PROPERTY PRICE PREDICTIONS

A Bayesian Regression Analysis

Bhargav Rele
s3761977
Submitted on: 22.9.2020
Master of Analytics| RMIT, Melbourne, Victoria, Australia

Table of Contents

Introduction

Property prices are often key indicators of the economic status and performance for a given geographical location. Predicting property prices often involves classical regression techniques that have proven to be effective. However, such techniques are based on the assumption that population parameters underlying the sample distribution, are constant and unknown. In this report however, we shall relax this assumption by; simulating the distributions of population parameters, that affect the measure of central tendency and the variation in property prices. In particular, we shall use Bayesian multi-regression techniques to determine a posterior distribution for property prices, that we can then use to make inferences and predictions, based on property features.

Descriptive Outlook

The property prices dataset being utilised within this analysis is a fabricated dataset that represents the real property prices from multiple property datasets. This is done to comply with the data security issues associated with the real data. The dataset itself consists of 10,000 property prices ($SalePrice_i$) in addition to various property features. The features themselves are; area, number of bedrooms, number of bathrooms, car park space available and property type. Area and SalePrice are quantitative features while the rest are qualitative.

a. Bivariate Distributions

In order to carry out a regression analysis, we need to first understand the behaviour of the features within our dataset, with respect to sale price. In other words, we try and visualise the extent of the relationship between the dependant variable and each independent variable. The bivariate relationships are exhibited in figure 1 below.

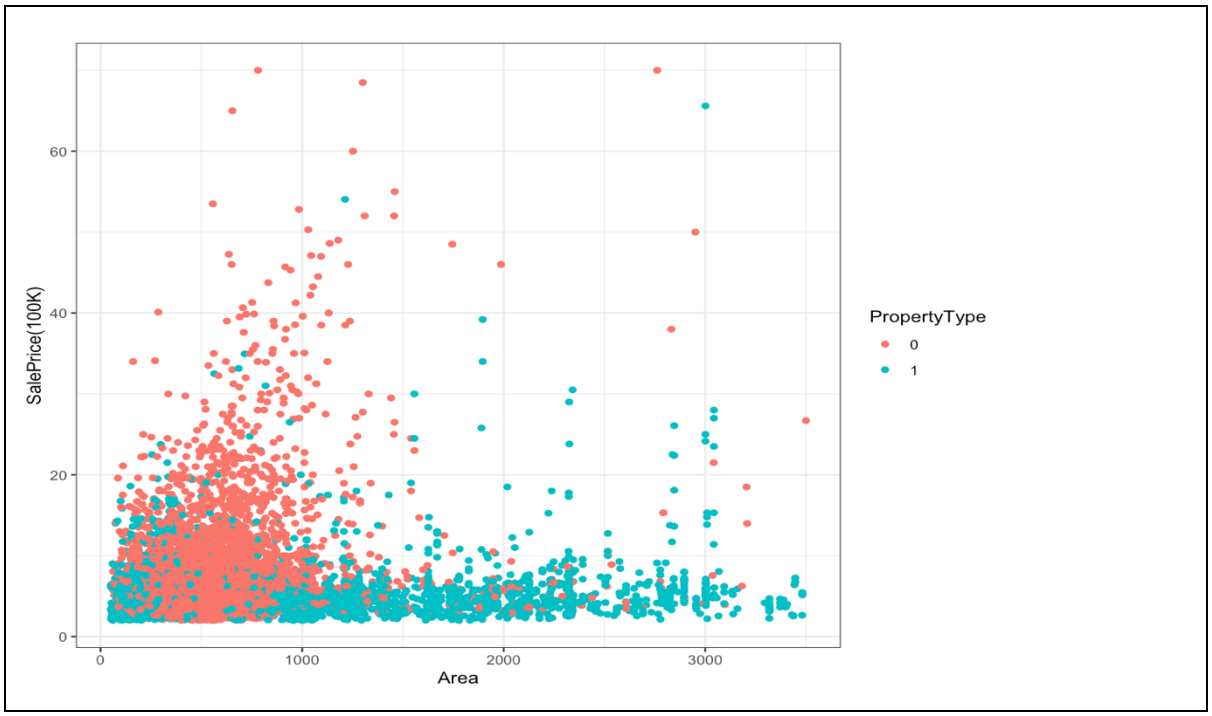


Figure 1.

The following can be said about the bivariate relationships between Sale Price and;

1) Area: On observing the scatterplot for Area and SalePrice, it can be visually deduced that there is a positive relationship between the variables, for properties that are houses (PropertyType=0). An increase in area by $1m^2$ increase the sale price of a house by a larger proportion. The same cannot be said for units (PropertyType=1). In fact, there seems to be no obvious relationship between the area and sale price for properties that are units.

2) Bedroom: The boxplots for bedrooms suggest that we have a higher median sale price for properties with a higher number of bedrooms.

3) Bathroom: The boxplot for bathrooms suggest that properties with a higher number of bathrooms, sell at a higher median price. However,

4) CarPark: The boxplot for car parks suggest that, properties that come with a higher number of car parks, sell at a higher median price. However, this isn't apparent because; tenants are less likely to have 4 or more cars. The appeal behind paying for four parking spaces isn't much, perhaps.

5) PropertyType: The boxplot for property-types suggest that units and houses sell for approximately similar median price.

b. Correlation Matrix

Prior to running a regression, we need to ensure that our independent variables are not strongly correlated with one and other. Any strong correlations amongst the independent variables may result in endogeneity and problems of multicollinearity [1]. The correlation matrix for the variables can be found in figure 2 below.

	SalePrice.100K.	Area	Bedrooms	Bathrooms	CarParks	PropertyType
SalePrice.100K.	1.00000000	0.07573894	0.2192343	0.30603104	0.21027979	-0.1199149
Area	0.07573894	1.00000000	-0.2698876	-0.08733214	-0.09626886	0.3201946
Bedrooms	0.21923427	-0.26988756	1.0000000	0.53815384	0.43513783	-0.5601464
Bathrooms	0.30603104	-0.08733214	0.5381538	1.00000000	0.36578982	-0.2896599
CarParks	0.21027979	-0.09626886	0.4351378	0.36578982	1.00000000	-0.3600459
PropertyType	-0.11991485	0.32019461	-0.5601464	-0.28965989	-0.36004594	1.0000000

Figure 2.

It is evident that the independent variables of our model have moderate and weak correlations amongst themselves. Given that the strength of the relationships is not strong, we can choose to include all the independent variables in our regression model. Furthermore, the correlation between the independent variables and the dependent variable isn't too high. This is perhaps because; four out of five independent variables within our model are qualitative figures (nominal and ordinal features).

c. Summary Statistics

We can then proceed with describing the features of the dataset using summary statistics, such as; mean, median, mode and value counts for each of feature. Figure 3 below, indicates that most of the property types are houses, and we have a median sale price of $4.5 (in $100K), i.e., $450,000. (Note: we use the median as a measure for central tendency given that we have a positive-skewed distribution).

SalePrice(100K)	Area	Bedrooms	Bathrooms	CarParks	PropertyType
Min. : 2.000	Min. : 50.0	1: 594	1:4499	2 :4814	0:6838
1st Qu.: 3.500	1st Qu.: 353.0	2:2517	2:4976	1 :4246	1:3162
Median : 4.500	Median : 568.0	3:4594	3: 463	3 : 385	
Mean : 6.094	Mean : 690.2	4:2030	4: 62	4 : 279	
3rd Qu.: 6.550	3rd Qu.: 752.0	5: 237		0 : 175	
Max. :70.000	Max. :3500.0	6: 22		5 : 48	
		7: 6		(Other): 53	

Figure 3.

