Comparing the Bayesian Linear Regression Model to Ordinary Least Squares

Joe Zhang and Christian Paravalos

Winter 2024

Abstract

Linear regression is often used to model relationships between a response variable and one or more explanatory variables. In traditional settings, the ordinary least squares method is used to estimate the model. In this paper, our goal is to develop a linear model by estimating the model parameters under a Bayesian framework and compare the model against the traditional ordinary least squares model using a dataset.

Introduction

To model the relationship between a response variable and one or more explanatory variables, the simplest way to estimate a model is to assume that there is a linear relationship between the response variable and the predictor variables. The formulation of the model is

$$\boldsymbol{y} = X\boldsymbol{\beta} + \boldsymbol{\epsilon},$$

where X is an $n \times k$ matrix with the observations of the explanatory variables, $\boldsymbol{y}$ is an $n \times 1$ vector of observations of the response variables, $\boldsymbol{\beta}$ is a $k \times 1$ vector of model parameters, and $\boldsymbol{\epsilon}$ is an $n \times 1$ vector of random errors associated with each observation, n is the number of observations, and k is the number of predictors The goal is to estimate $\boldsymbol{\beta}$ and then construct and evaluate the model from the estimated values. In other words, we have

$$\boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \quad X = \begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1(k-1)} \\ 1 & x_{21} & x_{22} & \cdots & x_{2(k-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & \cdots & x_{n(k-1)} \end{bmatrix}, \quad \boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_{k-1} \end{bmatrix}, \quad \boldsymbol{\epsilon} = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix}$$

The most well-known way is to use the ordinary least squares method to estimate the parameters. The goal of this paper is to compare the ordinary least squares method with the Bayesian method of estimating parameters.

Ordinary Least Squares Regression Revisited

Estimation of Model Parameters

Consider a multiple linear regression model

$$\boldsymbol{y} = X\boldsymbol{\beta} + \boldsymbol{\epsilon},$$

assuming that the random errors have mean 0, variance σ^2, are uncorrelated, and follow a normal distribution. In other words, we assume that $\boldsymbol{\epsilon} \sim N(\mathbf{0}, \sigma^2 I)$, where $\mathbf{0}$ is the zero vector, and I is the identity matrix.

To estimate the coefficients, the following quantity should be minimized:

$$S(\boldsymbol{\beta}) = \sum_{i=1}^{n} \epsilon_i^2 = \boldsymbol{\epsilon}^T \boldsymbol{\epsilon} = (\boldsymbol{y} - X\boldsymbol{\beta})^T (\boldsymbol{y} - X\boldsymbol{\beta})$$

We need to take derivatives of $S(\boldsymbol{\beta})$ with respect to $\boldsymbol{\beta}$. We get that

$$\frac{\partial S(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = -2X^T \boldsymbol{y} + 2X^T X\boldsymbol{\beta}$$

If we set this equation to 0 and we solve for $\boldsymbol{\beta}$, we get

$$X^T X\boldsymbol{\beta} = X^T \boldsymbol{y}$$

If we multiply both sides by the inverse of $X^T X$, assuming that it is invertible, then the least squares estimator of $\boldsymbol{\beta}$ is

$$\hat{\boldsymbol{\beta}} = (X^T X)^{-1} X^T \boldsymbol{y}$$

The least squares estimator of $\boldsymbol{\beta}$ has the following properties:

- $\mathbb{E}[\hat{\boldsymbol{\beta}}] = \boldsymbol{\beta}$
- $Var[\hat{\boldsymbol{\beta}}] = \sigma^2 (X^T X)^{-1}$

Estimation of Variance

In the frequentist approach, we estimate σ^2 using the residual sum of squares. We have

$$SS_{res} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} e_i^2 = \boldsymbol{e}^T \boldsymbol{e}$$

where $\hat{y}_i$ are the fitted values of y_i using the least squares model and $e_i = y_i - \hat{y}_i$ are the residuals. $\boldsymbol{e}$ is the vector of the residuals. If we simplify further, we get

$$SS_{res} = \boldsymbol{y}^T \boldsymbol{y} - \hat{\boldsymbol{\beta}}^T X^T \boldsymbol{y}$$

The residual sum of squares has $n - k$ degrees of freedom. The residual mean square is

$$MS_{res} = \frac{SS_{res}}{n - k}$$

We can show that

$$\mathbb{E}[MS_{res}] = \sigma^2$$

Therefore, an unbiased estimator for σ^2 is given by

$$\hat{\sigma^2} = MS_{res} = \frac{\boldsymbol{y}^T \boldsymbol{y} - \hat{\boldsymbol{\beta}}^T X^T \boldsymbol{y}}{n - k}$$

Inference on model parameters

t-tests

After obtaining the estimates of the model parameters, we need to make inference on them. The main goal of inference is to determine if all the explanatory variables are significant to the model. Sometimes when there are too many variables, the accuracy of predictions using the model can decrease significantly. Making inferences can help decide if some variables should not be included in the final model.

We have assumed that $\boldsymbol{\epsilon} \sim N(\mathbf{0}, \sigma^2 I)$. This consequently leads to that $\boldsymbol{y} \sim N(X\boldsymbol{\beta}, \sigma^2 I)$. Under this assumption, we can also get that

$$\hat{\boldsymbol{\beta}} \sim N(\boldsymbol{\beta}, \sigma^2 (X^T X)^{-1})$$

From this, we can also deduce that

$$\hat{\beta}_j \sim N(\beta_j, \sigma^2 (X^T X)^{-1}_{jj})$$

for $j = 1, 2, ..., k$ and $(X^T X)^{-1}_{jj}$ is the j-th diagonal element of the matrix $(X^T X)^{-1}$.

If we standardize $\hat{\beta}$, we get that

$$\frac{\hat{\beta}_j - \beta_j}{\sigma\sqrt{(X^T X)^{-1}_{jj}}} \sim N(0, 1)$$

In practice, we do not know the value of σ^2, which we will need to use in order to make inference on the model parameters. Therefore, the best way is to use the estimate for σ^2, which is MS_{res}. However, if we replace σ^2 with MS_{res}, we get

$$\frac{\hat{\beta}_j - \beta_j}{\sqrt{MS_{res}(X^T X)^{-1}_{jj}}} \sim t(n-k)$$

If we want to test the null hypothesis $H_0 : \beta_j = \beta_{j_0}$ against the hypothesis $H_1 : \beta_j \neq \beta_{j_0}$, we have that under H_0 that

$$t_0 = \frac{\hat{\beta}_j - \beta_{j_0}}{\sqrt{MS_{res}(X^T X)^{-1}_{jj}}} \sim t(n-k)$$

t_0 is the test statistic for the hypothesis $\beta_j = \beta_{j_0}$. We reject H_0 if $|t_0| > t_{\alpha/2}(n-k)$, where α is the significance level and $t_{\alpha/2}(n-k)$ is the $100(1-\alpha/2)$th quantile of the Student t distribution with $n-k$ degrees of freedom. We can also compute the p-value, which is

$$p\text{-value} = P(|T| > |t_0|) = 2P(T > t_0)$$

and we reject H_0 if the p-value is less than α.

