BAYESIAN ANALYSIS OF MELBOUNR PROPERTY PRICES

Assignment 2

Deepthi Suresh
S3991481

Table of Contents

Table of Figures **2**

Introduction **3**

Method **3**

Results **3**

1. **Descriptive Analysis** **3**
2. **Mathematical model** **5**
3. **Specify prior distribution** **6**
4. **Finding the posterior distribution** **7**
   - 4.1 Diagnostic Checking for Mean in Run 1 8
   - 4.2 Diagnostic Checking for Variance in Run 1 9
   - 4.3 Diagnostic Checking for predictors in Run 1 10
   - 4.4 Diagnostic Checking for Mean in Run 2 12
   - 4.5 Diagnostic Checking for Variance in Run 2 14
   - 4.6 Diagnostic Checking for predictors in Run 2 14
5. **Interpretation of the posteriors** **16**
   - 5.1 Posterior Predictive checks 17

References ***18***

Appendix ***18***

Table of Figures

Figure 1: Transformation of data type of the Property type variable..3
Figure 2: Scatter plot of the sales Variable with the other predictors 4
Figure 3: Histogram and Density plot of Sale Price 4
Figure 4:Summary of the dataset 5
Figure 5: Correlation Matrix of the predictors..5
Figure 6 JAGS Model Diagram 7
Figure 7:Predictors for the sale prices 8
Figure 8: Diagnostic checking for standardised Beta 9
Figure 9 Diagnostic checking for Tau..10
Figure 10:Diagnostic Checking for Predictors 11
Figure 11MCMC efficiency of Run 1 12
Figure 12: Standardised Beta diagnostic checking 13
Figure 13 diagnostic checking for Tau in Run 2 14
Figure 14: Diagnostic checking for predictors in run 2 15
Figure 15:Interpretation of Posteriors 16
Figure 16: Point estimates of the prediction 17
Figure 17:Predictive checks with the observed value...18

Introduction

The dataset consists of the sales prices of properties in Melbourne in 100,000 of AUD, along with the features of these properties such as the number of bedrooms, no of bathrooms, no of car parks, the area of the property and the Property Type. The objective of the project is to model the sale prices in Melbourne using the other predictors given in the dataset and expert knowledge from a real estate agent

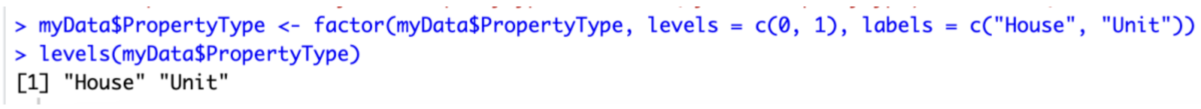
Method

To model the sale prices in Melbourne using the other predictors, Bayesian 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 property prices.

Results

1. Descriptive Analysis

The data consists of information regarding two property types – House and Unit. These have been converted into a factor variable once the data is loaded into R.



```
> myData$PropertyType <- factor(myData$PropertyType, levels = c(0, 1), labels = c("House", "Unit"))
> levels(myData$PropertyType)
[1] "House" "Unit"
```

Figure 1: Transformation of data type of the Property type variable

A scatter plot is used to look at the relationship between the Melbourne property prices and other predictors such as Area, Bedrooms, Bathrooms, Car parks. Figure 2 shows that there are no identifiable relationships between Sale prices and the Bedrooms, Bathrooms, Car Parks and property types. However a few interesting patterns can be noticed:

- Bedrooms: Houses are concentrated on the right side of the chart compares to the units, indicating that houses have more no of bedroom than units. There are outliers in both houses and units
- Bathrooms: There are 3 outliers in units however houses have multiple outliers. There are very few units with 4 bathrooms.
- Car Parks: There are 5 outliers in units however houses have multiple outliers. Houses tend to have more carparks than units
- Property type: houses tend to have higher sale price than units

In case of the relationship between Area an Sale price, there seems to be a positive relationship between the area and the sale prices of houses, however there doesn't seem to be any identifiable relationship for Units.

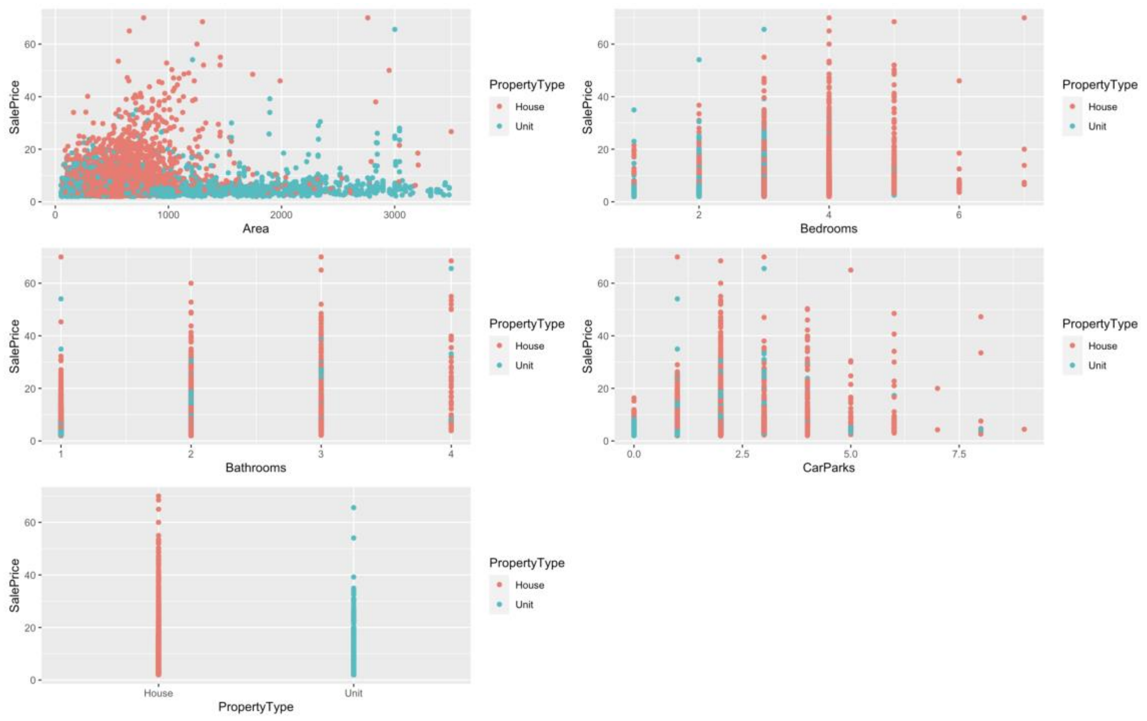


Figure 2: Scatter plot of the sales Variable with the other predictors

Figure 3 shows the histogram and density plot of the Sales prices. It is noticeable that the data is not symmetric and has a long right tail. This means that we can use a Gamma likelihood and perform a Gamma regression

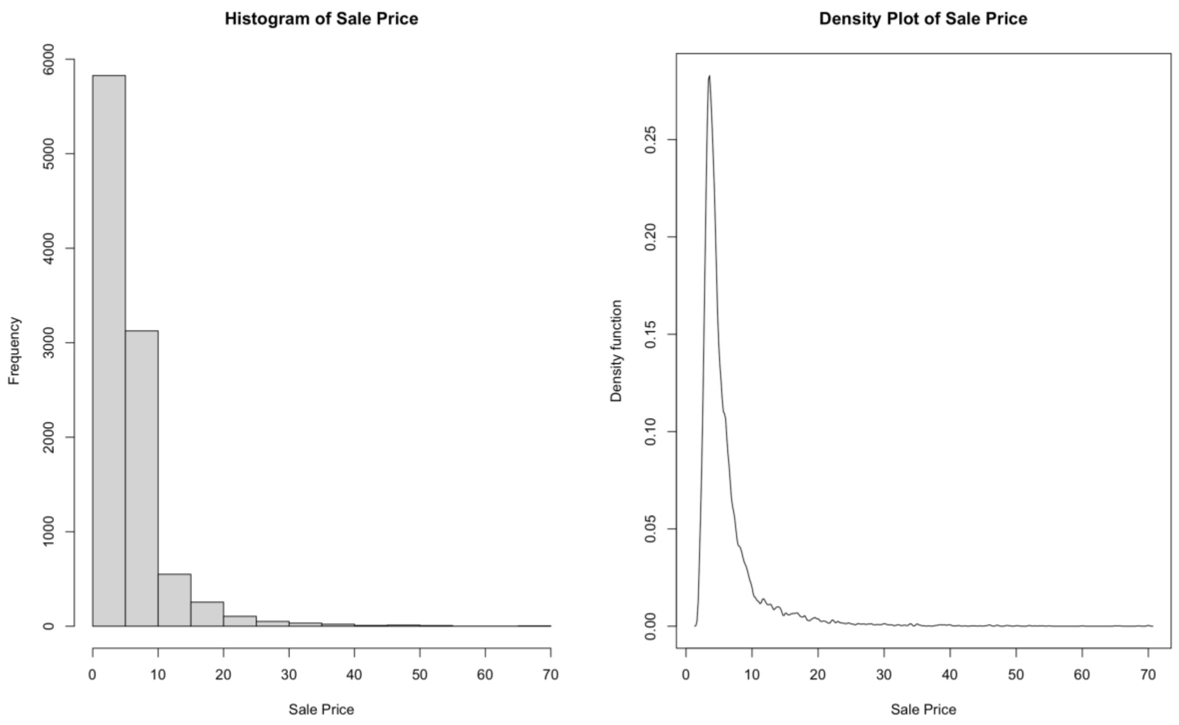
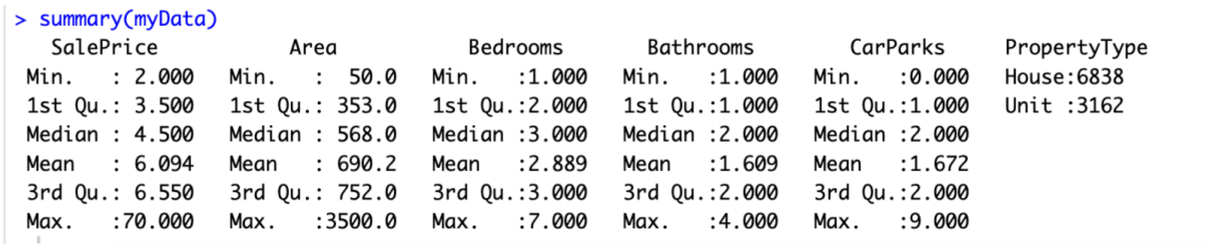


