```
############### 0 - safe choice A, 1 - risky choice B ####################
library(rstan); rstan_options(javascript=FALSE)
```

```
## Loading required package: StanHeaders
```

```
## Loading required package: ggplot2
```

```
## rstan (Version 2.21.8, GitRev: 2e1f913d3ca3)
```

```
## For execution on a local, multicore CPU with excess RAM we recommend calling
## options(mc.cores = parallel::detectCores()).
## To avoid recompilation of unchanged Stan programs, we recommend calling
## rstan_options(auto_write = TRUE)
```

```
library(bayesplot)
```

```
## This is bayesplot version 1.10.0
```

```
## - Online documentation and vignettes at mc-stan.org/bayesplot
```

```
## - bayesplot theme set to bayesplot::theme_default()
```

```
##    * Does _not_ affect other ggplot2 plots
```

```
##    * See ?bayesplot_theme_set for details on theme setting
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
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')
```

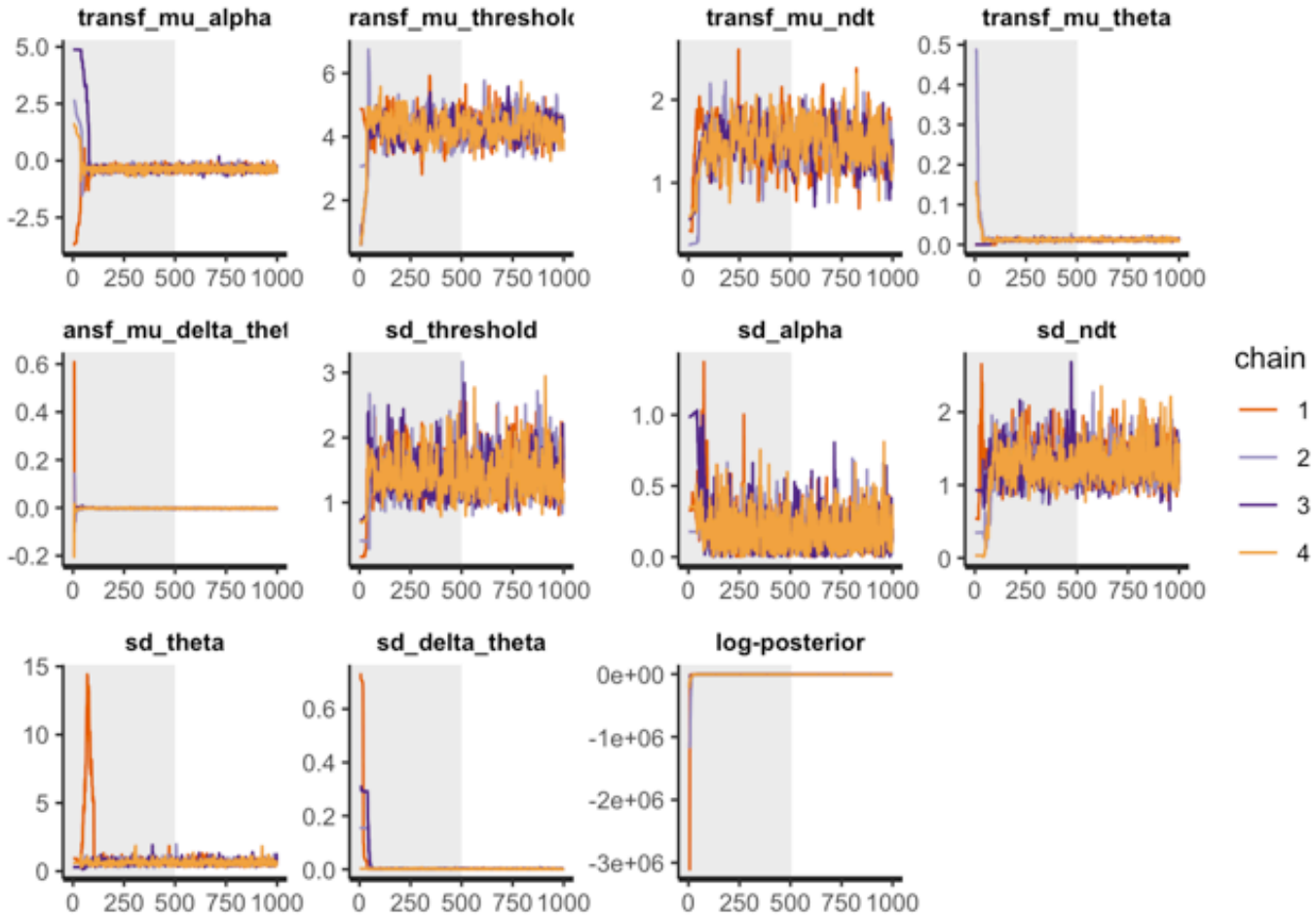
```
ids <- unique(dat$subject)
for(j in 1:length(ids)){
  dat$tid[dat$subject==ids[j]] <- j
}
tids <- unique(dat$tid)

dat <- dat %>%
  filter(test_part == 'cc' | test_part == 'ss')
dat <- dat %>%
  mutate(con = ifelse(test_part == "cc", 1, -1))
dat$rt <- dat$rt/1000
```

```
# 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)
```

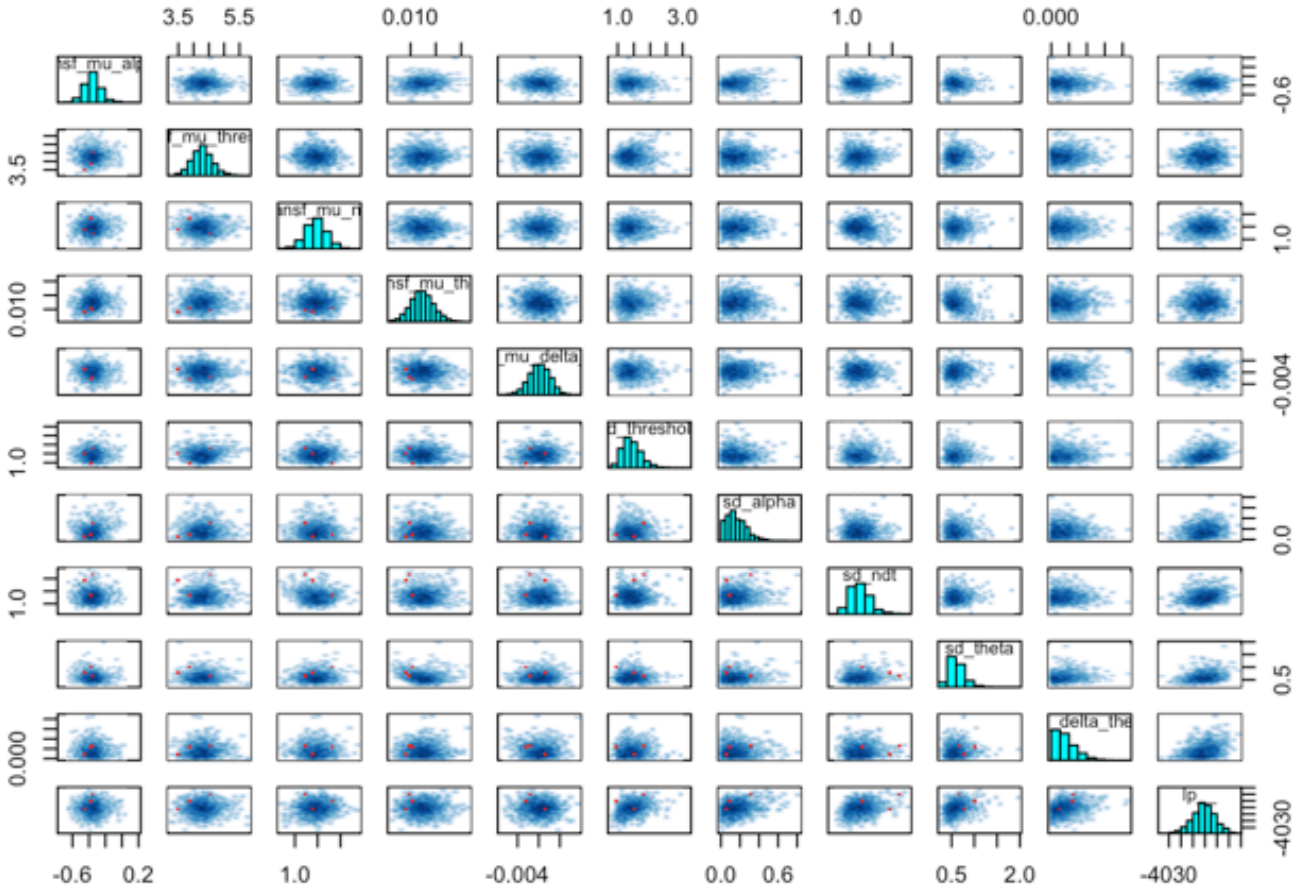
```
parameters = c("transf_mu_alpha","transf_mu_threshold","transf_mu_ndt", "transf_mu
_theta","transf_mu_delta_theta", 'sd_threshold',"sd_alpha","sd_ndt", 'sd_theta', '
