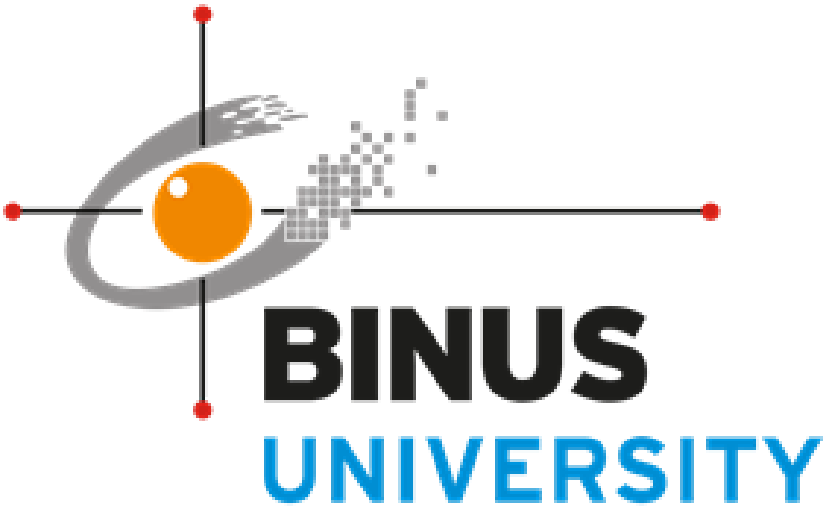
BAYESIAN DATA ANALYSIS MINI PROJECT REPORT



Members of the Group:

Brandon Agusta Wijaya	2602054146
Davin Edbert Santoso	2602067086
Nicholas Sugijono	2602064714
Steve Marcello Liem	2602071410
Wilson Gregory Pribadi	2602071000

Mata Kuliah: DTSC6010001 – Bayesian Data Analysis
Kelas: LA09
Nama Dosen : D6861- Ni Luh Putu Satyaning Pradnya Paramita, S.Si., M.Si., Ph.D

Tahun Ajaran:
Semester Ganjil 2023/2024

1. INTRODUCTION

Analyzing the 'housing_price_dataset' from Kaggle (link: https://www.kaggle.com/datasets/muhammadbinimran/housing-price-prediction-data/data). It includes six factors: 'SquareFeet,' 'Bedrooms,' 'Bathrooms,' 'Neighborhood,' 'YearBuilt,' and 'Price.' Most columns are numerical, except 'Neighborhood,' which has 'Rural,' 'Suburb,' and 'Urban.' The project's core focus is predicting housing prices, specifically the 'Price' column. Using info on square footage, bedrooms, bathrooms, neighborhood, and year built, the aim is to grasp the complex relationship impacting housing prices. 'SquareFeet,' 'Bedrooms,' 'Bathrooms,' and 'YearBuilt' are quantitative features, while 'Neighborhood' is categorical, reflecting diverse living environments affecting property values. This analysis provides insights for homeowners, real estate professionals, and housing market stakeholders.

2. MODELS

The analysis of house prices revolves around understanding the intricate factors influencing property valuations. The core model utilized is the Normal model

$$Y_i \sim \text{Normal}(\beta_0 + \sum_{j=1}^{p} X_{ij}\beta_j, \sigma^2)$$

,

A suitable model for continuous data, such as house prices, involves a formula with Y as the dependent variable to predict and X as the independent variable for prediction. The coefficients include β0 (bias) and βp (regression coefficients), and the model incorporates σ^2 as the error variance. Gaussian prior elements, with μ0 as the mean and $\sigma^2 0$ as the variance, enhance the predictive framework. In practical terms, house size, number of bedrooms, etc., constitute X, while house price is Y. The objective is to predict house prices based on these variables. Parameter values like β0 and σ^2 are determined using maximum likelihood estimation, and μ0 and $\sigma^2 0$ can be determined manually or via maximum a posteriori estimation. Once parameters are established, the formula can be employed for house price predictions.