Figure 3: Histogram and Density plot of Sale Price

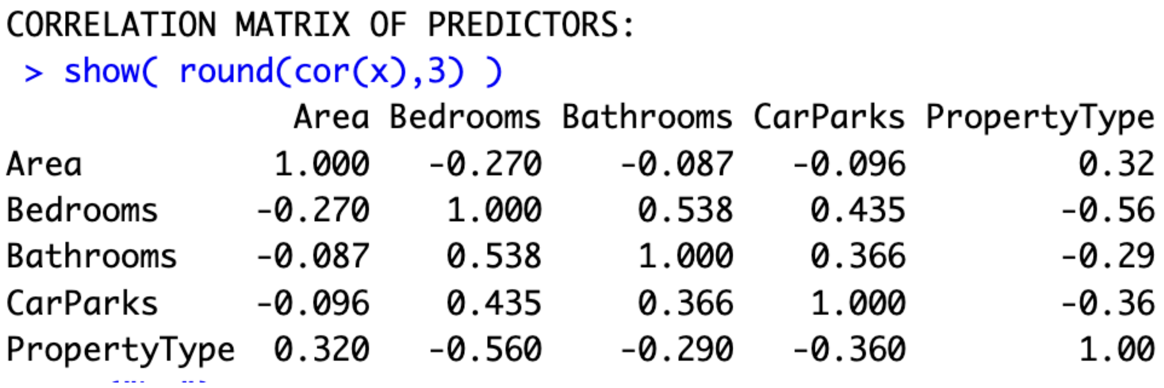
Figure 4 shows the summary statistics of the dataset. The minimum price of a properties in Melbourne is 200K, while the Max is 7000K, which shows a wide range. The average price of a property is 609K, while the median value is 450K. In terms of the area of the properties, the minimum area is 50 m^2 and maximum is 3500 m^2 which is a very wide range. The mean and median is close to each other at 690.2 and 568 m^2, indicating that it could be a symmetric distribution. The properties have minimum of 1 bedroom and bathroom and a maximum of 7 and 4 respectively. On an average the properties have 2.8 bedrooms (rounder to 3) and 1.6 bathrooms (rounded to 2). The minimum number of car parks that each property has is 0 and maximum is 9, with an average of 1.6. All these predictors seems to have a close mean and median value indicating a symmetric distribution. The dataset consists of 6838 houses and 3162 units.



```
> summary(myData)
   SalePrice           Area           Bedrooms        Bathrooms        CarParks      PropertyType
 Min.   : 2.000   Min.   :  50.0   Min.   :1.000   Min.   :1.000   Min.   :0.000   House:6838
 1st Qu.: 3.500   1st Qu.: 353.0   1st Qu.:2.000   1st Qu.:1.000   1st Qu.:1.000   Unit :3162
 Median : 4.500   Median : 568.0   Median :3.000   Median :2.000   Median :2.000
 Mean   : 6.094   Mean   : 690.2   Mean   :2.889   Mean   :1.609   Mean   :1.672
 3rd Qu.: 6.550   3rd Qu.: 752.0   3rd Qu.:3.000   3rd Qu.:2.000   3rd Qu.:2.000
 Max.   :70.000   Max.   :3500.0   Max.   :7.000   Max.   :4.000   Max.   :9.000
```

Figure 4:Summary of the dataset

Figure 5 shows the correlations between the predictors. Bedrooms and Bathrooms have the highest positive correlation. Property types and bedrooms have the largest negative correlation.



```
CORRELATION MATRIX OF PREDICTORS:
 > show( round(cor(x),3) )
                Area Bedrooms Bathrooms CarParks PropertyType
Area           1.000   -0.270    -0.087   -0.096         0.32
Bedrooms      -0.270    1.000     0.538    0.435        -0.56
Bathrooms     -0.087    0.538     1.000    0.366        -0.29
CarParks      -0.096    0.435     0.366    1.000        -0.36
PropertyType   0.320   -0.560    -0.290   -0.360         1.00
```

Figure 5: Correlation Matrix of the predictors

2. Mathematical model

$$Y : \text{sale price of properties in Melbourne}$$
$$Y \sim \text{Gamma}(\mu^2/\tau\ ,\ \mu/\tau)$$
$$\mu \sim \text{Normal}(\mu 0, \tau 0)$$
$$\tau \sim \text{Gamma}(a, b)$$

Since the dependent feature Y i.e. sale price has a long right tail, we can use a Gamma error distribution. Here the likelihood has a Gamma distribution of α and β. Since the model goes on top of the mean we need to reparametrize the α and β as μ^2/σ^2 , μ/σ^2. The mean sale price gives a prediction of the mean level of the sale prices in Melbourne, and the variance of the sale prices shows the amount of uncertainty around the property prices in Melbourne.

Y: Sale Price
$X1$: Area
$X2$: Bedrooms

$X3$: Bathrooms
$X4$: car parks
$X5$: Property Types

$$\boldsymbol{Y = \beta 0 + \beta 1\, X1 + \beta 2\, X2 + \beta 3\, X3 + \beta 4\, X4 + \beta 5\, X5 + \epsilon,}$$

Sale Price = $\beta 0$ + $\beta 1$ Area+ $\beta 2$ Bedrooms + $\beta 3$ Bathrooms + $\beta 4$ car parks + $\beta 5$ property types+ ϵ,

where $\epsilon \sim Gamma(\ \alpha, \beta)$

$\epsilon \sim Gamma(\ \mu^2/\ \sigma^2\ ,\ \mu/\ \sigma^2)$

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 parameters mean and Variance are continuous since it shows the sales prices of the properties.
2. **What is the domain of the parameter of interest?** The variance has a domain of (0, ∞). Since the model is placed on the Mean, the coefficients of the model has a domain of (-∞, ∞) as it represents the unit change and therefore can take on a negative value.
3. **How many parameters do you need to use**? We have expert information in both mean and variance therefore we have two prior distributions. In each prior distribution there are two additional parameters. Since we have two parameters, we can express the prior information and degree of belief by these parameters.
4. **Is the prior that I induced on the parameter of interest a conjugate prior? If not, is there a conjugate prior available**? Since we have two parameters to model jointly, we don't have a conjugate joint prior setting.
5. **Is the product of likelihood and prior mathematically tractable?** The posterior with gamma likelihood, normal prior on μ and a gamma prior on τ is not mathematically tractable.

The Prior information given are as follows:

- **Area**: Every m^2 increase in land size increases the sales price by 90 AUD. This is a very strong expert knowledge so a really small variance is given to indicate a high degree of belief. Here the μ = (90/100000) and σ = 0.1
- **Bedrooms**: Every additional bedroom increases the sales price by 100,000AUD. This is a weak expert knowledge, so a slightly high variance is given to indicate low degree of belief. Here the μ = 1 and σ = 4
- **Bathrooms**: There is no expert knowledge on the number of bathrooms, so a really high variance is given to indicate a non informative prior and mean is equal to 0 as there is no information to measure. Here the μ = 0 and σ = 50
- **Car Parks**: Every additional car space increases the sales price by 120,000AUD. This is a strong expert knowledge so a small variance is given to indicate a high degree of belief. Here the μ = 1.2 and σ = 1
- **Property Type**: If the property is a unit, the sale price will be 150,000 AUD less than that of a house on the average. This is a very strong expert knowledge so a really

small variance is given to indicate a high degree of belief. Here the μ = -1.5 and σ = 0.1

4. Finding the posterior distribution

In this model we have a Gamma Likelihood with unknown population parameters of Alpha and Beta, which is re-parametrised in terms of Mu(μ) and Sigma (σ). The model is then placed on top of the Mean, and a Gamma prior is used for the Variance. Each coefficients of the model is distributed as normal distribution, which are also priors. Figure 6 shows the JAGS Model Diagram

