STAT 656 HW 3

Satoshi Ido (ID: 34788706)

October 29, 2023

Gibbs sampling for normal hierarchical models

Q2

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

Write down the form of the conditional distributions above. Here, the material in Lecture 3, 7 and 13 will be useful.

- **Sampling**

$$\begin{aligned} p(\mu|\tau^2, \theta_1, \ldots, \theta_J, \sigma_1^2, \ldots, \sigma_J^2, Y) &\propto p(\theta_1, \ldots, \theta_J|\mu, \tau^2) p(\mu|\tau^2, \sigma_1^2, \ldots, \sigma_J^2) \\ &\propto p(\theta_1, \ldots, \theta_J|\mu, \tau^2) \\ &\sim \mathrm{N}(\hat{\mu}, \tau^2) \\ \text{where } \hat{\mu} &= \textstyle\sum_{j=1}^{J} \theta_j \end{aligned}$$

$$\begin{aligned} p(\tau^2|\mu, \theta_1, \ldots, \theta_J, \sigma_1^2, \ldots, \sigma_J^2, Y) &\propto p(\theta_1, \ldots, \theta_J|\mu, \tau^2) p(\tau^2|\mu, \sigma_1^2, \ldots, \sigma_J^2) \\ &\propto p(\theta_1, \ldots, \theta_J|\mu, \tau^2) \\ &\propto \left(\frac{1}{\tau}\right)^J \exp\left(\frac{-1}{2\tau^2} \sum_{j=1}^{J} (\theta_j - \mu)^2\right) \frac{1}{\tau} \\ &= \left(\frac{1}{\tau}\right)^{J+1} \exp\left(\frac{-1}{2\tau^2} \sum_{j=1}^{J} (\theta_j - \mu)^2\right) \\ &\sim \mathrm{InvGam}\left(\frac{J-1}{2}, \frac{\sum_{j=1}^{J} (\theta_j - \mu)^2}{2}\right) \end{aligned}$$

- **model 1**

  Each θ_j can be sampled from:

$$
\begin{aligned}
p(\theta_j|\sigma^2,\mu,\tau^2,Y) &\propto p(\theta_j|\sigma^2,\mu,\tau^2,\bar{y_j}) \\
&\propto p(\bar{y_j}|\theta_j,\mu,\tau^2)p(\theta_j|\mu,\tau^2) \\
&\sim \mathrm{N}\left(\frac{\frac{\mu}{\tau^2}+\frac{\sum_{i=1}^{n_j} y_{ij}}{\sigma^2}}{\frac{1}{\tau^2}+\frac{n_j}{\sigma^2}}, \frac{1}{\frac{1}{\tau^2}+\frac{n_j}{\sigma^2}}\right)
\end{aligned}
$$

σ^2 can be sampled from:

$$
\begin{aligned}
p(\sigma^2|\theta_1,\ldots,\theta_J,\mu,\tau^2,Y) &\propto p(\sigma^2|Y,\theta_1,\ldots,\theta_J) \\
&\propto p(Y|\theta_1,\ldots,\theta_J,\sigma^2)p(\sigma^2) \\
&\propto \prod_{j=1}^{J}\prod_{i=1}^{n_j}\left(\frac{1}{\sigma}\right)\exp\left(\frac{-(y_{ij}-\theta_j)^2}{2\sigma^2}\right)\frac{1}{\sigma} \\
&\propto \left(\frac{1}{\sigma}\right)^{n+1}\exp\left(\frac{-1}{2\sigma^2}\sum\sum(y_{ij}-\theta_j)^2\right) \\
&\sim \text{inv-gam}\left(\frac{n-1}{2}, \sum\sum(y_{ij}-\theta_j)^2\right)
\end{aligned}
$$

- **model 2**

  For each team j:

$$
\begin{aligned}
&p(\theta_j|\sigma_j^2,\mu,\tau^2,Y) \\
&\qquad\propto p(\theta_j|\sigma^2,\mu,\tau^2,\bar{y_j}) \\
&\qquad\propto p(\bar{y_j}|\theta_j,\mu,\tau^2,\sigma_j^2)p(\theta_j|\mu,\tau^2,\sigma_j^2) \\
&\qquad\sim \mathrm{N}\left(\frac{\frac{\mu}{\tau^2}+\frac{\sum_{i=1}^{n_j} y_{ij}}{\sigma_j^2}}{\frac{1}{\tau^2}+\frac{n_j}{\sigma_j^2}}, \frac{1}{\frac{1}{\tau^2}+\frac{n_j}{\sigma_j^2}}\right)
\end{aligned}
$$

σ_j^2 can be sampled from:

$$
\begin{aligned}
p(\sigma_j^2|\theta_j,\mu,\tau^2,Y) &\propto p(\sigma_j^2|Y,\theta_j) \\
&\propto p(Y|\theta_j,\sigma_j^2)p(\sigma_j^2) \\
&\propto \prod_{i=1}^{n_j}\left(\frac{1}{\sigma_j}\right)\exp\left(\frac{-(y_{ij}-\theta_j)^2}{2\sigma_j^2}\right)\frac{1}{\sigma_j} \\
&\propto \left(\frac{1}{\sigma_j}\right)^{n_j+1}\exp\left(\frac{-1}{2\sigma_j^2}\sum(y_{ij}-\theta_j)^2\right) \\
&\sim \text{inv-gam}\left(\frac{n_j-1}{2}, \sum(y_{ij}-\theta_j)^2\right)
\end{aligned}
$$

