```
############### 0 - safe choice A, 1 - risky choice B ####################
library(rstan); rstan_options(javascript=FALSE)
library(bayesplot)
library(dplyr)
options(mc.cores = parallel::detectCores())
rstan_options(auto_write = T)

dat <- read.csv('final_data.csv')
```

```
dat <- dat %>%
  filter(skew != 'control')
dat <- dat %>%
  mutate(cho = ifelse(true_response == 'f', 1, -1))
```

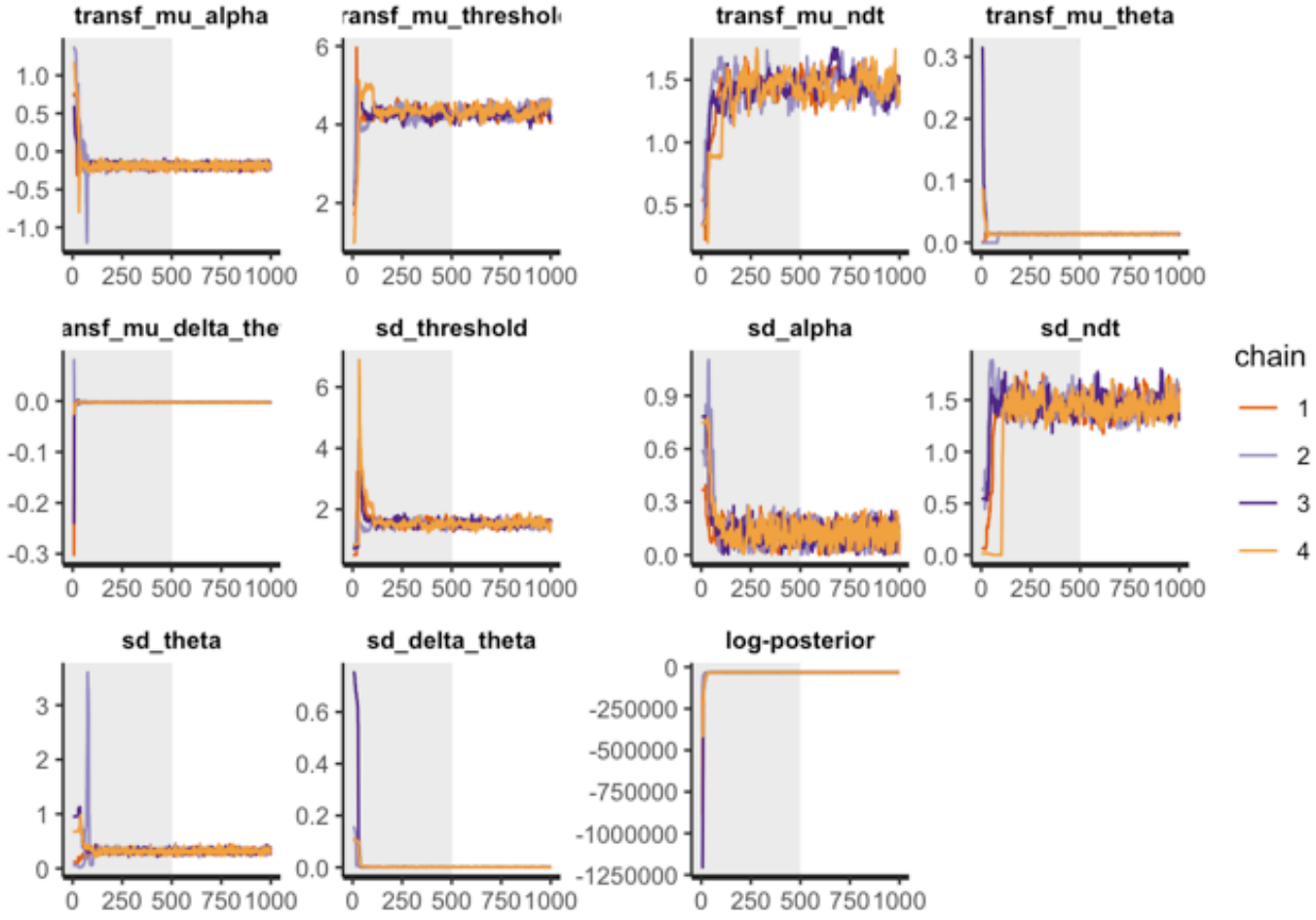
```
ids <- unique(dat$Prolific_ID)
for(j in 1:length(ids)){
  dat$tid[dat$Prolific_ID==ids[j]] <- j
}
tids <- unique(dat$tid)

dat$rt <- as.numeric(dat$rt/1000)

dat <- dat %>%
  filter(test_part == 'cc' | test_part == 'ss')

dat <- dat %>%
  mutate(con = ifelse(test_part == "cc", 1, -1))
```

```
# only condition no time pressure
dataList  = list(cho = dat$cho, rt = dat$rt, participant = dat$tid, N=nrow(dat),
L = length(tids), starting_point=0.5, evd = dat$evd, sdd = dat$sdd, con =  dat$co
