

```
##### 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: 2elf913d3ca3)
```

```
## 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)
```

```
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$choice==1,1,NA))
dat$cho2 <- ifelse(dat$choice==0,1,ifelse(dat$choice==1,0,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,]
# only condition no time pressure
dataList = list(cho = rcontrol_dat$cho, rt = rcontrol_dat$rt, participant = 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")
```

```
## hash mismatch so recompiling; make sure Stan code ends with a blank line
```

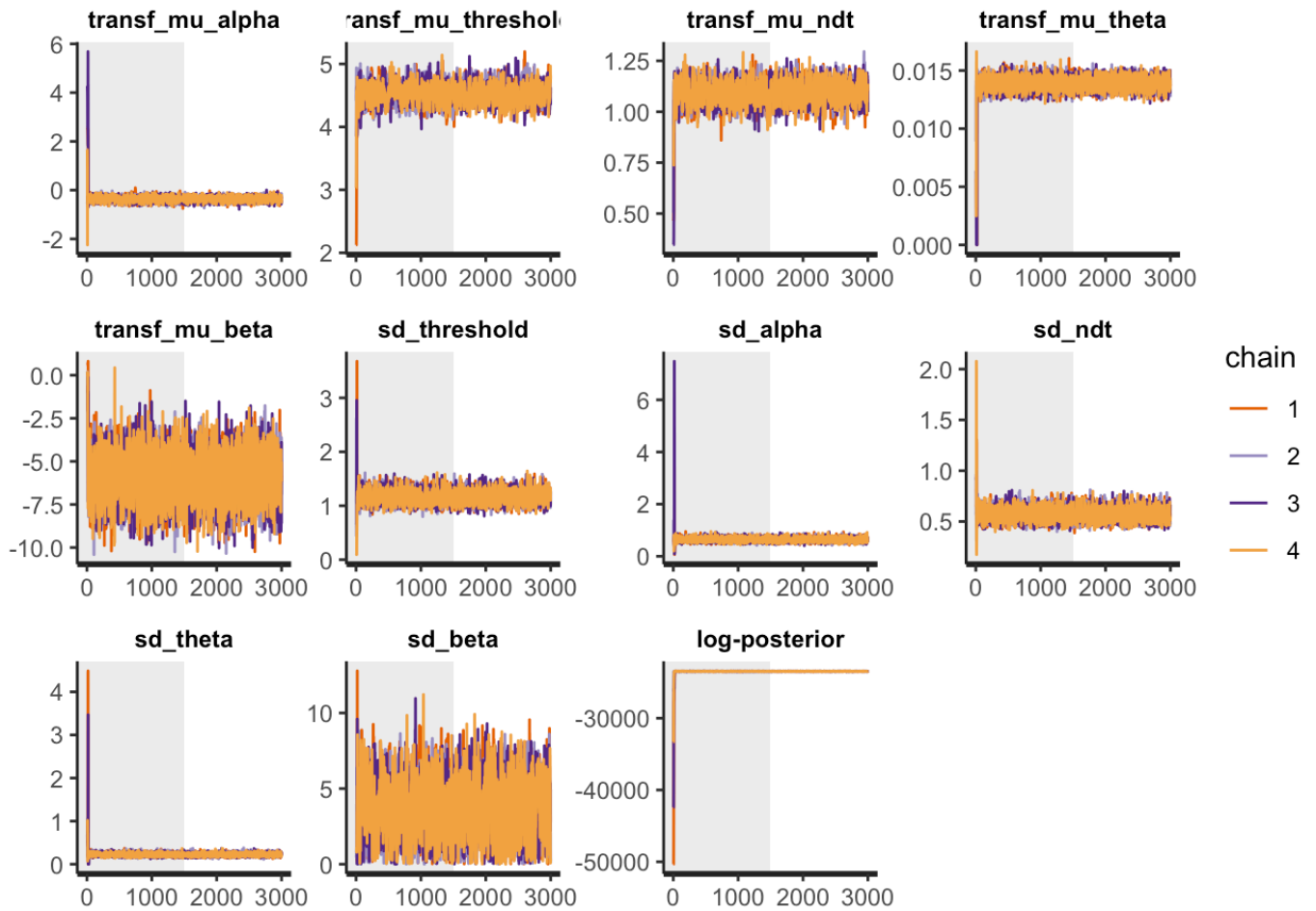
```
dsamples <- sampling(m,
  data=dataList,
  pars=parameters,
  iter=3000,
  chains=4,#If not specified, gives random inits
  init = initFunc(4),
  warmup = 1500, # Stands for burn-in; Default = iter/2
  seed = 12, # Setting seed; Default is random seed
  refresh = 0
)
```

```
## Warning: There were 1 divergent transitions after warmup. See
## https://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
## to find out why this is a problem and how to eliminate them.
```

```
## Warning: Examine the pairs() plot to diagnose sampling problems
```

