```
knitr::opts_chunk$set(message = FALSE, warning = FALSE)
```

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



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 <- dat %>%
  filter(test_part == 'cs' | test_part == 'sc')

dat <- dat %>%
  mutate(
    oa_complex = ifelse(test_part == 'cs', 1, -1),
    evd = evd * oa_complex,
    sdd = sdd * oa_complex,
    chose_complex = ifelse((oa_complex == 1 & cho == 1) | (oa_complex == -1 & cho
== -1), 1, -1)
  )

dat$rt <- dat$rt/1000



dat$P_A1 <- dat$P_A1 / 100
dat$P_A2 <- dat$P_A2 / 100
dat$P_B1 <- dat$P_B1 / 100
dat$P_B2 <- dat$P_B2 / 100
```

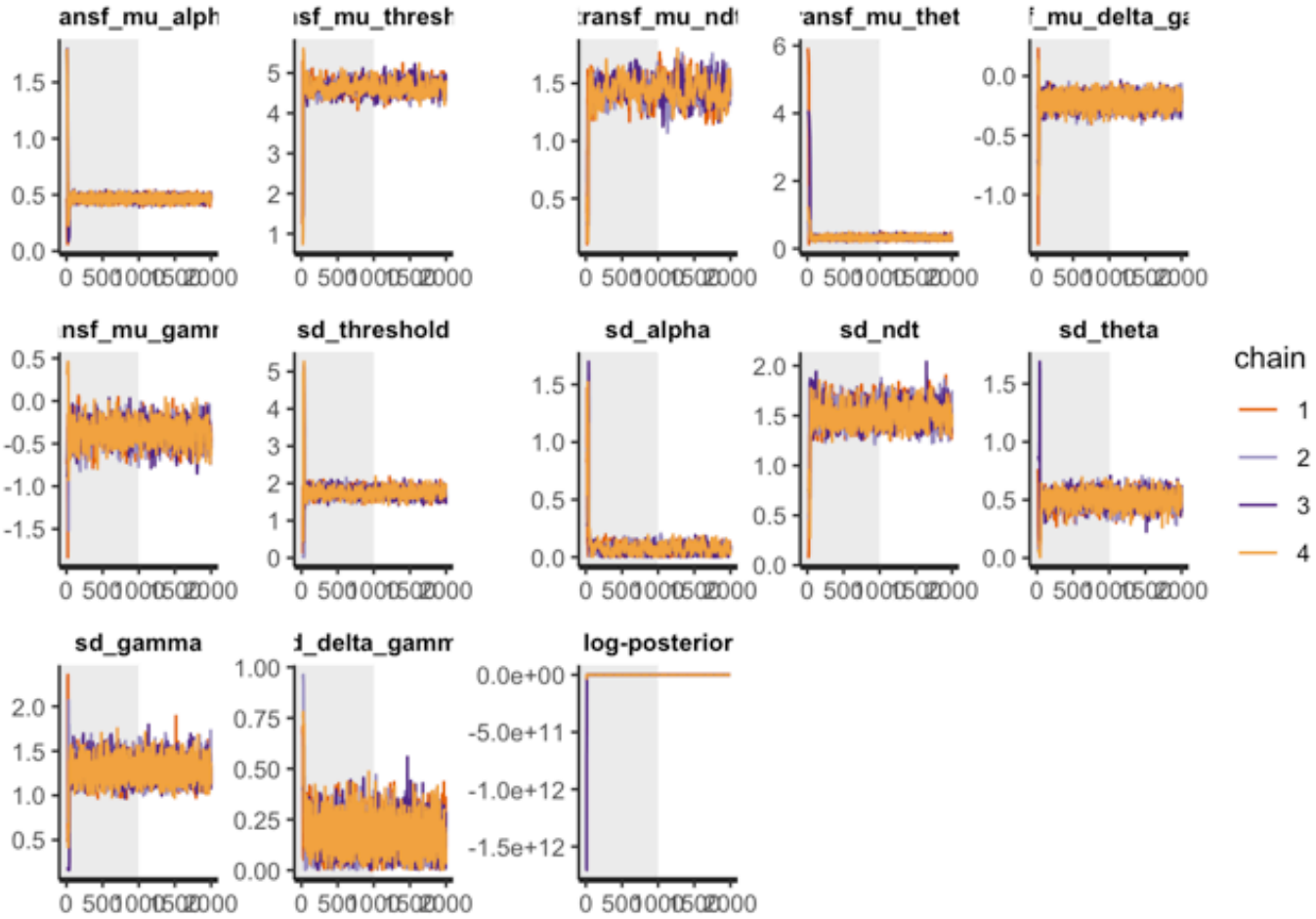
```
library(dplyr)
library(stringr)