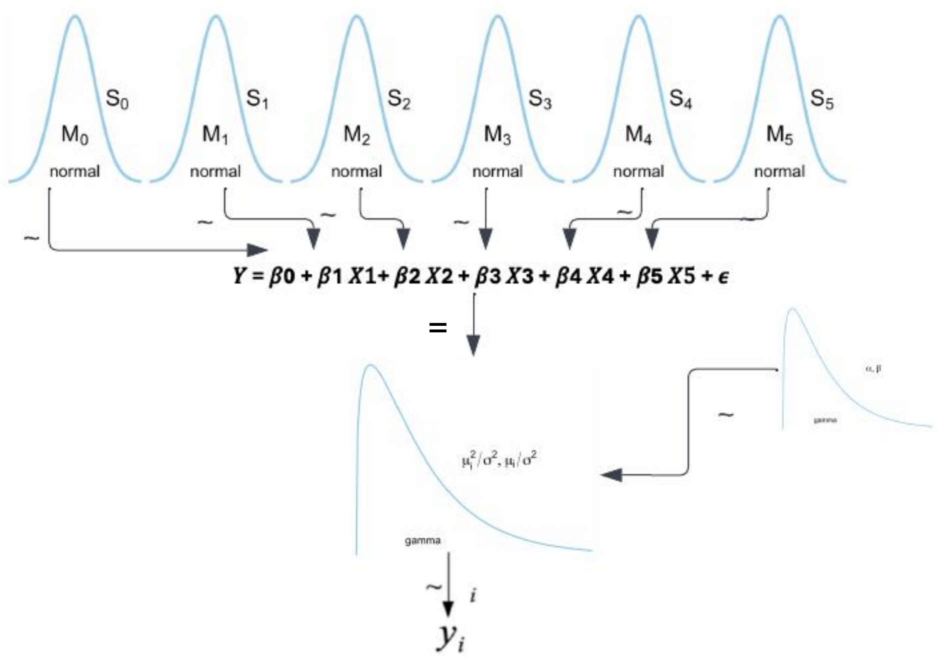
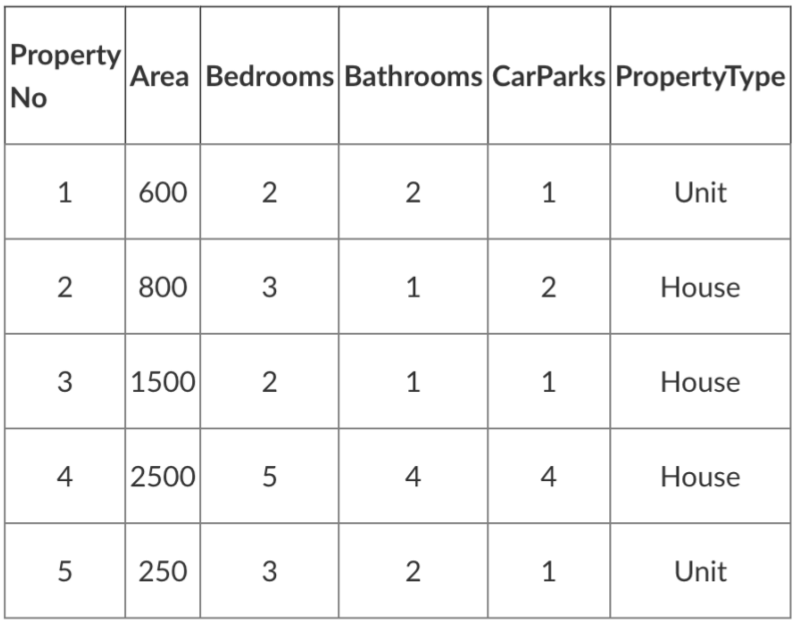

S_0
M_0
normal
S_1
M_1
normal
S_2
M_2
normal
S_3
M_3
normal
S_4
M_4
normal
S_5
M_5
normal
~
$Y = \beta 0 + \beta 1\, X1 + \beta 2\, X2 + \beta 3\, X3 + \beta 4\, X4 + \beta 5\, X5 + \epsilon$
=
α, β
gamma
$\mu_i^2/\sigma^2, \mu_i/\sigma^2$
gamma
i
y_i

Figure 6 JAGS Model Diagram

Along with the posteriors, we are also running predictions of the sale prices based on the following factors given in Figure 7:



Property No	Area	Bedrooms	Bathrooms	CarParks	PropertyType
1	600	2	2	1	Unit
2	800	3	1	2	House
3	1500	2	1	1	House
4	2500	5	4	4	House
5	250	3	2	1	Unit

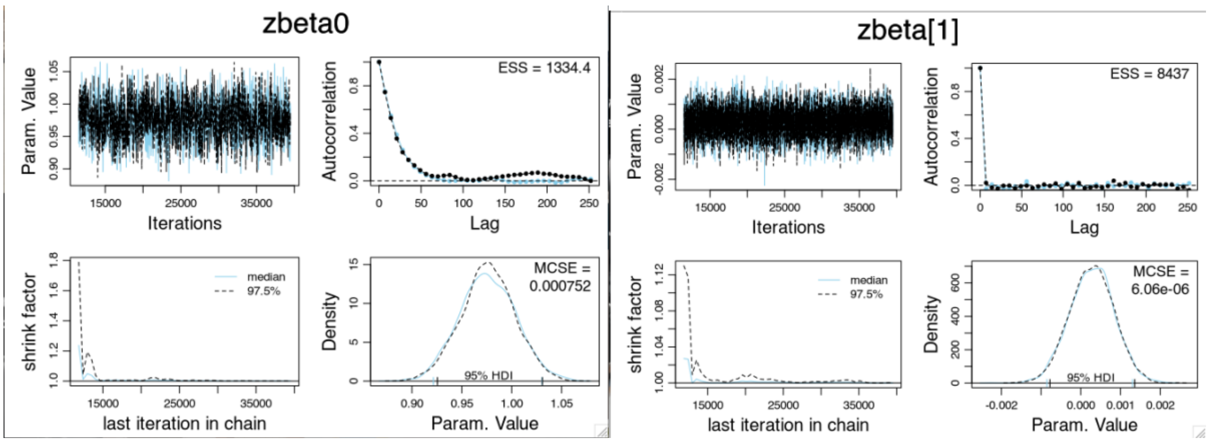
Figure 7:Predictors for the sale prices

4.1 Diagnostic Checking for Mean in Run 1

The run 1 is based on the following settings:

adaptSteps = 1500
burnInSteps = 10000
nChains = 2
thinSteps = 7
numSavedSteps = 4000

The figures below show the Density plots, Shrink Factor, Auto Correlations and Burn-in plot for the tau, 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.



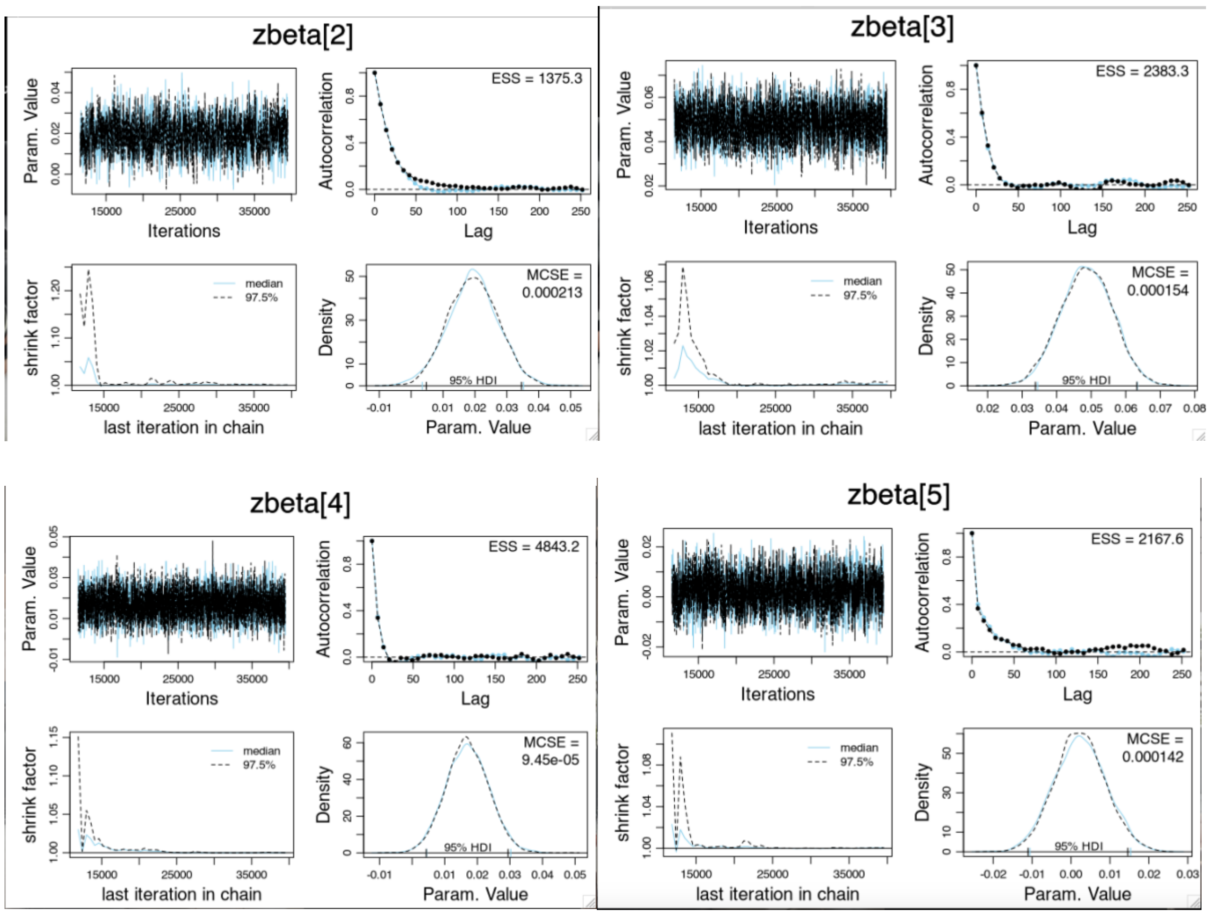


Figure 8: Diagnostic checking for standardised Beta

In Figure 8, 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 Beta1 which indicates that there are no autocorrelations. However, for all the other standardised Beta values, there are some autocorrelations in the beginning and therefore, we need to increase the thinning to get rid of it. Another measure is the ESS which is at 8437 for Beta 1 which is a really good indicator of accuracy. However, for the other the standardised Beta values the ESS is between 1000 and 4000, which is good but could be improved. 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.

