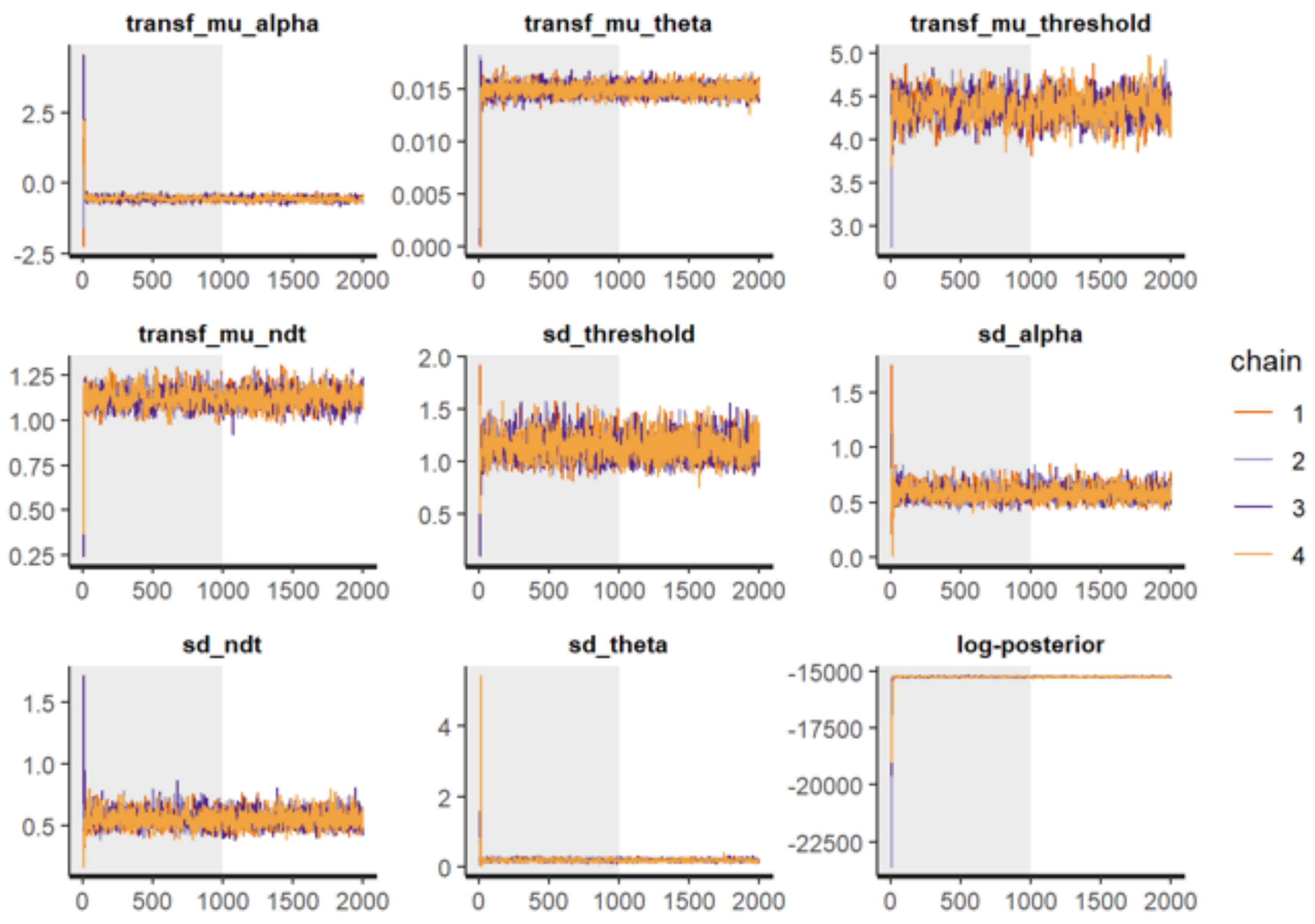
```
# gamble characteristics
dat$eva = dat$oa1*dat$pa1+dat$oa2*dat$pa2 + dat$oa3*dat$pa3+dat$oa4*dat$pa4
dat$evb = dat$ob1*dat$pb1+dat$ob2*dat$pb2 + dat$ob3*dat$pb3+dat$ob4*dat$pb4
dat$evd = dat$evb - dat$eva
dat$sda = sqrt((dat$oa1-dat$eva)^2*dat$pa1 + (dat$oa2-dat$eva)^2*dat$pa2 + (dat$oa
3-dat$eva)^2*dat$pa3 + (dat$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)


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)
  )
```

```
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)
```

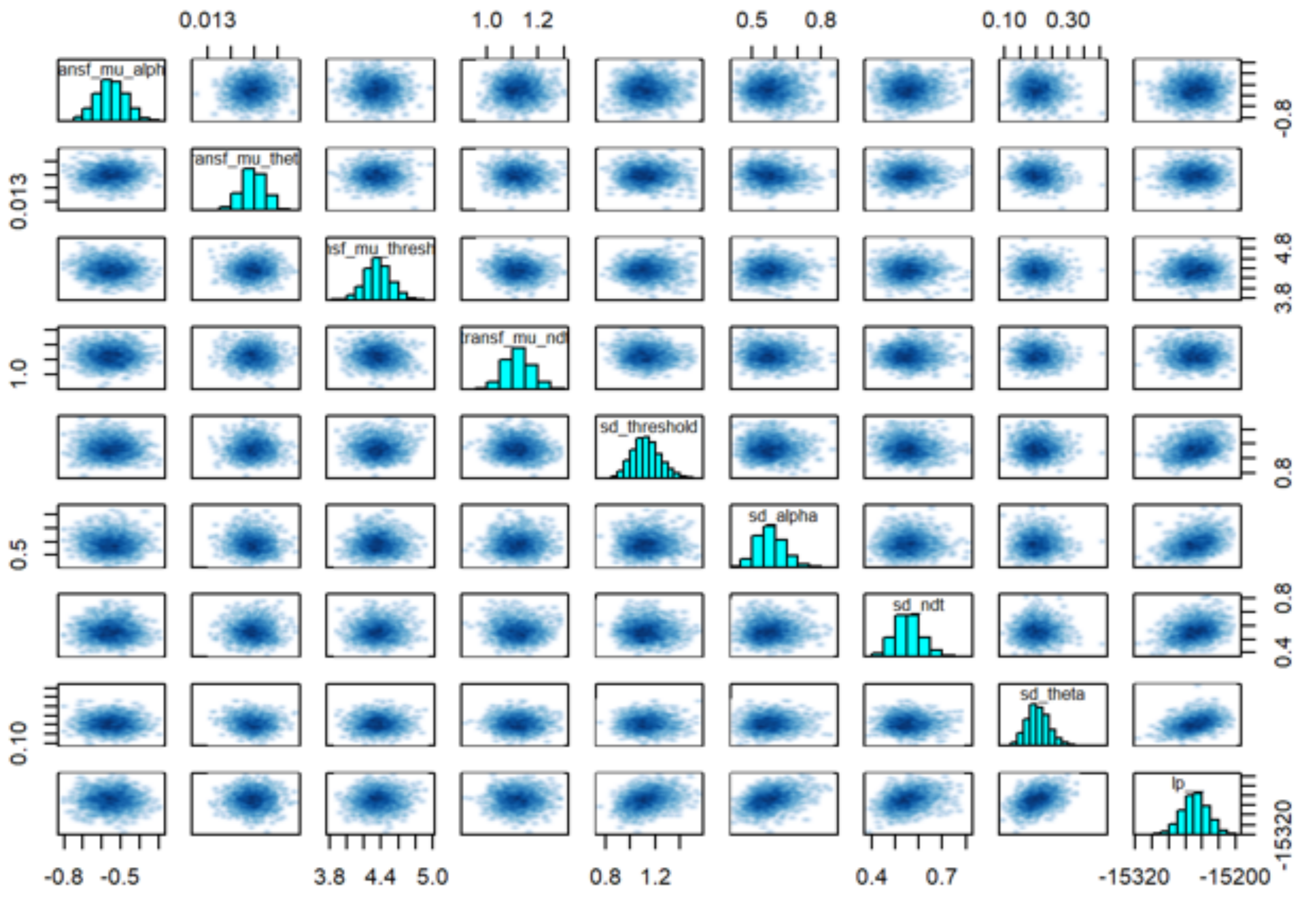
```r
parameters = c("transf_mu_alpha","transf_mu_threshold","transf_mu_ndt", "transf_mu_theta", 'sd_threshold',"sd_alpha","sd_ndt", 'sd_theta',  "alpha_sbj","threshold_sbj","ndt_sbj",'theta_sbj', "log_lik")

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)
}
```