df <- dat %>%
  # Swap values if oa_condition is not 0
  rowwise() %>%
  mutate(
    oc1 = if_else(test_part == 'sc', O_B1, O_A1),
    oc2 = if_else(test_part == 'sc', O_B2, O_A2),
    pc1 = if_else(test_part == 'sc', P_B1, P_A1),
    pc2 = if_else(test_part == 'sc', P_B2, P_A2),
    os1 = if_else(test_part == 'sc', O_A1, O_B1),
    os2 = if_else(test_part == 'sc', O_A2, O_B2),
    ps1 = if_else(test_part == 'sc', P_A1, P_B1),
    ps2 = if_else(test_part == 'sc', P_A2, P_B2),
  ) %>%
  ungroup()
```

```
df <- df %>%
  mutate(index1 = as.numeric(ifelse(oc1<oc2, 1, -1)) ,
         index2 = as.numeric(ifelse(os1<os2, 1, -1)),)

df <- df %>%
  # Swap values if oa_condition is not 0
  rowwise() %>%
  mutate(
    oc3 = if_else(index1 == 1, oc1, oc2),
    oc4 = if_else(index1 == 1, oc2, oc1),
    pc3 = if_else(index1 == 1, pc1, pc2),
    pc4 = if_else(index1 == 1, pc2, pc1),
    os3 = if_else(index2 == 1, os1, os2),
    os4 = if_else(index2 == 1, os2, os1),
    ps3 = if_else(index2 == 1, ps1, ps2),
    ps4 = if_else(index2 == 1, ps2, ps1),
  ) %>%
  ungroup()
```

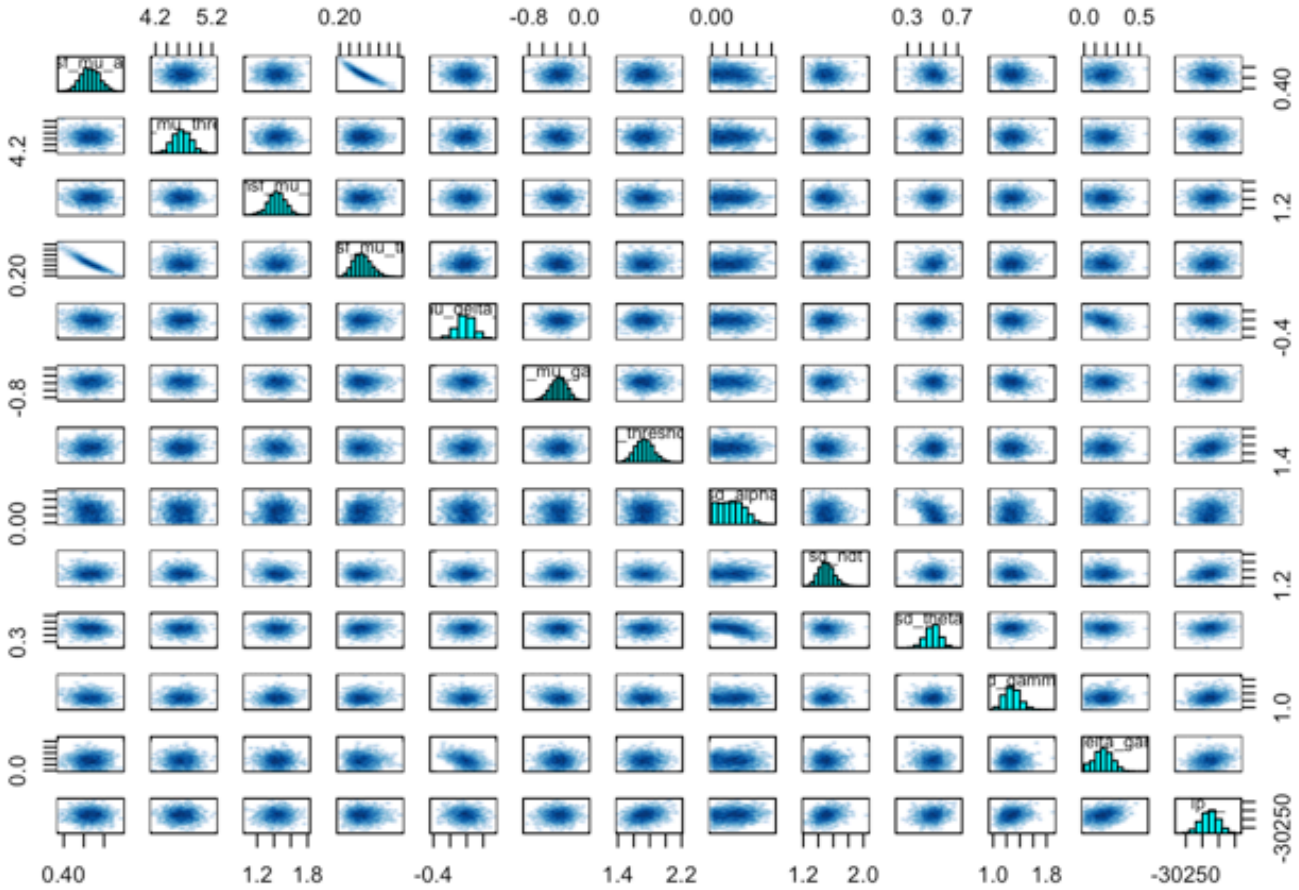
```
dataList  = list(cho = df$chose_complex, rt = df$rt, participant = df$tid,N=nrow(df),  L = length(tids),
                 oc = as.matrix(df[, c("oc3", "oc4")]),
                 os = as.matrix(df[, c("os3", "os4")]),