Another way to test the hypothesis H_0 against H_1 is to construct confidence intervals for the model parameters β_j. We can deduce that a $100(1-\alpha)\%$ confidence interval for β_j is

$$\hat{\beta}_j \pm t_{\alpha/2}(n-k) \cdot \sqrt{MS_{res}(X^T X)^{-1}_{jj}}$$

From the confidence interval, we accept H_0 if β_{j_0} is contained in the interval. Otherwise, we reject H_0.

Bayesian Linear Regression

Now, we turn to the Bayesian approach to estimate a linear regression model between a response variable and several explanatory variables. The form of the model is the same as in the ordinary least squares framework

$$\boldsymbol{y} = X\boldsymbol{\beta} + \boldsymbol{\epsilon}$$

Estimation of Parameters

We make the same assumptions as in the ordinary least squares framework, where $\boldsymbol{\epsilon} \sim N_n(\mathbf{0}, \sigma^2 I)$. We need to estimate $k+1$ parameters: the k model coefficients, and σ^2.

Likelihood Function

With the assumption, we have that

$$\boldsymbol{y} \sim N_n(X\boldsymbol{\beta}, \sigma^2 I)$$

Therefore, the likelihood function is

$$f(\boldsymbol{y}|X, \boldsymbol{\beta}, \sigma^2) = \frac{1}{(2\pi)^{n/2}(\sigma^2)^{n/2}} \exp\left[-\frac{1}{2\sigma^2}(\boldsymbol{y} - X\boldsymbol{\beta})^T(\boldsymbol{y} - X\boldsymbol{\beta})\right]$$

Choosing a prior distribution

To estimate $\boldsymbol{\beta}$ and σ^2, we treat them as random variables and we make a prior assumption on their distribution. The assumptions we will use are

$$\boldsymbol{\beta}|\sigma^2 \sim N_k(\boldsymbol{m}, \sigma^2 V)$$

and

$$\sigma^2 \sim IG(a, b)$$

where $\boldsymbol{m}$ is a $k \times 1$ vector of real numbers, V is a $k \times k$ matrix of real numbers, and a and b are both real numbers.

Then, we get that the prior for $(\boldsymbol{\beta}, \sigma^2)$ is

$$g(\boldsymbol{\beta}, \sigma^2) = g(\boldsymbol{\beta}|\sigma^2)g(\sigma^2)$$

where

$$g(\boldsymbol{\beta}|\sigma^2) = \frac{1}{(2\pi\sigma^2)^{k/2}}|V|^{-\frac{k}{2}} \exp\left(-\frac{1}{2\sigma^2}(\boldsymbol{\beta} - \boldsymbol{m})^T V^{-1}(\boldsymbol{\beta} - \boldsymbol{m})\right)$$

and

$$g(\sigma^2) = \frac{b^a}{\Gamma(a)}(\sigma^2)^{-(a+1)} \exp\left(-\frac{b}{\sigma^2}\right)$$

Computation of posterior distribution

Now that we have the likelihood function and the prior probability distribution of the parameters, we can now compute the posterior distribution of the parameters. We let $h(\boldsymbol{\beta}, \sigma^2|\boldsymbol{y}, X)$ represent the posterior distribution function:

$$\begin{aligned}
h(\boldsymbol{\beta}, \sigma^2|\boldsymbol{y}, X) &\propto f(\boldsymbol{y}|X, \boldsymbol{\beta}, \sigma^2)g(\boldsymbol{\beta}, \sigma^2) \\
&\propto f(\boldsymbol{y}|X, \boldsymbol{\beta}, \sigma^2)g(\boldsymbol{\beta}|\sigma^2)g(\sigma^2) \\
&\propto (2\pi\sigma^2)^{-\frac{n}{2}} \exp\left(-\frac{1}{2\sigma^2}(\boldsymbol{y} - X\boldsymbol{\beta})^T(\boldsymbol{y} - X\boldsymbol{\beta})\right)(2\pi\sigma^2)^{-\frac{k}{2}}|V|^{-\frac{k}{2}} \exp\left(-\frac{1}{2\sigma^2}(\boldsymbol{\beta} - \boldsymbol{m})^T V^{-1}(\boldsymbol{\beta} - \boldsymbol{m})\right) \\
&\times \frac{b^a}{\Gamma(a)}(\sigma^2)^{-(a+1)} \exp\left(-\frac{b}{\sigma^2}\right) \\
&\propto (\sigma^2)^{-\frac{n}{2}-\frac{k}{2}-(a+1)} \exp\left(-\frac{1}{2\sigma^2}[(\boldsymbol{y} - X\boldsymbol{\beta})^T(\boldsymbol{y} - X\boldsymbol{\beta}) + (\boldsymbol{\beta} - \boldsymbol{m})^T V^{-1}(\boldsymbol{\beta} - \boldsymbol{m}) + 2b]\right) \\
&\propto (\sigma^2)^{-\frac{n}{2}-\frac{k}{2}-(a+1)} \exp\left(-\frac{1}{2\sigma^2}A\right)
\end{aligned}$$

We now want to rewrite the equation above by simplifying the quantity inside the exponential function. We do the following:

$$\begin{aligned}
A &= (\boldsymbol{y} - X\boldsymbol{\beta})^T(\boldsymbol{y} - X\boldsymbol{\beta}) + (\boldsymbol{\beta} - \boldsymbol{m})^T V^{-1}(\boldsymbol{\beta} - \boldsymbol{m}) + 2b \\
&= \boldsymbol{y}^T\boldsymbol{y} - \boldsymbol{y}^T X\boldsymbol{\beta} - \boldsymbol{\beta}^T X^T\boldsymbol{y} + \boldsymbol{\beta}^T X^T X\boldsymbol{\beta} + \boldsymbol{\beta}^T V^{-1}\boldsymbol{\beta} - \boldsymbol{\beta}^T V^{-1}\boldsymbol{m} - \boldsymbol{m}^T V^{-1}\boldsymbol{\beta} + \boldsymbol{m}^T V^{-1}\boldsymbol{m} + 2b \\
&= \boldsymbol{\beta}^T(X^T X + V^{-1})\boldsymbol{\beta} - \boldsymbol{\beta}^T(X^T\boldsymbol{y} + V^{-1}\boldsymbol{m}) + (\boldsymbol{m}^T V^{-1}\boldsymbol{m} + 2b + \boldsymbol{y}^T\boldsymbol{y}) - (\boldsymbol{y}^T X + \boldsymbol{m}^T V^{-1})\boldsymbol{\beta}
\end{aligned}$$

We define the following quantities:

$$\Lambda = (X^T X + V^{-1})^{-1}$$

and

$$\boldsymbol{\mu} = (X^T X + V^{-1})^{-1}(X^T\boldsymbol{y} + V^{-1}\boldsymbol{m})$$

where Λ is a $k \times k$ matrix and $\boldsymbol{\mu}$ is a $k \times 1$ vector.