```r
m <- stan_model("MV_Baseline ce.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
                )
```

```r
#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_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__"))
```



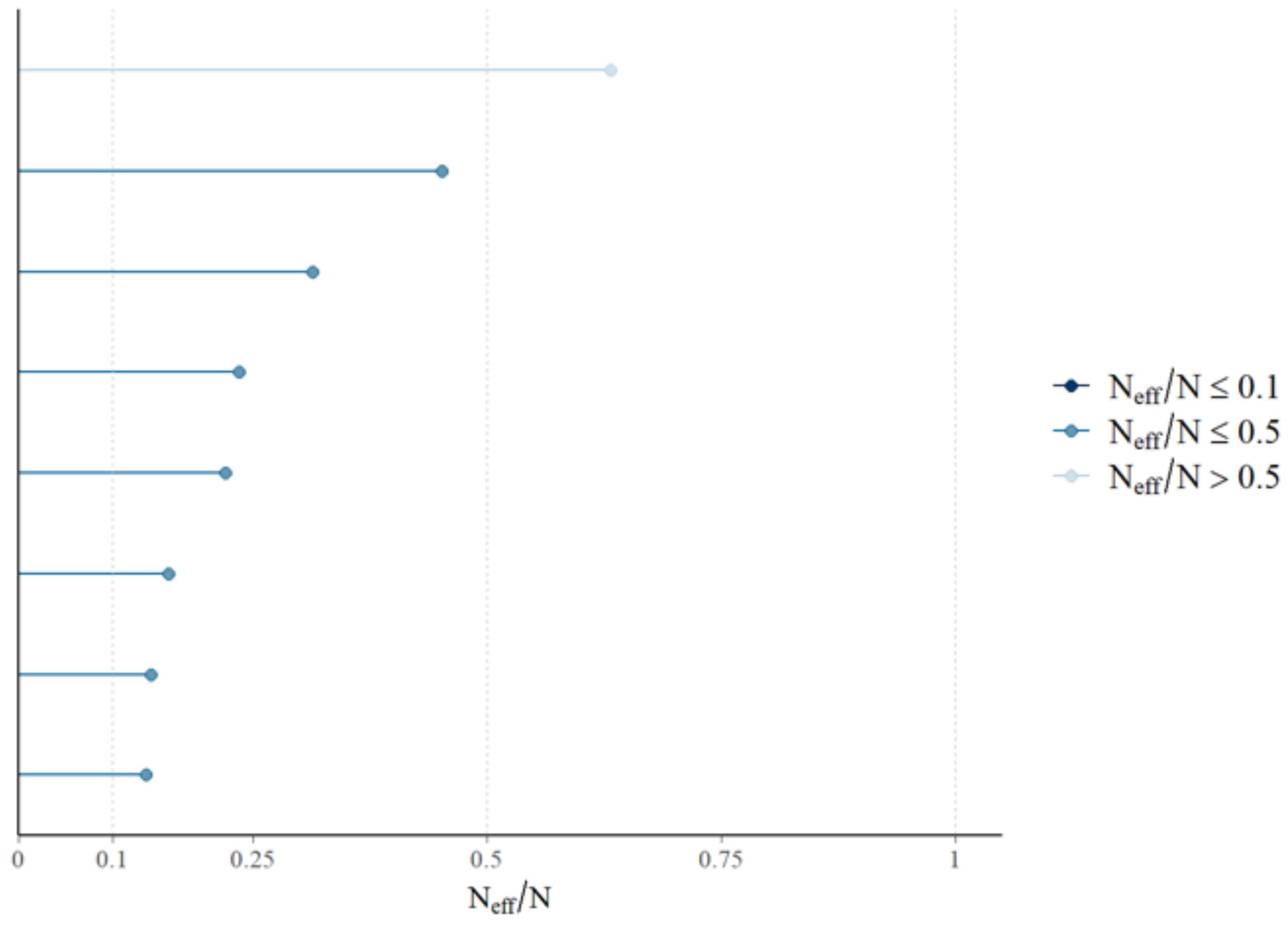
```
print(dsamples, pars = c("transf_mu_alpha", "transf_mu_theta", "transf_mu_threshol
d","transf_mu_ndt", 'sd_threshold',"sd_alpha","sd_ndt", 'sd_theta', "lp__"))
```

```
## Inference for Stan model: 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        -0.55    0.00  0.08     -0.70     -0.60     -0.55
## transf_mu_theta         0.01    0.00  0.00      0.01      0.01      0.01
## transf_mu_threshold     4.36    0.01  0.15      4.08      4.27      4.36
## transf_mu_ndt           1.12    0.00  0.05      1.03      1.09      1.12
## sd_threshold            1.13    0.00  