Fall 2024
Homework 3
STAT 656: Bayesian Data Analysis

Juanwu Lu*

Gibbs Sampling for Normal Hierarchical Models

In Lecture 7, we modeled the NBA data with a normal hierarchical model with known variances as follows:

$$p(\mu, \tau^2) = p(\mu|\tau^2)p(\tau^2) \propto p(\tau^2)$$

$$\theta_j|\mu, \tau^2 \overset{\text{i.i.d}}{\sim} \mathcal{N}(\mu, \tau^2), j = 1, \ldots, J$$

$$y_{ij}|\theta, \sigma^2, \mu, \tau^2 \overset{\perp\!\!\!\perp}{\sim} \mathcal{N}(\theta_j, \sigma_j^2), \quad i = 1, \ldots, n_j.$$

Now we will extend this to allow the variances σ_j^2 to be unknown as well. We will consider two models:

1. **Unknown but identical variances**: $\sigma_1^2 = \ldots = \sigma_J^2 = \sigma^2$, with $\sigma^2 \sim p_v$
2. **Unknown and independent variances**: $\sigma_1^2, \ldots, \sigma_J^2 \overset{\text{i.i.d}}{\sim} p_v$.

For both models, set $p_v(\sigma^2) \propto 1/\sigma$. Your task is to implement a Gibbs sampler to simulate all latent variables given the data from `nba_data.csv`. The Gibbs sampler involves the following steps for each iteration:

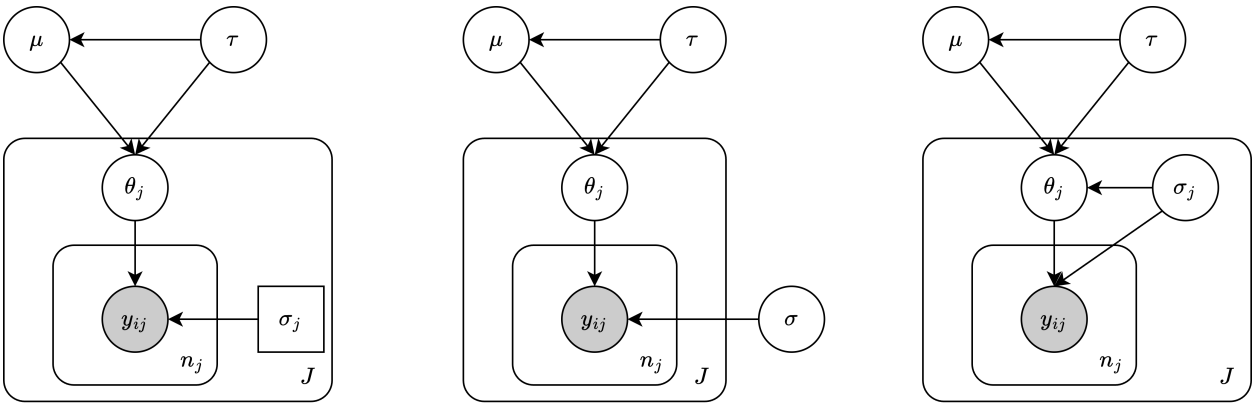
- (*For both models*) Sample from $\mu, \tau^2|\theta_1, \ldots, \theta_J, \sigma_1^2, \ldots, \sigma_J^2, Y$, and then
- (*For model 1*) Sample from $\theta_1, \ldots, \theta_J|\sigma^2, \mu, \tau^2, Y$, and then $\sigma^2|\theta_1, \ldots, \theta_J, \mu, \tau^2, Y$
- (*For model 2*) Sample from $(\theta_1, \sigma_1^2), \ldots, (\theta_J, \sigma_J^2)|\mu, \tau^2, Y$.

For full credit, for each model, you need to:

1. (15 points) Write down the form of the conditional distributions above.
2. (25 points) Write down R code to implement the Gibbs samplers.
3. (30 points) Run the two Gibbs samplers on the NBA dataset and summarize the results. Specifically, for μ, τ^2 and a few θ_j and σ_j, plot the MCMC traceplots, and diagnose mixing. Also plot the corresponding posterior distributions.
4. (30 points) Comment on how the posterior distributions differ from each other. Use each model to make predictions on the NBA games from the rest of the season (following code from Lecture 7), and comment on which model (or the model with known variances from Lecture 7) performs best.

*College of Engineering, Purdue University, West Lafayette, IN, USA

Solution:



(a) Model from Lecture 7 (b) Model 1 from Problem (c) Model 2 from Problem

Figure 1: Probabilistic graphical model for the NBA data.

First, we load in the data. Instead of recording the score for each game $y_{ij}, i = 1, \ldots, n_j$, we can see from the data that it only gives the average score $\bar{y}_j = \frac{1}{n_j} \sum_{i=1}^{n_j} y_{ij}$ for each team $j = 1, \ldots, J$.

```
if (file.exists("data/NBA_data.csv")) {
  df <- read.csv("data/NBA_data.csv")[, -1]
} else {
  stop("FileNotFound: data file not found at 'data/NBA_data.csv'.")
}
head(df)
```

```
##   number_games_played    Y_bar sample_sd number_remaining_games
## 1                  10 108.0000  5.868939                     72
## 2                  10 107.2000  8.841820                     72
## 3                   8 106.8750  6.957781                     74
## 4                  10 104.8000 11.545080                     72
## 5                  10 103.8000 13.870830                     72
## 6                   9 103.4444 14.292580                     73
##   average_remaining_games        team
## 1               110.50556     Phoenix
## 2               100.68556        Utah
## 3               105.18514      Denver
## 4               106.78167 GoldenState
## 5                94.70028     Boston
## 6               104.45041 Washington
```

As illustrated by Figure 1, the conditional structure among μ, τ, and $\theta_j, j = 1, \ldots, J$ are exact in the model from lecture 7 and models in this problem. According to Bayes' theorem, we have the following conditional distribution:

$$\begin{aligned}\log p(\mu, \tau^2 \mid \boldsymbol{\theta}, \boldsymbol{\sigma}^2, Y) &= \log p(\mu, \tau^2) + \sum_{j=1}^{J} \log p(\theta_j \mid \mu, \tau^2) + \text{constant} \\ &= -\log(\tau) - \frac{1}{2} \sum_{j=1}^{J} \left[\log(\tau^2) + \frac{(\theta_j - \mu)^2}{\tau^2} \right],\end{aligned}$$

where the constant is a term that is irrelevant to either μ or τ. Herein, we can complete the square of μ by

$$\sum_{j=1}^{J}(\theta_j-\mu)^2=\sum_{j=1}^{J}(\theta_j-\bar{\theta})^2+J(\mu-\bar{\theta})^2,\quad \text{where } \bar{\theta}=\frac{1}{J}\sum_{j=1}^{J}\theta_j.$$

Therefore, the log-posterior density of (μ,τ^2) writes

$$\log p(\mu,\tau^2\mid\boldsymbol{\theta},\boldsymbol{\sigma}^2,Y)=\log p(\mu,\tau^2|\boldsymbol{\theta})=-\log(\tau)-\frac{J}{2}\log(\tau^2)-\frac{1}{2\tau^2}\left[\sum_{j=1}^{J}(\theta_j-\bar{\theta})^2+J(\mu-\bar{\theta})^2\right].$$

By factorizing terms that are related to μ and τ^2 only, we can derive the conditional distribution of μ and τ^2 as follows:

$$\begin{aligned}&p(\mu\mid\tau^2,\boldsymbol{\theta},\boldsymbol{\sigma}^2,Y)\sim\mathcal{N}(\bar{\theta},\tau^2/J),\\&p(\tau^2\mid\mu,\boldsymbol{\theta},\boldsymbol{\sigma}^2,Y)\sim\text{Scal-inv-}\chi^2\left(J,\frac{1}{J}\sum_{j=1}^{J}(\theta_j-\mu)^2\right).\end{aligned}\tag{(1)}$$

