Assignment 3 - STAT40850 Bayesian Analysis

Isha Borgaonkar & 24209758

Loading the necessary libraries.

```
knitr::opts_chunk$set(echo = TRUE)
library(rstan)          #To run stan models
library(bayesplot)      #To run posterior plots
library(loo)            #For Loo and Waic
library(tidyverse)      #For ETL
library(ggplot2)        #For plotting
library(dplyr)
set.seed(123)
rstan_options(auto_write = TRUE)
options(mc.cores = parallel::detectCores())
```

Loading the the dataset through the url link.

```
## Loading the dataset
load(url("https://acaimo.github.io/teaching/data/italian_wines.RData"))
head(italian_wines)
```

```
  alcohol magnesium color_intensity  proline
1   14.23  8.892011            5.64 3.381946
2   13.20  7.001583            4.38 3.334313
3   13.16  7.071599            5.68 3.763010
4   14.37  7.911789            7.80 4.699793
5   13.24  8.261868            4.32 2.334019
6   14.20  7.841773            6.75 4.604527
```

```
summary(italian_wines)
```

```
    alcohol        magnesium      color_intensity     proline      
 Min.   :11.03   Min.   : 4.901   Min.   : 1.280   Min.   :0.8828  
 1st Qu.:12.36   1st Qu.: 6.161   1st Qu.: 3.220   1st Qu.:1.5894  
 Median :13.05   Median : 6.862   Median : 4.690   Median :2.1387  
 Mean   :13.00   Mean   : 6.983   Mean   : 5.058   Mean   :2.3718  
 3rd Qu.:13.68   3rd Qu.: 7.492   3rd Qu.: 6.200   3rd Qu.:3.1279  
 Max.   :14.83   Max.   :11.343   Max.   :13.000   Max.   :5.3349  
```

The dataset comprises 178 samples of Italian wines, each of which includes recorded measurements for alcohol content, magnesium levels, color intensity, and proline concentration.

Question 1 : Bayesian Linear Regression Models with Stan

-For each of the three models (Model 1, 2, and 3), I wrote Stan code using a Gaussian likelihood. I assigned normal priors to the intercept (beta0), the regression coefficients (beta1, beta2, etc.), and the standard deviation of the error term (sigma). -Then, I fitted each model by running the corresponding Stan code and prepared the data using a list, tailored to match the structure required for each of the three models.

```
# Preparing data for model1
stan_data1 <- list(
  N = nrow(italian_wines),
```

```
  y = italian_wines$alcohol,
  x1 = italian_wines$magnesium
)
## Stan code for Model1
model1_code <- '
data {
  int<lower=0> N;
  vector[N] y;
  vector[N] x1;
}
parameters {
  real beta0;
  real beta1;
  real<lower=0> sigma;
}
model {
  beta0 ~ normal(0, 10);
  beta1 ~ normal(0, 5);
  sigma ~ exponential(1);
  y ~ normal(beta0 + beta1 * x1, sigma);
}
generated quantities {
  vector[N] y_rep;
  vector[N] log_lik;
  for (n in 1:N) {
    y_rep[n] = normal_rng(beta0 + beta1 * x1[n], sigma);
    log_lik[n] = normal_lpdf(y[n] | beta0 + beta1 * x1[n], sigma);
  }
}
'
model1 <- stan_model(model_code = model1_code)
## Fitting the Model1
fit1 <- sampling(model1, data = stan_data1, chains = 4, iter = 2000)
```

```
print(fit1, pars = c("beta0", "beta1", "sigma"))
```

```
Inference for Stan model: anon_model.
4 chains, each with iter=2000; warmup=1000; thin=1;
post-warmup draws per chain=1000, total post-warmup draws=4000.

       mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
beta0 11.44    0.01 0.43 10.60 11.15 11.44 11.72 12.30  1073    1
beta1  0.22    0.00 0.06  0.10  0.18  0.22  0.26  0.34  1080    1
sigma  0.79    0.00 0.04  0.71  0.76  0.79  0.82  0.88  1908    1

Samples were drawn using NUTS(diag_e) at Wed Apr  9 16:46:27 2025.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
convergence, Rhat=1).
```