In our R code, we set the seed to 123 for reproducible random number generation. Using the cut function, the 'Price' variable stratifies the dataset into five categories. We define 'n_samples' as 1000, representing the total sample size. The strata function then performs stratified sampling on the 'df' dataframe, determining each stratum's sample size based on its proportion to the total dataset. The sampled indices are used to create 'stratified_sampled_data,' and the original 'df' is updated to include only these samples. This process ensures a representative stratified sample mirroring the original house price distribution. Different tuning priors are implemented for a nuanced understanding of the dataset.

1. **Uninformative Gaussian Prior (Without Thinning - thin = 1):**

$$\beta_j \sim Normal(0, 0.001)$$

This model employs an Uninformative Gaussian prior (dnorm(0, 0.001)) to foster a neutral, tabula rasa approach. This choice ensures a data-centric analysis free from biases. The simplicity of dnorm(0, 0.001) for parameters like beta reflects a deliberately weak prior, allowing the data to guide insights without imposing strong assumptions. The intercept alpha and precision tau are assigned Gaussian (dnorm(0, 0.001)) and Gamma (dgamma(0.1, 0.1)) priors, respectively, maintaining a balanced foundation for Bayesian inference. In essence, the model prioritizes an unfiltered, data-driven perspective while incorporating minimal prior information.

2. **Uninformative Gaussian Prior (With Modified priors and parameters):**

$$\beta_j \sim Normal(0, 0.01)$$

This model iteration diverges from the Stochastic Search Variable Selection (SSVS) approach, as every feature remained crucial in both the initial and post-SSVS models. Consequently, the decision was made to forego the SSVS-informed model. Instead, a new model was crafted, incorporating modified priors and parameters to strike a balance between simplicity and comprehensive consideration of each feature. Motivated by a desire to enhance model interpretability while ensuring the capture of individual feature significance, the updated model employs Gaussian priors (dnorm(0, 0.01)) for coefficients. This choice aims to facilitate a nuanced exploration of feature importance. Additionally, the intercept alpha and precision tau are updated with Gaussian (dnorm(0, 0.01)) and Gamma (dgamma(0.5, 0.5)) priors, respectively, contributing to a refined modeling framework.

The next step involves a focused comparison among the original, SSVS-informed, and the latest model variant with modified priors to determine the most effective model for our dataset. This iterative approach emphasizes continuous refinement for optimal model performance, showcasing a comprehensive exploration to discern the factors significantly shaping house prices. Each model variant provides a unique perspective on interpreting the dataset, with a shared goal of uncovering essential predictors while maintaining a delicate balance between model complexity and interpretability.

3. COMPUTATION

All model fittings were conducted using the JAGS package in R, ensuring a standardized approach for both the initial and refined second models. The process involved a burn-in phase, discarding the initial 5000 samples, followed by generating 10000 samples for each of the 2 chains, with a thinning factor of 1. Rigorous convergence checks were implemented to guarantee the robustness and reliability of the results.

The convergence assessment comprised three key components. Firstly, trace plots visually confirmed the convergence of the models. Secondly, the Gelman-Rubin statistic (PSRF) was scrutinized, with all values at 1, affirming convergence. Thirdly, Effective Sample Size (ESS) values for each beta coefficient, detailed in Table 1 and Table 1, consistently exceeded the 1000 benchmark, reinforcing convergence assurance. This comprehensive validation, incorporating visual and numerical checks, establishes a strong foundation for subsequent analyses, instilling confidence in the stability and reliability of both the initial and refined models.

Table 1 Effective Sample Size (ESS) of Model 1 and Model 2

	beta[1]	**beta[2]**	**beta[3]**	**beta[4]**	**beta[5]**
Model 1	17514.17	12246.87	10883.57	12921.04	13720.09
Model 2	17216.35	12255.85	11104	12811.25	14960.85

4. Model Comparisons

The SSVS results, presented in Table 2, reveal an inc_prob value of 1 for all 5 beta coefficients. This unequivocally suggests that every feature is deemed optimal for inclusion in the model. Consequently, there is a redundancy in constructing a model based on these features post-SSVS, rendering them all equally significant.

As a result, our mini project focuses on two distinctive models: Model 1, characterized by the Uninformative Gaussian Prior (Without Thinning - thin = 1), and Model 2, leveraging the Uninformative Gaussian Prior with modified priors and parameters. This deliberate narrowing down to two models streamlines our analysis, allowing for a focused comparison between the original uninformative approach and the nuanced refinement introduced in Model 2.

