Assignment 1 - Language development in autistic and neurotypical children

Assignment 1 - Language development in autistic and neurotypical children

Quick recap

Autism Spectrum Disorder is often related to language impairment. However, this phenomenon has rarely been empirically traced in detail: i) relying on actual naturalistic language production, ii) over extended periods of time.

We therefore videotaped circa 30 kids with ASD and circa 30 comparison kids (matched by linguistic performance at visit 1) for ca. 30 minutes of naturalistic interactions with a parent. We repeated the data collection 6 times per kid, with 4 months between each visit. We transcribed the data and counted: i) the amount of words that each kid uses in each video. Same for the parent. ii) the amount of unique words that each kid uses in each video. Same for the parent. iii) the amount of morphemes per utterance (Mean Length of Utterance) displayed by each child in each video. Same for the

parent.

This data is in the file you prepared in the previous class, but you can also find it here:https://www.dropbox.com/s/d6eerv6cl6eksf3/data_clean.csv?dl=0

The structure of the assignment

We will be spending a few weeks with this assignment. In particular, we will:

Part 1) simulate data in order to better understand the model we need to build, and to better understand how much data we would have to collect to run a meaningful study (precision analysis)

Part 2) analyze our empirical data and interpret the inferential results

Part 3) use your model to predict the linguistic trajectory of new children and assess the performance of the model based on that.

As you work through these parts, you will have to produce a written document (separated from the code) answering the following questions:

Q1 - Briefly describe your simulation process, its goals, and what you have learned from the simulation. Add at least a plot showcasing the results of the simulation. Make a special note on sample size considerations: how much data do you think you will need? what else could you do to increase the precision of your estimates?

Q2 - Briefly describe the empirical data and how they compare to what you learned from the simulation (what can you learn from them?). Briefly describe your model(s) and model quality. Report the findings: how does development differ between autistic and

neurotypical children (N.B. remember to report both population and individual level findings)? which additional factors should be included in the model? Add at least one plot showcasing your findings.

Q3 - Given the model(s) from Q2, how well do they predict the data? Discuss both in terms of absolute error in training vs testing; and in terms of characterizing the new kids' language development as typical or in need of support.

Below you can find more detailed instructions for each part of the assignment.

Part 1 - Simulating data

Before we even think of analyzing the data, we should make sure we understand the problem, and we plan the analysis. To do so, we need to simulate data and analyze the simulated data (where we know the ground truth).

In particular, let's imagine we have n autistic and n neurotypical children. We are simulating their average utterance length (Mean Length of Utterance or MLU) in terms of words, starting at Visit 1 and all the way to Visit 6. In other words, we need to define a few parameters: - average MLU for ASD (population mean) at Visit 1 and average individual deviation from that (population standard deviation) - average MLU for TD (population mean) at Visit 1 and average individual deviation from that (population standard deviation) - average change in MLU by visit for ASD (population mean) and average individual deviation from that (population standard deviation) - average change in MLU by visit for TD (population mean) and average individual deviation from that (population standard deviation) - an error term. Errors could be due to measurement, sampling, all sorts of noise.

Note that this makes a few assumptions: population means are exact values; change by visit is linear (the same between visit 1 and 2 as between visit 5 and 6). This is fine for the exercise. In real life research, you might want to vary the parameter values much more, relax those assumptions and assess how these things impact your inference.

We go through the literature and we settle for some values for these parameters: - average MLU for ASD and TD: 1.5 (remember the populations are matched for linguistic ability at first visit) - average individual variability in initial MLU for ASD 0.5; for TD 0.3 (remember ASD tends to be more heterogeneous) - average change in MLU for ASD: 0.4; for TD 0.6 (ASD is supposed to develop less) - average individual variability in change for ASD 0.4; for TD 0.2 (remember ASD tends to be more heterogeneous) - error is identified as 0.2

This would mean that on average the difference between ASD and TD participants is 0 at visit 1, 0.2 at visit 2, 0.4 at visit 3, 0.6 at visit 4, 0.8 at visit 5 and 1 at visit 6.

With these values in mind, simulate data, plot the data (to check everything is alright); and set up an analysis pipeline.

Remember the usual bayesian workflow: - define the formula - define the prior - prior predictive checks - fit the model - model quality checks: traceplots, divergences, rhat, effective samples - model quality checks: posterior predictive checks, prior-posterior update checks - model comparison

Once the pipeline is in place, loop through different sample sizes to assess how much data you would need to collect. N.B. for inspiration on how to set this up, check the tutorials by Kurz that are linked in the syllabus.

BONUS questions for Part 1: what if the difference between ASD

and TD was 0? how big of a sample size would you need? What about different effect sizes, and different error terms?

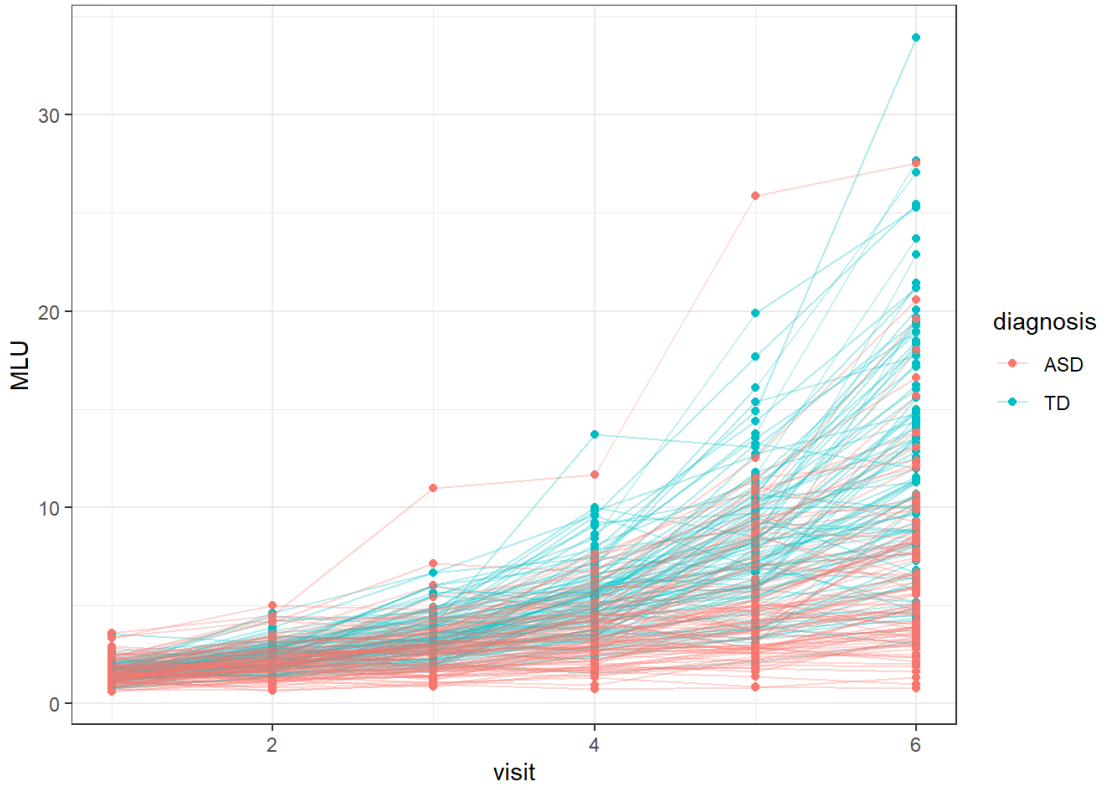
Bryan ### Simulating data To make beta values between each visit we would need the standard diviation between visits

```
average_mlu <- log(1.5)
 sd_mlu_asd <- log(1.5+0.5)-log(1.5)
 sd_mlu_td <- log(1.5+0.3)-log(1.5)

 change_mlu_asd <- 0.4/1.5
 change_mlu_td <- 0.6/1.5
 change_sd_mlu_asd <- 0.4*(0.4/1.5)
 change_sd_mlu_td <- 0.2*(0.6/1.5)
 e <- 0.2

n <- 100
int_asd <- rnorm(n, mean=average_mlu, sd=sd_mlu_asd)
int_td <- rnorm(n, mean=average_mlu, sd=sd_mlu_td)

slope_asd <- rnorm(n, mean=change_mlu_asd,
sd=change_sd_mlu_asd)
slope_td <- rnorm(n, mean = change_mlu_td,
sd=change_sd_mlu_td)
 sim_data <-
   tibble(diagnosis=rep(c('TD', 'ASD'), each=n)) %>%
   mutate(intercept=ifelse(diagnosis=='TD', int_td,
int_asd)) %>%
   mutate(slope=ifelse(diagnosis=='TD', slope_td,
slope_asd)) %>%
   mutate(error=ifelse(diagnosis=='TD', e, e)) %>%
   dplyr::mutate(ID=row_number()) %>%
   slice(rep(1:n(), each=6)) %>%
   add_column(visit=rep(c(1,2,3,4,5,6), times=n+n))

 for(i in seq(nrow(sim_data))){
```

```
   sim_data$MLU[i] <- exp(rnorm(1,
sim_data$intercept[i]+
(sim_data$slope[i]*(sim_data$visit[i]-1)),
sim_data$error[i]))
 }
## Warning: Unknown or uninitialised column: `MLU`.
```