For the model lecture, as illustrated by Figure 1(a), we have $\bar{y}_j\sim\mathcal{N}\left(\theta_j,\frac{1}{n_j}\sigma_j^2\right)$, where $\bar{y}_j=\frac{1}{n_j}\sum_{i=1}^{n_j}y_{ij}$. Therefore, for model 1 in Figure 1(b), the conditional distribution of θ_j can be derived as follows:

$$\begin{aligned}\log p(\theta_1,\ldots,\theta_j\mid\mu,\tau^2,\sigma^2,Y)&=\sum_{j=1}^{J}\log p(\theta_j|\mu,\tau^2)+\log p(\bar{y}_j|\theta_j,\sigma_j^2)+\text{constant}\\&=-\frac{1}{2}\sum_{j=1}^{J}\frac{(\bar{y}_j-\theta_j)^2}{\sigma^2/n_j}+\frac{(\theta_j-\mu)^2}{\tau^2}+\text{constant},\end{aligned}$$

where the constant is a term that is irrelevant to any of the θ_j. Similarly, we can complete the square of θ_j by

$$\frac{(\bar{y}_j-\theta_j)^2}{\sigma^2/n_j}+\frac{(\theta_j-\mu)^2}{\tau^2}=\left(\frac{n_j}{\sigma^2}+\frac{1}{\tau^2}\right)\cdot\left(\theta_j-\left(\frac{n_j}{\sigma^2}+\frac{1}{\tau^2}\right)^{-1}\cdot\left(\frac{n_j\bar{y}_j}{\sigma^2}+\frac{\mu}{\tau^2}\right)\right)^2.$$

Therefore, the conditional distribution of θ_j writes $\theta_j\sim\mathcal{N}(\hat{\mu}_j,\sigma_\mu^2)$, where

$$\begin{aligned}\hat{\mu}_j&=\frac{\tau^2\bar{y}_j+\sigma^2\mu/n_j}{\tau^2+\sigma^2/n_j},\\\sigma_\mu^2&=\left(\frac{n_j}{\sigma^2}+\frac{1}{\tau^2}\right)^{-1}.\end{aligned}$$

The conditional posterior distribution of σ^2 can be derived as follows:

$$\begin{aligned}\log p(\sigma^2\mid\mu,\tau^2,\boldsymbol{\theta},Y)&=\log p_v+\log p(Y\mid\theta,\sigma^2)+\text{constant}\\&=-\log(\sigma)-\frac{1}{2}\sum_{j=1}^{J}\sum_{i=1}^{n_j}\left[\log(\sigma^2)+\frac{(y_{ij}-\theta_j)^2}{\sigma^2}\right]+\text{constant},\end{aligned}$$

where the constant is a term that is irrelevant to σ^2. Therefore, the conditional distribution of σ^2 writes

$$p(\sigma^2\mid\mu,\tau^2,\boldsymbol{\theta},Y)\sim\text{Scale-inv-}\chi^2\left(\sum_{j=1}^{J}n_j,\frac{1}{\sum_{j=1}^{J}n_j}\sum_{j=1}^{J}\sum_{i=1}^{n_j}(y_{ij}-\theta_j)^2\right).$$

The implementation of the Gibbs sampler for the two models is as follows:

```r
gibbs_sampler_1 <- function(y_bars, n_j, sigma_j_sq, n_iter, burn) {
  # Get number of teams
  n_teams <- length(y_bars)

  # Initialize the Gibbs sampler
  pstr_mu <- rep(NA, nrow = n_iter)
  pstr_mu[1] <- mean(y_bars)

  pstr_tau_sq <- rep(NA, nrow = n_iter)
  pstr_tau_sq[1] <- 1

  pstr_theta <- matrix(NA, nrow = n_iter, ncol = n_teams)
  pstr_theta[1, ] <- y_bar

  pstr_sigma_sq <- rep(NA, nrow = n_iter)
  pstr_sigma_sq[1] <- 1

  # Run the Gibbs sampler
  for (i in 2:n_iter) {
    # Update mu
    pstr_mu[i] <- (
      mean(pstr_theta[i - 1, ]) + rnorm(1) * sqrt(pstr_tau_sq[i - 1] / n_teams)
    )

    # Update tau-square
    tau_scale <- sum((pstr_theta[i - 1, ] - pstr_mu[i - 1])^2) / (n_teams - 1)
    pstr_tau_sq[i] <- (n_teams - 1) * tau_scale / rchisq(1, n_teams - 1)

    # Update theta and sigma-square
    prcsn <- 1 / pstr_tau_sq[i] + n_j / pstr_sigma_sq[i - 1]
    mu_hat <- (
      pstr_mu[i] / pstr_tau_sq[i] + n_j * y_bars / pstr_sigma_sq[i - 1]
    ) / prcsn
    pstr_theta[i, ] <- mu_hat + rnorm(n_teams) / sqrt(prcsn)

    sigma_sq_scale <- (
      sum(n_j * sigma_j_sq + n_j * ((pstr_theta[i] - y_bar)^2)) / sum(n_j)
    )
    pstr_sigma_sq[i] <- sigma_sq_scale * sum(n_j) / rchisq(1, sum(n_j))
  }

  return(
    list(
      tau = sqrt(pstr_tau_sq)[(burn + 1):n_iter],
      mu = pstr_mu[(burn + 1):n_iter],
      theta = pstr_theta[(burn + 1):n_iter, ],
      sigma = sqrt(pstr_sigma_sq)[(burn + 1):n_iter]
    )
  )
}

# Extract the required variables
n <- df$number_games_played
```

```r
sigma_sq <- df$sample_sd
y_bar <- df$Y_bar

# Run the Gibbs sampler for model 1
set.seed(42)
m1_smpls <- gibbs_sampler_1(
  y_bars = y_bar,
  n_j = n,
  sigma_j_sq = sigma_sq,
  n_iter = 5000,
  burn = 3000
)
```

For model 2, as illustrated by Figure 1(c), we have a conjugate structure between (θ_j, σ_j) and y_{ij}. Therefore, the conditional distribution of (θ_j, σ_j) can be derived as follows:

$$\begin{aligned}\log p(\theta_j, \sigma_j^2 \mid \mu, \tau^2, Y) &= \sum_{j=1}^{J} \log p_v + \log p(\theta_j|\mu, \tau^2\sigma^2) + \log p(\bar{y}_j \mid \theta_j, \sigma_j^2) + \text{constant} \\ &= -\sum_{j=1}^{J} \log(\sigma_j) - \frac{1}{2}\sum_{j=1}^{J}\sum_{i=1}^{n_j}\left[\log(\sigma_j^2) + \frac{(y_{ij}-\theta_j)^2}{\sigma_j^2} + \log(\tau^2\sigma_j^2) + \frac{(\theta_j-\mu)^2}{2\tau^2\sigma_j^2}\right] + \text{constant},\end{aligned}$$

where the constant is a term that is irrelevant to either θ_j or σ_j. By completing the square of θ_j and σ_j^2, we can derive the conditional distribution of (θ_j, σ_j^2) as follows:

$$\begin{aligned}p(\theta_j \mid \sigma_j^2, \mu, \tau^2, Y) &\sim \mathcal{N}\left(\frac{\tau^2\bar{y}_j + \mu/n_j}{\tau^2 + 1/n_j}, \left(\frac{n_j}{\sigma^2} + \frac{1}{\tau^2\sigma^2}\right)^{-1}\right) \\ p(\sigma_j^2 \mid \theta_j, \mu, \tau^2, Y) &\sim \text{Scale-inv-}\chi^2\left(n_j, \frac{1}{n_j}\left[\frac{(\theta_j-\mu)^2}{\tau^2} + \sum_{i=1}^{n_j}(y_{ij}-\theta_j)^2\right]\right).\end{aligned}$$

