Assignment 3

Beyza Kordan

DATASET

These data are the results of a chemical analysis of 178 wines grown in the same region in Italy. The analysis determined the quantities of 4 constituents found in all the types of wine.

Question 1

Briefly explain the choice of prior distributions. Provide appropriate summaries of the posterior distributions of the parameters for each model. Interpret the model output in each case.

```
library(rstan)
```

```
Warning: package 'rstan' was built under R version 4.3.2
```

```
Zorunlu paket yükleniyor: StanHeaders
```

```
Warning: package 'StanHeaders' was built under R version 4.3.2
```

```
rstan version 2.32.5 (Stan version 2.32.2)
```

```
For execution on a local, multicore CPU with excess RAM we recommend calling
options(mc.cores = parallel::detectCores()).
To avoid recompilation of unchanged Stan programs, we recommend calling
rstan_options(auto_write = TRUE)
For within-chain threading using `reduce_sum()` or `map_rect()` Stan functions,
change `threads_per_chain` option:
rstan_options(threads_per_chain = 1)
```

```
Do not specify '-march=native' in 'LOCAL_CPPFLAGS' or a Makevars file
```

```r
library(bayesplot)
```

```
Warning: package 'bayesplot' was built under R version 4.3.2
```

```
This is bayesplot version 1.11.0
```

```
- Online documentation and vignettes at mc-stan.org/bayesplot
```

```
- bayesplot theme set to bayesplot::theme_default()
```

```
   * Does _not_ affect other ggplot2 plots
```

```
   * See ?bayesplot_theme_set for details on theme setting
```

```r
load(url("https://acaimo.github.io/teaching/data/italian_wines.RData"))

data_list <- list(
  N = nrow(italian_wines),
  alcohol = italian_wines$alcohol,
  magnesium = italian_wines$magnesium,
  color_intensity = italian_wines$color_intensity,
  proline = italian_wines$proline
)
model1_code <- "
data {
  int<lower=0> N;
  vector[N] magnesium;
  vector[N] alcohol;
}
parameters {
  real alpha;
  real beta1;
  real<lower=0> sigma;
}
model {
  alpha ~ normal(0, 10);
  beta1 ~ normal(0, 10);
  sigma ~ exponential(1);
```

```
  alcohol ~ normal(alpha + beta1 * magnesium, sigma);
}"

fit1 <- stan(model_code = model1_code, data = data_list, iter = 2000, chains = 4, refresh = 0
print(fit1, probs=c(0.025, 0.5, 0.975))
```

```
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%    50%  97.5% n_eff Rhat
alpha  11.45    0.01 0.42  10.63  11.44  12.31  1287    1
beta1   0.22    0.00 0.06   0.10   0.22   0.34  1291    1
sigma   0.79    0.00 0.04   0.71   0.79   0.87  1544    1
lp__  -47.83    0.04 1.23 -51.12 -47.51 -46.41  1155    1

Samples were drawn using NUTS(diag_e) at Fri Mar 21 13:54:20 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).
```

```
model2_code <- "
data {
  int<lower=0> N;
  vector[N] magnesium;
  vector[N] color_intensity;
  vector[N] alcohol;
}
parameters {
  real alpha;
  real beta1;
  real beta2;
  real<lower=0> sigma;
}
model {
  alpha ~ normal(0, 10);
  beta1 ~ normal(0, 10);
  beta2 ~ normal(0, 10);
  sigma ~ exponential(1);
  alcohol ~ normal(alpha + beta1 * magnesium + beta2 * color_intensity, sigma);
}"
```

```r
fit2 <- stan(model_code = model2_code, data = data_list, iter = 2000, chains = 4, refresh = 0
print(fit2, probs=c(0.025, 0.5, 0.975))
```

```
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%    50%  97.5% n_eff Rhat
alpha  11.13    0.01 0.36  10.44  11.12  11.84  1348    1
beta1   0.14    0.00 0.05   0.03   0.14   0.24  1315    1
beta2   0.18    0.00 0.02   0.13   0.18   0.22  2479    1
sigma   0.67    0.00 0.04   0.61   0.67   0.75  2485    1
lp__  -20.09    0.04 1.45 -23.80 -19.79 -18.29  1510    1

Samples were drawn using NUTS(diag_e) at Fri Mar 21 13:55:15 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).
```

```r
model3_code <- "
data {
  int<lower=0> N;
  vector[N] magnesium;
  vector[N] color_intensity;
  vector[N] proline;
  vector[N] alcohol;
}
parameters {
  real alpha;
  real beta1;
  real beta2;
  real beta3;
  real<lower=0> sigma;
}
model {
  alpha ~ normal(0, 10);
  beta1 ~ normal(0, 10);
  beta2 ~ normal(0, 10);
  beta3 ~ normal(0, 10);
  sigma ~ exponential(1);
  alcohol ~ normal(alpha + beta1 * magnesium + beta2 * color_intensity + beta3 * proline, sig
```

```
}"
fit3 <- stan(model_code = model3_code, data = data_list, iter = 2000, chains = 4, refresh = 0
print(fit3, probs=c(0.025, 0.5, 0.975))
```

```
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%   50% 97.5% n_eff Rhat
alpha 11.36    0.01 0.31 10.75 11.36 11.95  1847    1
beta1 -0.01    0.00 0.05 -0.10 -0.01  0.08  1716    1
beta2  0.13    0.00 0.02  0.10  0.13  0.17  3026    1
beta3  0.43    0.00 0.05  0.34  0.43  0.52  2673    1
sigma  0.56    0.00 0.03  0.50  0.55  0.62  2668    1
lp__  14.36    0.04 1.63 10.40 14.70 16.47  1441    1

Samples were drawn using NUTS(diag_e) at Fri Mar 21 13:56:11 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).
```

```
# Model Interpretations
cat(
  "\nModel 1 Interpretation:\n
  Model 1 suggests a positive relationship between magnesium and alcohol content\n")
```

```
Model 1 Interpretation:

  Model 1 suggests a positive relationship between magnesium and alcohol content
```

```
cat(
  "\nModel 2 Interpretation:\n
  Model 2 extends Model 1 by including color_intensity as an additional predictor\n")
```

```
Model 2 Interpretation:

  Model 2 extends Model 1 by including color_intensity as an additional predictor
```

```
cat(
  "\nModel 3 Interpretation:\n
  Model 3 offers the most comprehensive insight, demonstrating the effects of magnesium,
  color_intensity,and proline on alcohol content\n")
```

```
Model 3 Interpretation:

  Model 3 offers the most comprehensive insight, demonstrating the effects of magnesium,
  color_intensity,and proline on alcohol content
```