plot simulated data

```
ggplot(sim_data, aes(visit,MLU, color=diagnosis,
group=ID))+
  theme_bw()+
  geom_point()+
  geom_line(alpha=0.3)
```



##Analysing simulated data

###define formula

```
MLU_f1 <- bf(MLU ~ 0 + diagnosis + diagnosis:visit + (1
+ visit|ID))
```

```
lognorm_fam <- brmsfamily('lognormal', bhaz =
list(Boundary.knots=c(-1,31)))
```

###Investigate and set priors

```
get_prior(data = sim_data, family = lognorm_fam,
MLU_f1)
```

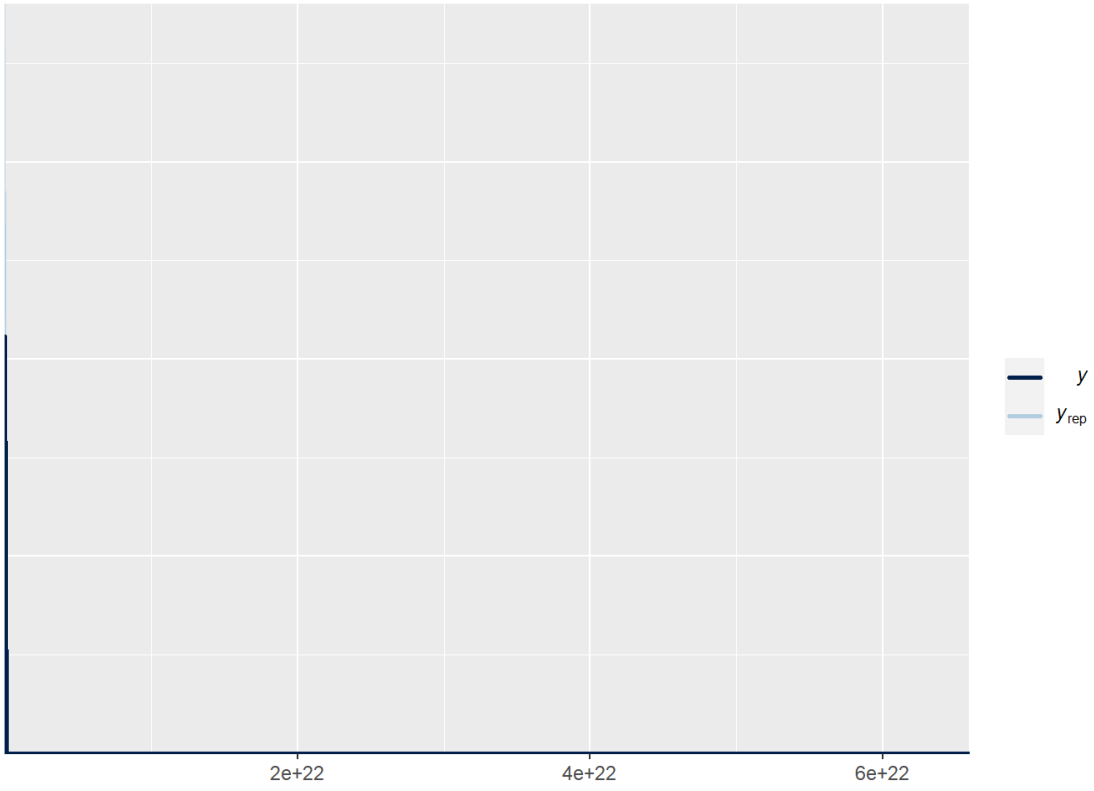
```
priors <- c(
prior(normal(1.5,0.5),class=b,coef="diagnosisASD"),
prior(normal(1.5,0.3),class=b,coef="diagnosisTD"),
prior(normal(0,0.5),class=b),
prior(normal(0,0.5),class=sd),
prior(lkj(2),class=cor))
```

###Model using priors

```
MLU_prior_m1 <- brm(
  MLU_f1,
  data = sim_data,
  prior = priors,
  family = lognorm_fam,
  refresh=0,
  sample_prior = 'only',
  iter=6000,
  warmup = 2500,
  backend = "cmdstanr",
  threads = threading(2),
  chains = 2,
  cores = 2,
  control = list(
    adapt_delta = 0.99,
    max_treedepth = 20
)
)
```

###prior predictive checks

```
pp_check(MLU_prior_m1, ndraws=100)
```



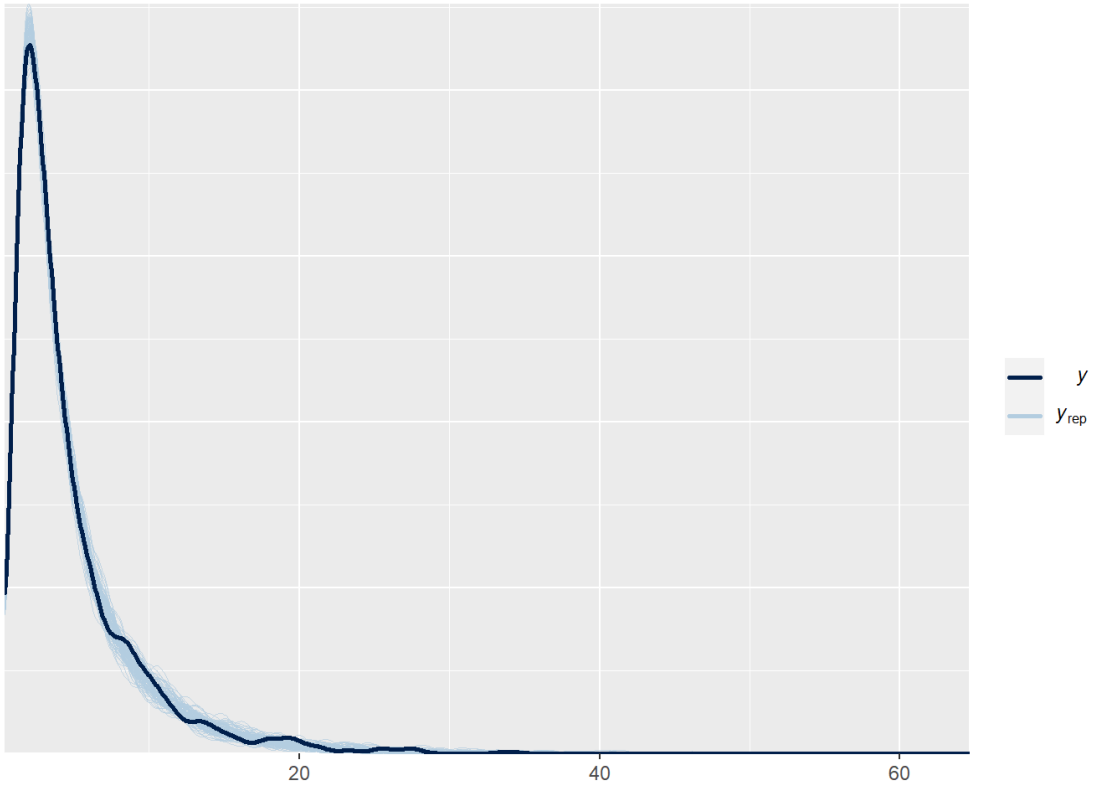
###fit the model

```
MLU_prior_m1_fit <- brm(
  MLU_f1,
  data = sim_data,
  prior = priors,
  family = lognorm_fam,
  refresh=0,
  sample_prior = TRUE,
  iter=6000,
  warmup = 2500,
  backend = "cmdstanr",
  threads = threading(2),
  chains = 2,
  cores = 2,
  control = list(
    adapt_delta = 0.99,
    max_treedepth = 20
```

```
)
)
```