4.2 Diagnostic Checking for Variance in Run 1

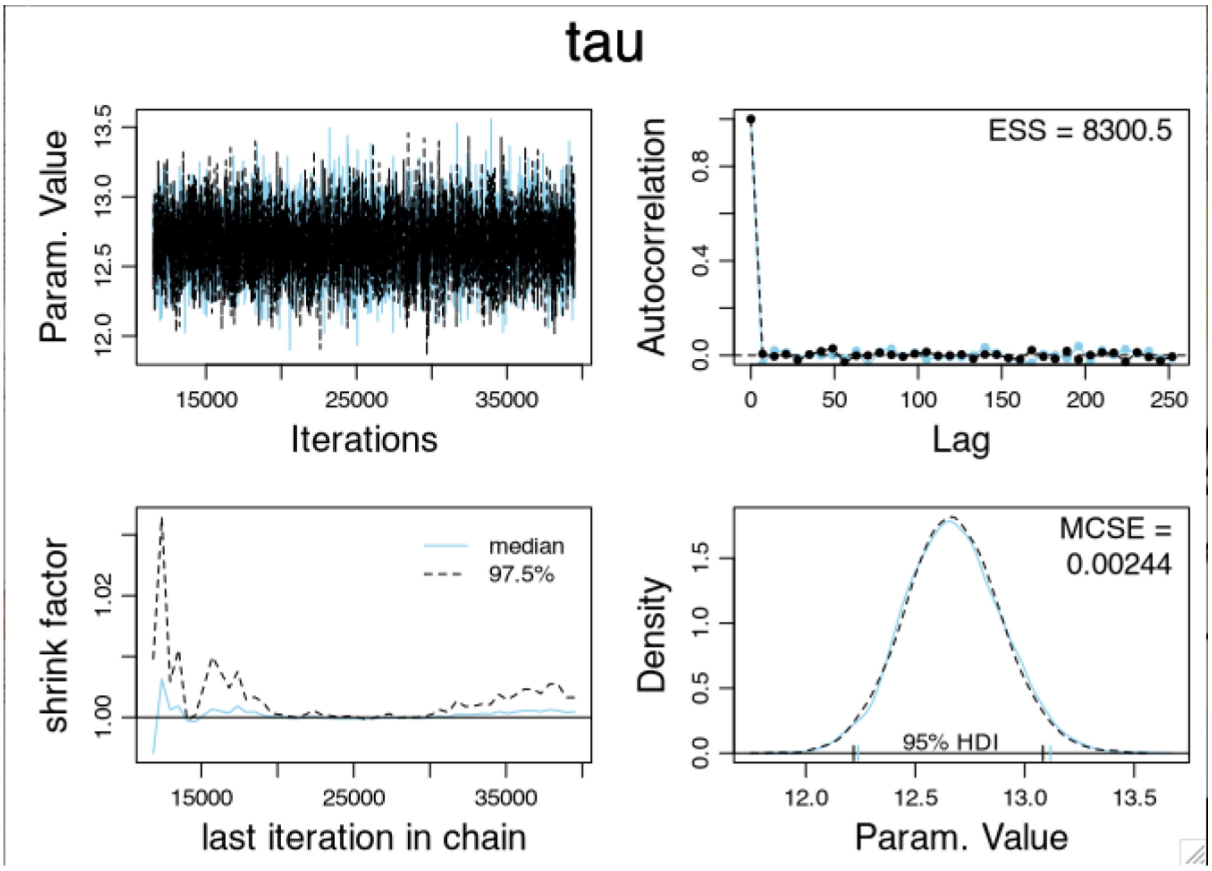


Figure 9 Diagnostic checking for Tau

In Figure 9, The graph 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. Shrink Factor is less than 1.2; MCSE is very close to 0 at a value of 0.00244 and ESS is pretty good at 8300. The autocorrelation plot shows a sudden drop and all further autocorrelations around 0 which indicates that there is no autocorrelations. The MCMC setting seems to be both representative and accurate for Tau.

4.3 Diagnostic Checking for predictors in Run 1

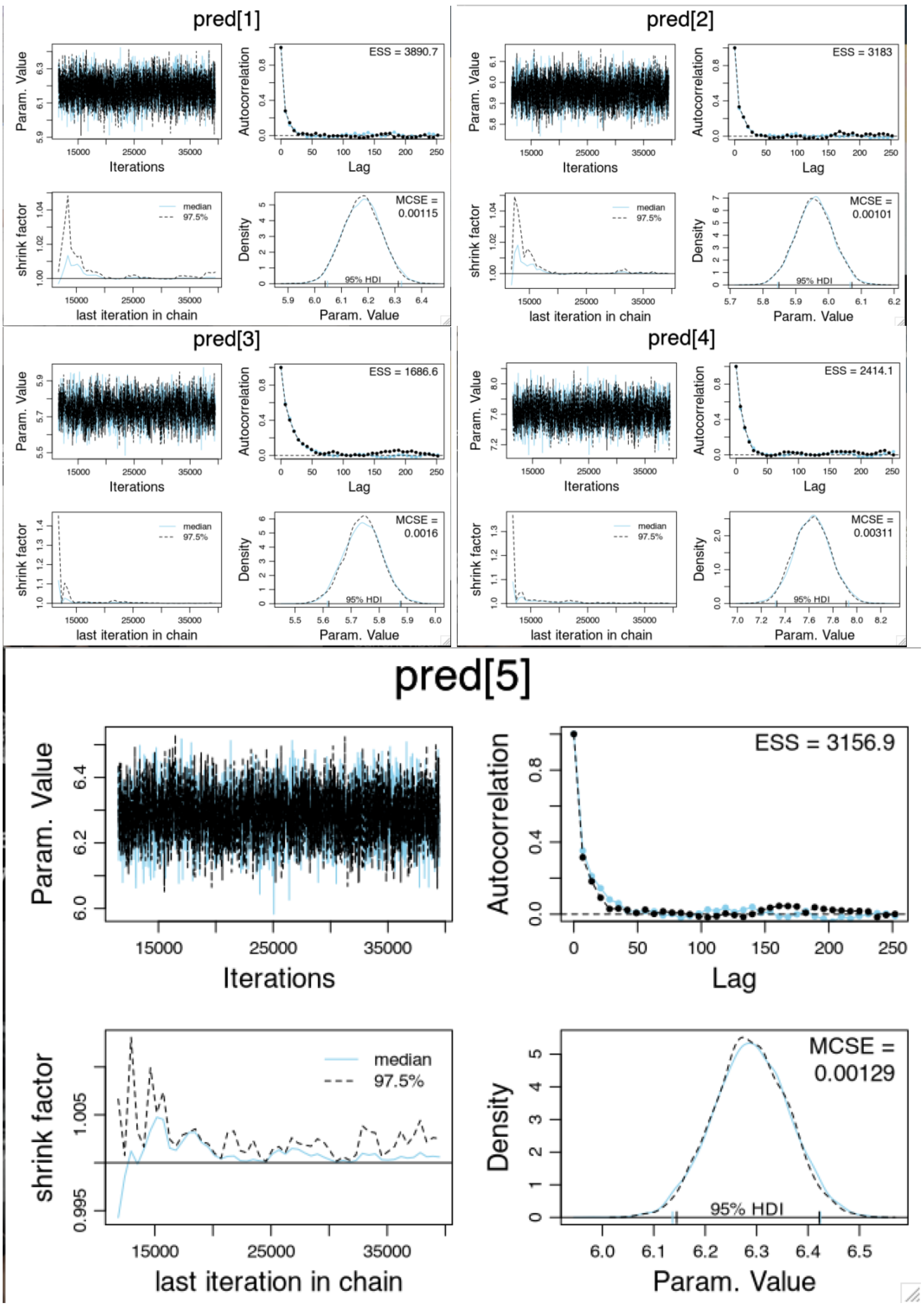
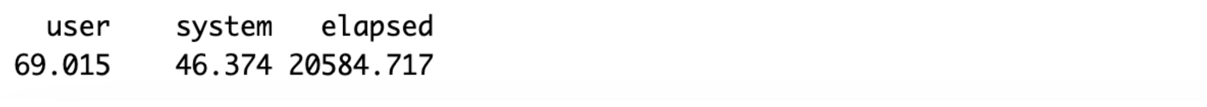


Figure 10:Diagnostic Checking for Predictors

In Figure 10, 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. However, all the predictors have auto correlation in the beginning and needs to be adjusted by increasing the thinning.

