# 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 with each observation, $n$ is the number of observations, and $k$ is the number of parameters. The goal is to estimate $\boldsymbol{\beta}$ and then construct and evaluate the model from the estimated values. In other words, we have

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

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

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

assume that the random errors have mean 0, variance $\sigma^2$, are uncorrelated, and they 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:

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

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 $\sigma^2$ 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

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

We can show that

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

Therefore, an unbiased estimator for $\sigma^2$ 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 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 $\sigma^2$, 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 $\sigma^2$, which is $MS_{res}$. However, if we replace $\sigma^2$ 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 $\alpha$ 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 it is less than $\alpha$.

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

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

From the confidence interval, we can reject $H_0$ if $\beta_{j_0}$ is contained in the interval. Otherwise, we accept $H_0$.

### ANOVA tests

There's another way to test whether predictors are significant to the model, but instead of testing individual parameters only, we can test a subset of all parameters. This is known as the ANOVA test.

## Prediction

The purpose of constructing a model between a response variable and several explanatory variables is to make predictions of future values of the response given certain values of the explanatory variables. To predict $\boldsymbol{y}$ at a given $X$, we simply use the fitted model

$$\hat{\boldsymbol{y}} = X\hat{\boldsymbol{\beta}}$$

To construct a confidence interval for a prediction, we have that

# 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 $\sigma^2$.

### 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 $\sigma^2$, we treat them as random variables and we make a prior assumption on their distribution. The assumption 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:

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

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 $\Lambda$ 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

$$\boldsymbol{\beta} | \sigma^2, \boldsymbol{y}, X \sim N_k(\boldsymbol{\mu}, \sigma^2 \Lambda^{-1})$$

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 $\beta_j$, we get that

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

where $\mu_j$ is the j-th element of the vector $\boldsymbol{\mu}$ and $\Lambda_{jj}$ is the j-th diagonal element of the matrix $\Lambda$. Then the marginal distribution for $\beta_j$ 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 $\beta_j$ 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 $\beta_j$ 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 credible region contain 0, this means that there is a $1-\alpha$ probability that 0 is contained in the region. If $1-\alpha$ is very large, 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 $\beta_j$, we can use the following algorithm:

1. Load the data.
2. Initialize the chosen parameters $\boldsymbol{m}$, $V$, $a$, and $b$.
3. Compute $\boldsymbol{\mu}$, $\Lambda$, $\alpha$, and $r$ using the data and the initialized prior parameters.
4. Simulate $\sigma^2$ a large number of times from the $IG(\alpha, r)$ distribution.
5. For each simulated value of $\sigma^2$, simulate $\beta_j$ from the $N(\mu_j, \sigma^2\Lambda_{jj})$ distribution.
6. Draw a graph for the simulated distribution of $\beta_j$, 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.

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:

```
library(caret)
```

```
## Loading required package: ggplot2

## Loading required package: lattice
```

```
library(olsrr)
```

```
##
## 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, ]
```

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

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

anova(full_model)
```

```
## Analysis of Variance Table
##
## Response: MEDV
##            Df Sum Sq Mean Sq  F value    Pr(>F)
## CRIM        1 3620.7  3620.7 160.0139 < 2.2e-16 ***
## ZN          1 2510.3  2510.3 110.9414 < 2.2e-16 ***
## INDUS       1 1666.5  1666.5  73.6500 5.549e-16 ***
## NX          1   13.8    13.8   0.6099 0.4354431
## RM          1 5761.3  5761.3 254.6178 < 2.2e-16 ***
## AGE         1   70.6    70.6   3.1203 0.0783650 .
## DIS         1 1082.9  1082.9  47.8604 2.881e-11 ***
## RAD         1    5.0     5.0   0.2193 0.6399534
## TAX         1  291.6   291.6  12.8882 0.0003875 ***
## PTRATIO     1  604.9   604.9  26.7329 4.334e-07 ***
## B           1  390.2   390.2  17.2463 4.311e-05 ***
## LSTAT       1 1942.2  1942.2  85.8354 < 2.2e-16 ***
## Residuals 293 6629.8    22.6
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
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
```

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

training_MSE <- mean(RSS)

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

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

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

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

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