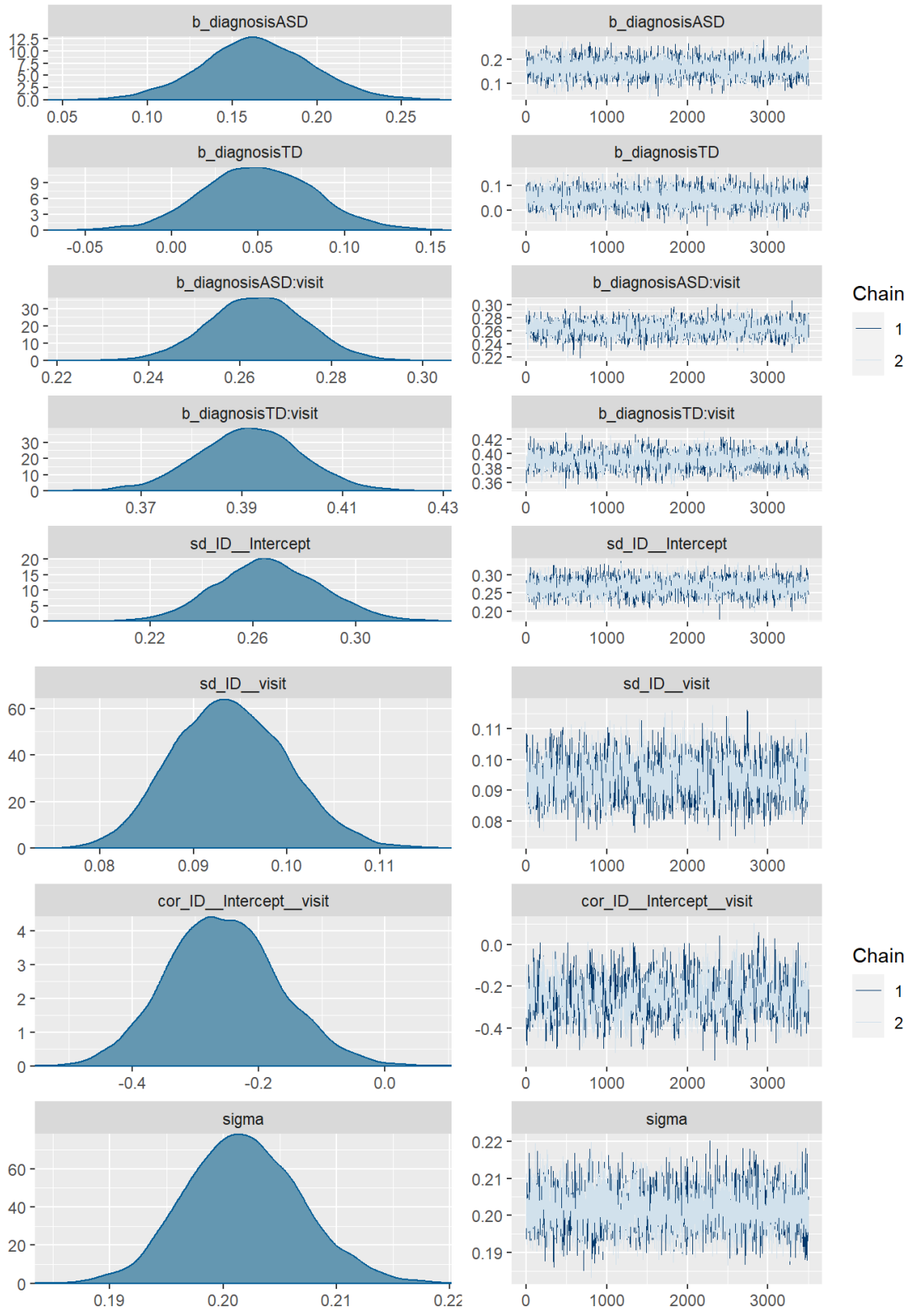
###posterior predictive check

```
pp_check(MLU_prior_m1_fit, ndraws = 100)
```



###traceplot for fitted model

```
plot(MLU_prior_m1_fit)
```



Ditlev ### parameter recovery from fitted model

```
print(MLU_prior_m1_fit)
##  Family: lognormal
##   Links: mu = identity; sigma = identity
## Formula: MLU ~ 0 + diagnosis + diagnosis:visit + (1
+ visit | ID)
##    Data: sim_data (Number of observations: 1200)
##   Draws: 2 chains, each with iter = 6000; warmup =
2500; thin = 1;
##          total post-warmup draws = 7000
##
## Group-Level Effects:
## ~ID (Number of levels: 200)
##                     Estimate Est.Error l-95% CI
u-95% CI Rhat Bulk_ESS
## sd(Intercept)            0.27      0.02     0.23
0.31 1.00     3180
## sd(visit)                0.09      0.01     0.08
0.11 1.00     1420
## cor(Intercept,visit)    -0.26      0.09    -0.42
-0.08 1.00      818
##                     Tail_ESS
## sd(Intercept)           4997
## sd(visit)               2859
## cor(Intercept,visit)    1449
##
## Population-Level Effects:
##                   Estimate Est.Error l-95% CI u-95%
CI Rhat Bulk_ESS Tail_ESS
## diagnosisASD          0.16      0.03     0.10
0.23 1.00     7144     5530
## diagnosisTD           0.05      0.03    -0.01
0.11 1.00     7420     5327
## diagnosisASD:visit    0.26      0.01     0.24
0.28 1.00     3753     3858
## diagnosisTD:visit     0.39      0.01     0.37
```

