BAYESIAN ANALYSIS OF PROPERTY PRICES IN MELBOURNE

Submitted By: Shivangi Agrawal (s3802226)

RMIT UNIVERSITY

Executive Summary

This report is based on the Bayesian Analysis of Property prices in Melbourne, Australia in 2020. This is the second phase of analysis where the main focus is on including more parameters and draw Bayesian estimates of the variable 'SalePrice' (which represents property costs) using them. The dataset provided is fabricated by running multiple analyses on real data. There are five independent variables in the dataset based on which the value of **dependent variable ('SalePrice' in 100,000 AUD)** is estimated. A Multiple Linear Regression Model is fit on the data to carry out Bayesian Analysis. We have prior knowledge of independent variables coming from expert estate agent and we predict the SalePrice keeping that knowledge in mind. The model is developed in R using JAGS programming and goodness of fit is determined using **ggplot()** function. The reliance of model is tested using **Markov chain Monte Carlo** (**MCMC**) diagnostics. Value of property price is calculated for different set of values of independent parameters using Bayesian prediction.

Table of Contents

1. Introduction

Prediction of property prices in Melbourne has been a research interest for analysts since a long time. This problem is attempted to be solved using Bayesian Analysis methodologies. The goal of this project is to implement **Multiple Linear Regression** model and carry out Bayesian Analysis to predict the property prices using prior information of independent variables. The model parameters $ß_0$, $ß_1$,…., $ß_k$ (where k is the number of independent variables) are assumed to be normally distributed. The distribution of data is determined with the help of descriptive analysis and the regression model is fit on the mean distribution of data (μ_i). The analysis consists of eight major steps:

- Creation of JAGS model diagram showing multiple linear regression setting of this task.
- Specification of prior exhibiting expert information.
- Implementation of JAGS model diagram into JAGS data and model blocks.
- Compilation of model and generation and assessment of MCMC diagnostics.
- Prediction of Bayesian estimates using the created model.
- Prediction of SalePrice for given set of values of independent variables.

2. Descriptive Analysis:

The data X has:

5 independent variables: Area, Bedrooms, Bathrooms, CarParks, and PropertyType

1 dependent variable: SalePrice.100K.

The SalePrice is given in 100,000 AUD with a total of 10,000 observations.

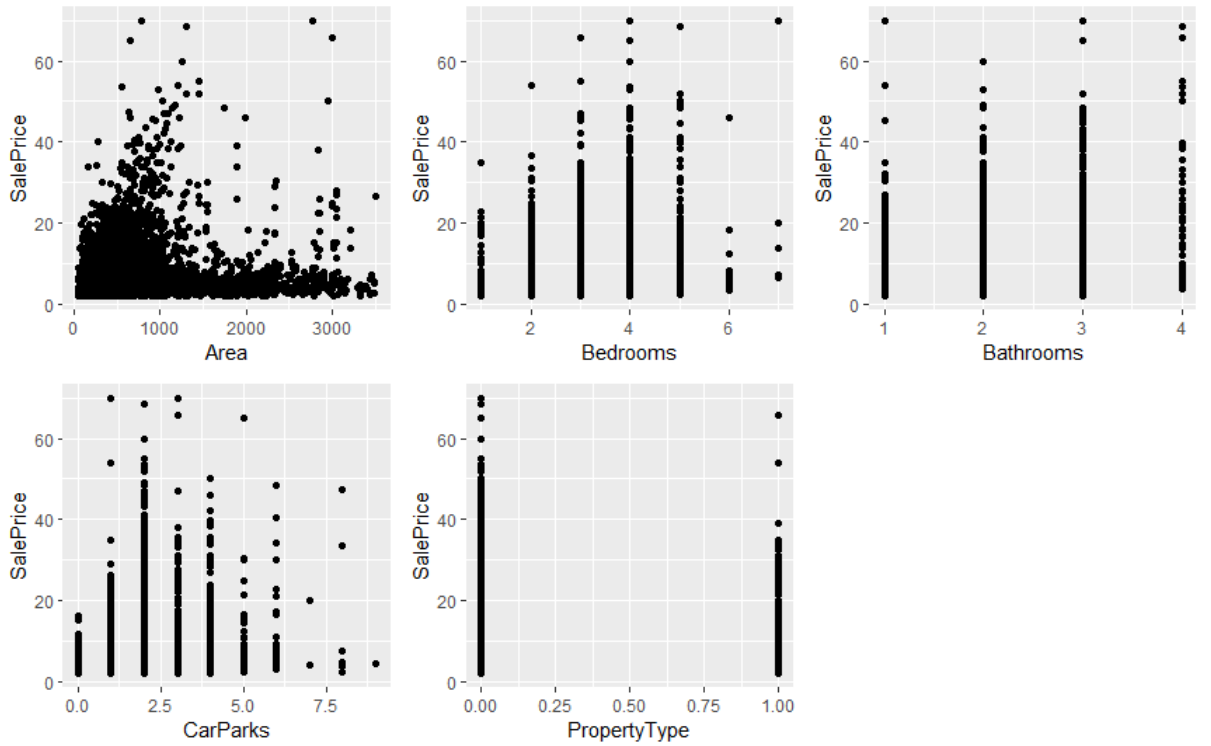

Fig. 1

Fig. 1 shows relationship of the dependent variable SalePrice with each independent variable separately.

Inferences:

- Evidently, sale price is affected by each parameter but is **not solely dependent** on a single one. The scattered arrangement of Sale Price component when plotted with Area, Bedrooms, Bathrooms, and CarParks shows that all factors collectively decide the sale price and no single parameter can be considered independently.
- The graph of PropertyType vs SalePrice shows that House (encoded as 0) is constantly **higher** than that of Unit (encoded as 1).

