MATH2269 - Assignment 2

Implementing Multiple Linear Regression model to predict

Property Sales Prices in Melbourne

using Bayesian Statistics

Table of Contents

1. Introduction

In this report we will be implementing a Bayesian Multiple Linear Regression Model to predict the sales price of properties in Melbourne. While implementing this model, we will take into consideration the data available for sales price and also the expert information given by real estate agent.

The data used for this analysis consists of 6 variables. Out of these 6 variables, *SalePrice* is the dependent variable and *Area*, *Bedrooms*, *Bathrooms*, *CarParks* and *PropertyType* are the independent variables(predictors). The size of data available is 10,000.

Considering the above mentioned dependent and independent variables, the linear regression equation can be given as:

$$\textbf{SalePrice} = \beta_0 + \beta_1\textbf{Area} + \beta_2\textbf{Bedrooms} + \beta_3\textbf{Bathrooms} + \beta_4\textbf{CarParks} + \beta_5\textbf{PropertyType}$$

In the above equation, β_0 is the intercept and β_1, β_2, β_3, β_4, β_5 are the coefficients for predictors *Area*, *Bedrooms*, *Bathrooms*, *CarParks* and *PropertyType* respectively.

In this analysis, we will try to find the Bayesian estimate for coefficient of every parameter (predictor). We will perform this estimation using JAGS (Just Another Gibbs Sampler). While estimating the coefficients, we will include the given expert knowledge as the prior information for each parameter along with the data available. We will then check for MCMC diagnostics for each parameter to ensure that estimation has been correctly done. Then we will find our Bayesian point estimate for every coefficient by observing the posterior distribution of all the coefficients. After obtaining these Bayesian estimates, we will use the above regression equation to predict the sales price using different values for *Area*, *Bedrooms*, *Bathrooms*, *CarParks* and *PropertyType*. Lastly, using the obtained regression model and available data, we will compute the predicted sales price and then compare the KDE(Kernel Density Estimate) of observed sales price and the predicted sales price.

We will start the analysis by performing a descriptive check on data available.

2. Descriptive Check

To define the Bayesian model for above specified regression equation, it is important to know the distribution of the variable which we are trying to predict, which in this case would be *SalePrice*. So, the KDE for SalePrice can be given as below:

(P.T.O.)

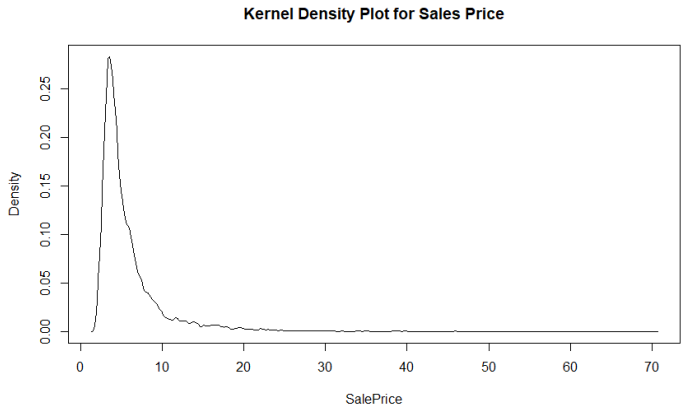

Figure 1: Kernel Density Estimate plot for *SalePrice* (Refer Appendix 1 for R code)

From the above output, we can see that the curve corresponds to a gamma distribution. So, for our analysis, we can consider a normal-gamma model where the posterior distribution and likelihood will follow a gamma distribution and the prior distributions can be specified as in the form of normal distributions.

Now, using scatter plots, we will check the relation of dependent variable *SalePrice* with every independent variable. The respective scatter plots are as follows:

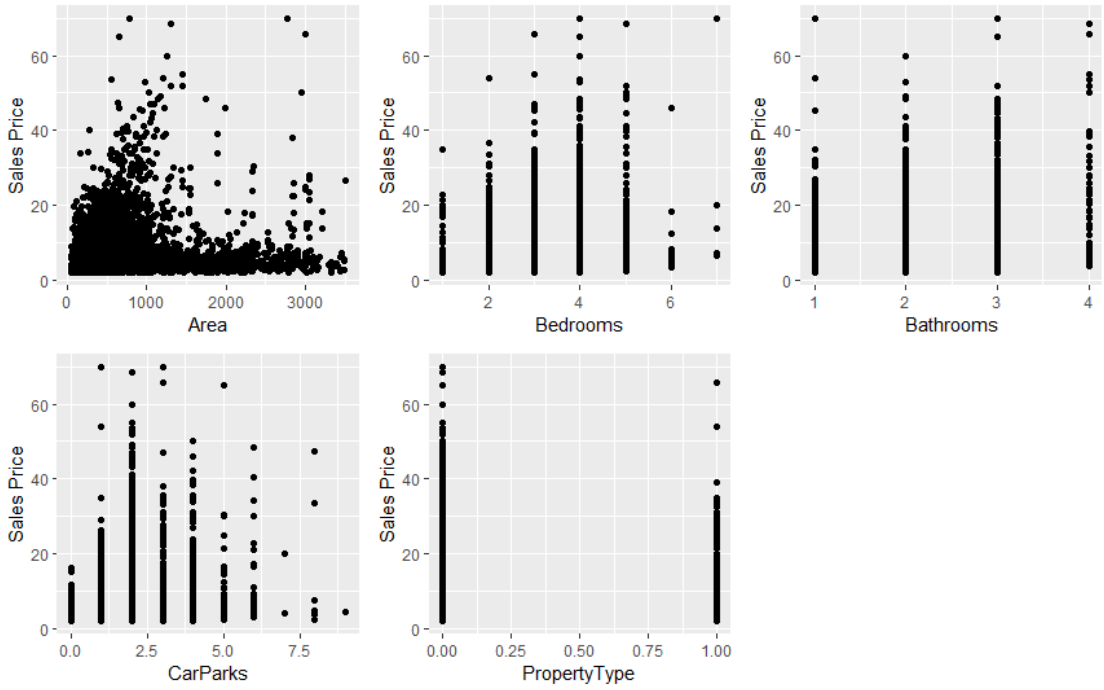

Figure 2: Scatter plots to represent the relation of dependent variable with independent variables (Refer Appendix 2 for R code)

As seen in the above output, we cannot see a strong relation of *SalePrice* with any of the independent variable. Other thing which we can see from above output is that *Area* is a continuous variable. *Bedroom*, *Bathroom* and *CarParks* are discrete variables and *PropertyType* is a binary variable. From the description of data available for *PropertyType*, 0 represents a house and 1 represents a unit.

Lastly, while implementing a regression model, it is important that there is no significant correlation present between the independent variables. To check this, we will look at the correlation matrix of all the independent variables. The correlation matrix is as below:

	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 3: Correlation matrix for independent variables (Refer Appendix 3 for R code)

From the above output we can see that there is no significant correlation present between any of the independent variables.

Now that we have a better understanding of the data to be used, we will proceed for defining the Bayesian Regression Model.

3. Model Specification

As mentioned earlier, we will implement this Bayesian Regression Model using JAGS. The model can be given as below:

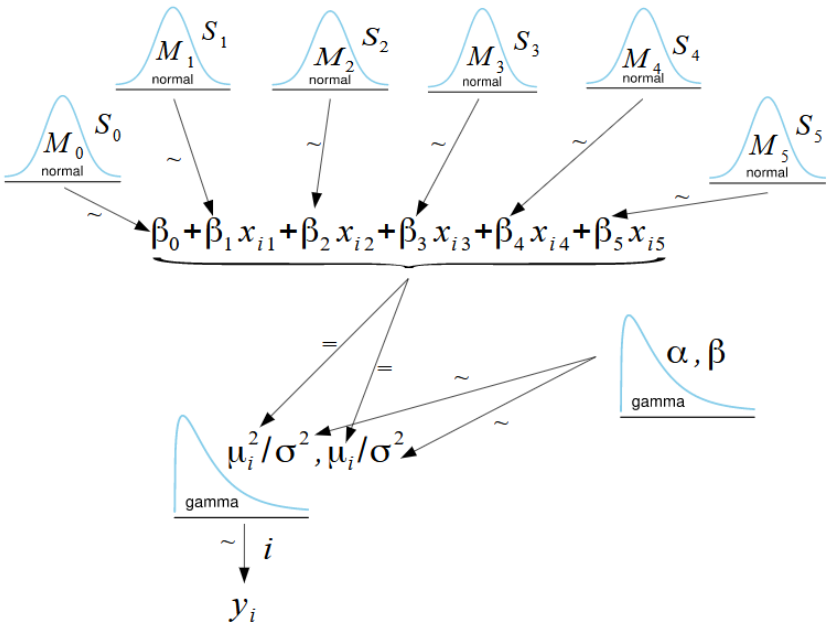

Figure 4: Model diagram for Bayesian Regression Analysis

In the above model, y_i represents the posterior distribution of predicted *SalePrice*.

