LOGISTIC REGRESSION MODEL

FOR OCCUPANCY DETECTION

Deepthi Suresh
S3991481

Table of Contents

Table of Figures

Introduction

The dataset consists of the Occupancy status of a room based on factors like Temperature, Humidity, Light, Humidity Ratio and CO2. The original dataset from UC Irvine Machine Learning Repository consists of 20560 instances and 6 features, however for this study, only 500 instances are considered as training and 21 instances are considered as testing data. The objective of the study is to predict the occupancy status of a room using the other variables in the dataset.

Method

To model the Occupancy status using the other predictors, Robust Logistic Bayesian regression analysis using JAGS Model is conducted. The analysis starts off at descriptive level to identify the mathematical model, which further helps in identifying and setting up the prior distribution. Based on the prior distribution, a suitable Bayesian model is used to conduct the analysis. Diagnostic checking is done on the mean and variance to ensure that the model fits well. Based on the suitability, predictions are made for the occupation status.

Results

1. Descriptive Analysis

The occupation detection dataset taken from the UC Irvine Machine Learning Repository is used to detect the occupancy of an office room based on Temperature, Humidity, Light, Humidity Ratio and CO2 factors. The original dataset consists of 20560 instances and 6 features. However for this study, a subset of 500 instances are considered as training dataset and stored into “myData” variable. The dataset consists of the following variables : "date", "Temperature", "Humidity", "Light", "CO2", "Humidity Ratio", "Occupancy", however the “date” variable will not be taken into consideration. Here, Occupancy is a binary variable with 0 for not occupied, 1 for occupied status.

Figure 1 shows the histogram of the Occupancy variable, it can be noted that the frequency at which the office room is not occupied is higher than the occupied. Since Occupancy is a binary variable, Logistic Regression model would be a good fit for predictions. It is important to note that the original dataset is also as imbalanced as the sample.

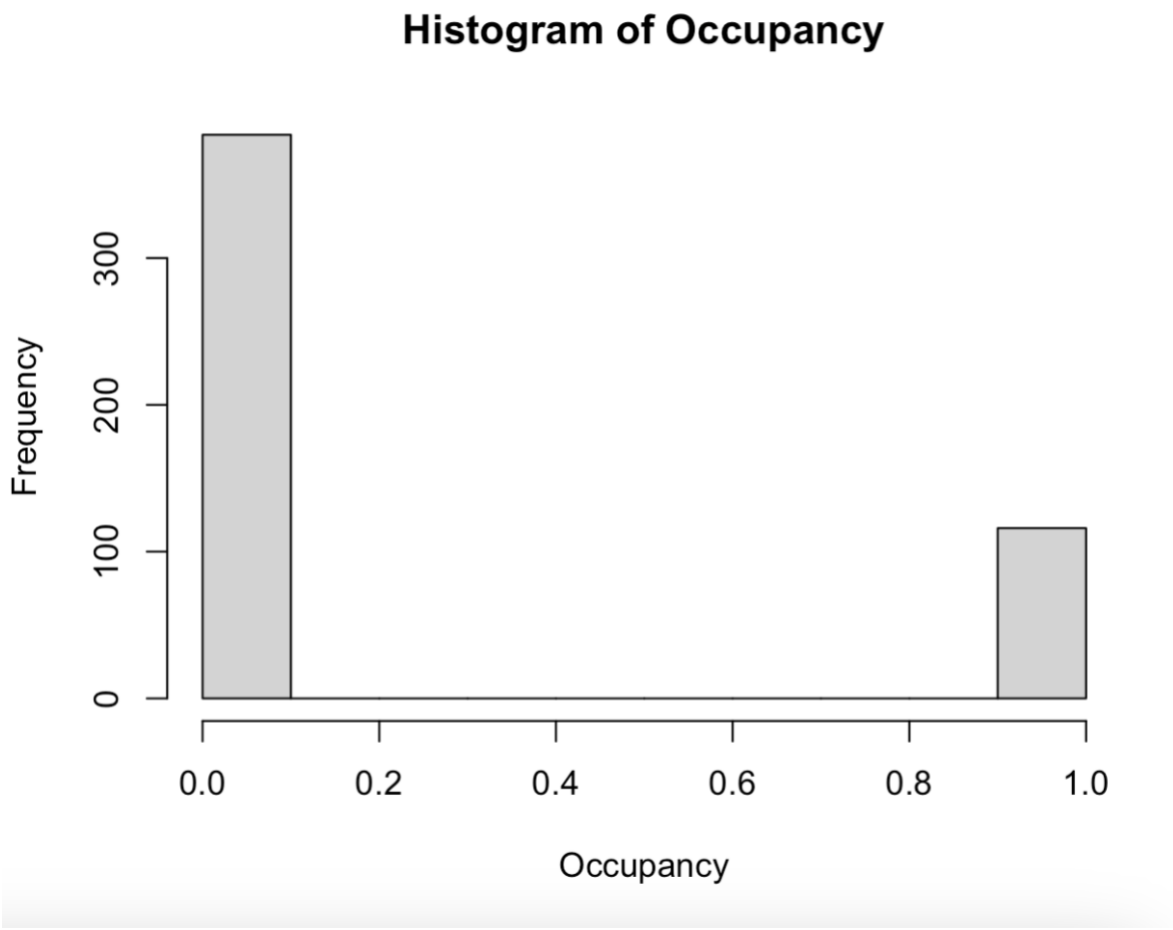

Figure 1: Histogram of Occupancy

Figure 2 shows the scatter plot depicting the relationship between Occupancy and the other predictors in the dataset. In the case of temperature, it looks like occupancy is almost equal irrespective of the temperature, however there is a noticeable gap in between 22 and 23 degrees when occupancy status is 1 . In terms of the light variable, occupancy is higher when the light is higher. Occupancy also tends to be higher with the higher CO2 levels. In terms of Humidity ratio and Humidity, there seems to be an almost equal effect on the occupancy. However, there is one noticeable outlier in both cases