Q3

Write down R code to implement the Gibbs samplers

Model 1

```r
# we first prepare the data in a general format
n_j <- df[, 1]
y_bar <- df[, 2]
sample_sd <- df[, 3]
sigma_sq <- sample_sd^2/n_j
J <- length(y_bar)

# prior
n_iter <- 20000
## initial values for the parameters
mu <- 100
tau_sq <- 1

# for saving the results
theta_vec1 <- c()
sigma_sq_vec1 <- c()
mu_vec1 <- c()
tau_sq_vec1 <- c()

for (i in 1:n_iter) {

  # theta
  theta_var1 <- 1/(1/tau_sq + n_j/sigma_sq)
  theta_mean1 <- (mu/tau_sq + n_j * y_bar/sigma_sq)
  theta_mean1 <- theta_mean1 * theta_var1
  theta <- rnorm(length(theta_mean1), mean = theta_mean1, sd = sqrt(theta_var1))
  theta_vec1 <- rbind(theta_vec1, theta)

  # sigma_sq
  shape_sigma <- (sum(n_j) - 1)/2
  rate_sigma <- 1/2 * (sum((n_j - 1) * sample_sd^2) + sum(n_j * ((y_bar - theta)^2)))

  sigma_sq <- rinvgamma(1, shape = shape_sigma, rate = rate_sigma)
  sigma_sq_vec1 <- c(sigma_sq_vec1, sigma_sq)

  # mu
  mu_mean <- mean(theta)
  mu <- rnorm(1, mean = mu_mean, sd = sqrt(tau_sq))
  mu_vec1 <- c(mu_vec1, mu)

  # tau_sq given theta and mu
  shape_tau <- (J - 1)/2
  rate_tau <- sum((mu - theta)^2)/2
  tau_sq <- rinvgamma(1, shape = shape_tau, rate = rate_tau)
  tau_sq_vec1 <- c(tau_sq_vec1, tau_sq)
}
```

Model 2

```
# we first prepare the data in a general format
n_j <- df[, 1]
y_bar <- df[, 2]
sample_sd <- df[, 3]
sigma_sq <- sample_sd^2/n_j
J <- length(y_bar)

# prior
n_iter <- 20000
## initial values for the parameters
mu <- 100
tau_sq <- 1

# for saving the results
theta_vec2 <- c()
sigma_sq_vec2 <- c()
mu_vec2 <- c()
tau_sq_vec2 <- c()

for (i in 1:n_iter) {
  # theta
  theta_var2 <- 1/(1/tau_sq + n_j/sigma_sq)
  theta_mean2 <- (mu/tau_sq + n_j * y_bar/sigma_sq)
  theta_mean2 <- theta_mean2 * theta_var2
  theta <- rnorm(length(theta_mean2), mean = theta_mean2, sd = sqrt(theta_var2))
  theta_vec2 <- rbind(theta_vec2, theta)

  # sigma_sq
  shape_sigma <- (n_j - 1)/2
  rate_sigma <- 1/2 * (sample_sd^2 * (n_j - 1) + n_j * (theta - y_bar)^2)

  sigma_sq <- rinvgamma(J, shape = shape_sigma, rate = rate_sigma)
  sigma_sq_vec2 <- rbind(sigma_sq_vec2, sigma_sq)

  # mu given theta and tau_sq
  mu_mean <- mean(theta)
  mu <- rnorm(1, mean = mu_mean, sd = sqrt(tau_sq/J))
  mu_vec2 <- c(mu_vec2, mu)

  # tau_sq given theta and mu
  shape_tau <- (J - 1)/2
  rate_tau <- sum((mu - theta)^2)/2
  tau_sq <- rinvgamma(1, shape = shape_tau, rate = rate_tau)
  tau_sq_vec2 <- c(tau_sq_vec2, tau_sq)
}
```