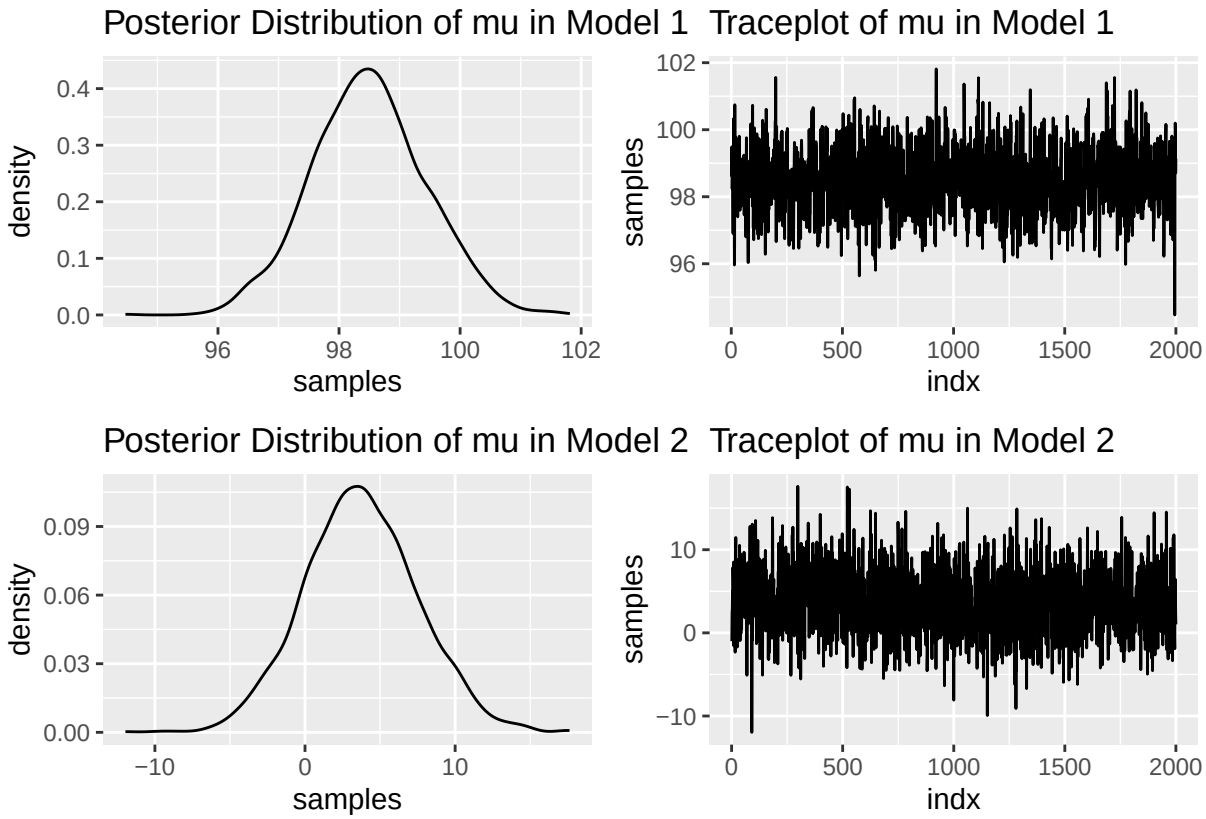
```r
gibbs_sampler_2 <- function(y_bars, n_j, sigma_j_sq, n_iter, burn) {
  # Get number of teams
  n_teams <- length(y_bars)

  # Initialize the Gibbs sampler
  pstr_mu <- numeric(n_iter)
  pstr_mu[1] <- mean(y_bars)

  pstr_tau_sq <- numeric(n_iter)
  pstr_tau_sq[1] <- 1

  pstr_theta <- matrix(NA, nrow = n_iter, ncol = n_teams)
  pstr_theta[1, ] <- y_bars

  pstr_sigma_sq <- matrix(NA, nrow = n_iter, ncol = n_teams)
  pstr_sigma_sq[1, ] <- sigma_j_sq

  # Run the Gibbs sampler
  for (i in 2:n_iter) {
    # Update mu
    pstr_mu[i] <- (
      mean(pstr_theta[i - 1, ]) + rnorm(1) * sqrt(pstr_tau_sq[i - 1] / n_teams)
```

```
    )

    # Update tau-square
    tau_scale <- sum((pstr_theta[i - 1, ] - pstr_mu[i - 1])^2) / (n_teams - 1)
    pstr_tau_sq[i] <- (n_teams - 1) * tau_scale / rchisq(1, n_teams - 1)

    # Update theta and sigma-square
    prcsn <- (
      1 / pstr_tau_sq[i] / pstr_sigma_sq[i - 1] + n_j / pstr_sigma_sq[i - 1]
    )
    mu_hat <- (
      pstr_mu[i] / pstr_tau_sq[i] / pstr_sigma_sq[i - 1]
      + n_j * y_bars / pstr_sigma_sq[i - 1, ]
    ) / prcsn
    pstr_theta[i, ] <- mu_hat + rnorm(n_teams) / sqrt(prcsn)

    for (j in 1:n_teams) {
      sigma_sq_scale <- (
        n_j[j] * sigma_j_sq[j]
        + n_j[j] * (pstr_theta[i, j] - y_bars[j]) ^ 2
        + (pstr_theta[i, j] - pstr_mu[i]) ^ 2 / pstr_tau_sq[i]
      )
      pstr_sigma_sq[i, j] <- sigma_sq_scale / rchisq(1, n_j[j])
    }


  }
  return(
    list(
      tau = sqrt(pstr_tau_sq)[(burn + 1):n_iter],
      mu = pstr_mu[(burn + 1):n_iter],
      theta = pstr_theta[(burn + 1):n_iter, ],
      sigma = sqrt(pstr_sigma_sq)[(burn + 1):n_iter, ]
    )
  )
}

# Extract the required variables
n <- df$number_games_played
sigma_sq <- df$sample_sd
y_bar <- df$Y_bar

# Run the Gibbs sampler for model 1
set.seed(42)
m2_smpls <- gibbs_sampler_2(
  y_bars = y_bar,
  n_j = n,
  sigma_j_sq = sigma_sq,
  n_iter = 5000,
  burn = 3000
)
```

The following visualization shows the posterior distributions and the MCMC trace plots of μ in model 1 and model 2. The trace plots show that the MCMC chains are well-mixed and converged. The posterior

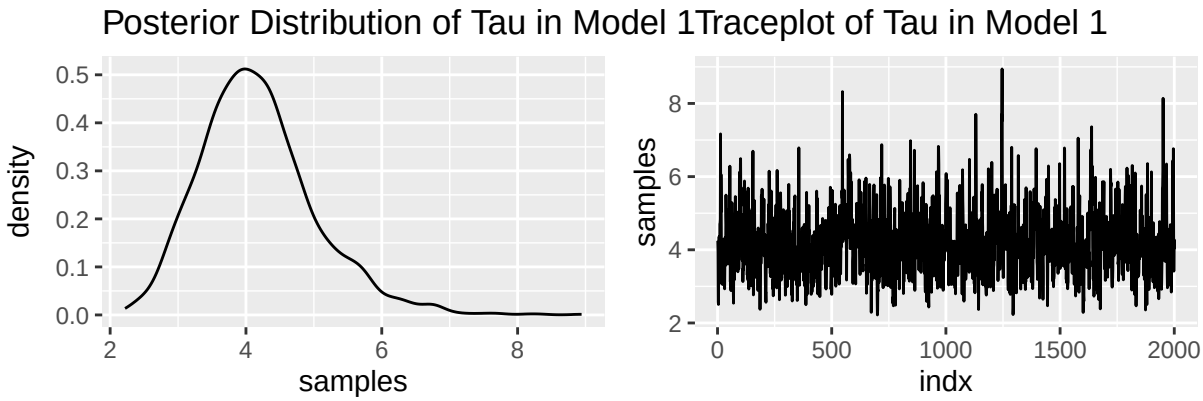
distributions show that the posterior distribution of μ in model 1 is more concentrated than that in model 2. The posterior distribution of τ in model 1 is more concentrated than that in model 2. The posterior distribution of θ_1 in model 1 is more concentrated than that in model 2. The posterior distribution of σ_1 in model 1 is more concentrated with an higher average than that in model 2.