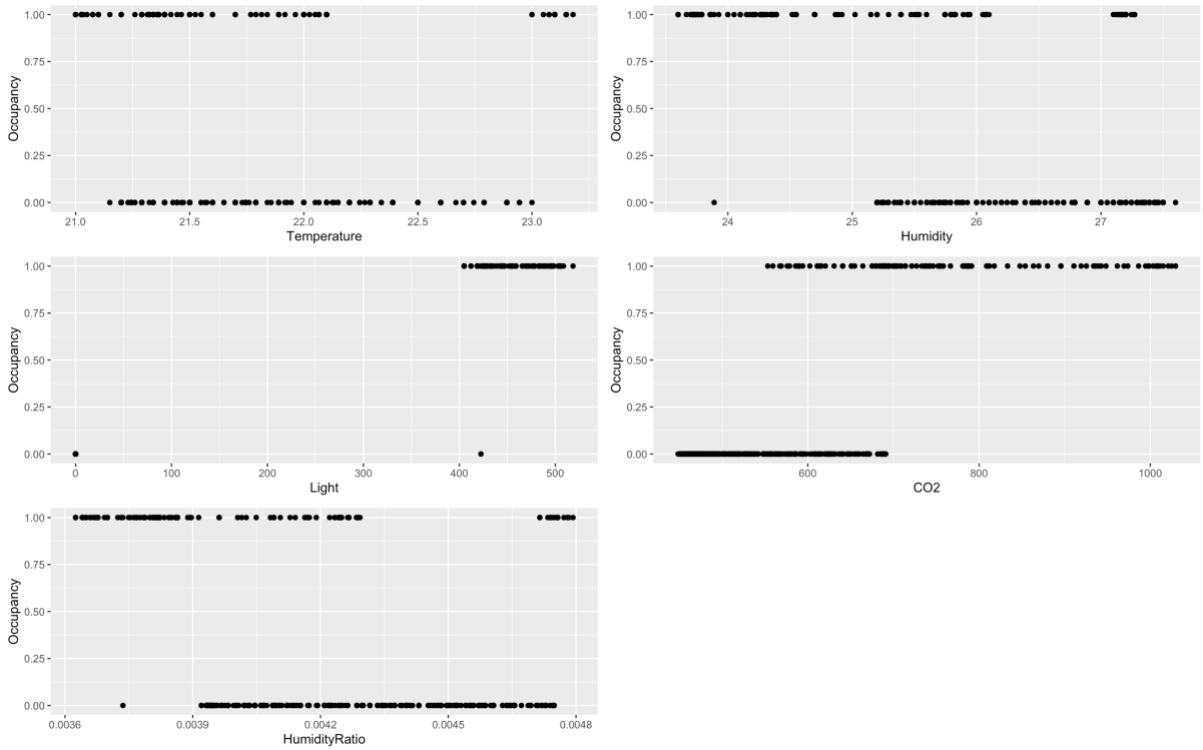

Figure 2: Scatter Plot of the different variables in the dataset

Figure 3 shows the summary statistics of the dataset. it can be noted that there are no missing values in this dataset. Temperature ranges from a minimum of 21 and 23.18, with an average of 21.69. A mean humidity of 25.97 with the median close to this (25.75) suggests a fairly balanced distribution, without many extreme values. A similar situation can be seen with the Humidity ratio as well. There is a really high range for light between 0 and 518.5, with a mean of 104.9, indicating a significant variation. CO2 levels range from a minimum of 448.5 to a maximum of 1029.5, with a mean of 579.3. It can also be noted that except for light, most of these variable have a mean and median that are fairly close to each other indicating a balanced distribution.

```
> summary(myData)
  Temperature       Humidity        Light             CO2         HumidityRatio
 Min.   :21.00   Min.   :23.60   Min.   :  0.0   Min.   : 448.5   Min.   :0.003625
 1st Qu.:21.29   1st Qu.:25.34   1st Qu.:  0.0   1st Qu.: 477.9   1st Qu.:0.003950
 Median :21.50   Median :25.75   Median :  0.0   Median : 528.2   Median :0.004090
 Mean   :21.69   Mean   :25.97   Mean   :104.9   Mean   : 579.3   Mean   :0.004169
 3rd Qu.:22.00   3rd Qu.:27.10   3rd Qu.:  0.0   3rd Qu.: 646.0   3rd Qu.:0.004394
 Max.   :23.18   Max.   :27.60   Max.   :518.5   Max.   :1029.5   Max.   :0.004793
   Occupancy
 Min.   :0.000
 1st Qu.:0.000
 Median :0.000
 Mean   :0.232
 3rd Qu.:0.000
 Max.   :1.000
>
```

Figure 3: Summary statistics of the dataset

2. Mathematical model

$$Y \sim Bernoulli(\theta)$$

θ = logistic(β0 + β1 * temperature + β2 * Humidity + β3 * Light + β4* CO2 + β5 * Humidity Ratio)