Q4

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^2, plot the MCMC traceplots, and diagnose mixing. Also plot the corresponding posterior distributions.

```
mudata <- data.frame(mu_vec1, mu_vec2)
colnames(mudata) <- c("model1", "model2")

taudata <- data.frame(tau_sq_vec1, tau_sq_vec2)
colnames(taudata) <- c("model1", "model2")

sigma_sq_data_mod1 <- data.frame(sigma_sq_vec1)
colnames(sigma_sq_data_mod1) <- "identical_sigma_sq"
sigma_sq_data_mod2 <- data.frame(sigma_sq_vec2[, 1],
  sigma_sq_vec2[, 2], sigma_sq_vec2[, 3])
colnames(sigma_sq_data_mod2) <- c("team1", "team2",
  "team3")

theta_data_mod1 <- data.frame(theta_vec1[, 1], theta_vec1[,
  2], theta_vec1[, 3])
colnames(theta_data_mod1) <- c("team1", "team2", "team3")
theta_data_mod2 <- data.frame(theta_vec2[, 1], theta_vec2[,
  2], theta_vec2[, 3])
colnames(theta_data_mod2) <- c("team1", "team2", "team3")

theta_data <- data.frame(theta_vec1[, 1], theta_vec1[,
  2], theta_vec1[, 3], theta_vec2[, 1], theta_vec2[,
  2], theta_vec2[, 3])
colnames(theta_data) <- c("team1_mod1", "team2_mod1",
  "team3_mod1", "team1_mod2", "team2_mod2", "team3_mod2")

muplot <- ggplot(mudata, aes(x = 1:20000)) + geom_line(aes(y = model1),
  color = "#293352") + geom_line(aes(y = model2),
  color = "#E7B800") + ggtitle("MCMC trace plot for mu") +
  xlab("iteration") + ylab("mu") + theme(plot.title = element_text(hjust = 0.5))

tau_sq_plot <- ggplot(taudata, aes(x = 1:20000)) +
  geom_line(aes(y = model1), color = "#293352") +
  geom_line(aes(y = model2), color = "#E7B800") +
  ggtitle("MCMC trace plot for tau_sq") + xlab("iteration") +
  ylab("tau_sq") + theme(plot.title = element_text(hjust = 0.5))

sigma_sq_iden_plot <- ggplot(sigma_sq_data_mod1, aes(x = 1:20000)) +
  geom_line(aes(y = identical_sigma_sq), color = "#293352") +
  ggtitle("MCMC trace plot for the identical sigma squared") +
  xlab("iteration") + ylab("sigma_sq") + theme(plot.title = element_text(hjust = 0.5))

sigma_sq_noniden <- ggplot(sigma_sq_data_mod2, aes(x = 1:20000)) +
  geom_line(aes(y = team1, color = "Team 1"), color = "#293352",
    show.legend = TRUE) + geom_line(aes(y = team2,
  color = "Team 2"), color = "#E7B800", show.legend = TRUE) +
  geom_line(aes(y = team3, color = "Team 3"), color = "#FC4E07",
    show.legend = TRUE) + ggtitle("MCMC trace plot for the non-identical sigma squared") +
  xlab("iteration") + ylab("sigma_sq") + labs(color = "Team",
  title = "MCMC trace plot for the non-identical sigma squared",
  x = "Iteration", y = "Sigma Squared") + theme(plot.title = element_text(hjust = 0.5)) +
  scale_color_identity(guide = "legend", breaks = c("Team 1",
    "Team 2", "Team 3"), labels = c("Team 1", "Team 2",
```

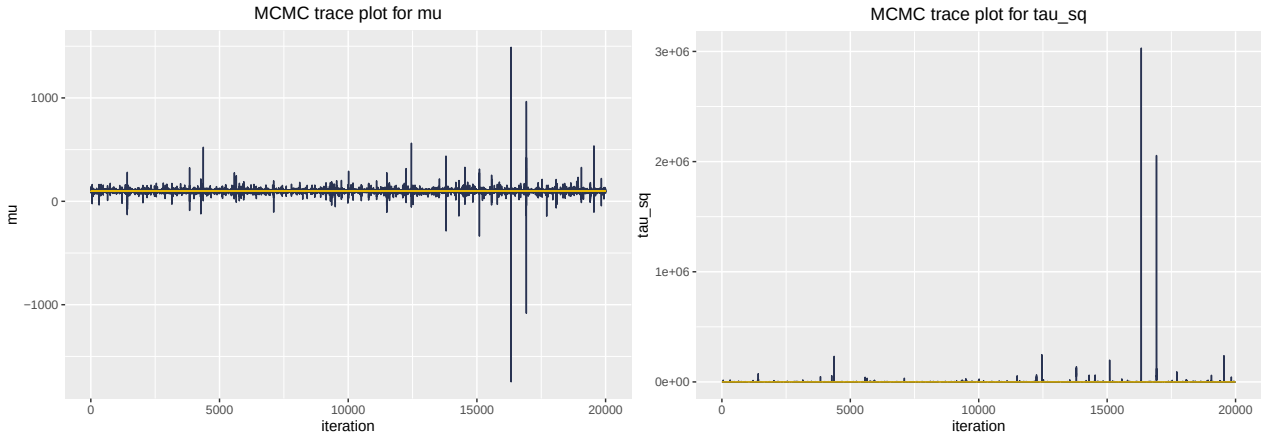
```
    "Team 3"))

theta_team1_plot <- ggplot(theta_data, aes(x = 1:20000)) +
  geom_line(aes(y = team1_mod1), color = "#293352") +
  geom_line(aes(y = team1_mod2), color = "#E7B800") +
  ggtitle("MCMC trace plot for theta of team 1") +
  xlab("iteration") + ylab("theta") + theme(plot.title = element_text(hjust = 0.5))

theta_team2_plot <- ggplot(theta_data, aes(x = 1:20000)) +
  geom_line(aes(y = team2_mod1), color = "#293352") +
  geom_line(aes(y = team2_mod2), color = "#E7B800") +
  ggtitle("MCMC trace plot for theta of team 2") +
  xlab("iteration") + ylab("theta") + theme(plot.title = element_text(hjust = 0.5))

theta_team3_plot <- ggplot(theta_data, aes(x = 1:20000)) +
  geom_line(aes(y = team3_mod1), color = "#293352") +
  geom_line(aes(y = team3_mod2), color = "#E7B800") +
  ggtitle("MCMC trace plot for theta of team 3") +
  xlab("iteration") + ylab("theta") + theme(plot.title = element_text(hjust = 0.5))
```

