```
library(rstan); rstan_options(javascript=FALSE)
library(dplyr)
options(mc.cores = parallel::detectCores())
rstan_options(auto_write = T)
```

```
dat <- read.csv('final_data.csv')
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$rt <- as.numeric(dat$rt/1000)

dat <- dat %>%
  filter(test_part == 'cc' | test_part == 'ss',
         rt >= 1,
         subject != '4ld6kjtr',
         subject != 'm73bj2hn')

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

```
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
```

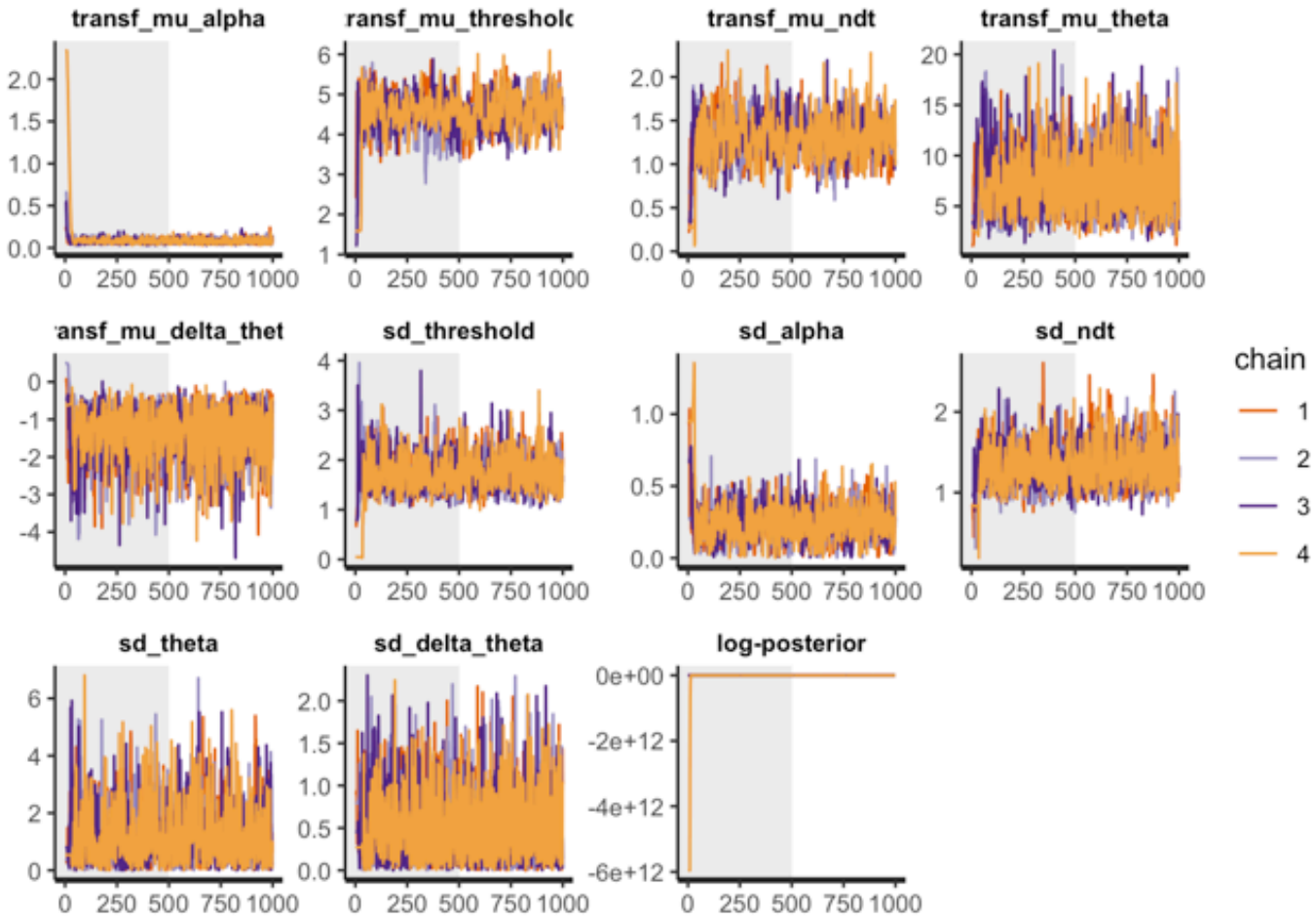
```
oa = as.matrix(dat[, c("O_A1", "O_A2")])
ob = as.matrix(dat[, c("O_B1", "O_B2")])
pa = as.matrix(dat[, c("P_A1", "P_A2")])
pb = as.matrix(dat[, c("P_B1", "P_B2")])
```

```
dataList  = list(cho = dat$cho,rt = dat$rt, participant = dat$tid,N=nrow(dat),  L
= length(tids),starting_point=0.5,
                 oa = as.matrix(dat[, c("O_A1", "O_A2")]),
                 ob = as.matrix(dat[, c("O_B1", "O_B2")]),