Table 2 Summary of the posteriors from SSVS

	inc_prob	**50%**	**5%**	**95%**
beta[1]	1	0.21	0.20	0.22
beta[2]	1	0.05	0.04	0.06
beta[3]	1	-0.07	-0.08	-0.06
beta[4]	1	0.06	0.05	0.07
beta[5]	1	-0.07	-0.08	-0.06

The evaluation of model performance relies on the Deviance Information Criterion (DIC) and Multi PSRF values from Gelman-Rubin Statistics in Table 3. Despite modifications to priors

and parameters in the second model, minimal differentiation is observed. Similar performance metrics suggest that adjustments may not significantly impact overall model performance. The final model selection is detailed in the Results section, where convergence diagnostics, goodness of fit, and performance metrics inform a well-founded decision aligning with analysis objectives.

Table 3 Model comparisons

	Mean Deviance	**Penalty**	**DIC**	**Multi PSRF**
Model 1	975.9	6.962	982.9	1
Model 2	976	7.03	983.1	1

Here's the breakdown of DIC for both models:
1.For Model 1, the mean deviance is 975.9 with a penalty of 6.962, leading to a penalized deviance (DIC) of 982.9.
2.For Model 2, the mean deviance is 976 with a penalty of 7.03, leading to a penalized deviance (DIC) of 983.1.

Deviance Information Criterion (DIC)
The DIC is composed of two parts:

1. The mean deviance, which reflects the fit of the model to the data. A lower mean deviance indicates a better fit.
2. The penalty, which accounts for model complexity. A more complex model will have a higher penalty.

The penalized deviance, a combination of mean deviance and penalty, guides model selection, favoring lower values for a better fit-complexity trade-off. The marginal difference in penalized deviance (0.2) between the two models is negligible, especially considering MCMC sampling variability. Practically, both models are equivalent in terms of DIC. Similarly, close Multivariate PSRF values indicate effective convergence for both models, reinforcing their comparability.

5. Results

The examination of p-values in Table 4 and 6 for each model reveals noteworthy patterns, particularly in the cases of Min X, Min Y, and Range Y. The proximity of these values to 1 and 0 is indicative of potential issues. P-values serve as a metric assessing the extremeness of observed data concerning the sampling distribution. When the p-value tends towards either zero or one, it suggests a lack of fit in the model.

In essence, the proximity of Min X, Min Y, and Range Y p-values to the extremes raises concerns about the adequacy of model fit. Such observations prompt a critical evaluation of the model's appropriateness and call for a closer examination of potential adjustments or alternative modeling strategies. It's important to highlight that even though our model's parameters seem to

stabilize well (converge), the p-values tell us that the model doesn't quite match the actual data. This discrepancy suggests that, despite the parameters behaving well, the model may not be the best fit for our dataset.

This discrepancy underscores the importance of an evaluation that goes beyond convergence diagnostics alone. The convergence of parameters is a necessary but not a sufficient condition for a model to be considered appropriate. The incongruence with the data, as indicated by the p-values, prompts a careful consideration of potential model refinements or alternative strategies. This nuanced approach ensures that the chosen model not only converges well but also effectively captures the underlying data patterns, aligning with the overarching goals of the analysis.

Table 4 p-value from the posterior predictive checks Model 1 and Model 2

	Min Y	**Max Y**	**Range Y**
Model 1	1	1	0.0268
Model 2	1	1	0.0255

Based on the provided information in Table 3, both models have the same DIC values, which are 983. Low DIC values indicate that the models perform well in predicting data. Therefore, both models can be considered to have equally good performance. However, there are some things to consider when comparing the two models. Firstly, the models use different priors. The Gaussian Non-Informative Prior (No Shrinkage - thinness = 1) is a neutral prior, while the Gaussian Non-Informative Prior (With Modified Prior and Parameters) is a modified prior aimed at improving model interpretability. Secondly, both models use the same 5 features. However, the influence of these features on house prices may differ. This can be seen from the beta values produced by both models.

Based on these factors, it can be said that the best model depends on the user's needs and the intended use of the model. If a model with good interpretability is needed, then the Gaussian Non-Informative Prior (With Modified Prior and Parameters) may be a better choice. However, if a model with better performance in predicting data is required, then both models can be used interchangeably to see which one performs better on specific data.