```r
#Save model to csv file
model_summary <- summary(full_model)$coefficients  # Extracting coefficients, std. error, t-value, and

# Extracting 95% confidence intervals
conf_intervals <- confint(full_model, level = 0.95)

# Creating a data frame
results_df <- data.frame(
  Estimate = model_summary[, "Estimate"],
  StdError = model_summary[, "Std. Error"],
  Lower95CI = conf_intervals[, "2.5 %"],
  Upper95CI = conf_intervals[, "97.5 %"]
)

# Save to CSV
write.csv(results_df, "full_model_summary.csv", row.names = TRUE)
```

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. From the ANOVA table, we can see that given the variables $X_1$ (CRIM), $X_2$ (ZN), and $X_3$ (INDUS), we have that adding $X_4$ (NX) is not significant to the model. We also have that given the variables $X_1$ (CRIM), $X_2$ (ZN), $X_3$ (INDUS), $X_4$ (NX), and $X_5$ (RM), $X_6$ (AGE) is not significant. We also have that given the variables $X_1$ (CRIM), $X_2$ (ZN), $X_3$ (INDUS), $X_4$ (NX), $X_5$ (RM), $X_6$ (AGE), and $X_7$ (DIS), $X_8$ (RAD) is not significant to the model.

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:

```r
#print(summary(model))

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

print(stepwise_model_results)
```

```
##
##
##                                Stepwise Summary
## -----------------------------------------------------------------------------
## Step    Variable         AIC         SBC         SBIC        R2        Adj. R2
## -----------------------------------------------------------------------------
##  0      Base Model     2214.661    2222.108    1343.801    0.00000    0.00000
##  1      LSTAT (+)      1976.856    1988.027    1106.638    0.54328    0.54177
##  2      RM (+)         1906.965    1921.859    1037.156    0.63890    0.63652
##  3      PTRATIO (+)    1886.200    1904.818    1016.479    0.66479    0.66146
##  4      DIS (+)        1874.171    1896.512    1004.546    0.67981    0.67556
##  5      B (+)          1858.067    1884.132     988.864    0.69821    0.69318
##  6      NX (+)         1850.248    1880.037     981.343    0.70774    0.70187
##  7      RAD (+)        1848.473    1881.986     979.688    0.71132    0.70454
##  8      TAX (+)        1843.509    1880.745     975.080    0.71782    0.71022
##  9      CRIM (+)       1838.514    1879.473     970.541    0.72419    0.71581
##  10     ZN (+)         1834.475    1879.158     966.989    0.72958    0.72042
## -----------------------------------------------------------------------------
##
## Final Model Output
## ------------------
##
##                          Model Summary
## ----------------------------------------------------------------
## R                        0.854       RMSE                 4.662
## R-Squared                0.730       MSE                 22.541
## Adj. R-Squared           0.720       Coef. Var           21.046
## Pred R-Squared           0.701       AIC               1834.475
## MAE                      3.208       SBC               1879.158
## ----------------------------------------------------------------
##  RMSE: Root Mean Square Error
##  MSE: Mean Square Error
##  MAE: Mean Absolute Error
##  AIC: Akaike Information Criteria
##  SBC: Schwarz Bayesian Criteria
##
##                                   ANOVA
## -----------------------------------------------------------------------
##                  Sum of
##                 Squares         DF    Mean Square      F         Sig.
## -----------------------------------------------------------------------
## Regression    17940.367         10       1794.037    79.591    0.0000
## Residual       6649.478        295         22.541
## Total         24589.845        305
## -----------------------------------------------------------------------
##
##                                    Parameter Estimates
## ------------------------------------------------------------------------------------------
##       model        Beta    Std. Error    Std. Beta       t        Sig        lower     upper
## ------------------------------------------------------------------------------------------
## (Intercept)     31.665         6.811                  4.649    0.000       18.260    45.070
##       LSTAT     -0.626         0.063       -0.457    -9.970    0.000       -0.750    -0.503
##          RM      3.952         0.541        0.288     7.304    0.000        2.887     5.016
##     PTRATIO     -0.824         0.176       -0.198    -4.674    0.000       -1.170    -0.477
##         DIS     -1.508         0.235       -0.348    -6.410    0.000       -1.971    -1.045
##           B      0.011         0.003        0.119     3.344    0.001        0.005     0.018
##          NX    -13.172         4.458       -0.174    -2.955    0.003      -21.945    -4.399
##         RAD      0.325         0.079        0.315     4.111    0.000        0.169     0.481
##         TAX     -0.013         0.004       -0.244    -3.158    0.002       -0.021    -0.005
##        CRIM     -0.123         0.044       -0.116    -2.803    0.005       -0.210    -0.037
##          ZN      0.043         0.018        0.110     2.425    0.016        0.008     0.078
## ------------------------------------------------------------------------------------------
```

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

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

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.