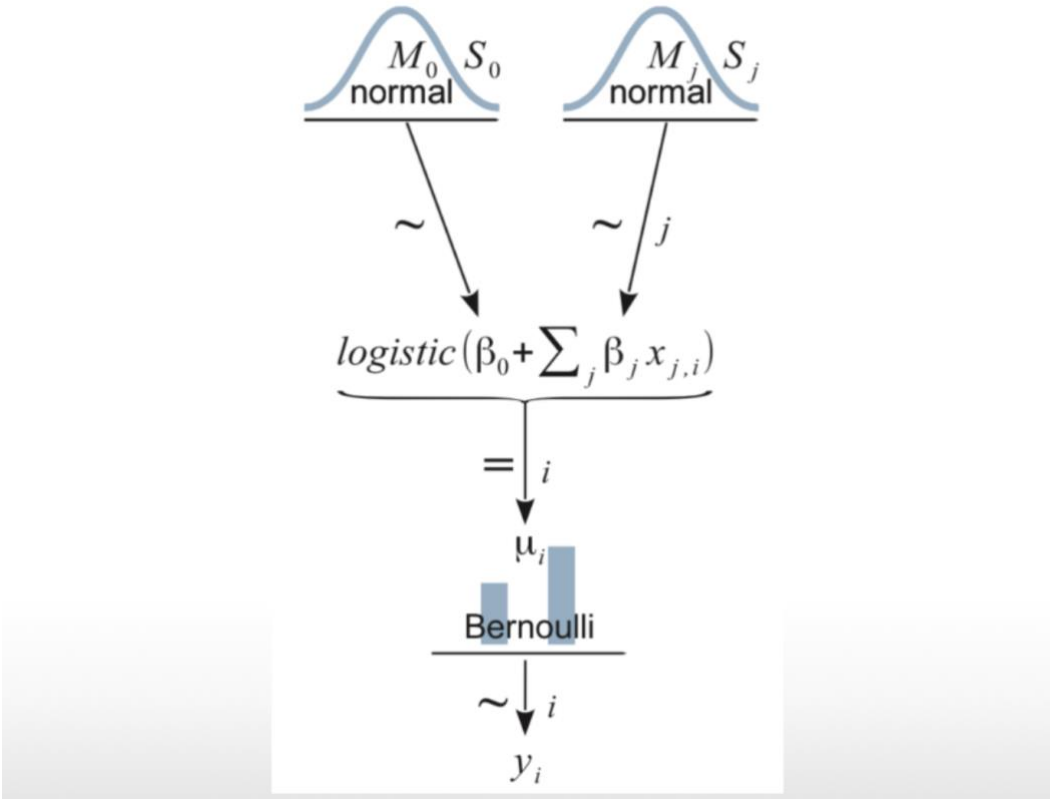

Figure 4: Mathematical model and JAGS model for Bernoulli Distribution

The dependent variable Y representing the Occupation variable, follows a Bernoulli distribution with a success probability of theta. Theta is the probability of occupancy in this application. The logistic model is then placed on top of the success probability theta, with normal priors for the model coefficients.

3. Specify prior distribution

The following questions are answered to determine a suitable prior distribution:

1. **Is the parameter of interest continuous or discrete?** The parameter of interest here is theta which is the success probability and therefore is continuous.
2. **What is the domain of the parameter of interest?** The domain of the parameter lies between 0 and 1
3. **How many parameters do you need to use**? In a logistic regression model, there is typically one parameter per predictor variable plus an intercept term. There are no expert information available in this case.
4. **Is the prior that I induced on the parameter of interest a conjugate prior? If not, is there a conjugate prior available**? In logistic regression, a conjugate prior does not exist for the Bernoulli likelihood

5. **Is the product of likelihood and prior mathematically tractable?** The posterior for logistic regression with a Bernoulli likelihood and normal priors on the coefficients is not mathematically tractable.

4. Finding the posterior distribution

In order to develop predictions, 21 unseen observation has been selected as a testing data. Multiple test runs were completed to find a diagnostic that was acceptable. Given below is the diagnostics for a successful run after several trial and errors.

4.1 Diagnostic checking for the parameter

The run is based on the following settings:

adaptSteps = 3500
burnInSteps = 20000
nChains = 2
thinSteps = 15
numSavedSteps = 6000

The figures below show the Density plots, Shrink Factor, Auto Correlations and Burn-in plot for the standardised Beta and Predictors. Standardised Beta is used to make the MCMC more efficient by scaling the data and then transforming the parameters back to the original scale. MCMC representation is checked through the Burn-in plot, Density plots and Shrink Factor and the MCMC Accuracy is checked through the Autocorrelation plot, ESS and MCSE.