n)
```

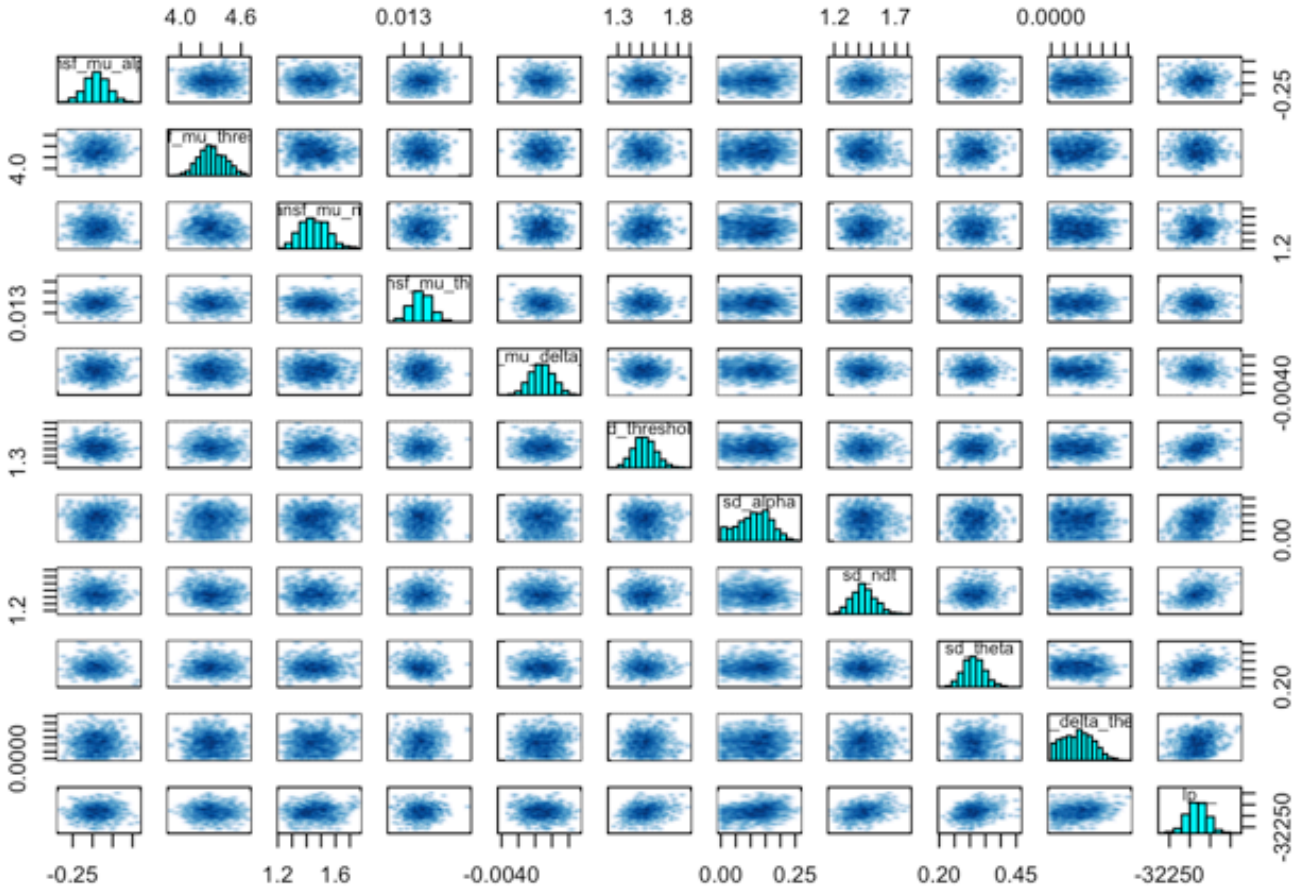
```
parameters = c("transf_mu_alpha","transf_mu_threshold","transf_mu_ndt", "transf_mu_theta","transf_mu_delta_theta", 'sd_threshold',"sd_alpha","sd_ndt", 'sd_theta', 'sd_delta_theta', "alpha_sbj","threshold_sbj","ndt_sbj",'theta_sbj','delta_theta_sbj', "log_lik")

initFunc <-function (i) {
  initList=list()
  for (ll in 1:i){
    initList[[ll]] = list(mu_alpha = runif(1, -5, 5),
                          sd_alpha = runif(1,0,1),
                          mu_threshold = runif(1,-0.5, 5),
                          sd_threshold = runif(1, 0, 1),
                          mu_ndt = runif(1, -1.5, 0),
                          sd_ndt = runif(1, 0, 1),
                          mu_theta = runif(1,-20, 1),
                          sd_theta = runif(1,0,1),
                          mu_delta_theta = runif(1, -1, 1),