```
#anova_comparison <- anova(model, stepwise_model$model)
anova_comparison <- anova(stepwise_model, full_model)
print(anova_comparison)
```

```
## Analysis of Variance Table
##
## Model 1: MEDV ~ CRIM + ZN + NX + RM + DIS + RAD + TAX + PTRATIO + B +
##     LSTAT
## Model 2: MEDV ~ CRIM + ZN + INDUS + NX + RM + AGE + DIS + RAD + TAX +
##     PTRATIO + B + LSTAT
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1    295 6649.5
## 2    293 6629.8  2    19.693 0.4352 0.6476
```

```
#anova(stepwise_model, full_model)
```

(overall we can see that the full model with the F test (explain further) shows that the full model with all of the predictors versus the reduced model with AGE (proportion of owner-occupied units built prior to 1940) and INDUS (proportion of non-retail business acres per town))

## Bayesian Linear Regression Model

### Setting Prior Distribution Parameters

We must now choose the prior parameters for each of the prior distributions. We start with setting the parameters for \$\sigma$^2$ \sim IG(a,b)\$. As we are not experts in the field of predicting home prices, we do not know the specifics of the variability of home pr
$\sigma^2$ to a non-informative prior. We decided to choose the $IG(1,1)$ prior for $\sigma^2$ as we believe that $\sigma^2$ is mostly likely small due to the high quality source of the data, but there might be ....

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
from matplotlib.colors import LinearSegmentedColormap
from tabulate import tabulate
import random

#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=',')
```

```python
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 = 0.001
b = 0.001

#Set the parameters for the prior distribution for mu (normal) (non-informative)

V = np.diag([1] * num_predictors)
m = np.zeros(num_predictors)

# m = np.array([100 for i in range(num_predictors)])
```

