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

```
## 载入需要的程辑包：StanHeaders
```

```
##
## rstan version 2.26.23 (Stan version 2.26.1)
```

```
## 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)
## For within-chain threading using `reduce_sum()` or `map_rect()` Stan functions,
## change `threads_per_chain` option:
## rstan_options(threads_per_chain = 1)
```

```
## Do not specify '-march=native' in 'LOCAL_CPPFLAGS' or a Makevars file
```

```
options(mc.cores = parallel::detectCores())
rstan_options(auto_write = T)


# Get list of files in 'data_2' folder with the pattern "riskytimed"
files <- dir(path = "data_2", pattern="riskytimed")

# Read all csv files in the list
data_list <- lapply(paste0("data_2/", files), read.table, header = TRUE, skip = 0,
fill = TRUE, sep= ";")

# Concatenate rows of all items in the list into a data frame
dat <- do.call("rbind", data_list)
```

```
# gamble characteristics

dat$eva = dat$oa1*dat$pa1+dat$oa2*dat$pa2 + dat$oa3*dat$pa3+dat$oa4*dat$pa4
dat$evb = dat$ob1*dat$pb1+dat$ob2*dat$pb2 + dat$ob3*dat$pb3+dat$ob4*dat$pb4
dat$evd = dat$evb - dat$eva

dat$sda = sqrt((dat$oa1-dat$eva)^2*dat$pa1 + (dat$oa2-dat$eva)^2*dat$pa2 + (dat$oa
3-dat$eva)^2*dat$pa3 + (dat$oa4-dat$eva)^2*dat$pa4)
dat$sdb = sqrt((dat$ob1-dat$evb)^2*dat$pb1 + (dat$ob2-dat$evb)^2*dat$pb2 + (dat$ob
3-dat$evb)^2*dat$pb3 + (dat$ob4-dat$evb)^2*dat$pb4)
dat$sdd = dat$sdb - dat$sda


dat$evdummy = ifelse(dat$evd>0,1,0)
```

```
# transform to +/- 1; safe - 1, risky +1
dat$cho <- ifelse(dat$choice==0,-1,ifelse(dat$choice==1,1,NA))
ids <- unique(dat$id)
for(j in 1:length(ids)){
  dat$tid[dat$id==ids[j]] <- j
}
tids <- unique(dat$tid)
# only control data
control_dat <- dat[dat$cond=="control",]
# remove fast RTs
rcontrol_dat <- control_dat[control_dat$rt>1,]
```

```
library(dplyr)
```

```
##
## 载入程辑包：'dplyr'
```

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

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

```
rcontrol_dat <- rcontrol_dat %>%
  rowwise() %>%
  mutate(
    oa_condition = sum(c_across(starts_with("oa")) == 0),
    ob_condition = sum(c_across(starts_with("ob")) == 0)
  ) %>%
  filter(
    (oa_condition == 2 & ob_condition == 0) |
    (oa_condition == 0 & ob_condition == 2)
  ) %>%
  mutate(
    oa_complex = ifelse(oa_condition == 2, -1, 1),
    evd = evd * (-oa_complex),
    sdd = sdd * (-oa_complex),
    chose_complex = ifelse((oa_complex == 1 & choice == 0) | (oa_complex == -1 & c
hoice == 1), 1, -1)
  )
rcontrol_dat
```

```
## # A tibble: 5,988 × 44
## # Rowwise:
##    date       id block order cond  trial gameid leftright   oa1   oa2   oa3   oa
4
##    <chr>  <int> <int> <int> <chr> <int>  <int>     <int> <int> <int> <int> <int
>
##  1 2019_…     1     1     0 cont…     2     58         0    37    58    76     3
4
##  2 2019_…     1     1     0 cont…     4    100         1    27    14    59     5
2
##  3 2019_…     1     1     0 cont…     6     11         0    17    36     0
0
##  4 2019_…     1     1     0 cont…     7     92         1    90    86    65     9
4
##  5 2019_…     1     1     0 cont…    11     37         0    14    98     0
0
##  6 2019_…     1     1     0 cont…    12     89         0    26    41    13     5
6
##  7 2019_…     1     1     0 cont…    13     17         1    81    52     0
0
##  8 2019_…     1     1     0 cont…    14     74         1    66    47    52     4
4
##  9 2019_…     1     1     0 cont…    15     19         1    22    53     0
0
## 10 2019_…     1     1     0 cont…    16      5         1    70     5     0
0
## # ℹ 5,978 more rows
## # ℹ 32 more variables: pa1 <dbl>, pa2 <dbl>, pa3 <dbl>, pa4 <dbl>, ob1 <int>,
## #   ob2 <int>, ob3 <int>, ob4 <int>, pb1 <dbl>, pb2 <dbl>, pb3 <dbl>,
## #   pb4 <dbl>, choice <int>, rt <dbl>, paytrial <int>, payout <int>,
## #   alter <int>, geschlecht <int>, comment <chr>, eva <dbl>, evb <dbl>,
## #   evd <dbl>, sda <dbl>, sdb <dbl>, sdd <dbl>, evdummy <dbl>, cho <dbl>,
## #   tid <int>, oa_condition <int>, ob_condition <int>, oa_complex <dbl>, …
```