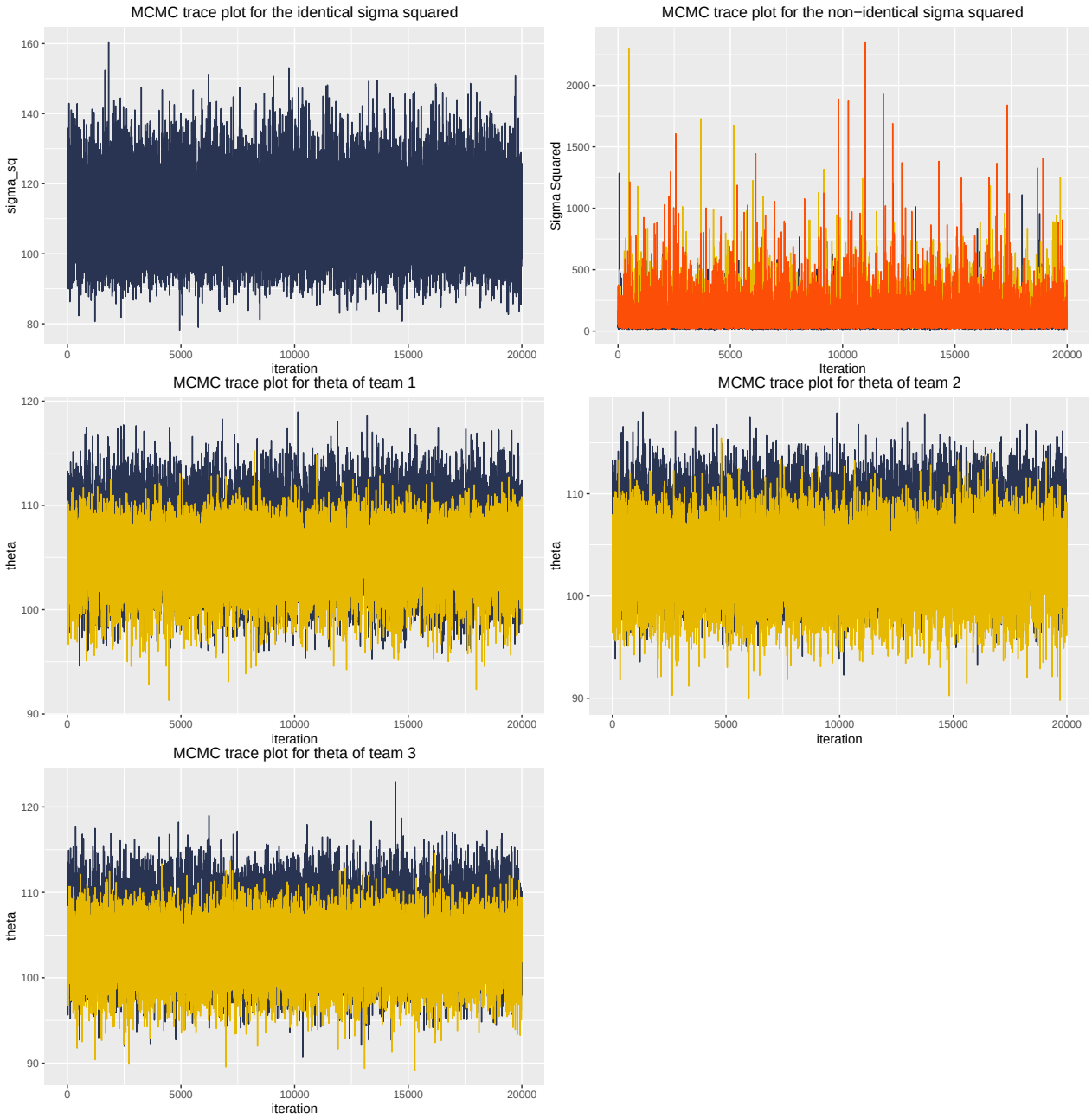
The trace plot below reveals that there are noticeable fluctuations in the sampling sequences for μ and τ^2. However, the overall trend remains consistent, with the exception of a few outliers.

Subsequently, we exhibit the trace plot for θ as a random illustration. In the case of θ_1 and θ_2, the first model demonstrates reduced variability in comparison to the second model. Yet, when observing θ_3, the fluctuations in model 1 are more pronounced.

In addition, the outcomes for a consistent σ^2 appear to be steady. Nonetheless, when observing varying σ^2 values, σ^2_{12} exhibits more instability than both σ^2_{22} and σ^2_{32}. This variability may influence the central tendency of the post-sampling value $\bar{y}$.

```
muplot; tau_sq_plot
sigma_sq_iden_plot; sigma_sq_noniden
theta_team1_plot; theta_team2_plot; theta_team3_plot
```