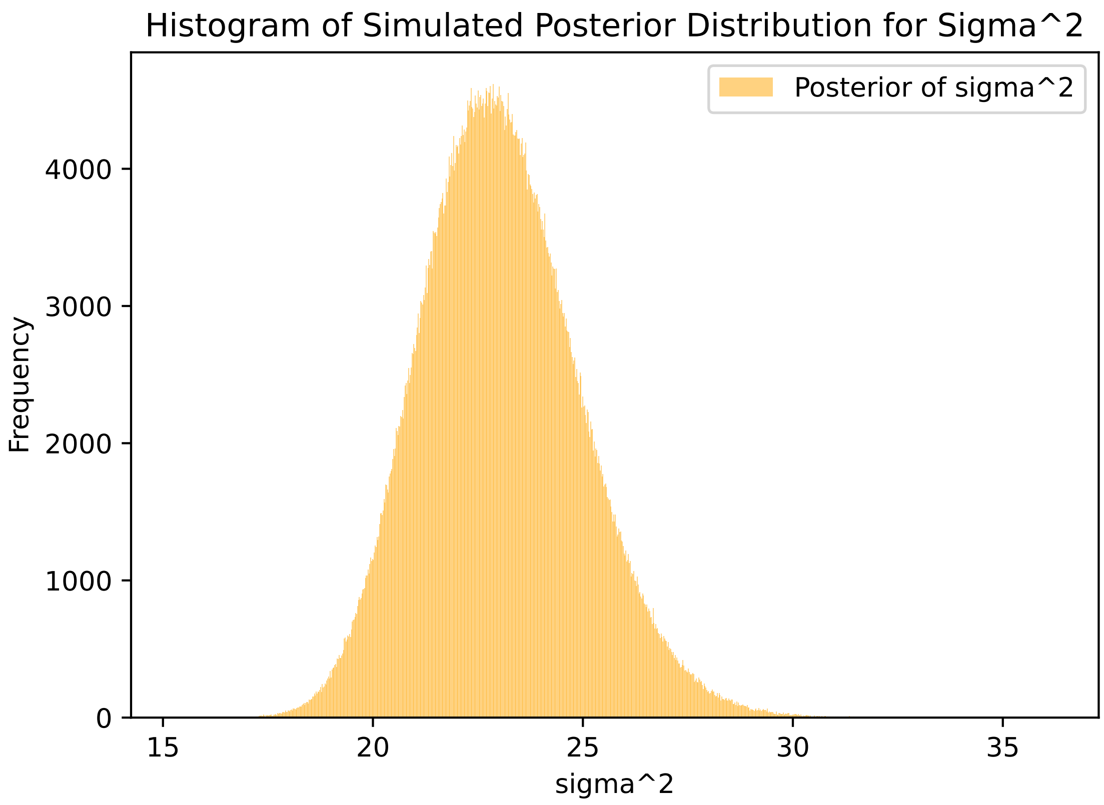
**Algorithm**

We now execute the algorithm for finding the posterior distribution for $\sigma^2$ and the marginal posterior distributions for $\beta_i, i \in \{0, 1, \ldots, 13\}$, with the $i^{th}$ $\beta$ corresponding to the $i^{th}$ predictor $(X_i)$.

```python
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 = 1000000

#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 @

# Step 4: Drawing sigma2 from the inverse gamma distribution
sigma2 = 1 / np.random.gamma(shape=alpha_post, scale=1/r_post, size=sampling_size)

#Create the vector for beta_hat
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**

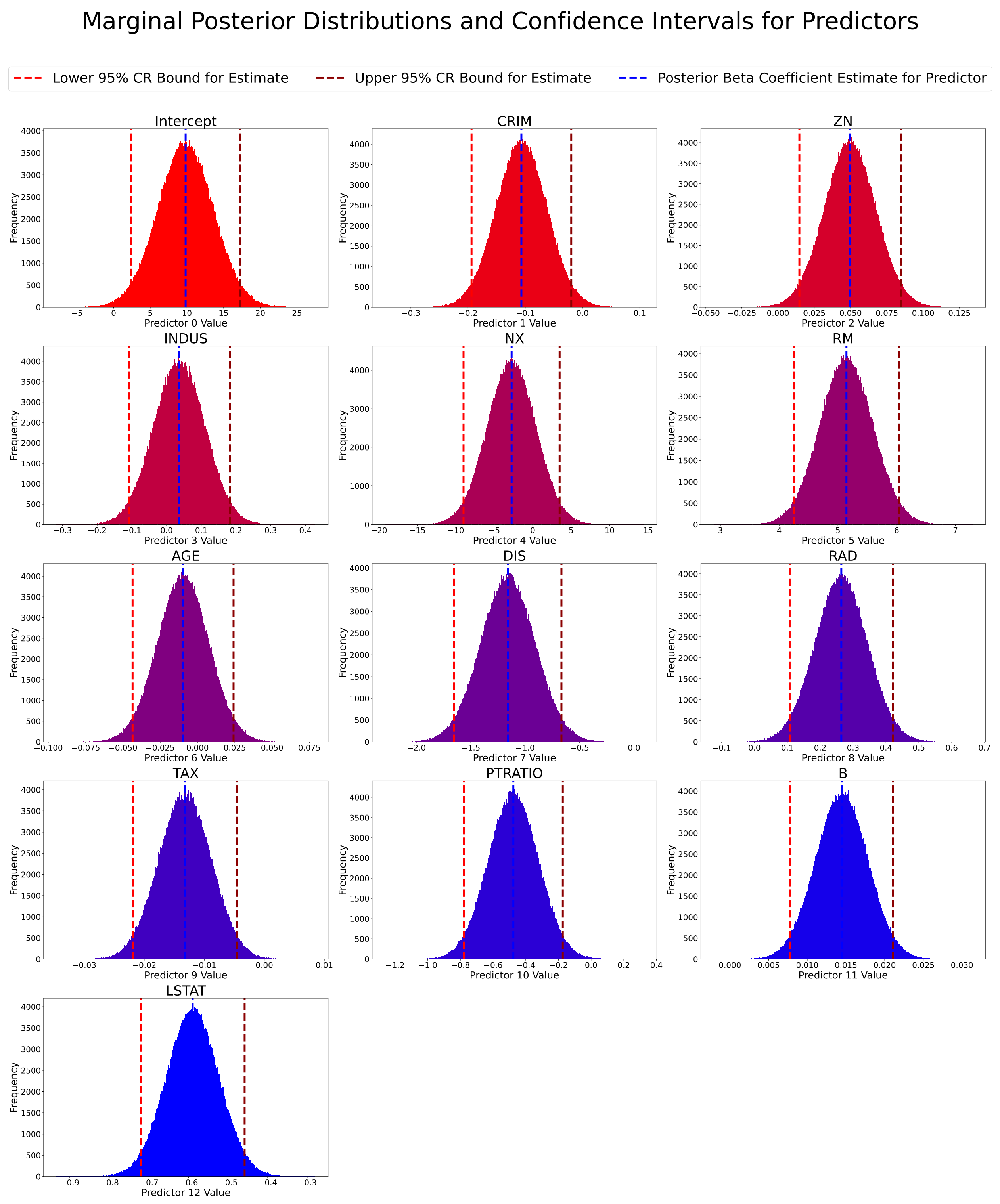
```
#Plot for posterior distribution for sigma^2
plt.hist(sigma2, bins=int(sampling_size/1000), alpha=0.5, color='orange', label='Posterior of sigma^2')
plt.xlabel('sigma^2')
plt.ylabel('Frequency')
plt.title('Histogram of Simulated Posterior Distribution for Sigma^2')
plt.legend()
plt.show()
```



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

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 = 'Uppe
    mean_est = subaxis.axvline(x=mu_post[i], color='b', linestyle='--', linewidth=5, label = 'Posterior

    subaxis.set_xlabel(f'Predictor {i} Value', fontsize = 25)
    subaxis.set_ylabel('Frequency', fontsize=25)
    subaxis.set_title(f'{train_set.columns[i]}', fontsize=35)