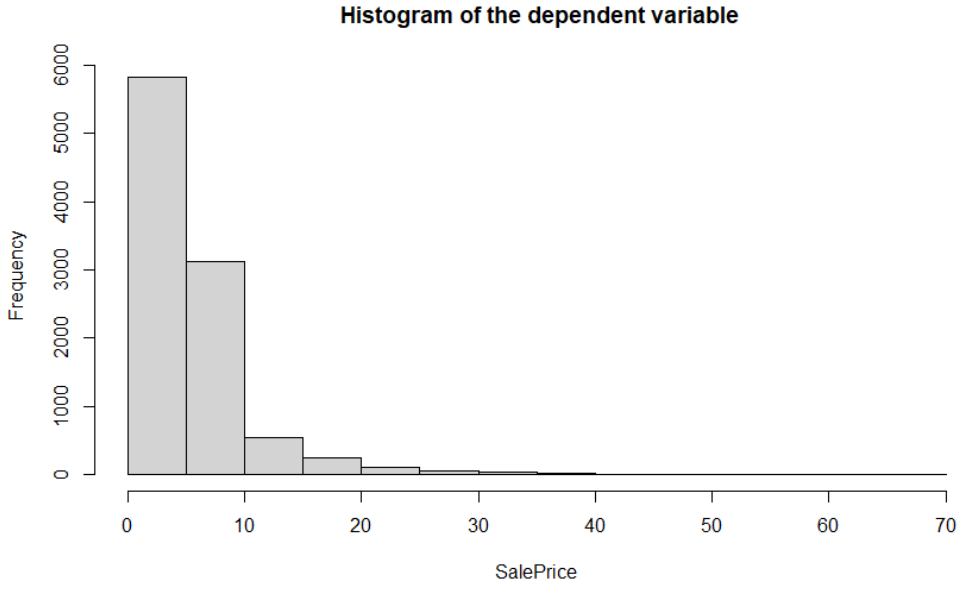

Fig. 2

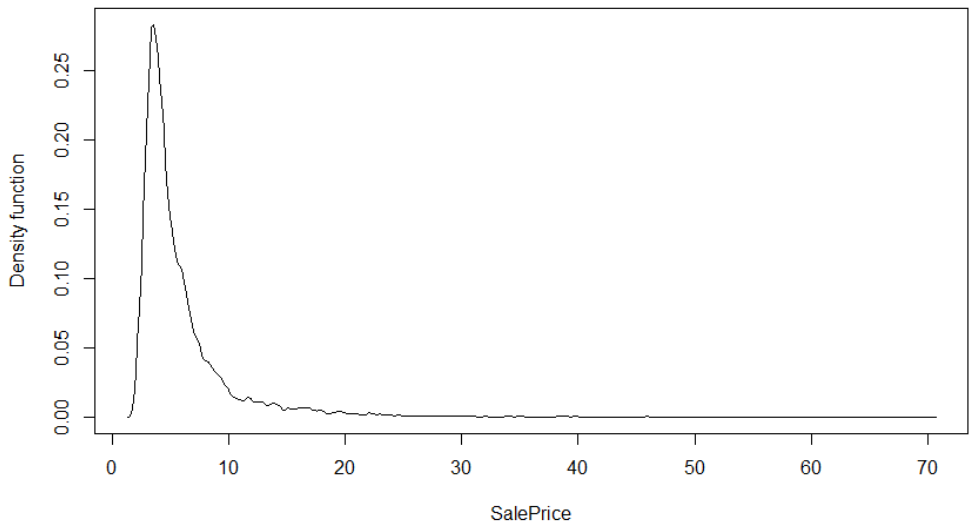

Fig. 3

Fig. 2 and Fig. 3 show the distribution of the dependent variable (SalePrice). As evident, the distribution lies between **0 to inf** and has a peak near zero. The curve flattens as it increases and moves towards inf, which gives it a long tail. This shows that **Gamma distribution** will be well suited as error distribution of this data.

3. Mathematical Model:

The following Multiple Linear Regression model is implemented on the data:

$Y = ß_0 + ß_1X_1 + ß_2X_2 + ß_3X_3 + ß_4X_4 + ß_5X_5 + \varepsilon$

$\varepsilon \sim \text{Gamma}\ (\mu^2/\sigma^2,\ \mu/\sigma^2)$

where,

Y = Model curve in 2-D space.

$ß_{0..5}$ = Model parameters that are assumed to be normally distributed priors.

$X_{1..5}$ = Independent variables in data

ε = Error distribution

This implies:

$\mu = ß_0 + ß_1X_1 + ß_2X_2 + ß_3X_3 + ß_4X_4 + ß_5X_5$

and, Y is assumed to have a **Gamma distribution**.

Error is assumed to have a Gamma distribution by looking at the descriptive analysis of the data. There are **5 regression parameters** in this model.

4. Specification of Prior and Building JAGS Model Diagram:

The JAGS Model Diagram for this model is:

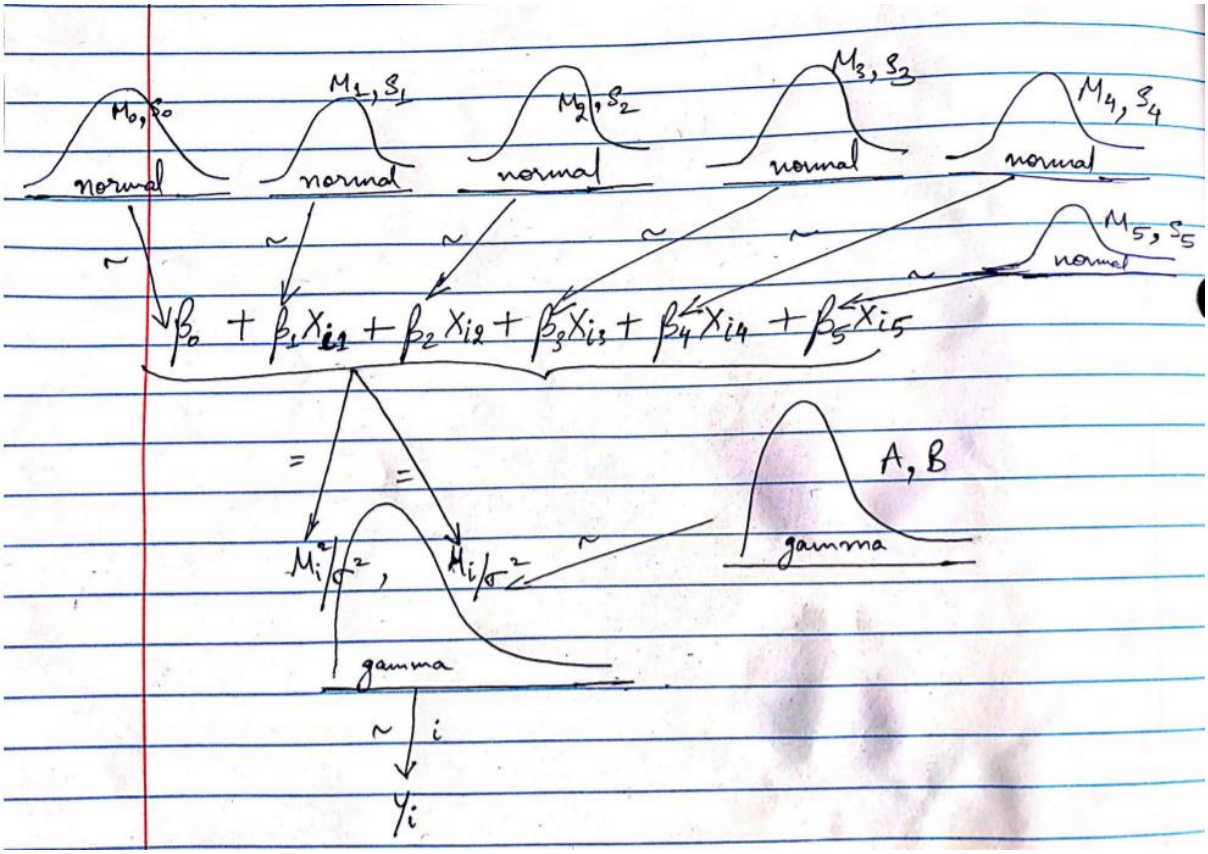

Fig. 4

Evidently, the variance across Y is assumed to be constant across the values of independent variables. This is the homogeneity of variances assumption. All the independent features are assumed to be uncorrelated to avoid **multicollinearity issue**. Regression parameters are assumed to have a normal distribution. The regression model is fit on the mean of gamma distributed error distribution.

