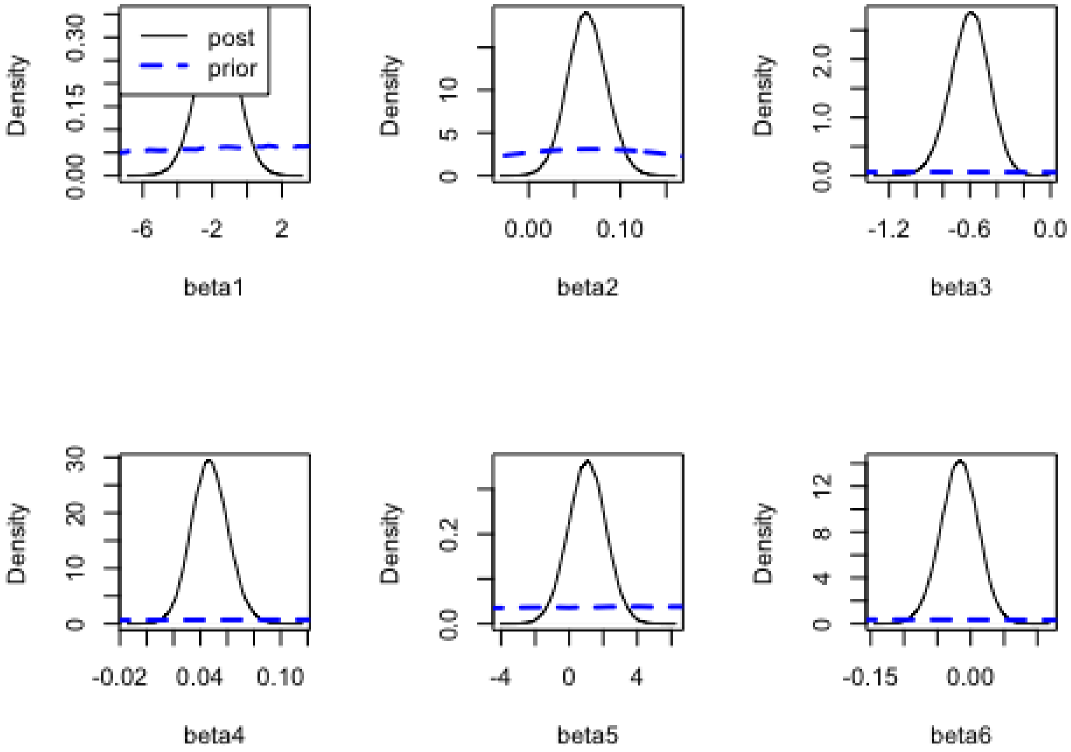
Posterior of Betas from Full Dataset Run

	Parameter	mu.vect	sd.vect	2.5%	97.5%	P>0
betas[1]	Intercept	-1.7179	1.1291	-3.9556	0.4705	0.0623
betas[2]	ISS	0.0639	0.0211	0.0231	0.1057	0.9988
betas[3]	RTS	-0.6013	0.1427	-0.8906	-0.33	0

betas[4]	Age	0.0472	0.0137	0.0211	0.075	0.9998
betas[5]	ti	1.0488	1.1022	-1.1258	3.2106	0.8292
betas[6]	Age * ti	-0.0169	0.0281	-0.073	0.0374	0.2754

- A set of the 6 plots of the prior and posterior for the betas

```
temp3 = Output.hw1$Burnin.sims.matrix
temp4 = Output.hw2$Burnin.sims.matrix
par(mfrow=c(2,3))
plot(density(temp4[,1]),main="",xlab="beta1")
# beta1.
lines(density(temp3[,1],bw=.055),lty=2,lwd=2, col="blue")
# prior
legend("topleft", legend=c("post","prior"), col=c("black", "blue"), lty=1:2 ,
 lwd=c(1,2))
plot(density(temp4[,2]),main="",xlab="beta2")
# beta2.
lines(density(temp3[,2],bw=.05),lty=2,lwd=2, col="blue")
# prior
plot(density(temp4[,3]),main="",xlab="beta3")
# beta3.
lines(density(temp3[,3],bw=.075),lty=2,lwd=2, col="blue")
# prior
plot(density(temp4[,4]),main="",xlab="beta4")
# beta4.
lines(density(temp3[,4],bw=.045),lty=2,lwd=2, col="blue")
# prior
plot(density(temp4[,5]),main="",xlab="beta5")
# beta5.
lines(density(temp3[,5],bw=.085),lty=2,lwd=2, col="blue")
# prior
plot(density(temp4[,6]),main="",xlab="beta6")
# beta6.
lines(density(temp3[,6],bw=.085),lty=2,lwd=2, col="blue")
```



```
# prior
```

3. Turn in the results of the TODO step 2 properly formatted and your figures nicely annotated. TODO STEP2: Compare output for 3 different runs: (i) prior (no data), (ii) partial data (T=100) posterior and then (iii) with the full data (T=300).