    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('Marginal Posterior Distributions and Confidence Intervals for Predictors', fontsize=60, y

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)
```

```
# fig.text(0.5, 0.01, 'Figure 1', ha='center', va='bottom', fontsize=20)

plt.show()
```



```
print(beta_estimates)
```

```
##              Beta Estimate SD Estimate for Beta          95% Credible Region
## Intercept         9.819143            14.669005   [2.35275568, 17.28077673]
## CRIM             -0.10678              0.001963   [-0.19392261, -0.0196579]
## ZN               0.049665            0.000314    [0.01474696, 0.08461949]
## INDUS            0.036922            0.005489   [-0.10827988, 0.18216155]
## NX                -2.7444            10.41378    [-8.9962755, 3.49911258]
## RM               5.144975             0.18651    [4.25408753, 6.03837272]
## AGE              -0.00966            0.000273   [-0.04352554, 0.02416039]
## DIS             -1.158618            0.062075  [-1.65238738, -0.66692825]
## RAD              0.264289            0.006908    [0.10703723, 0.42174891]
## TAX             -0.013212             0.00002  [-0.02186968, -0.00456208]
## PTRATIO         -0.475741            0.022533  [-0.77867611, -0.17343679]
## B                0.014443            0.000011    [0.00781772, 0.02108903]
## LSTAT            -0.58945            0.005079  [-0.72078793, -0.45820058]
```