0.11      0.93      1.05      1.12
## sd_alpha                0.58    0.00  0.06      0.47      0.54      0.58
## sd_ndt                  0.56    0.00  0.06      0.45      0.52      0.55
## sd_theta                0.20    0.00  0.04      0.14      0.18      0.20
## lp__               -15248.69    0.60 16.22 -15281.59 -15259.43 -15248.59
##                          75%     97.5% n_eff Rhat
## transf_mu_alpha        -0.50     -0.40   639 1.01
## transf_mu_theta         0.02      0.02  2526 1.00
## transf_mu_threshold     4.46      4.66   541 1.01
## transf_mu_ndt           1.16      1.23   564 1.01
## sd_threshold            1.20      1.37   939 1.00
## sd_alpha                0.62      0.71   883 1.00
## sd_ndt                  0.59      0.69  1256 1.00
## sd_theta                0.23      0.28  1806 1.00
## lp__               -15237.53 -15218.01   734 1.00
##
## Samples were drawn using NUTS(diag_e) at Wed Nov  1 15:13:34 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).
```

```
parameters = c("transf_mu_alpha","transf_mu_theta", "transf_mu_threshold","transf_
mu_ndt", 'sd_threshold',"sd_alpha","sd_ndt", 'sd_theta')
ratios_cp <- neff_ratio(dsamples, pars = c("transf_mu_alpha","transf_mu_theta", "t
ransf_mu_threshold","transf_mu_ndt", 'sd_threshold',"sd_alpha","sd_ndt", 'sd_thet
a'))
df_ratios_cp <- as.data.frame(ratios_cp)
print(df_ratios_cp)
```

```
##                     ratios_cp
## transf_mu_alpha     0.1597721
## transf_mu_theta     0.6314082
## transf_mu_threshold 0.1353355
## transf_mu_ndt       0.1410209
## sd_threshold        0.2346502
## sd_alpha            0.2208060
## sd_ndt              0.3140410
## sd_theta            0.4514514
```

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