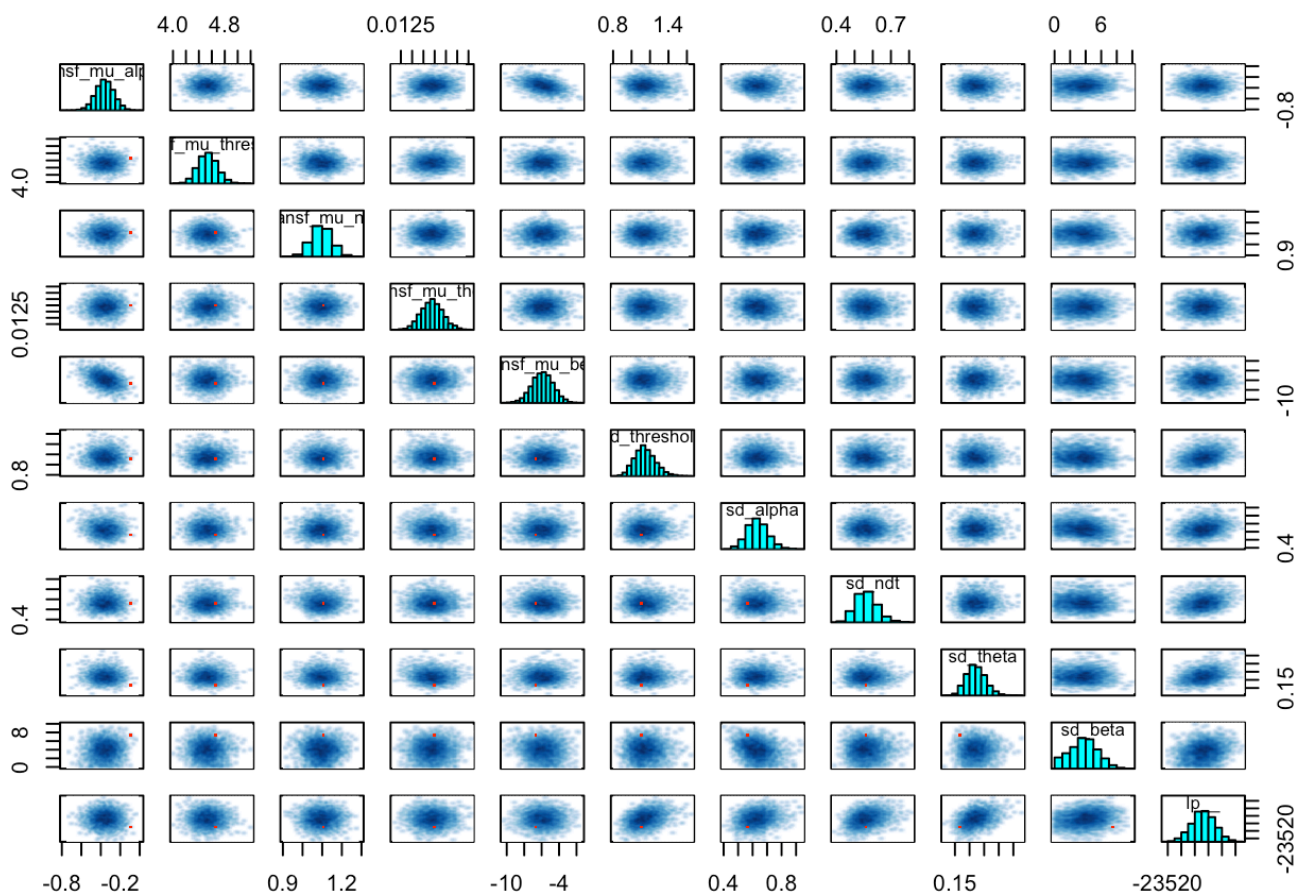
```
#parameters = c("transf_mu_alpha","transf_mu_threshold","transf_mu_ndt", "transf_mu_theta", 'sd_threshold',"sd_alpha","sd_ndt", 'sd_theta', "alpha_sbj","threshold_sbj","ndt_sbj",'theta_sbj',"log_lik")
```

```
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__"))
```

```
## Warning in par(usr): argument 1 does not name a graphical parameter
## Warning in par(usr): argument 1 does not name a graphical parameter
## Warning in par(usr): argument 1 does not name a graphical parameter
## Warning in par(usr): argument 1 does not name a graphical parameter
## Warning in par(usr): argument 1 does not name a graphical parameter
## Warning in par(usr): argument 1 does not name a graphical parameter
## Warning in par(usr): argument 1 does not name a graphical parameter
## Warning in par(usr): argument 1 does not name a graphical parameter
## Warning in par(usr): argument 1 does not name a graphical parameter
## Warning in par(usr): argument 1 does not name a graphical parameter
```



```
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: MV_discount.
## 4 chains, each with iter=3000; warmup=1500; thin=1;
## post-warmup draws per chain=1500, total post-warmup draws=6000.
##
##
```