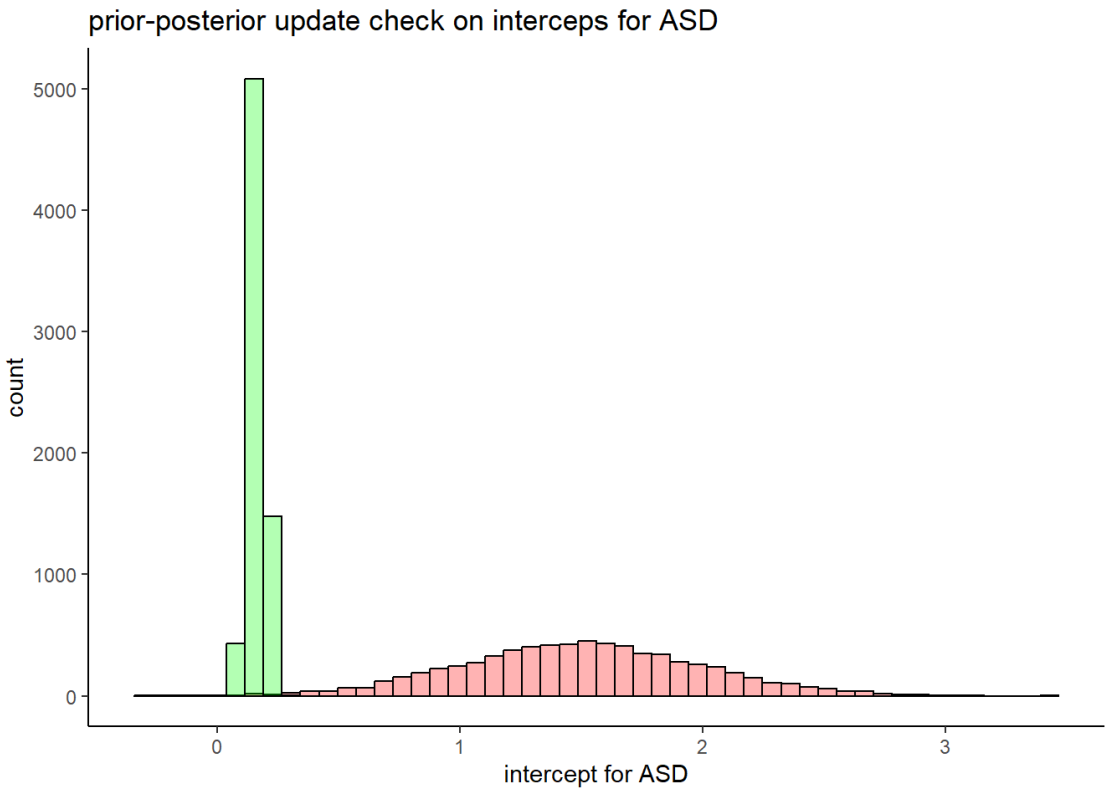
```
0.41 1.00     4086     4820
##
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat
Bulk_ESS Tail_ESS
## sigma     0.20      0.01     0.19     0.21 1.00
4407     4653
##
## Draws were sampled using sample(hmc). For each
parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and
Rhat is the potential
## scale reduction factor on split chains (at
convergence, Rhat = 1).
```

prior posterior update check

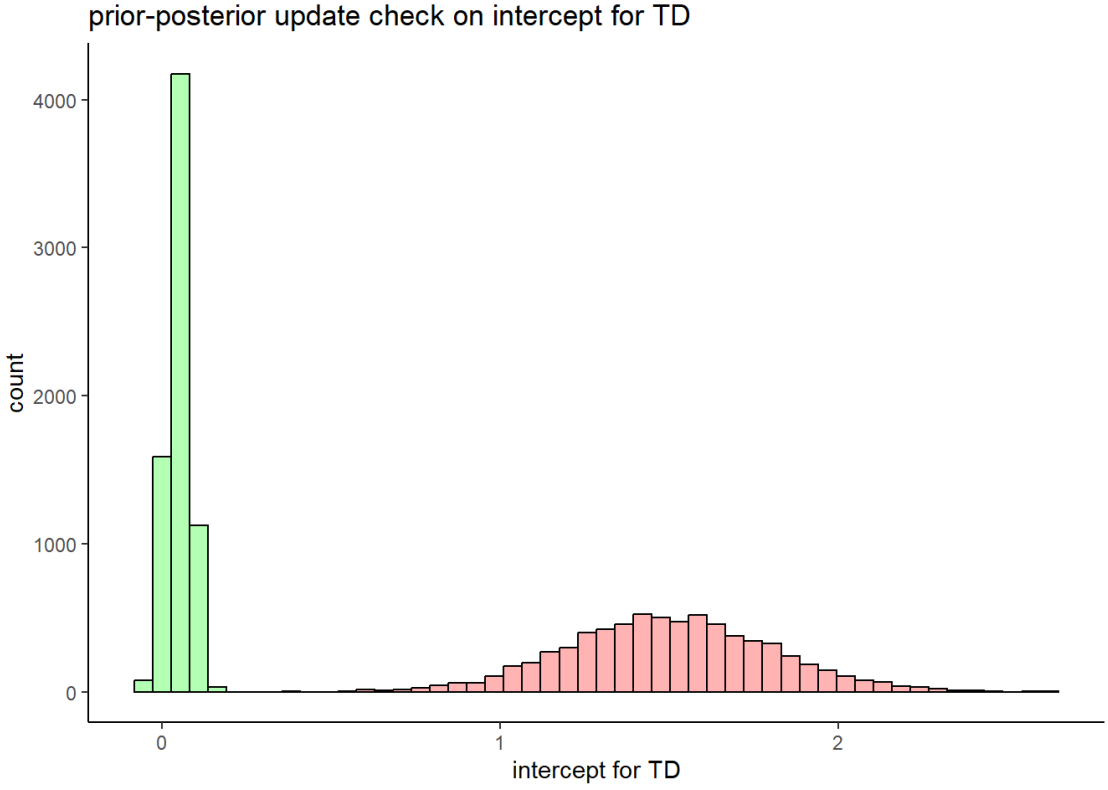
```
posterior <- as_draws_df(MLU_prior_m1_fit)

plot1 <- ggplot(posterior)+
  geom_histogram(aes(prior_b_diagnosisASD), fill='red',
color='black', alpha=0.3, bins=50)+
  geom_histogram(aes(b_diagnosisASD), fill='green',
color='black', alpha=0.3, bins=50)+
  theme_classic()+
  ggtitle('prior-posterior update check on interceps
for ASD')+
  xlab('intercept for ASD')

plot2 <- ggplot(posterior)+
  geom_histogram(aes(prior_b_diagnosisTD), fill='red',
color='black', alpha=0.3, bins=50)+
  geom_histogram(aes(b_diagnosisTD), fill='green',
color='black', alpha=0.3, bins=50)+
  theme_classic()+
  ggtitle('prior-posterior update check on intercept
```

```
for TD')+
  xlab('intercept for TD')

plot3 <- ggplot(posterior)+
  geom_histogram(aes(`prior_b_diagnosisASD:visit`),
fill='red', color='black', alpha=0.3, bins=50)+
  geom_histogram(aes(`b_diagnosisASD:visit`),
fill='green', color='black', alpha=0.3, bins=50)+
  theme_classic()+
  ggtitle('prior-posterior update check on slope for
ASD')+
  xlab("Slope for ASD")

plot4 <- ggplot(posterior)+
  geom_histogram(aes(`prior_b_diagnosisTD:visit`),
fill='red', color='black', alpha=0.3, bins=50)+
  geom_histogram(aes(`b_diagnosisTD:visit`),
fill='green', color='black', alpha=0.3, bins=50)+
  theme_classic()+
  ggtitle('prior-posterior update check on slope for
TD')+
  xlab("slope for TD")


plot5 <- ggplot(posterior)+
  geom_histogram(aes(prior_cor_ID), fill='red',
color='black', alpha=0.3, bins=50)+
  geom_histogram(aes(cor_ID__Intercept__visit),
fill='green', color='black', alpha=0.3, bins=50)+
  theme_classic()+
  ggtitle('prior-posterior update check on correlation
between varying intercepts and slopes')+
  xlab("Correlation")

plot6 <- ggplot(posterior)+
```