Gamma(μ_i^2/σ^2, μ_i/σ^2) represents likelihood. As we know likelihood represents the data. We saw in the scatter plots that values for all the independent variables were positive. Also, we saw that normal-gamma model would be suitable here. So, we chose gamma model to represent the likelihood. To consider the mean(μ) and variance(σ) in gamma model, we performed re-parameterization. We used the first parameter of gamma distribution as μ_i^2/σ^2 and the second parameter of gamma distribution as μ_i/σ^2. Here, mean(μ_i) will represent the predicted value and variance(σ^2) will represent the confidence level in the predicted value.

From above we know that variance(σ^2) will represent the level of confidence in estimated value. So, as a part of the Bayesian estimation, we will define a gamma prior distribution gamma(α, β) for variance(σ^2).

Now, as mean(μ_i) will represent the predicted value in gamma(μ_i^2/σ^2, μ_i/σ^2) distribution, using the above specified regression equation, mean(μ_i)can be given as:

$$\beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \beta_3 x_{3i} + \beta_4 x_{4i} + \beta_5 x_{5i}.$$

where x_{1i}, x_{2i}, x_{3i}, x_{4i} and x_{5i} represent the data for variables *Bedroom*, *Bathroom* and *CarParks* are discrete variables and *PropertyType* respectively. Here, as along with the prediction y_i we are trying to find the Bayesian estimate for β_0, β_1, β_2, β_3, β_4, and β_5, we will define a distribution for each of them. These distributions will also represent the prior information of how various parameters affect the sales price.

As β_0, β_1, β_2, β_3, β_4, and β_5 are the intercept and coefficients of a regression equation, they can take any value from (-∞, +∞). So, we will use normal distribution to represent these priors. Moreover, the mean and variance for each of these prior distributions will help us specify the expert knowledge.

So, the prior distributions for β_0, β_1, β_2, β_3, β_4, and β_5 can be given as:

$\beta_0 \sim$ normal(M_0, S_0)

$\beta_1 \sim$ normal(M_1, S_1)

$\beta_2 \sim$ normal(M_2, S_2)

$\beta_3 \sim$ normal(M_3, S_3)

$\beta_4 \sim$ normal(M_4, S_4)

$\beta_5 \sim$ normal(M_5, S_5)

Here, M_0 to M_5 are the corresponding mean values and S_0 to S_5 are the corresponding variance values. We will use these values to specify the expert knowledge for each parameter.

So, above mentioned is our model for the Bayesian regression. While specifying this model in JAGS, we need to specify 2 parts. First part will specify how to standardize the data. Second part will specify our actual model. The first and second part will altogether form the model string for our model. The model string can be found in Appendix A.

After defining the model, we will specify the prior information for coefficients of all the parameters. We will also check the sensitivity of these prior distributions on the corresponding posterior distributions.

4. Prior Specification

Here, we will specify the prior distributions observe its effect on the corresponding posterior distributions.

We will start by specifying the prior distribution for variance(σ^2). We will then specify the prior distribution for the intercept β_0. Next, we will specify the prior distribution for the coefficient of each parameter. β_1, β_2, β_3, β_4, and β_5 are the coefficients for parameters *Area*, *Bedrooms*, *Bathrooms*, *CarParks* and *PropertyType* respectively.

While specifying the prior distributions and computing their respective posterior distributions, we will use a random sample of 1000 data points from the available data of 10,000 data points. Also, the values of MCMC parameters are set as below:

Adaptation Steps: 1000

Burn-in Steps: 100

Chains: 3

Thinning Steps: 61

Saved Steps: 1800

The specification of prior distributions are as below:

4.1. For variance σ^2

As seen in the model specification, the prior distribution for σ^2 is:

$$\sigma^2 \sim \text{gamma}(\alpha, \beta)$$

where α represents first parameter of the distribution and β represents second parameter of the distribution. Here, we will specify this distribution as non-informative by setting the value of both the parameters to 0.01

So, for this prior specification, the posterior distribution for σ^2 is as below:

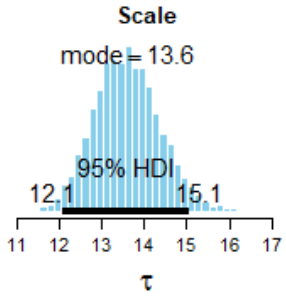

Figure 5: Posterior Distribution for σ^2 (tau) (Refer Appendix A, D, E and C for R code)

(P.T.O.)

4.2. For intercept β_0

As seen in the model specification, the prior distribution for β_0 is:

$$\beta_0 \sim \text{normal}(M_0, S_0)$$

where M_0 represents mean of the distribution and S_0 represents variance of the distribution. As β_0 is the intercept of regression model, it is not associated with any of the parameters. So, it does not represent any expert information. For this reason, we will specify this distribution as non-informative. Here, the mean(M_0) will be zero and the variance(S_0) would be 4. The variance here will not be standardized and will represent a non-informative prior.

So, for this prior specification, the posterior distribution for β_0 is as below:

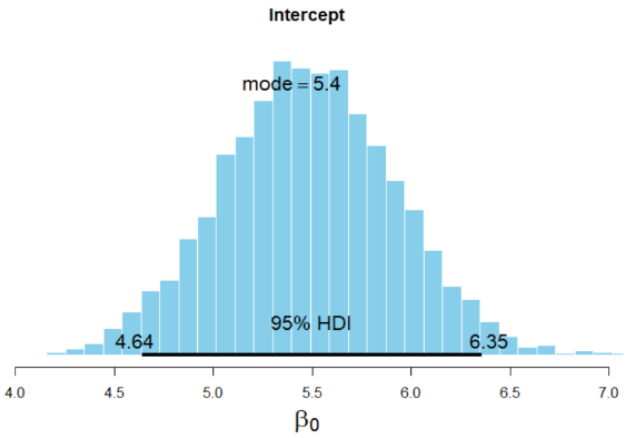

Figure 6: Posterior Distribution for β_0 (Refer Appendix A, D, E and C for R code)

From above output, we can see that the Bayesian estimate of β_0 is 5.4 with a 95% High Density Interval(HDI) of [4.64, 6.35].

4.3. For coefficient β_1 of parameter *Area*

As seen in the model specification, the prior distribution for β_1 is,

$$\beta_1 \sim \text{normal}(M_1, S_1)$$

where M_1 represents mean of the distribution and S_1 represents variance of the distribution. The prior information which we have for *Area* parameter is that every m^2 increase in land size increases the sales price by 90 AUD. This is a very strong expert knowledge.

As the sales price in data is in the scale of 100,000K, the mean(M_1) for this prior distribution would be 0.0009. As this is very strong expert knowledge, the variance(S_1) would be 1. Low value of variance represents a very strong belief in the information.

(P.T.O.)

So, for this prior specification, the posterior distribution of β_1 is as below:

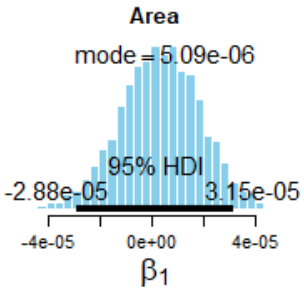

Figure 7: Posterior Distribution for β_1 (Refer Appendix A, D, E and C for R code)

From above output, we can see that the Bayesian estimate of β_1 is 0.000509 with a 95% High Density Interval(HDI) of [0.00288, 0.00315]. To check the sensitivity of above posterior distribution, we will change the variance(S_1) from 1 to 0.1. This means, we will make our belief in the expert information stronger.