In terms of efficiency (as shown in Figure 11) the model took 20584.717 seconds or 5.71 hours to run, which is not the best efficiency, however considering the size of the data it is acceptable.



```
   user    system   elapsed
 69.015    46.374 20584.717
```

Figure 11MCMC efficiency of Run 1

4.4 Diagnostic Checking for Mean in Run 2

In the second run, the thinning is increased to 9 and conducted on a subset of the original data consisting of 445 samples per chain and following are the diagnostics for Mean (given in Figure 12)

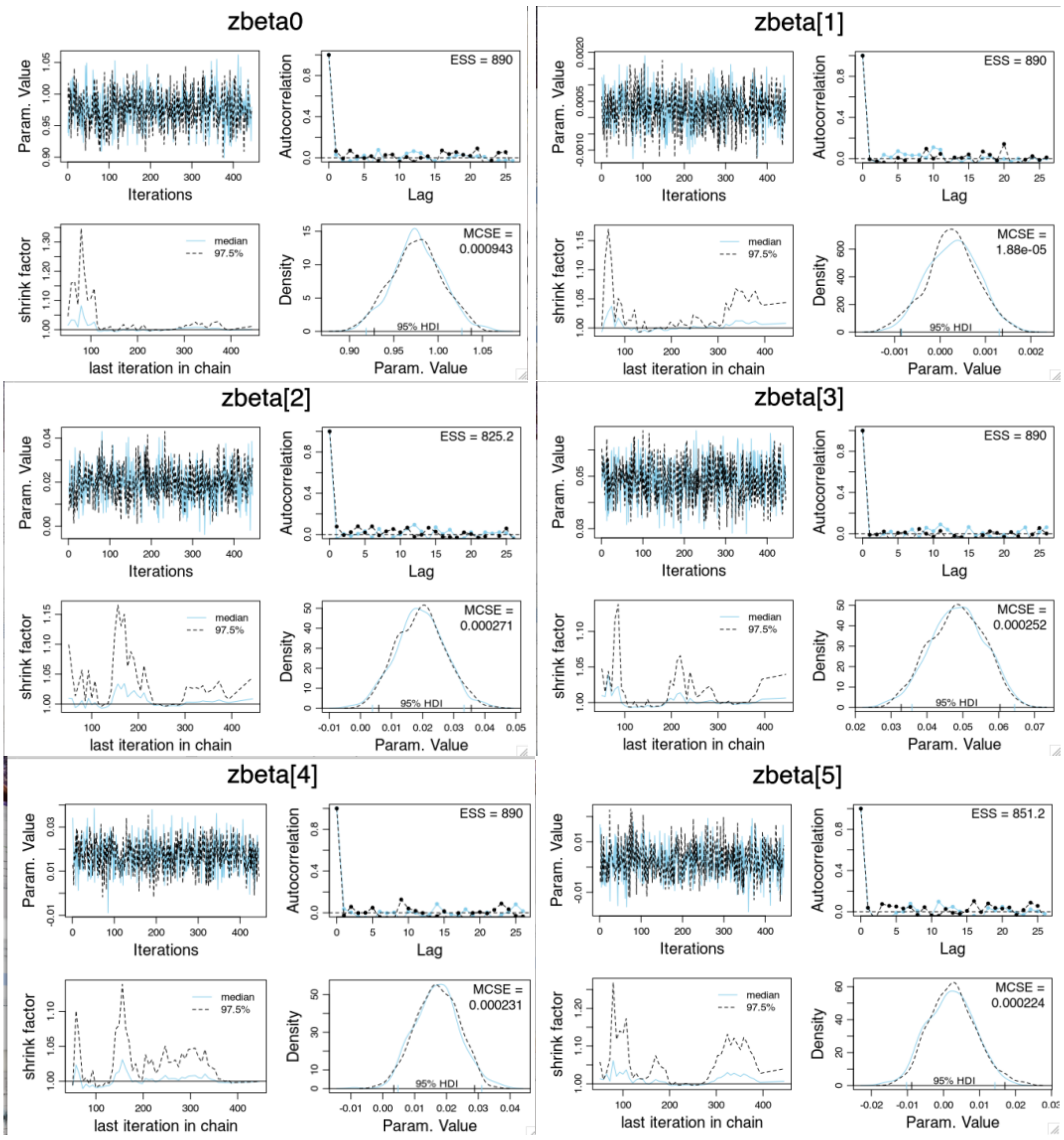


Figure 12: Standardised Beta diagnostic checking

In run 2, trace plot of all the standardised Beta 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. Shrink Factor, ideally needs 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 all standardised Beta values which indicates that there are no autocorrelations. ESS is around 800 for all Beta values which is almost double the sample size indicating a good accuracy. The third measure is the MCSE, which is very close to 0 and therefore demonstrates good accuracy. Increasing the thinning to 9 seems to have worked.

4.5 Diagnostic Checking for Variance in Run 2

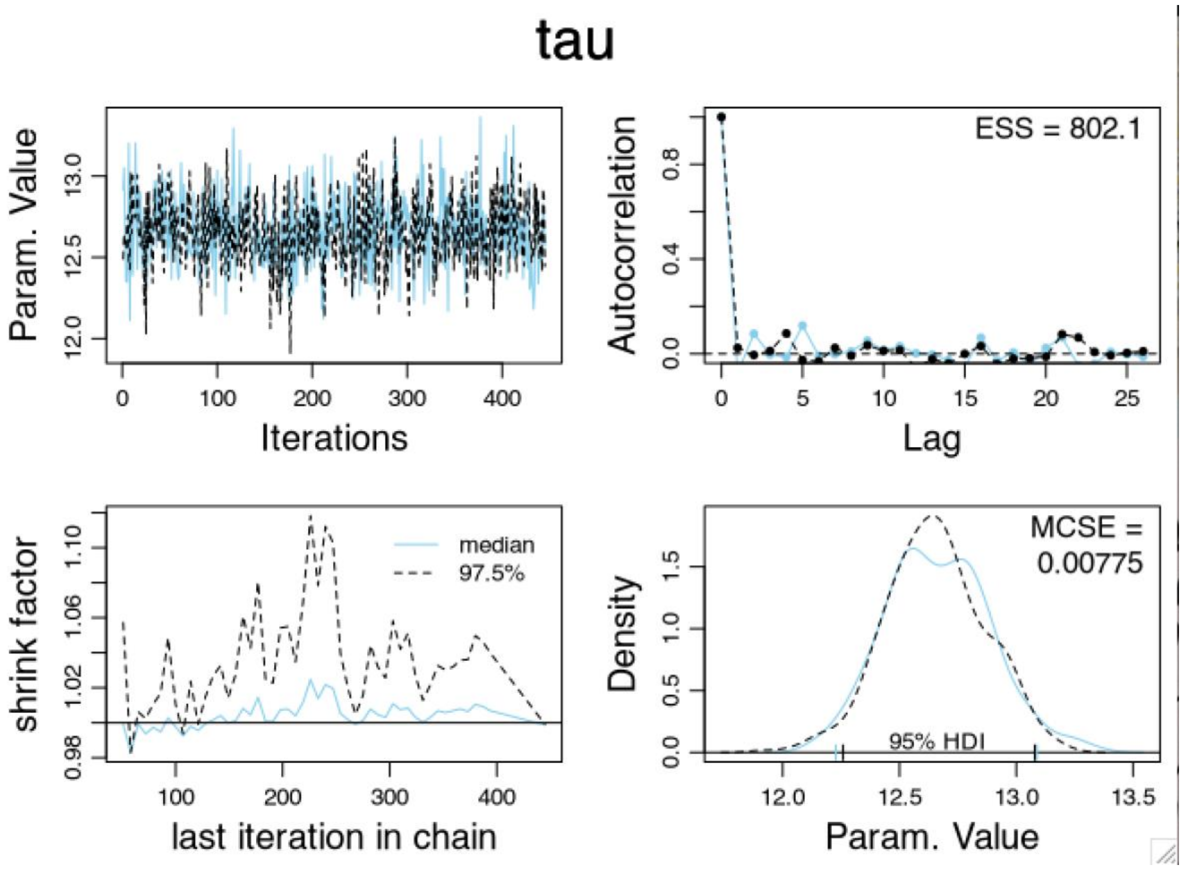


Figure 13 diagnostic checking for Tau in Run 2

In Figure 13, The graph 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, however it is not the best fit. Shrink Factor is less than 1.2; MCSE is very close to 0 at a value of 0.00775 and ESS is very good at 802.1. 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 Tau.