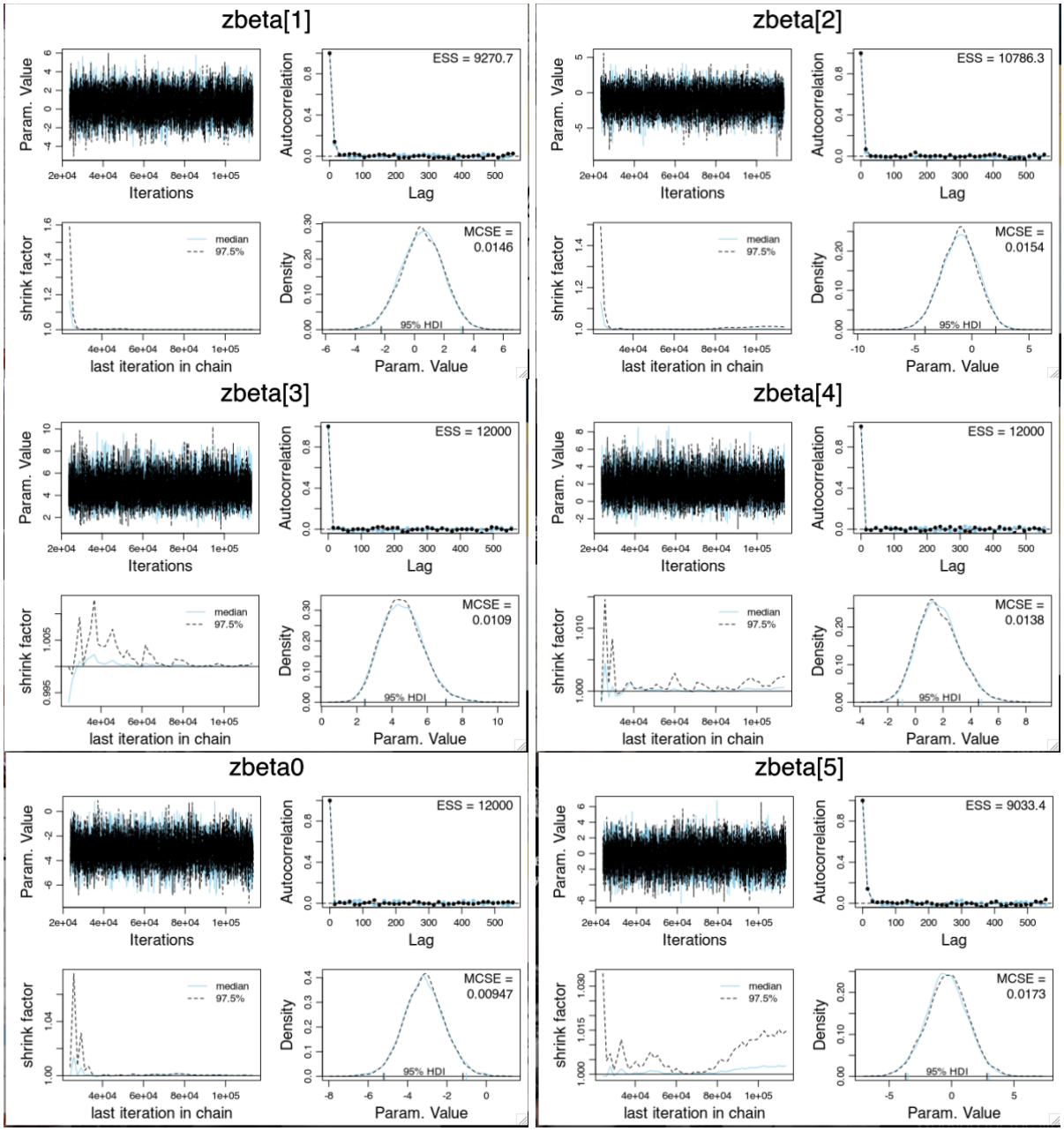

Figure 5: Diagnostic checking

In Figure 5, The trace plots for all standardised Beta on the top left shows that all the chains are around a common mean value and mixed in well, therefore looks like the chains have converged. Density plot on the bottom right, shows that the chain overlap in most places for all the standardised Beta values. However, it is not a perfect fit, suggesting that the chains are producing representative values from the posterior distribution. Another criterion to assess the MCMC representativeness is the Shrink Factor. Ideally you want the shrink factor to be less than 1.2, which is the case in all the Beta values and therefore we can say that the chains have converged.

In terms of accuracy, the autocorrelation plot shows a sudden drop and all further autocorrelations around 0 for Betas which indicates that there are no autocorrelations. Another measure is the ESS which is between 9000 an 12000 which is a really good indicator of accuracy. The third measure is the MCSE, which is the error value. Ideally, we want a

value closer to 0. In this case the MCSE for all beta values are very close to 0 and therefore demonstrates good accuracy.

In terms of efficiency (as shown in Figure 6) the model took 172.488 seconds or 2.8 minutes to run, which indicates that the model is efficient.

```
> show(duration)
   user  system elapsed
  0.859   0.393 172.488
```

Figure 6: Duration of the run

4.2 Diagnostic checking for the predictors

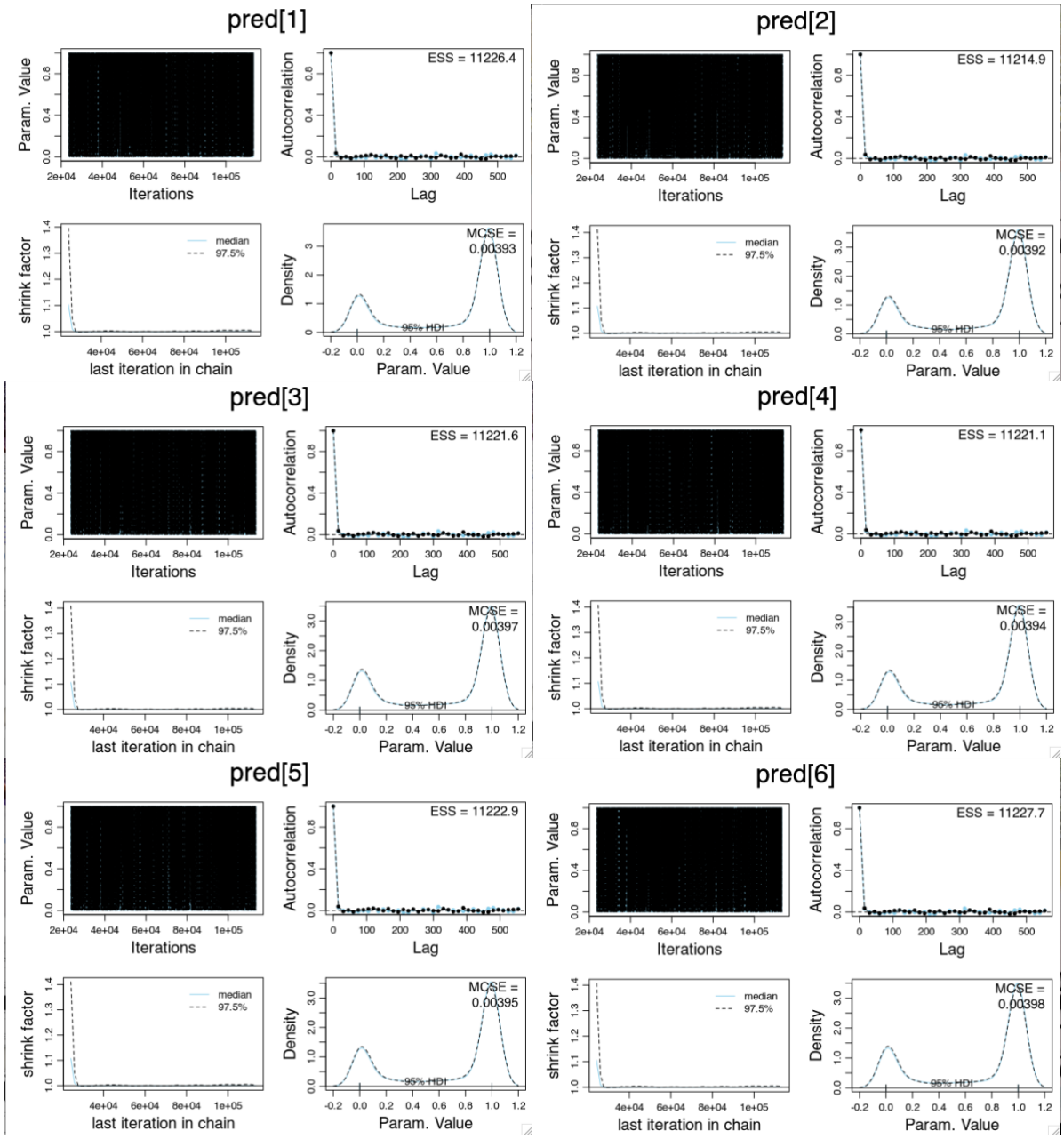

Figure 7: Diagnostic Checking for the predictors

For this study, we have taken 21 unseen observations as predictors, however the diagnostics for only 6 of those are displayed here (similar results are seen in the other predictors as well). In Figure 7, The trace plots on the top left for all the predictors shows that all the chains are around a common mean value and mixed in well, therefore looks like the chains have converged. Density plot on the bottom right for all the predictors, shows that the chain overlap in most places, however it is not a perfect fit. Shrink Factor is less than 1.2 for all the predictors; MCSE is very close to 0 and ESS is high for all the predictors. The autocorrelation plot shows a sudden drop and all further autocorrelations around 0 which indicates that there are no autocorrelations. The MCMC setting seems to be both representative and accurate for all the predictors.