So, for the updated prior specification, the posterior distribution of β_1 is as below:

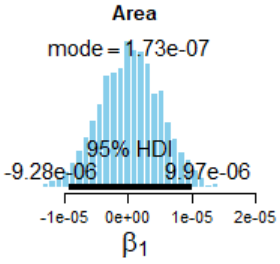

Figure 8: Posterior Distribution for β_1 with updated prior distribution (Refer Appendix A, D, E and C for R code)

From the above output, for change in variance(S_1) from 1 to 0.1, the Bayesian estimate for β_1 changed from 0.000509 to 0.0000173. Also, the 95% HDI change from [0.00288, 0.00315] to [0.000928, 0.000997].

4.4. For coefficient β_2 of parameter *Bedrooms*

As seen in the model specification, the prior distribution for β_2 is,

$$\beta_2 \sim \text{normal}(M_2, S_2)$$

where M_2 represents mean of the distribution and S_2 represents variance of the distribution. The prior information which we have for *Bedrooms* parameter is that for every additional bedroom, the sales price increases by 100,000AUD. This is a weak expert knowledge.

As the sales price in data is in the scale of 100,000K, the mean(M_1) for this prior distribution would be 1. As this is a weak expert knowledge, the variance(S_1) would be 10. High value of variance represents weak belief in the information.

(P.T.O.)

So, for this prior specification, the posterior distribution of β_2 is as below:

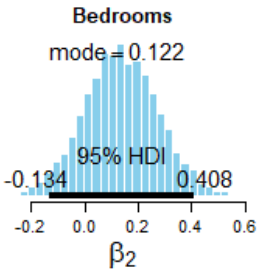

Figure 9: Posterior Distribution for β_2 (Refer Appendix A, D, E and C for R code)

From above output, we can see that the Bayesian estimate of β_2 is 0.122 with a 95% HDI of [-0.134, 0.408]. To check the sensitivity of above posterior distribution, we will change the variance(S_2) from 10 to 20. This means, we will make our belief in the expert information weaker.

So, for the updated prior specification, the posterior distribution of β_2 is as below:

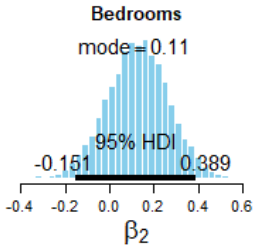

Figure 10: Posterior Distribution for β_2 with updated prior distribution (Refer Appendix A, D, E and C for R code)

From the above output, for change in variance(S_2) from 10 to 20, the Bayesian estimate for β_2 changed from 0.122 to 0.11. Also, the 95% HDI change from [-0.134, 0.408] to [-0.151, 0.389].

4.5. For coefficient β_3 of parameter *Bathrooms*

As seen in the model specification, the prior distribution for β_3 is,

$$\beta_3 \sim \text{normal}(M_3, S_3)$$

where M_3 represents mean of the distribution and S_3 represents variance of the distribution. We do not have any prior information for *Bathrooms* parameter. So, we will specify this distribution as non-informative. Here, the mean(M_3) will be zero and the variance(S_3) would be 4. The variance here will not be standardized and will represent a non-informative prior.

So, for this prior specification, the posterior distribution for β_3 is as below:

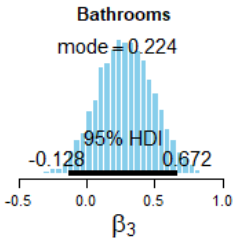

Figure 11: Posterior Distribution for β_3 (Refer Appendix A, D, E and C for R code)

From above output, we can see that the Bayesian estimate of β_3 is 0.224 with a 95% High Density Interval(HDI) of [-0.128, 0.672].

4.6. For coefficient β_4 of parameter *CarParks*

As seen in the model specification, the prior distribution for β_4 is,

$$\beta_4 \sim \text{normal}(M_4, S_4)$$

where M_4 represents mean of the distribution and S_4 represents variance of the distribution. The prior information which we have for *CarParks* parameter is that every additional car space increases the sales price by 120,000AUD. This is a strong expert knowledge.

As the sales price in data is in the scale of 100,000K, the mean(M_4) for this prior distribution would be 1.2. As this is strong expert knowledge, the variance(S_4) would be 2. This value of variance represents strong belief in the information.

So, for this prior specification, the posterior distribution of β_4 is as below:

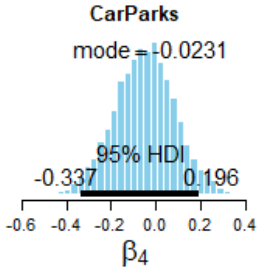

Figure 12: Posterior Distribution for β_4 (Refer Appendix A, D, E and C for R code)

From above output, we can see that the Bayesian estimate of β_4 is -0.0231 with a 95% High Density Interval(HDI) of [-0.337, 0.196]. To check the sensitivity of above posterior distribution, we will change the variance(S_4) from 2 to 1. This means, we will make our belief in the expert information stronger.

So, for the updated prior specification, the posterior distribution of β_4 is as below:

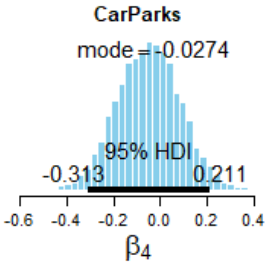

Figure 13: Posterior Distribution for β_4 with updated prior distribution (Refer Appendix A, D, E and C for R code)

From the above output, for change in variance(S_4) from 2 to 1, the Bayesian estimate for β_4 changed from -0.0231 to -0.0274. Also, the 95% HDI change from [-0.337, 0.196] to [-0.313, 0.211].

4.7. For coefficient β_5 of parameter *PropertyType*

As seen in the model specification, the prior distribution for β_5 is,

$$\beta_5 \sim \text{normal}(M_5, S_5)$$

where M_5 represents mean of the distribution and S_5 represents variance of the distribution. The prior information which we have for *PropertyType* parameter is that 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.

As the sales price in data is in the scale of 100,000K, the mean(M_5) for this prior distribution would be -1.5. As this is very strong expert knowledge, the variance(S_5) would be 1. Low value of variance represents a very strong belief in the information.

So, for this prior specification, the posterior distribution of β_5 is as below:

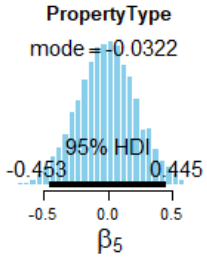

Figure 14: Posterior Distribution for β_5 (Refer Appendix A, D, E and C for R code)

From above output, we can see that the Bayesian estimate of β_5 is -0.0322 with a 95% High Density Interval(HDI) of [-0.453, 0.445]. To check the sensitivity of above posterior distribution, we will change the variance(S_5) from 1 to 0.1. This means, we will make our belief in the expert information stronger.

So, for the updated prior specification, the posterior distribution of β_5 is as below:

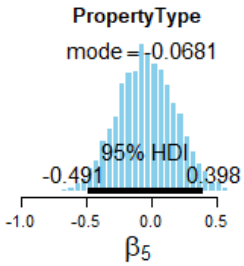

Figure 15: Posterior Distribution for β_5 with updated prior distribution (Refer Appendix A, D, E and C for R code)

From the above output, for change in variance(S_5) from 1 to 0.1, the Bayesian estimate for β_5 changed from -0.0322 to -0.0681. Also, the 95% HDI change from [-0.453, 0.445] to [-0.491, 0.398].

(P.T.O.)

5. MCMC Diagnostics

After running the Bayesian MCMC estimation, it is important to check the generated chains for their representativeness and accuracy. We also need check for the efficiency. Here, we will check the MCMC diagnostics for the chains generated of coefficients for all the parameters. We will assess them for their representativeness and accuracy. To improve the representativeness and accuracy wherever required, we will tune the MCMC parameters like burn-in steps, number of chains, number of thinning steps and number of saved steps.

As in the previous sector we observed the posterior distributions of the variance(σ^2), intercept β_0 and coefficients β_1, β_2, β_3, β_4, and β_5, here we will perform a MCMC diagnostic check on the chains generated while estimating these values. As we had considered a sample of 1000 to obtain the posterior distributions and due to technical limitations, we will again consider the same sample of 1000 for assessing MCMC diagnostics.