Furthermore, we primarily have houses with 2 to 4 bedrooms, less than 2 bathrooms, and less than 2 car park facilities.

d. Histogram and density plot for Sale Price (Y_i)

In order to model the data using Bayesian Regression techniques, we need to specify the distribution of Sale Price, in order to identify the relevant population parameters (that we assume follow their own distributions; on which we shall make inferences). In order to

identify the distribution that appropriately fits the Sale Price data that we have, we construct a simple histogram and density plot; that can be found in figure 4 below.

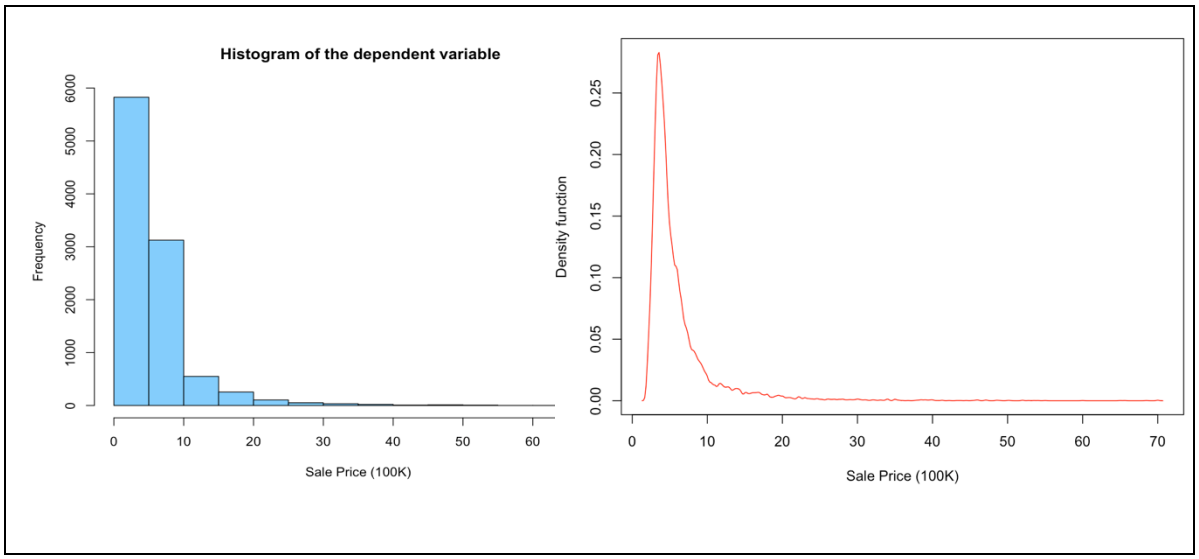


Figure 4.

The histogram and density plot indicate that the distribution of sale price is positively skewed. That is, despite having a high concentration of low-priced properties, the distribution is pulled to the right by high-priced properties. Given that the distribution is not symmetric, a normal distribution would not be able to appropriately represent the sale prices. Instead, we use a distribution that is positively skewed; the gamma distribution. The gamma distribution has two parameters; a and b, that determine the location (centre) and concentration of the distribution [2]. However, we carry out reparameterization in order to represent these parameters using μ_i and σ^2 [3]. Furthermore, given that sale prices for property cannot be negative, it would be inappropriate to use a normal distribution, given that a normal distribution can take up values between $-\infty$ and ∞. On the other hand, gamma distributions are purely positive distributions that cannot take up negative values. Hence, we have:

$$\text{SalePrice}_i \sim \text{gamma}(a,b) \text{ and,}$$
$$\text{Through reparameterization we have; SalePrice}_i \sim \text{gamma}(\mu^2_i / \sigma^2, \mu_i / \sigma^2)$$

For notational purposes, we shall change the variable names. Hence, from here on SalePrice_i will be referred to Y_i and, Area_i, Bedroom_i, Bathroom_i, CarPark_i and PropertyType_i will be referred to as X_1, X_2, X_3, X_4 and X_5 respectively. Hence, our dependent variable's distribution can be noted as: $Y_i \sim \text{gamma}(\mu^2_i / \sigma^2, \mu_i / \sigma^2)$.

Model Creation

The syntax and inner workings of the algorithm are explained in detail by Kruschke in his book; "Doing Bayesian Data Analysis" [4]. The algorithm utilises JAGS and R in order to run Markov Chain Monte Carlo simulations to come up with posterior distributions, when there is mathematical-intractability between the prior distributions and the likelihood function of the dependent variable. A few minor changes are made to ensure we can run simulations on our

data. In particular, we specify; 1) the prior distributions for all parameters, using expert knowledge and our belief in this knowledge, 2) the initial values for each variable within the algorithm and, 3) the specifics for each simulation.

Given that we have 10,000 property prices we are trying to regress, the capabilities of R in being able to run these simulations is hindered by computational load and elapsed time. Hence, we are limited when tuning the specifics of the simulations. Nonetheless we try to run the simulations by tuning the specifics based on their MCMC diagnostics. We are also able to change the initial values we would want the algorithm to take up in each run. Finally, the location of the prior distribution for each variable is specified by experts. However, the variance is selected based on the belief we place in the expert's knowledge. Therefore, the selection of; simulation specifics, initial values and prior distribution variances are arbitrary and, can make a huge difference is enabling us to come up with an appropriate posterior distribution that fits the data well. This paired in with the problem of computational load and elapsed time for each run, limits us in being able to find the *best* posterior distribution. Instead, we try to find the *most suitable and achievable* posterior distribution that gives us good MCMC diagnostics.

[Please note, that the specifications and results that are explained in the coming sections, is for the most suitable and achievable posterior distribution, i.e., the model that gave us the best results. The appendix section can be viewed for the specifics of each simulation that enabled us to narrow our specifications to those used in the final model.]

Model Specifications

Before specifying the model details, we shall specify the statistical logic that has gone into making these models.

a. Prior Knowledge for Independent variables

The prior distribution for each variable and their parameter specifications can be found in figure 5 below.

Variable Name	Mean(AUD)	Mean(1000 AUD)	Degree of Belief	Variance Factor
$Area_i$ (X_1)	90	0.09	Very strong	5
$Bedrooms_i$ (X_2)	100,000	100	Weak	1
$Bathrooms_i$ (X_3)	*No expert knowledge*	*No expert knowledge*	*No expert knowledge*	*No expert knowledge*
$CarParks_i$ (X_4)	120,000	120	Strong	4
$PropertyType_i$ (X_5)	-150,000	-150	Very strong	5

Figure 5.

[Note: the variance factor is specified based on degree of belief as the variance of the distributions will be scaled (more on this later), by the standard deviation of the variable itself. Hence, we have a high numerator to represent a stronger belief in expert knowledge and a lower numerator to represent a weaker belief in expert knowledge.]

b. Prior Distributions for all Variables

We specify Y_i using the following regression equation:

$$Y_i = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \beta_4 X_4 + \beta_5 X_5 + e_i$$

Hence, the mean for Y_i is simply the expectancy of Y_i and is given by:

$$E(Y_i) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \beta_4 X_4 + \beta_5 X_5 + 0$$
$$\Rightarrow \quad \mu_i = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \beta_4 X_4 + \beta_5 X_5$$

Therefore, the mean for Y_i is specified in the equation above. We assume errors are independently and identically distributed with a mean of 0 and a variance of σ^2. Hence, the expected value of errors is assumed to be 0, given that we are trying to find the line of best fit that minimises this value. The variance of Y_i represents the variability of the line of best fit. Variation that is not captured by the line of best fit, is captured by the error term. Hence, any unexpected variation of Y_i can be identified as variation in e_i. Therefore, the variance of Y_i is simply σ^2. The prior distributions for the parameters μ_i and σ^2 can then be specified.