Q5

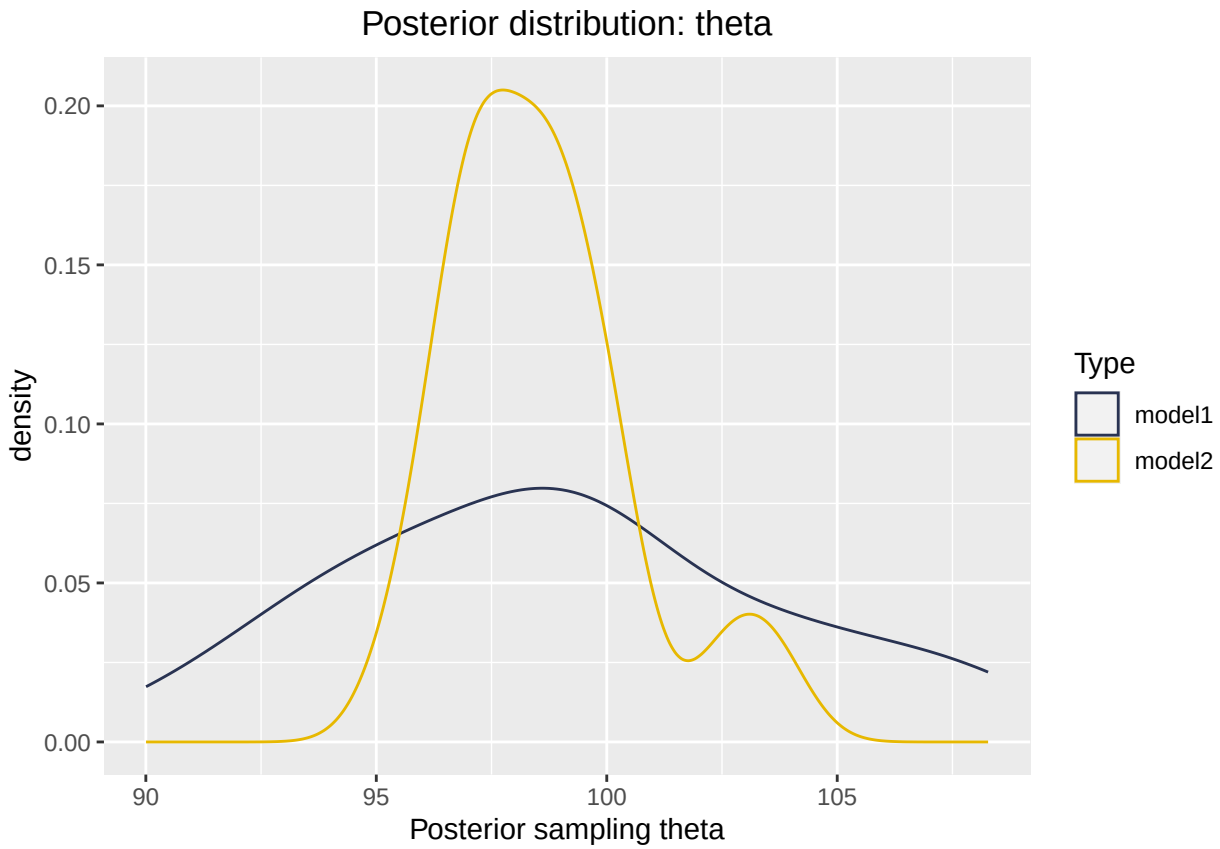
Comment on how the posterior distributions differ from each other. Use each model to make predictions on the NBA games from the rest of the season (following code from Lecture 7), and comment on which model (or the model with known variances from Lecture 7) performs best.

From the result of the posterior distribution, model2 seems to work better.

The model1 allows more randomness among the prediction, and thus it has wider variation of data. the model2 is more conservative and gives us the values closer to $\bar{\theta}$.

```
# posterior
number_draws <- 50000
```

```r
posterior_draws_mod1 <- matrix(NA, nrow = number_draws,
    ncol = length(y_bar))
posterior_draws_mod2 <- matrix(NA, nrow = number_draws,
    ncol = length(y_bar))

for (i in 1:number_draws) {
    posterior_draws_mod1[i, ] <- rnorm(J, theta_mean1,
        sqrt(theta_var1))
    posterior_draws_mod2[i, ] <- rnorm(J, theta_mean2,
        sqrt(theta_var2))
}

# burn-in
burn_posterior_draws_mod1 <- posterior_draws_mod1[5001:50000,
    ]
burn_posterior_draws_mod2 <- posterior_draws_mod2[5001:50000,
    ]

posterior_means_theta_mod1 <- colMeans(burn_posterior_draws_mod1)
posterior_means_theta_mod2 <- colMeans(burn_posterior_draws_mod2)

posterior_theta <- c(posterior_means_theta_mod1,
    posterior_means_theta_mod2)
type <- c(rep("model1", 30), rep("model2", 30))
posterior_theta_data <- data.frame(posterior_theta,
    type)
posterior_theta_plot <- ggplot(posterior_theta_data,
    aes(x = posterior_theta, color = type)) +
    geom_density() + xlab("Posterior sampling theta") +
    ylab("density") + ggtitle("Posterior distribution: theta") +
    theme(plot.title = element_text(hjust = 0.5)) +
    scale_color_manual(values = c(model1 = "#293352",
        model2 = "#E7B800"), guide = guide_legend(title = "Type"))

posterior_theta_plot
```



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 (Stroop, 1935) 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 Bayesian 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 magnitude 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 discuss 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.

Firstly, we define a parameter as below:

```
stroop <- read.csv("/Users/satoshiido/Documents/statistical-analysis/656/hw/hw3/stroop_dataset.csv")
head(stroop)
```

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

$$\theta_i \mid \mu, \tau^2 \sim \mathrm{N}(\mu, \tau^2), \quad i = 1, 2, \ldots, n$$
$$x_{ij} \mid \mu, \tau^2, \sigma_j^2, \theta_i \sim \mathrm{N}(\theta_i, \sigma_j^2), \quad j = 1, 2, \ldots$$
$$y_{ij} \mid \theta_i, \mu, \tau^2, \sigma_j^2, x_{ij}, s^2 \sim \mathrm{N}(x_{ij}, s^2)$$

y_{ij} here is the stroop effect of individual i for color j. \

x_{ij} here is the mean of stroop effect of individual i for color j, and θ_i is the difference of the mean reaction times for individual i under incongruent and congruent settings. Therefore, we are interested in parameters x_{ij} and θ_i^2.