Figure 1: Bayesian Posterior Distribution Graphs

**Results**

```
bayesian_results = tabulate(beta_estimates, headers='keys', tablefmt='latex',  numalign="left", straligr

with open('bayesian_results.tex', 'w') as f:
    f.write(bayesian_results)
```

```
## 1261
```

| | Beta Estimate | SD Estimate for Beta | 95% Credible Region |
|---|---|---|---|
| Intercept | 9.81914 | 14.669 | [2.35275568, 17.28077673] |
| CRIM | -0.10678 | 0.00196314 | [-0.19392261, -0.0196579] |
| ZN | 0.0496652 | 0.000314251 | [0.01474696, 0.08461949] |
| INDUS | 0.0369218 | 0.00548907 | [-0.10827988, 0.18216155] |
| NX | -2.7444 | 10.4138 | [-8.9962755, 3.49911258] |
| RM | 5.14497 | 0.18651 | [4.25408753, 6.03837272] |
| AGE | -0.00965967 | 0.000272945 | [-0.04352554, 0.02416039] |
| DIS | -1.15862 | 0.062075 | [-1.65238738, -0.66692825] |
| RAD | 0.264289 | 0.00690763 | [0.10703723, 0.42174891] |
| TAX | -0.0132118 | 2.01375e-05 | [-0.02186968, -0.00456208] |
| PTRATIO | -0.475741 | 0.0225333 | [-0.77867611, -0.17343679] |
| B | 0.0144427 | 1.11569e-05 | [0.00781772, 0.02108903] |
| LSTAT | -0.58945 | 0.00507899 | [-0.72078793, -0.45820058] |

Table 1: Bayesian Posterior Estimates Summary

```
#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: 22.463083437343656
```

```
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: 22.463083437343656
```

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 $\sigma^2$ estimates between the OLS and Bayesian method:

```
bayesian_results = tabulate(beta_estimates, headers='keys', tablefmt='latex',  numalign="left", stralig

with open('sigma2_comparison.tex', 'w') as f:
    f.write(bayesian_results)
```

```
## 1261
```

```
bayesian_beta_estimates = beta_estimates.drop("SD Estimate for Beta", axis=1)
OLS_beta_estimates = pd.read_csv('full_model_summary.csv', index_col=0).rename(index={'(Intercept)': 'I

beta_estimates_comparison = bayesian_beta_estimates.rename(columns={'Beta Estimate': 'Bayesian Beta Est

beta_estimates_comparison['Beta Estimate Difference (OLS - Bayesian)'] = bayesian_beta_estimates['Beta 
beta_estimates_comparison.insert(0, 'OLS Beta Estimate', OLS_beta_estimates['Estimate'])

beta_estimates_table = tabulate(beta_estimates_comparison, headers='keys', tablefmt='latex',  numalign=

with open('beta_estimates.tex', 'w') as f:
    f.write(beta_estimates_table)
```

```
## 1582
```

```
beta_extra_info = pd.DataFrame()
beta_extra_info['OLS Beta SD Estimate'] = OLS_beta_estimates['StdError']
beta_extra_info['Bayesian Beta SD Estimate'] = beta_estimates['SD Estimate for Beta']

OLS_beta_estimates['Lower95CI'] = OLS_beta_estimates['Lower95CI'].round(6)
OLS_beta_estimates['Upper95CI'] = OLS_beta_estimates['Upper95CI'].round(6)
beta_extra_info['OLS 95% Beta Confidence Interval'] = '[' + OLS_beta_estimates['Lower95CI'].astype(str)

beta_extra_info['Bayesian 95% Beta Credible Region'] = beta_estimates['95% Credible Region']

sd_beta_table = tabulate(beta_extra_info.iloc[:, 0:2], headers='keys', tablefmt='latex',  numalign="lef

with open('sd_beta_table.tex', 'w') as f:
    f.write(sd_beta_table)
```

```
## 1021
```

```
beta_extra_info = pd.DataFrame()
beta_extra_info['OLS Beta SD Estimate'] = OLS_beta_estimates['StdError']
beta_extra_info['Bayesian Beta SD Estimate'] = beta_estimates['SD Estimate for Beta']

beta_extra_info['Bayesian 95% Beta Credible Region'] = beta_estimates['95% Credible Region']

OLS_beta_estimates['Lower95CI'] = OLS_beta_estimates['Lower95CI'].round(6)
OLS_beta_estimates['Upper95CI'] = OLS_beta_estimates['Upper95CI'].round(6)
beta_extra_info['OLS 95% Beta Confidence Interval'] = '[' + OLS_beta_estimates['Lower95CI'].astype(str)

beta_95_CI_CR_table = tabulate(beta_extra_info.iloc[:, 2:], headers='keys', tablefmt='latex',  numalign

with open('beta_95_CI_CR_table.tex', 'w') as f:
    f.write(beta_95_CI_CR_table)
```

```
## 1303
```