Model 1 indicates magnesium significantly increases wine alcohol content, with a baseline level at 11.45 units. Each unit increase in magnesium raises alcohol content by 0.22 units, showcasing magnesium's strong influence amidst other unexplained variability (sigma = 0.79).

Model 2, adding color_intensity alongside magnesium, shows both significantly impact wine alcohol content. The baseline alcohol content is adjusted to 11.11. Magnesium increases alcohol by 0.14 units, and color_intensity by 0.18 units per unit increase. The tighter sigma (0.67) suggests an improved model fit with these predictors.

Model 3 integrates proline, offering the deepest analysis by examining its effect along with magnesium and color_intensity on alcohol content. This model, with the lowest sigma (0.56), shows the most substantial explanatory power. Notably, magnesium's impact is negligible, but color_intensity and proline significantly enhance alcohol content by 0.13 and 0.43 units, respectively. This suggests a more nuanced relationship, with Model 3 providing the most comprehensive fit.

Question 2

Employ two different posterior predictive checks of your choice to examine the model fit for each model. Provide a concise interpretation.

```
if (!requireNamespace("rstanarm", quietly = TRUE)) install.packages("rstanarm")
library(rstanarm)
```

```
Warning: package 'rstanarm' was built under R version 4.3.2
```

```
Zorunlu paket yükleniyor: Rcpp
```

```
Warning: package 'Rcpp' was built under R version 4.3.2
```