As specified earlier, the dependent variable Y_i is distributed as a gamma distribution; where mean is given as μ^2_i / σ^2 and variance is given as μ_i / σ^2 (in accordance with the reparameterization mentioned earlier). Hence, we have **Y_i ~ gamma(μ^2_i / σ^2, μ_i / σ^2).**

The distribution for σ^2 is assumed to be a gamma distribution. This is because σ^2 can only take up values between 0 and infinity. Hence, we would require a positively skewed distribution that takes up values between 0 and infinity, to represent the prior distribution of σ^2. Therefore, **σ^2 ~ gamma(A, B).**

For μ_i we have several prior distributions. Given that μ_i is a multiple-regression equation, each β follows its own prior distribution. Therefore, we don't have one prior distribution for μ_i, in fact we have six prior distributions; one for each β. We assume each β approximately follows a normal distribution with a mean of M_k and a variance of S_k, where k= 0..5. Formally, **β_k ~ normal(M_k, S_k)** where k = 0,1,2,3,4,5.

The diagram provided in figure 6 below, visualises the aforementioned prior distributions.

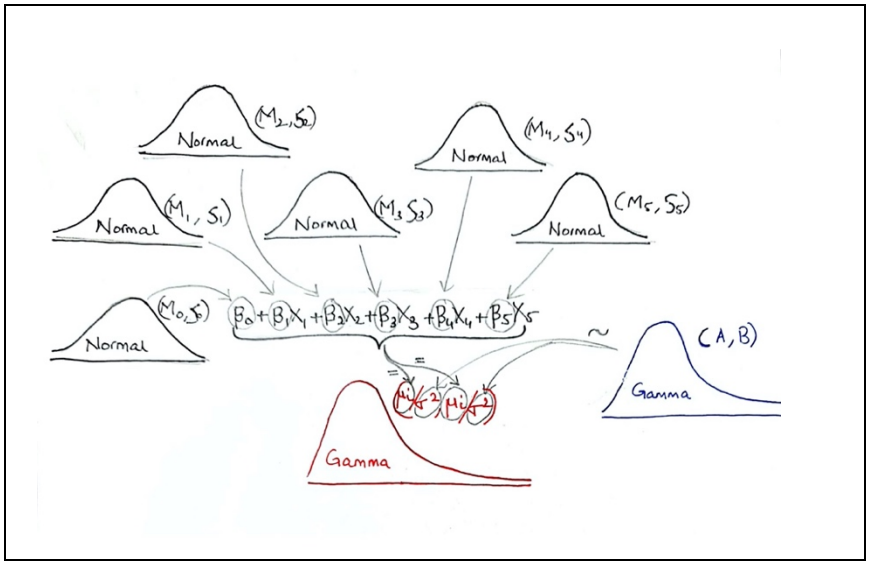


Figure 6.

By utilising the values provided in subsection 'a' earlier, we can specify the mean and variance of each prior distribution as follows:

$$Y_i \sim \text{gamma}(\mu^2_i / \sigma^2, \mu_i / \sigma^2)$$
$$\sigma^2 \sim \text{gamma}(0.01, 0.01)$$
$$\beta_0 \sim \text{normal}(0, 1/4)$$
$$\beta_1 \sim \text{normal}(0.09, 1/5)$$
$$\beta_2 \sim \text{normal}(100, 1/0.1)$$
$$\beta_3 \sim \text{normal}(0, 1/4)$$
$$\beta_4 \sim \text{normal}(120, 1/2.5)$$
$$\beta_5 \sim \text{normal}(150, 1/5)$$

Within the algorithm we divide each of these mean and variance figures by the standard deviation of the respective variables. This is done for the purpose of scaling our variables given that they are in separate units. While analysing the results, the figures will represent the re-scaled values however. The re-scaled values are found by multiplying the scaled figures with the standard deviation of Y. This is done to achieve results that can be interpreted in the original units that had been provided to us. Furthermore, while specifying the normal distributions, we set the variance of each prior normal distribution as "1/Variance". This is done in order to fall in line with the instructions provided by Kruschke [4], on how to use his JAGS algorithm.

c. Simulation Specifications

The simulation specifications for each model can be found in figure 12., provided in the appendix section. The 'Burn in steps', 'Thinning steps' and the 'Number of saved steps' were adjusted repeatedly until an ideal model was found. Furthermore, we first ran models without specifying initials. The results of the best model without initials, were used to determine the initials to be used in our final model. That is, the 'zbeta' values and variance values from the results of previous models were used as initials in the final model, with a greater number of thinning steps, in order to reduce the time taken by the simulation to complete. Figure 7

provides the values used for the simulation specifications, for our final model (the model that delivers us the best MCMC results, from all tentative models).

Argument	Value
Adaptation steps	1,500
Burn in steps	3,000
Number of chains	3
Thinning steps	15
Number of saved steps	10,000
Number of iterations	(Number of saved steps x Thinning steps) / Number of chains
Initials List	zbeta0 = 0.951 zbeta[k] = c(0.0101, 0.0276, 0.0479, 0.0171, 0.00192) var= 12.6

Figure 7.

Model Results

To evaluate the results of our model, we shall take a look at the Markov Chain Monte Carlo diagnostics, beta values for independent variables (and their significance), posterior distributions and the ability for the model to fit the data's likelihood function. Followed by which we shall predict the sale price for properties based on several independent variable values.

a. Markov Chain Monte Carlo Diagnostics

Ideally, we would want a model that has the following diagnostic results for each population distribution (or for each population parameter) [5]:

a.1 Chain plots

After the burn in period, we expect the chains to converge and overlap each other, to be representative of the posterior distribution. We ideally do not want chains to linger at a particular value or, be isolated from others. The number of saved steps and the burn-in period in the simulations specifications is repeatedly adjusted until this is achieved.

a.2 Density plots

We ideally want to see all density plots and HDI intervals overlapping for the chains to be considered representative. The number of saved steps and the number of chains within the simulation specifications, is repeatedly adjusted until a desirable density plot is achieved.

a.3 Shrink factor

As a measure of representativeness, we want the shrink factor to be 1.0 or at least below 1.2. A value above this threshold implies a stuck or orphaned chain (inadequate convergence). The number of saved steps is repeatedly adjusted until this is achieved.

a.4 Autocorrelation plot

If the chains within the plot for autocorrelations gradually descend to an autocorrelation value of 0, this is indicative of existent autocorrelations. Ideally, we would want this descent to be immediate; implying that successive steps are not influencing each other. The number of thinning steps in the simulations specifications are repeatedly adjusted until this is achieved.

a.5 Effective sample size

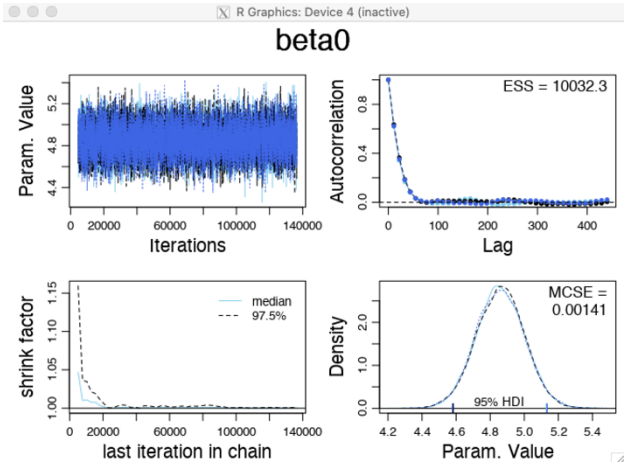
The ESS is a measure of the amount of independent information in autocorrelated chains. Hence, we would want the ESS to be a high number.

a.6 Monte Carlo Standard Error

The MCSE is a measure of the accuracy and stability of the chain. Hence, a low MCSE is ideal given that it is an error measure. However, it should be noted that an MCSE value can be high and nonetheless acceptable if it is close to the actual values of the parameter (given the units of the parameter).