Then we can write

$$\begin{aligned}
A &= \boldsymbol{\beta}^T\Lambda^{-1}\boldsymbol{\beta} - \boldsymbol{\beta}^T\Lambda^{-1}\boldsymbol{\mu} - \boldsymbol{\mu}^T\Lambda^{-1}\boldsymbol{\beta} + \boldsymbol{m}^T V^{-1}\boldsymbol{m} + 2b + \boldsymbol{y}^T\boldsymbol{y} \\
&= (\boldsymbol{\beta} - \boldsymbol{\mu})^T\Lambda^{-1}(\boldsymbol{\beta} - \boldsymbol{\mu}) - \boldsymbol{\mu}^T\Lambda^{-1}\boldsymbol{\mu} + \boldsymbol{m}^T V^{-1}\boldsymbol{m} + 2b + \boldsymbol{y}^T\boldsymbol{y}
\end{aligned}$$

Now, we go back to the posterior distribution and we can write:

$$\begin{aligned}
h(\boldsymbol{\beta}, \sigma^2|\boldsymbol{y}, X) &\propto (\sigma^2)^{-\frac{n}{2}-\frac{k}{2}-(a+1)} \exp\left(-\frac{1}{2\sigma^2}A\right) \\
&\propto (\sigma^2)^{-\frac{n}{2}-\frac{k}{2}-(a+1)} \exp\left(-\frac{1}{2\sigma^2}((\boldsymbol{\beta} - \boldsymbol{\mu})^T\Lambda^{-1}(\boldsymbol{\beta} - \boldsymbol{\mu}) - \boldsymbol{\mu}^T\Lambda^{-1}\boldsymbol{\mu} + \boldsymbol{m}^T V^{-1}\boldsymbol{m} + 2b + \boldsymbol{y}^T\boldsymbol{y})\right) \\
&\propto (\sigma^2)^{-\frac{k}{2}} \exp\left(-\frac{(\boldsymbol{\beta} - \boldsymbol{\mu})^T\Lambda^{-1}(\boldsymbol{\beta} - \boldsymbol{\mu})}{2\sigma^2}\right)(\sigma^2)^{-(\frac{n}{2}+a+1)} \exp\left(-\frac{\boldsymbol{m}^T V^{-1}\boldsymbol{m} - \boldsymbol{\mu}^T\Lambda^{-1}\boldsymbol{\mu} + 2b + \boldsymbol{y}^T\boldsymbol{y}}{2\sigma^2}\right)
\end{aligned}$$

From looking at the posterior distribution, we can see that it is the product of a multivariate normal distribution and an inverse gamma distribution. We can parameterize the posterior the following way:

$$h(\boldsymbol{\beta}, \sigma^2|\boldsymbol{y}, X) = h(\boldsymbol{\beta}|\sigma^2, \boldsymbol{y}, X)h(\sigma^2|\boldsymbol{y}, X)$$

where

$$h(\boldsymbol{\beta}|\sigma^2, \boldsymbol{y}, X) = \frac{1}{(2\pi\sigma^2)^{\frac{k}{2}}} \exp\left(-\frac{(\boldsymbol{\beta} - \boldsymbol{\mu})^T\Lambda^{-1}(\boldsymbol{\beta} - \boldsymbol{\mu})}{2\sigma^2}\right)$$

and

$$h(\sigma^2|\boldsymbol{y}, X) = \frac{(\frac{1}{2}(\boldsymbol{m}^T V^{-1}\boldsymbol{m} - \boldsymbol{\mu}^T\Lambda^{-1}\boldsymbol{\mu} + 2b + \boldsymbol{y}^T\boldsymbol{y}))^{\frac{n}{2}+a}}{\Gamma(a + \frac{n}{2})}(\sigma^2)^{-(\frac{n}{2}+a+1)} \exp\left(-\frac{\boldsymbol{m}^T V^{-1}\boldsymbol{m} - \boldsymbol{\mu}^T\Lambda^{-1}\boldsymbol{\mu} + 2b + \boldsymbol{y}^T\boldsymbol{y}}{2\sigma^2}\right)$$

Then we get that

$$\boldsymbol{\beta}|\sigma^2, \boldsymbol{y}, X \sim N_k(\boldsymbol{\mu}, \sigma^2\Lambda)$$

and

$$\sigma^2|\boldsymbol{y}, X \sim IG(\alpha, r)$$

where

$$\boldsymbol{\mu} = (X^T X + V^{-1})^{-1}(X^T \boldsymbol{y} + V^{-1}\boldsymbol{m})$$
$$\Lambda = (X^T X + V^{-1})^{-1}$$
$$\alpha = a + \frac{n}{2}$$
$$r = \frac{\boldsymbol{m}^T V^{-1}\boldsymbol{m} - \boldsymbol{\mu}^T \Lambda^{-1}\boldsymbol{\mu} + 2b + \boldsymbol{y}^T \boldsymbol{y}}{2}$$

Making Bayesian inference

We now have the posterior distribution of $(\boldsymbol{\beta}, \sigma^2)$. From that, we also get that

$$\boldsymbol{\beta}|\sigma^2, \boldsymbol{y}, X \sim N_k(\boldsymbol{\mu}, \sigma^2\Lambda)$$

and

$$\sigma^2|\boldsymbol{y}, X \sim IG(\alpha, r)$$

The main goal is to estimate the parameters. We first need to find the marginal distribution of $\boldsymbol{\beta}$. This is hard to do by hand since we have to solve the following integral:

$$h(\boldsymbol{\beta}|\boldsymbol{y}, X) = \int_{-\infty}^{\infty} h(\boldsymbol{\beta}|\sigma^2, \boldsymbol{y}, X)h(\sigma^2|\boldsymbol{y}, X)\, d(\sigma^2)$$

Therefore, we need to use computational methods to simulate the posterior marginal distribution for $\boldsymbol{\beta}$.

To make inference on individual parameters, we have that since $\boldsymbol{\beta}$ follows a multivariate normal distribution, we can also conclude that for any model parameter β_j, we get that

$$\beta_j|\sigma^2, \boldsymbol{y}, X \sim N(\mu_j, \sigma^2\Lambda_{jj})$$

where μ_j is the j-th element of the vector $\boldsymbol{\mu}$ and Λ_{jj} is the j-th diagonal element of the matrix Λ. Then the marginal distribution for β_j is

$$h(\beta_j|\boldsymbol{y}, X) = \int_{-\infty}^{\infty} h(\beta_j|\sigma^2, \boldsymbol{y}, X)h(\sigma^2|\boldsymbol{y}, X)\, d(\sigma^2)$$

This integral is also hard to compute, so we need to use computational methods.

Given the marginal distribution, we can use it to find the mean and variance of β_j and we can also construct credible regions to decide if the explanatory variable associated with the parameter is significant to the model. A $100(1-\alpha)\%$ credible region for a parameter β_j is the region I such that $P(\beta_j \in I|\text{Data}) = 1 - \alpha$. There are many different such regions, but our goal is to find the shortest credible region since we want a more accurate estimate. If the 95% credible region contains 0, then we can conclude that the parameter and its associated explanatory variable is not significant to the model. Otherwise, we conclude that it is significant.