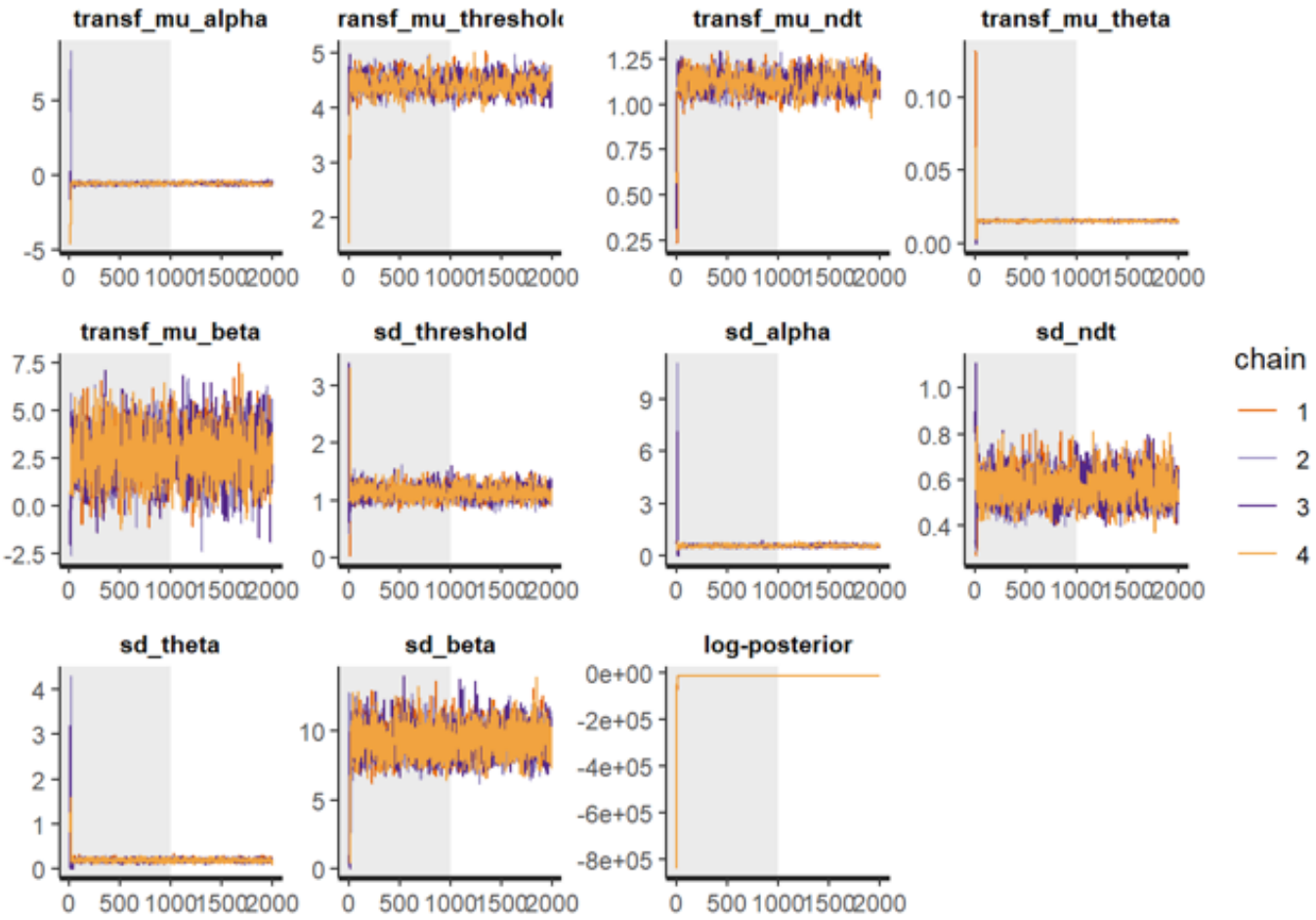
```
dataList  = list(cho = rcontrol_dat$chose_complex, rt = rcontrol_dat$rt, participa
nt = rcontrol_dat$tid,N=nrow(rcontrol_dat),  L = length(tids),starting_point=0.5,
evd = rcontrol_dat$evd, sdd = rcontrol_dat$sdd)
```

```
parameters = c("transf_mu_alpha","transf_mu_threshold","transf_mu_ndt", "transf_mu
_theta",'transf_mu_beta', 'sd_threshold',"sd_alpha","sd_ndt", 'sd_theta', 'sd_beta
', "alpha_sbj","threshold_sbj","ndt_sbj",'theta_sbj', 'beta_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_beta = runif(1,-1, 1),
                          sd_beta = 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_beta = runif(length(tids),-0.1,0.1)


    )
  }

  return(initList)
}
```