                 pc = as.matrix(df[, c("pc3", "pc4")]),
                 ps = as.matrix(df[, c("ps3", "ps4")]),
                 starting_point = 0.5
                 )

parameters = c("transf_mu_alpha","transf_mu_threshold","transf_mu_ndt", "transf_mu_theta",'transf_mu_delta_gamma', 'transf_mu_gamma', 'sd_threshold',"sd_alpha","sd_ndt", 'sd_theta', 'sd_gamma','sd_delta_gamma', "alpha_sbj","threshold_sbj","ndt_sbj",'theta_sbj', 'gamma_sbj', 'delta_gamma_sbj', "log_lik")
```

```
initFunc <-function (i) {
  initList=list()
  for (ll in 1:i){
    initList[[ll]] = list(
                          mu_alpha = runif(1,-1.4587,2.5413),
                          sd_alpha = runif(1,0,1),
                          mu_threshold = runif(1,-0.5, 2.5),
                          sd_threshold = runif(1,0,1),
                          mu_ndt = runif(1, -1.5, 0),
                          sd_ndt = runif(1, 0, 1),
                          mu_theta = runif(1,0, 6),
                          sd_theta = runif(1,0,1),
                          mu_gamma = runif(1,-1, 1),
                          sd_gamma = runif(1, 0, 1),
                          mu_delta_gamma = runif(1,-1, 1),
                          sd_delta_gamma = 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_gamma = runif(length(tids),-0.1,0.1),
                          z_delta_gamma = runif(length(tids),-0.1,0.1)
    )
  }