5. Interpretation of the posteriors

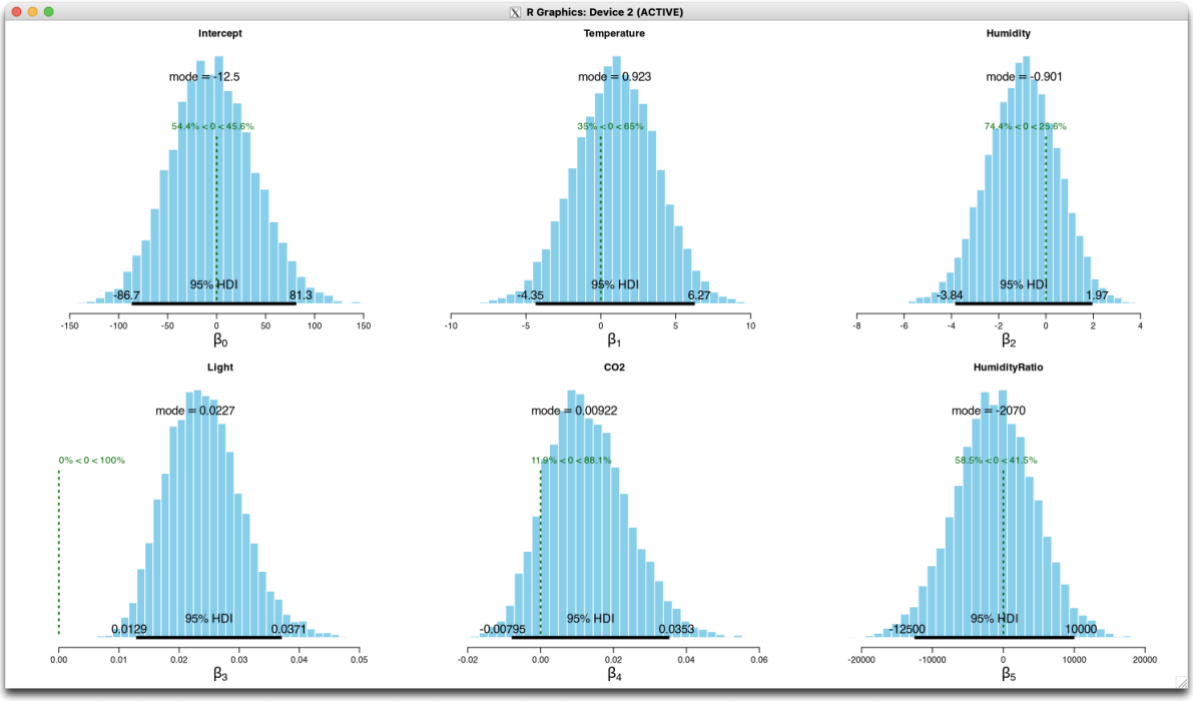

Figure 8: Interpretation of posteriors

The posteriors for the run is given in Figure 8. To understand if the predictors are significant or not we look at whether 0 is a part of the HDI interval. In this case, except for light , 0 seems to be part of all the plots indicating that all the predictors have an insignificant coefficients. However, CO2 could be considered a significant coefficient as 88.1% of the observations lie above 0. Similarly, in terms of humidity 74.4 % of the observations lie below 0 and for temperature 65% of the observation lies above 0.

A predictive check has also been conducted to check the accuracy of the model. The threshold considered here for this purpose is 0.5, i.e. if the probability is less than 0.5, it is classified as 0 or not occupied else it is classified as 1 or occupied. Figure 9 shows the prediction for the observations in the testing class

```
> BayesTheta
    pred[1]     pred[2]     pred[3]     pred[4]     pred[5]     pred[6]     pred[7]     pred[8]
0.992926585 0.992476199 0.994108643 0.993378210 0.993586460 0.994723440 0.994031947 0.992838988
    pred[9]    pred[10]    pred[11]    pred[12]    pred[13]    pred[14]    pred[15]    pred[16]
0.992360653 0.993123705 0.001300363 0.001294193 0.001309049 0.001321576 0.001287874 0.001275318
   pred[17]    pred[18]    pred[19]    pred[20]    pred[21]
0.001330924 0.001292675 0.001259348 0.001268203 0.001270852
> BayesClass
 pred[1]  pred[2]  pred[3]  pred[4]  pred[5]  pred[6]  pred[7]  pred[8]  pred[9] pred[10]
       1        1        1        1        1        1        1        1        1        1
pred[11] pred[12] pred[13] pred[14] pred[15] pred[16] pred[17] pred[18] pred[19] pred[20]
       0        0        0        0        0        0        0        0        0        0
pred[21]
       0
```

Figure 9: Predictions for the testing class

These predictions are then compared with the actual class. The confusion matrix in Figure 10 shows that all 21 of the observations have been correctly predicted, which has resulted in accuracy, precision, recall and f1 to be 1.

```
> conf
          response
predicted  0  1
        0 11  0
        1  0 10
> accuracy
[1] 1
> precision
[1] 1
> recall
[1] 1
> f1
[1] 1
```

Figure 10: Predictive check

Figure 11 shows the calculation of Exp(Beta) and 1/exp(Beta) to interpret the coefficients of a Logistic Model.