The expert information and degree of belief for every predictor is given as follows:

- Area: Every unit increase increases the Sale Price by 90 AUD. It is a **very strong** expert knowledge. (μ = 90/100000)
- Bedrooms: Every additional bedroom increases the Sale Price by 100,000 AUD. It is a **weak** expert knowledge. (μ = 1)
- Bathrooms: There is **no** expert knowledge on number of bathrooms. (μ = 0)
- CarParks: Every additional Car Park increases the Sale Price by 120,000 AUD. It is a **strong** expert knowledge. (μ = 1.2)

- PropertyType: A “unit” will cost 150,000 AUD less than “house” at an average. This is a **very strong** expert knowledge. ($\mu = -1.5$)

5. Finding the posterior distribution:

The posterior distribution and predictions for desired set of values of predictors are obtained by running R scripts.

The **JAGS Model Diagram** is coded into R and predictions are made based on the equation:

$$\text{Sale Price} = ß_0 + ß_1.\text{Area} + ß_2.\text{Bedrooms} + ß_3.\text{Bathrooms} + ß_4.\text{CarParks} + ß_5.\text{PropertyType} + \varepsilon$$

$$\varepsilon \sim \text{Gamma}(\mu^2/\sigma^2, \mu/\sigma^2)$$

The regression parameters $ß_{0..5}$ have a high correlation among themselves which is a major drawback as it causes disruptions in MCMC diagnostics. Hence, scaling or standardization is performed to overcome this issue. Beta parameters are replaced by scaled ‘zbeta’ parameters. The formula for scaling these values is:

$$z_x = x/SD_x \text{ , } z_y = y/SD_y$$

where, SD_x and SD_y are standard deviations of independent X and Y variables, respectively.

(X_i: Independent variables or predictors

Y: Sale Price)

JAGS programming is done using Rstudio where the multiple linear regression model is run with scaling using parallel processing.

Steps followed to build and run the model in JAGS:

- Loading the data in R after installing all the necessary packages required to run a JAGS model.
- The data is converted to matrix format using the command ‘as.matrix’ to determine correlation between the independent variables. If there is a significant correlation between two predictors, it is better to avoid using one of them as it adversely affects the MCMC diagnostics and the results are bad.

- To find the correlation, we plot the correlation matrix using R functionalities and get the following result:

```
CORRELATION MATRIX OF PREDICTORS:
 > show( round(cor(x),4) )
                Area Bedrooms Bathrooms CarParks PropertyType
Area          1.0000  -0.2699   -0.0873  -0.0963       0.3202
Bedrooms     -0.2699   1.0000    0.5382   0.4351      -0.5601
Bathrooms    -0.0873   0.5382    1.0000   0.3658      -0.2897
CarParks     -0.0963   0.4351    0.3658   1.0000      -0.3600
PropertyType  0.3202  -0.5601   -0.2897  -0.3600       1.0000
> cat("\n")
```

Fig. 5

In Fig. 5, we can see that there is a high correlation between **Area and PropertyType, Bedrooms and Bathrooms, Bedrooms and CarParks, and CarParks and Bathrooms**. This correlation interferes with generation of good MCMC diagnostics and hence, these independent variables should be tested sparetely for better results.

- To ship the data to JAGS, we create a list and pass the data into it.
- Initials are set for each parametric value. If the initials are not manually set, JAGS assigns them automatically.
- Now, we specify the model in double quotes (as text). R cannot read this specification, but JAGS can read and process it.
- Inside the model text, we implement the JAGS model diagram (Fig. 4) and specify all the distributions and parameters using functions such as ‘dgamma()’ for specifying gamma distribution and ‘dnorm()’ for specifying normal distribution.
- **Scaling** is implemented inside model text, followed by model specification.
- Now, we pass the prior information using **dnorm()** function. This is done because all the regression parameters are assumed to be normally distributed.
- After this processing completes, scaled data is transformed into its original form to produce appropriate desired outputs for predictions.
- Computation of predictions using given predictor values is performed.
- Now, this model specification is written into an external text file and parameter vectors are specified.
- **MCMC settings** are made, for instance, number of burn-in steps and number of saved steps are given as an input.

- Now, Parallel running is performed using **run.jags()** function. The model text file is taken as an input and model is built using the specified MCMC parameters.
- After the model is created, **codaSamples** are generated. These are used to plot MCMC diagnostics which help us check the reliability of the model.
- Summary statistics are generated and displayed and **goodness of fit** of generated model is checked by plotting predicted values against observed values using **ggplot()** function.

5.1. **Experimentation with different variance values:**

Priors specified for all the regression parameters (predictors) is assumed to be normally distributed and variance depends on the degree of belief in prior information. For a very strong knowledge, variance should be minute as degree of belief is very high. For strong knowledge variance should be small as degree of belief is high and for weak knowledge, variance should be high as degree of belief is low. The following table displays the tested variance values that were put into model one-by-one:

INDEPENDENT PARAMETERS	VARIANCE MODEL I	VARIANCE MODEL II	VARIANCE MODEL III	VARIANCE MODEL IV
Area (very strong knowledge)	0.1	0.01	0.001	0.0001
Bedrooms (weak knowledge)	4	4	9	10
Bathrooms (no knowledge)	4	5	6	9
CarParks (strong knowledge)	1	0.5	0.5	0.1
PropertyType(very strong knowledge)	0.1	0.01	0.001	0.0001

Table 1

5.2 Experimentation with MCMC parameters:

MCMC Parameters	Model I	Model II	Model III	Model IV
adaptSteps	1500	1500	1500	1500
burnInSteps	3000	4000	5000	6000
nChains	3	3	3	3
thinSteps	3	7	15	31
numSavedSteps	4000	5000	7000	10000

Table 2

After careful examination of all 4 models and checking their goodness of fit, it is found that Model IV fits best and hence Model IV is analysed and explained further in this report. The other models can be analysed in a similar fashion and be used for making predictions. Here, Model IV is used for making predictions.