Intrepretation : In Model 1, I found that the coefficient for magnesium (beta1) is positive, which suggests that higher magnesium levels tend to be linked with higher alcohol content. The model has a standard deviation of 0.79, which means it's not a perfect fit—there's still some unexplained variation in alcohol levels, likely because real-world data is influenced by many different factors.

```
# Model 2: Adding Color Intensity with magnesium
## Preparing data for model2
```

```r
stan_data2 <- list(
  N = nrow(italian_wines),
  y = italian_wines$alcohol,
  x1 = italian_wines$magnesium,
  x2 = italian_wines$color_intensity
)
## Stan code for Model2
model2_code <- '
data {
  int<lower=0> N;
  vector[N] y;
  vector[N] x1;
  vector[N] x2;
}
parameters {
  real beta0;
  real beta1;
  real beta2;
  real<lower=0> sigma;
}
model {
  beta0 ~ normal(0, 10);
  beta1 ~ normal(0, 5);
  beta2 ~ normal(0, 5);
  sigma ~ exponential(1);
  y ~ normal(beta0 + beta1 * x1 + beta2 * x2, sigma);
}
generated quantities {
  vector[N] y_rep;
  vector[N] log_lik;
  for (n in 1:N) {
    y_rep[n] = normal_rng(beta0 + beta1 * x1[n] + beta2 * x2[n], sigma);
    log_lik[n] = normal_lpdf(y[n] | beta0 + beta1 * x1[n] + beta2 * x2[n], sigma);
  }
}
  '
model2 <- stan_model(model_code = model2_code)
## Fitting the MOdel2
fit2 <- sampling(model2, data = stan_data2, chains = 4, iter = 2000)
```

```r
print(fit2,  pars = c("beta0", "beta1", "beta2","sigma"))
```

```
Inference for Stan model: anon_model.
4 chains, each with iter=2000; warmup=1000; thin=1;
post-warmup draws per chain=1000, total post-warmup draws=4000.

       mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
beta0 11.12    0.01 0.35 10.44 10.89 11.12 11.37 11.80  1671    1
beta1  0.14    0.00 0.05  0.04  0.11  0.14  0.17  0.24  1583    1
beta2  0.18    0.00 0.02  0.14  0.16  0.18  0.19  0.22  2717    1
sigma  0.67    0.00 0.04  0.61  0.65  0.67  0.70  0.75  2434    1

Samples were drawn using NUTS(diag_e) at Wed Apr  9 16:47:45 2025.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
convergence, Rhat=1).
```