  return(initList)
}
```

```
#"transf_mu_alpha","transf_mu_threshold","transf_mu_ndt", "transf_mu_theta",'transf_mu_delta', 'transf_mu_gamma', 'sd_threshold',"sd_alpha","sd_ndt", 'sd_theta', 'sd_gamma','sd_delta', "alpha_sbj","threshold_sbj","ndt_sbj",'theta_sbj', 'gamma_sbj', 'delta_sbj',
rstan::traceplot(dsamples, pars=c("transf_mu_alpha","transf_mu_threshold","transf_mu_ndt", "transf_mu_theta",'transf_mu_delta_gamma', 'transf_mu_gamma', 'sd_threshold',"sd_alpha","sd_ndt", 'sd_theta', 'sd_gamma','sd_delta_gamma', "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_gamma', 'transf_mu_gamma', 'sd_threshold',"sd_alpha","sd_ndt", 'sd_theta', 'sd_gamma','sd_delta_gamma', "lp__"))
```

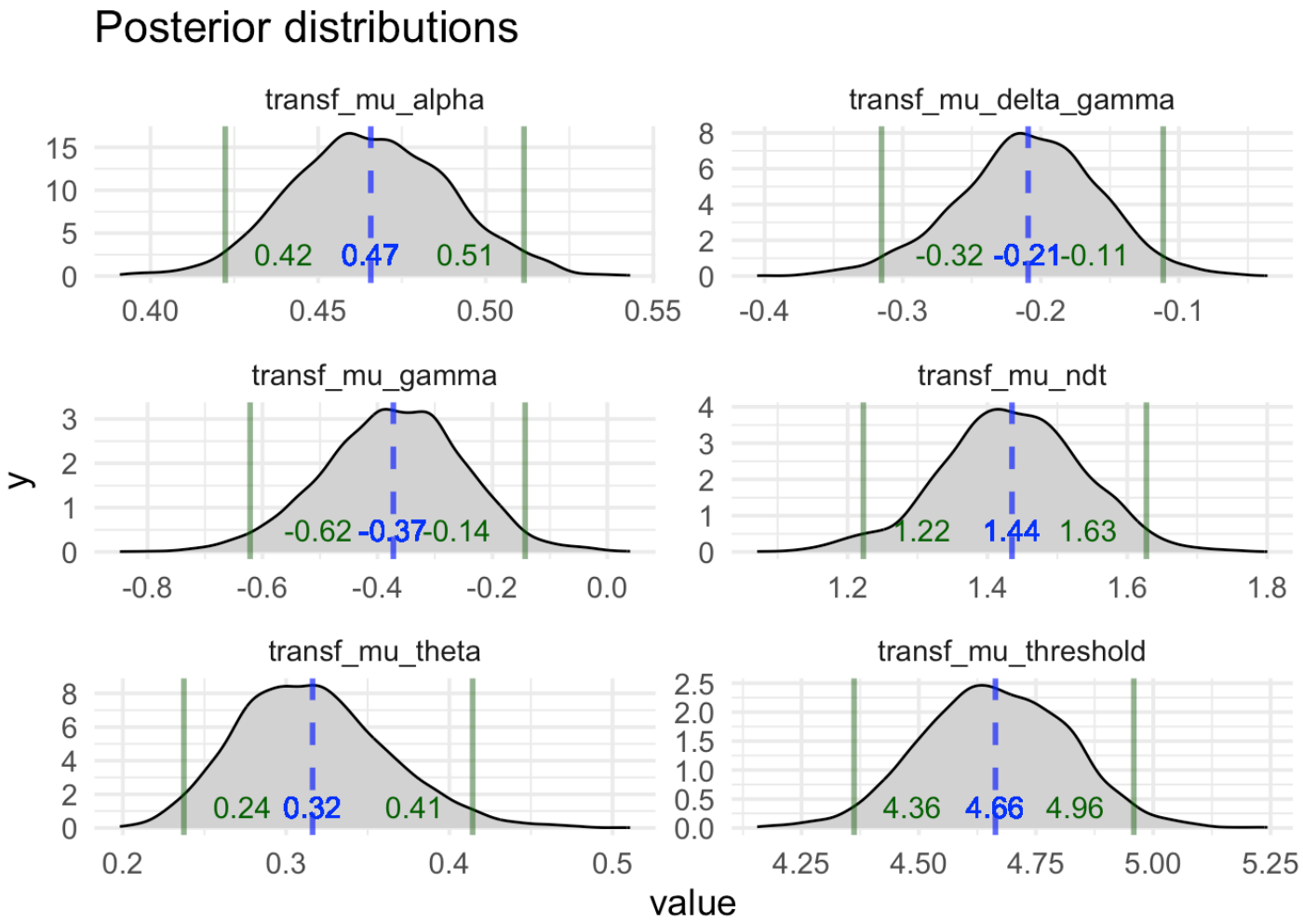