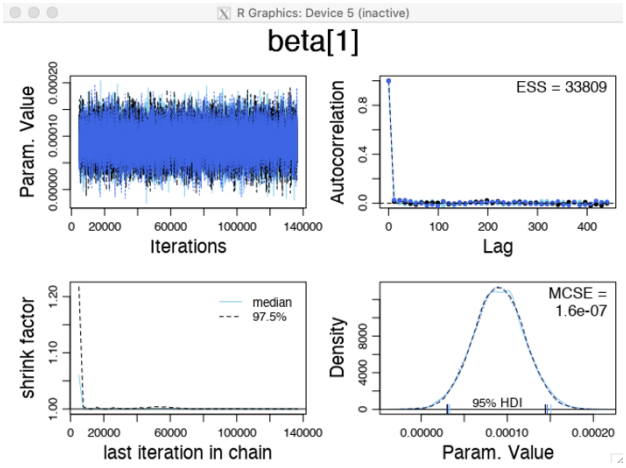
Figure 8 summarises the above diagnostics for each parameter's distribution, for the model we have selected:

β_0



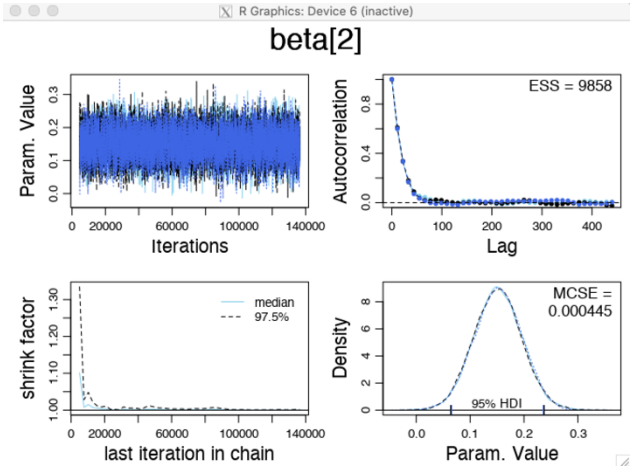
The MCMC diagnostics for β_0 are ideal given that; 1) the chain plots are overlapping and deviate around a mean value, 2) the density plots for all chains, and their HDI's are overlapping, 3) the shrink factor (median) is below the threshold of 1.2, 4) all chains immediately descend to a low autocorrelation value (thus suggesting we have no autocorrelations), 5) effective sample size is high at 10,032.3 and, 6) we have a low Markov Chain Monte Carlo standard error of 0.00141.

β_1



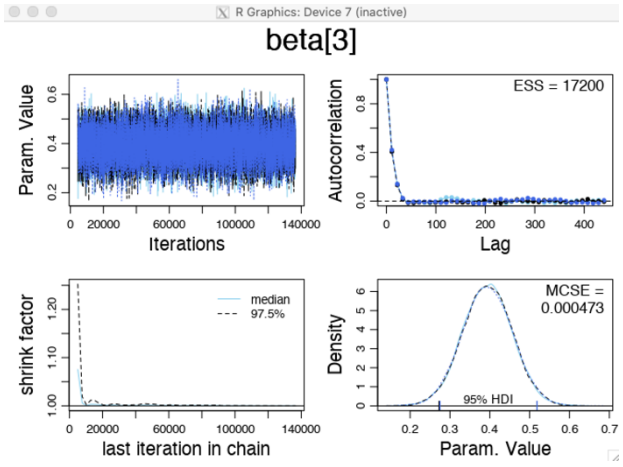
The MCMC diagnostics for β_1 are ideal given that; 1) the chain plots are overlapping and deviate around a mean value, 2) the density plots for all chains, and their HDI's are overlapping, 3) the shrink factor (median) is below the threshold of 1.2, 4) all chains immediately descend to a low autocorrelation value (thus suggesting we have no autocorrelations), 5) effective sample size is high at 33,809 and, 6) we have a low Markov Chain Monte Carlo standard error of 0.00000016.

β_2



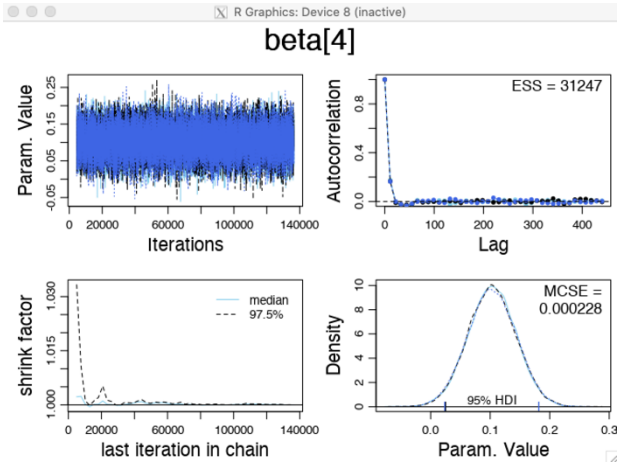
The MCMC diagnostics for β_2 are ideal given that; 1) the chain plots are overlapping and deviate around a mean value, 2) the density plots for all chains, and their HDI's are overlapping, 3) the shrink factor (median) is below the threshold of 1.2, 4) all chains immediately descend to a low autocorrelation value (thus suggesting we have no autocorrelations), 5) effective sample size is high at 9858 and, 6) we have a low Markov Chain Monte Carlo standard error of 0.000445.

β_3



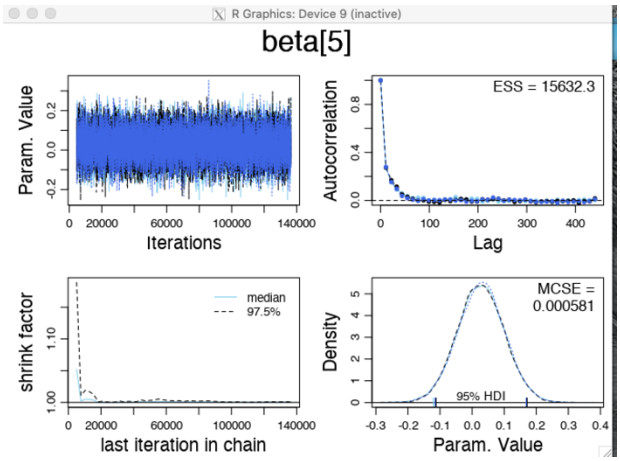
The MCMC diagnostics for β_3 are ideal given that; 1) the chain plots are overlapping and deviate around a mean value, 2) the density plots for all chains, and their HDI's are overlapping, 3) the shrink factor (median) is below the threshold of 1.2, 4) all chains immediately descend to a low autocorrelation value (thus suggesting we have no autocorrelations), 5) effective sample size is high at 17,200 and, 6) we have a low Markov Chain Monte Carlo standard error of 0.000473.