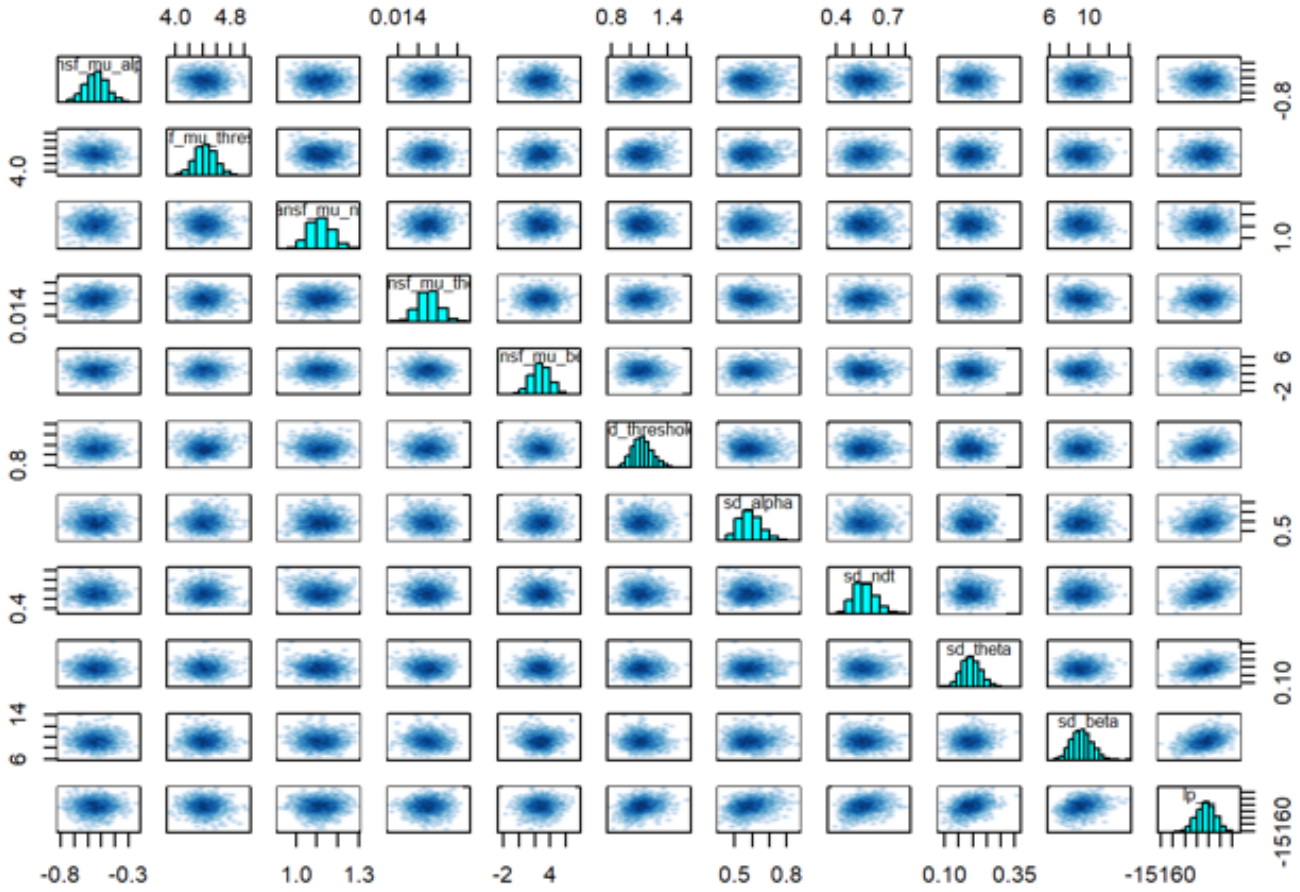
```
m <- stan_model("MV_discount.stan")
dsamples <- sampling(m,
                data=dataList,
                pars=parameters,
                iter=2000,
                chains=4,#If not specified, gives random inits
                init = initFunc(4),
                warmup = 1000,  # Stands for burn-in; Default = iter/2
                seed = 12,  # Setting seed; Default is random seed
                refresh = 0
                )
```

```
## Warning: Bulk Effective Samples Size (ESS) is too low, indicating posterior mea
ns and medians may be unreliable.
## Running the chains for more iterations may help. See
## https://mc-stan.org/misc/warnings.html#bulk-ess
```

```
#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_sbj","ndt_sbj",'theta_sbj',"log_lik")

rstan::traceplot(dsamples, pars=c("transf_mu_alpha","transf_mu_threshold","transf_mu_ndt", "transf_mu_theta",'transf_mu_beta', 'sd_threshold',"sd_alpha","sd_ndt", 'sd_theta', 'sd_beta', "lp__"), inc_warmup = TRUE, nrow = 3)
```




```
pairs(dsamples, pars = c( "transf_mu_alpha","transf_mu_threshold","transf_mu_ndt", "transf_mu_theta",'transf_mu_beta', 'sd_threshold',"sd_alpha","sd_ndt", 'sd_theta', 'sd_beta', "lp__"))
```




```
print(dsamples, pars = c("transf_mu_alpha","transf_mu_threshold","transf_mu_ndt",
"transf_mu_theta",'transf_mu_beta', 'sd_threshold',"sd_alpha","sd_ndt", 'sd_thet
a', 'sd_beta', "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         -0.54    0.00  0.08     -0.70     -0.60     -0.54