	mean	se_mean	sd	2.5%	25%	50%
transf_mu_alpha	-0.36	0.00	0.09	-0.55	-0.42	-0.36
transf_mu_threshold	4.54	0.01	0.15	4.26	4.43	4.53
transf_mu_ndt	1.10	0.00	0.05	1.00	1.06	1.10
transf_mu_theta	0.01	0.00	0.00	0.01	0.01	0.01
transf_mu_beta	-5.90	0.01	1.24	-8.36	-6.72	-5.89
sd_threshold	1.15	0.00	0.11	0.95	1.07	1.14
sd_alpha	0.63	0.00	0.07	0.50	0.58	0.63
sd_ndt	0.57	0.00	0.06	0.46	0.52	0.56
sd_theta	0.22	0.00	0.03	0.16	0.20	0.22
sd_beta	3.72	0.06	1.79	0.36	2.45	3.77
lp__	-23467.64	0.49	17.11	-23501.54	-23478.88	-23467.81

```
##
```

	75%	97.5%	n_eff	Rhat
transf_mu_alpha	-0.30	-0.18	1374	1.00
transf_mu_threshold	4.63	4.84	436	1.02
transf_mu_ndt	1.13	1.19	799	1.01
transf_mu_theta	0.01	0.01	3175	1.00
transf_mu_beta	-5.07	-3.47	7616	1.00
sd_threshold	1.22	1.39	1350	1.00
sd_alpha	0.68	0.79	1698	1.00
sd_ndt	0.60	0.69	1614	1.00
sd_theta	0.24	0.29	2507	1.00
sd_beta	4.95	7.22	813	1.00
lp__	-23455.86	-23434.20	1200	1.00

```
##
## Samples were drawn using NUTS(diag_e) at Wed Oct 25 15:08:05 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()
```