```
print(dsamples, pars = c("transf_mu_alpha","transf_mu_threshold","transf_mu_ndt",
"transf_mu_theta",'transf_mu_delta_gamma', 'transf_mu_gamma', 'sd_threshold',"sd_a
lpha","sd_ndt", 'sd_theta', 'sd_gamma','sd_delta_gamma', "lp__"))
```

```
## 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%
## transf_mu_alpha           0.47    0.00  0.02      0.42      0.45      0.47
## transf_mu_threshold       4.66    0.01  0.16      4.36      4.56      4.66
## transf_mu_ndt             1.44    0.01  0.10      1.22      1.37      1.43
## transf_mu_theta           0.32    0.00  0.05      0.24      0.28      0.31
## transf_mu_delta_gamma    -0.21    0.00  0.05     -0.32     -0.24     -0.21
## transf_mu_gamma          -0.37    0.00  0.12     -0.62     -0.45     -0.37
## sd_threshold              1.73    0.01  0.11      1.52      1.66      1.73
## sd_alpha                  0.07    0.00  0.04      0.00      0.03      0.07
## sd_ndt                    1.51    0.00  0.10      1.33      1.44      1.50
## sd_theta                  0.50    0.00  0.06      0.38      0.46      0.50
## sd_gamma                  1.28    0.00  0.12      1.08      1.20      1.28
## sd_delta_gamma            0.17    0.00  0.08      0.02      0.12      0.17
## lp__                 -30177.45    0.99 26.67 -30229.62 -30194.98 -30177.11
##                           75%     97.5% n_eff Rhat
## transf_mu_alpha           0.48      0.51  4054 1.00
## transf_mu_threshold       4.77      4.96   268 1.02
## transf_mu_ndt             1.50      1.63   171 1.03
## transf_mu_theta           0.34      0.41  4162 1.00
## transf_mu_delta_gamma    -0.17     -0.11  2231 1.00
## transf_mu_gamma          -0.29     -0.14   602 1.01
## sd_threshold              1.81      1.97   496 1.01
## sd_alpha                  0.10      0.15   190 1.02
## sd_ndt                    1.57      1.73   573 1.01
## sd_theta                  0.54      0.61   621 1.00
## sd_gamma                  1.35      1.53  1652 1.00
## sd_delta_gamma            0.23      0.33   856 1.00
## lp__                 -30159.56 -30126.01   724 1.01
##
## Samples were drawn using NUTS(diag_e) at Wed Jan 17 13:33:27 2024.
## 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_threshold","transf_mu_ndt", "transf_m
```

```
u_theta",'transf_mu_delta_gamma', 'transf_mu_gamma')

# 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 = 'none') +
  labs(title = "Posterior distributions")

print(p)
```


Posterior distributions
transf_mu_alpha
0.42 0.47 0.51
transf_mu_delta_gamma
-0.32 -0.21 -0.11
transf_mu_gamma
-0.62 -0.37 -0.14
transf_mu_ndt
1.22 1.44 1.63
transf_mu_theta
0.24 0.32 0.41
transf_mu_threshold
4.36 4.66 4.96
y
value