4.6 Diagnostic Checking for predictors in Run 2

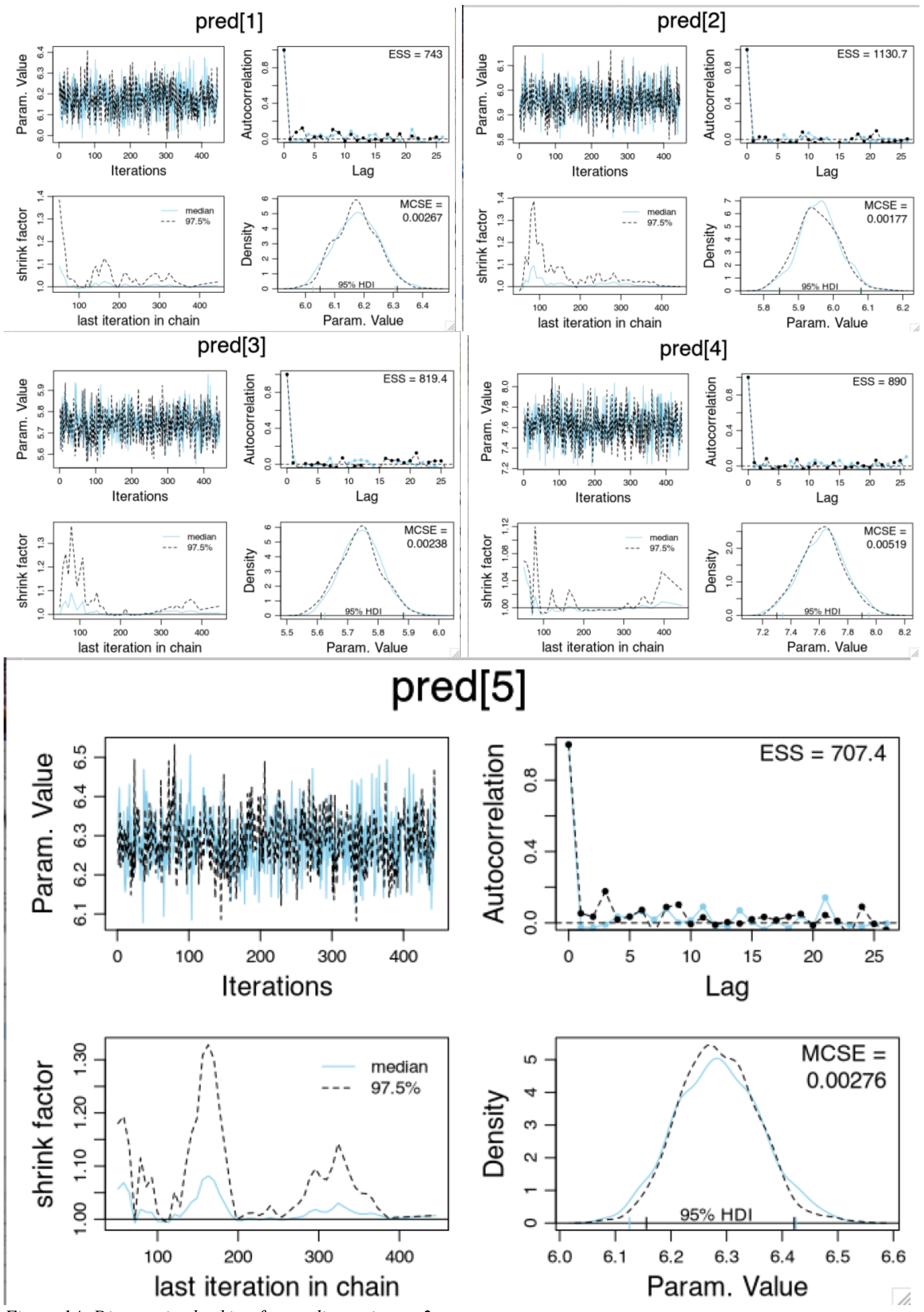


Figure 14: Diagnostic checking for predictors in run 2

In Figure 14, 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

The posteriors for the run is given in Figure 15. To identify the impact of the independent variables we look at the Beta. We can see the for every m2 increase in the Area the sale prices increase by 0.271 AUD, given that the rest of the independent variables are unchanged. 0 is a part of the HDI estimate and therefore Area is a significant indicator of the sale price. In terms of the area, it can be noted that mode of the posterior μ is closer to the likelihood of 90 AUD indicating a informative prior. Every additional bedroom increases the sales price by 11300 AUD. Every additional bathroom increases the sales price by 41700 AUD. Every additional car space increases the sales price by 10500 AUD. 0 is just outside the HDI estimates for Bedrooms, Bathrooms and Car Parks and therefore it may not be a significant indicator, however property Type is a significant Indicator. In terms of the property Type, it can be noted that mode of the posterior μ is closer to the likelihood of -1.5KAUD indicating a informative prior.

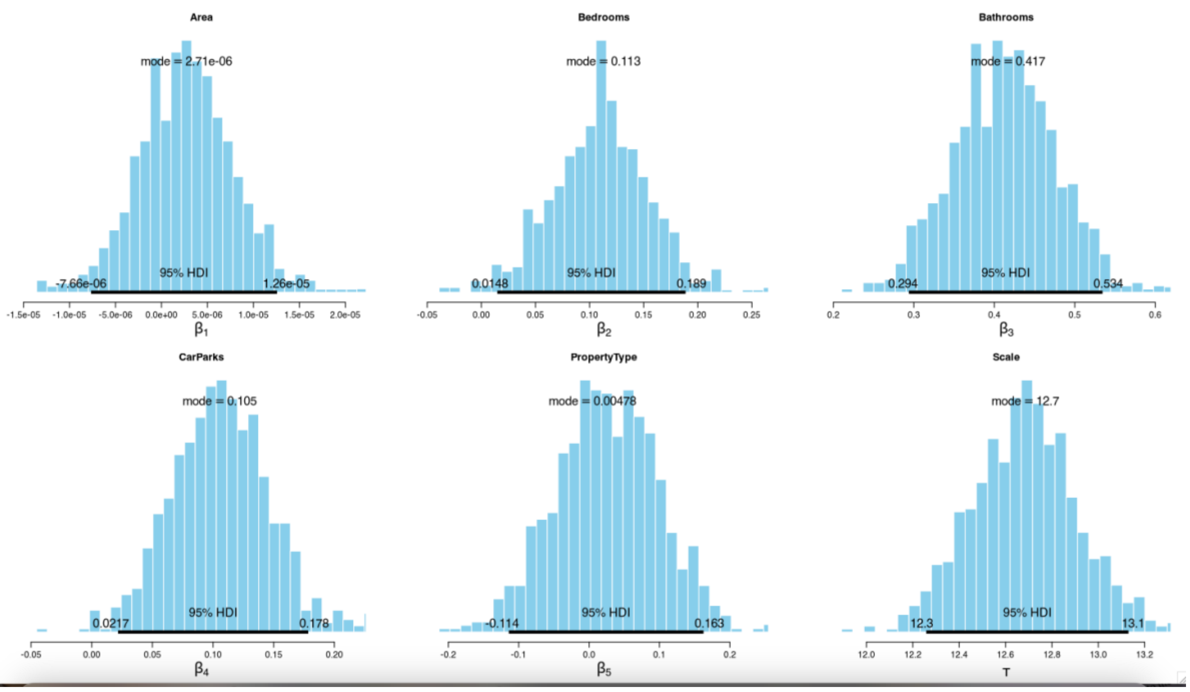


Figure 15:Interpretation of Posteriors

Figure 16 gives the point estimates of the prediction. Under the conditions in prediction 1 of 600m2 in Area, 2 bedrooms, 2 bathrooms, 1 car park and Unit property, the predicted sale price will be 617K AUD. Under these setting the probability that the sale price will be between 604K and 631K AUD is 0.95. Under the conditions in prediction 2 of 800 m2 in Area, 3 bedrooms, 1 bathrooms, 2 car park and House property, the predicted sale price will be 596K AUD. Under these setting the probability that the sale price will be between 585K and 607K AUD is 0.95. Under the conditions in prediction 3 of 1500 m2 in Area, 2 bedrooms, 1 bathrooms, 1 car park and House property, the predicted sale price will be 575K AUD. Under these setting the probability that the sale price will be between 562K and 586K

AUD is 0.95. Under the conditions in prediction 4 of 2500 m2 in Area, 5 bedrooms, 4 bathrooms, 4 car park and House property, the predicted sale price will be 765K AUD. Under these setting the probability that the sale price will be between 731K and 792K AUD is 0.95. Under the conditions in prediction 5 of 250 m2 in Area, 3 bedrooms, 2 bathrooms, 1 car park and Unit property, the predicted sale price will be 630K AUD. Under these setting the probability that the sale price will be between 615K and 642K AUD is 0.95.

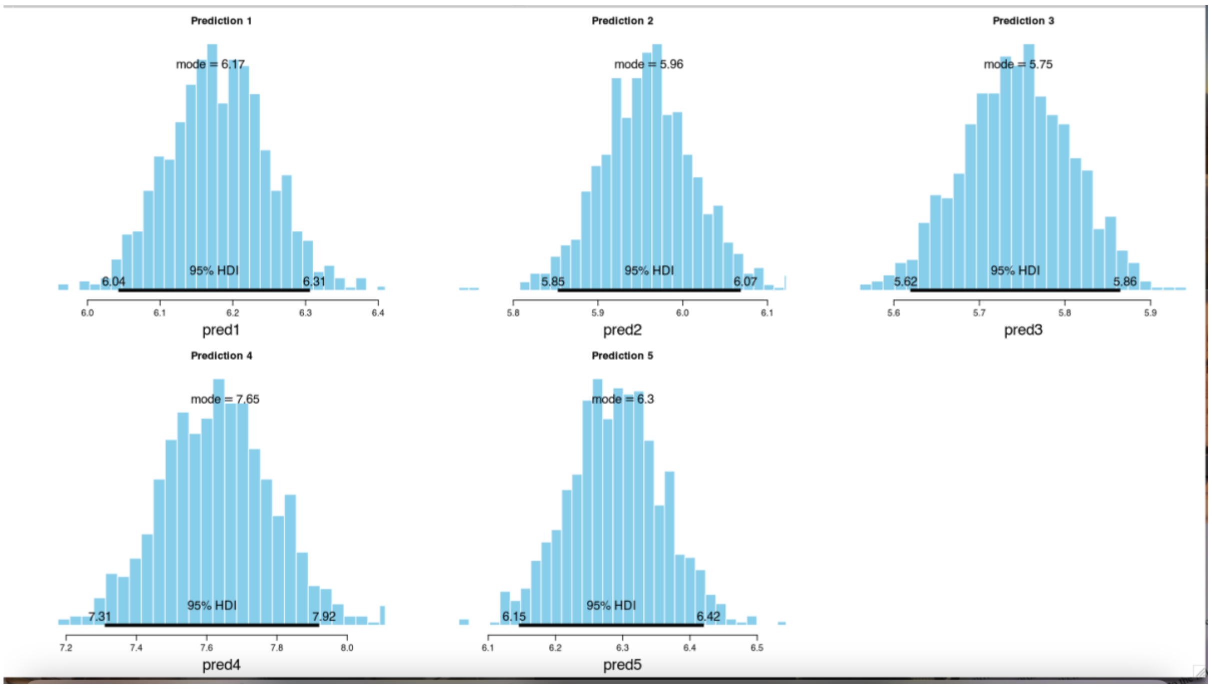


Figure 16: Point estimates of the prediction

5.1 Posterior Predictive checks

In a predictive check, a random data is generated based on the coefficients from the model, this is then checked with the observed values to see the fitness of the model. Figure 17 shows the predictive check of the above model. It can be noticed that the model overlaps a lot of the observations at the bottom of the graphs. However the model doesn't take into account the higher valued observations. The parameter estimates here are not too bad but its not the perfect fit.