Simulation Algorithm to Compute Posterior

To simulate the marginal distribution for a parameter β_j, we can use the following algorithm:

1. Load the data.
2. Initialize the chosen parameters $\boldsymbol{m}$, V, a, and b.
3. Compute $\boldsymbol{\mu}$, Λ, α, and r using the data and the initialized prior parameters.
4. Simulate σ^2 a large number of times from the $IG(\alpha, r)$ distribution.
5. For each simulated value of σ^2, simulate β_j from the $N(\mu_j, \sigma^2\Lambda_{jj})$ distribution.
6. Draw a graph for the simulated distribution of β_j, then compute its center and a 95% credible region.

Application

Now we have established the theory behind Bayesian linear regression, we will compare the Bayesian linear regression model to the ordinary least squares model on the Boston Housing dataset [citation].

Data Description and Preprocessing

The Boston Housing dataset contains information about various factors affecting housing prices in the Boston area. It includes features such as the per capita crime rate, average number of rooms per dwelling, proportion of residential land zoned for lots over 25,000 square feet, and more.

The variables of the data are as follows:

- CRIM: per capita crime rate by town
- ZN: proportion of residential land zoned for lots over 25,000 sq.ft.
- INDUS: proportion of non-retail business acres per town
- CHAS: Charles River dummy variable (1 if tract bounds river; 0 otherwise)
- NOX: nitric oxides concentration (parts per 10 million)
- RM: average number of rooms per dwelling
- AGE: proportion of owner-occupied units built prior to 1940
- DIS: weighted distances to five Boston employment centres
- RAD: index of accessibility to radial highways
- TAX: full-value property-tax rate per $10,000
- PTRATIO: pupil-teacher ratio by town
- B: $1000(Bk - 0.63)^2$ where Bk is the proportion of [people of African American descent] by town
- LSTAT: % lower status of the population
- MEDV: Median value of owner-occupied homes in $1000s (target variable)

Our goal is to construct a linear regression model that describes the relationship between the median value of owner-occupied homes and the factors that affect housing prices in the Boston area. We build a model using both the frequentist and the Bayesian framework. We are removing the variable CHAS because it is an indicator variable. We construct the models and then we compare them.

To make things easier to read, we will be using the following notations for variables:

- $X_1 : CRIM$
- $X_2 : ZN$
- $X_3 : INDUS$
- $X_4 : NOX$
- $X_5 : RM$
- $X_6 : AGE$
- $X_7 : DIS$
- $X_8 : RAD$
- $X_9 : TAX$
- $X_{10} : PRATIO$

- $X_{11} : B$
- $X_{12} : LSTAT$
- $Y : MEDV$

Dataset Source: The data were derived from information collected by the U.S. Census Service concerning housing in the area of Boston, Massachusetts.

We first divide the data into training and test sets randomly:

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'olsrr'
```

```
## The following object is masked from 'package:datasets':
##
##     rivers
```

```
set.seed(1)

boston_housing_dataset <- read.csv("Boston.csv")

#Remove the indicator variable for river
boston_housing_dataset <- subset(boston_housing_dataset, select = -CHAS)

train_set_indices <- createDataPartition(boston_housing_dataset$MEDV, p = 0.6, list = FALSE)

train_set_dataset <- boston_housing_dataset[train_set_indices, ]

test_set_dataset <- boston_housing_dataset[-train_set_indices, ]

#For use in future Python code
write.csv(train_set_dataset, "train_set_dataset.csv", row.names = FALSE)
write.csv(test_set_dataset, "test_set_dataset.csv", row.names = FALSE)
```

Ordinary Least Squares Model

We first fit a linear regression model under the frequentist framework. Then we make inference and evaluate the quality of the model. We consider 2 possible models: one using all predictors and one that is selected using stepwise selection.

We first fit the full model with all variables:

We have that the full frequentist linear regression model fitted using the training dataset is $Y = 32.044484 - 0.121517X_1 + 0.042545X_2 + 0.066926X_3 - 13.994681X_4 + 4.021977X_5 - 0.004531X_6 - 1.469667X_7 + 0.343192X_8 - 0.014404X_9 - 0.855679X_{10} + 0.011628X_{11} - 0.0629198X_{12}$, where the variables are as described in the previous section. From looking at the summary table of the model, we see that the variables X_3 (INDUS) and X_6 (AGE) are not statistically significant under the significance level $\alpha = 0.05$ because the p-values from the t tests are greater than 0.05.

```
full_model=lm(MEDV ~ .,data=train_set_dataset)

summary(full_model)
```

```
##
## Call:
```

```
## lm(formula = MEDV ~ ., data = train_set_dataset)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.0346  -2.6668  -0.7731   1.5473  25.7176
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  32.044484   6.856733   4.673 4.52e-06 ***
## CRIM         -0.121517   0.044156  -2.752 0.006292 **
## ZN            0.042545   0.017788   2.392 0.017396 *
## INDUS         0.066926   0.074910   0.893 0.372368
## NX          -13.994681   4.821217  -2.903 0.003980 **
## RM            4.021977   0.555961   7.234 4.11e-12 ***
## AGE          -0.004531   0.017408  -0.260 0.794815
## DIS          -1.469667   0.261205  -5.626 4.30e-08 ***
## RAD           0.343192   0.082127   4.179 3.87e-05 ***
## TAX          -0.014404   0.004416  -3.262 0.001238 **
## PTRATIO      -0.855679   0.181010  -4.727 3.54e-06 ***
## B             0.011628   0.003434   3.386 0.000806 ***
## LSTAT        -0.629198   0.067913  -9.265  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.757 on 293 degrees of freedom
## Multiple R-squared:  0.7304, Adjusted R-squared:  0.7193
## F-statistic: 66.14 on 12 and 293 DF,  p-value: < 2.2e-16
```

```
RSS <- residuals(full_model)^2

training_MSE <- mean(RSS)

cat("Training MSE:", training_MSE, "\n")
```

```
## Training MSE: 21.66596
```

```
test_MSE <- mean((test_set_dataset$MEDV - predict(full_model, newdata=test_set_dataset))^2)

cat("Test MSE:", test_MSE)
```

```
## Test MSE: 24.70457
```

We then evaluated the model using both the training dataset and the test dataset. The training error is 21.66596 and the test error is 24.70547.

Now, we build a reduced linear regression model using stepwise selection:

```
stepwise_model_results <- ols_step_both_p(full_model, details = FALSE, p_enter = 0.10, p_remove = 0.15)

stepwise_model <- lm(MEDV ~ CRIM + ZN + NX + RM + DIS + RAD + TAX + PTRATIO + B + LSTAT,
                     data = train_set_dataset)

summary(stepwise_model)
```

```
##
## Call:
## lm(formula = MEDV ~ CRIM + ZN + NX + RM + DIS + RAD + TAX + PTRATIO +
```