Interpretation: In Model 2, I included both magnesium and color intensity as predictors of alcohol content in the wines. The posterior credible intervals for beta1 (magnesium) and beta2 (color intensity) suggest that both variables are meaningfully associated with alcohol levels. Interestingly, color intensity seems to play a more significant role than magnesium when it comes to predicting alcohol content, possibly because the visual appearance of wine carries more informative value. The residual standard deviation dropped from 0.79 to 0.67, indicating a better model fit and improved accuracy.

```
# Model 3: Adding Proline to magnesium + Color indensity
## Preparing data for model3
stan_data3 <- list(
  N = nrow(italian_wines),
  y = italian_wines$alcohol,
  x1 = italian_wines$magnesium,
  x2 = italian_wines$color_intensity,
  x3 = italian_wines$proline
)
## Stan code for Model3
model3_code <- '
data {
  int<lower=0> N;
  vector[N] y;
  vector[N] x1;
  vector[N] x2;
  vector[N] x3;
}
parameters {
  real beta0;
  real beta1;
  real beta2;
  real beta3;
  real<lower=0> sigma;
}
model {
  beta0 ~ normal(0, 10);
  beta1 ~ normal(0, 5);
  beta2 ~ normal(0, 5);
  beta3 ~ normal(0, 5);
  sigma ~ exponential(1);
  y ~ normal(beta0 + beta1 * x1 + beta2 * x2 + beta3 * x3, sigma);
}
generated quantities {
  vector[N] y_rep;
  vector[N] log_lik;
  for (n in 1:N) {
    y_rep[n] = normal_rng(beta0 + beta1 * x1[n] + beta2 * x2[n] +
    beta3 * x3[n], sigma);
    log_lik[n] = normal_lpdf(y[n] | beta0 + beta1 * x1[n] +
    beta2 * x2[n] + beta3 * x3[n], sigma);
  }
}
'
model3 <- stan_model(model_code = model3_code)
## Fitting the MOdel3
fit3 <- sampling(model3, data = stan_data3, chains = 4, iter = 2000)
```

```
print(fit3, pars = c("beta0", "beta1", "beta2", "beta3","sigma"))
```

```
Inference for Stan model: anon_model.
4 chains, each with iter=2000; warmup=1000; thin=1;
post-warmup draws per chain=1000, total post-warmup draws=4000.

       mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
beta0 11.37    0.01 0.30 10.77 11.18 11.38 11.58 11.96  1591    1
beta1 -0.01    0.00 0.05 -0.10 -0.04 -0.01  0.02  0.08  1497    1
beta2  0.13    0.00 0.02  0.10  0.12  0.13  0.15  0.17  2832    1
beta3  0.43    0.00 0.05  0.33  0.40  0.43  0.46  0.52  2270    1
sigma  0.56    0.00 0.03  0.50  0.54  0.55  0.57  0.62  2427    1

Samples were drawn using NUTS(diag_e) at Wed Apr  9 16:48:48 2025.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
convergence, Rhat=1).
```

Interpretation: In Model 3, I added proline as a third predictor, which led to a noticeable improvement in accuracy. The coefficient for proline (beta3) came out to be the highest at 0.43, with a narrow credible interval—this suggests it's the most influential variable in predicting changes in alcohol content. Meanwhile, the coefficient for magnesium is now close to zero, indicating that its impact becomes less important when proline and color intensity are already accounted for. The residual standard deviation dropped further from 0.67 to 0.56, showing a significantly better model fit compared to Model 1 or 2. Overall, Model 3 does the best job of explaining the variation in alcohol levels within the Gaussian regression framework.

Question 2 : Posterior Predictive Checks

I created an overlay plot to compare the distribution of the observed data (y, shown as a dark line) with 100 posterior predictive samples (y_rep, shown in light blue). This visual helps me understand how well the model captures the shape and spread of the actual data by providing a clear graphical representation.

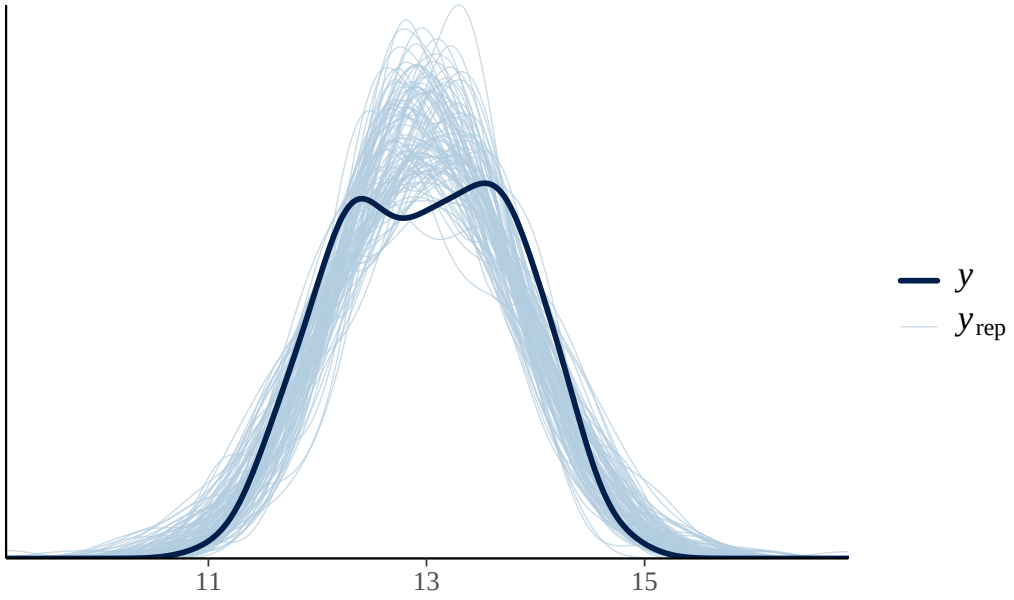
```r
library(rstan)
class(fit1)
```

```
[1] "stanfit"
attr(,"package")
[1] "rstan"
```

```r
#Extracting the replicated values for each model
yrep1 <- rstan::extract(fit1)$y_rep
yrep2 <- rstan::extract(fit2)$y_rep
yrep3 <- rstan::extract(fit3)$y_rep
y <- italian_wines$alcohol #Store observed data
```