For our response, we will use the mean of the difference of observations both in incongruent and congruent settings. We then fit the model with Stan to predict y_{ij}.

```
# preprocess the data
unique_subj <- unique(stroop$subj)
diff_color_effect <- as.data.frame(matrix(NA, nrow = length(unique_subj),
  ncol = 3))
color_list <- c("red", "green", "blue")
names(diff_color_effect) <- color_list

for (i in 1:length(unique_subj)) {
  for (c in (1:length(color_list))) {
    diff_color_effect[i, color_list[c]] <- mean(stroop[stroop$subj ==
      unique_subj[i] & stroop$color == color_list[c] &
      stroop$condition == "Incongruent", "RT"]) -
      mean(stroop[stroop$subj == unique_subj[i] &
        stroop$color == color_list[c] & stroop$condition ==
        "Congruent", "RT"])
  }
}

head(diff_color_effect)
```

```
##          red      green      blue
## 1   53.57143 120.571429 180.71429
## 2  -64.83333   9.071429  26.85714
## 3 -145.85714  31.428571  65.64286
## 4  309.15152 137.000000  66.02198
## 5  -96.70330 149.785714  74.21429
## 6   43.64286 100.028571  54.01099
```

```
hierarchical_cauchy_model <- "
data {
  int<lower=1> N; //the number of observations
  vector[N] diff_red; //the response variable for red
  vector[N] diff_green; //the response variable for green
```

```
  vector[N] diff_blue; //the response variable for blue
}
parameters {
  real mu;
  vector[N] x;
  vector<lower=0>[3] sigma;
  vector[N] mu_red;
  vector[N] mu_green;
  vector[N] mu_blue;
  real<lower=0> tau;
  real<lower=0> X;
}
model {
  //priors
  mu ~ normal(20,100); //weakly informative priors on the regression coefficients
  tau ~ cauchy(10,5); //weakly informative priors, see section 6.9 in STAN user guide
  x ~ normal(mu, tau); // first layer
  sigma ~ gamma(5,0.1); //weakly informative priors, see section 6.9 in STAN user guide
  X ~ normal(50, 100);
  for(n in 1:N){
    mu_red[n] ~ normal(x[n], sigma[1]);
    mu_green[n] ~ normal(x[n], sigma[2]);
    mu_blue[n] ~ normal(x[n], sigma[3]);
    diff_red[n] ~ normal(mu_red[n], X);
    diff_green[n] ~ normal(mu_green[n],X);
    diff_blue[n] ~ normal(mu_blue[n],X);
} }
"
hier_model <- stan_model(model_code = hierarchical_cauchy_model)
```

```
hier_data <- list(
  N = nrow(diff_color_effect),
  diff_red = diff_color_effect$red,
  diff_green = diff_color_effect$green,
  diff_blue = diff_color_effect$blue
)
nfit <- sampling(
  hier_model,
  data = hier_data,
  iter = 10000, warmup = 2000, chains = 1
)
```

```
##
## SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 4.3e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.43 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 10000 [  0%]  (Warmup)
## Chain 1: Iteration: 1000 / 10000 [ 10%]  (Warmup)
## Chain 1: Iteration: 2000 / 10000 [ 20%]  (Warmup)
## Chain 1: Iteration: 2001 / 10000 [ 20%]  (Sampling)
```

```
## Chain 1: Iteration: 3000 / 10000 [ 30%]  (Sampling)
## Chain 1: Iteration: 4000 / 10000 [ 40%]  (Sampling)
## Chain 1: Iteration: 5000 / 10000 [ 50%]  (Sampling)
## Chain 1: Iteration: 6000 / 10000 [ 60%]  (Sampling)
## Chain 1: Iteration: 7000 / 10000 [ 70%]  (Sampling)
## Chain 1: Iteration: 8000 / 10000 [ 80%]  (Sampling)
## Chain 1: Iteration: 9000 / 10000 [ 90%]  (Sampling)
## Chain 1: Iteration: 10000 / 10000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 1.628 seconds (Warm-up)
## Chain 1:                5.225 seconds (Sampling)
## Chain 1:                6.853 seconds (Total)
## Chain 1:
```

```
post_smp <- as.data.frame(nfit)
colmean_posterior <- colMeans(post_smp)
rowmean_post_red <- rowMeans(post_smp[c(55:104)])
rowmean_post_green <- rowMeans(post_smp[c(105:154)])
rowmean_post_blue <- rowMeans(post_smp[c(155:204)])
row_vec <- cbind(rowmean_post_red, rowmean_post_green,
  rowmean_post_blue)

post_red <- colmean_posterior[c(55:104)]
post_green <- colmean_posterior[c(105:154)]
post_blue <- colmean_posterior[c(155:204)]

mean_post_red <- mean(post_red)
mean_post_green <- mean(post_green)
mean_post_blue <- mean(post_blue)

mean_post_color <- mean(colmean_posterior[c(2:51)])
sd_post_red <- sqrt(colmean_posterior[c(52)])
sd_post_green <- sqrt(colmean_posterior[c(53)])
sd_post_blue <- sqrt(colmean_posterior[c(54)])
```