## transf_mu_threshold      4.44    0.01  0.15      4.14      4.34      4.43
## transf_mu_ndt            1.12    0.00  0.05      1.02      1.08      1.12
## transf_mu_theta          0.02    0.00  0.00      0.01      0.02      0.02
## transf_mu_beta           2.76    0.05  1.23      0.34      1.94      2.76
## sd_threshold             1.13    0.00  0.11      0.93      1.05      1.12
## sd_alpha                 0.59    0.00  0.06      0.48      0.54      0.59
## sd_ndt                   0.56    0.00  0.06      0.45      0.52      0.56
## sd_theta                 0.19    0.00  0.03      0.13      0.17      0.19
## sd_beta                  9.25    0.03  1.03      7.40      8.53      9.21
## lp__                -15084.97    0.65 17.99 -15124.10 -15096.43 -15084.07
##                           75%     97.5% n_eff Rhat
## transf_mu_alpha         -0.49     -0.38   361 1.00
## transf_mu_threshold      4.54      4.74   378 1.01
## transf_mu_ndt            1.15      1.22   359 1.01
## transf_mu_theta          0.02      0.02  1949 1.00
## transf_mu_beta           3.59      5.17   600 1.00
## sd_threshold             1.20      1.36   621 1.02
## sd_alpha                 0.63      0.73   525 1.01
## sd_ndt                   0.60      0.70   956 1.00
## sd_theta                 0.22      0.26  1252 1.00
## sd_beta                  9.92     11.39  1172 1.00
## lp__                -15072.92 -15050.82   755 1.00
##
## Samples were drawn using NUTS(diag_e) at Wed Nov  1 23:11:25 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(bayesplot)
```