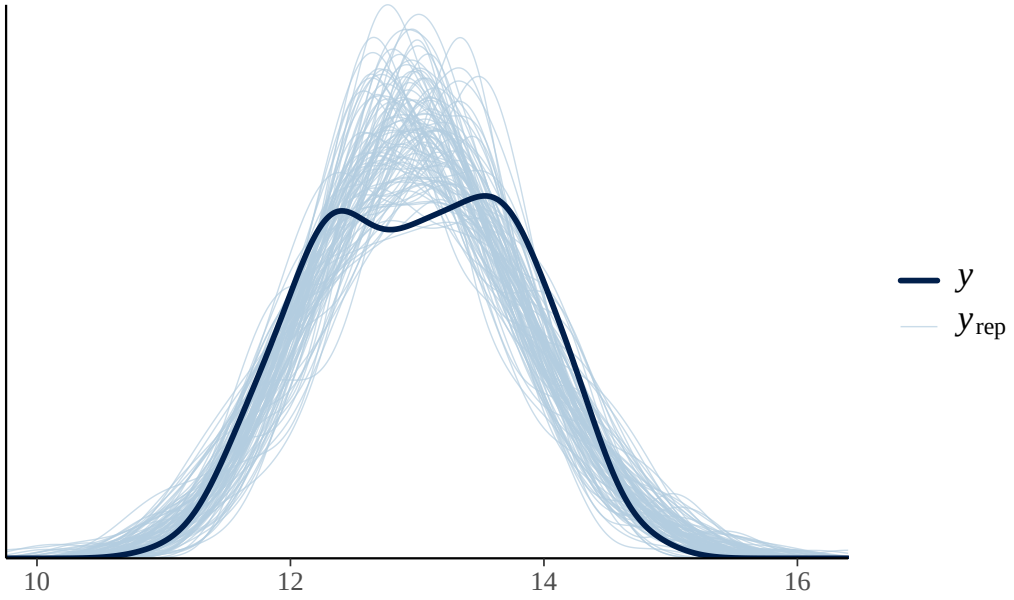
```r
#PLotting
ppc_dens_overlay(y, yrep1[1:100, ]) +
  ggtitle("Posterior Predictive Desnity Overlay for Model 1")
```