                          sd_delta_theta = runif(1,0,1),
                          z_alpha = runif(length(tids),-0.1,0.1),
                          z_theta = runif(length(tids),-0.1,0.1),
                          z_threshold = runif(length(tids),-0.1,0.1),
                          z_ndt = runif(length(tids),-0.1,0.1),
                          z_delta_theta = runif(length(tids),-0.1,0.1)

    )
  }

  return(initList)
}
```

```
m <- stan_model("MV_Baseline.stan")
dsamples <- sampling(m,
                data=dataList,
                pars=parameters,
                iter=1000,
                chains=4,#If not specified, gives random inits
                init = initFunc(4),
                warmup = 500,  # Stands for burn-in; Default = iter/2
                seed = 12
                )
```

```
#parameters = c("transf_mu_alpha","transf_mu_threshold","transf_mu_ndt", "transf_mu_theta","transf_mu_delta_theta", 'sd_threshold',"sd_alpha","sd_ndt", 'sd_theta', 'sd_delta_theta', "alpha_sbj","threshold_sbj","ndt_sbj",'theta_sbj','delta_theta_sbj', "log_lik")
rstan::traceplot(dsamples, pars=c("transf_mu_alpha","transf_mu_threshold","transf_mu_ndt", "transf_mu_theta","transf_mu_delta_theta", 'sd_threshold',"sd_alpha","sd_ndt", 'sd_theta', 'sd_delta_theta', "lp__"), inc_warmup = TRUE, nrow = 3)
```




```
pairs(dsamples, pars = c("transf_mu_alpha","transf_mu_threshold","transf_mu_ndt",
"transf_mu_theta","transf_mu_delta_theta", 'sd_threshold',"sd_alpha","sd_ndt", 'sd
_theta', 'sd_delta_theta', "lp__"))
```

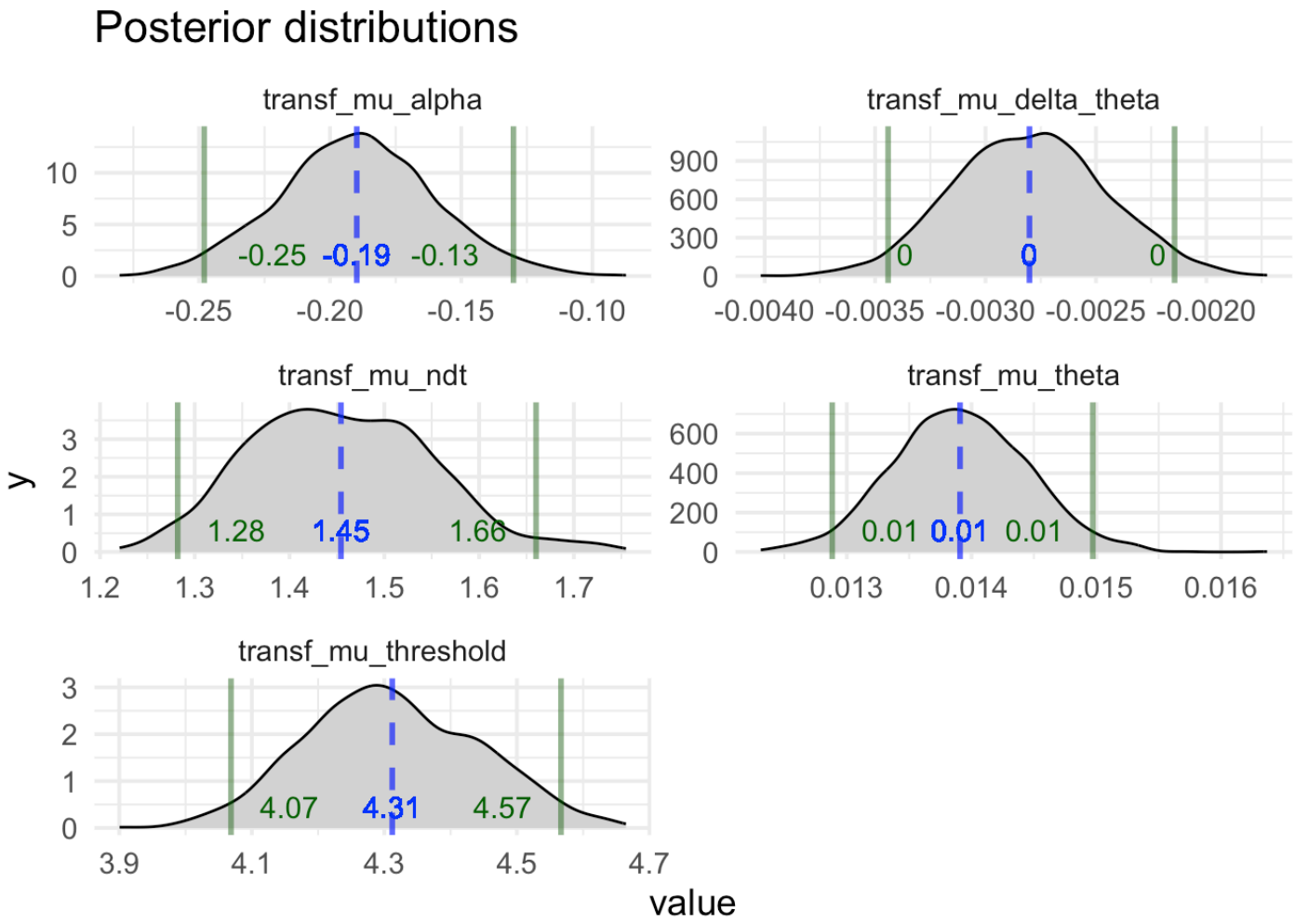



```
print(dsamples, pars = c("transf_mu_alpha","transf_mu_threshold","transf_mu_ndt",
"transf_mu_theta","transf_mu_delta_theta", 'sd_threshold',"sd_alpha","sd_ndt", 'sd
_theta', 'sd_delta_theta', "lp__"))
```