5.1. Initial settings

We will first check the diagnostics for MCMC chains generated by below MCMC parameters:

Adaptation Steps: 1000

Burn-in Steps: 100

Chains: 3

Thinning Steps: 11

Saved Steps: 1000

(P.T.O.)

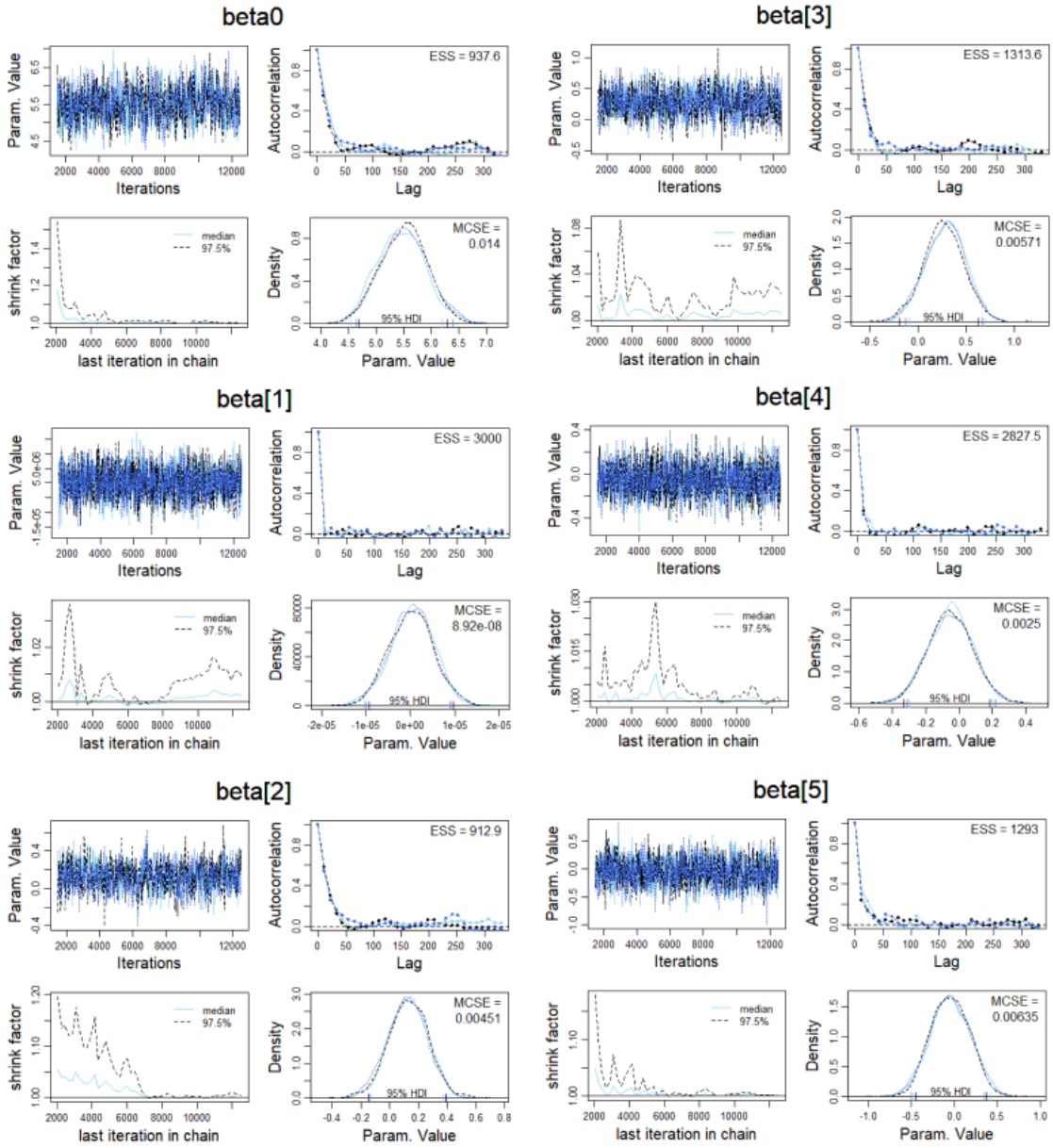

Figure 16: MCMC Diagnostic Checks for β_0, β_1, β_2, β_3, β_4, and β_5 (Refer Appendix A, D, E and F for R code)

First, we will check the representativeness of the MCMC chains. From the above output, the shrink factor for chains of intercept β_0 and coefficients β_1, β_2, β_3, β_4, and β_5, is under 1.2. Next, all the chains converge at the mean level and mix well. Also, the density distributions for almost all the parameters overlap well. Coming to the autocorrelation, it should be near to zero. However, except for β_1, there is autocorrelation present in the MCMC chains. This can be fixed by increasing the number of thinning steps.

Next, we will check the accuracy of the MCMC chains. From the above output, Estimated Sample Size (ESS) for β_0 and β_4 are acceptable at around 3000, whereas the ESS for β_0, β_2, β_3 and β_5 is relatively low at around 900 and 1300. This can be fixed by increasing the number of saved steps. The Monte Carlo Standard Error(MCSE) for all the parameters is pretty good as all the values are below 0.

So, overall, the MCMC chains for above parameters setting are acceptable in terms of accuracy but not so good in terms of representativeness. To eliminate the issues mentioned above, we will change the values of MCMC parameters, viz., Number of Thinning Steps and the Number of Saved Steps.

5.2. Updated settings

We will increase the Number of Thinning Steps from 11 to 45. This will help us decrease the autocorrelation wherever necessary. Next, we will increase the Number of Saved Steps from 1000 to 1500. This will help us obtain bigger ESS. So, the diagnostic check for MCMC chains generated for intercept β_0 and coefficients β_1, β_2, β_3, β_4, and $\beta 5$ using updated MCMC parameters is as below:

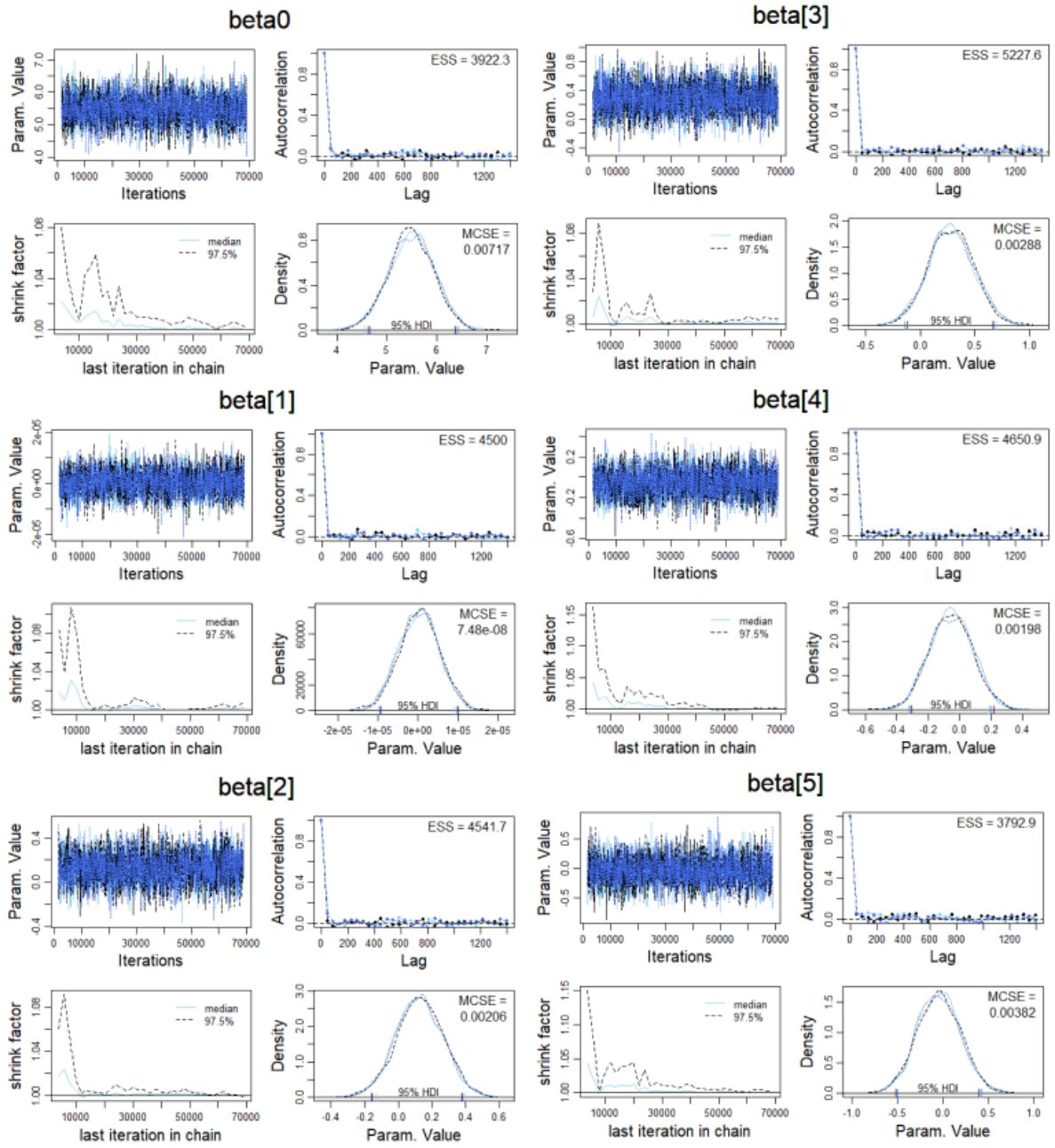

Figure 17: MCMC Diagnostic Checks for β_0, β_1, β_2, β_3, β_4, and β_5 with updated MCMC parameters (Refer Appendix A, D, E and F for R code)

From the above output, in terms of representativeness, the autocorrelation for all the parameters now seems very close to 0. It can still be brought closer to zero for β_0 and β_5, but it is acceptable. In terms, of accuracy the ESS for all the parameters is now in the acceptable range. From the above output, we can also see that by updating the MCMC parameters settings, other factors in the diagnostic check have not been affected negatively. For all the parameters, the chains converge at the mean level and mix well, the shrink factor is below 1.2, the distributions for all the parameters overlap well, and the MCSE is well below 0 for all the parameters.

So, for below MCMC parameter settings,

Adaptation Steps: 1000

Burn-in Steps: 100

Chains: 3

Thinning Steps: 45

Saved Steps: 1500

we got good results out of MCMC diagnostic checks for chains of intercept β_0 and coefficients β_1, β_2, β_3, β_4, and β_5.

For above MCMC parameter settings, the MCMC diagnostics for chains generated while estimating variance(σ^2) are as below:

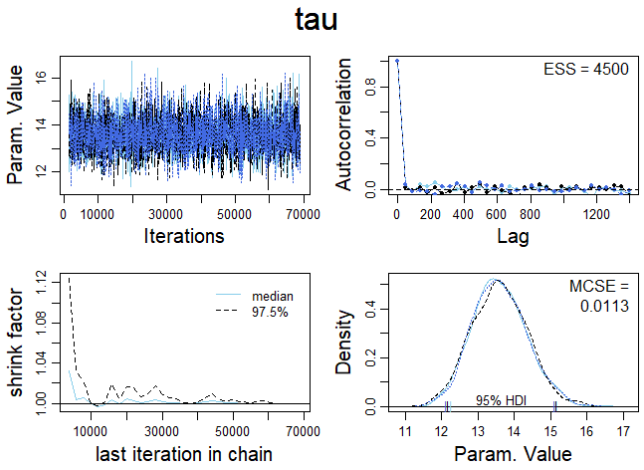

Figure 18: MCMC Diagnostic Checks for σ^2 (tau) (Refer Appendix A, D, E and F for R code)

From the above output, in terms of representativeness, the shrink factor is good as it is less than 1.2. The chains converge at mean level and are mixed well. The distribution and respective HDI limits for all the chains overlap well. Also, the autocorrelation is near to zero. In terms of accuracy, the ESS is acceptable at 4500. Also, the MCSE is well below 0.0113 which is good.

In terms of efficiency, the parallel run completed in lesser time compared to the sequential run. The multiple linear regression model specified using JAGS ran around 30% faster when run using parallel mode as compared to the sequential mode. The efficiency can be further improved by using the functionality of R packages like dclone and gputools. This has been discussed further in the Recommendations section.

6. Results

Using the best values for MCMC parameters and the best values for variance of prior distributions to represent the degree of belief, obtained previously, we run the MCMC estimation for the multiple linear regression model defined above. Here, we use the entire dataset with total of 10,000 data points. We will first obtain the Bayesian estimates for all the parameters and then obtain the Bayesian estimates of predictions for *SalePrice* using different values of *Area*, *Bedrooms*, *Bathrooms*, *CarParks* and *PropertyType*.

6.1. Bayesian Estimates for Parameters

Here, we will obtain the Bayesian estimates for variance(σ^2), intercept β_0, and the coefficients β_1, β_2, β_3, β_4, and β_5 for predictors *Area*, *Bedrooms*, *Bathrooms*, *CarParks* and *PropertyType* respectively. The prior distributions for all the parameters can be given as follows:

- σ^2 ~ gamma(0.01, 0.01)
- β_0 ~ normal(0, 4)
- β_1 ~ normal(0.0009, 0.1), standardized
- β_2 ~ normal(1, 20), standardized
- β_3 ~ normal(0, 4)
- β_4 ~ normal(1.2, 1), standardized
- β_5 ~ normal(-1.5, 0.1), standardized

As we are considering all the data, we need to change the MCMC settings. The MCMC parameters settings can be given as:

- **Adaptation Steps:** 3000
- **Burn-in Steps:** 7000
- **Chains:** 5
- **Thinning Steps:** 45
- **Saved Steps:** 1200

After running the model using above mentioned prior distributions and the MCMC parameters settings, the MCMC diagnostics for all the above parameters are as below:

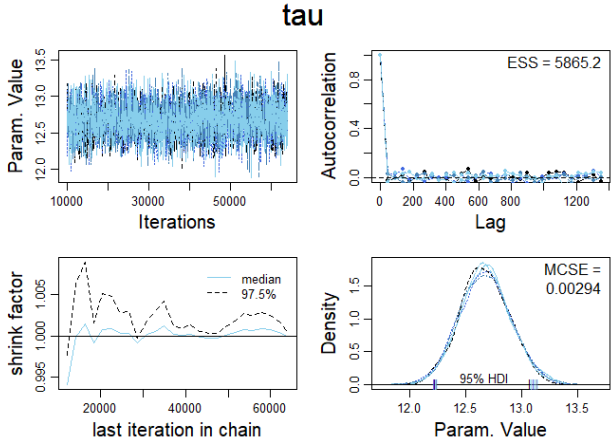

Figure 19: MCMC Diagnostic Checks for σ^2 (tau) (Refer Appendix A, D, E and F for R code)