sd_delta_theta', "alpha_sbj","threshold_sbj","ndt_sbj",'theta_sbj','delta_theta_sb
j', "log_lik")

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, # Setting seed; Default is random seed
                refresh = 0
                )
```

```
#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: MV_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         -0.36    0.00 0.10    -0.55    -0.42    -0.36    -0.30
## transf_mu_threshold      4.29    0.02 0.36     3.62     4.05     4.28     4.50
## transf_mu_ndt            1.46    0.01 0.23     1.00     1.31     1.46     1.61
## transf_mu_theta          0.01    0.00 0.00     0.01     0.01     0.01     0.01
## transf_mu_delta_theta    0.00    0.00 0.00     0.00     0.00     0.00     0.00
## sd_threshold             1.44    0.01 0.30     0.97     1.22     1.40     1.62
## sd_alpha                 0.17    0.00 0.11     0.01     0.08     0.15     0.23
## sd_ndt                   1.29    0.01 0.23     0.93     1.13     1.27     1.43
## sd_theta                 0.61    0.01 0.18     0.35     0.49     0.59     0.71
## sd_delta_theta           0.00    0.00 0.00     0.00     0.00     0.00     0.00
## lp__                 -4001.05    0.43 9.28 -4020.21 -4006.90 -4000.96 -3994.85
##                         97.5% n_eff Rhat
## transf_mu_alpha         -0.14  1206 1.00
## transf_mu_threshold      5.07   293 1.01
## transf_mu_ndt            1.90   269 1.00
## transf_mu_theta          0.02   553 1.01
## transf_mu_delta_theta    0.00  1961 1.00
## sd_threshold             2.10   506 1.01
## sd_alpha                 