		Mean	Median	Mode	EXP(mode)	1/EXP(mode)	ESS	HDImass	HDIlow	HDIhigh
Intercept	beta0	-4.1728165	-5.18607	-12.49941	3.7289E-06	268179.054	9531.8	0.95	-86.73	81.309
Temperature	beta[1]	1.02408698	1.04025	0.92304784	2.51694998	0.39730627	9265.3	0.95	-4.35411	6.27414
Humidity	beta[2]	-1.0036135	-0.9725625	-0.9008392	0.40622862	2.461668	10778.5	0.95	-3.83992	1.96684
Light	beta[3]	0.0241246	0.02379295	0.02268414	1.02294338	0.97757121	12000	0.95	0.012855	0.0370857
CO2	beta[4]	0.01277852	0.01191755	0.00921657	1.00925917	0.99082577	12000	0.95	-0.0079476	0.0353446
Humidity ratio	beta[5]	-1213.7936	-1216.215	-2071.4461	0	#DIV/0!	9028.7	0.95	-12541.5	10022.5

Figure 11:Interpretation of the coefficients

With every unit increase in temperature, the likelihood of room occupancy is 2.5 times the likelihood of non occupancy. This suggests a strong positive relationship between temperature and the likelihood of occupancy, implying that higher temperatures may be associated with occupied spaces. With every unit increase in Humidity, the likelihood of

room non-occupancy is 2.46 times the likelihood of occupancy. This negative relationship suggests that higher humidity levels may be associated with unoccupied spaces. With every unit increase in light, the likelihood of room occupancy is 1.02 times the likelihood of non occupancy. With every unit increase in CO2, the likelihood of room occupancy is 1.009 times the likelihood of non occupancy. In both these cases, since the odds ratio is so close to 1, it could also be interpreted that light and CO2 have no association with the occupancy of a room. In other words, the odds of occupancy given there is light or CO2 is the same as the odds of occupancy given there is no light or CO2. The odds ratio for Humidity ratio is 0, indicating that as the humidity ratio increases the likelihood of occupancy approaches 0.

References

- Dr. Demirhan (2024) 'Logistic Regression' [PowerPoint slides, MATH2269], RMIT University, Melbourne
- Goodwin, G., & Ryu, S. Y. (2023). Understanding the odds: Statistics in public health. *Frontiers for Young Minds, 10*. https://doi.org/10.3389/frym.2022.926624
- *UCI Machine Learning Repository*. (n.d.). https://archive.ics.uci.edu/dataset/357/occupancy+detection