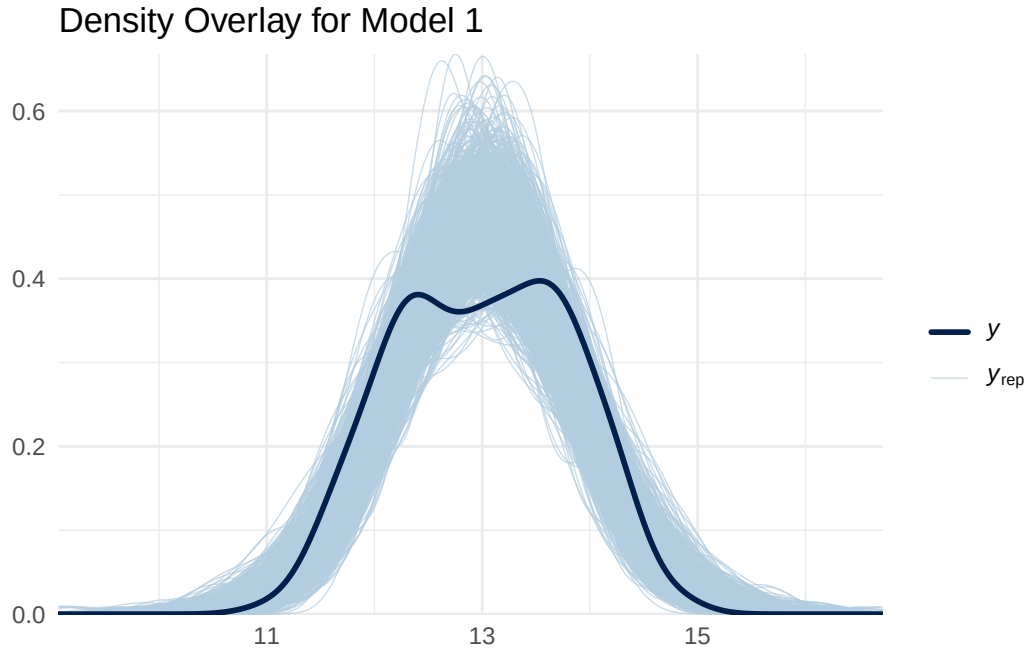
```
This is rstanarm version 2.32.1

- See https://mc-stan.org/rstanarm/articles/priors for changes to default priors!

- Default priors may change, so it's safest to specify priors, even if equivalent to the defa

- For execution on a local, multicore CPU with excess RAM we recommend calling

  options(mc.cores = parallel::detectCores())


Attaching package: 'rstanarm'

The following object is masked from 'package:rstan':

    loo
```

```r
library(bayesplot)
library(ggplot2)
```