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

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

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

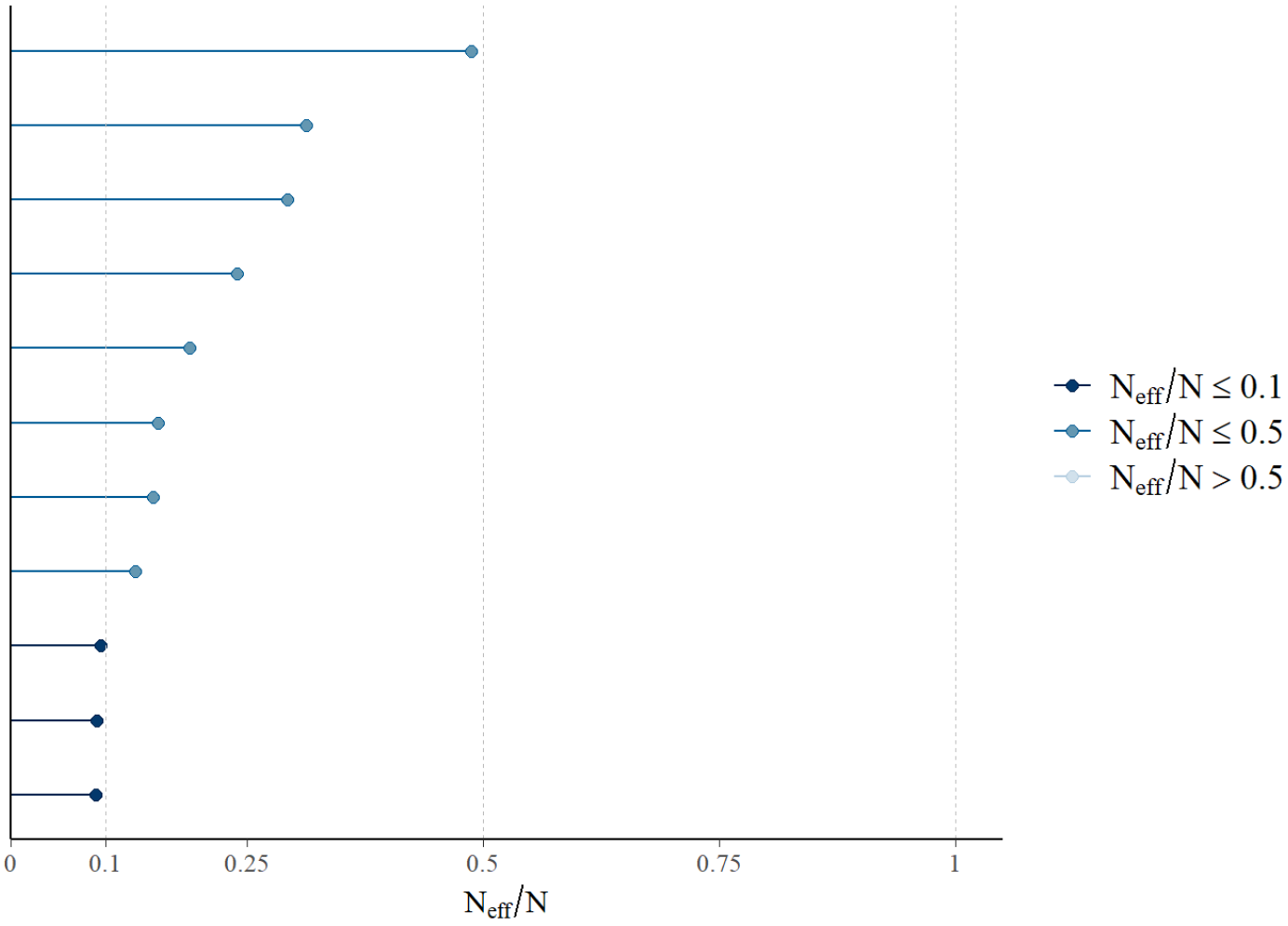
```
##    * Does _not_ affect other ggplot2 plots
```

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

```
ratios_cp <- neff_ratio(dsamples, pars = c("transf_mu_alpha","transf_mu_threshol
d","transf_mu_ndt", "transf_mu_theta",'transf_mu_beta', 'sd_threshold',"sd_alph
a","sd_ndt", 'sd_theta', 'sd_beta',"lp__"))
df_ratios_cp <- as.data.frame(ratios_cp)
print(df_ratios_cp)
```

```
##                       ratios_cp
## transf_mu_alpha      0.09014932
## transf_mu_threshold 0.09438631
## transf_mu_ndt        0.08972369
## transf_mu_theta      0.48730363
## transf_mu_beta       0.15007940
## sd_threshold         0.15520221
## sd_alpha             0.13123878
## sd_ndt               0.23903841
## sd_theta             0.31304596
## sd_beta              0.29293876
## lp__                 0.18886755
```

```
mcmc_neff(ratios_cp, size = 2)
```



```
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_beta')

# 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)
```