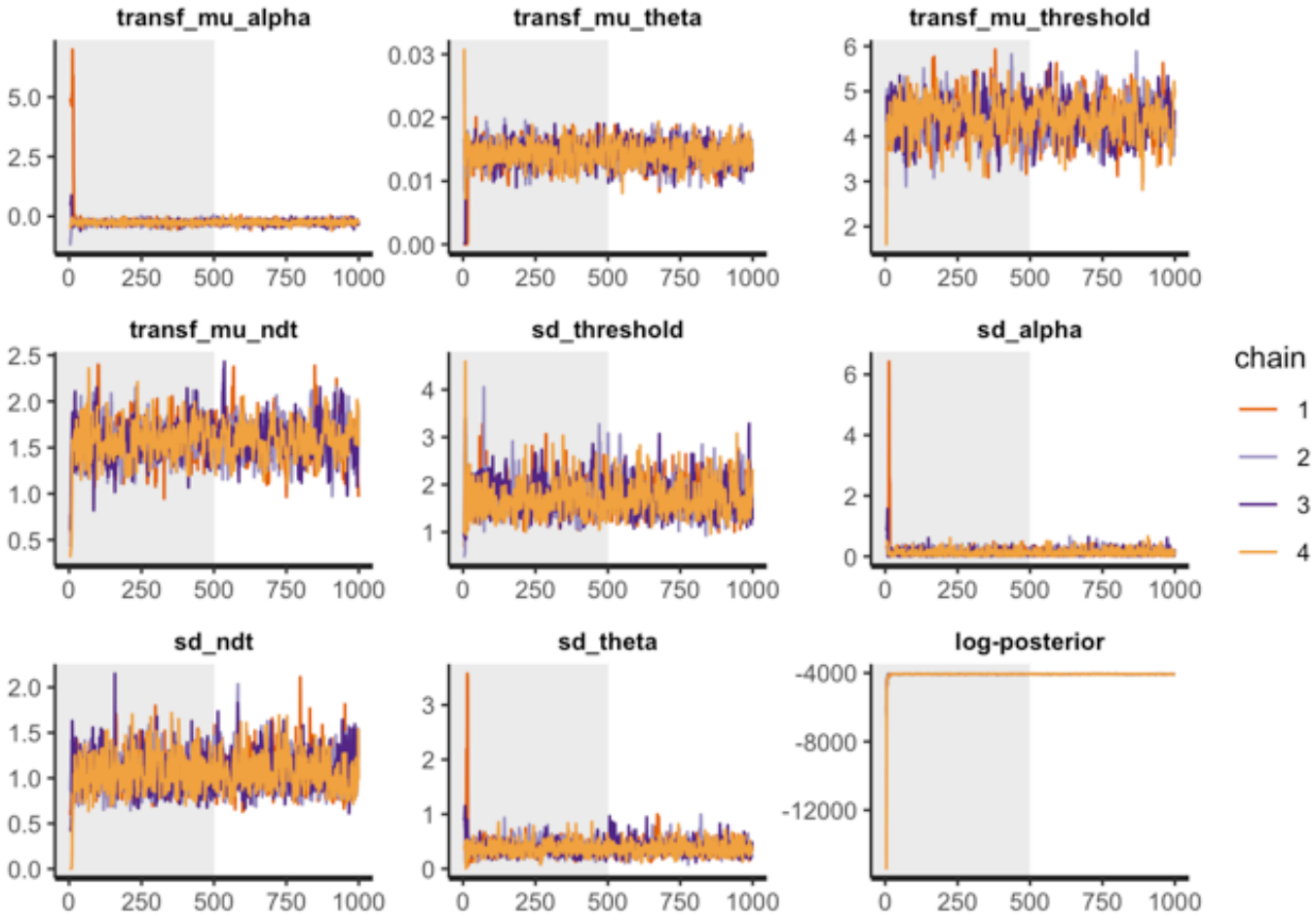
```
############### 0 - safe choice A, 1 - risky choice B ####################
library(rstan); rstan_options(javascript=FALSE)
library(bayesplot)
library(dplyr)
options(mc.cores = parallel::detectCores())
rstan_options(auto_write = T)

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

dat = dat %>%
  mutate(cho = 0,
         cho = ifelse(response == "f", 1*risk_index, cho),
         cho = ifelse(response == "j", -1*risk_index, cho))

dat <- dat %>%
  filter(skew != 'control')


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

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

dat$rt <- dat$rt/1000

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)
  )
```

```
dataList  = list(cho = dat$chose_complex, rt = dat$rt, participant = dat$tid, N=nr
ow(dat),  L = length(tids), starting_point=0.5, evd = dat$evd, sdd = dat$sdd)
```

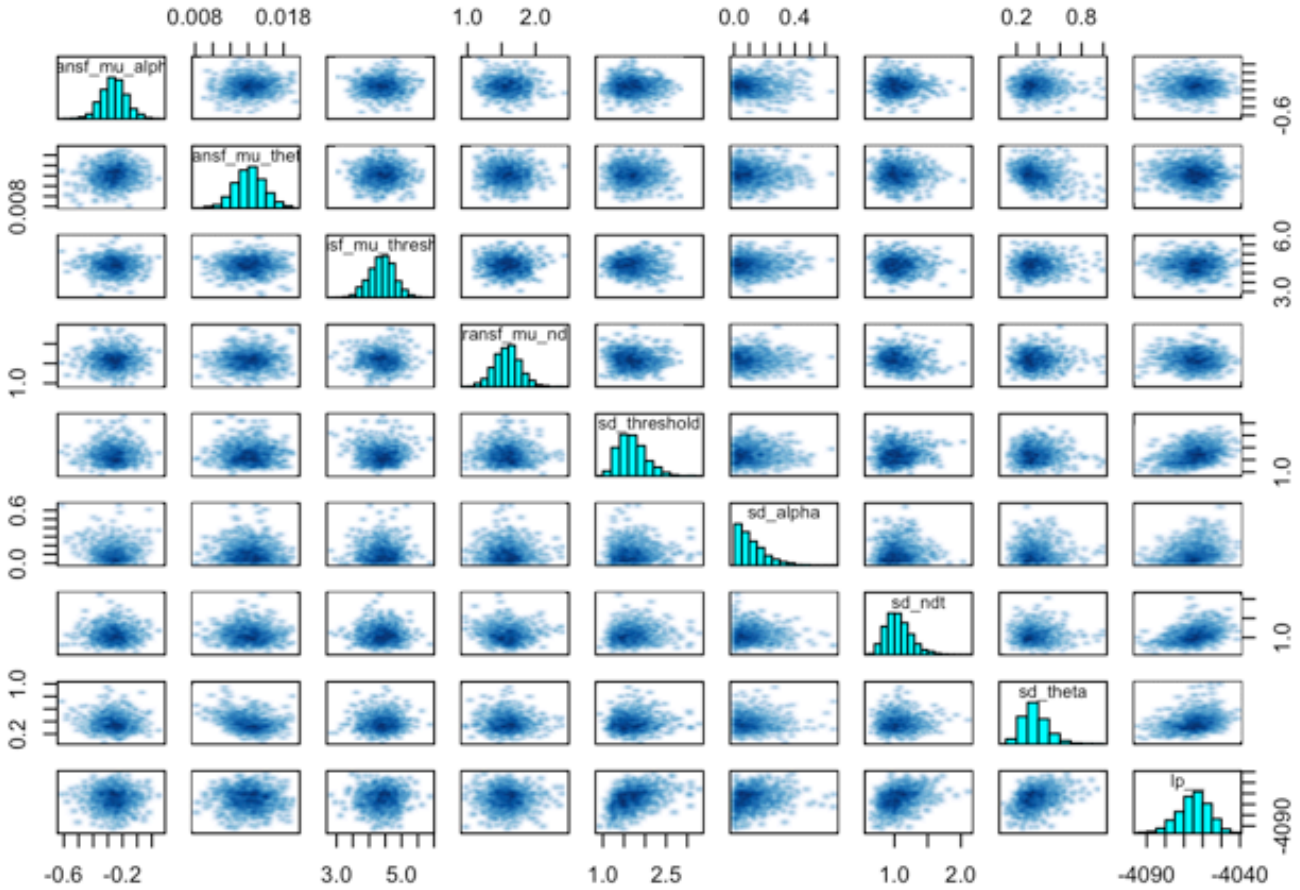
```