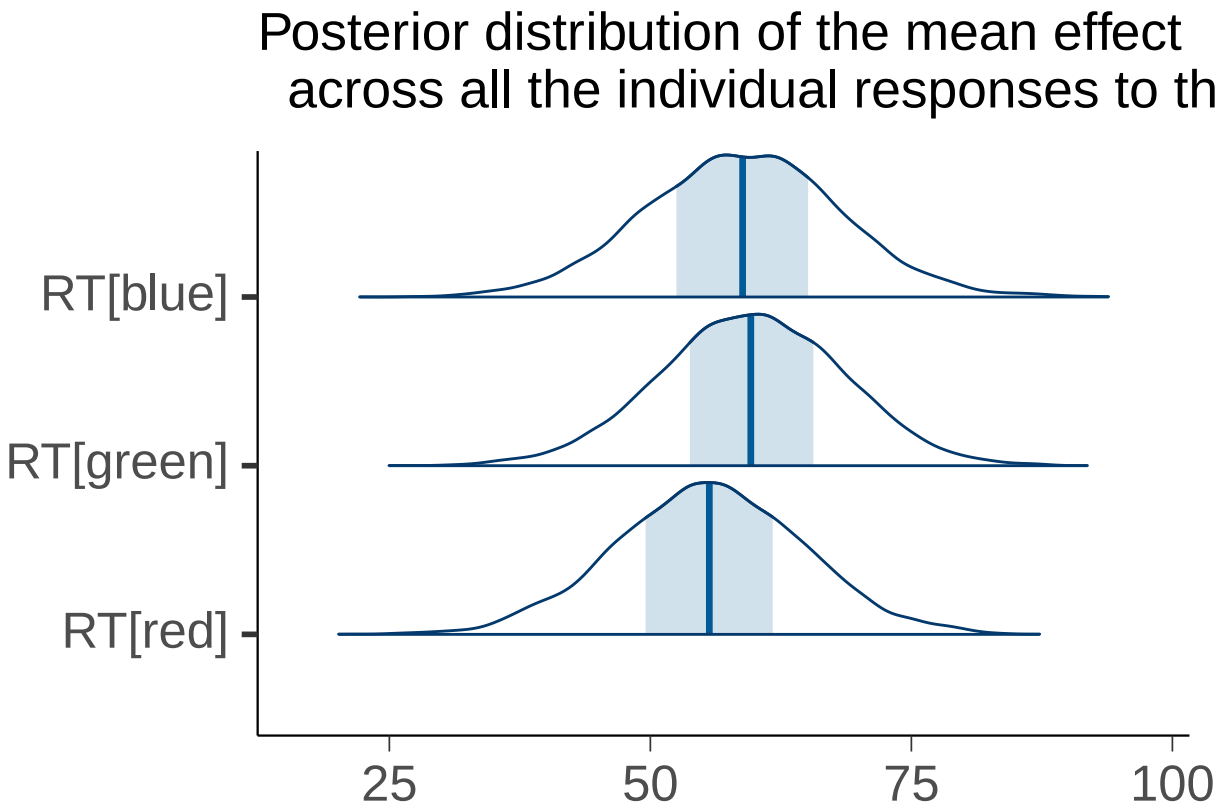
We model as "Response Time (Incongruent) - Response Time (Congruent)" as new responses. Hence, it should be positive.

According to the result above. The mean effect across all the individual responses to three color are: $RT_{red} = 55.79$, $RT_{green} = 59.65$, $RT_{blue} = 58.93$

As we can see the plots, all of the parameters (color) are the significant.

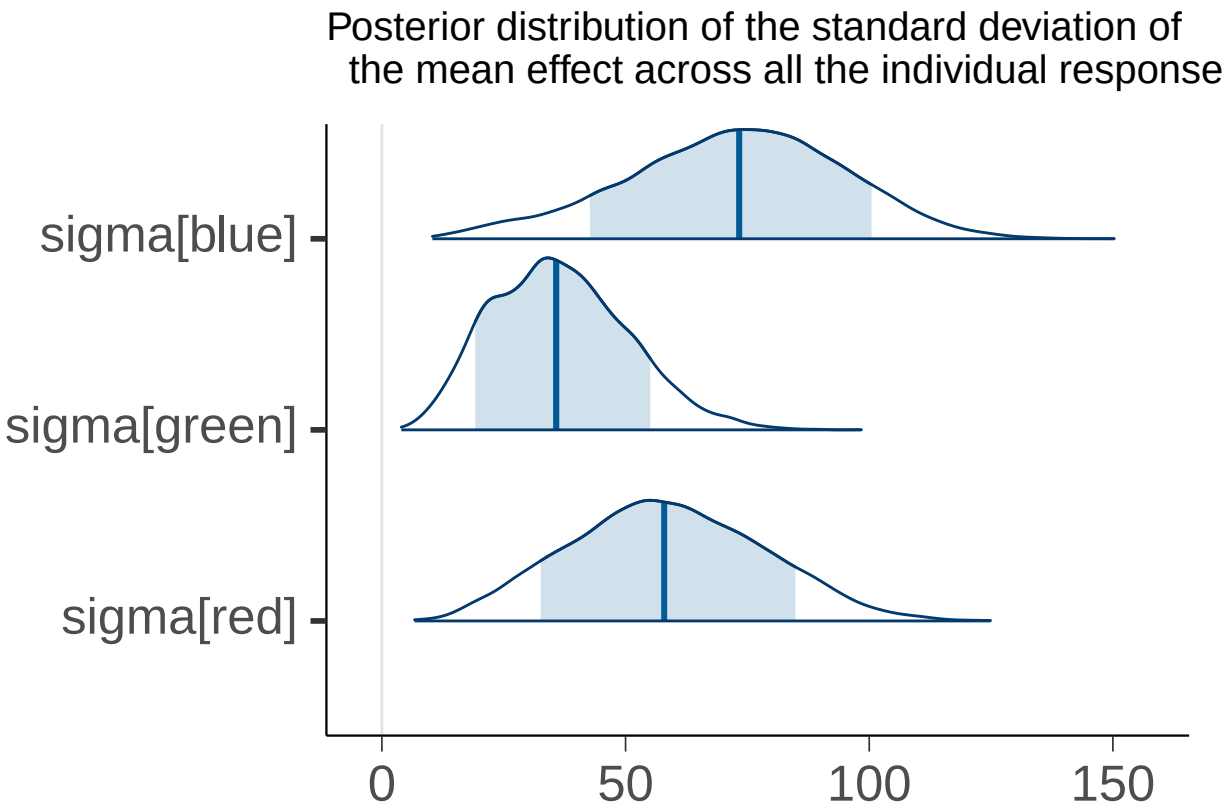
the posterior distribution of the mean effect across all the individual responses to three color are all positive. Therefore, we can say that the reaction time in the incongruent condition is on average larger than in the congruent condition.

```
mcmc_areas(row_vec) +
  scale_y_discrete(labels = c(
    "rowmean_post_red" = expression(paste("RT[red]")),
    "rowmean_post_green" = expression(paste("RT[green]")),
    "rowmean_post_blue" = expression(paste("RT[blue]"))
  )) +
  ggtitle("Posterior distribution of the mean effect
  across all the individual responses to three color") +
  theme(plot.title = element_text(size = 20))
```