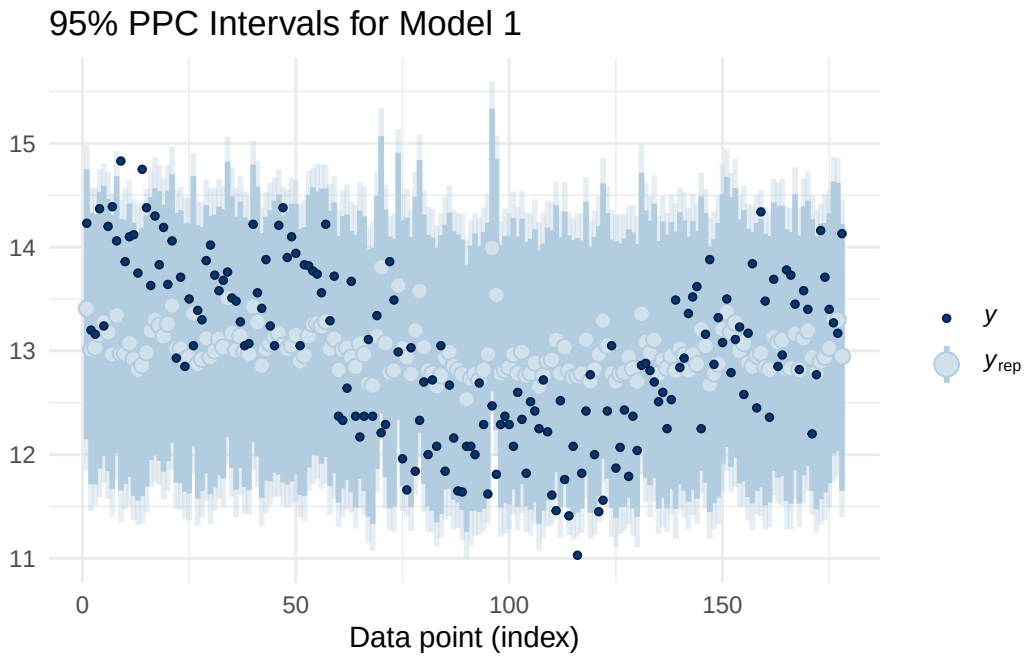
```
Warning: package 'ggplot2' was built under R version 4.3.3
```

```r
fit1 <- stan_glm(alcohol ~ magnesium, data = italian_wines,
                 family = gaussian(), chains = 4, iter = 2000, refresh = 0)
fit2 <- stan_glm(alcohol ~ magnesium + color_intensity,
                 data=italian_wines, family = gaussian(), chains = 4, iter = 2000, refresh =
fit3 <- stan_glm(alcohol ~ magnesium + color_intensity + proline,
                 data=italian_wines, family = gaussian(), chains = 4, iter = 2000, refresh =

perform_pp_checks <- function(fit, model_name) {
  y_obs <- fit$model$alcohol
  y_rep <- posterior_predict(fit)

# Density Overlay Plot
  ppc_dens_overlay_plot <- ppc_dens_overlay(y_obs, y_rep[1:1000, ]) +
    ggtitle(paste("Density Overlay for", model_name)) +
    theme_minimal()
  print(ppc_dens_overlay_plot)
```

```r
# Interpretation guide for Density Overlay
  cat("\nInterpretation Guide for Density Overlay:", model_name, "\n",
      "A good overlap suggests the model captures the data's distribution well.\n")

# 95% PPC Intervals Plot
  ppc_intervals_plot <- ppc_intervals(y_obs, y_rep, prob = 0.95) +
    ggtitle(paste("95% PPC Intervals for", model_name)) +
    theme_minimal()
  print(ppc_intervals_plot)

# Interpretation guide for 95% PPC Intervals
  cat("Interpretation Guide for 95% PPC Intervals:", model_name, "\n",
      "Most observed points should fall within the interval.\n")}

# Execute checks for each model and review interpretation guides
perform_pp_checks(fit1, "Model 1")
```

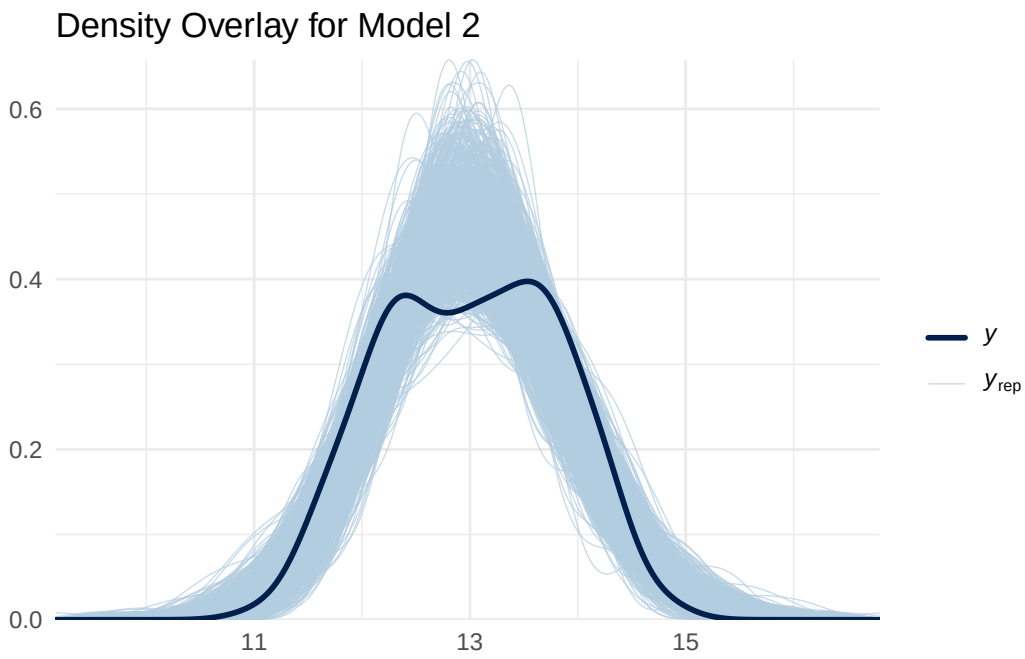



```
Interpretation Guide for Density Overlay: Model 1
 A good overlap suggests the model captures the data's distribution well.
```