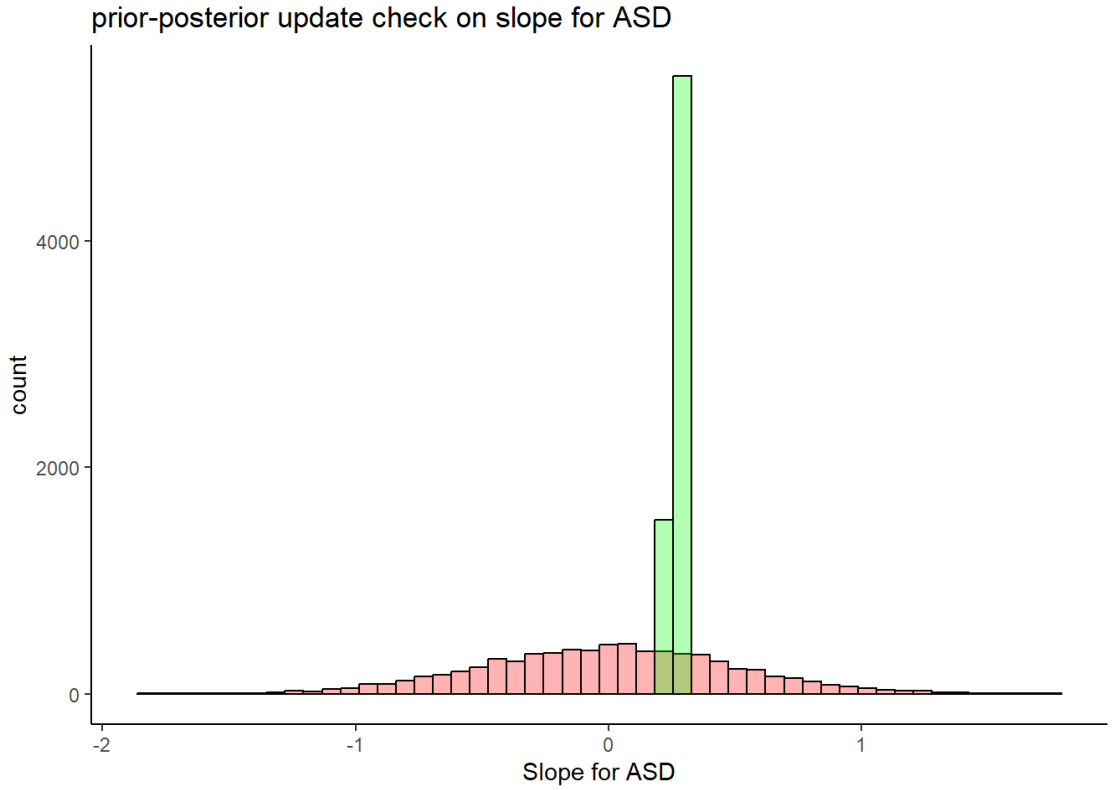
```
  geom_histogram(aes(prior_sd_ID), fill='red',
color='black', alpha=0.3, bins=50)+
  geom_histogram(aes(sd_ID__Intercept), fill='green',
color='black', alpha=0.3, bins=50)+
  theme_classic()+
  ggtitle('Prior-posterior update check, the
variability of the intercept')+
  xlab("Intercept")

plot7 <- ggplot(posterior)+
  geom_histogram(aes(prior_sd_ID), fill='red',
color='black', alpha=0.3, bins=50)+
  geom_histogram(aes(sd_ID__visit), fill='green',
color='black', alpha=0.3, bins=50)+
  theme_classic()+
  ggtitle('Prior-posterior update check, the
variability of the slopes')+
  xlab("Intercept")


plot1
```


prior-posterior update check on interceps for ASD
count
5000
4000
3000
2000
1000
0
0
1
2
3
intercept for ASD

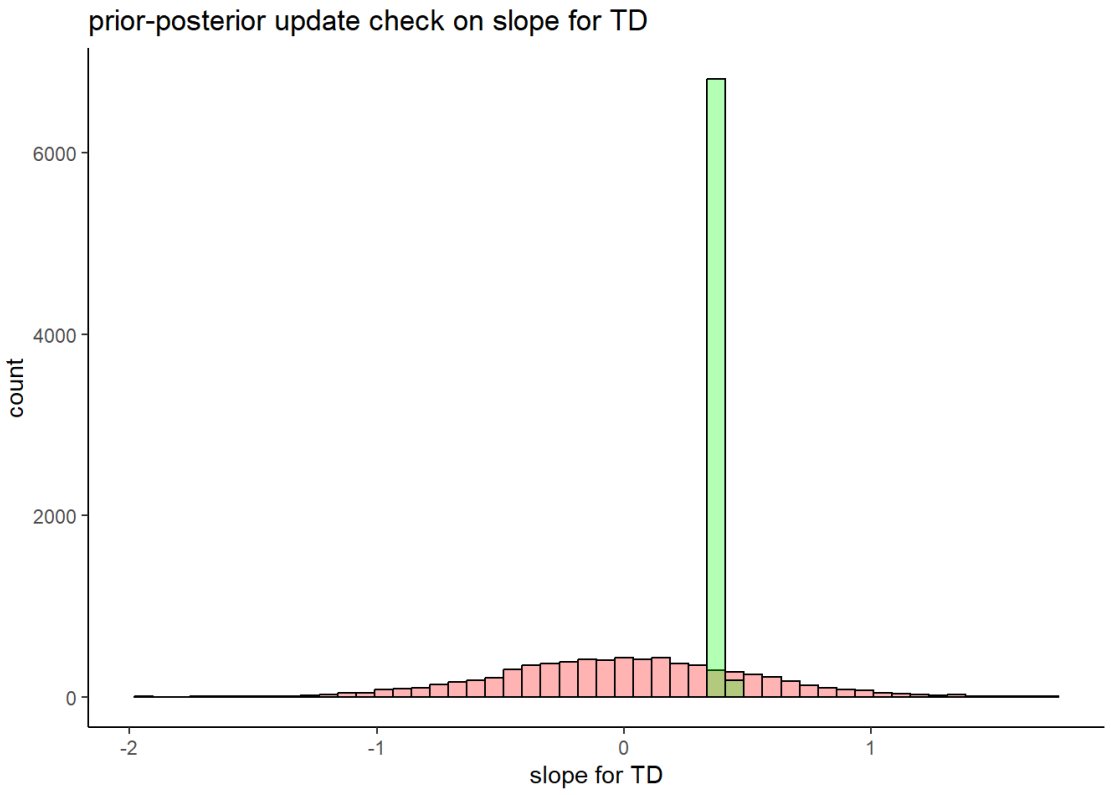
```
plot2
```


prior-posterior update check on intercept for TD
count
4000
3000
2000
1000
0
0
1
2
intercept for TD

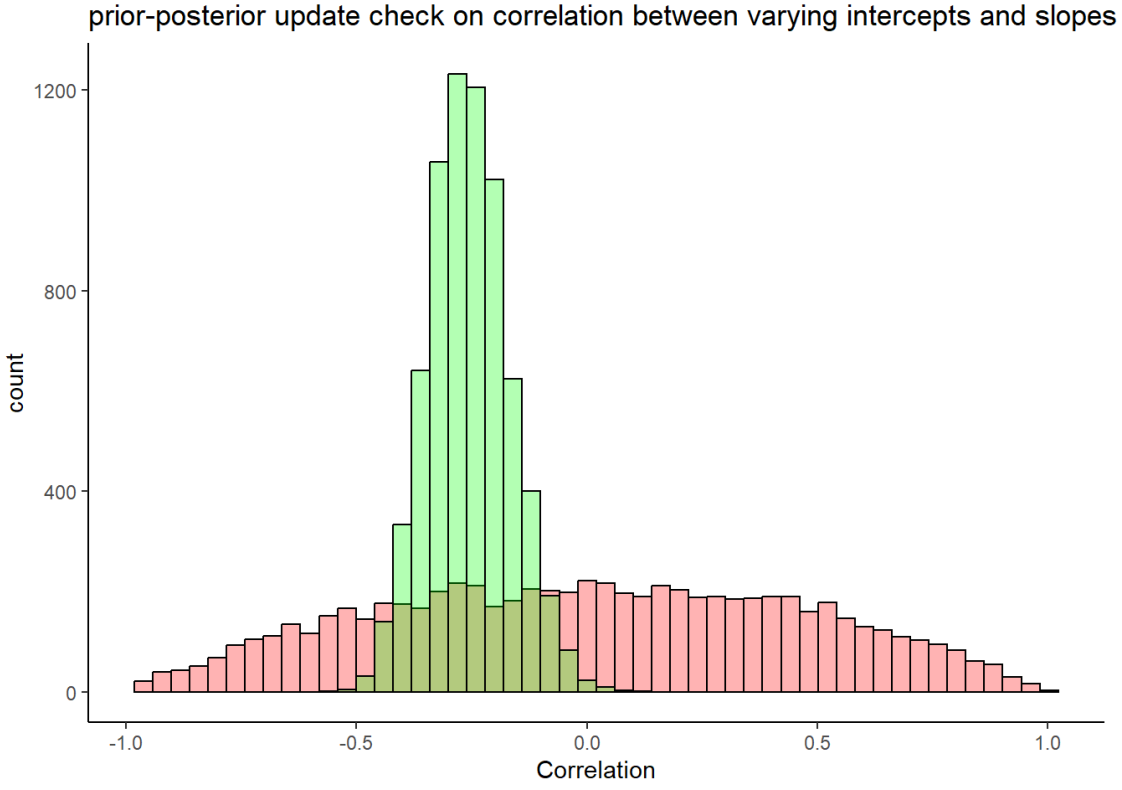
```
plot3
```


prior-posterior update check on slope for ASD
count
4000
2000
0
-2
-1
0
1
Slope for ASD