==========

#Load the library

```{r}
library(dplyr)
library(labelled)
library(rjags)
library(sampling)
library(knitr)
library(kableExtra)
```

#Load the data

```{r}
df <- read.csv("D:\\DATA SCIENCE SEMESTER 3\\BAYESIAN DATA ANALYTICS\\PROJECT\\housing_price_dataset.csv")
df
```

#Data description

```{r}
str(df)
```

#Count missing values

```{r}
null_count <- colSums(is.na(df))
null_count
```

#Encode the categorical variable to numerical

```{r}
neighborhood_levels <- c("Rural", "Suburb", "Urban")
neighborhood_labels <- c("1", "2", "3")
df$Neighborhood <- factor(df$Neighborhood, levels = neighborhood_levels, labels = neighborhood_labels)
```

#Encode all int type of variable to numeric

```{r}
df$SquareFeet <- as.numeric(df$SquareFeet)
df$Bedrooms <- as.numeric(df$Bedrooms)
df$Bathrooms <- as.numeric(df$Bathrooms)
df$Neighborhood <- as.numeric(df$Neighborhood)
df$YearBuilt <- as.numeric(df$YearBuilt)
df$Price <- as.numeric(df$Price)
df
```

#Use stratified sampling

```{r}
set.seed(123)
df$strata <- cut(df$Price, breaks = 5)
n_samples <- 1000
stratified_sample <- strata(df, "strata", size = floor(n_samples * prop.table(table(df$strata))))
sampled_indices <- as.numeric(unlist(stratified_sample))
stratified_sampled_data <- df[sampled_indices, ]
df <- stratified_sampled_data
df
```

```{r}
Y <- log(df$Price)
X <- as.matrix(df[, c("SquareFeet", "Bedrooms", "Bathrooms", "Neighborhood", "YearBuilt")])
```

#Standardize the covariates

```{r}
X = scale(X)
```

#Put the data in JAGS format
```{r}
n <- length(Y)
p <- ncol(X)
data <- list(Y=Y,X=X,n=n,p=p)
params <- c("beta")
burn <- 5000
n.iter <- 10000
n.chains <- 2
```

#Fit the uninformative Gaussian model
==========
| MODEL 1 |
==========
#Define the model as a string
```{r}
model_string <- textConnection("model
{
 # Likelihood
 for(i in 1:n)
 {
  Y[i]  ~ dnorm(alpha+mu[i],tau)
  mu[i] <- inprod(X[i,],beta[])
 }

 # Priors
 for(j in 1:p)
 {
  beta[j] ~ dnorm(0,0.001)
 }

 alpha ~ dnorm(0,0.001)
 tau ~ dgamma(0.1, 0.1)

}")
```

#Load the data and compile the MCMC code
```{r}
model <- jags.model(model_string, data = data, n.chains = n.chains, quiet = FALSE)
```

#Burn in for 5000 samples
```{r}
update(model, burn)
```

#Generate 10000 post-burn-in samples
```{r}
samples <- coda.samples(model, variable.names=params, n.iter=n.iter)
```

#Summarize the output
```{r}
summary(samples)
```

#Graphical Convergence Diagnostics
```{r}
plot(samples)
```

#Numerical Convergence Diagnostics: Gelman-Rubin Statistics
```{r}
gelman.diag(samples)
```
PSRF = 1, means that it's perfect and indicates convergence

#Effective Sample Size (ESS)
```{r}
effectiveSize(samples)
```

#Deviance Information Criteria (DIC)
```{r}
DIC1 <- dic.samples(model,n.iter=n.iter,progress.bar="none")
```

DIC1
```

#Feature Selection using SSVS
```{r}
model_string2 <-
textConnection("model
{
 # Likelihood
 for(i in 1:n)
 {
  Y[i] ~
dnorm(alpha+inprod(X[i,],beta[]),tau)
 }