                 pa = as.matrix(dat[, c("P_A1", "P_A2")]),
                 pb = as.matrix(dat[, c("P_B1", "P_B2")]),
                 con =  dat$con
                 )

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_sb
j', "log_lik")
```

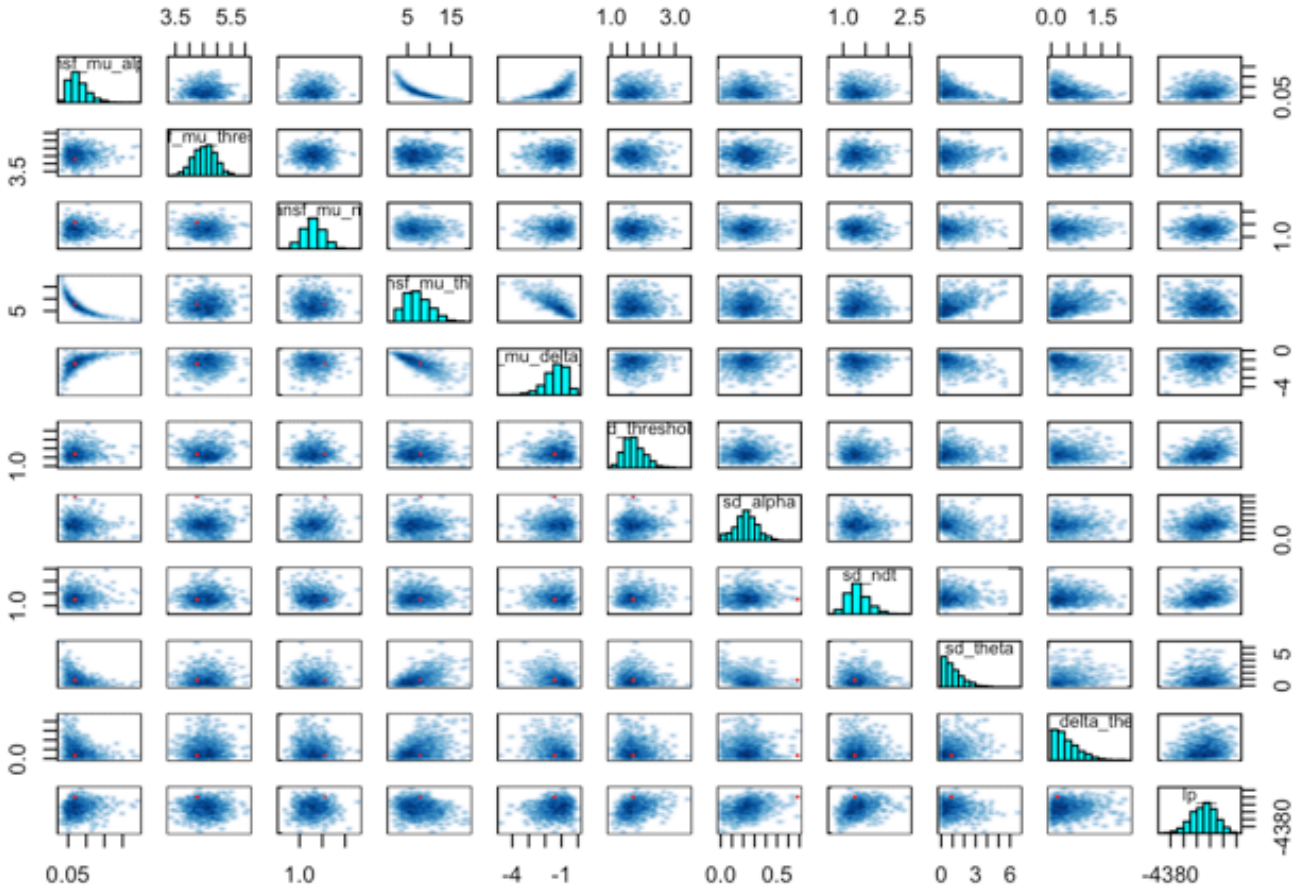
```
initFunc <-function (i) {
  initList=list()
  for (ll in 1:i){
    initList[[ll]] = list(mu_alpha = runif(1,-1.4578,2.5413),
                          sd_alpha = runif(1,0,1),
                          mu_threshold = runif(1,-0.5, 2.5),
                          sd_threshold = runif(1,0,1),
                          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_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("EU_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,  # Setting seed; Default is random seed
                refresh = 0
                )
```

```
#"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__"))
```

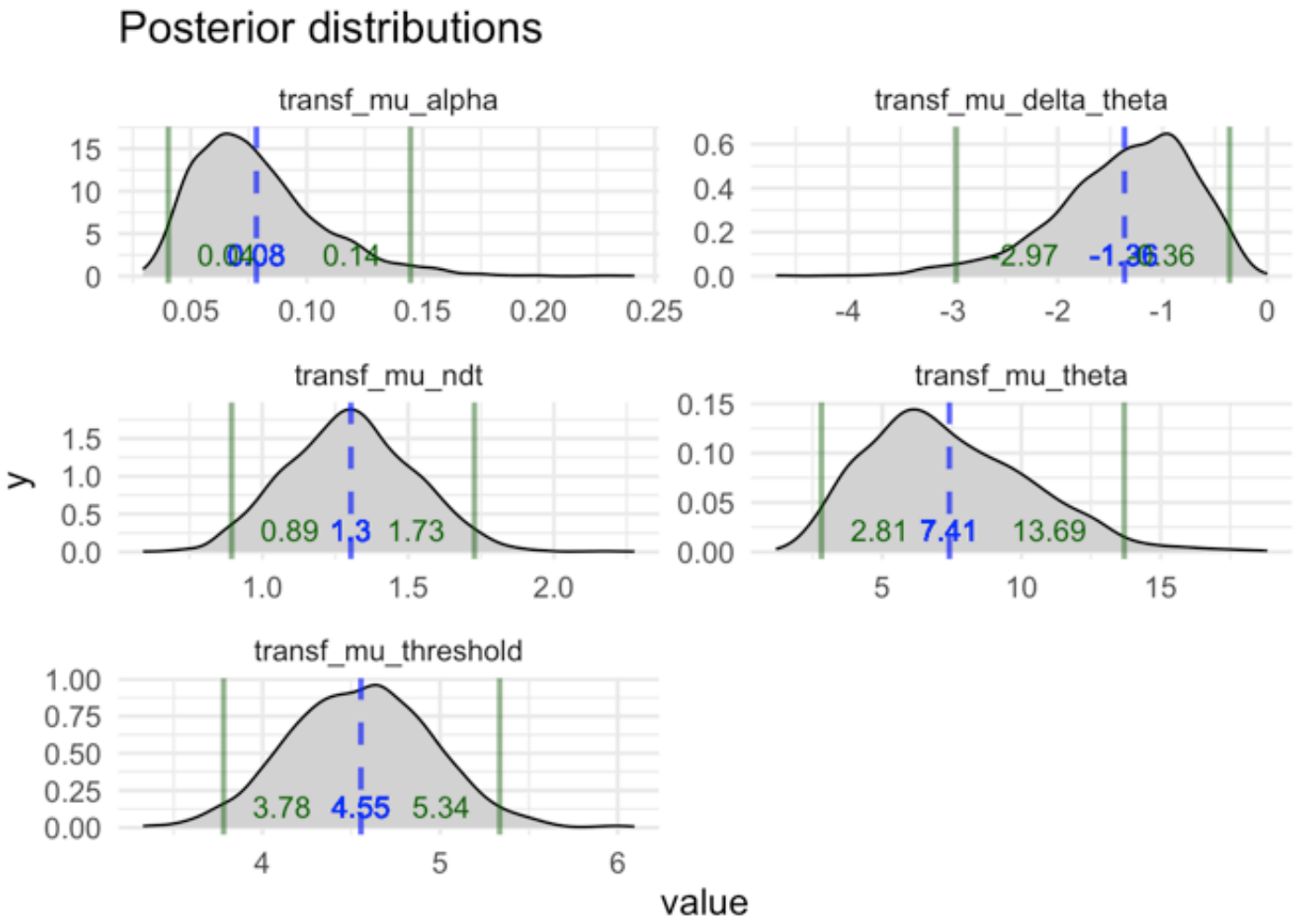



```
print(dsamples, pars = c("transf_mu_alpha","transf_mu_threshold","transf_mu_ndt",