```
## Inference for Stan model: anon_model.
## 4 chains, each with iter=1000; warmup=500; thin=1;
## post-warmup draws per chain=500, total post-warmup draws=2000.
##
##                           mean se_mean    sd      2.5%       25%       50%
## transf_mu_alpha          -0.19    0.00  0.03     -0.25     -0.21     -0.19
## transf_mu_threshold       4.31    0.01  0.13      4.07      4.22      4.30
## transf_mu_ndt             1.45    0.01  0.10      1.28      1.38      1.45
## transf_mu_theta           0.01    0.00  0.00      0.01      0.01      0.01
## transf_mu_delta_theta     0.00    0.00  0.00      0.00      0.00      0.00
## sd_threshold              1.52    0.01  0.09      1.35      1.46      1.52
## sd_alpha                  0.11    0.00  0.06      0.01      0.07      0.12
## sd_ndt                    1.44    0.01  0.10      1.27      1.37      1.44
## sd_theta                  0.31    0.00  0.03      0.25      0.29      0.31
## sd_delta_theta            0.00    0.00  0.00      0.00      0.00      0.00
## lp__                 -32181.69    1.75 26.04 -32234.43 -32199.01 -32181.22
##                           75%     97.5% n_eff Rhat
## transf_mu_alpha          -0.17     -0.13  2198 1.00
## transf_mu_threshold       4.41      4.57    77 1.04
## transf_mu_ndt             1.52      1.66    55 1.07
## transf_mu_theta           0.01      0.01  1062 1.00
## transf_mu_delta_theta     0.00      0.00  1922 1.00
## sd_threshold              1.58      1.72   147 1.03
## sd_alpha                  0.15      0.21   305 1.01
## sd_ndt                    1.50      1.64   126 1.05
## sd_theta                  0.34      0.39   955 1.00
## sd_delta_theta            0.00      0.00   502 1.00
## lp__                 -32164.62 -32131.46   222 1.04
##
## Samples were drawn using NUTS(diag_e) at Thu Dec 21 14:29:37 2023.
## 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).
```

```
library(ggplot2)
library(tidyverse) # for the gather function

samples_matrix <- as.matrix(dsamples)
means <- colMeans(samples_matrix)
hpd_interval <- t(apply(samples_matrix, 2, function(x) quantile(x, probs=c(0.025,
0.975))))



parameters <- c("transf_mu_alpha", "transf_mu_theta", "transf_mu_threshold",
                "transf_mu_ndt", "transf_mu_delta_theta")

# Reshape data to a long format
df_long <- as.data.frame(samples_matrix) %>%
```

```
  gather(key = "parameter", value = "value", parameters)

# Convert hpd_interval to a data frame and name the columns
hpd_interval_sub <- hpd_interval[parameters, ]
hpd_df <- as.data.frame(hpd_interval_sub)
colnames(hpd_df) <- c("lower", "upper")
rownames(hpd_df) <- parameters
hpd_df$parameter <- rownames(hpd_df)



# Aesthetic enhancements
theme_set(theme_minimal(base_size = 14)) # Set the default theme

custom_palette <- c("density_fill" = "lightgray",
                    "mean_line" = "blue",
                    "hpd_line" = "darkgreen")

# Add text labels for mean, lower, and upper HPD values
df_long <- df_long %>%
  group_by(parameter) %>%
  mutate(mean = means[parameter])

hpd_df <- hpd_df %>%
  mutate(mid = (lower + upper) / 2)

p <- ggplot(df_long, aes(x = value)) +
  geom_density(aes(fill = "density_fill")) +
  scale_fill_manual(values = custom_palette, guide = FALSE) +
  geom_vline(aes(xintercept = mean, color = "mean_line"), linetype = "dashed", siz
e = 1, alpha = 0.7) +
  geom_text(data = df_long, aes(x = mean, y = 0, label = round(mean, 2)), vjust =
-0.5, hjust = 0.5, size = 4, color = custom_palette["mean_line"]) +
  geom_vline(data = hpd_df, aes(xintercept = lower, color = "hpd_line"), linetype
= "solid", size = 1, alpha = 0.5) +
  geom_text(data = hpd_df, aes(x = lower, y = 0, label = round(lower, 2)), vjust =
-0.5, hjust = -0.5, size = 4, color = custom_palette["hpd_line"]) +
  geom_vline(data = hpd_df, aes(xintercept = upper, color = "hpd_line"), linetype
= "solid", size = 1, alpha = 0.5) +
  geom_text(data = hpd_df, aes(x = upper, y = 0, label = round(upper, 2)), vjust =
-0.5, hjust = 1.5, size = 4, color = custom_palette["hpd_line"]) +
  facet_wrap(~ parameter, scales = "free", ncol = 2) +
  scale_color_manual(values = custom_palette, guide = FALSE) +
  labs(title = "Posterior distributions")

print(p)
```


Posterior distributions
transf_mu_alpha
-0.25 -0.19 -0.13
transf_mu_delta_theta
0 0 0
transf_mu_ndt
1.28 1.45 1.66
transf_mu_theta
0.01 0.01 0.01
transf_mu_threshold
4.07 4.31 4.57
y
value