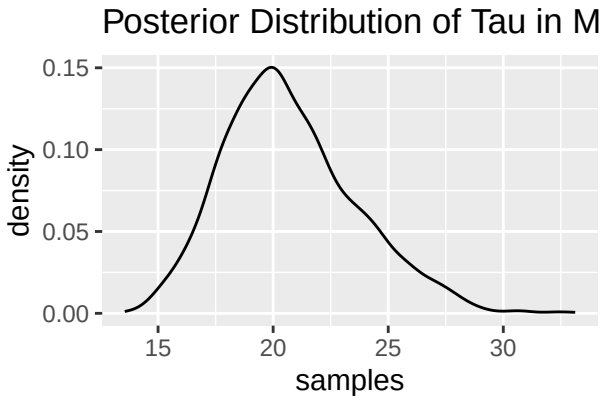
```
# Plot the MCMC trace plots
m1_df <- data.frame(indx = 1:2000, samples = m1_smpls$mu)
m2_df <- data.frame(indx = 1:2000, samples = m2_smpls$mu)
plot1 <- (
  ggplot(m1_df)
    + geom_density(aes(x=samples))
    + labs(title = "Posterior Distribution of mu in Model 1")
)
plot2 <- (
  ggplot(m1_df)
    + geom_line(aes(x=indx, y=samples))
    + labs(title = "Traceplot of mu in Model 1")
)
plot3 <- (
  ggplot(m2_df)
    + geom_density(aes(x=samples))
    + labs(title = "Posterior Distribution of mu in Model 2")
)
plot4 <- (
  ggplot(m2_df)
    + geom_line(aes(x=indx, y=samples))
    + labs(title = "Traceplot of mu in Model 2")
)
(plot1 + plot2) / (plot3 + plot4)
```

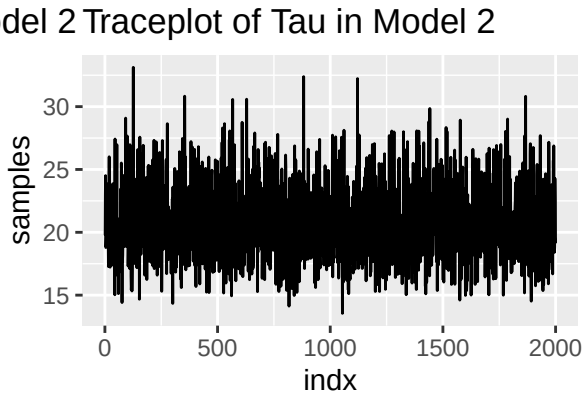


The following visualization shows the posterior distributions and the MCMC trace plots of τ in model 1 and model 2. The trace plots show that the MCMC chains are well-mixed and converged. The posterior distributions show that the posterior distribution of τ in model 1 is overall lower than that in model 2.

```
# Plot the MCMC trace plots
m1_df <- data.frame(indx = 1:2000, samples = m1_smpls$tau)
m2_df <- data.frame(indx = 1:2000, samples = m2_smpls$tau)
plot1 <- (
  ggplot(m1_df)
    + geom_density(aes(x=samples))
    + labs(title = "Posterior Distribution of Tau in Model 1")
)
plot2 <- (
  ggplot(m1_df)
    + geom_line(aes(x=indx, y=samples))
    + labs(title = "Traceplot of Tau in Model 1")
)
plot3 <- (
  ggplot(m2_df)
    + geom_density(aes(x=samples))
    + labs(title = "Posterior Distribution of Tau in Model 2")
)
plot4 <- (
  ggplot(m2_df)
    + geom_line(aes(x=indx, y=samples))
    + labs(title = "Traceplot of Tau in Model 2")
)
(plot1 + plot2) / (plot3 + plot4)
```