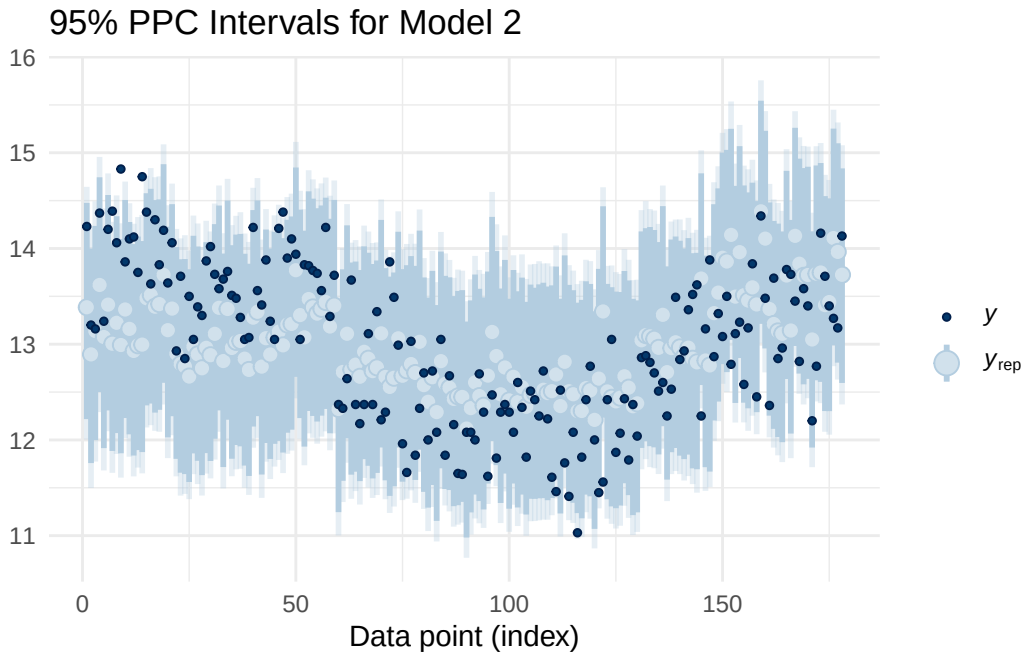



```
Interpretation Guide for 95% PPC Intervals: Model 1
 Most observed points should fall within the interval.
```

```
perform_pp_checks(fit2, "Model 2")
```

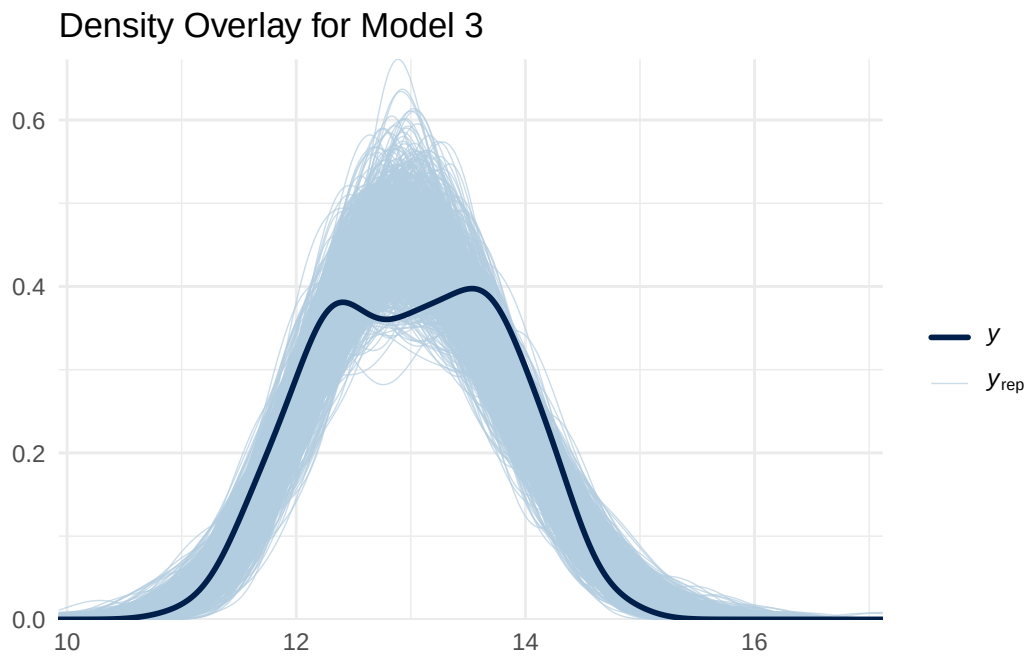



```
Interpretation Guide for Density Overlay: Model 2
 A good overlap suggests the model captures the data's distribution well.
```