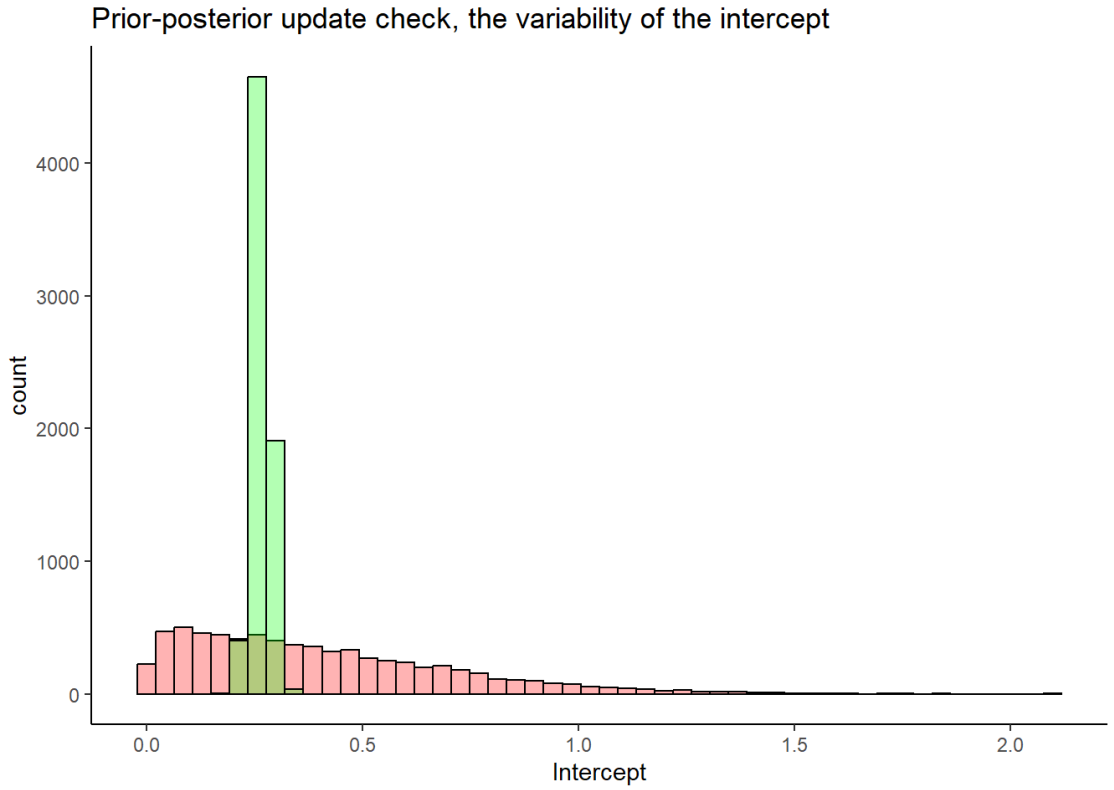
```
plot4
```


prior-posterior update check on slope for TD
6000
4000
2000
0
count
-2
-1
0
1
slope for TD

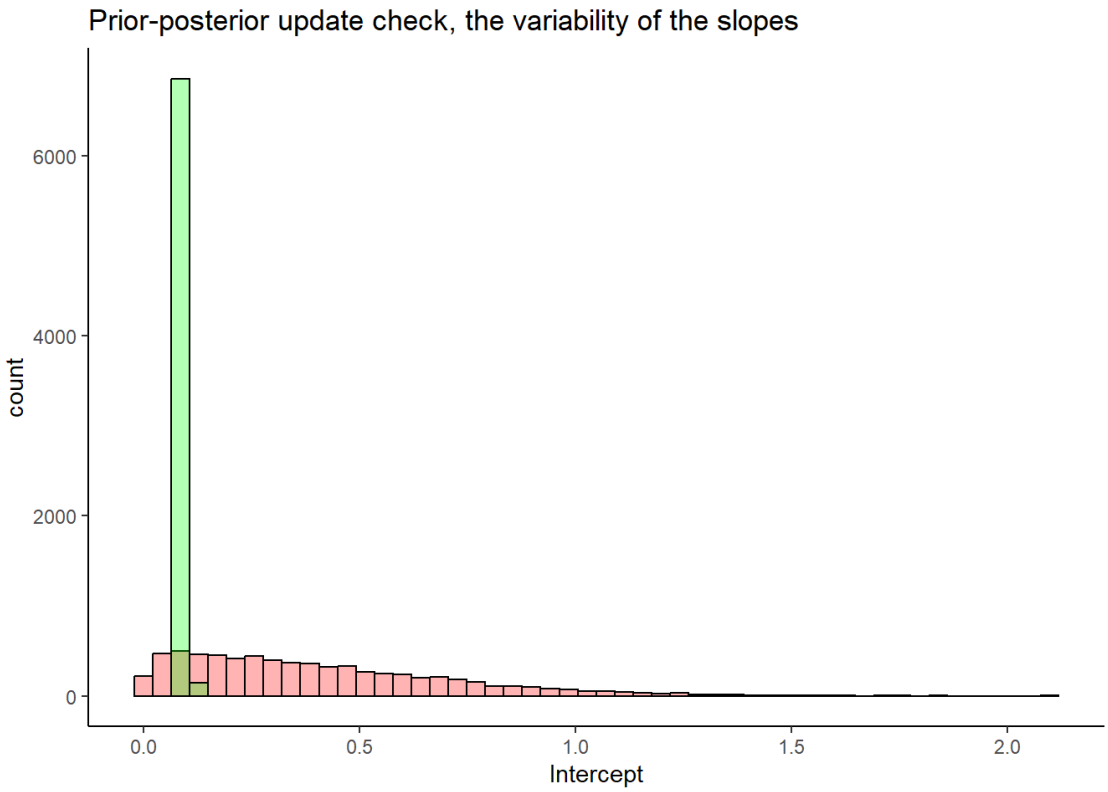
```
plot5
```


prior-posterior update check on correlation between varying intercepts and slopes
1200
800
400
0
count
-1.0
-0.5
0.0
0.5
1.0
Correlation

```
plot6
```


Prior-posterior update check, the variability of the intercept
count
4000
3000
2000
1000
0
0.0
0.5
1.0
1.5
2.0
Intercept

```
plot7
```