parameters = c("transf_mu_alpha","transf_mu_threshold","transf_mu_ndt", "transf_mu
_theta", 'sd_threshold',"sd_alpha","sd_ndt", 'sd_theta',  "alpha_sbj","threshold_s
bj","ndt_sbj",'theta_sbj', "log_lik")

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),
                          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)

    )
  }

  return(initList)
}
```

```
m <- stan_model("MV_Baseline ce.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, # Setting seed; Default is random seed
                refresh = 0
                )
```

```
#parameters = c("transf_mu_alpha","transf_mu_threshold","transf_mu_ndt", "transf_m
u_theta",'sd_threshold',"sd_alpha","sd_ndt", 'sd_theta', "alpha_sbj","threshold_sb
j","ndt_sbj",'theta_sbj',"log_lik")

rstan::traceplot(dsamples, pars=c("transf_mu_alpha","transf_mu_theta", "transf_mu_
threshold","transf_mu_ndt", 'sd_threshold',"sd_alpha","sd_ndt", 'sd_theta', "lp_
_"), inc_warmup = TRUE, nrow = 3)
```




```
pairs(dsamples, pars = c("transf_mu_alpha","transf_mu_theta","transf_mu_threshol
d","transf_mu_ndt", 'sd_threshold',"sd_alpha","sd_ndt", 'sd_theta', "lp__"))
```

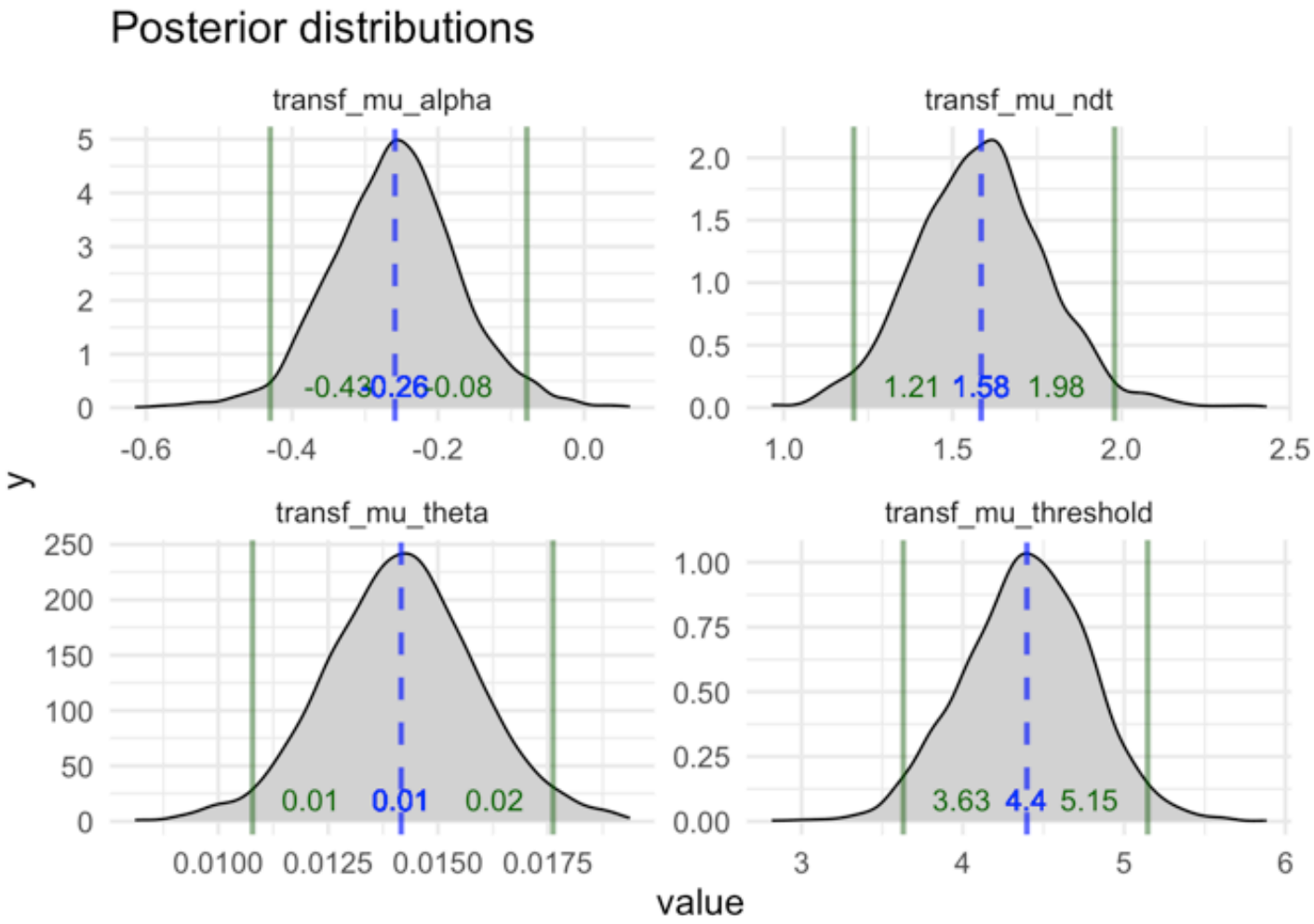



```
print(dsamples, pars = c("transf_mu_alpha", "transf_mu_theta", "transf_mu_threshol
d","transf_mu_ndt", 'sd_threshold',"sd_alpha","sd_ndt", 'sd_theta', "lp__"))
```

```
## Inference for Stan model: MV_Baseline ce.
## 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%      75%
## transf_mu_alpha         -0.26    0.00 0.09    -0.43    -0.31    -0.26    -0.20
## transf_mu_theta          0.01    0.00 0.00     0.01     0.01     0.01     0.02
## transf_mu_threshold      4.40    0.02 0.39     3.63     4.14     4.40     4.66
## transf_mu_ndt            1.58    0.01 0.20     1.21     1.45     1.58     1.71
## sd_threshold             1.70    0.01 0.33     1.18     1.45     1.67     1.89
## sd_alpha                 0.12    0.00 0.10     0.00     0.04     0.09     0.16
## sd_ndt                   1.06    0.01 0.19     0.76     0.92     1.04     1.18
## sd_theta                 0.37    0.00 0.12     0.18     0.29     0.35     0.44
## lp__                 -4064.84    0.44 8.46 -4082.99 -4070.02 -4064.36 -4059.20
##                         97.5% n_eff Rhat
## transf_mu_alpha         -0.08  2137 1.00
## transf_mu_theta          0.02   641 1.00
## transf_mu_threshold      5.15   302 1.01
## transf_mu_ndt            1.98   285 1.00
## sd_threshold             2.45   547 1.00
## sd_alpha                 0.36  1041 1.00
## sd_ndt                   1.51   491 1.01
## sd_theta                 0.65   728 1.01
## lp__                 -4048.97   365 1.01
##
## Samples were drawn using NUTS(diag_e) at Wed Nov 22 14:57:39 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")

# 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
-0.43 -0.26 -0.08
transf_mu_ndt
1.21 1.58 1.98
transf_mu_theta
0.01 0.01 0.02
transf_mu_threshold
3.63 4.4 5.15
y
value