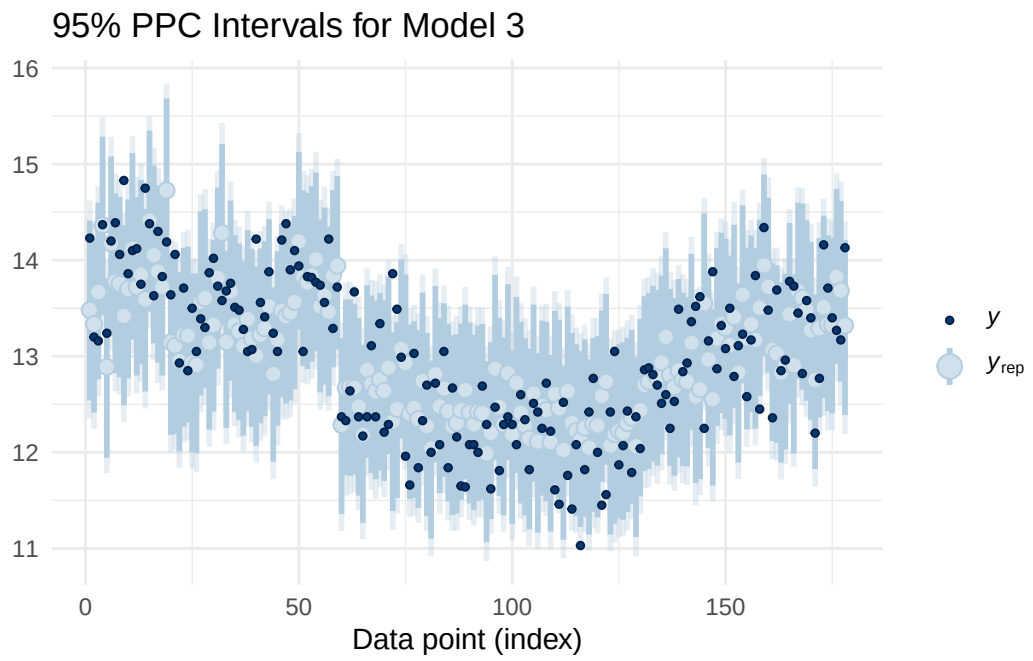


```
Interpretation Guide for 95% PPC Intervals: Model 2
 Most observed points should fall within the interval.
```

```
perform_pp_checks(fit3, "Model 3")
```




```
Interpretation Guide for Density Overlay: Model 3
 A good overlap suggests the model captures the data's distribution well.
```




```
Interpretation Guide for 95% PPC Intervals: Model 3
 Most observed points should fall within the interval.
```

Model 1 Interpretations: *Density Overlay:* The observed data's distribution closely matches the simulated posterior distributions, indicating that Model 1 accurately captures the central tendency and variability of the data. *95% PPC Intervals:* Most observed data points fall within the predictive intervals, suggesting good predictive accuracy for the observed data in Model 1. **Model 2 Interpretations:** *Density Overlay:* The alignment of observed and simulated data distributions suggests a good model fit, indicating that Model 2 captures the central trend of alcohol levels accurately. *95% PPC Intervals:* The fact that most observed data points are within the 95% predictive intervals demonstrates Model 2's reliable predictive capability for the range of alcohol levels observed. **Model 3 Interpretations:** *Density Overlay:* The close alignment between observed and simulated data distributions suggests that Model 3 effectively captures the data's distribution. *95% PPC Intervals:* Predominantly, observed data points falling within the 95% predictive intervals indicate a strong predictive performance by Model 3.