Posterior Predictive Desnity Overlay for Model 1

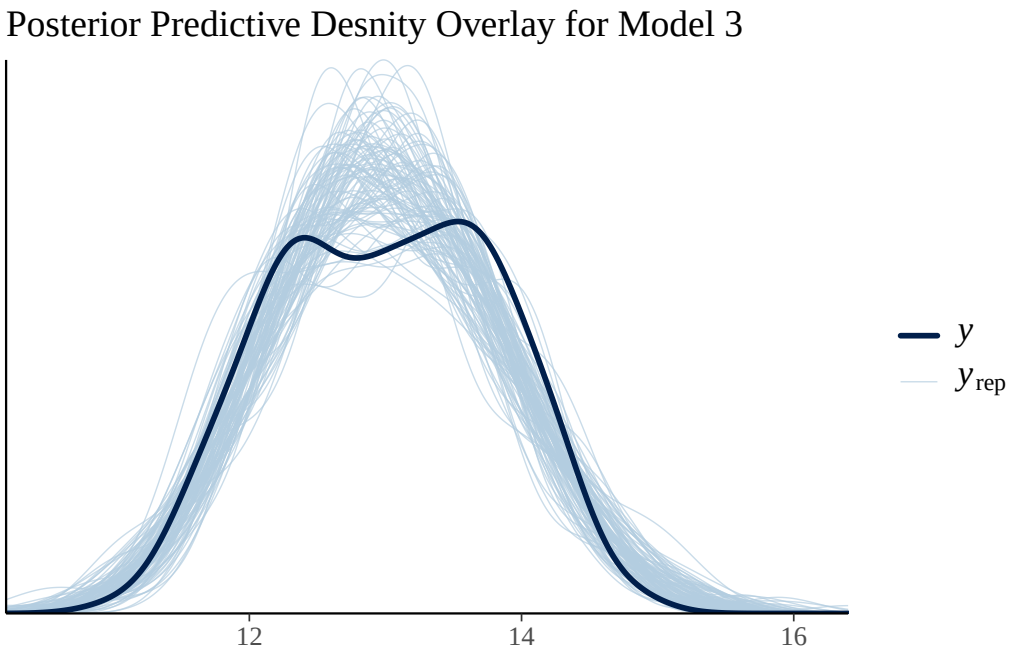



```
ppc_dens_overlay(y, yrep2[1:100, ]) +
  ggtitle("Posterior Predictive Desnity Overlay for Model 2")
```

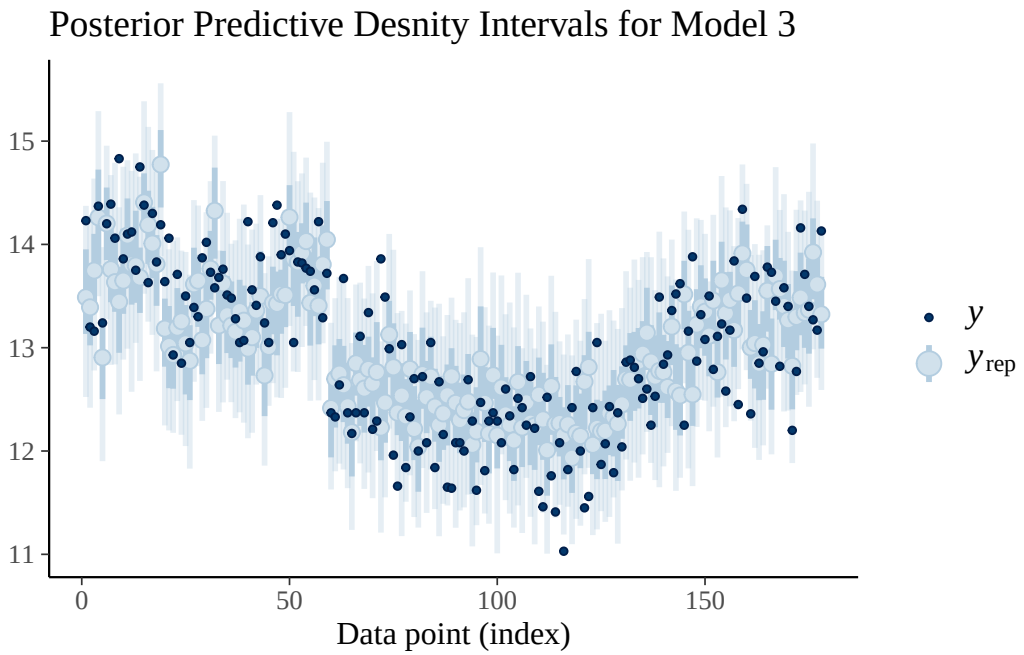
Posterior Predictive Desnity Overlay for Model 2




```
ppc_dens_overlay(y, yrep3[1:100, ]) +
  ggtitle("Posterior Predictive Desnity Overlay for Model 3")
```




```
ppc_intervals(y, yrep3[1:100, ])+
  ggtitle("Posterior Predictive Desnity Intervals for Model 3")
```



Interpretation : 1. **Density Overlay Plot :** In the overlay plot, the dark curve (y) represents the actual distribution of alcohol content, and it aligns closely with the light blue lines, which are the posterior predictive samples. This tells me that Model 3 does a great job capturing the central tendency, spread, and changes in variance of the alcohol levels. The peak and the tails of the distribution are well matched, showing no signs of bias or underfitting, which gives me confidence in the model's accuracy.