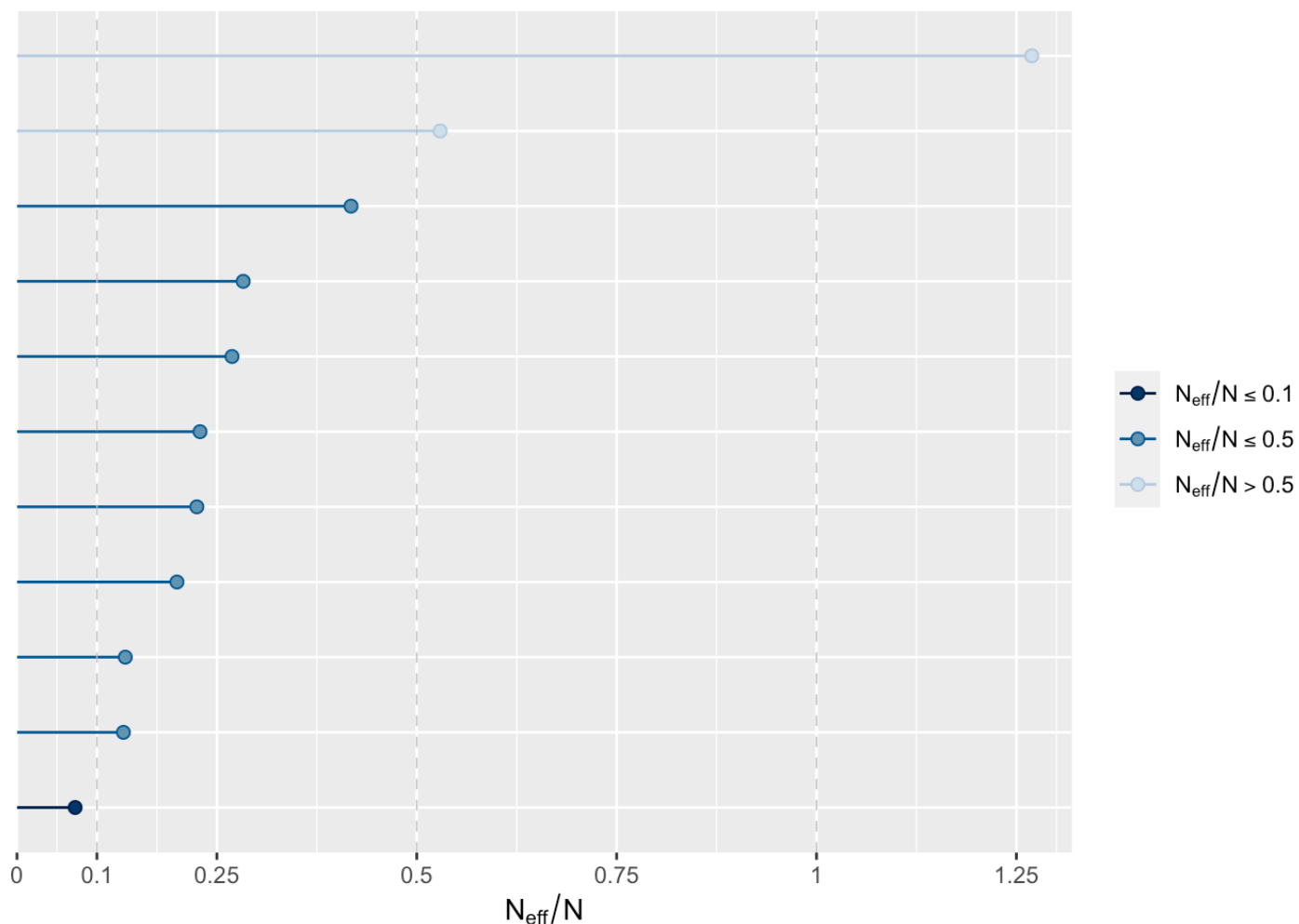
```
##      * Does _not_ affect other ggplot2 plots
```

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

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

```
##              ratios_cp
## transf_mu_alpha      0.22893685
## transf_mu_threshold 0.07271721
## transf_mu_ndt        0.13309415
## transf_mu_theta      0.52921184
## transf_mu_beta       1.26928188
## sd_threshold         0.22492990
## sd_alpha             0.28296654
## sd_ndt               0.26894733
## sd_theta             0.41782191
## sd_beta              0.13549497
## lp__                 0.20007998
```

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



```
library(ggplot2)
library(tidyverse) # for the gather function
```

```
## — Attaching core tidyverse packages ————— tidyverse 2.0.0 —
## ✓ dplyr      1.1.0      ✓ readr      2.1.4
## ✓ forcats    1.0.0      ✓ stringr    1.5.0
## ✓ lubridate  1.9.2      ✓ tibble     3.1.8
## ✓ purrr      1.0.1      ✓ tidyr      1.3.0
## — Conflicts ————— tidyverse_conflicts() —
## ✖ tidyr::extract() masks rstan::extract()
## ✖ dplyr::filter()  masks stats::filter()
## ✖ dplyr::lag()     masks stats::lag()
## i Use the `conflicted::http://conflicted.r-lib.org/` package to force all 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_threshold","transf_mu_ndt", "transf_mu_theta", 'transf_mu_beta')

# 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", size = 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")

```

```

## 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 deprecated 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