β_4



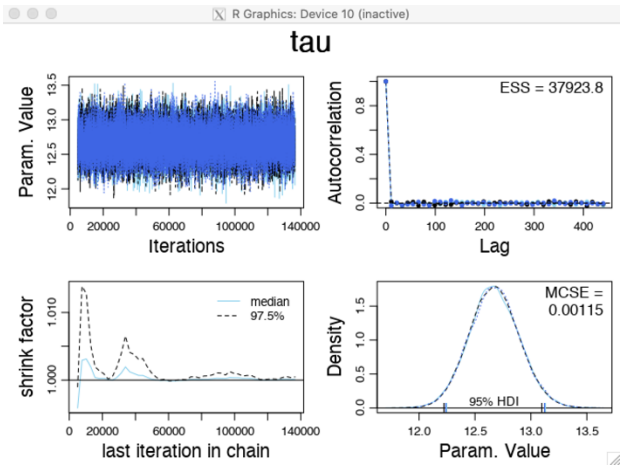
The MCMC diagnostics for β_4 are ideal given that; 1) the chain plots are overlapping and deviate around a mean value, 2) the density plots for all chains, and their HDI's are overlapping, 3) the shrink factor (median) is below the threshold of 1.2, 4) all chains immediately descend to a low autocorrelation value (thus suggesting we have no autocorrelations), 5) effective sample size is high at 31,247 and, 6) we have a low Markov Chain Monte Carlo standard error of 0.000228.

β_5



The MCMC diagnostics for β_5 are ideal given that; 1) the chain plots are overlapping and deviate around a mean value, 2) the density plots for all chains, and their HDI's are overlapping, 3) the shrink factor (median) is below the threshold of 1.2, 4) all chains immediately descend to a low autocorrelation value (thus suggesting we have no autocorrelations), 5) effective sample size is high at 15,632.3 and, 6) we have a low Markov Chain Monte Carlo standard error of 0.000581.

τ



The MCMC diagnostics for τ are ideal given that; 1) the chain plots are overlapping and deviate around a mean value, 2) the density plots for all chains, and their HDI's are overlapping, 3) the shrink factor (median) is below the threshold of 1.2, 4) all chains immediately descend to a low autocorrelation value (thus suggesting we have no autocorrelations), 5) effective sample size is high at 37,923.8 and, 6) we have a low Markov Chain Monte Carlo standard error of 0.00115.

Figure 8.

It should be noted that although these results are ideal; there is slight delay when the rate at which the autocorrelations decay for β_2 and β_3. Furthermore, the shrink factor for these population parameters seem to be higher that the shrink factor for other population parameters. This is because of the prior information we have for β_2 and β_3. These Beta values correspond to variables; Bedrooms and Bathrooms. The degree of belief on expert knowledge for Bedrooms was weak and, we did not have any prior information for Bathrooms. The lack of prior information on these variables, may have reduced the ability for the algorithm to find a suitable distribution, that has a quicker decay in autocorrelations. In other tentative models, we were unable to achieve a quicker delay for these variables (the above results were obtained after thinning by a factor of 15). Furthermore, the selection of variance factors for each distribution is arbitrary. Hence, the results may be improved upon, by appropriate selecting these variables. In fact, the variance figures had been readjusted several times and the specification for each model can be found in the appendix section.

b. Beta Values

Figure 9 provided below, summarises the model's estimates for all population parameters we are estimating.

	Mean	Median	Mode	ESS	HDImass	HDIlow	HDIhigh
CHAIN	2.000000e+00	2.000000000	9.980714e-01	1.5	0.95	1.00000e+00	3.00000e+00
zbeta0	9.484645e-01	0.948447500	9.475952e-01	9901.3	0.95	8.94454e-01	1.00213e+00
zbeta[1]	9.981999e-03	0.009978505	9.741229e-03	33244.5	0.95	3.59886e-03	1.63415e-02
zbeta[2]	2.650407e-02	0.026524850	2.652777e-02	10179.4	0.95	1.13790e-02	4.14433e-02
zbeta[3]	4.666270e-02	0.046673600	4.739833e-02	17197.5	0.95	3.24361e-02	6.12767e-02
zbeta[4]	1.665843e-02	0.016648400	1.641116e-02	30538.1	0.95	3.97215e-03	2.94048e-02
zbeta[5]	2.497076e-03	0.002521000	3.091998e-03	15816.2	0.95	-1.04762e-02	1.54393e-02
beta0	4.859209e+00	4.859125000	4.854755e+00	9901.3	0.95	4.58250e+00	5.13413e+00
beta[1]	9.053270e-05	0.000090501	8.834906e-05	33244.5	0.95	3.26402e-05	1.48211e-04
beta[2]	1.514137e-01	0.151532500	1.515488e-01	10179.4	0.95	6.50063e-02	2.36759e-01
beta[3]	3.940859e-01	0.394178000	4.002985e-01	17197.5	0.95	2.73936e-01	5.17507e-01
beta[4]	1.030015e-01	0.102939500	1.014723e-01	30538.2	0.95	2.45604e-02	1.81814e-01
beta[5]	2.751117e-02	0.027774750	3.406619e-02	15816.2	0.95	-1.15420e-01	1.70100e-01
tau	1.267294e+01	12.670400000	1.266194e+01	37902.4	0.95	1.22341e+01	1.31140e+01
zVar	4.828236e-01	0.482727000	4.824043e-01	37902.4	0.95	4.66106e-01	4.99626e-01
pred[1]	6.135041e+00	6.135155000	6.135495e+00	24966.5	0.95	5.99622e+00	6.27577e+00
pred[2]	5.985965e+00	5.985445000	5.977191e+00	21343.3	0.95	5.87593e+00	6.09748e+00
pred[3]	5.794923e+00	5.794510000	5.803174e+00	14770.5	0.95	5.66376e+00	5.93326e+00
pred[4]	7.830959e+00	7.831140000	7.854145e+00	17696.4	0.95	7.50835e+00	8.15092e+00
pred[5]	6.254768e+00	6.254555000	6.256635e+00	24403.9	0.95	6.11150e+00	6.39949e+00

Figure 9.

β_0 : Beta 0 is the mean sale price for properties in our dataset, assuming all independent variables take up the value of 0. A house that; takes up 0m^2, has no bedrooms, has no bathrooms and no car park space, has a sale price of $485,929 (4.859209 in 100K AUD). Furthermore, there is a 95% probability of observing a sale price between $458,250 and $513,413. The 95% HDI for β_0 does not include the value of 0, thus suggesting that it is statistically significant at the 5% level of significance.

β_1 : Beta 1 is the amount (in 100K AUD) by which sale price for properties increase/decrease, if Area was to increase by $1m^2$. A beta value of 0.0000905 suggests that an additional $1m^2$ of Area, increases the sale price of a property by $9.05 (0.0000905 in 100K AUD). Furthermore, there is a 95% probability of observing a sale price increase between $3.26 and $14.8211, when Area increases by $1m^2$. The 95% HDI for β_1 does not include the value of 0, thus suggesting that it is statistically significant at the 5% level of significance.