a. Compare all estimates and standard deviations in a table. [Given that you have three runs to compare, you'll not want to have as many summaries of the posteriors as if you were doing a table of a single model. Think about how to best arrange the numbers in the table.]

```
names <- paste0("betas[", 1:6, "]")
col <- c('mu.vect', 'sd.vect')
prior <-  round(Output.hw1$Burnin.Summary[names, col], 4)
p.post <- round(Output.hw2$Burnin.Summary[names, col], 4)
f.post <- round(Output.hw3$Burnin.Summary[names, col], 4)
table2 <- rbind(prior, p.post, f.post)
rownames(table2) <- c()
parameter <- rep(c("Beta 1", "Beta 2", "Beta 3", "Beta 4", "Beta 5", "Beta 6"
), 3)
table2 <- cbind(parameter, table2)
rownames(table2) = c("Prior", " ", " ", " ", " ", " ", "Partial Posterior", "
 ", " ", "", " ", " ", "Full Posterior", " ", " ", "", " ", " ")
```

```
knitr::kable(table2, caption = "Parameter estimates from Prior, Partial Data,
 and Full Data models")
```

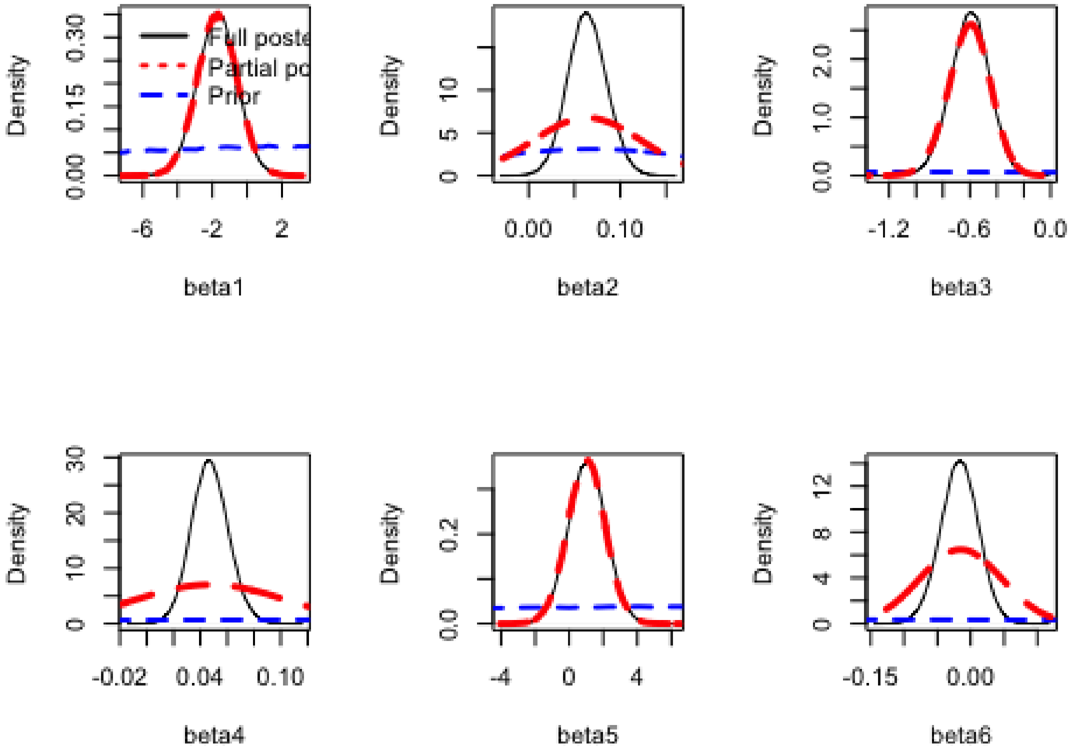
Parameter estimates from Prior, Partial Data, and Full Data models

	parameter	mu.vect	sd.vect
Prior	Beta 1	4.5628	16.7856
	Beta 2	0.0643	0.1226
	Beta 3	-3.0272	6.5588
	Beta 4	0.251	0.5964
	Beta 5	15.7398	41.8758
	Beta 6	-0.4402	1.1878
Partial Posterior	Beta 1	-1.7179	1.1291
	Beta 2	0.0639	0.0211
	Beta 3	-0.6013	0.1427
	Beta 4	0.0472	0.0137
	Beta 5	1.0488	1.1022
	Beta 6	-0.0169	0.0281
Full Posterior	Beta 1	-1.717	1.1486
	Beta 2	0.064	0.0214
	Beta 3	-0.6052	0.1448
	Beta 4	0.0477	0.014
	Beta 5	1.0028	1.1031
	Beta 6	-0.0151	0.0279