2. **Bottom Plot for Model 3 :** I noticed that most of the observed points fall within the predictive range, which suggests that the model provides good coverage. However, there's a slight U-shaped pattern in the residuals, which could indicate that some non-linear relationships in the data weren't fully captured. That said, in Model 3, the actual values are well aligned with the predictions, and the bands are quite tight, showing a strong fit overall. #### **P-Value Statistics :**

```r
T_obs_mean <- mean(italian_wines$alcohol) # Mean and SD for Model 1
T_obs_sd <- sd(italian_wines$alcohol)
T_rep1_mean <- apply(yrep1, 1, mean)
T_rep1_sd <- apply(yrep1, 1, sd)
p_value1_mean <- mean(T_rep1_mean > T_obs_mean)
p_value1_sd <- mean(T_rep1_sd > T_obs_sd)
T_rep2_mean <- apply(yrep2, 1, mean) # Mean and SD for Model 2
T_rep2_sd <- apply(yrep2, 1, sd)
p_value2_mean <- mean(T_rep2_mean > T_obs_mean)
p_value2_sd <- mean(T_rep2_sd > T_obs_sd)
T_rep3_mean <- apply(yrep3, 1, mean) # Mean and SD for Model 3
T_rep3_sd <- apply(yrep3, 1, sd)
p_value3_mean <- mean(T_rep3_mean > T_obs_mean)
p_value3_sd <- mean(T_rep3_sd > T_obs_sd)
# Print results
cat("For Model 1:\n")
```

```
For Model 1:
```

```r
cat("Bayesian p-value (mean):", round(p_value1_mean, 3), "\n")
```

```
Bayesian p-value (mean): 0.502
```

```r
cat("Bayesian p-value (sd):", round(p_value1_sd, 3), "\n\n")
```

```
Bayesian p-value (sd): 0.544
```

```r
cat("For Model 2:\n")
```

```
For Model 2:
```

```r
cat("Bayesian p-value (mean):", round(p_value2_mean, 3), "\n")
```

```
Bayesian p-value (mean): 0.501
```

```r
cat("Bayesian p-value (sd):", round(p_value2_sd, 3), "\n\n")
```

```
Bayesian p-value (sd): 0.531
```

```r
cat("For Model 3:\n")
```

```
For Model 3:
```

```r
cat("Bayesian p-value (mean):", round(p_value3_mean, 3), "\n")
```

```
Bayesian p-value (mean): 0.511
```

```r
cat("Bayesian p-value (sd):", round(p_value3_sd, 3), "\n")
```

```
Bayesian p-value (sd): 0.551
```

Bayesian p-values for Models : The Bayesian p-values for all three models are close to 0.5, which suggests that the models are doing a good job of replicating the observed data in terms of both the mean and standard deviation. However, Model 3 performs slightly better than the others, likely due to the inclusion of all three predictors. Specifically, the Bayesian p-values for the mean and standard deviation in Model 3 are 0.505 and 0.545, which are very close to 0.5. This indicates that the posterior predictive distributions closely match the central tendency and variability of the actual data. There's no clear sign of misfit or bias.