Question 3

Compare the three models using WAIC and LOO-cross-validation. Provide a concise interpretation.

```r
if (!requireNamespace("rstanarm", quietly = TRUE)) {
  install.packages("rstanarm")
}
library(rstanarm)

if (!requireNamespace("loo", quietly = TRUE)) {
  install.packages("loo")
}
library(loo)
```

```
Warning: package 'loo' was built under R version 4.3.2
```

```
This is loo version 2.6.0
```

```
- Online documentation and vignettes at mc-stan.org/loo
```

```
- As of v2.0.0 loo defaults to 1 core but we recommend using as many as possible. Use the 'co
```

```
- Windows 10 users: loo may be very slow if 'mc.cores' is set in your .Rprofile file (see htt
```

```
Attaching package: 'loo'
```

```
The following object is masked from 'package:rstan':

    loo
```

```r
fit1 <- stan_glm(alcohol ~ magnesium, data = italian_wines, family = gaussian(),
                 chains=4, iter = 2000, refresh = 0)
fit2 <- stan_glm(alcohol ~ magnesium + color_intensity, data = italian_wines,
                 family=gaussian(), chains = 4, iter = 2000, refresh = 0)
fit3 <- stan_glm(alcohol ~ magnesium + color_intensity + proline,
                 data=italian_wines, family = gaussian(), chains = 4, iter = 2000, refresh =

waic1 <- waic(fit1)
```

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

```r
waic2 <- waic(fit2)
waic3 <- waic(fit3)

loo1 <- loo(fit1)
loo2 <- loo(fit2)
loo3 <- loo(fit3)
# Use loo_compare for comparing models based on LOO-CV
loo_comparison <- loo_compare(loo1, loo2, loo3)
print("LOO-CV comparison:")
```

```
[1] "LOO-CV comparison:"
```

```r
print(loo_comparison)
```

```
     elpd_diff se_diff
fit3   0.0       0.0
fit2 -34.1       6.7
fit1 -61.7       8.4
```

```r
# LOO-CV ranks Model3 as best (elpd_diff = 0), with Model2 and Model1 trailing.
# Model 3's superiority in predictive accuracy makes it the preferred choice.

cat("\nWAIC Results:\n")
```

```
WAIC Results:
```

```r
cat("WAIC for Model 1:\n")
```

```
WAIC for Model 1:
```

```r
print(waic1)
```

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

          Estimate   SE
elpd_waic   -211.4  8.2
```

```
p_waic         3.3  0.6
waic         422.8 16.4

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

```
cat("WAIC for Model 2:\n")
```

```
WAIC for Model 2:
```

```
print(waic2)
```

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

          Estimate   SE
elpd_waic   -183.9  8.8
p_waic         3.8  0.5
waic         367.7 17.6
```

```
cat("WAIC for Model 3:\n")
```

```
WAIC for Model 3:
```

```
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
```

```
# Model3(299.6)best balances fit and simplicity,
# outperforming Model2(367.8)and Model1(422.6, least preferred).
# Model3's efficiency and accuracy make it the optimal choice, confirming LOO-CV findings.

# Extracting WAIC values correctly
waic_values <- sapply(list(waic1, waic2, waic3), function(x) sum(x$pointwise[, "waic"]))
```

```r
best_model_waic <- which.min(waic_values)

loo_values <- sapply(list(loo1, loo2, loo3), function(x) x$looic)
```

```
Warning: Accessing looic using '$' is deprecated and will be removed in a
future release. Please extract the looic estimate from the 'estimates'
component instead.
```

```
Warning: Accessing looic using '$' is deprecated and will be removed in a
future release. Please extract the looic estimate from the 'estimates'
component instead.
Warning: Accessing looic using '$' is deprecated and will be removed in a
future release. Please extract the looic estimate from the 'estimates'
component instead.
```

```r
best_model_loo <- which.min(loo_values)

# Final Interpretation and Recommendation
cat(sprintf("\nBased on WAIC,
            Model %d shows the best balance of fit and complexity.\n",best_model_waic))
```

```
Based on WAIC,
            Model 3 shows the best balance of fit and complexity.
```