6. Assessment of MCMC Diagnostics for each parameter:

- Beta0:

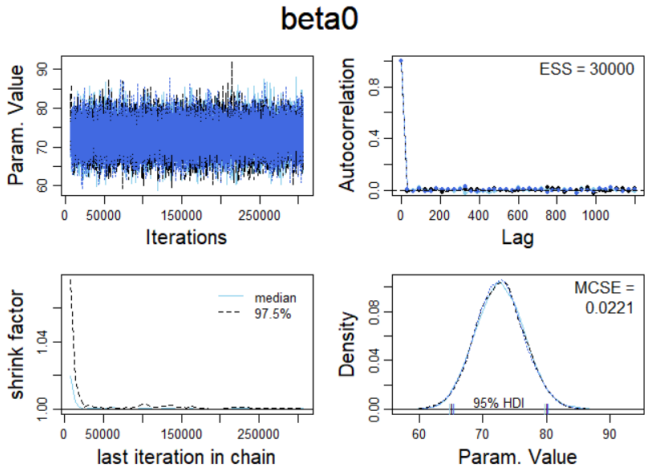

Fig. 6

We can see, **shrink factor is less than 1.2**, **chains overlap** appropriately, **density plots and HDI limits overlap** significantly. The **autocorrelation is minute** and Estimated Sample Size **(ESS) is large**. **MCSE (standard error) is very small**. Hence, we can imply that the Bayesian estimate of Beta0 is reliable.

- Beta1:

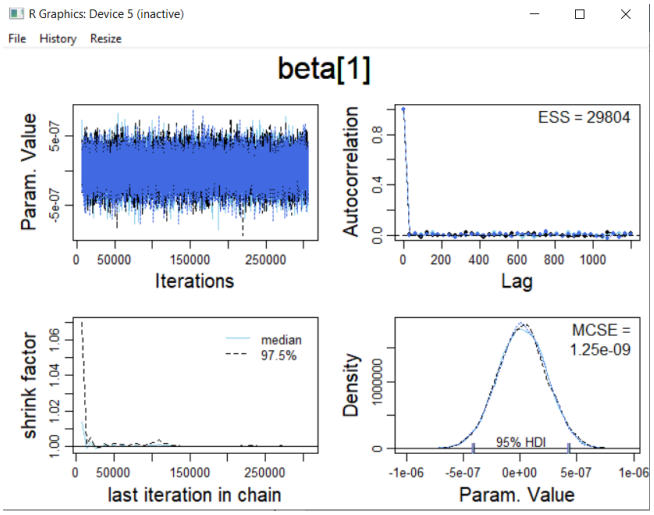

Fig. 7

We can see, **shrink factor is less than 1.2**, **chains overlap** appropriately, **density plots and HDI limits overlap** significantly. The **autocorrelation is minute** and Estimated Sample Size **(ESS) is large**. **MCSE (standard error) is very small**. Hence, we can imply that the Bayesian estimate of Beta1 is reliable.

- Beta2:

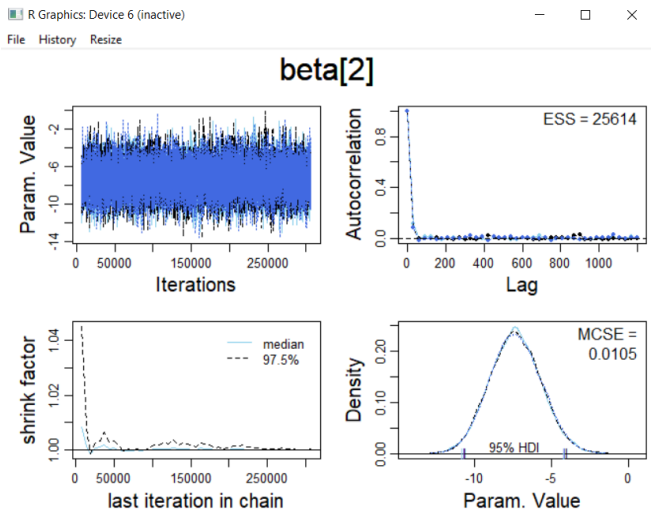

Fig. 8

We can see, **shrink factor is less than 1.2**, **chains overlap** appropriately, **density plots and HDI limits overlap** significantly. The **autocorrelation is minute** and Estimated Sample Size **(ESS) is large**. **MCSE (standard error) is very small**. Hence, we can imply that the Bayesian estimate of Beta2 is reliable.

- Beta3:

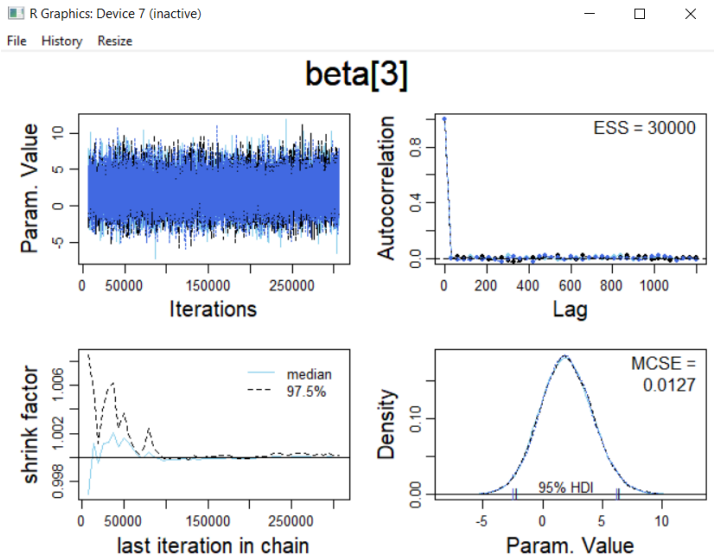

Fig. 9

We can see, **shrink factor is less than 1.2**, **chains overlap** appropriately, **density plots and HDI limits overlap** significantly. The **autocorrelation is minute** and Estimated Sample Size **(ESS) is**

large. **MCSE (standard error) is very small**. Hence, we can imply that the Bayesian estimate of Beta3 is reliable.

- Beta4:

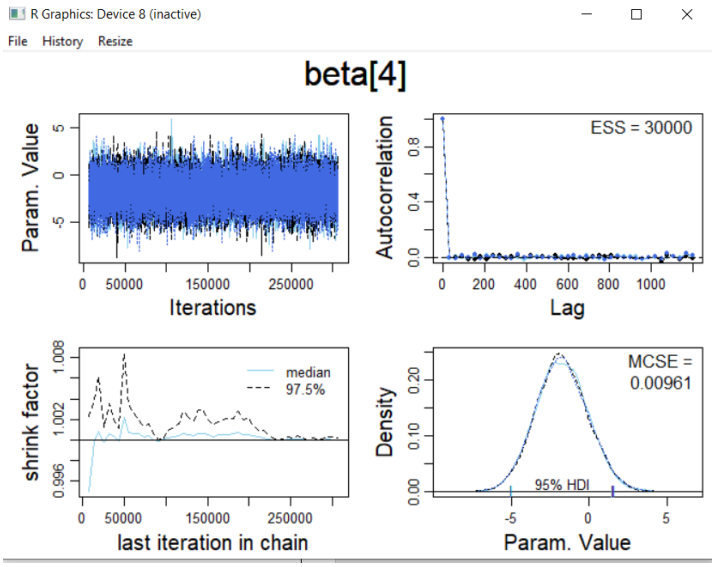

Fig. 10

We can see, **shrink factor is less than 1.2**, **chains overlap** appropriately, **density plots and HDI limits overlap** significantly. The **autocorrelation is minute** and Estimated Sample Size **(ESS) is large**. **MCSE (standard error) is very small**. Hence, we can imply that the Bayesian estimate of Beta4 is reliable.