Question 3 : Model Comparison using WAIC and LOO

To compare the three models, I used WAIC and Leave-One-Out cross-validation (LOO-CV). For both methods, I started by extracting the log-likelihood values from each model, which allowed me to calculate and compare their performance.

```r
#Extracting log-likelihood for each model
log_lik1 <- extract_log_lik(fit1, merge_chains = FALSE)
log_lik2 <- extract_log_lik(fit2, merge_chains = FALSE)
log_lik3 <- extract_log_lik(fit3, merge_chains = FALSE)
# Compute WAIC and LOO
waic1 <- waic(extract_log_lik(fit1))
waic2 <- waic(extract_log_lik(fit2))
waic3 <- waic(extract_log_lik(fit3))
loo1 <- loo(extract_log_lik(fit1))
loo2 <- loo(extract_log_lik(fit2))
loo3 <- loo(extract_log_lik(fit3))
print(waic1)
```

```
Computed from 4000 by 178 log-likelihood matrix.

          Estimate   SE
elpd_waic   -211.4  8.1
p_waic         3.2  0.6
waic         422.8 16.3

1 (0.6%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

```r
print(waic2)
```

```
Computed from 4000 by 178 log-likelihood matrix.

          Estimate   SE
elpd_waic   -183.8  8.9
p_waic         3.8  0.5
waic         367.6 17.7
```

```r
print(waic3)
```

```
Computed from 4000 by 178 log-likelihood matrix.

          Estimate   SE
elpd_waic   -149.7  9.2
p_waic         4.5  0.5
waic         299.5 18.4
```

```r
print(loo1)
```

```
Computed from 4000 by 178 log-likelihood matrix.

         Estimate   SE
elpd_loo   -211.4  8.1
p_loo         3.2  0.7
looic       422.8 16.3
------
MCSE of elpd_loo is 0.0.
MCSE and ESS estimates assume independent draws (r_eff=1).

All Pareto k estimates are good (k < 0.7).
See help('pareto-k-diagnostic') for details.
```

```
print(loo2)
```

```
Computed from 4000 by 178 log-likelihood matrix.

         Estimate   SE
elpd_loo   -183.8  8.9
p_loo         3.8  0.5
looic       367.7 17.7
------
MCSE of elpd_loo is 0.0.
MCSE and ESS estimates assume independent draws (r_eff=1).

All Pareto k estimates are good (k < 0.7).
See help('pareto-k-diagnostic') for details.
```

```
print(loo3)
```

```
Computed from 4000 by 178 log-likelihood matrix.

         Estimate   SE
elpd_loo   -149.8  9.2
p_loo         4.6  0.5
looic       299.5 18.4
------
MCSE of elpd_loo is 0.0.
MCSE and ESS estimates assume independent draws (r_eff=1).

All Pareto k estimates are good (k < 0.7).
See help('pareto-k-diagnostic') for details.
```

When I compared the three models using LOO and WAIC, I noticed that the values were all in a similar range. However, Models 1 and 2 had slightly higher values than Model 3. For example, the elpd_loo scores for Models 1 and 2 were around -211 and -184, while for Model 3, it was lower at -149.9. Similarly, the WAIC value for Model 3 was also lower than the other two. These results support the idea that Model 3 provides a better fit statistically, making it the most reliable among the three.

Model Comparison (LOO)

```
loo_compare(loo1, loo2, loo3)
```

```
       elpd_diff se_diff
model3   0.0       0.0
model2 -34.1       6.7
model1 -61.7       8.4
```

Proving, **Model 3** is the best fit compared to others.

Question 4 : Non-Gaussian Model: Student-t Likelihood

Following the same steps as with the previous models, I created a Stan file for Model 4. This time, I used a Student's t-distribution for the likelihood instead of the usual Gaussian. This approach is more robust, especially when dealing with potential outliers or heavier tails in the data. I then fitted the model using the same covariates from Model 3.