Estimating effectsize, baysian power analysis

Function simulates data and return CI of slope difference

```
fun_sim_data <- function(seed,n){
  set.seed(seed)

  average_mlu <- log(1.5)
sd_mlu_asd <- log(1.5+0.5)-log(1.5)
sd_mlu_td <- log(1.5+0.3)-log(1.5)

change_mlu_asd <- 0.4/1.5
change_mlu_td <- 0.6/1.5
change_sd_mlu_asd <- 0.4*(0.4/1.5)
change_sd_mlu_td <- 0.2*(0.6/1.5)
```

```
 e <- 0.2

 int_asd <- rnorm(n, mean=average_mlu, sd=sd_mlu_asd)
int_td <- rnorm(n, mean=average_mlu, sd=sd_mlu_td)

slope_asd <- rnorm(n, mean=change_mlu_asd,
sd=change_sd_mlu_asd)
slope_td <- rnorm(n, mean = change_mlu_td,
sd=change_sd_mlu_td)



   data <-
     tibble(diagnosis=rep(c('TD', 'ASD'), each=n)) %>%
     mutate(intercept=ifelse(diagnosis=='TD', int_td,
int_asd)) %>%

     mutate(slope=ifelse(diagnosis=='TD', slope_td,
slope_asd)) %>%
     mutate(error=ifelse(diagnosis=='TD', e, e)) %>%
     dplyr::mutate(ID=row_number()) %>%
     slice(rep(1:n(), each=6)) %>%
     add_column(visit=rep(c(1,2,3,4,5,6), times=n+n))

 for(i in seq(nrow(data))){
   data$MLU[i] <- exp(rnorm(1,data$intercept[i]+
(data$slope[i]*(data$visit[i]-1)), data$error[i]))
 }
   data <- data[,c(1,5,6,2,3,4,7)]
   post <- update(MLU_prior_m1_fit,
                  newdata = data,
                  seed=seed) %>%
     as_draws_df() %>%
     mutate(slope_diff=(`b_diagnosisTD:visit`-
`b_diagnosisASD:visit`))
```

```
   CI <- as.data.frame(t(quantile(post$slope_diff,
probs=c(0.025, 0.975)))) %>%
     add_column(mean=mean(post$slope_diff))
   return(CI)}
```

Manuela #### running the functiong with different amount of participants

```
# n_sim <- 10
#
# s10 <- tibble(seed=1:n_sim) %>%
#   mutate(b1=purrr::map(seed, fun_sim_data, n=10)) %>%
#   unnest(b1)
#
# s15 <- tibble(seed=1:n_sim) %>%
#   mutate(b1=purrr::map(seed, fun_sim_data, n=15)) %>%
#   unnest(b1)
#
# s20 <- tibble(seed=1:n_sim) %>%
#   mutate(b1=purrr::map(seed, fun_sim_data, n=20)) %>%
#   unnest(b1)
#
# s30 <- tibble(seed=1:n_sim) %>%
#   mutate(b1=purrr::map(seed, fun_sim_data, n=30)) %>%
#   unnest(b1)
#
# s40 <- tibble(seed=1:n_sim) %>%
#   mutate(b1=purrr::map(seed, fun_sim_data, n=40)) %>%
#   unnest(b1)
#
# s50 <- tibble(seed=1:n_sim) %>%
#   mutate(b1=purrr::map(seed, fun_sim_data, n=50)) %>%
#   unnest(b1)
#
# s75 <- tibble(seed=1:n_sim) %>%
```

```
#   mutate(b1=purrr::map(seed, fun_sim_data, n=75)) %>%
#   unnest(b1)
#
# s100 <- tibble(seed=1:n_sim) %>%
#   mutate(b1=purrr::map(seed, fun_sim_data, n=100))
%>%
#   unnest(b1)
#
# s180 <- tibble(seed=1:n_sim) %>%
#   mutate(b1=purrr::map(seed, fun_sim_data, n=180))
%>%
#   unnest(b1)
#
# s250 <- tibble(seed=1:n_sim) %>%
#   mutate(b1=purrr::map(seed, fun_sim_data, n=250))
%>%
#   unnest(b1)
#
# s300 <- tibble(seed=1:n_sim) %>%
#   mutate(b1=purrr::map(seed, fun_sim_data, n=300))
%>%
#   unnest(b1)
```

Effectsize of slope difference (plots)

```
# plot_s10 <- s10 %>%
#   ggplot(aes(x=seed, y=mean, ymin = `2.5%`, ymax=
`97.5%` ))+
#   geom_pointrange(fatten = 1/2)+
#   geom_hline(yintercept = c(0, 0.5), colour= 'green')
+
#   labs(x="seed (simulation index)", y= "slope
difference")+
#   ggtitle("slope difference, 10 participants")
#
# plot_s15 <- s15 %>%
```

```
#   ggplot(aes(x=seed, y=mean, ymin = `2.5%`, ymax=
`97.5%` ))+
#   geom_pointrange(fatten = 1/2)+
#   geom_hline(yintercept = c(0, 0.5), colour= 'green')
+
#   labs(x="seed (simulation index)", y= "slope
difference")+
#   ggtitle("slope difference, 15 participants")
#
# plot_s20 <- s20 %>%
#   ggplot(aes(x=seed, y=mean, ymin = `2.5%`, ymax=
`97.5%` ))+
#   geom_pointrange(fatten = 1/2)+
#   geom_hline(yintercept = c(0, 0.5), colour= 'green')
+
#   labs(x="seed (simulation index)", y= "slope
difference")+
#   ggtitle("slope difference, 20 participants")
#
# plot_s30 <- s30 %>%
#   ggplot(aes(x=seed, y=mean, ymin = `2.5%`, ymax=
`97.5%` ))+
#   geom_pointrange(fatten = 1/2)+
#   geom_hline(yintercept = c(0, 0.5), colour= 'green')
+
#   labs(x="seed (simulation index)", y= "slope
difference")+
#   ggtitle("slope difference, 30 participants")
#
# plot_s40 <- s40 %>%
#   ggplot(aes(x=seed, y=mean, ymin = `2.5%`, ymax=
`97.5%` ))+
#   geom_pointrange(fatten = 1/2)+
#   geom_hline(yintercept = c(0, 0.5), colour= 'green')
+
```

```
#   labs(x="seed (simulation index)", y= "slope
difference")+
#   ggtitle("slope difference, 40 participants")
#
# plot_s50 <- s50 %>%
#   ggplot(aes(x=seed, y=mean, ymin = `2.5%`, ymax=
`97.5%` ))+
#   geom_pointrange(fatten = 1/2)+
#   geom_hline(yintercept = c(0, 0.5), colour= 'green')
+
#   labs(x="seed (simulation index)", y= "slope
difference")+
#   ggtitle("slope difference, 50 participants")
#
# plot_s75 <- s75 %>%
#   ggplot(aes(x=seed, y=mean, ymin = `2.5%`, ymax=
`97.5%` ))+
#   geom_pointrange(fatten = 1/2)+
#   geom_hline(yintercept = c(0, 0.5), colour= 'green')
+
#   labs(x="seed (simulation index)", y= "slope
difference")+
#   ggtitle("slope difference, 75 participants")
#
# plot_s100 <- s100 %>%
#   ggplot(aes(x=seed, y=mean, ymin = `2.5%`, ymax=
`97.5%` ))+
#   geom_pointrange(fatten = 1/2)+
#   geom_hline(yintercept = c(0, 0.5), colour= 'green')
+
#   labs(x="seed (simulation index)", y= "slope
difference")+
#   ggtitle("slope difference, 100 participants")
#
# plot_s180 <- s180 %>%
```

```
#   ggplot(aes(x=seed, y=mean, ymin = `2.5%`, ymax=
`97.5%` ))+
#   geom_pointrange(fatten = 1/2)+
#   geom_hline(yintercept = c(0, 0.5), colour= 'green')
+
#   labs(x="seed (simulation index)", y= "slope
difference")+
#   ggtitle("slope difference, 180 participants")
#
# plot_s250 <- s250 %>%
#   ggplot(aes(x=seed, y=mean, ymin = `2.5%`, ymax=
`97.5%` ))+
#   geom_pointrange(fatten = 1/2)+
#   geom_hline(yintercept = c(0, 0.5), colour= 'green')
+
#   labs(x="seed (simulation index)", y= "slope
difference")+
#   ggtitle("slope difference, 150 participants")
#
# plot_s300 <- s300 %>%
#   ggplot(aes(x=seed, y=mean, ymin = `2.5%`, ymax=
`97.5%` ))+
#   geom_pointrange(fatten = 1/2)+
#   geom_hline(yintercept = c(0, 0.5), colour= 'green')
+
#   labs(x="seed (simulation index)", y= "slope
difference")+
#   ggtitle("slope difference, 300 participants")
#
# grid.arrange(
#   plot_s10,
#   plot_s15,
#   plot_s20,
#   plot_s30,
#   plot_s40)
```

```
# grid.arrange(
#   plot_s50,
#   plot_s75,
#   plot_s100,
#   plot_s180,
#   plot_s250,
#   plot_s300)
```