- Beta5:

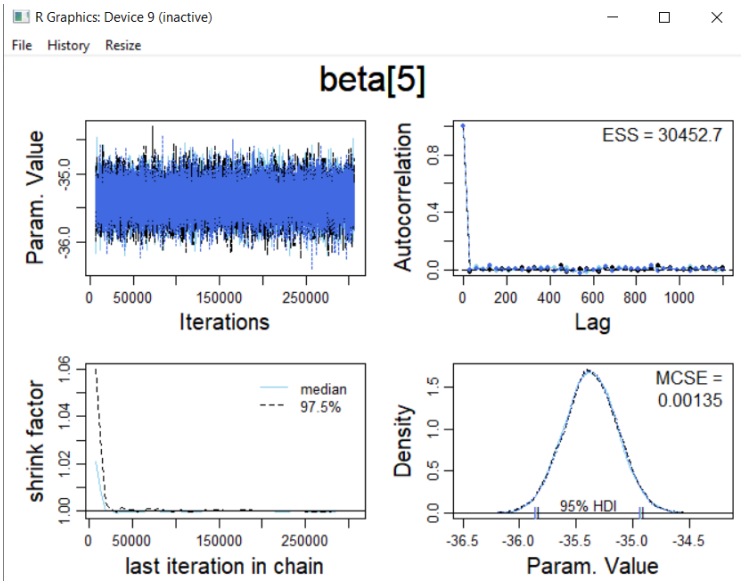

Fig. 11

We can see, **shrink factor is less than 1.2**, **chains overlap** appropriately, **density plots and HDI limits overlap** significantly. The **autocorrelation is minute** and Estimated Sample Size **(ESS) is large**. **MCSE (standard error) is very small**. Hence, we can imply that the Bayesian estimate of Beta5 is reliable.

- Tau (Variance):

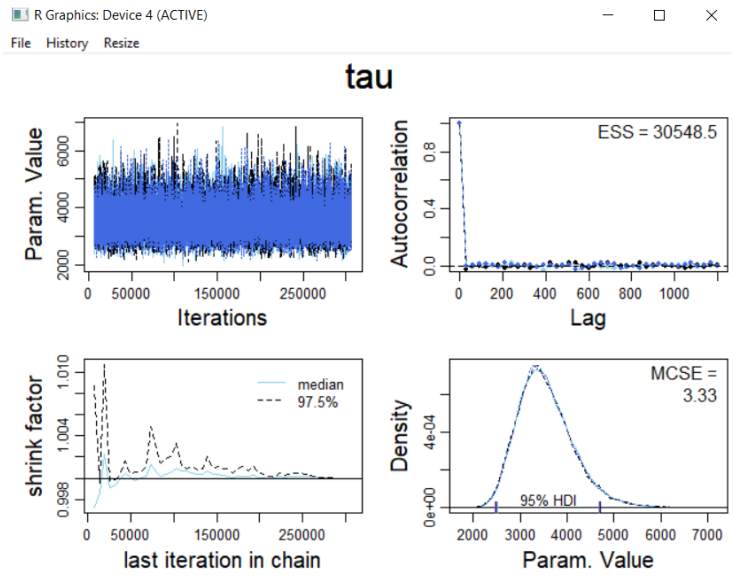

Fig. 12

We can see, **shrink factor is less than 1.2**, **chains overlap** appropriately, **density plots and HDI limits overlap** significantly. The **autocorrelation is minute** and Estimated Sample Size **(ESS) is large**. **MCSE (standard error) is very small**. Hence, we can imply that the Bayesian estimate of tau is reliable.