b. Draw plots with three densities for each coefficient. The plot should show the prior density, the partial data posterior and full data posteriors. Label appropriately

```
temp3 = Output.hw1$Burnin.sims.matrix
temp4 = Output.hw2$Burnin.sims.matrix
temp5 = Output.hw3$Burnin.sims.matrix
par(mfrow=c(2,3))
plot(density(temp4[,1]),main="",xlab="beta1")
# beta1.
lines(density(temp3[,1],bw=.055),lty=2,lwd=2, col="blue")
# prior
lines(density(temp5[,1],bw=.055),lty=2,lwd=3, col="red")
```

```
legend("topleft", lty = c(1,3,2), col = c("black", "red", "blue"), lwd = c(2,
2,2), legend = c("Full posterior", "Partial posterior", "Prior"), bty = "n")

plot(density(temp4[,2]),main="",xlab="beta2")
# beta2.
lines(density(temp3[,2],bw=.05),lty=2,lwd=2, col="blue")
# prior
lines(density(temp5[,2],bw=.055),lty=2,lwd=3, col="red")

plot(density(temp4[,3]),main="",xlab="beta3")
# beta3.
lines(density(temp3[,3],bw=.075),lty=2,lwd=2, col="blue")
# prior
lines(density(temp5[,3],bw=.055),lty=2,lwd=3, col="red")

plot(density(temp4[,4]),main="",xlab="beta4")
# beta4.
lines(density(temp3[,4],bw=.045),lty=2,lwd=2, col="blue")
# prior
lines(density(temp5[,4],bw=.055),lty=2,lwd=3, col="red")

plot(density(temp4[,5]),main="",xlab="beta5")
# beta5.
lines(density(temp3[,5],bw=.085),lty=2,lwd=2, col="blue")
# prior
lines(density(temp5[,5],bw=.055),lty=2,lwd=3, col="red")

plot(density(temp4[,6]),main="",xlab="beta6")
# beta6.
lines(density(temp3[,6],bw=.085),lty=2,lwd=2, col="blue")
# prior
lines(density(temp5[,6],bw=.055),lty=2,lwd=3, col="red")
```



4. Turn in your answer to TODO step 4: The model tracks the parameters π_1 to π_6, what is the interpretation of these parameters once the data has been incorporated?

- π_1 to π_6 are used to estimate the survival probability that 6 patients or predict the survival probability the patients with same condition as these 6 patients.

Extra credit: you may (but don't need to) Turn in your answer to TODO step 3.(1) Extra credit 1: add code to calculate the probability that someone with Xs of 1 2 7.55 25 0 0 1 11 7.8408 42 1 42 1 16 7.8408 80 1 80 will die.

```
# sink("Lab3.Posteriors.pred.txt")
# cat("
# model{
#
#   betas<-invXp %*% logitp[]
#
#   for(j in 1:6){
#       logitp[j]<-logit(pie[j])
#   }
#   pie[1]~dbeta(1.1,8.5)
#   pie[2]~dbeta(3.0,11.0)
#   pie[3]~dbeta(5.9,1.7)
#   pie[4]~dbeta(1.3,12.9)
#   pie[5]~dbeta(1.1,4.9)
#   pie[6]~dbeta(1.5,5.5)
#
#
#       for(i in 1:T){
#       y[i] ~ dbern(p[i])
#       p[i]<-ilogit(inprod(x[i,],betas[]))
#       }
# for(k in 1:3) {
#       predPi[k] = ilogit(inprod(newx[k,], betas[]))
#   }
#
# }
#   ",fill = TRUE)
# sink()
```

```
x1 <- c(1,2,7.55,25,0,0)
x2 <- c(1,11,7.8408,42,1,42)
x3 <- c(1,16,7.8408,80,1,80)
newx <- rbind(x1, x2, x3)
colnames(newx) = colnames(Xobs)
hw4.data = list(x=Xobs, y=Yobs, T = 300, invXp=invXp, newx = newx)
hw4.inits = rep(list(list(pie=c(0.5,0.5,0.5,0.5,0.5,0.5))),5)
hw4.parameters = c("betas", "pie[1:6]", "predPi[1:3]")
hw4.out2 = jags(hw4.data, hw4.inits, hw4.parameters, "Lab3.Posteriors.pred.tx
t", n.chains=5, n.iter=51000, n.burnin=0, n.thin=2, DIC=F)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 300
##    Unobserved stochastic nodes: 6
##    Total graph size: 3046
##
## Initializing model
```

```
Output.hw4 = AddBurnin(hw4.out2$BUGSoutput$sims.array,burnin=1000,n.thin=2)

predPi <- paste0("predPi[", 1:3, "]")
table1 <- cbind(Parameter, round(Output.hw4$Burnin.Summary[predPi, ], 4))
```

```
## Warning in cbind(Parameter, round(Output.hw4$Burnin.Summary[predPi, ], 4))
:
## number of rows of result is not a multiple of vector length (arg 1)
```

```
knitr::kable(table1, caption = "Posterior of prediction of Pi")
```

Posterior of prediction of Pi

	Parameter	mu.vect	sd.vect	2.5%	97.5%	P>0
predPi[1]	Intercept	0.0081	0.0049	0.0019	0.0205	1
predPi[2]	ISS	0.0377	0.0212	0.0093	0.0902	1
predPi[3]	RTS	0.1875	0.1622	0.0122	0.6131	1