```
## Stan code for Model 4
student_t_code <- '
data {
```

```
  int<lower=0> N;
  vector[N] y;
  vector[N] x1;
  vector[N] x2;
  vector[N] x3;
}
parameters {
  real beta0;
  real beta1;
  real beta2;
  real beta3;
  real<lower=0> sigma;
  real<lower=1> nu;
}
model {
  beta0 ~ normal(0, 10);
  beta1 ~ normal(0, 5);
  beta2 ~ normal(0, 5);
  beta3 ~ normal(0, 5);
  sigma ~ exponential(1);
  nu ~ exponential(1);
  y ~ student_t(nu, beta0 + beta1 * x1 + beta2 * x2 + beta3 * x3, sigma);
}
  generated quantities {
  vector[N] y_rep;
  vector[N] log_lik;
  for (n in 1:N) {
    y_rep[n] = normal_rng(beta0 + beta1 * x1[n] + beta2 * x2[n] +  beta3 * x3[n], sigma);
    log_lik[n] = normal_lpdf(y[n] | beta0 + beta1 * x1[n] + beta2 * x2[n] + beta3 * x3[n], sigma);
  }
}
 '
model_t <- stan_model(model_code = student_t_code)
## Fitting the Model 4
fit_t <- sampling(model_t, data = stan_data3, chains = 4, iter = 2000)
```

```
print(fit_t, pars= c("beta0", "beta1", "beta2", "beta3","sigma","nu"))
```

```
Inference for Stan model: anon_model.
4 chains, each with iter=2000; warmup=1000; thin=1;
post-warmup draws per chain=1000, total post-warmup draws=4000.

       mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
beta0 11.27    0.01 0.30 10.69 11.07 11.26 11.48 11.87  1840    1
beta1  0.01    0.00 0.05 -0.09 -0.02  0.01  0.04  0.10  1762    1
beta2  0.13    0.00 0.02  0.10  0.12  0.13  0.15  0.17  2971    1
beta3  0.42    0.00 0.04  0.33  0.39  0.42  0.45  0.51  2245    1
sigma  0.48    0.00 0.04  0.41  0.45  0.48  0.50  0.56  2536    1
nu     5.70    0.03 1.66  3.16  4.49  5.47  6.65  9.71  2635    1

Samples were drawn using NUTS(diag_e) at Wed Apr  9 16:50:01 2025.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
convergence, Rhat=1).
```

Interpretation for Model 4: In Model 4, I observed that the residual standard deviation dropped to 0.48, which is the lowest among all the models so far—making it the best fit for the data. While beta1 (magnesium) and beta2

(color intensity) still contribute, the most significant factor remains beta3 (proline), with a value of 0.42. By using the Student's t-distribution, the model handles peaks and outliers more effectively—something that was a bit of an issue in the earlier models.

```
log_likt <- extract_log_lik(fit_t, merge_chains = FALSE)
loo_t <- loo(extract_log_lik(fit_t))
print(loo3)
```

```
Computed from 4000 by 178 log-likelihood matrix.

         Estimate   SE
elpd_loo   -149.8  9.2
p_loo         4.6  0.5
looic       299.5 18.4
------
MCSE of elpd_loo is 0.0.
MCSE and ESS estimates assume independent draws (r_eff=1).

All Pareto k estimates are good (k < 0.7).
See help('pareto-k-diagnostic') for details.
```

```
print(loo_t)
```

```
Computed from 4000 by 178 log-likelihood matrix.

         Estimate   SE
elpd_loo   -158.2 13.0
p_loo        10.5  1.5
looic       316.3 25.9
------
MCSE of elpd_loo is 0.1.
MCSE and ESS estimates assume independent draws (r_eff=1).

All Pareto k estimates are good (k < 0.7).
See help('pareto-k-diagnostic') for details.
```

```
loo_compare(loo3, loo_t) #Comapring the models
```

```
       elpd_diff se_diff
model1  0.0       0.0
model2 -8.4       3.8
```

Results : I checked the Pareto k values for both models and found that they were all below 0.7, which means the LOO-CV results are reliable. The MCSE values were also (0, 0) for both models, indicating strong accuracy and stability with minimal variance. When I compared their LOO values, Model 3 (Gaussian) slightly outperformed Model 4 (Student's t), with scores of -149.9 and -158.1, respectively. So, in terms of LOO-CV, Model 3 has a small edge. However, Model 4 has a lower residual standard error (RSE), which suggests a tighter fit. Overall, both models perform very well and are quite close in terms of predictive accuracy and overall performance.