7. Posterior Distribution and Bayesian Estimations for each parameter:

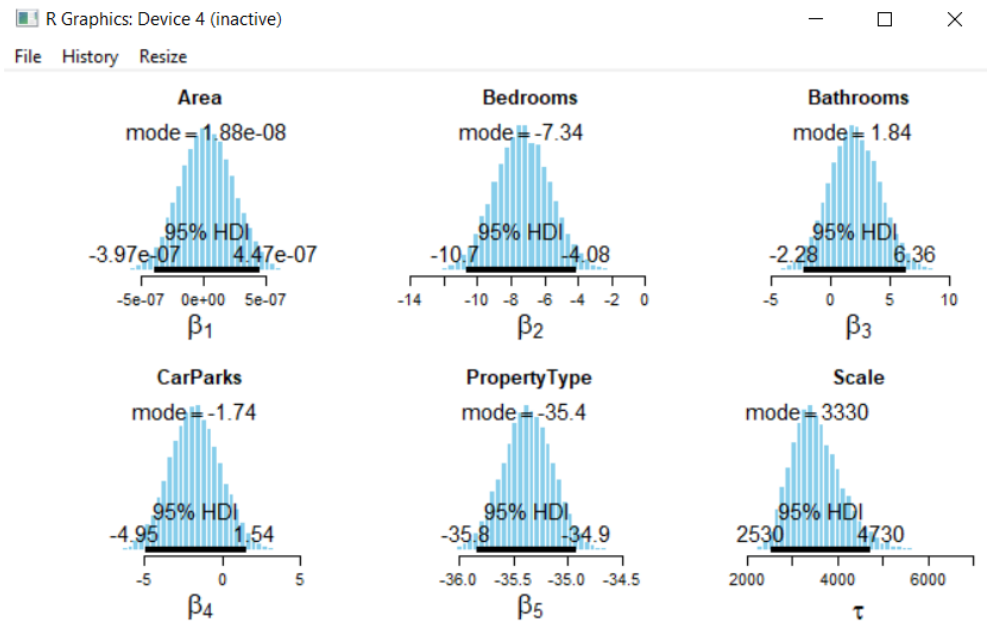

Fig. 13

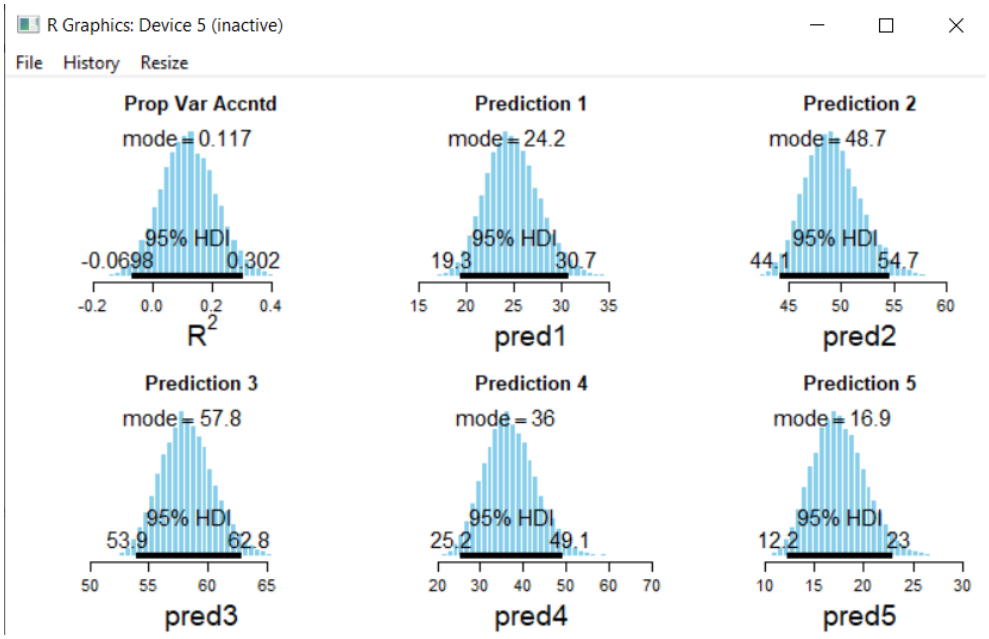

Fig. 14

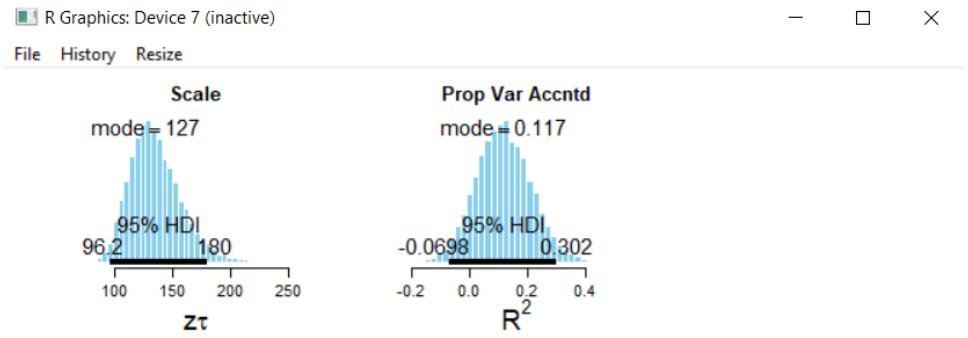

Fig. 15

Fig. 13, 14, and 15 show Posterior distributions of all parameters. The significance of each regression parameter in finding the dependent variable (SalePrice) can be determined by looking at these plots.

- Area: 0 lies significantly inside the 95% HDI. Hence, **Area is a significant parameter**.
- Bedrooms: 0 is significantly outside the range of HDI and hence, **Bedrooms are very less significant.**
- Bathrooms: 0 lies significantly inside the 95% HDI. Hence, **Bathrooms is a significant parameter.**
- CarParks: 0 lies significantly inside the 95% HDI. Hence, **CarParks is a significant parameter**.
- PropertyType: 0 is significantly outside the range of HDI and hence, **significance of Bedrooms is negligible** as HDI limits are far away from 0.

R^2 value can be improved by changing MCMC settings such as increasing the burn-in, thinning, and saved steps. It can also be improved by assuming Exponential distribution of error rather than Gamma.

The Bayesian point and 95% HDI estimates for the parameters are as follows:

Beta0 = Mode: 7.27e+01, HDI = 6.52e+01-8.01e+01

Beta1 = Mode: 1.87e-08, HDI = -3.97e-07-4.46e-07

Beta2 = Mode: -7.34, HDI = -1.06e+01-4.07

Beta3 = Mode: 1.84, HDI = 2.27-6.35

Beta4 = Mode: -1.73, HDI: -4.95-1.53

Beta5 = Mode: -3.53e+01, HDI: -3.58e+01 - -3.49e+01

Tau = Mode: 3.33e+03, HDI: 2.52e03-4.73e+03

8. Predictions Model:

Predictions model based on the Bayesian Analysis is:

μ = 7.27e+01 + 1.87e-08.Area + (-7.34).Bedrooms + 1.84.Bathrooms + (-1.73).CarParks + -3.53e+01.PropertyType

9. Predictions of Sale Prices for given set of variables:

Property No	Area	Bedrooms	Bathrooms	CarParks	PropertyType	Predicted Sale Price (in 100,000 AUD)
1	600	2	2	1	Unit	24.206
2	800	3	1	2	House	48.70
3	1500	2	1	1	House	57.832
4	2500	5	4	4	House	36.014
5	250	3	2	1	Unit	16.898

Table 3

95% HDI Bayesian estimates for all the predictions are:

- Pred1: 19.3-30.7
- Pred2: 45.1-54.6
- Pred3: 53.8-62.8
- Pred4: 25.2-49.1
- Pred5: 12.2-22.9

10. Assessment of MCMC Diagnostics for Predictions:

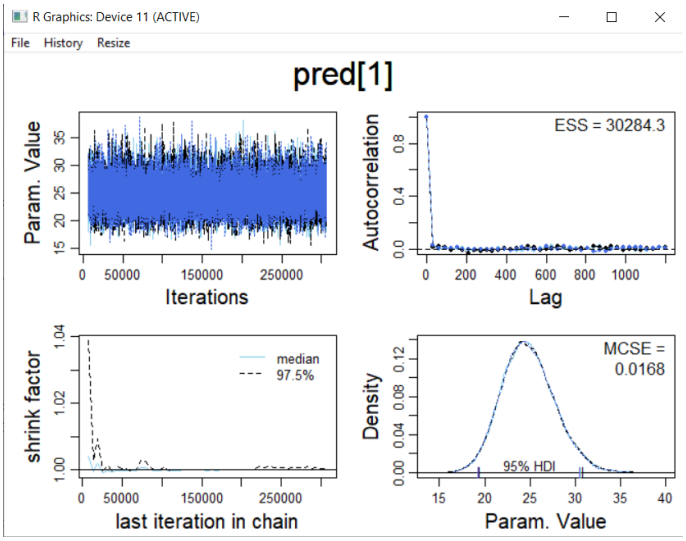

Fig. 16

We can see, **shrink factor is less than 1.2**, **chains overlap** appropriately, **density plots and HDI limits overlap** significantly. The **autocorrelation is minute** and Estimated Sample Size **(ESS) is large**. **MCSE (standard error) is very small**. Hence, we can imply that the Bayesian estimate of pred1 is reliable.

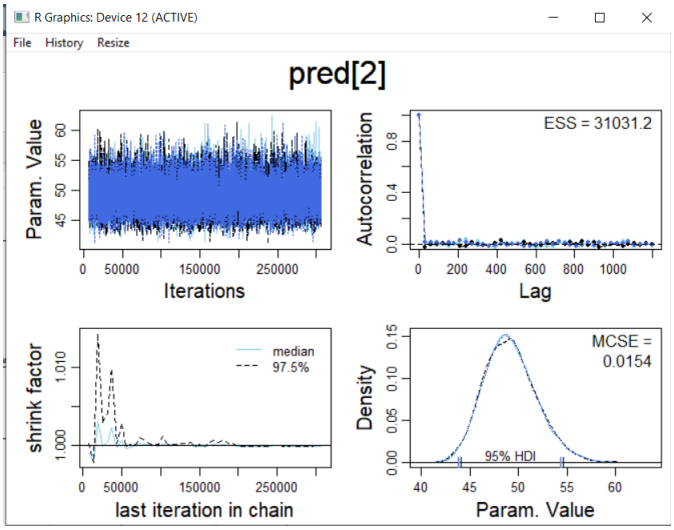

Fig. 17

We can see, **shrink factor is less than 1.2**, **chains overlap** appropriately, **density plots and HDI limits overlap** significantly. The **autocorrelation is minute** and Estimated Sample Size **(ESS) is large**. **MCSE (standard error) is very small**. Hence, we can imply that the Bayesian estimate of pred2 is reliable.