| | Beta Estimate | SD Estimate for Beta | 95% Credible Region |
|---|---|---|---|
| Intercept | 9.81914 | 14.669 | [2.35275568, 17.28077673] |
| CRIM | -0.10678 | 0.00196314 | [-0.19392261, -0.0196579] |
| ZN | 0.0496652 | 0.000314251 | [0.01474696, 0.08461949] |
| INDUS | 0.0369218 | 0.00548907 | [-0.10827988, 0.18216155] |
| NX | -2.7444 | 10.4138 | [-8.9962755, 3.49911258] |
| RM | 5.14497 | 0.18651 | [4.25408753, 6.03837272] |
| AGE | -0.00965967 | 0.000272945 | [-0.04352554, 0.02416039] |
| DIS | -1.15862 | 0.062075 | [-1.65238738, -0.66692825] |
| RAD | 0.264289 | 0.00690763 | [0.10703723, 0.42174891] |
| TAX | -0.0132118 | 2.01375e-05 | [-0.02186968, -0.00456208] |
| PTRATIO | -0.475741 | 0.0225333 | [-0.77867611, -0.17343679] |
| B | 0.0144427 | 1.11569e-05 | [0.00781772, 0.02108903] |
| LSTAT | -0.58945 | 0.00507899 | [-0.72078793, -0.45820058] |

Table 2: $\sigma^2$ Results Comparison between OLS and Bayesian method

| | OLS Beta SD Estimate | Bayesian Beta SD Estimate |
|---|---|---|
| Intercept | 6.85673 | 14.669 |
| CRIM | 0.0441557 | 0.00196314 |
| ZN | 0.0177879 | 0.000314251 |
| INDUS | 0.0749102 | 0.00548907 |
| NX | 4.82122 | 10.4138 |
| RM | 0.555961 | 0.18651 |
| AGE | 0.0174077 | 0.000272945 |
| DIS | 0.261205 | 0.062075 |
| RAD | 0.0821271 | 0.00690763 |
| TAX | 0.00441613 | 2.01375e-05 |
| PTRATIO | 0.18101 | 0.0225333 |
| B | 0.00343399 | 1.11569e-05 |
| LSTAT | 0.0679132 | 0.00507899 |

Table 3: $\beta$ Coefficient Standard Deviation Estimates Comparison between OLS and Bayesian method

| | Bayesian 95% Beta Credible Region | OLS 95% Beta Confidence Interval |
|---|---|---|
| Intercept | [2.35275568, 17.28077673] | [18.549792, 45.539176] |
| CRIM | [-0.19392261, -0.0196579] | [-0.20842, -0.034615] |
| ZN | [0.01474696, 0.08461949] | [0.007537, 0.077554] |
| INDUS | [-0.10827988, 0.18216155] | [-0.080504, 0.214356] |
| NX | [-8.9962755, 3.49911258] | [-23.483286, -4.506077] |
| RM | [4.25408753, 6.03837272] | [2.927794, 5.11616] |
| AGE | [-0.04352554, 0.02416039] | [-0.038791, 0.029729] |
| DIS | [-1.65238738, -0.66692825] | [-1.983743, -0.955591] |
| RAD | [0.10703723, 0.42174891] | [0.181558, 0.504825] |
| TAX | [-0.02186968, -0.00456208] | [-0.023095, -0.005713] |
| PTRATIO | [-0.77867611, -0.17343679] | [-1.211924, -0.499434] |
| B | [0.00781772, 0.02108903] | [0.004869, 0.018386] |
| LSTAT | [-0.72078793, -0.45820058] | [-0.762857, -0.495538] |

Table 4: $\beta$ Coefficient Estimates 95

## Conclusion

## 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