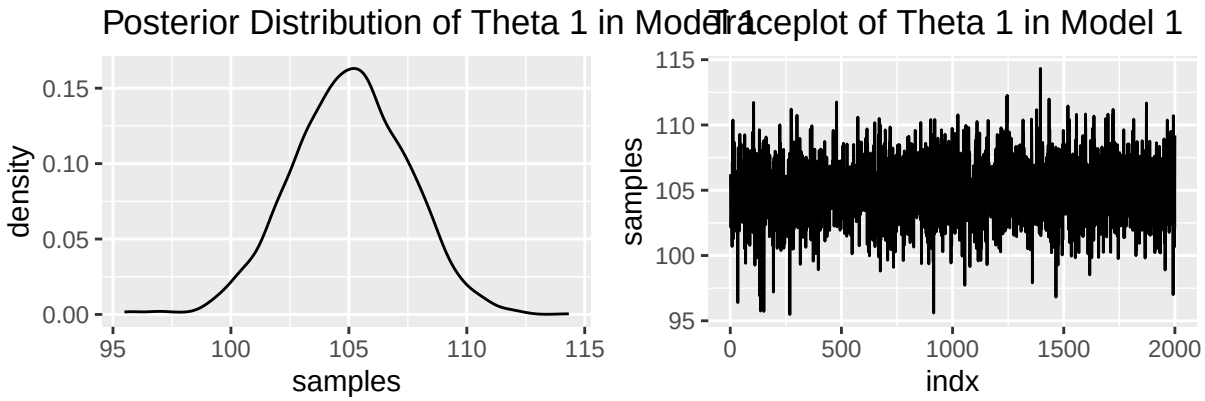


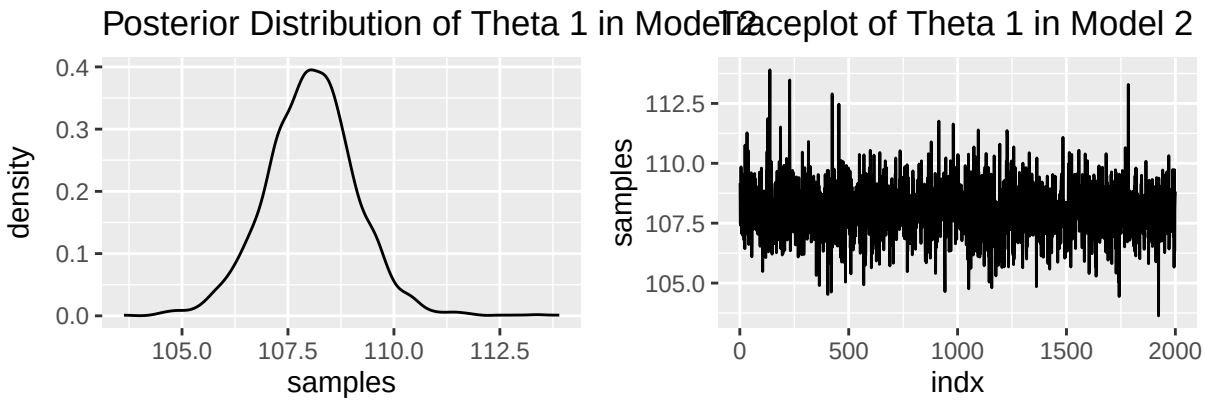


The following visualization shows the posterior distributions and the MCMC trace plots of θ_1 in model 1 and model 2. The trace plots show that the MCMC chains are well-mixed and converged. The posterior distributions show that the posterior distribution of θ_1 in model 2 is more concentrated than that in model 1. The mean of the two distribution is similar.

```
# Plot the MCMC trace plots
m1_df <- data.frame(indx = 1:2000, samples = m1_smpls$theta[, 1])
m2_df <- data.frame(indx = 1:2000, samples = m2_smpls$theta[, 1])
plot1 <- (
  ggplot(m1_df)
    + geom_density(aes(x=samples))
    + labs(title = "Posterior Distribution of Theta 1 in Model 1")
)
plot2 <- (
  ggplot(m1_df)
    + geom_line(aes(x=indx, y=samples))
    + labs(title = "Traceplot of Theta 1 in Model 1")
)
plot3 <- (
  ggplot(m2_df)
    + geom_density(aes(x=samples))
    + labs(title = "Posterior Distribution of Theta 1 in Model 2")
)
plot4 <- (
  ggplot(m2_df)
    + geom_line(aes(x=indx, y=samples))
    + labs(title = "Traceplot of Theta 1 in Model 2")
)
```

```
(plot1 + plot2) / (plot3 + plot4)
```

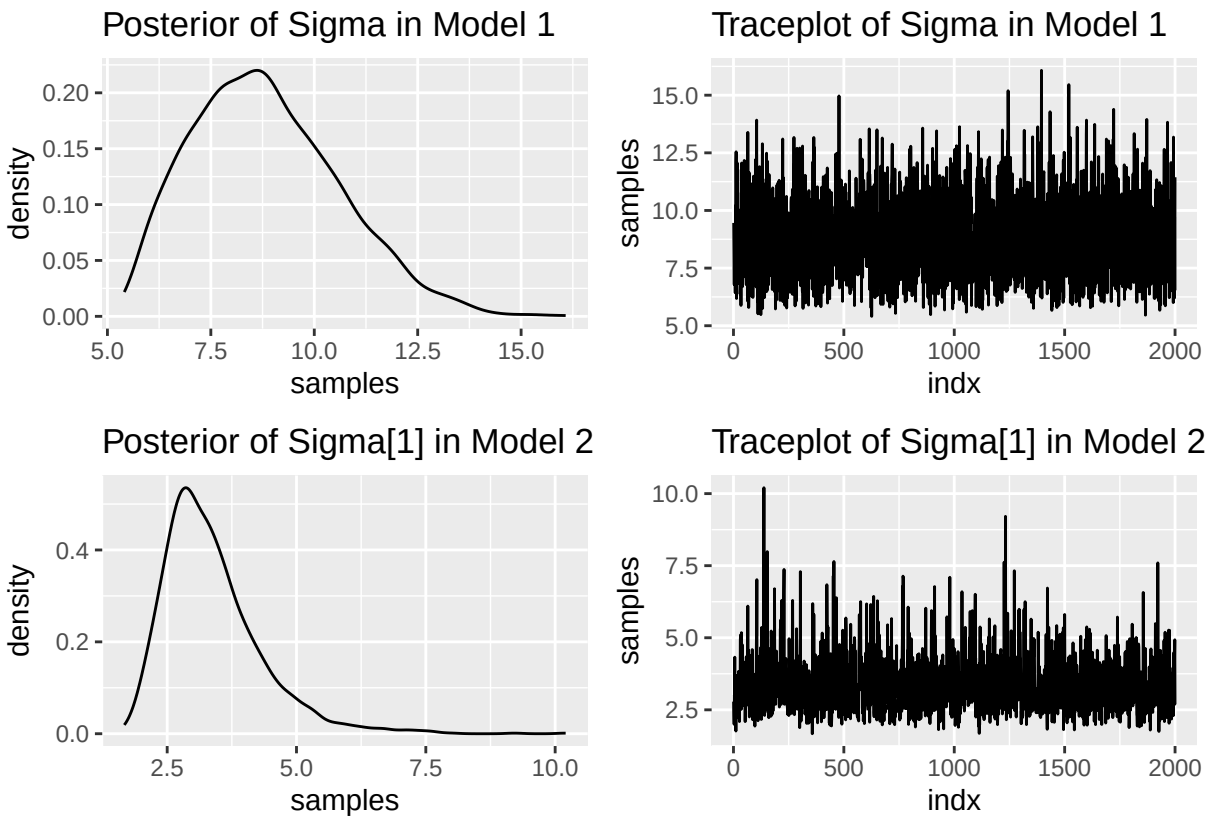




The following visualization shows the posterior distributions and the MCMC trace plots of σ_1 in model 1 and model 2. The trace plots show that the MCMC chains are well-mixed and converged. The posterior distributions show that the posterior distribution of σ_1 in model 2 is more concentrated with an lower mode than that in model 1.

```
# Plot the MCMC trace plots
m1_df <- data.frame(indx = 1:2000, samples = m1_smpls$sigma)
m2_df <- data.frame(indx = 1:2000, samples = m2_smpls$sigma[, 1])
plot1 <- (
  ggplot(m1_df)
    + geom_density(aes(x=samples))
    + labs(title = "Posterior of Sigma in Model 1")
)
plot2 <- (
  ggplot(m1_df)
    + geom_line(aes(x=indx, y=samples))
    + labs(title = "Traceplot of Sigma in Model 1")
)
plot3 <- (
  ggplot(m2_df)
    + geom_density(aes(x=samples))
    + labs(title = "Posterior of Sigma[1] in Model 2")
)
plot4 <- (
  ggplot(m2_df)
    + geom_line(aes(x=indx, y=samples))
    + labs(title = "Traceplot of Sigma[1] in Model 2")
```

```
)
(plot1 + plot2) / (plot3 + plot4)
```