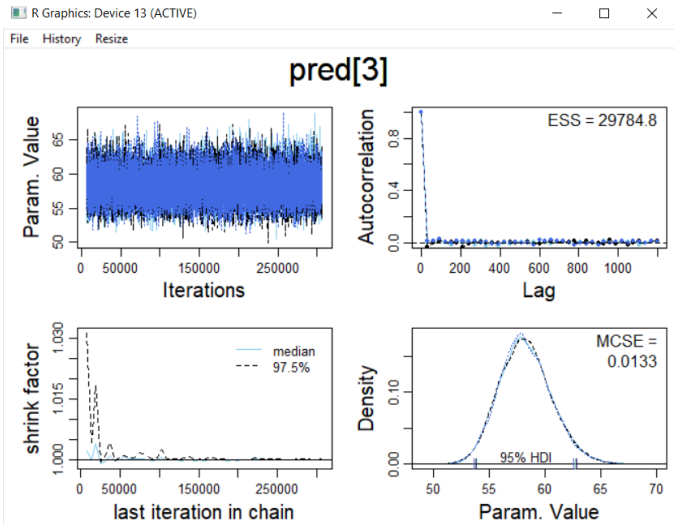

Fig. 18

We can see, **shrink factor is less than 1.2**, **chains overlap** appropriately, **density plots and HDI limits overlap** significantly. The **autocorrelation is minute** and Estimated Sample Size **(ESS) is large**. **MCSE (standard error) is very small**. Hence, we can imply that the Bayesian estimate of pred3 is reliable.

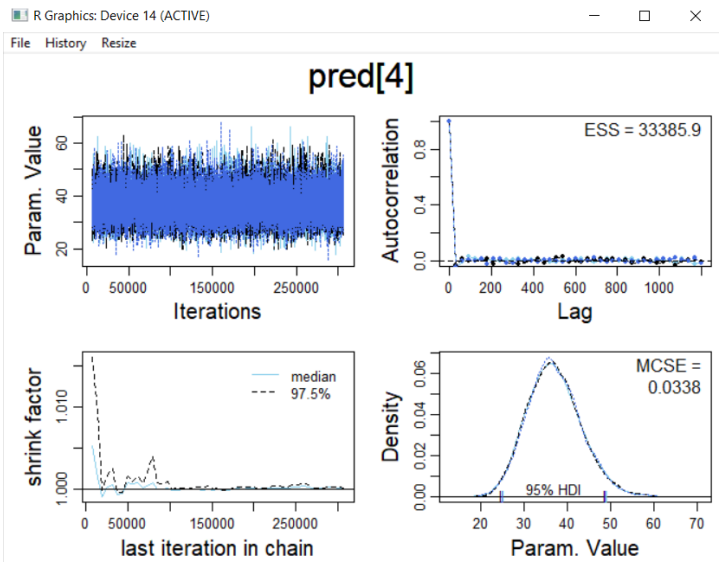

Fig. 19

We can see, **shrink factor is less than 1.2**, **chains overlap** appropriately, **density plots and HDI limits overlap** significantly. The **autocorrelation is minute** and Estimated Sample Size **(ESS) is large**. **MCSE (standard error) is very small**. Hence, we can imply that the Bayesian estimate of pred4 is reliable.

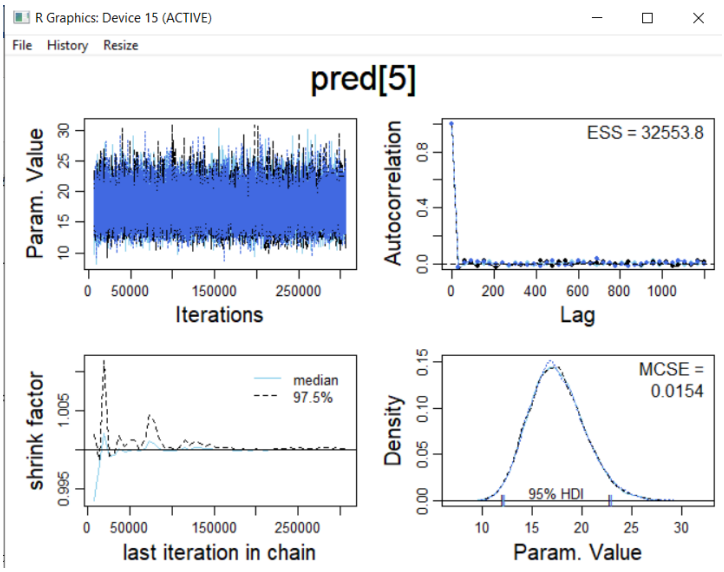

Fig. 20

We can see, **shrink factor is less than 1.2**, **chains overlap** appropriately, **density plots and HDI limits overlap** significantly. The **autocorrelation is minute** and Estimated Sample Size **(ESS) is**

large. **MCSE (standard error) is very small**. Hence, we can imply that the Bayesian estimate of pred5 is reliable.