Appendix

```
graphics.off() # This closes all of R's graphics windows.
rm(list=ls())  # Careful! This clears all of R's memory!
source("DBDA2E-utilities.R")
library(ggplot2)
library(ggpubr)
library(ks)
library(rjags)
library(runjags)
library(benchmarkme)
setwd("~/OneDrive - RMIT University/Bayesian Statistics/Assignemnt 3")

#===============PRELIMINARY FUNCTIONS FOR POSTERIOR INFERENCES====================

smryMCMC = function(  codaSamples , compVal = NULL,  saveName=NULL) {
  summaryInfo = NULL
  mcmcMat = as.matrix(codaSamples,chains=TRUE)
  paramName = colnames(mcmcMat)
  for ( pName in paramName ) {
    if (pName %in% colnames(compVal)){
      if (!is.na(compVal[pName])) {
```

```
      summaryInfo = rbind( summaryInfo , summarizePost( paramSampleVec =
mcmcMat[,pName] ,
                                          compVal = as.numeric(compVal[pName]) ))
    }
    else {
      summaryInfo = rbind( summaryInfo , summarizePost( paramSampleVec =
mcmcMat[,pName] ) )
    }
  } else {
    summaryInfo = rbind( summaryInfo , summarizePost( paramSampleVec =
mcmcMat[,pName] ) )
  }
 }
 rownames(summaryInfo) = paramName

 # summaryInfo = rbind( summaryInfo ,
 #                      "tau" = summarizePost( mcmcMat[,"tau"] ) )
 if ( !is.null(saveName) ) {
  write.csv( summaryInfo , file=paste(saveName,"SummaryInfo.csv",sep="") )
 }
 return( summaryInfo )
}

#=============================================================================
==============

plotMCMC = function( codaSamples , data , xName="x" , yName="y", preds = FALSE ,
                     showCurve=FALSE ,  pairsPlot=FALSE , compVal = NULL,
                     saveName=NULL , saveType="jpg" ) {
 # showCurve is TRUE or FALSE and indicates whether the posterior should
 #   be displayed as a histogram (by default) or by an approximate curve.
 # pairsPlot is TRUE or FALSE and indicates whether scatterplots of pairs
 #   of parameters should be displayed.
 #-----------------------------------------------------------------------------
 y = data[,yName]
 x = as.matrix(data[,xName])
 mcmcMat = as.matrix(codaSamples,chains=TRUE)
 chainLength = NROW( mcmcMat )
 zbeta0 = mcmcMat[,"zbeta0"]
 zbeta  = mcmcMat[,grep("^zbeta$|^zbeta\\[",colnames(mcmcMat))]
 if ( ncol(x)==1 ) { zbeta = matrix( zbeta , ncol=1 ) }
 beta0 = mcmcMat[,"beta0"]
 beta  = mcmcMat[,grep("^beta$|^beta\\[",colnames(mcmcMat))]
 if ( ncol(x)==1 ) { beta = matrix( beta , ncol=1 ) }
 if (preds){
  pred = mcmcMat[,grep("^pred$|^pred\\[",colnames(mcmcMat))]
 } # Added by Demirhan

 #-----------------------------------------------------------------------------
 # Compute R^2 for credible parameters:
```

```
YcorX = cor( y , x ) # correlation of y with each x predictor
Rsq = zbeta %*% matrix( YcorX , ncol=1 )
#-----------------------------------------------------------------------------
if ( pairsPlot ) {
  # Plot the parameters pairwise, to see correlations:
  openGraph()
  nPtToPlot = 1000
  plotIdx = floor(seq(1,chainLength,by=chainLength/nPtToPlot))
  panel.cor = function(x, y, digits=2, prefix="", cex.cor, ...) {
    usr = par("usr"); on.exit(par(usr))
    par(usr = c(0, 1, 0, 1))
    r = (cor(x, y))
    txt = format(c(r, 0.123456789), digits=digits)[1]
    txt = paste(prefix, txt, sep="")
    if(missing(cex.cor)) cex.cor <- 0.8/strwidth(txt)
    text(0.5, 0.5, txt, cex=1.25 ) # was cex=cex.cor*r
  }
  pairs( cbind( beta0 , beta , tau )[plotIdx,] ,
         labels=c( "beta[0]" ,
                   paste0("beta[",1:ncol(beta),"]\n",xName) ,
                   expression(tau) ) ,
         lower.panel=panel.cor , col="skyblue" )
  if ( !is.null(saveName) ) {
    saveGraph( file=paste(saveName,"PostPairs",sep=""), type=saveType)
  }
}
#-----------------------------------------------------------------------------
# Marginal histograms:

decideOpenGraph = function( panelCount , saveName , finished=FALSE ,
                            nRow=2 , nCol=3 ) {
  # If finishing a set:
  if ( finished==TRUE ) {
    if ( !is.null(saveName) ) {
      saveGraph( file=paste0(saveName,ceiling((panelCount-1)/(nRow*nCol))),
                 type=saveType)
    }
    panelCount = 1 # re-set panelCount
    return(panelCount)
  } else {
    # If this is first panel of a graph:
    if ( ( panelCount %% (nRow*nCol) ) == 1 ) {
      # If previous graph was open, save previous one:
      if ( panelCount>1 & !is.null(saveName) ) {
        saveGraph( file=paste0(saveName,(panelCount%/%(nRow*nCol))),
                   type=saveType)
      }
      # Open new graph
      openGraph(width=nCol*7.0/3,height=nRow*2.0)
      layout( matrix( 1:(nRow*nCol) , nrow=nRow, byrow=TRUE ) )
```

```
      par( mar=c(4,4,2.5,0.5) , mgp=c(2.5,0.7,0) )
    }
    # Increment and return panel count:
    panelCount = panelCount+1
    return(panelCount)
  }
}

# Original scale:
panelCount = 1
panelCount = decideOpenGraph( panelCount , saveName=paste0(saveName,"PostMarg") )
histInfo = plotPost( beta0 , cex.lab = 1.75 , showCurve=showCurve ,
                     xlab=bquote(beta[0]) , main="Intercept", compVal =
as.numeric(compVal["beta0"] ))
for ( bIdx in 1:ncol(beta) ) {
  panelCount = decideOpenGraph( panelCount , saveName=paste0(saveName,"PostMarg") )
  if (!is.na(compVal[paste0("beta[",bIdx,"]")])){
    histInfo = plotPost( beta[,bIdx] , cex.lab = 1.75 , showCurve=showCurve ,
                         xlab=bquote(beta[.(bIdx)]) , main=xName[bIdx],
                         compVal = as.numeric(compVal[paste0("beta[",bIdx,"]")]))
  } else{
    histInfo = plotPost( beta[,bIdx] , cex.lab = 1.75 , showCurve=showCurve ,
                         xlab=bquote(beta[.(bIdx)]) , main=xName[bIdx])
  }
}
panelCount = decideOpenGraph( panelCount , saveName=paste0(saveName,"PostMarg") )
histInfo = plotPost( Rsq , cex.lab = 1.75 , showCurve=showCurve ,
                     xlab=bquote(R^2) , main=paste("Prop Var Accntd") , finished=TRUE )

panelCount = 1
if ( pred){

  for ( pIdx in 1:ncol(pred) ) {
    panelCount = decideOpenGraph( panelCount , saveName=paste0(saveName,"PostMarg")
)
    histInfo = plotPost( pred[,pIdx] , cex.lab = 1.75 , showCurve=showCurve ,
                         xlab=bquote(pred[.(pIdx)]) , main=paste0("Prediction ",pIdx) )
  }
}# Added by Demirhan
# Standardized scale:
panelCount = 1
panelCount = decideOpenGraph( panelCount , saveName=paste0(saveName,"PostMargZ")
)
histInfo = plotPost( zbeta0 , cex.lab = 1.75 , showCurve=showCurve ,