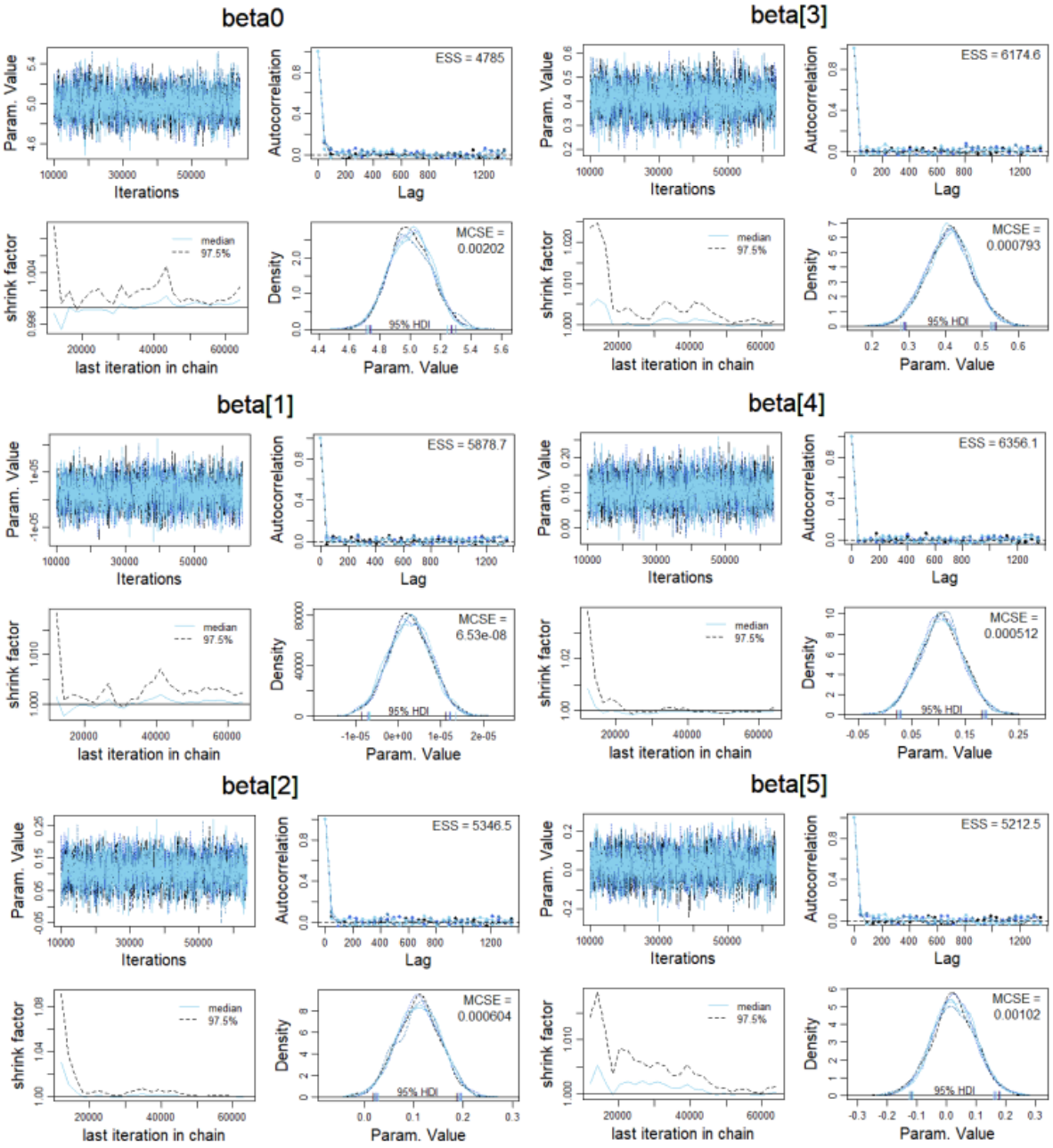

Figure 20: MCMC Diagnostic Checks for β_0, β_1, β_2, β_3, β_4, and β_5 (Refer Appendix A, D, E and F for R code)

In terms of representativeness, the MCMC chains for all the parameters have shrink factor less than 1.2. All the chains converge at respective mean level and mix well. The density curves along with respective 95% HDIs of chains for all the parameters seem to nearly overlap. Also, the autocorrelation for chains of all the parameters is very close to 0. So, in terms of representativeness, the generated MCMC chains are good.

In terms of accuracy, the MCMC chains for al the parameters have a good ESS of around 5000 to 6000. Also, the MCSE for chains of all the parameters is well below 0. So, even in terms of accuracy, the generated MCMC chains are goods.

After validating the MCMC chains for all the parameters, the respective posterior distributions for those parameters are as below:

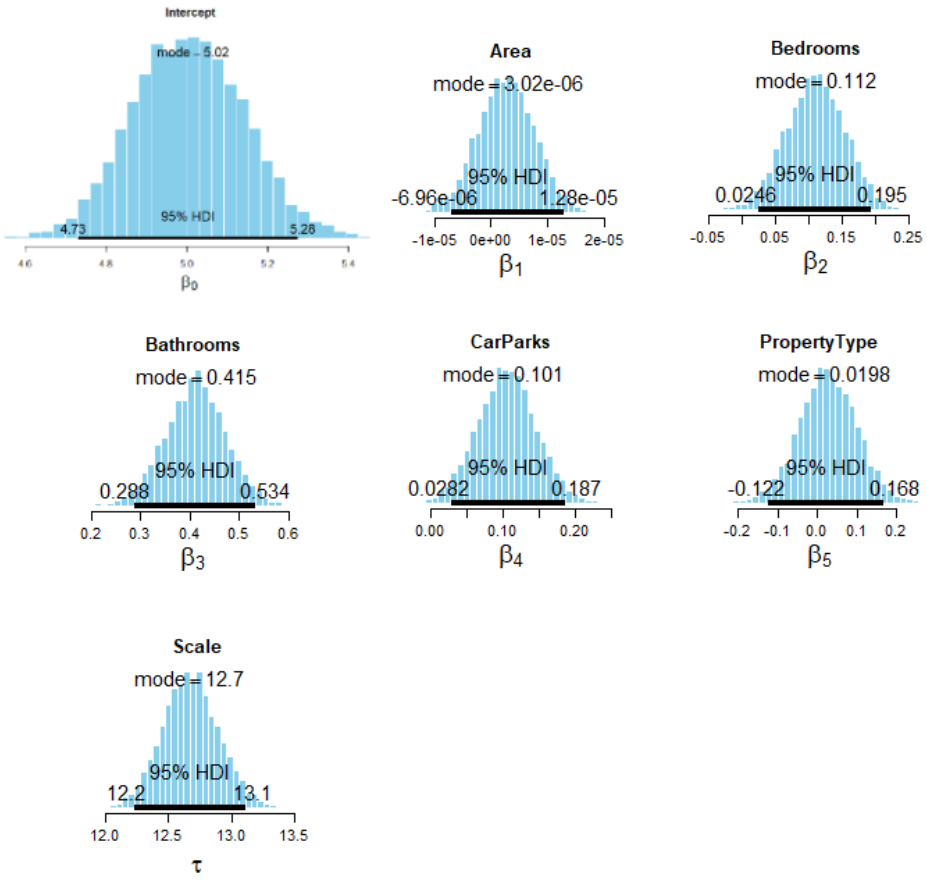

Figure 21: Posterior Distributions for β_0, β_1, β_2, β_3, β_4, β_5 and σ^2 (tau) (Refer Appendix A, D, E and C for R code)

As seen in the above output, the Bayesian estimate along with the respective HDIS for all the parameters can be summarized as below:

- σ^2 (tau) – Bayesian estimate of 12.7 with HDI of [12.2, 13.1]
- β_0 – Bayesian estimate of 5.02 with HDI of [14.73, 5.28]
- β_1 – Bayesian estimate of 0.000302 with HDI of [-0.000696, 0.00128]
- β_2 – Bayesian estimate of 0.112 with HDI of [0.0246, 0.195]
- β_3 – Bayesian estimate of 0.415 with HDI of [0.288, 0.534]
- β_4 – Bayesian estimate of 0.101 with HDI of [0.0282, 0.187]
- β_5 – Bayesian estimate of 0.0198 with HDI of [-0.122, 0.168]

The interpretation of above values can be given as below:

- **β_0** – If none of the predictor value is given, the sales price for property would be 502,000K AUD.
- **β_1** – For every m^2 increase in area of the property, the sales price would increase by 30.2 AUD.

- β_2 – For every additional bedroom, the sales price would increase by 11,200 AUD.
- β_3 – For every additional bathroom, the sales price would increase by 41,500 AUD.
- β_4 – For every additional var park, the sales price would increase by 10,100 AUD.
- β_5 – If the type of property is a unit, on an average, the sales price would be 1,980 AUD more than a house property.
- **σ^2 (tau)** – The distribution representing the predicted sales price will have a variance of 1,270,000 AUD.