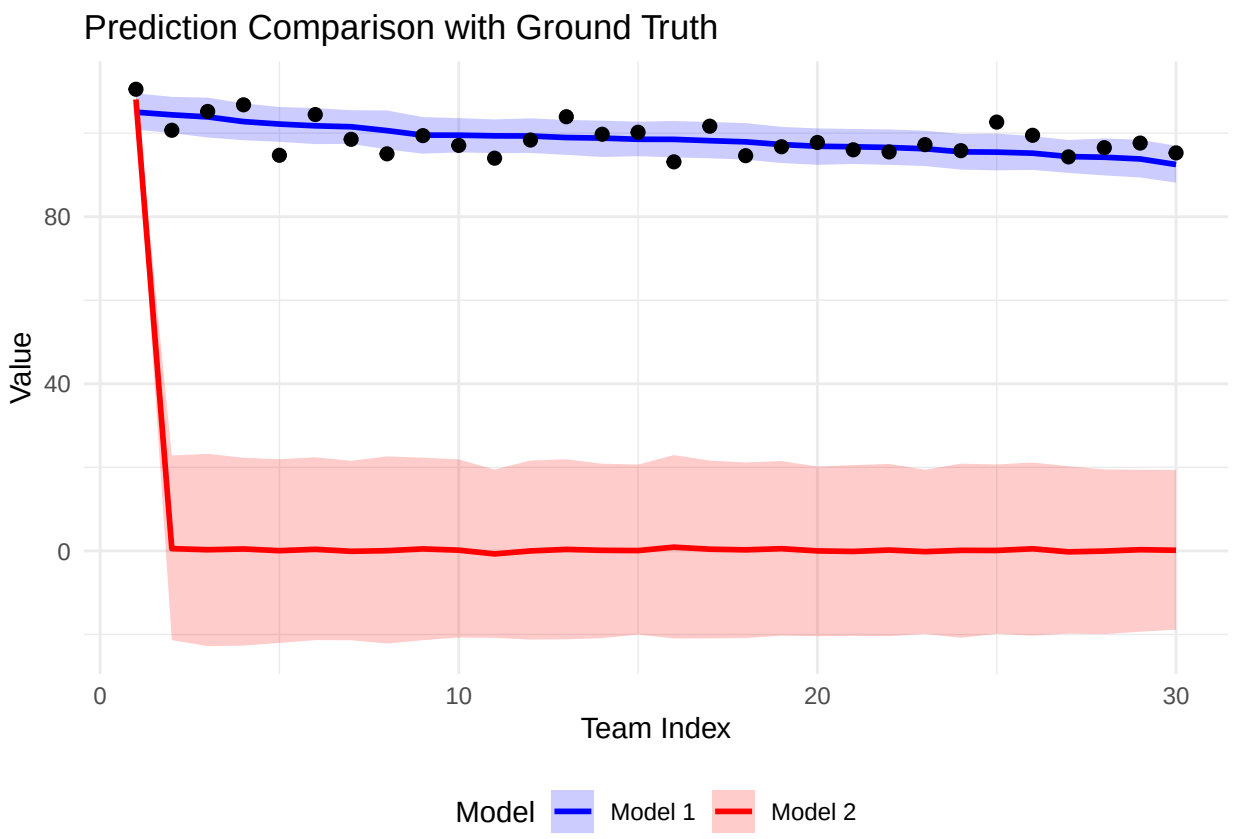


Finally, we can use the two models to make predictions on the NBA games from the rest of the season. We can use the following code to make predictions. From the visualization, we can see that the prediction of model with unknown but identical variances is more accurate than that with unknown and independent variances. From earlier visualizations, we can see that the fitted hyper-parameter μ in the second model is near-zero. I suspect the additional degree of freedom from independent σ_j in the second model may lead to the over-fitting of history data.

```
# Extract the required variables
n_future <- df$number_remaining_games
y_bar_true <- df$average_remaining_games

# Make predictions using models
y_pred_1 <- matrix(0, nrow = 2000, ncol = length(y_bar_true))
y_pred_2 <- matrix(0, nrow = 2000, ncol = length(y_bar_true))
for (i in 1:2000) {
  for (j in 1:length(y_bar_true)) {
    y_pred_1[i, j] <- mean(
      m1_smpls$theta[i, j] + rnorm(n_future[j]) * m1_smpls$sigma[i]
    )
    y_pred_2[i, j] <- mean(
      m2_smpls$theta[i, j] + rnorm(n_future[j]) * m2_smpls$sigma[i, j]
    )
  }
}
```

```r
# Plot the predictions
summary_1 <- t(
  apply(y_pred_1, 2, function(x) {
    c(mean = mean(x), lower = quantile(x, 0.05), upper = quantile(x, 0.95))
  })
)
summary_2 <- t(
  apply(y_pred_2, 2, function(x) {
    c(mean = mean(x), lower = quantile(x, 0.05), upper = quantile(x, 0.95))
  })
)
summary <- data.frame(
  x = 1:length(y_bar_true),
  true = y_bar_true,
  mean1 = summary_1[, "mean"],
  lower1 = summary_1[, "lower.5%"],
  upper1 = summary_1[, "upper.95%"],
  mean2 = summary_2[, "mean"],
  lower2 = summary_2[, "lower.5%"],
  upper2 = summary_2[, "upper.95%"]
)
ggplot(summary, aes(x = x)) +
  geom_ribbon(aes(ymin = lower1, ymax = upper1, fill = "Model 1"), alpha = 0.2) +
  geom_ribbon(aes(ymin = lower2, ymax = upper2, fill = "Model 2"), alpha = 0.2) +
  geom_line(aes(y = mean1, color = "Model 1"), size = 1) +
  geom_line(aes(y = mean2, color = "Model 2"), size = 1) +
  geom_point(aes(y = true), color = "black", size = 2) +
  scale_color_manual(values = c("blue", "red"),
                     name = "Model") +
  scale_fill_manual(values = c("blue", "red"),
                    name = "Model") +
  labs(title = "Prediction Comparison with Ground Truth",
       x = "Team Index",
       y = "Value") +
  theme_minimal() +
  theme(legend.position = "bottom")
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```


Prediction Comparison with Ground Truth
Value
80
40
0
0
10
20
30
Team Index
Model
Model 1
Model 2

The Stroop Effect

Consider a psychological task where subjects are presented with a word at the center of a computer screen (`'red'`, `'blue'`, or `'green'`). Further, the word is colored either red, blue or green. In some trials, the word matches the color of the text ('congruent' condition); otherwise they do not match ('incongruent' condition, *e.g., the word is `'red'` but it is colored blue*). Subjects are told to focus only on the color that the word is written in, and press 1 if the color is red, 2 if it is blue, and 3 if it is green. In each case, the experimenter measures the reaction time (*i.e., how long it takes them to press the correct button*). The Stroop effect is a robust effect in psychology where the reaction time in the incongruent condition is on average **larger** than in the congruent condition.

Your task is to use the data in `stroop_data.csv` to verify if this is the case. The data measures multiple reactions times of different subjects in congruent and incongruent settings. You will model this with a hierarchical Bayeisan model, with the goal of determining:

- how much longer reaction times are for each color in incongruent vs congruent cases, and whether this difference is significant.
- how different the effect is for each color, and whether these differences are significant.
- how different individuals in the study are from each other.

Your model should account for the fact that

- each response of each individual involves random fluctuations
- reaction times and the manitude of the Stroop effect can be different for different individuals
- reaction times and the magnitude of the Stroop effect can be different for different colors (*e.g., it might be smaller for red where you have to press* 1 vs others)

Your hierarchical model should allow statistical sharing among individuals and possibly among different colors. Justify your model and prior choices, implement it in `Stan` and sicusss your findings, being sure to include visualizations and predictive checks of model fit. You must present your final results in a form that can be readily understood by a general audience.

Solution:

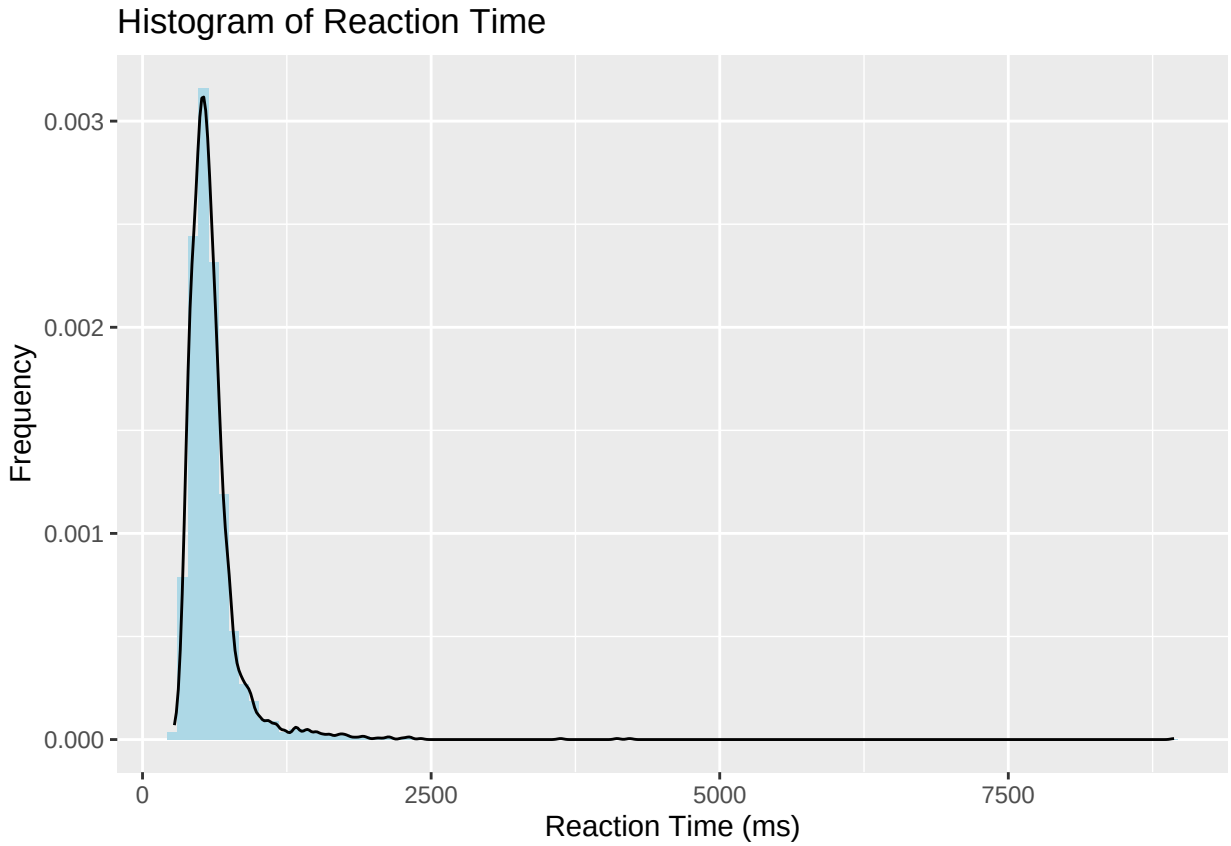
```r
# Read the data
if (file.exists("data/stroop_dataset.csv")) {
  data <- read.csv("data/stroop_dataset.csv", header = TRUE, row.names = 1)
} else {
  stop("FileNotFound: data file not found at 'data/stroop_dataset.csv'.")
}
head(data)
```