 # Priors
 for(j in 1:p)
 {
  beta[j] <- gamma[j]*delta[j]
  gamma[j] ~ dbern(0.5)
  delta[j] ~ dnorm(0,tau)
 }

 alpha ~ dnorm(0,0.001)
 tau ~ dgamma(0.1, 0.1)
}")

model2 <- jags.model(model_string2,
data = data, n.chains = n.chains, quiet =
TRUE)
update(model2, burn)
samples2 <- coda.samples(model2,
variable.names=params, n.iter=n.iter)
```

```{r}
beta <- NULL
for(l in 1: n.chains){
 beta <- rbind(beta,samples2[[l]])
}

inc_prob <- apply(beta!=0,2,mean)
q <-
t(apply(beta,2,quantile,c(0.5,0.05,0.95)))
out <- cbind(inc_prob,q)
kable(round(out,2))
```

Karena inc_prob untuk setiap beta sudah 1, maka tidak perlu melakukan features selections.
```

==========
| MODEL 2 |
==========

```{r}
# Model 2: Update the model_string to modify priors and parameters
model_string_2 <- textConnection("model {
  # Likelihood
  for(i in 1:n) {
    Y[i] ~ dnorm(alpha + mu[i], tau)
    mu[i] <- inprod(X[i,], beta[])
  }

  # Priors
  for(j in 1:p) {
    beta[j] ~ dnorm(0, 0.01)  # Updated prior
  }

  alpha ~ dnorm(0, 0.01)    # Updated prior
  tau ~ dgamma(0.5, 0.5)     # Updated prior

}")

# Compile and run Model 2
model_2 <- jags.model(model_string_2, data = data, n.chains = n.chains, quiet = TRUE)
update(model_2, burn)
samples_2 <- coda.samples(model_2, variable.names = params, n.iter = n.iter)

# Summary of Model 2
summary(samples_2)
```

#Graphical Convergence Diagnostics

```{r}
plot(samples_2)
```

#Numerical Convergence Diagnostics: Gelman-Rubin Statistics

```{r}
gelman.diag(samples_2)
```

PSRF = 1, means that it's perfect and indicates convergence

#Effective Sample Size (ESS)

```{r}
effectiveSize(samples_2)
```

#Deviance Information Criteria (DIC)

```{r}
DIC1 <- dic.samples(model_2,n.iter=n.iter,progress.bar="none")
DIC1
```

#Posterior Predictive Checks USING MODEL 1

```{r}
model_string <- textConnection("model
{
  # Likelihood
  for(i in 1:n)
  {
    Y[i] ~ dnorm(alpha+inprod(X[i,],beta[]),tau)
  }

  # Priors
  for(j in 1:p)
  {
    beta[j] ~ dnorm(0,0.001)
  }

  alpha ~ dnorm(0,0.001)
  tau ~ dgamma(0.1, 0.1)

  #Posterior predictive checks
  for(i in 1:n){
```

```
   Y2[i] ~
dnorm(alpha+inprod(X[i,],beta[]),tau)
  }
  D[1] <- min(Y2[])
  D[2] <- max(Y2[])
  D[3] <- max(Y2[])-min(Y2[])
}")
```

```{r}
model <- jags.model(model_string, data
= data, n.chains = n.chains, quiet =
TRUE)
update(model, burn)
samples <- coda.samples(model,
variable.names=c("D", "beta"),
n.iter=n.iter)
```

```{r}
summary(samples)
```

```{r}
D1 <- samples[[1]]
D2 <- samples[[2]]

#Compute the test stats for the data
D0 <-
c(min(Y),max(Y),max(Y)-min(Y))
Dnames <- c("Min Y","Max Y","Range
Y")

#Compute the test stats for the model
pval <- rep(0,3)
names(pval) <- Dnames
```

```{r}
for (j in 1:3) {
  plot(density(D1[, j]), xlim =
range(c(D0[j], D1[, j], D2[, j])), xlab =
"D",
      ylab = "Posterior probability", main
= Dnames[j], col = "blue", lty = 1)
  lines(density(D2[, j]), col = "red", lty =
1)
  abline(v = D0[j], col = "green", lty = 1)
  legend("topleft", c("Data", "Chain 1",
"Chain 2"), lty = c(1, 1, 1, 1),
         col = c("green", "blue", "red"), bty
= "n")

  pval[j] <- mean(D1[, j] > D0[j])
}
```

#Results
```{r}
pval
```