```
##     B + LSTAT, data = train_set_dataset)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -10.9539  -2.7252  -0.7731   1.5728  25.7119
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  31.664864   6.811394   4.649 5.04e-06 ***
## CRIM         -0.123388   0.044025  -2.803 0.005404 **
## ZN            0.042781   0.017643   2.425 0.015918 *
## NX          -13.171578   4.457736  -2.955 0.003381 **
## RM            3.951521   0.540978   7.304 2.61e-12 ***
## DIS          -1.507618   0.235212  -6.410 5.77e-10 ***
## RAD           0.325008   0.079054   4.111 5.11e-05 ***
## TAX          -0.012985   0.004112  -3.158 0.001753 **
## PTRATIO      -0.823535   0.176185  -4.674 4.49e-06 ***
## B             0.011428   0.003418   3.344 0.000934 ***
## LSTAT        -0.626362   0.062823  -9.970  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.748 on 295 degrees of freedom
## Multiple R-squared:  0.7296, Adjusted R-squared:  0.7204
## F-statistic: 79.59 on 10 and 295 DF,  p-value: < 2.2e-16
```

```
RSS <- residuals(stepwise_model)^2

training_MSE <- mean(RSS)

cat("Training MSE:", training_MSE, "\n")
```

```
## Training MSE: 21.73032
```

```
test_MSE <- mean((test_set_dataset$MEDV - predict(stepwise_model, newdata=test_set_dataset))^2)

cat("Test MSE:", test_MSE)
```

```
## Test MSE: 24.52705
```

We have that the reduced model selected via stepwise regression is $Y = 31.665 - 0.0.123X_1 + 0.043X_2 - 13.171578X_4 + 3.951521X_5 - 1.507618X_7 + 0.325008X_8 - 0.012985X_9 - 0.823535X_{10} + 0.011428X_{11} - 0.626362X_{12}$. All predictors are statistically significant in this model based on the t tests.

After evaluating the model, we obtain that the training error is 21.73032 and the test error is 24.52705. This is similar to the full model, indicating that the reduced model explains as well as the full model the median value of homes in Boston. If we look at the R^2_{adj} values for the reduced vs full model, we can see that the value for the full model is 0.7193 and the value for the reduced model is 72%. This indicates that the reduced model, when penalizing for the increase in parameters in the full model, explains the model slightly worse than the reduced model.

Bayesian Linear Regression Model

Setting Prior Distribution Parameters

We must now choose the prior parameters for each of the prior distributions.

For setting the parameters for the joint prior for the β coefficients (given σ^2), we will set $\boldsymbol{m} = 0$ (mean of each

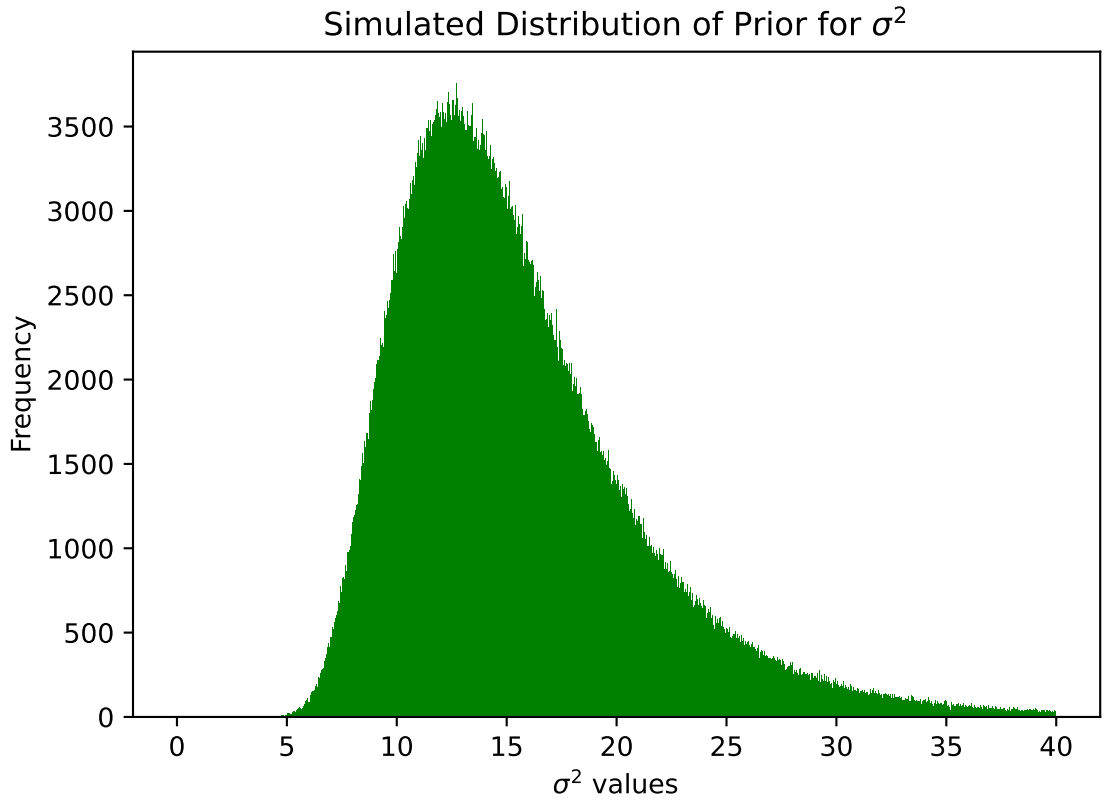
beta coefficient is set to 0) and V to $25I$. Now, when we set the means to 0 and V^{-1} to λI (in other words $V = \frac{1}{\lambda} I$) ($\lambda = \frac{1}{25}$ in this case), where I is the identity matrix and $\lambda \in \mathbb{R}$, we can see that the posterior best estimates under squared error loss for the β parameters are the same as in the frequentist method of ridge regression. The larger λ is, the more shrinking of non-significant estimates occurs towards 0. The purpose of this is to attempt to not overfit the training data while being able to more easily generalize to unseen data by not capturing as much noise and potentially unexplanatory predictors. We set $\lambda = \frac{1}{25}$ to balance the potential for overfitting versus underfitting the model (too large a λ value could lead to underfitting).

For the prior for σ^2, we used the $IG(a, b)$ prior with $a = 9$ and $b = 125$. We chose the values for a and b because we want the mean and mode of the distribution to closely match a value for σ^2 such that most of the data we collected for the response variable (MEDV) are within 2 standard deviations from the mean. We also made sure that this distribution does not have too low or high variability to include the possibility that the true σ^2 is way different than what we thought (we are not too confident in our prior). Now, most of the data should be within the range of 4σ. The values for MEDV are between 0 and 50, so we want to have the peak of the distribution to be around 12.5. As can be seen below in the plot and statistics for this prior distribution, the variance of this distribution is around 21, the median is 12.5, and the mean is around 15.6.