                     xlab=bquote(z*beta[0]) , main="Intercept" )
for ( bIdx in 1:ncol(beta) ) {
  panelCount = decideOpenGraph( panelCount , saveName=paste0(saveName,"PostMargZ")
)
  histInfo = plotPost( zbeta[,bIdx] , cex.lab = 1.75 , showCurve=showCurve ,
                       xlab=bquote(z*beta[.(bIdx)]) , main=xName[bIdx] )
```

```
  }
  panelCount = decideOpenGraph( panelCount , saveName=paste0(saveName,"PostMargZ")
)
  histInfo = plotPost( Rsq , cex.lab = 1.75 , showCurve=showCurve ,
                       xlab=bquote(R^2) , main=paste("Prop Var Accntd") )
  panelCount = decideOpenGraph( panelCount , finished=TRUE ,
saveName=paste0(saveName,"PostMargZ") )

  #------------------------------------------------------------------------
}

#===============PRELIMINARY FUNCTIONS FOR POSTERIOR
INFERENCES====================

#Subset of 500 observations have been considered as the training data
myData <- read.table("datatraining copy.txt", sep = ",")
colnames(myData)
myData$Occupancy <- as.numeric(as.factor(myData$Occupancy)) - 1 # converting to a
factor variable and back to numerical since it is a categorical variable
myData <- myData[ -c(1) ] # Dropping the date column
head(myData)

# Prepare the data
y = myData[,"Occupancy"]
x = as.matrix(myData[,1:5])

#------------------------------------------------------------------------------
## Descriptive Analysis ##
#------------------------------------------------------------------------------

# Histogram
hist(myData$Occupancy, xlab = "Occupancy", main = "Histogram of Occupancy")


# Scatter plots
p1 <- ggplot(myData, aes(x=Temperature, y=Occupancy)) +
  geom_point() #Temperature

p2 <- ggplot(myData, aes(x=Humidity, y=Occupancy)) +
  geom_point() #Humidity

p3 <- ggplot(myData, aes(x=Light, y=Occupancy)) +
  geom_point() #Light

p4 <- ggplot(myData, aes(x=CO2, y=Occupancy)) +
  geom_point() #CO2

p5 <- ggplot(myData, aes(x=HumidityRatio, y=Occupancy)) +
  geom_point() #HumidityRatio
```

```
figure <- ggarrange(p1, p2, p3, p4, p5, nrow = 3, ncol = 2)
figure

summary(myData)

PredData= read.table("datatest2 copy 2.txt", sep = ",") #Only 21 observations considerd for
prediction in testing data
xPred = as.matrix(PredData[,2:6])

# Specify the data in a list, for later shipment to JAGS:
dataList <- list(
  x = x ,
  y = y ,
  xPred = xPred ,
  Ntotal = length(y),
  Nx = ncol(x),
  Npred = nrow(xPred)
)

# First run without initials!
initsList <- list(
  beta0 = 0,
  beta1 = 0,
  beta2 = 0,
  beta3 = 0,
  beta4 = 0,
  beta5 = 0
)

modelString = "
data {
  for ( j in 1:Nx ) {
    xm[j]  <- mean(x[,j])
    xsd[j] <-   sd(x[,j])
    for ( i in 1:Ntotal ) {
      zx[i,j] <- ( x[i,j] - xm[j] ) / xsd[j]
    }
  }
}

model {
  # Non-informative run with standardisation
  for ( i in 1:Ntotal ) {
    # In JAGS, ilogit is logistic:
    y[i] ~ dbern( mu[i] )
      mu[i] <- ( guess*(1/2) + (1.0-guess)*ilogit(zbeta0+sum(zbeta[1:Nx]*zx[i,1:Nx])) )
  }
  # Priors vague on standardized scale:
  zbeta0 ~ dnorm( 0 , 1/2^2 )
```

```
  # non-informative run
  for ( j in 1:Nx ) {
    zbeta[j] ~ dnorm( 0 , 1/2^2 )
  }
  #Transform to original scale:
  beta[1:Nx] <- zbeta[1:Nx] / xsd[1:Nx]
  beta0 <- zbeta0 - sum( zbeta[1:Nx] * xm[1:Nx] / xsd[1:Nx] )

  guess ~ dbeta(1,9)

  # Compute predictions at every step of the MCMC
  for ( k in 1:Npred){
    pred[k] <- ilogit(beta0 + sum(beta[1:Nx] * xPred[k,1:Nx]))
  }
}
" # close quote for modelString
# Write out modelString to a text file
writeLines( modelString , con="TEMPmodel.txt" )

# Alternatively you can use a for loop if you have many parameters
parameters = c( "zbeta0" , "beta0")
for ( i in 1:5){
  parameters = c(parameters, paste0("zbeta[",i,"]"), paste0("beta[",i,"]"))
}

for ( i in 1:nrow(xPred)){
  parameters = c(parameters, paste0("pred[",i,"]"))
}

adaptSteps = 3500  # Number of steps to "tune" the samplers
burnInSteps = 20000
nChains = 2
thinSteps = 15
numSavedSteps = 6000
nIter = ceiling( ( numSavedSteps * thinSteps ) / nChains )

# startTime = proc.time()
# runJagsOut <- run.jags( method="parallel" ,
#                     model="TEMPmodel.txt" ,
#                     monitor=parameters  ,
#                     data=dataList ,
#                     # inits=initsList ,
#                     n.chains=nChains ,
#                     adapt=adaptSteps ,
#                     burnin=burnInSteps ,
#                     sample=numSavedSteps ,
#                     thin=thinSteps , summarise=FALSE , plots=FALSE )
# stopTime = proc.time()
duration = stopTime - startTime
show(duration)
```

```
#save.image(file='RobustL_Run2-21testdata_500_2chains_3.5
adapt_6k_saved_20kburnin_15Thin.RData') #Save the run
load(file='RobustL_Run2-21testdata_500_2chains_3.5
adapt_6k_saved_20kburnin_15Thin.RData')

get_cpu() # To assess efficiency.

codaSamples = as.mcmc.list( runJagsOut )

diagMCMC( codaSamples , parName="beta0" )
for ( i in 1:5){
  diagMCMC( codaSamples , parName=paste0("beta[",i,"]") )
}
diagMCMC( codaSamples , parName="zbeta0" )
for ( i in 1:5){
  diagMCMC( codaSamples , parName=paste0("zbeta[",i,"]") )
}
for ( i in 1:nrow(xPred)){
  diagMCMC( codaSamples , parName=paste0("pred[",i,"]") )
}

graphics.off()

compVal <- data.frame("beta0" = 0, "beta[1]" = 0, "beta[2]" = 0, "beta[3]" = 0, "beta[4]" =  0,
"beta[5]" =  0, check.names=FALSE)

summaryInfo <- smryMCMC( codaSamples = codaSamples , compVal = compVal)
print(summaryInfo)
write.csv(summaryInfo,file = "SummaryInfo.csv")

colnames(myData)

plotMCMC( codaSamples = codaSamples , data = myData,
xName=c("Temperature","Humidity","Light","CO2","HumidityRatio") ,
        yName="Occupancy", compVal = compVal, preds = TRUE)

# ============ Predictive check ============

BayesTheta = summaryInfo[14:34,3] # Take the Bayesian estimates for the test set
# Classify based on the threshold of 0.5
BayesClass = BayesTheta
BayesClass[which(BayesTheta < 0.5)] = 0
BayesClass[which(BayesTheta > 0.5)] = 1
# Actual  occupancy
ActualClass = as.numeric(as.factor(PredData$Occupancy)) - 1

# Confusion matrix
classRes <- data.frame(response = ActualClass, predicted = BayesClass)
conf = xtabs(~ predicted + response, data = classRes)
```

```
conf

accuracy = sum(diag(conf))/sum(conf)
accuracy
precision = conf[1,1]/(conf[1,1]+conf[1,2])
precision
recall = conf[1,1]/(conf[1,1]+conf[2,1])
recall
f1 = 2 * precision * recall / (precision + recall)
f1
```