11. Goodness of Fit:

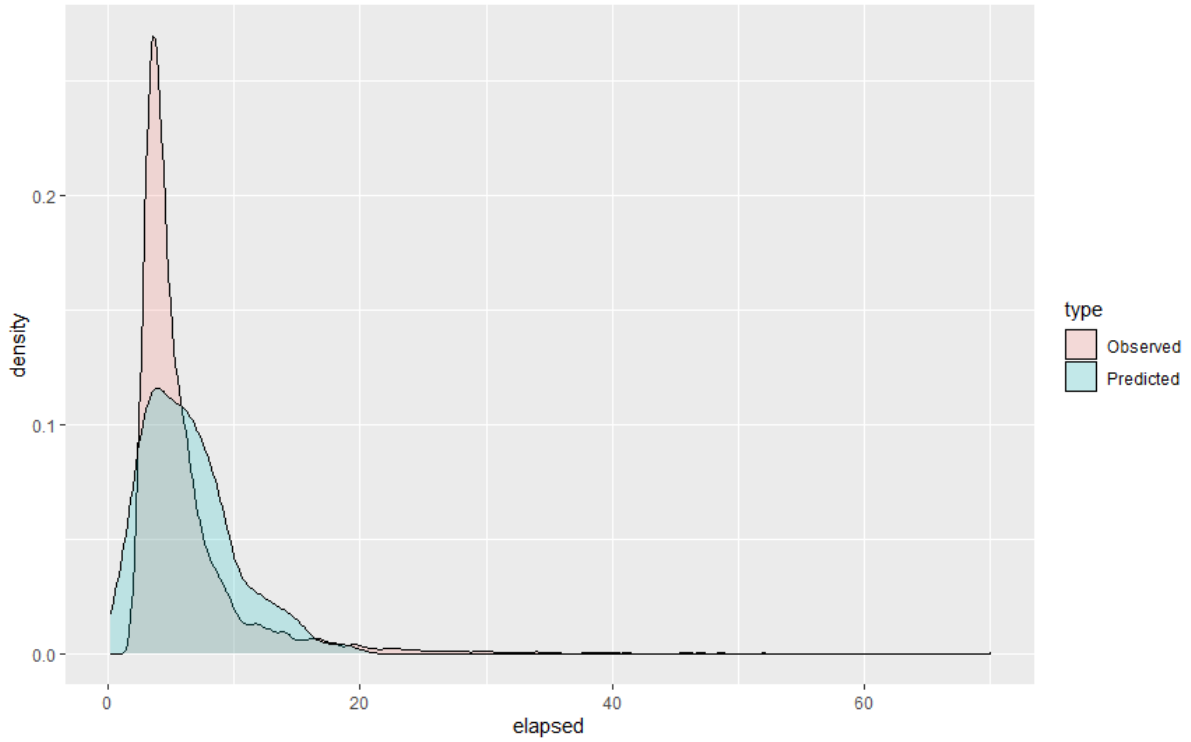

Fig. 21

The goodness of fit of the model built can be seen from Fig. 21. The model fits almost all the values including the long tail of the data but the peak of observed data is higher than that of predicted values. There is no overfitting issue here but the model fit might improve if exponential distribution is assumed as error distribution instead of Gamma distribution.

12. Conclusion and Inferences:

- There is a high correlation between **Area and PropertyType, Bedrooms and Bathrooms, Bedrooms and CarParks, and CarParks and Bathrooms**. This correlation interferes with generation of good MCMC diagnostics and hence, these independent variables should be tested sparetely for better results.
- **Area, Bathrooms, and CarParks** are the most significant parameters to to determine the value of dependent variable (SalePrice).

- Values of regression parameters are most likely to lie between:
    - Beta0: 6.52e+01-8.01e+01
    - Beta1: -3.97e-07-4.46e-07
    - Beta2: -1.06e+01-4.07
    - Beta3: 2.27-6.35
    - Beta4: -4.95-1.53
    - Beta5: -3.58e+01 - -3.49e+01

13. Drawbacks

- The model was tried and tested with one set of initials: zbeta0 = 2000, zbeta1 = 100, zbeta2 = 1, zbeta3 = 1, zbeta4 = 0.5, zbeta5 = 0, Var = 12000000. **More sets of initials** can be tried and tested for examining more number of scenarios.
- **Number of chains** for generation of MCMC diagnostics was kept 3 for every model testing. This value can be fluctuated for better results.
- The model was built with independent variables with **high correlation** values. To get better estimates, these parameters could be taken separately and analysed.
- **The model fit** can be improved highly as distribution of error was assumed to be Gamma distribution, but **Exponential distribution** might give better results too as the peak of data is near 0 and it has a long tail tending towards inf.
- In this analysis, 4 models with varied parameters, both regression and MCMC, were tested, but **more number of models can be constructed** with different values of these parameters to attain better results.

14. Appendix

```
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 )
}
```

Fig. 22

```
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]"]
  pred4 = mcmcMat[,"pred[4]"]
  pred5 = mcmcMat[,"pred[5]"]
  #-----------------------------------------------------------------------------
  # 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)
    }
  }
```

Fig. 23

```
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:
```

Fig. 24

```
# THE DATA.
y = myData[,"SalePrice.100K."]
x = as.matrix(myData[,c("Area","Bedrooms","Bathrooms","CarParks","PropertyType")])

# Some more descriptives
cat("\nCORRELATION MATRIX OF PREDICTORS:\n ")
show( round(cor(x),4) )
cat("\n")

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)
xPred

# 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]
)
```

Fig. 25

```
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
  zbeta[1] ~ dnorm( 0.0009/xsd[1] , 1/(0.0001/xsd[1]^2) ) # 1/ variance for normal distribution
  zbeta[2] ~ dnorm( 1/xsd[2] , 1/(10/xsd[2]^2) ) # 1/ variance for normal distribution
  zbeta[3] ~ dnorm( 0 , 1/4 ) # 1/ variance for normal distribution
  zbeta[4] ~ dnorm( 1.2/xsd[4] , 1/(1/xsd[4]^2) ) # 1/ variance for normal distribution
  zbeta[5] ~ dnorm( -1.5/xsd[5] , 1/(0.0001/xsd[5]^2) )
  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
  }
 # pred[1]<- beta0 + beta[1]*xPred[1,1] + beta[2]*xPred[1,2] + beta[3]*xPred[1,3] + beta[4]*xPred[1,4] + beta[5]*xPred[
  #pred[2]<- beta0 + beta[1]*xPred[2,1] + beta[2]*xPred[2,2] + beta[3]*xPred[2,3] + beta[4]*xPred[2,4] + beta[5]*xPred[
  #pred[3]<- beta0 + beta[1]*xPred[3,1] + beta[2]*xPred[3,2] + beta[3]*xPred[3,3] + beta[4]*xPred[3,4] + beta[5]*xPred[
#  pred[4]<- beta0 + beta[1]*xPred[4,1] + beta[2]*xPred[4,2] + beta[3]*xPred[4,3] + beta[4]*xPred[4,4] + beta[5]*xPred[
 # pred[5]<- beta0 + beta[1]*xPred[5,1] + beta[2]*xPred[5,2] + beta[3]*xPred[5,3] + beta[4]*xPred[5,4] + beta[5]*xPred[

}
```

Fig. 26

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

adaptSteps = 1500  # Number of steps to "tune" the samplers
burnInSteps = 5000
nChains = 3
thinSteps = 30 # First run for 3
numSavedSteps = 10000
nIter = ceiling( ( numSavedSteps * thinSteps ) / nChains )

# Parallel run
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 )

# save.image(file="rEnvironment.RData")
#load(file="rEnvironment.RData") # Load the results with 124,000 iterations

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]" )
```

Fig. 27

```
summaryInfo <- smryMCMC_HD( codaSamples = codaSamples , compVal = compVal )
print(summaryInfo)


plotMCMC_HD( codaSamples = codaSamples , data = myData, xName=c("Area","Bedrooms","Bathrooms","CarParks","PropertyTy
          yName="SalePrice.100K.", compVal = compVal)

# ============ Predictive check ============
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)
```

Fig. 28

15. Bibliography

- The data was provided by course instructor through Canvas.
- The R and JAGS codes used to build and run Multiple Linear Regression Model were inspired by codes developed and provided by **Dr. Hayadar Demirhan**.