```
#Set the random seed to make the code reproducible
random.seed(1)

#Load the training and test data that has been preprocessed
train_set = pd.read_csv('train_set_dataset.csv', header=0, sep=',')
test_set = pd.read_csv('test_set_dataset.csv', header=0, sep=',')

num_predictors = len(train_set.columns)

#Set the parameters for the prior distribution for sigma2 (inverse gamma) (non-informative)
a = 9
b = 125

#Set the parameters for the prior distribution for mu (normal) (non-informative)
V = np.diag([25] * num_predictors)
m = np.zeros(num_predictors)

#Plot of prior for sigma^2 and generate statistics for it
sigma2 = 1 / np.random.gamma(shape=a, scale=1/b, size=1000000)
mode = b/(a+1)
mean = b/(a-1)
variance = (b^2)/((a-1)^2*(a-2))
print(f"Statistics for sigma^2 prior \n Mode: {mode}, Mean: {mean}, Variance: {variance}")
```

```
## Statistics for sigma^2 prior
##  Mode: 12.5, Mean: 15.625, Variance: 21.166666666666668
```

```
plt.hist(sigma2, bins=1000, alpha=1, color='green', range=(0, 40))
plt.title('Simulated Distribution of Prior for $\\sigma^2$')
plt.xlabel('$\\sigma^2$ values')
plt.ylabel('Frequency')
plt.show()
```



Algorithm

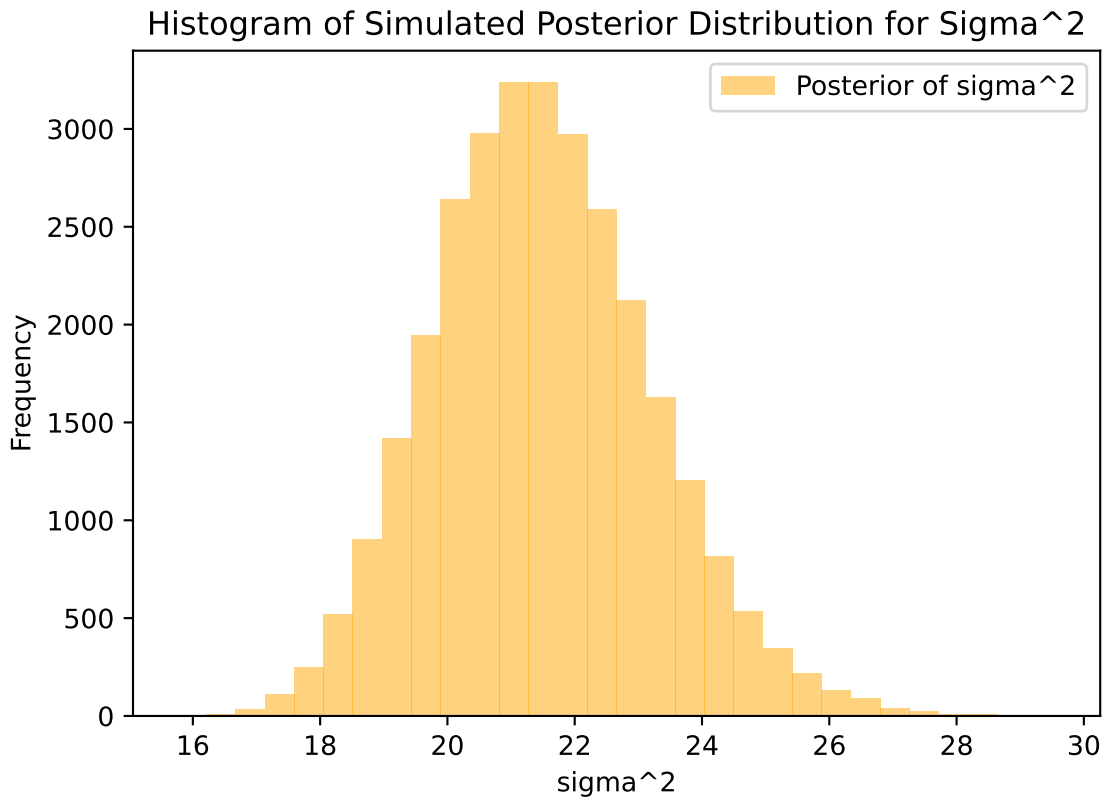
We now execute the algorithm for finding the posterior distribution for σ^2 and the marginal posterior distributions for $\beta_i, i \in \{1, \ldots, 12\}$, with the i^{th} β corresponding to the i^{th} predictor (X_i) (β_0/X_0 is intercept term).

```
train_set.insert(0, 'Intercept', 1)
test_set.insert(0, 'Intercept', 1)

X_training_set = train_set.iloc[:, 0:num_predictors].to_numpy()
y_training_set = train_set.iloc[:, num_predictors].to_numpy()

X_test_set = test_set.iloc[:, 0:num_predictors].to_numpy()
y_test_set = test_set.iloc[:, num_predictors].to_numpy()

#Set sampling size
sampling_size = 30000

#Compute the inverse of V
V_inv = np.linalg.inv(V)

#Compute the mean of the posterior distribution of beta given sigma2 (normal)
mu_post = np.linalg.inv(X_training_set.T @ X_training_set + V_inv) @ (X_training_set.T @ y_training_set

#Compute the lambda matrix of the posterior distribution of beta given sigma2 (normal)
lambda_post = np.linalg.inv(X_training_set.T @ X_training_set + V_inv)
```

```
#Compute alpha paramater of sigma2 posterior distribution (inverse gamma)
alpha_post = a + len(y_training_set) / 2
r_post = (m.T @ V_inv @ m - mu_post.T @ np.linalg.inv(lambda_post) @ mu_post + 2*b + y_training_set.T @

# Draw sigma2 from the inverse gamma distribution
sigma2 = 1 / np.random.gamma(shape=alpha_post, scale=1/r_post, size=sampling_size)

beta_hat = np.zeros((sampling_size, num_predictors))

#Sample the joint posterior distributions for the Beta coefficients
for i in range(sampling_size):
    beta_hat[i, :] = np.random.multivariate_normal(mu_post, lambda_post * sigma2[i])
```

Plots

```
#Plot for posterior distribution for sigma^2
plt.hist(sigma2, bins=int(sampling_size/1000), alpha=0.5, color='orange', label='Posterior of sigma^2')
plt.xlabel('sigma^2')
plt.ylabel('Frequency')
plt.title('Histogram of Simulated Posterior Distribution for Sigma^2')
plt.legend()
plt.show()
```




```
colors = LinearSegmentedColormap.from_list("gradient", ["red", "purple", "blue"], N=num_predictors)

beta_estimates = pd.DataFrame(index=train_set.columns[0:num_predictors], columns=["Beta Estimate", "SD
```

```
fig, axes = plt.subplots(5, 3, figsize=(35, 43))
axes = axes.flatten()