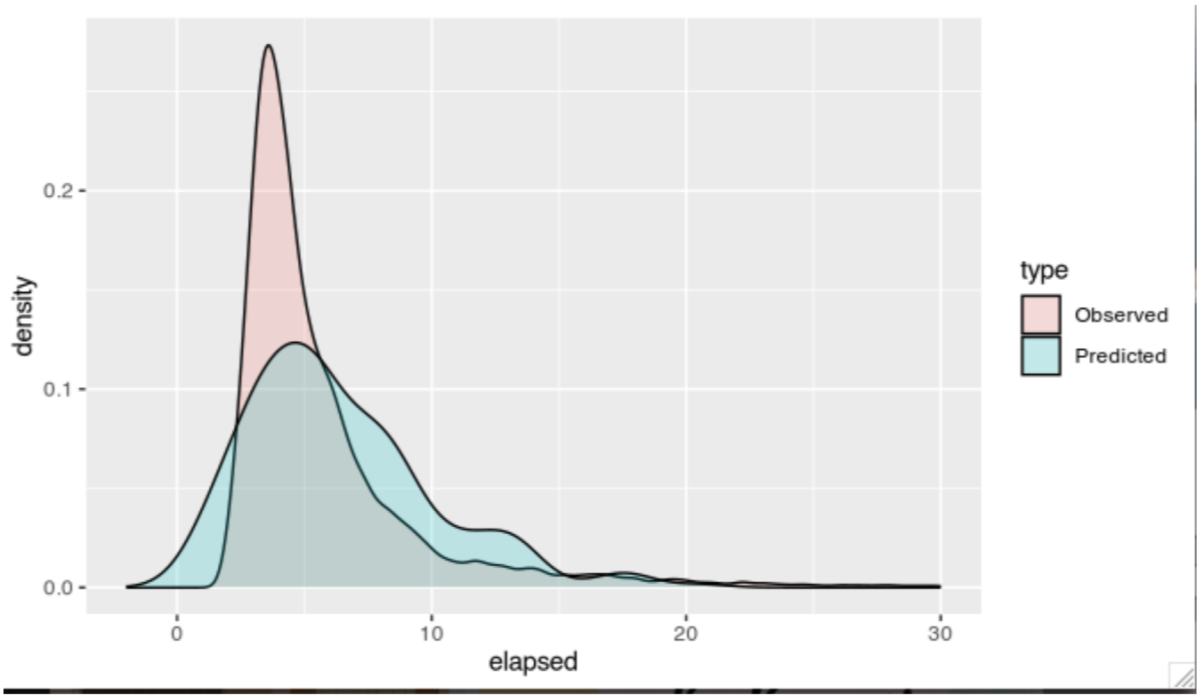


Figure 17:Predictive checks with the observed value

References

• Dr. Demirhan (2024) 'Markov Chain Monte Carlo Methods' [PowerPoint slides, MATH2269], RMIT University, Melbourne

Appendix

The code used to run the model:

```
graphics.off() # This closes all of R's graphics windows.
rm(list=ls())  # Careful! This clears all of R's memory!
library(ggplot2)
library(ggpubr)
library(ks)
library(rjags)
library(runjags)
#setwd("~/Documents/MATH2269_Bayesian/2024/presentations/Module 6")

#------------------------------------------------------------------------------
# Load the relevant model into R's working memory:
source("DBDA2E-utilities.R")
```

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

smryMCMC_HD = 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_HD = function( codaSamples , data , xName="x" , yName="y" ,
                        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 ) }
zVar = mcmcMat[,"zVar"]
beta0 = mcmcMat[,"beta0"]
beta  = mcmcMat[,grep("^beta$|^beta\\[",colnames(mcmcMat))]
if ( ncol(x)==1 ) { beta = matrix( beta , ncol=1 ) }
tau = mcmcMat[,"tau"]
pred1 = mcmcMat[,"pred[1]"] # Added by Demirhan
pred2 = mcmcMat[,"pred[2]"] # Added by Demirhan
pred3 = mcmcMat[,"pred[3]"] # Added by Demirhan
pred4 = mcmcMat[,"pred[4]"] # Added by Demirhan
pred5 = mcmcMat[,"pred[5]"] # Added by Demirhan
#-----------------------------------------------------------------------------
# Compute R^2 for credible parameters:
YcorX = cor( y , x ) # correlation of y with each x predictor
Rsq = beta %*% matrix( YcorX , ncol=1 )
Rsq = Rsq[which((Rsq[,1]>=0)&(Rsq[,1]<=1)) ,1] # HD: modified
#-----------------------------------------------------------------------------
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
  if (!is.na(compVal["beta0"])){
    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"] ))
  } else {
    histInfo = plotPost( beta0 , cex.lab = 1.75 , showCurve=showCurve ,
                       xlab=bquote(beta[0]) , main="Intercept")
  }
  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( tau , cex.lab = 1.75 , showCurve=showCurve ,
                     xlab=bquote(tau) , main=paste("Scale") )
  # 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") )
  panelCount = decideOpenGraph( panelCount ,  saveName=paste0(saveName,"PostMarg") )
  histInfo = plotPost( pred1 , cex.lab = 1.75 , showCurve=showCurve ,
                  xlab="pred1" , main="Prediction 1" ) # Added by Demirhan
  panelCount = decideOpenGraph( panelCount ,  saveName=paste0(saveName,"PostMarg") )
  histInfo = plotPost( pred2 , cex.lab = 1.75 , showCurve=showCurve ,
                  xlab="pred2" , main="Prediction 2" ) # Added by Demirhan
  panelCount = decideOpenGraph( panelCount ,  saveName=paste0(saveName,"PostMarg") )
  histInfo = plotPost( pred3 , cex.lab = 1.75 , showCurve=showCurve ,
                  xlab="pred3" , main="Prediction 3" ) # Added by Demirhan
  panelCount = decideOpenGraph( panelCount ,  saveName=paste0(saveName,"PostMarg") )
  histInfo = plotPost( pred4 , cex.lab = 1.75 , showCurve=showCurve ,
                  xlab="pred4" , main="Prediction 4" ) # Added by Demirhan
  panelCount = decideOpenGraph( panelCount ,  saveName=paste0(saveName,"PostMarg") )
  histInfo = plotPost( pred5 , cex.lab = 1.75 , showCurve=showCurve ,
                  xlab="pred5" , main="Prediction 5" ) # 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( zVar , cex.lab = 1.75 , showCurve=showCurve ,
                  xlab=bquote(z*tau) , main=paste("Scale") )
  # 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====================


myData <- read.csv("Assignment2PropertyPrices.csv")
yName = "SalePrice" ; xName = c("Area",    "Bedrooms",    "Bathrooms",  "CarParks",
           "PropertyType")
```

```
# converting the PropertyType as a factor since it is a categorical variable
myData$PropertyType <- as.factor(myData$PropertyType)
head(myData)

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

# Histogram
par(mfrow = c(1, 2))
hist(myData$SalePrice, xlab = "Sale Price", main = "Histogram of Sale Price")
# Kernel density estimation
plot(kde(myData$SalePrice), xlab = "Sale Price", main = "Density Plot of Sale Price") #
density plot
par(mfrow = c(1, 1))

# Scatter plots
p1 <- ggplot(myData, aes(x=Area, y=SalePrice, color=PropertyType)) +
  geom_point() #Area

p2 <- ggplot(myData, aes(x=Bedrooms, y=SalePrice, color=PropertyType)) +
  geom_point() #Bedrooms

p3 <- ggplot(myData, aes(x=Bathrooms, y=SalePrice, color=PropertyType)) +
  geom_point() #Bathrooms

p4 <- ggplot(myData, aes(x=CarParks, y=SalePrice,color=PropertyType)) +
  geom_point() #CarParks

p5 <- ggplot(myData, aes(x=PropertyType, y=SalePrice,color=PropertyType)) +
  geom_point() #PropertyType


figure <- ggarrange(p1, p2, p3, p4, p5, nrow = 4, ncol = 2)
figure

summary(myData)

#-----------------------------------------------------------------------------
# Drawing the Jags model Diagram:
#-----------------------------------------------------------------------------

# Reads in the functions plot_dist, plot_dist_svg, plot_dist_png and a
# list of predefined distributions called dists.

# source("plot_dist.R")
# plot_dist(dists$gamma, labels = c(params = "A, B"))
# plot_dist(dists$gamma, labels = expression(list(alpha, beta)))
# plot_dist(dists$gamma, labels = expression(list(mu[i]^2/sigma^2, mu[i]/sigma^2)))
# par(mfrow = c(1, 6))
# plot_dist(dists$normal, labels = c(mean = expression(M[0]), right_sd = expression(S[0])))
# plot_dist(dists$normal, labels = c(mean = expression(M[1]), right_sd = expression(S[1])))
# plot_dist(dists$normal, labels = c(mean = expression(M[2]), right_sd = expression(S[2])))
# plot_dist(dists$normal, labels = c(mean = expression(M[3]), right_sd = expression(S[3])))
# plot_dist(dists$normal, labels = c(mean = expression(M[4]), right_sd = expression(S[4])))
# plot_dist(dists$normal, labels = c(mean = expression(M[5]), right_sd = expression(S[5])))
# par(mfrow = c(1, 1))

#-------------------------------------------------------------------------------

#-------------------------------------------------------------------------------
# Creating the Jags Data and Model Block:
#-------------------------------------------------------------------------------

# THE DATA.
myData$PropertyType <- as.numeric(as.character(myData$PropertyType))
y = myData[,yName]
x = as.matrix(myData[,c("Area","Bedrooms","Bathrooms","CarParks","PropertyType")])