"transf_mu_theta","transf_mu_delta_theta", 'sd_threshold',"sd_alpha","sd_ndt", 'sd
_theta', 'sd_delta_theta', "lp__"))
```

```
## Inference for Stan model: EU_Baseline.
## 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          0.08    0.00 0.03     0.04     0.06     0.07     0.09
## transf_mu_threshold      4.55    0.02 0.40     3.78     4.27     4.56     4.83
## transf_mu_ndt            1.30    0.01 0.22     0.89     1.15     1.30     1.45
## transf_mu_theta          7.41    0.10 2.92     2.81     5.25     6.97     9.31
## transf_mu_delta_theta   -1.36    0.02 0.66    -2.97    -1.75    -1.28    -0.88
## sd_threshold             1.72    0.02 0.33     1.20     1.49     1.67     1.92
## sd_alpha                 0.23    0.00 0.11     0.02     0.17     0.23     0.29
## sd_ndt                   1.34    0.01 0.24     0.96     1.17     1.31     1.47
## sd_theta                 1.11    0.04 0.94     0.04     0.41     0.87     1.56
## sd_delta_theta           0.49    0.01 0.39     0.01     0.19     0.39     0.70
## lp__                 -4354.83    0.46 9.35 -4374.30 -4361.21 -4354.26 -4348.10
##                         97.5% n_eff Rhat
## transf_mu_alpha          0.14   736 1.00
## transf_mu_threshold      5.34   264 1.01
## transf_mu_ndt            1.73   339 1.01
## transf_mu_theta         13.69   825 1.00
## transf_mu_delta_theta   -0.36   882 1.00
## sd_threshold             2.50   427 1.02
## sd_alpha                 0.45   488 1.00
## sd_ndt                   1.88   607 1.00
## sd_theta                 3.55   540 1.01
## sd_delta_theta           1.46  1248 1.00
## lp__                 -4337.90   408 1.02
##
## Samples were drawn using NUTS(diag_e) at Wed Nov 29 14:05:14 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_threshold","transf_mu_ndt", "transf_m
u_theta","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
0.08
0.14
transf_mu_delta_theta
-2.97
-1.36
transf_mu_ndt
0.89
1.3
1.73
transf_mu_theta
2.81
7.41
13.69
transf_mu_threshold
3.78
4.55
5.34
y
value