#Create the graphs
for i in range(num_predictors):
    sorted_beta_hat_i = np.sort(beta_hat[:, i])
    value_at_975 = round(sorted_beta_hat_i[int(np.floor(0.975 * sampling_size))], 8)
    value_at_25 = round(sorted_beta_hat_i[int(np.floor(0.025 * sampling_size))], 8)

    beta_estimates.iloc[i, 0] = mu_post[i]
    beta_estimates.iloc[i, 1] = lambda_post[i][i] * sigma2[i]
    beta_estimates.iloc[i, 2] = f"[{value_at_25}, {value_at_975}]"

    subaxis = axes[i]
    n, bins, patches = subaxis.hist(beta_hat[:, i], bins=int(sampling_size / 1000), color=colors(i), al
    low_cr = subaxis.axvline(x=value_at_25, color='r', linestyle='--', linewidth=5, label = 'Lower 95% C
    high_cr = subaxis.axvline(x=value_at_975, color='darkred', linestyle='--',linewidth=5, label = 'Upp
    mean_est = subaxis.axvline(x=mu_post[i], color='c', linestyle='--', linewidth=5, label = 'Posterior

    subaxis.set_xlabel('$\\beta_{' + str(i) + '}$ Value', fontsize=25)
    #subaxis.set_xlabel('$\\beta$' + str(i) + ' Value', fontsize=25)
    subaxis.set_ylabel('Frequency', fontsize=25)
    subaxis.set_title('$X_{' + str(i) + '}$ Value (' + train_set.columns[i] + ')', fontsize=3)
    #subaxis.set_title('$X$ ' + str(i) + ' Value (' + train_set.columns[i] + ')', fontsize=3)
    subaxis.tick_params(axis='both', which='major', labelsize=20)

lines_labels = [ax.get_legend_handles_labels() for ax in fig.axes][1:2]
lines, labels = [sum(lol, []) for lol in zip(*lines_labels)]
fig.legend(lines, labels, loc='upper center', ncol=3, fontsize=35, bbox_to_anchor=(0.5, 0.94))

fig.suptitle('Simulated Marginal Posterior Distributions and Credible Regions for Beta Coefficients of M

plt.tight_layout(rect=[0, 0, 1, 0.92])

for i in range(num_predictors, 15):
    axes[i].axis('off')
```

```
## (0.0, 1.0, 0.0, 1.0)
## (0.0, 1.0, 0.0, 1.0)
```

```
plt.show()
```

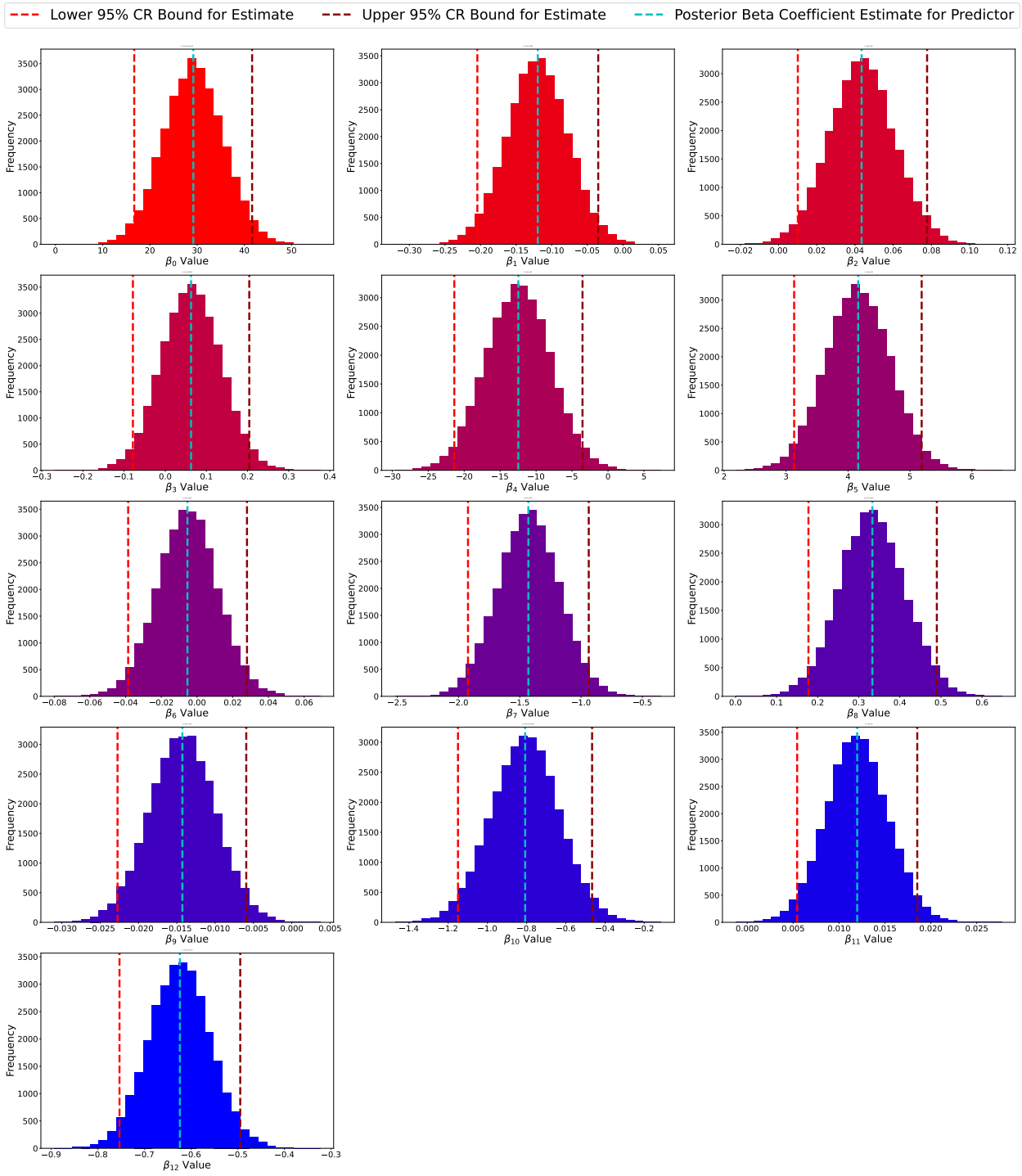


Figure 1: Simulated Bayesian Marginal Posterior Distributions and Credible Regions for Beta Coefficients of Predictors

Results

```
## 1275
```

	Beta Estimate	SD Estimate for Beta	95% Credible Region
Intercept	29.2027	43.9389	[16.68464302, 41.66225169]
CRIM	-0.119609	0.00195964	[-0.2045047, -0.03471359]
ZN	0.0434526	0.000308539	[0.01005158, 0.07773268]
INDUS	0.0626662	0.00513683	[-0.07868531, 0.20418256]
NX	-12.4738	24.7635	[-21.37486016, -3.54826191]
RM	4.16421	0.254059	[3.13044058, 5.18901402]
AGE	-0.00535482	0.000278523	[-0.03856884, 0.0280109]
DIS	-1.43013	0.0629495	[-1.92284279, -0.93704818]
RAD	0.333189	0.00616361	[0.17754902, 0.49014988]
TAX	-0.0142721	1.87485e-05	[-0.02273553, -0.0059376]
PTRATIO	-0.807487	0.0295909	[-1.15015037, -0.46605029]
B	0.0119776	1.0398e-05	[0.00543713, 0.0185228]
LSTAT	-0.624253	0.00415207	[-0.75367921, -0.49485802]

Table 1: Bayesian Posterior Beta Coefficients Estimates Summary

Write here