# Some more descriptives
cat("\nCORRELATION MATRIX OF PREDICTORS:\n ")
show( round(cor(x),3) )
cat("\n")

#Setting up the prediction Matrix
xPred = array(NA, dim = c(5,5))
xPred[1,] = c(600, 2, 2, 1,1)
xPred[2,] = c(800, 3, 1, 2,0)
xPred[3,] = c(1500, 2, 1, 1,0)
xPred[4,] = c(2500, 5, 4, 4,0)
xPred[5,] = c(250, 3, 2, 1,1)

# Specify the data in a list, for later shipment to JAGS:
dataList <- list(
  x = x ,
  y = y ,
  xPred = xPred ,
  Nx = dim(x)[2] ,
  Ntotal = dim(x)[1]
)

#Setting the initials
initsList <- list(
  zbeta0 = 2000,
  zbeta = c(100, 1, 1, 0.5,1),
  Var = 12000000
)


# Running the model with scaling
# THE MODEL.
modelString = "
# Standardize the data:
data {
  ysd <- sd(y)
  for ( i in 1:Ntotal ) {
    zy[i] <- y[i] / ysd
  }
  for ( j in 1:Nx ) {
    xsd[j] <-   sd(x[,j])
    for ( i in 1:Ntotal ) {
      zx[i,j] <- x[i,j] / xsd[j]
    }
  }
}


# Specify the model for scaled data:
model {
  for ( i in 1:Ntotal ) {
    zy[i] ~ dgamma( (mu[i]^2)/zVar , mu[i]/zVar )
    mu[i] <- zbeta0 + sum( zbeta[1:Nx] * zx[i,1:Nx] )
  }
  # Priors on standardized scale:
  zbeta0 ~ dnorm( 0 , 1/(2)^2 )  # 1/ variance for normal distribution - NON informative prior
for intercept
  zbeta[1] ~ dnorm( 0.00009/xsd[1] , 1/(0.1/xsd[1]^2) ) # Area- very strong expert knowledge
  zbeta[2] ~ dnorm( 1/xsd[2] , 1/(4/xsd[2]^2) ) # Bedroom - weak expert knowledge
  zbeta[3] ~ dnorm( 0 , 1/50 ) # Bathrooms, no expert knowledge
  zbeta[4] ~ dnorm( (1.2)/xsd[4] , 1/(1/xsd[4]^2) ) # carparks - strong expert knowledge
  zbeta[5] ~ dnorm( (-1.5)/xsd[5] , 1/(0.1/xsd[5]^2) ) # property type - very strong expert
knowledge #######CHECK THIS

  zVar ~ dgamma( 0.01 , 0.01 )
  # Transform to original scale:
  beta[1:Nx] <- ( zbeta[1:Nx] / xsd[1:Nx] ) * ysd
  beta0 <- zbeta0*ysd
  tau <- zVar * (ysd)^2

  # Compute predictions at every step of the MCMC
  for ( i in 1:5){
    pred[i] <- beta0 + beta[1] * xPred[i,1] + beta[2] * xPred[i,2] + beta[3] * xPred[i,3] +
beta[4] * xPred[i,4]+ beta[5] * xPred[i,5]
  }

}
" # close quote for modelString
# Write out modelString to a text file
writeLines( modelString , con="TEMPmodel.txt" )
```

```
parameters = c( "zbeta0" ,  "zbeta" , "beta0" ,  "beta" ,  "tau", "zVar") # Here beta is a vector!

#------------------------------------------------------------------------------
# Running the JAGS Model:
#------------------------------------------------------------------------------

#Run 1
adaptSteps = 1500  # Number of steps to "tune" the samplers
burnInSteps = 10000
nChains = 2
thinSteps = 7 # First run for 3
numSavedSteps = 4000
nIter = ceiling( ( numSavedSteps * thinSteps ) / nChains )


# Parallel run
startTime = proc.time()
runJagsOut <- run.jags( method="parallel" ,
                        model="TEMPmodel.txt" ,
                        monitor=c( "zbeta0" ,  "zbeta" , "beta0" ,  "beta" ,  "tau", "zVar", "pred")  ,
                        data=dataList ,
                        inits=initsList ,
                        n.chains=nChains ,
                        adapt=adaptSteps ,
                        burnin=burnInSteps ,
                        sample=numSavedSteps ,
                        thin=thinSteps , summarise=FALSE , plots=FALSE )
codaSamples = as.mcmc.list( runJagsOut )
stopTime = proc.time()
elapsedTime = stopTime - startTime
show(elapsedTime)

save.image(file='10KBurnin_4000SavedIter_7Thin_TEST5.RData') #Save the run
#load('10KBurnin_4000SavedIter_7Thin_TEST5.RData') # load the run

#nrow(codaSamples[[1]])

#------------------------------------------------------------------------------
# Diagnostic Checking - Run 1
#------------------------------------------------------------------------------

diagMCMC( codaSamples , parName="beta0" )
diagMCMC( codaSamples , parName="beta[1]" )
diagMCMC( codaSamples , parName="beta[2]" )
diagMCMC( codaSamples , parName="beta[3]" )
diagMCMC( codaSamples , parName="beta[4]" )
diagMCMC( codaSamples , parName="beta[5]" )
diagMCMC( codaSamples , parName="tau" )
diagMCMC( codaSamples , parName="pred[1]" )
diagMCMC( codaSamples , parName="pred[2]" )
diagMCMC( codaSamples , parName="pred[3]" )
diagMCMC( codaSamples , parName="pred[4]" )
diagMCMC( codaSamples , parName="pred[5]" )
diagMCMC( codaSamples , parName="zbeta0" )
diagMCMC( codaSamples , parName="zbeta[1]" )
diagMCMC( codaSamples , parName="zbeta[2]" )
diagMCMC( codaSamples , parName="zbeta[3]" )
diagMCMC( codaSamples , parName="zbeta[4]" )
diagMCMC( codaSamples , parName="zbeta[5]" )
graphics.off()

#Run 2 - Do further thinning from the codaSamples

furtherThin <- 9
thiningSequence <- seq(1,nrow(codaSamples[[1]]), furtherThin)
newCodaSamples <- mcmc.list()
for ( i in 1:nChains){
  newCodaSamples[[i]] <- as.mcmc(codaSamples[[i]][thiningSequence,])
}

summary(codaSamples)

summary(newCodaSamples)

#------------------------------------------------------------------------------
# Diagnostic Checking - Run 2
#------------------------------------------------------------------------------
diagMCMC( newCodaSamples , parName="beta0" )
diagMCMC( newCodaSamples , parName="beta[1]" )
diagMCMC( newCodaSamples , parName="beta[2]" )
diagMCMC( newCodaSamples , parName="beta[3]" )
diagMCMC( newCodaSamples , parName="beta[4]" )
diagMCMC( newCodaSamples , parName="beta[5]" )
diagMCMC( newCodaSamples , parName="tau" )
diagMCMC( newCodaSamples , parName="pred[1]" )
diagMCMC( newCodaSamples , parName="pred[2]" )
diagMCMC( newCodaSamples , parName="pred[3]" )
diagMCMC( newCodaSamples , parName="pred[4]" )
diagMCMC( newCodaSamples , parName="pred[5]" )
diagMCMC( newCodaSamples , parName="zbeta0" )
diagMCMC( newCodaSamples , parName="zbeta[1]" )
diagMCMC( newCodaSamples , parName="zbeta[2]" )
diagMCMC( newCodaSamples , parName="zbeta[3]" )
diagMCMC( newCodaSamples , parName="zbeta[4]" )
diagMCMC( newCodaSamples , parName="zbeta[5]" )
graphics.off()

#------------------------------------------------------------------------------
# Posteriors
```

```
#------------------------------------------------------------------------------

compVal <- data.frame("beta0" = NA, "beta[1]" = NA, "beta[2]" = NA, "beta[3]" = NA,
"beta[4]" =  NA,"beta[5]" =  NA, "tau" = NA , check.names=FALSE)
#summaryInfo <- smryMCMC_HD( codaSamples = codaSamples , compVal = compVal )
summaryInfo <- smryMCMC_HD( codaSamples = newCodaSamples , compVal = compVal )
print(summaryInfo)

# plotMCMC_HD( codaSamples = codaSamples , data = myData, xName=xName ,
#           yName=yName, compVal = compVal)
plotMCMC_HD( codaSamples = newCodaSamples , data = myData, xName=xName ,
          yName=yName, compVal = compVal)

#------------------------------------------------------------------------------
# ============ Predictive check ============
#------------------------------------------------------------------------------
summaryInfo[8:13,3]
summaryInfo[14,3]
coefficients <- summaryInfo[8:13,3] # Get the model coefficients out
Variance <- summaryInfo[14,3] # Get the variance out
# Since we imposed the regression model on the mean of the gamma likelihood,
# we use the model (X*beta) to generate the mean of gamma population for each
# observed x vector.
meanGamma <- as.matrix(cbind(rep(1,nrow(x)),  x)) %*% as.vector(coefficients)
# Generate random data from the posterior distribution. Here I take the
# reparameterisation back to alpha and beta.
randomData <- rgamma(n= 231,shape=meanGamma^2/Variance, rate =
meanGamma/Variance)

# Display the density plot of observed data and posterior distribution:
predicted <- data.frame(elapsed = randomData)
observed <- data.frame(elapsed = y)
predicted$type <- "Predicted"
observed$type <- "Observed"
dataPred <- rbind(predicted, observed)

ggplot(dataPred, aes(elapsed, fill = type)) + geom_density(alpha = 0.2)+xlim(-2,30)
```