β_2 : Beta 2 is the amount (in 100K AUD) by which sale price for properties increase/decrease, if the number of bedrooms available increases by 1. A beta value of 0.1514137 suggests that an additional bedroom, increases the sale price of a property by $15,413.7 (0.1514137 in 100K AUD). Furthermore, there is a 95% probability of observing a sale price increase between $6,500 and $23,675.9, when the number of bedrooms increase by 1. The 95% HDI for β_2 does not include the value of 0, thus suggesting that it is statistically significant at the 5% level of significance.

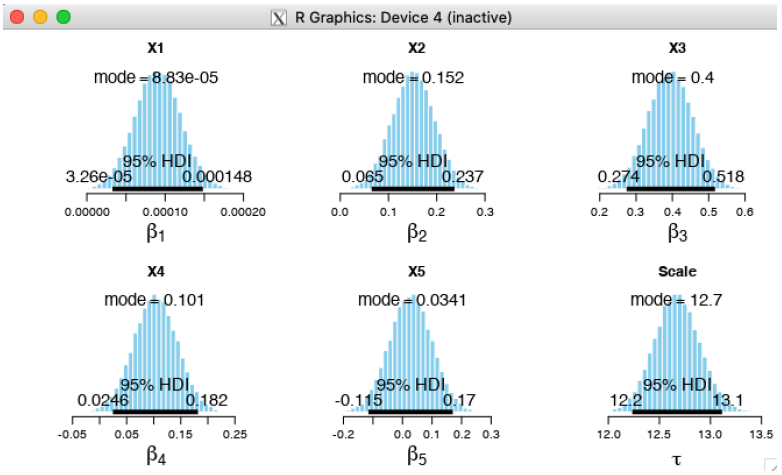
β_3 : Beta 3 is the amount (in 100K AUD) by which sale price for properties increase/decrease, if the number of bathrooms available increases by 1. A beta value of 0.3940859 suggests that an additional bathroom, increases the sale price of a property by $39,408.59 (0.3940859 in 100K AUD). Furthermore, there is a 95% probability of observing a sale price increase between $27,393.69 and $51,750.7, when the number of bathrooms increase by 1. The 95% HDI for β_3 does not include the value of 0, thus suggesting that it is statistically significant at the 5% level of significance.

β_4 : Beta 4 is the amount (in 100K AUD) by which sale price for properties increase/decrease, if the number of car parks that come with the property increases by 1. A beta value of 0.1030015 suggests that an additional car park, increases the sale price of a property by $10,300.15 (0.1030015 in 100K AUD). Furthermore, there is a 95% probability of observing a sale price increase between $2,456.04 and $18,181.4, when the number of car parks increase by 1. The 95% HDI for β_4 does not include the value of 0, thus suggesting that it is statistically significant at the 5% level of significance.

β_5 : Beta 5 is the amount (in 100K AUD) by which sale price for properties increase/decrease, if the property was a unit. A beta value of 0.02751117 suggests that a property, that is a unit, increases the sale price by $2,751.12 (0.02751117 in 100K AUD). Furthermore, there is a 95% probability of observing a sale price increase between -$11,542 and $17,010, when the property is a unit. The 95% HDI for β_5 does include the value of 0, thus suggesting that the beta value may be statistically insignificant at the 5% level of significance. We will however need further tests to acknowledge this, given we expect that X5 (PropertyType) takes up a value of 1 for units and 0 for houses. We can make the safe assumption that a house would sell for a higher price than a unit. Hence, if X5 was to take up a value of 1, we would expect sale price to reduce, not increase. Hence, it may be acceptable to have negative values for β_5.

c. Posterior Distributions

Figure 10 provided below, summarises the posterior distributions for each parameter:



β_1: The data is centred around the mode of 0.0000883 ($8.83).

β_2: The data is centred around the mode of 0.152 ($15,200).

β_3: The data is centred around the mode of 0.4 ($40,000).

β_4: The data is centred around the mode of 0.101 ($10,100).

β_5: The data is centred around the mode of 0.0341 ($3,410).

τ: The variance is centred around the mode of 12.7.

Figure 10.

d. Does the posterior distribution for SalesPrice$_i$ (Y_i) fit the likelihood function?

Figure 11 exhibits the density plots for the likelihood function and, the estimated posterior distribution for sales price for properties in our dataset. The posterior distribution is created by utilising the prior distributions of the parameters and the likelihood function of the dataset. We use the selected regression model to predict sale prices for houses, over randomized data. The distribution for predicted sale prices is then overlayed on the actual sale prices, to observe the ability of our regression model in its ability to fit the data well.

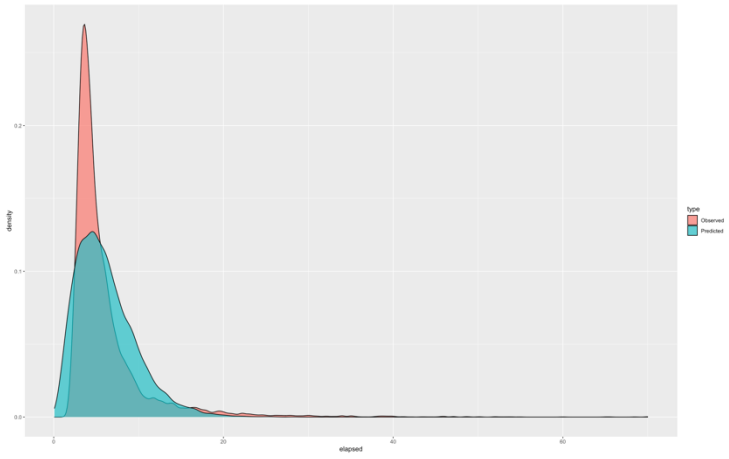


Figure 11

The density plot for the posterior distribution created by our model, does in fact fit the data well. Hence, we can proceed with analysing the property price predictions in the next section.

Predictions

The following table consists of assumed independent variable values, over which we shall predict the dependent variable values. Particularly, we shall predict the sale price of properties that have the following features:

Property	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 12.

The following table summarises the predictions of property price, given the above features:

Property	Price Prediction (in 100k AUD)	95% lower HDI (in 100k AUD)	95% upper HDI (in 100k AUD)
1	6.135041	5.99622	6.27577
2	5.985965	5.87593	6.09748
3	5.794923	5.66376	5.93326
4	7.830959	7.50835	8.15092
5	6.254768	6.11150	6.39949

Figure 13.

Conclusion

The report explored the use of Bayesian Regression techniques to predict property prices, based on; area of the property, number of bedrooms available, number of bathrooms available, number of car parks available and property type. In order to be able to achieve this, a posterior distribution for the sale price of properties was created by utilising prior distributions of population parameters and, the likelihood function of sales price given the data available to us. The population parameters in question were the beta values of our multi-regression model in addition to the variance of sales price itself. Hence, a combined of 7 prior distributions (6 normal distributions for beta estimates and 1 gamma distribution for variance estimates) had been utilised. Furthermore, the likelihood function for the sale price data available to us, was assumed to be a gamma distribution given that its density plot had a positive skew.

The location (central tendency) of the prior distributions for each population parameter was determined and specified based on expert knowledge. Based on the belief we placed in this expert knowledge, variance for each prior distribution was also specified. Thus, determining the informativeness of each prior distribution, relative to the informativeness of the likelihood function.