```
#Compute MSE
training_MSE = np.mean((y_training_set - X_training_set @ mu_post)**2)

print(f"Training Set Bayesian MSE: {training_MSE}")
```

```
## Training Set Bayesian MSE: 21.679226432779082
```

```
test_MSE = np.mean((y_test_set - X_test_set @ mu_post)**2)

print(f"Test Set Bayesian MSE: {test_MSE}")
```

```
## Test Set Bayesian MSE: 24.866295863386487
```

We can see that the training MSE of our Bayesian model on a training set is around 20, while the test MSE is much higher at around 28. This could indicate slight overfitting in the training data (as we did not remove non-significant predictors in our final model).

Comparison Between Models

We will now compare the two methods based on their estimates for the β coefficients and σ^2:

```
## 1582
```

```
## 1021
```

```
## 1303
```

	OLS Beta Estimate	Bayesian Beta Estimate	Beta Estimate Difference (OLS - Bayesian)
Intercept	32.0445	29.2027	-2.84174
CRIM	-0.121517	-0.119609	0.00190815
ZN	0.0425453	0.0434526	0.000907318
INDUS	0.066926	0.0626662	-0.00425973
NX	-13.9947	-12.4738	1.52086
RM	4.02198	4.16421	0.14223
AGE	-0.00453122	-0.00535482	-0.000823595
DIS	-1.46967	-1.43013	0.0395355
RAD	0.343192	0.333189	-0.0100028
TAX	-0.0144041	-0.0142721	0.000131971
PTRATIO	-0.855679	-0.807487	0.0481923
B	0.0116279	0.0119776	0.000349729
LSTAT	-0.629198	-0.624253	0.00494478

Table 2: β Coefficient Estimates Comparison between OLS and Bayesian method

We used a prior with $\boldsymbol{\beta}|\sigma^2 \sim N_k(\boldsymbol{m}, \sigma^2 V)$ and $\sigma^2 \sim IG(a, b)$, where $\boldsymbol{m} = \mathbf{0}$, $V = 25I$ (I is the identity matrix), $a = 0.001$, and $b = 0.001$. We used this prior to simulate the posterior distribution for σ^2 and for each of the β_j coefficients of the linear regression model. Above are the estimates and 95% credible regions of each parameter.

	OLS Beta SD Estimate	Bayesian Beta SD Estimate
Intercept	6.85673	43.9389
CRIM	0.0441557	0.00195964
ZN	0.0177879	0.000308539
INDUS	0.0749102	0.00513683
NX	4.82122	24.7635
RM	0.555961	0.254059
AGE	0.0174077	0.000278523
DIS	0.261205	0.0629495
RAD	0.0821271	0.00616361
TAX	0.00441613	1.87485e-05
PTRATIO	0.18101	0.0295909
B	0.00343399	1.0398e-05
LSTAT	0.0679132	0.00415207

Table 3: β Coefficient Standard Error/Deviation Estimates Comparison between OLS and Bayesian method

We can see that the Bayesian estimates for the β coefficients are not much different compared to the ordinary least squares estimates. From looking at table 2, the last column shows the difference between the Bayesian and ordinary least squares estimates. The magnitude of the differences are all not very high (less than 1). This shows that the estimated linear regression models under both the ordinary least squares and the Bayesian framework are very similar. However, if we look at the estimates of the standard deviation of the parameters, some of them are very similar under both the ordinary least squares and Bayesian framework, and some are quite different (notably for the intercept and for the predictor NX).

	Bayesian 95% Beta Credible Region	OLS 95% Beta Confidence Interval
Intercept	[16.68464302, 41.66225169]	[18.549792, 45.539176]
CRIM	[-0.2045047, -0.03471359]	[-0.20842, -0.034615]
ZN	[0.01005158, 0.07773268]	[0.007537, 0.077554]
INDUS	[-0.07868531, 0.20418256]	[-0.080504, 0.214356]
NX	[-21.37486016, -3.54826191]	[-23.483286, -4.506077]
RM	[3.13044058, 5.18901402]	[2.927794, 5.11616]
AGE	[-0.03856884, 0.0280109]	[-0.038791, 0.029729]
DIS	[-1.92284279, -0.93704818]	[-1.983743, -0.955591]
RAD	[0.17754902, 0.49014988]	[0.181558, 0.504825]
TAX	[-0.02273553, -0.0059376]	[-0.023095, -0.005713]
PTRATIO	[-1.15015037, -0.46605029]	[-1.211924, -0.499434]
B	[0.00543713, 0.0185228]	[0.004869, 0.018386]
LSTAT	[-0.75367921, -0.49485802]	[-0.762857, -0.495538]

Table 4: β Coefficient Estimates 95

Table 4 above compares the 95% confidence intervals of the parameters under the ordinary least squares framework and the 95% credible regions of the parameters under the Bayesian framework. From looking at the table, the bounds of the credible regions are not much different compared to the bounds of the confidence intervals. We can also see that for the predictors X_3 (INDUS) and X_6 (AGE), the 95% confidence intervals and credible regions for the parameters both contain 0. This means that under both frameworks, we can conclude that the predictors X_3 and X_6 are not statistically significant to the model. Overall, the two methods match pretty closely in terms of confidence intervals.

```
#Get the MSE version that adjusts for the number of parameters in the model
training_MSE_adj <- (training_MSE*nrow(train_set_dataset))/(nrow(train_set_dataset)-length(coef(full_mo
cat("OLS sigma^2 best estimate:", training_MSE_adj)
```

```
## OLS sigma^2 best estimate: 22.69446
```

```
#Plot of prior for sigma^2
sigma2 = 1 / np.random.gamma(shape=alpha_post, scale=1/r_post, size=sampling_size)
mode = r_post/(alpha_post+1)
mean = r_post/(alpha_post-1)
variance = (r_post**2)/((alpha_post-1)**2 *(alpha_post-2))
print(f"Statistics for sigma^2 prior \n Mode: {mode}, Mean: {mean}, Variance: {variance}")
```

```
## Statistics for sigma^2 prior
##  Mode: 21.24233588589226, Mean: 21.506215834785333, Variance: 2.890733247077321
```

Compare the estimates for σ^2 between OLS and Bayesian methods here.

Conclusion/Future Steps

All in all, we explored the differences between Ordinary Least Squares Regression (Frequentist Method) and the Bayesian method of settings priors to find posterior distributions for the model parameters. We see that using the prior we picked for the Bayesian model, the estimates for the parameters are similar for both the ordinary least squares model and the Bayesian linear regression model. With the Bayesian model, we could further explore prediction of new observations, hypothesis testing of multiple parameters, and transformations.

References

Note: These are not cited in proper format yet.

https://en.wikipedia.org/wiki/Bayesian_linear_regression

https://gregorygundersen.com/blog/2020/02/04/bayesian-linear-regression/

https://www.researchgate.net/publication/333917874_Bayesian_Linear_Regression#pf18