6.2. Bayesian Estimates for Predictions

Now that we have estimated the parameters, we will predict the sales price for property for below values of *Area*, *Bedrooms*, *Bathrooms*, *CarParks* and *PropertyType*:

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

The predictions obtained for above values are as below:

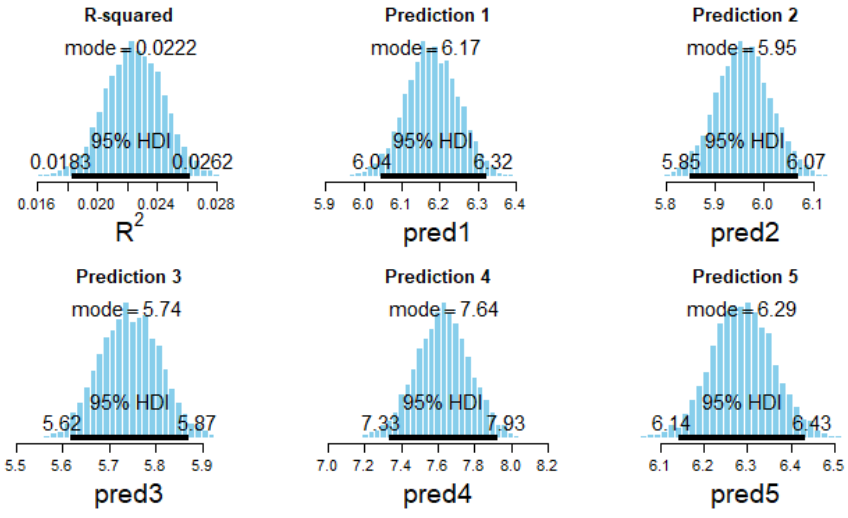

Figure 21: Posterior distributions for predictions and R-squared value (Refer Appendix A, D, E and C for R code)

The above results can be summarized as below:

- For Area = 600, Bedrooms = 2, Bathrooms = 2, CarParks = 1 and PropertyType = 1(Unit), the sales price for this property will be 617,000 AUD. Also, there is 95% chance that the price for this property would lie in between 604,000 AUD and 632,000 AUD.
- For Area = 800, Bedrooms = 3, Bathrooms = 1, CarParks = 2 and PropertyType = 0(House), the sales price for this property will be 595,000 AUD. Also, there is 95% chance that the price for this property would lie in between 585,000 AUD and 607,000 AUD.
- For Area = 1500, Bedrooms = 2, Bathrooms = 1, CarParks = 1 and PropertyType = 0(House), the sales price for this property will be 574,000 AUD. Also, there is 95% chance that the price for this property would lie in between 562,000 AUD and 587,000 AUD.
- For Area = 2500, Bedrooms = 5, Bathrooms = 4, CarParks = 4 and PropertyType = 0(House), the sales price for this property will be 764,000 AUD. Also, there is 95% chance that the price for this property would lie in between 733,000 AUD and 793,000 AUD.
- For Area = 250, Bedrooms = 3, Bathrooms = 2, CarParks = 1 and PropertyType = 1(Unit), the sales price for this property will be 629,000 AUD. Also, there is 95% chance that the price for this property would lie in between 614,000 AUD and 643,000 AUD.

The R-squared value for this multiple linear regression model is 0.022. This means that the model covers just 2% of the variation in the data. However, in Bayesian version of linear regression, R-squared value is not a reliable measure. This is because the model takes into consideration even the expert knowledge in form of prior information.

Lastly, we will now plot a kernel density estimate for observed sales price values and the sales price values predicted by the model.

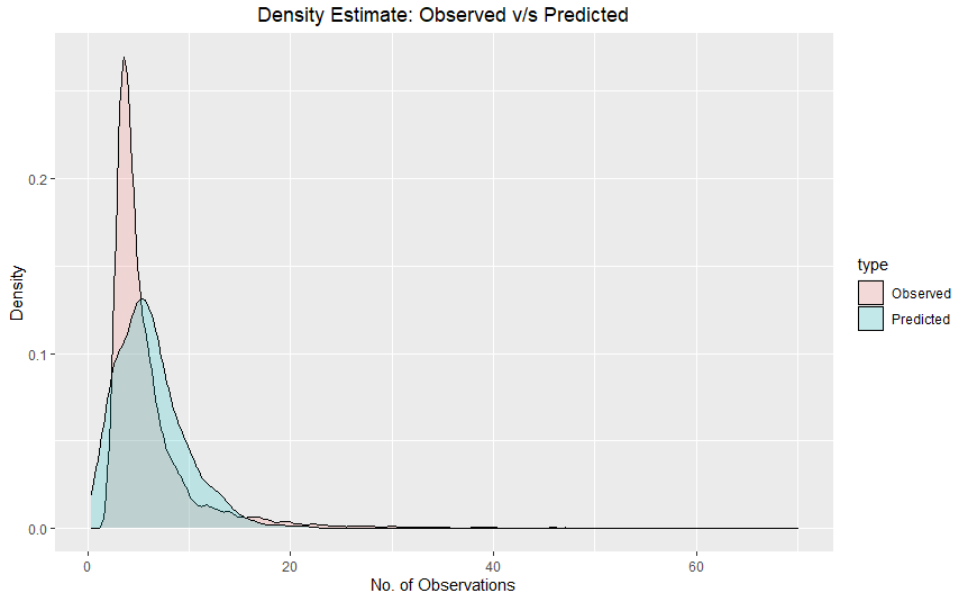

Figure 22: Kernel Density Estimate: Observed v/s Predicted (Refer Appendix G for R code)

From the above plot, we can see that the model is not able to cover all the variations in data. This supports the poor R-squared value obtained. So, from the comparison density estimate and the R-squared value, we know that the goodness of fit for this model is poor. However, this can be improved by multiple way. Few of such methods are discussed in the Recommendations section.

7. Conclusion

In this analysis, using Bayesian approach, the multiple linear regression model obtained was as below:

SalePrice = 5.02 + 0.000302*Area + 0.112*Bedrooms + 0.415*Bathrooms + 0.101*CarParks + 0.0198*PropertyType

8. Recommendations

8.1. Efficiency

The efficiency of MCMC simulations can be improved in following ways:

- By using the parallel functionality provided by the dclone R package. The function that can be used in jags.parfit().
- By utilizing the GPUs. This can be done by writing custom functions to run the MCMC simulation and utilizing the GPUs using R packages like gputools and cudaBayesreg.

8.2. Goodness of Fit

The goodness of fit for the model can be improved in following ways:

- Considering a bigger sample of data.
- Considering more parameters(predictors) to describe the dependent variable.
- Considering different distribution for variables, for instance, using exponential distribution instead of gamma distribution, or, using uniform distribution to specify non-informative prior.

9. References

- Prof. Haydar Demirhan, MATH2269 Module 5 notes – Just Another Gibbs Sampler - JAGS, School of Science, RMIT university.
- Prof. Haydar Demirhan, MATH2269 Module 6 notes – Bayesian Linear Regression, School of Science, RMIT university.
- Patric Zhao(August 4, 2014), Accelerate R Applications with CUDA, https://developer.nvidia.com/blog/accelerate-r-applications-cuda/#:~:text=The%20GPU%2DAccelerated%20R%20Software,including%20C%2C%20C%2B%2B%20and%20Fortran, accessed on 16-Sep-20.
- Peter Solymos (2010). dclone: Data Cloning in R. The R Journal 2(2), 29-37. https://journal.r-project.org/.
- R codes adopted from DBDA2Eprograms.zip by Kruschke, J. K. (2015). Doing Bayesian Data Analysis, Second Edition: A Tutorial with R, JAGS, and Stan. Academic Press / Elsevier.

(P.T.O.)

10. Appendix

A1. Model String: Data

```
data {
  y_sd <- sd(y)
  for (i in 1:Ntotal) {
    zy[i] <- y[i] / y_sd
  }
  for (j in 1:Nx) {
    x_sd[j] <- sd(x[, j])
    for (i in 1:Ntotal) {
      zx[i, j] <- x[i, j] / x_sd[j]
    }
  }
}
```

A2. Model String: Model

```
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(M0, 1 / (S0^2))
  zbeta[1] ~ dnorm(M1 / x_sd[1], 1 / (S1/x_sd[1]^2))
  zbeta[2] ~ dnorm(M2 / x_sd[2], 1 / (S2/x_sd[2]^2))
  zbeta[3] ~ dnorm(M3, 1/( S3^2))
  zbeta[4] ~ dnorm(M4 / x_sd[4], 1 / (S4/x_sd[4]^2))
  zbeta[5] ~ dnorm(M5 / x_sd[5], 1 / (S5/x_sd[5]^2))
  zVar ~ dgamma(α, β)

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

  # Transform to original scale:
  beta[1:Nx] <- (zbeta[1:Nx] / x_sd[1:Nx]) * y_sd
  beta0 <- zbeta0 * y_sd
  tau <- zVar * (y_sd)^2

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

  # Compute predictions at every step of the MCMC
  for (i in 1:Nx){
    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]
  }

}
```

(P.T.O.)