Given that the prior distributions and the likelihood function, were not mathematically-tractable, a Markov Chain Monte Carlo algorithm (courtesy of Kruschke [4] and Haydar [3]) was implemented to find the posterior distribution for property sale prices. The simulation was run several times, using several specifications, until an ideal model was created.

The model was determined to be ideal based on evaluations of MCMC diagnostics. Followed by which, we further evaluated the posterior distribution of sale price and the simulated prior distributions- which will behave as prior distributions in the next period)- visually.

Once an ideal model was found, property prices were predicted for 5 different properties based on their features. The ability for the posterior distribution to predict property prices efficiently, was examined by a visual comparison of the likelihood function and the density plot of the posterior distribution. It was found, that our selected model does in fact fit the data well and the predictions made by the model are reliable.

The methodology provided in this report deviates from methodology utilised in classical statistics. Specifically, in classical statistics we would have run a regression model that regresses sale price on the independent variables, using only the likelihood function, while making assumptions about the population parameters. In Bayesian statistics, we instead simulate the distributions (prior distributions) of the population parameters based on expert knowledge and, utilise these simulated distributions in addition to what the data already tell us, to create a posterior distribution. The regression itself is carried out to determine the location (centre) of the posterior distribution, by using the simulated prior distributions for each beta. Like this, we are able to create a posterior distribution for property prices that is assumed to contain more information than the likelihood function on which we can base our inferences. This being said, the report does not give preference to one method over the other. The Bayesian way of creating posterior distributions is simply an alternative method that *can* provide more accurate results.

References

[1] H. M. Blalock, Jr., “Correlated Independent Variables: The Problem of Multicollinearity,” *Social Science,* Vol. 42, no. 2, pp. 233-273, 1963.

[2] Wikipedia. “Gamma Distribution”. Wikipedia. https://en.wikipedia.org/wiki/Gamma_distribution (accessed September 19th, 2020).

[3] H. Demirhan, “Modeul 6 – Bayesian Linear Regression”. Royal Melbourne Institute of Technology, Victoria, Australia, Semester 2, 2020.

[4] J. K. Kruschke, *Doing Bayesian Data Analysis,* 2nd ed. A Tutorial with R, JAGS, and Stan, Academic Press / Elsevier, 2015.

[5] B. Rele “Property Price Predictions: A Bayesian Analysis,” unpublished, Aug. 2020.

APPENDIX

Initial States	Diagnostics Satisfaction

`initsList <- list(`
`  zbeta0 = 90,`
`  zbeta = c(50, 1, 1, 0,1),`
`  Var = 50`
`)`
 	
`adaptSteps = 1500  # Number of steps to "tune" the samplers`
`burnInSteps = 2000`
`nChains = 3`
`thinSteps = 2 # First run for 3`
`numSavedSteps = 4000`
`nIter = ceiling( ( numSavedSteps * thinSteps ) / nChains )`
	Overall Result: Unsatisfactory.
	
	MCMC diagnostics indicated autocorrelations and high shrink factor.

`initsList <- list(`
`  zbeta0 = 20,`
`  zbeta = c(0.001, 1, 1, 0,1),`
`  Var = 26`
`)`
`adaptSteps = 1500  # Number of steps to "tune" the samplers`
`burnInSteps = 5000`
`nChains = 3`
`thinSteps = 11 # First run for 3`
`numSavedSteps = 8000`
`nIter = ceiling( ( numSavedSteps * thinSteps ) / nChains )`
	Overall Result: Unsatisfactory.
	
	MCMC diagnostics indicated autocorrelations and high shrink factor. The chains were not overlapping; possibly due to bad initial values selection.

`adaptSteps = 1500  # Number of steps to "tune" the samplers`
`burnInSteps = 3000`
`nChains = 3`
`thinSteps = 5 # First run for 3`
`numSavedSteps = 8000`
`nIter = ceiling( ( numSavedSteps * thinSteps ) / nChains )`
 	
`runJagsOut <- run.jags( method="parallel" ,`
`model="TEMPmodel2.txt" ,`
`monitor=c( "zbeta0" ,  "zbeta" , "beta0" ,  "bet`
`data=dataList ,`
`#inits=initsList ,`
`n.chains=nChains ,`
`adapt=adaptSteps ,`
`burnin=burnInSteps ,`
`sample=numSavedSteps ,`
`thin=thinSteps , summarise=FALSE , plots=FALSE )`
`codaSamples = as.mcmc.list( runJagsOut )`
	Overall Result: Unsatisfactory.
	
	MCMC diagnostics indicated autocorrelations and high shrink factor. The chains were overlapping, unlike previous models; possibly because initials were taken out.

`adaptSteps = 1500  # Number of steps to "tune" the samplers`
`burnInSteps = 3000`
`nChains = 3`
`thinSteps = 11 # First run for 3`
`numSavedSteps = 12000`
`nIter = ceiling( ( numSavedSteps * thinSteps ) / nChains )`
`runJagsOut <- run.jags( method="parallel" ,`
`model="TEMPmodel3.txt"`
`monitor=c( "zbeta0" ,`
`data=dataList ,`
`#inits=initsList ,`
`n.chains=nChains ,`
`adapt=adaptSteps ,`
`burnin=burnInSteps ,`
`sample=numSavedSteps ,`
`thin=thinSteps , summar`
`codaSamples = as.mcmc.list( runJagsOut )`
 	
`save.image(file="rEnvironment3.RData")`
	Overall Result: Satisfactory, however; beta 0, beta 3 and beta 4 still have high autocorrelations and, low ESS' in addition to high shrink factors. Perhaps because the information being used in priors is not suitable. Hence, we change the prior information and, include initials in the next run.

initsList <- list(
  zbeta0 = -18.2,
  zbeta = c(-0.00023, 101, -32.7, 11.4, 46.5),
  Var = 9230000
)
 	
adaptSteps = 1500 # Number of steps to "tune" the samplers
burnInSteps = 3000
nChains = 3
thinSteps = 11 # First run for 3
numSavedSteps = 12000
nIter = ceiling((numSavedSteps * thinSteps) / nChains)
 	
# Priors on standardized scale: 	
zbeta0 ~ dnorm(0 , 1/2^2) # 1/ variance for normal distribution
zbeta[1] ~ dnorm(0.09/xsd[1] , 1/(5/xsd[1]^2)) # 1/ variance for normal distribution
zbeta[2] ~ dnorm(100/xsd[2] , 1/(1/xsd[2]^2)) # 1/ variance for normal distribution
zbeta[3] ~ dnorm(0 , 1/4) # 1/ variance for normal distribution
zbeta[4] ~ dnorm(120/xsd[4] , 1/(2.5/xsd[4]^2)) # 1/ variance for normal distribution
zbeta[5] ~ dnorm(-150/xsd[5] , 1/(5/xsd[5]^2)) # 1/ variance for normal distribution
	Overall Result: Satisfactory.
	