```
##   subj trial   condition  word color   RT
## 1    1     0   Congruent   red   red 1484
## 2    1     1 Incongruent green  blue 1316
## 3    1     2 Incongruent  blue green  628
## 4    1     3   Congruent green green  511
## 5    1     4   Congruent  blue  blue  509
## 6    1     5 Incongruent   red  blue  903
```

First we take a look at the distribution of reaction time over all subjects and trials. The following histogram shows that reaction time has a support of all positive real numbers, with a right-skewed long-tailed distribution. Therefore, we can model reaction time with a log-normal distribution.

```r
ggplot(data, aes(x = RT)) +
  geom_histogram(aes(y = after_stat(density)), bins = 100, fill = "lightblue") +
  labs(
    title = "Histogram of Reaction Time",
    x = "Reaction Time (ms)",
```

```
    y = "Frequency"
  ) +
  geom_density()
```



According to the problem description, we have three colors and three words. Without loss of generality, we can represent them as discrete random variables. Then we obtain a binary indicator to represent the congruence of the word and the color. In this formulation, we introduce means for congruent and incongruent conditions, where both of them follow a log-normal distribution. In the distribution of the reaction time, we activate either one of them based on the binary congruence indicator. Finally, we pose standard Gamma priors on the standard deviations in the log-normal distribution of reaction time.

```
hierarchical_model_code <- "
  data {
      int<lower=0> n;                       // number of responses
      int<lower=0> k;                       // number of subjects
      int<lower=0> c;                       // number of colors
      real<lower=0> pr_std;                 // Prior standard deviations
      array[n] int<lower=1, upper=k> subjs; // Subject ID for each response
      array[n] int<lower=1, upper=c> color; // Color of the word
      array[n] int<lower=1, upper=c> word;  // Word in each trial
      array[n] real y;                      // Reaction time of responses
  }

  parameters {
      vector<lower=0>[k] std;       // Emission standard deviation
      matrix[k, c] mu_congruent;    // Mean of reaction time for congruent
      matrix[k, c] mu_incongruent;  // Mean of reaction time for incongruent
```

```
  }

  transformed parameters {
      // Calculate the binary indicator for congruence
      vector[n] congruent;
      for (i in 1:n) {
          congruent[i] = (color[i] == word[i]);
      }

      // Calculate the emission mean as a Bernoulli mixture
      vector[n] mu;
      for (i in 1:n) {
          mu[i] = (
              mu_congruent[subjs[i], color[i]] * congruent[i]
              + mu_incongruent[subjs[i], color[i]] * (1 - congruent[i])
          );
      }
  }

  model {
      // Sample subject parameters from priors
      for (i in 1:k) {
          for (j in 1:c) {
              mu_congruent[i, j] ~ normal(0, pr_std);
              mu_incongruent[i, j] ~ normal(1, pr_std);
          }
      }
      std ~ gamma(1, 1);
      for (i in 1:n) {
          y[i] ~ lognormal(mu[i], std[subjs[i]]);
      }
  }

  generated quantities {
      array[n] real y_hat;
      for (i in 1:n) {
          y_hat[i] = lognormal_rng(mu[i], std[subjs[i]]);
      }
  }
"
hierarchical_model <- stan_model(
  model_name = "stroop_hierarchical",
  model_code = hierarchical_model_code
)
```

We sample $5,000$ samples from the hierarchical model using the data as follows. In the samples, we obtain two matrix representing the logarithm of mean of reaction time for each subject for each color in congruent and incongruent conditions, respectively. The two matrix is used for answering the three questions in the problem description.

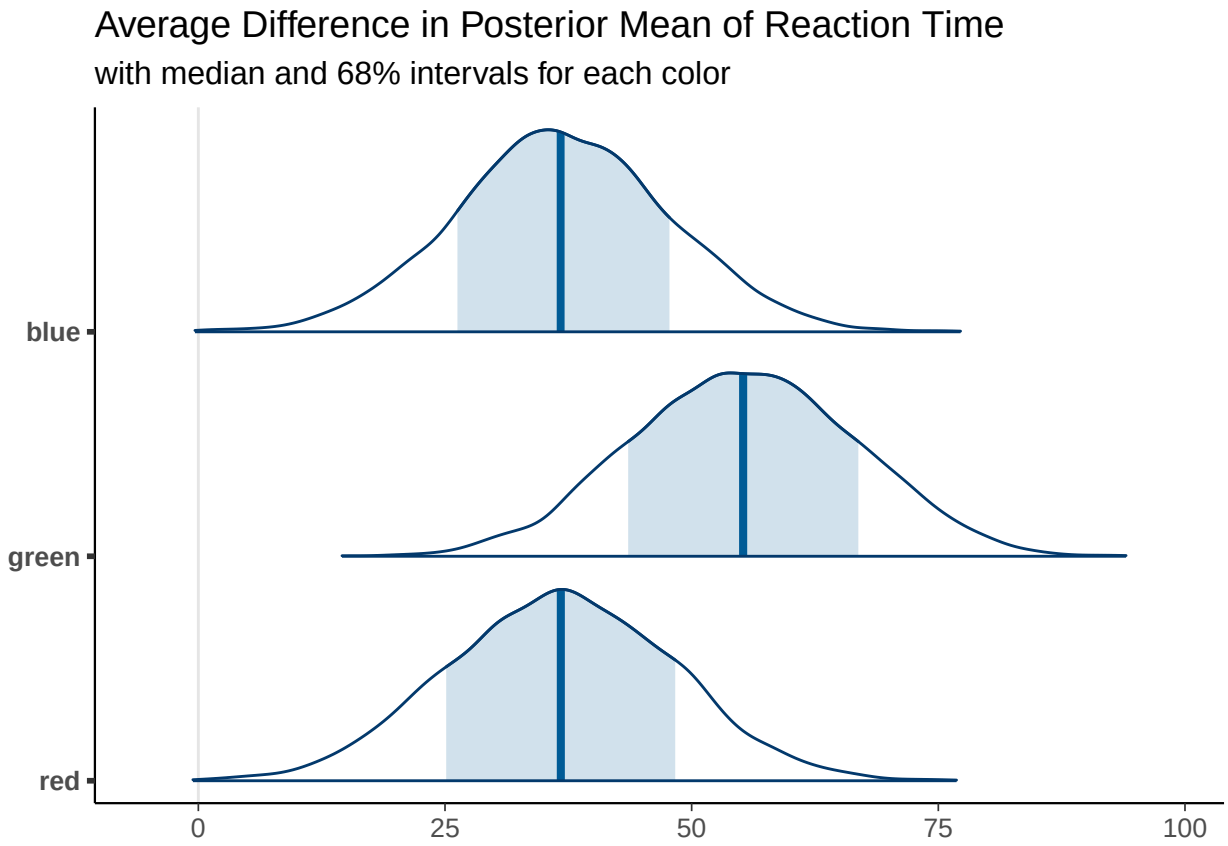
```
n <- dim(data)[1]
k <- length(unique(data$subj))
c <- length(unique(data$color))
subjs <- data$subj
```