```
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 1.9.3     ✔ tibble    3.2.1
## ✔ purrr     1.0.2     ✔ 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 conflicted package (<http://conflicted.r-lib.org/>) to force all conf
licts to become errors
```

```r
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)
```

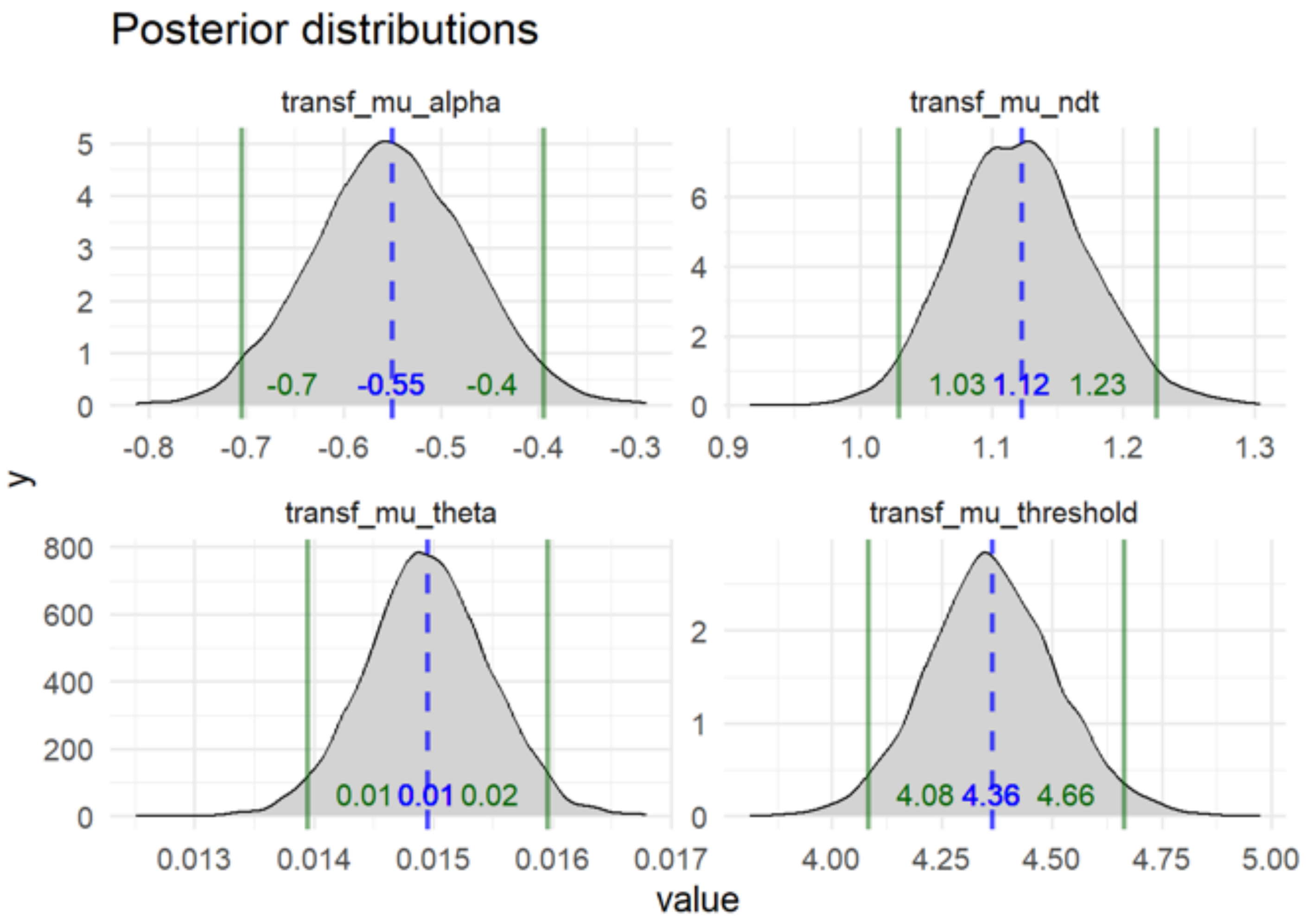
```
## Warning: Using an external vector in selections was deprecated in tidyselect 1.
1.0.
## ℹ Please use `all_of()` or `any_of()` instead.
##   # Was:
##   data %>% select(parameters)
##
##   # Now:
##   data %>% select(all_of(parameters))
##
## See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## 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.
```


Posterior distributions
transf_mu_alpha
transf_mu_ndt
transf_mu_theta
transf_mu_threshold
y
value