	Results had substantially improved. The change in prior information for variance of each population parameter made a big difference

initsList <- list(
  zbeta0 = 0.951,
  zbeta = c(0.0101, 0.0276, 0.0479, 0.0171, 0.00192),
  Var = 12.6
)
 	
adaptSteps = 1500 # Number of steps to "tune" the samplers
burnInSteps = 3000
nChains = 3
thinSteps = 15 # First run for 3
numSavedSteps = 10000
nIter = ceiling((numSavedSteps * thinSteps) / nChains)
 	
# Priors on standardized scale: 	
zbeta0 ~ dnorm(0 , 1/2^2) # 1/ variance for normal distribution
zbeta[1] ~ dnorm(0.09/xsd[1] , 1/(5/xsd[1]^2)) # 1/ variance for normal distribution
zbeta[2] ~ dnorm(100/xsd[2] , 1/(1/xsd[2]^2)) # 1/ variance for normal distribution
zbeta[3] ~ dnorm(0 , 1/4) # 1/ variance for normal distribution
zbeta[4] ~ dnorm(120/xsd[4] , 1/(4/xsd[4]^2)) # 1/ variance for normal distribution
zbeta[5] ~ dnorm(-150/xsd[5] , 1/(5/xsd[5]^2)) # 1/ variance for normal distribution
	Overall Result: Satisfactory.
	
	This is the chosen model in the analysis. The model diagnostics and evaluation can be found in the report.

Figure 12.

Codes for Runnings JAGS on R:

```
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)
library(readr)
setwd("~/Desktop/Applied Bayesian Statistics/assignment 2- property prices prof/R working")
source("DBDA2E-utilities.R")

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

decideOpenGraph = function( panelCount , saveName , finished=FALSE ,
                            nRow=2 , nCol=3 ) {
  # If finishing a set:
  if ( finished==TRUE ) {
    if ( !is.null(saveName) ) {
      saveGraph( file=paste0(saveName,ceiling((panelCount-1)/(nRow*nCol))),
                 type=saveType)
    }
    panelCount = 1 # re-set panelCount
    return(panelCount)
  } else {
    # If this is first panel of a graph:
    if ( ( panelCount %% (nRow*nCol) ) == 1 ) {
```

```
      # If previous graph was open, save previous one:
      if ( panelCount>1 & !is.null(saveName) ) {
       saveGraph( file=paste0(saveName,(panelCount%/%(nRow*nCol))),
              type=saveType)
      }
      # Open new graph
      openGraph(width=nCol*7.0/3,height=nRow*2.0)
      layout( matrix( 1:(nRow*nCol) , nrow=nRow, byrow=TRUE ) )
      par( mar=c(4,4,2.5,0.5) , mgp=c(2.5,0.7,0) )
     }
     # Increment and return panel count:
     panelCount = panelCount+1
     return(panelCount)
   }
  }

  # Original scale:
  panelCount = 1
  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 ,  finished=TRUE ,
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====================

#data importation
myData <- read_csv("Assignment2PropertyPrices.csv")
head(myData)

#basic summary stats
myData$Bedrooms <- factor(myData$Bedrooms, labels=c("1","2","3","4","5","6","7"))
myData$Bathrooms <- factor(myData$Bathrooms,labels=c("1","2","3","4"))
myData$CarParks <-
factor(myData$CarParks,labels=c("0","1","2","3","4","5","6","7","8","9"))
myData$PropertyType <- factor(myData$PropertyType,labels=c("0","1"))
myData %>% summary()
colSums(is.na(myData))

p1 <- myData %>% ggplot(aes(x=Area, y=`SalePrice(100K)`,color=PropertyType)) +
  geom_point() +
  theme_bw()+
  xlab("Area") +
  ylab("SalePrice(100K)")
p1

p2 <- myData %>% ggplot(aes(x=Bedrooms, y=`SalePrice(100K)`, fill=Bedrooms)) +
  scale_fill_brewer(palette= "Set3")+
  geom_boxplot(show.legend=FALSE) +
  theme_bw() +
  xlab("Bedrooms") +
  ylab("SalePrice(100K)")
p2

p3 <- myData %>% ggplot(aes(x=Bathrooms, y=`SalePrice(100K)`,fill=Bathrooms)) +
  scale_fill_brewer(palette= "Set3")+
  geom_boxplot(show.legend=FALSE) +
  theme_bw()+
  xlab("Bathrooms") +
  ylab("SalePrice(100K)")
p3

p4 <- myData %>% ggplot(aes(x=CarParks, y=`SalePrice(100K)`, fill=CarParks)) +
  scale_fill_brewer(palette= "Set3")+
  geom_boxplot(show.legend=FALSE) +
  theme_bw()+
  xlab("CarParks") +
  ylab("SalePrice(100K)")
```

```
p4

p5 <- myData %>% ggplot(aes(x=PropertyType, y=`SalePrice(100K)`, fill=PropertyType)) +
  scale_fill_brewer(palette= "Set3")+
  geom_boxplot(show.legend=FALSE) +
  theme_bw()+
  xlab("PropertyType") +
  ylab("SalePrice(100K)")
p5

#re-import dataset so the features are numeric again (to get rid of earlier factorization)
myData <- read.csv("Assignment2PropertyPrices.csv")
cor(myData)

# Histogram
hist(myData$SalePrice.100K,col="lightskyblue",
    main= " Histogram of the dependent variable", xlab = "Sale Price (100K)")
# Kernel density estimation
plot(kde(myData$SalePrice.100K.),col="red",xlab = "Sale Price (100K)") # with default
settings

#change column names for modelling phase
names(myData)[1] <- "Y"
names(myData)[2] <- "X1"
names(myData)[3] <- "X2"
names(myData)[4] <- "X3"
names(myData)[5] <- "X4"
names(myData)[6] <- "X5"
colnames(myData)

# THE DATA----------------------------------------------------------------------------
y = myData[,"Y"]
x = as.matrix(myData[,c("X1","X2","X3","X4","X5")])

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]
)

# INITIALS----------------------------------------------------------------------------------------------------------
-------
initsList <- list(
  zbeta0 = 0.951,
  zbeta = c(0.0101, 0.0276, 0.0479, 0.0171, 0.00192),
  Var = 12.6
)

# 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
  zbeta[1] ~ dnorm( 0.09/xsd[1] , 1/(5/xsd[1]^2) ) # 1/ variance for normal distribution
  zbeta[2] ~ dnorm( 100/xsd[2] , 1/(1/xsd[2]^2) ) # 1/ variance for normal distribution
  zbeta[3] ~ dnorm( 0 , 1/4 ) # 1/ variance for normal distribution
  zbeta[4] ~ dnorm( 120/xsd[4] , 1/(4/xsd[4]^2) ) # 1/ variance for normal distribution
  zbeta[5] ~ dnorm( -150/xsd[5] , 1/(5/xsd[5]^2) ) # 1/ variance for normal distribution

  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="TEMPmodel5.txt" )

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

#DIAGNOSTICS FOCUSED ADJUSTMENTS----------------------------------------------------------------
----------------
adaptSteps = 1500  # Number of steps to "tune" the samplers
burnInSteps = 3000
nChains = 3
thinSteps = 15 # First run for 3
numSavedSteps = 10000
nIter = ceiling( ( numSavedSteps * thinSteps ) / nChains )

# PARALLEL RUN-----------------------------------------------------------------------------------------
-----
runJagsOut <- run.jags( method="parallel" ,
                    model="TEMPmodel5.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="rEnvironment5.RData")
load(file="rEnvironment5.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]" )
diagMCMC( codaSamples , parName="zbeta[2]" )
diagMCMC( codaSamples , parName="zbeta[3]" )
diagMCMC( codaSamples , parName="zbeta[4]" )
diagMCMC( codaSamples , parName="zbeta[5]" )

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 )
print(summaryInfo)


plotMCMC_HD( codaSamples = codaSamples , data = myData,
xName=c("X1","X2","X3","X4","X5") ,
          yName="Y", 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=10000,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(data=dataPred, aes(elapsed, fill = type))  + geom_density(alpha = 0.7)
```