```
color <- as.numeric(factor(data$color))
word <- as.numeric(factor(data$word))
y <- data$RT
hm_data <- list(
  n = n,
  k = k,
  c = c,
  pr_std = 10.0,
  subjs = subjs,
  color = color,
  word = word,
  y = y
)
nfit <- sampling(
  hierarchical_model,
  data = hm_data,
  chains = 1,
  iter = 10000,
  warmup = 5000,
  show_message = FALSE,
  cores = 16,
  seed = 42,
)
samples <- as.data.frame(nfit)
```

The following visualization shows the distribution of difference in the posterior mean of reaction time between incongruent and congruent conditions for each color averaged by subjects. The plot shows that the Stroop effect is significant for all colors. Among the three colors, the Stroop effect is most significant if the words are green, followed by red and blue. Difference in effect between red and blue is not significant.

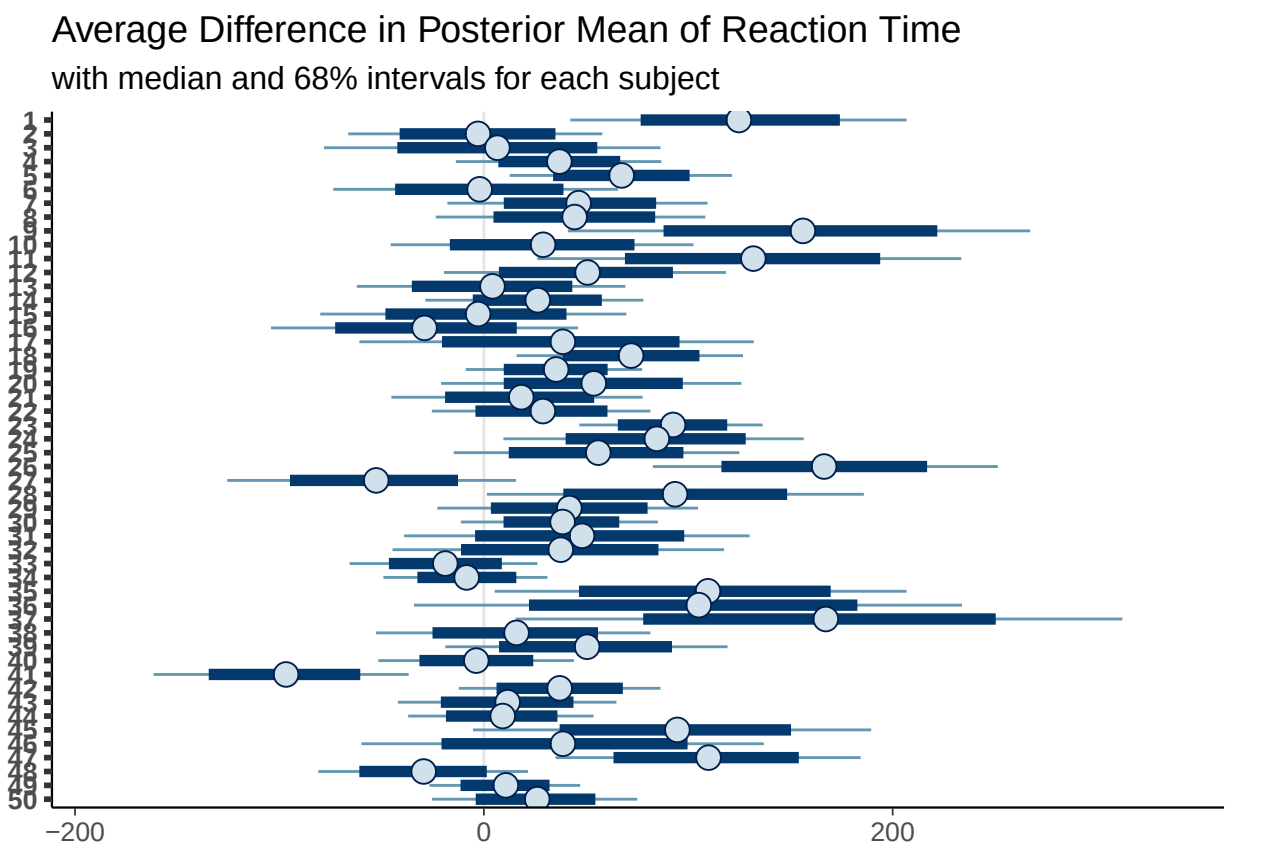
```
mu_c <- exp(
  samples[
    , grep(pattern = "^mu_congruent", x = colnames(samples))
  ]
)
mu_inc <- exp(
  samples[
    , grep(pattern = "^mu_incongruent", x = colnames(samples))
  ]
)
diff <- data.matrix(mu_inc) - data.matrix(mu_c)

color_diff <- array(0, dim = c(dim(samples)[1], c))
colnames(color_diff) <- levels(factor(data$color))
for (i in 1:c) {
  color_diff[, i] <- apply(
    diff[, (k * (i - 1) + 1):(k * i)],
    c(1),
    mean
  )
}
mcmc_areas(
  x = color_diff,
  par = levels(factor(data$color)),
```

```
  prob = 0.68,
) + labs(
  title = "Average Difference in Posterior Mean of Reaction Time",
  subtitle = "with median and 68% intervals for each color"
)
```



The following visualization shows the distribution of difference in the posterior mean of reaction time between incongruent and congruent conditions for each subject averaged by colors. Based on the result, the mean difference varies significantly among subjects, where we observed the Stroop effect in most of their posterior distributions, with a few exceptions (*e.g., subject* 15, subject 27, subject 41, etc.).

```
subjs_diff <- array(0, dim = c(dim(samples)[1], k))
colnames(subjs_diff) <- paste("", unique(data$subj))
for (i in 1:k) {
  subjs_diff[, i] <- apply(
    diff[, seq(1, k * c, by = k) + (i - 1)],
    c(1),
    mean
  )
}
mcmc_intervals(
  x = subjs_diff,
  par = colnames(subjs_diff),
  prob = 0.68,
) + labs(
  title = "Average Difference in Posterior Mean of Reaction Time",
  subtitle = "with median and 68% intervals for each subject"
)
```


Average Difference in Posterior Mean of Reaction Time
with median and 68% intervals for each subject
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
−200
0
200