0.43   716 1.01
## sd_ndt                   1.82   528 1.01
## sd_theta                 1.04   593 1.01
## sd_delta_theta           0.00   719 1.00
## lp__                 -3983.38   458 1.00
##
## Samples were drawn using NUTS(diag_e) at Wed Nov 22 13:04:07 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
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ─
─
## ✔ forcats   1.0.0     ✔ stringr   1.5.0
## ✔ lubridate 1.9.2     ✔ tibble    3.1.8
## ✔ purrr     1.0.1     ✔ tidyr     1.3.0
## ✔ readr     2.1.4
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ─
─
## ✖ tidyr::extract() masks rstan::extract()
## ✖ dplyr::filter()  masks stats::filter()
## ✖ dplyr::lag()     masks stats::lag()
## ℹ Use the ]8;;http://conflicted.r-lib.org/conflicted package]8;; to force a
ll conflicts to become errors
```

```
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)
```

```
## Warning: Using an external vector in selections was deprecated in tidyselect 1.
1.0.
## ℹ Please use `all_of()` or `any_of()` instead.
##   # Was:
##   data %>% select(parameters)
##
##   # Now:
##   data %>% select(all_of(parameters))
##
## See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
# 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")
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## ℹ Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
print(p)
```

```
## Warning: The `guide` argument in `scale_*()` cannot be `FALSE`. This was deprec
ated in
## ggplot2 3.3.4.
## ℹ Please use "none" instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```