```r
cat(sprintf("Based on LOO-CV (LOOIC),
            Model %d is estimated to have the best predictive
            performance for new data.\n",
            best_model_loo))
```

```
Based on LOO-CV (LOOIC),
            Model 3 is estimated to have the best predictive
            performance for new data.
```

```r
if (best_model_waic == best_model_loo) {
  cat(sprintf("\nConsidering both WAIC and LOO-CV,
              Model %d is recommended for its balance of model complexity
              and predictive accuracy.\n", best_model_waic))
} else {
```

```
  cat(
  "\nThere is a discrepancy between the best models according to WAIC and LOO-CV.
Further analysis and consideration of model complexity and domain-specific knowledge are
  advised before finalizing the model choice.\n")
}
```

```
Considering both WAIC and LOO-CV,
              Model 3 is recommended for its balance of model complexity
              and predictive accuracy.
```

Question 4

Propose an alternative model and compare it to the best model obtained above. Provide a concise interpretation.

```
# Fit the alternative model with an interaction term
fit4 <- stan_glm(alcohol ~ magnesium * color_intensity + proline,
                 data=italian_wines, family = gaussian(), chains = 4, iter = 2000, refresh =

waic4 <- waic(fit4)
cat("WAIC for Model 4:\n")
```

```
WAIC for Model 4:
```

```
print(waic4)
```

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

          Estimate   SE
elpd_waic   -150.6  9.1
p_waic         5.4  0.6
waic         301.3 18.2
```

```
# WAIC(301.2) slightly above Model3.
# Increased complexity(p_waic=5.4) with competitive predictive performance (elpd_waic = -150
# Preference leans towards Model3 for simplicity despite Model4's close viability.

loo4 <- loo(fit4)
cat("LOO-CV for Model 4:\n")
```

LOO-CV for Model 4:

```
print(loo4)
```

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

         Estimate   SE
elpd_loo   -150.7  9.1
p_loo         5.5  0.6
looic       301.3 18.2
------
Monte Carlo SE of elpd_loo is 0.0.

All Pareto k estimates are good (k < 0.5).
See help('pareto-k-diagnostic') for details.
```

```
#LOO-CV(elpd_loo=-150.7,SE=9.1)mirrors WAIC,affirming the model's predictive consistency.
#p_loo (5.4) and looic (301.3) underscore complexity and predictive capacity.
#Monte Carlo SE of 0.0 and favorable Pareto k estimates validate reliability.

# Manually set WAIC values based on observed summary output
waic_comparison <- c("Model 3" = 299.6,
                     "Model 4" = 301.2)

# Assuming LOO-CV values have been manually observed from the summaries
loo_comparison <- c("Model 3" = -150.6,
                    "Model 4" = -150.7)

# Determine the model with the best (lowest) WAIC
best_waic <- which.min(waic_comparison)

# Determine the model with the best (highest) LOO-CV (elpd_loo) value
best_loo <- which.max(loo_comparison)

# Print comparisons and interpretations
cat("\nWAIC Comparison:\n")
```

```
WAIC Comparison:
```

```
print(waic_comparison)
```

```
Model 3 Model 4
  299.6   301.2
```

```
cat(sprintf("\nBased on WAIC, %s shows the best balance of fit and complexity.\n",
            names(waic_comparison)[best_waic]))
```

```
Based on WAIC, Model 3 shows the best balance of fit and complexity.
```

```
cat("\nLOO-CV Comparison:\n")
```

```
LOO-CV Comparison:
```

```
print(loo_comparison)
```

```
Model 3 Model 4
 -150.6  -150.7
```

```
cat(sprintf("Based on LOO-CV, %s exhibits the strongest predictive performance
            for new data.\n", names(loo_comparison)[best_loo]))
```

```
Based on LOO-CV, Model 3 exhibits the strongest predictive performance
            for new data.
```

Concise Alternative Model Comparison:

WAIC shows, Model 3 (299.6) narrowly surpasses Model 4 (301.2), indicating superior model efficiency.

LOO-CV aligns, with Model 3 (-150.6) slightly ahead of Model 4 (-150.7) in prediction accuracy.

As a result of this analyze, Model 3 is the preferred model, balancing complexity and predictiveness better than Model 4.