We can check the posterior distribution of the standard deviation of the mean effect across all the individual responses to three color as below:

```
mcmc_areas(post_smp[c(52:54)], prob = 0.8) +
  scale_y_discrete(labels = c(
    "sigma[1]" = expression(paste("sigma[red]")),
    "sigma[2]" = expression(paste("sigma[green]")),
    "sigma[3]" = expression(paste("sigma[blue]"))
  )) +
  ggtitle("Posterior distribution of the standard deviation of
  the mean effect across all the individual responses to three color") +
  theme(plot.title = element_text(size = 15))
```

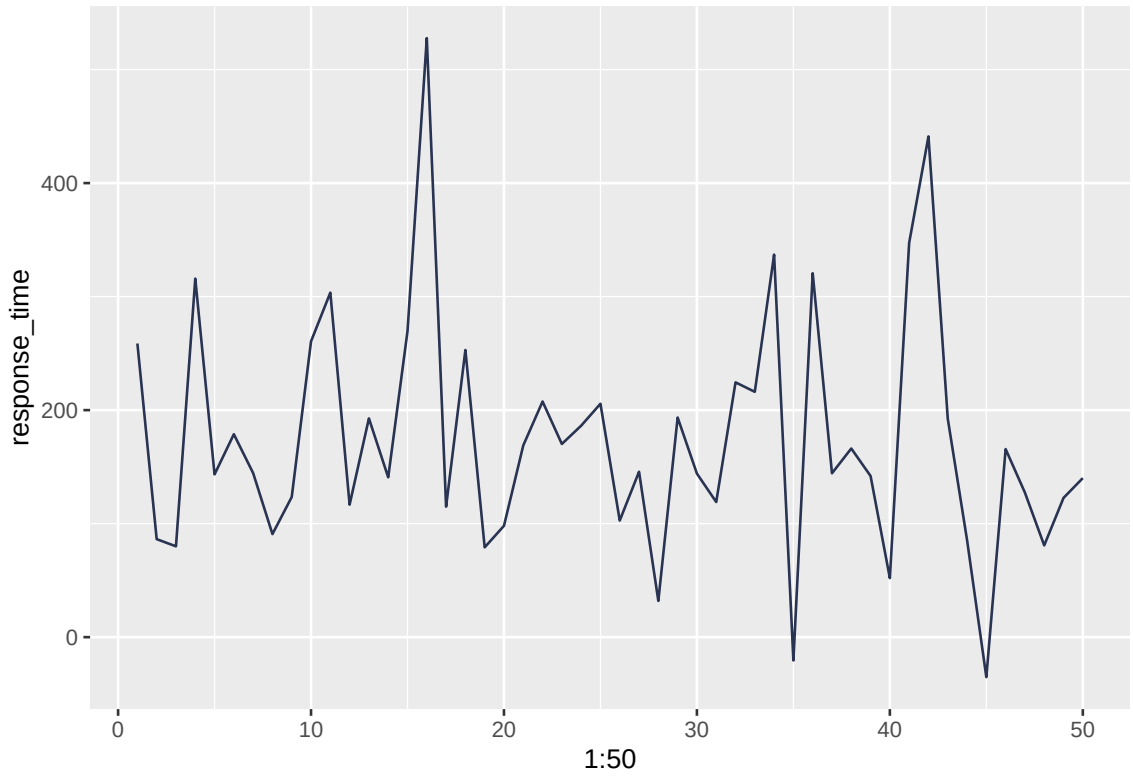


The 95% credible interval of the mean effect across all the individual responses to three color are:

$$\text{Red} : 55.79 \pm 1.96 \times 7.436$$
$$\text{Green} : 59.65 \pm 1.96 \times 5.893$$
$$\text{Blue} : 58.93 \pm 1.96 \times 8.425$$

Regarding the standard deviation of the mean effect across all the individual responses to three color, we can see there is significance among individuals.

```
individual_RT <- matrix(0, nrow = 50)

for (i in 1:50) {
  individual_RT[i] <- mean(colMeans(post_smp[c(54 + i)]) + colMeans(post_smp[c(104 +
    i)]) + colMeans(post_smp[c(154 + i)]))
}

individual_RT <- as.data.frame(individual_RT)
names(individual_RT)[1] <- "response_time"
ind_plot <- ggplot(individual_RT, aes(x = 1:50)) + geom_line(aes(y = response_time),
  color = "#293352")
ind_plot
```


response_time
400
200
0
0
10
20
30
40
50
1:50