B. R code for function summary_MCMC()

```
summary_MCMC = function(codaSamples, compVal = 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

  return(summaryInfo)
}
```

C. R code for function plot_MCMC()

```
plot_MCMC = function(codaSamples, data, xName = "x", yName = "y", showCurve
= FALSE,
                     compVal = NULL, saveName = NULL, saveType = "jpg") {

  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]"]
  pred2 = mcmcMat[, "pred[2]"]
  pred3 = mcmcMat[, "pred[3]"]
  pred4 = mcmcMat[, "pred[4]"]
  pred5 = mcmcMat[, "pred[5]"]
```

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

  # Compute R-squared value

  Rsq = zbeta %*% matrix(cor(y, x), ncol=1)

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

  # Marginal histograms

  plot_hist = 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 = plot_hist(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 = plot_hist(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 = plot_hist(panelCount, saveName = paste0(saveName,
"PostMarg"))
  histInfo = plotPost(tau, cex.lab = 1.75, showCurve = showCurve,
                      xlab = bquote(tau), main = paste("Scale"))

  panelCount = plot_hist(panelCount, saveName = paste0(saveName,
"PostMarg"))
  histInfo = plotPost(Rsq, cex.lab = 1.75, showCurve = showCurve,
                      xlab = bquote(R^2), main = paste("R-squared"))

  panelCount = plot_hist(panelCount, saveName = paste0(saveName,
"PostMarg"))
  histInfo = plotPost(pred1, cex.lab = 1.75, showCurve = showCurve,
                      xlab = "pred1", main = "Prediction 1")

  panelCount = plot_hist(panelCount, saveName = paste0(saveName,
"PostMarg"))
  histInfo = plotPost(pred2, cex.lab = 1.75, showCurve = showCurve,
                      xlab = "pred2", main = "Prediction 2")

  panelCount = plot_hist(panelCount,  saveName = paste0(saveName,
"PostMarg"))
  histInfo = plotPost(pred3, cex.lab = 1.75, showCurve = showCurve,
                      xlab = "pred3", main = "Prediction 3")

  panelCount = plot_hist(panelCount,  saveName = paste0(saveName,
"PostMarg"))
  histInfo = plotPost(pred4, cex.lab = 1.75 ,showCurve = showCurve,
                      xlab="pred4", main = "Prediction 4")

  panelCount = plot_hist(panelCount, finished = TRUE, saveName =
paste0(saveName, "PostMarg"))
  histInfo = plotPost(pred5, cex.lab = 1.75, showCurve = showCurve,
                      xlab = "pred5", main = "Prediction 5")

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

  # Standardized scale:

  panelCount = 1
  panelCount = plot_hist(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 = plot_hist(panelCount, saveName = paste0(saveName,
"PostMargZ"))
    histInfo = plotPost(zbeta[, bIdx], cex.lab = 1.75, showCurve =
showCurve,
                        xlab = bquote(z * beta[.(bIdx)]), main =
xName[bIdx])
  }

  panelCount = plot_hist(panelCount, saveName = paste0(saveName,
"PostMargZ"))
  histInfo = plotPost(zVar, cex.lab = 1.75, showCurve = showCurve,
                      xlab = bquote(z * tau), main = paste("Scale"))

  panelCount = plot_hist(panelCount, saveName = paste0(saveName,
"PostMargZ"))
  histInfo = plotPost(Rsq, cex.lab = 1.75, showCurve = showCurve,
                      xlab=bquote(R^2), main = paste("R-squared"))

  panelCount = plot_hist(panelCount, finished = TRUE, saveName =
paste0(saveName, "PostMargZ"))

}
```

D. R code for preparing data

```
# Importing data
sales_dat <- read.csv("Assignment2PropertyPrices.csv")

#extract sample
set.seed(999)
sales_dat_sample <- sample_n(sales_dat, 1000)

#separate dependent and independent variables
y = sales_dat[, "SalePrice"]
x = as.matrix(sales_dat[, c("Area", "Bedrooms", "Bathrooms", "CarParks",
"PropertyType")])

#specify input values for dependent variables
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)

#wrap the data in list for JAGS
dataList <- list(
  x = x ,
  y = y ,
  xPred = xPred ,
  Nx = dim(x)[2] ,
  Ntotal = dim(x)[1]
)
```

(P.T.O.)

```
# specify initial values for MCMC model
initsList <- list(
  zbeta0 = 100,
  zbeta = c(100, 1, 1, 1, 1),
  Var = 1000
)
```

E. R code for specifying MCMC parameters and running MCMC simulation using JAGS

```
# Specify MCMC settings
adaptSteps = 3000
burnInSteps = 7000
nChains = 5
thinSteps = 45
numSavedSteps = 1200

# Run the model
runJagsOut <- run.jags(method = "parallel",
                       model = modelString,
                       monitor = c("beta0", "beta", "tau", "zbeta0",
"zbeta", "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)
```

F. R code to find diagnostic check for specific parameter

```
diagMCMC(codaSamples, parName = "parameter_name")
```

G. R code for predictive check

```
#compute required values
coefficients <- summary[c(2:7), 3]
Variance <- summary[8, 3]

meanGamma <- as.matrix(cbind(rep(1, nrow(x)), x)) %*%
as.vector(coefficients)
randomData <- rgamma(n = 500,
                     shape = meanGamma^2 / Variance,
                     rate = meanGamma / Variance)

predicted <- data.frame(elapsed = randomData)
observed <- data.frame(elapsed = y)

predicted$type <- "Predicted"
observed$type <- "Observed"
dataPred <- rbind(predicted, observed)
```

```
# Display the density plot of observed data and predicted
ggplot(dataPred,
       aes(elapsed, fill = type)) +
  geom_density(alpha = 0.2) +
  xlab("No. of Observations") +
  ylab("Density") +
  ggtitle("Density Estimate: Observed v/s Predicted") +
  theme(plot.title = element_text(hjust = 0.5))
```

1. R code for Kernel Density Estimate plot for *SalePrice*

```
# Kernel density estimation
plot(
  kde(sales_dat$SalePrice),
  main = "Kernel Density Plot for Sales Price",
  xlab = "SalePrice",
  ylab = "Density"
)
```

2. R code for Scatter plots to represent the relation of dependent variable with independent variables

```
# Scatter plots
plot_1 <- ggplot(sales_dat, aes(x = Area, y = SalePrice)) +
  geom_point() +
  xlab("Area") +
  ylab("Sales Price")

plot_2 <- ggplot(sales_dat, aes(x = Bedrooms, y = SalePrice)) +
  geom_point() +
  xlab("Bedrooms") +
  ylab("Sales Price")


plot_3 <- ggplot(sales_dat, aes(x = Bathrooms, y = SalePrice)) +
  geom_point() +
  xlab("Bathrooms") +
  ylab("Sales Price")


plot_4 <- ggplot(sales_dat, aes(x = CarParks, y = SalePrice)) +
  geom_point() +
  xlab("CarParks") +
  ylab("Sales Price")

plot_5 <- ggplot(sales_dat, aes(x = PropertyType, y = SalePrice)) +
  geom_point() +
  xlab("PropertyType") +
  ylab("Sales Price")

fig <- ggarrange(plot_1,
                 plot_2,
                 plot_3,
                 plot_4,
                 plot_5,
                 nrow = 2,
                 ncol = 3,)
fig
```

3. R code for Correlation matrix for independent variables

```
#correlation between dependent variables
cor(sales_dat[, c("Area", "Bedrooms", "Bathrooms", "CarParks",
"PropertyType")]) %>% round(3)
```

(E.O.F.)