Patrik #### Power analysis

```
# power_analysis_fun <- function(sim_nr, n){
#   sim_nr %>%
#     mutate(two_half=ifelse(`2.5%`>0,1,0 )) %>%
#     summarise(power=mean(two_half)) %>%
#     add_column(number_of_participants=n)
# }
#
# power_analysis_sum <- bind_rows(
#   power_analysis_fun(s10, 10),
#   power_analysis_fun(s15, 15),
#   power_analysis_fun(s20, 20),
#   power_analysis_fun(s30, 30),
#   power_analysis_fun(s40, 40),
#   power_analysis_fun(s50, 50),
#   power_analysis_fun(s75, 75),
#   power_analysis_fun(s100, 100),
#   power_analysis_fun(s180, 180),
#   power_analysis_fun(s250, 250),
#   power_analysis_fun(s300, 300))
#
# power_analysis_sum
```

Part 2 - Strong in the Bayesian ken, you are now

ready to analyse the actual data

- Describe your sample (n, age, gender, clinical and cognitive features of the two groups) and critically assess whether the groups (ASD and TD) are balanced. Briefly discuss whether the data is enough given the simulations in part 1.

- Describe linguistic development (in terms of MLU over time) in TD and ASD children (as a function of group). Discuss the difference (if any) between the two groups.

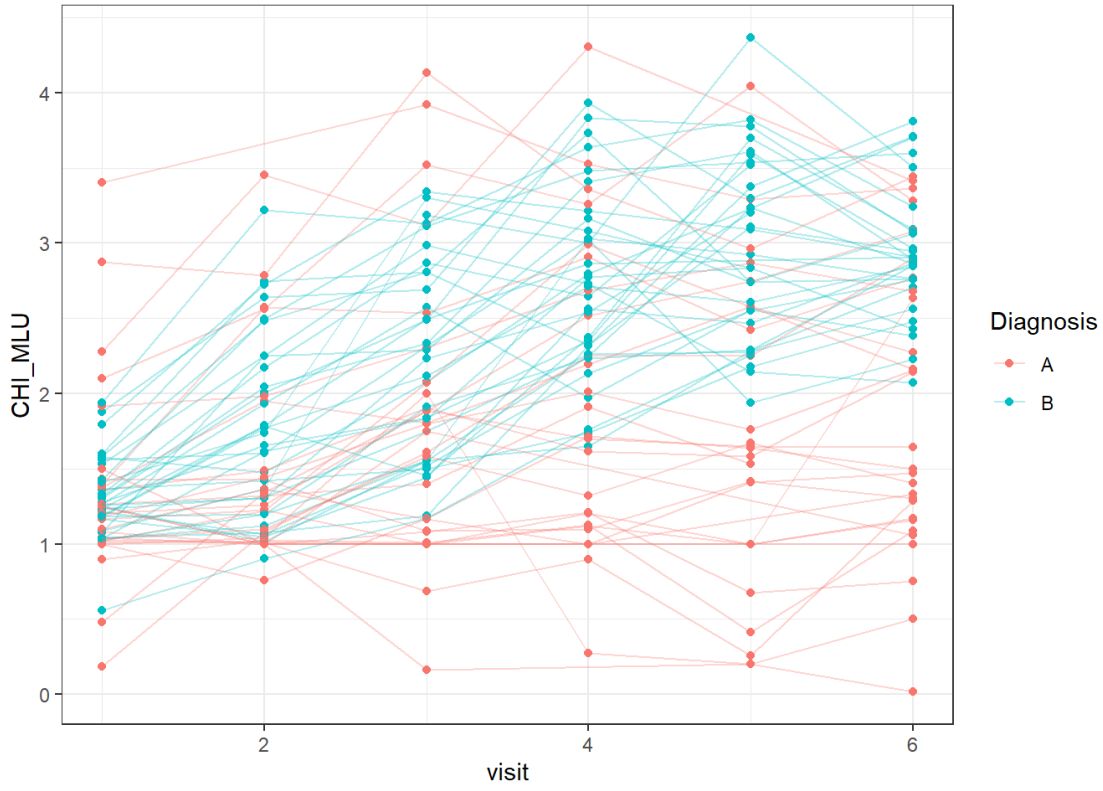
- Describe individual differences in linguistic development: do all kids follow the same path? Are all kids reflected by the general trend for their group?

- Include additional predictors in your model of language development (N.B. not other indexes of child language: types and tokens, that'd be cheating). Identify the best model, by conceptual reasoning, model comparison or a mix. Report the model you choose (and name its competitors, if any) and discuss why it's the best model.

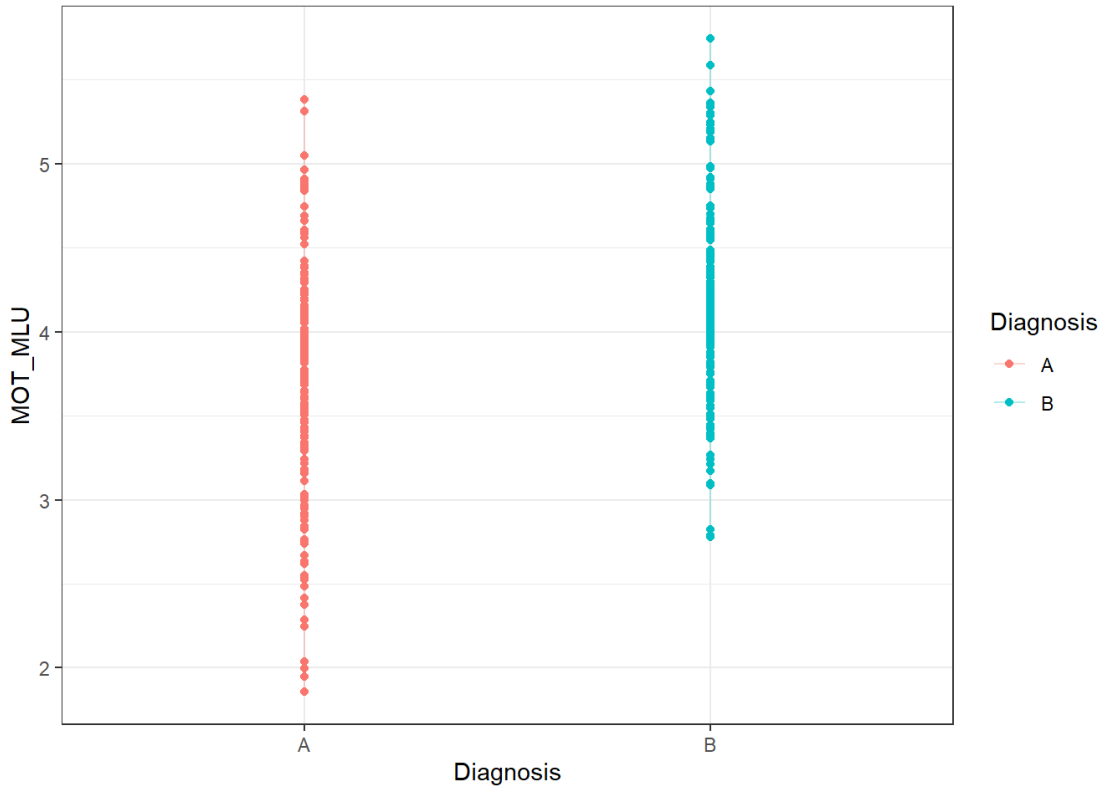
```
real_data <- read_csv('assignment_data_clean.csv')
## Rows: 352 Columns: 20
## ── Column specification
──────────────────────────────────────────────────────────────────────
─
## Delimiter: ","
```

```
## chr  (3): Diagnosis, Ethnicity, Gender
## dbl (17): id, ADOS1, non_verbalIQ1, verbalIQ1,
Socialization1, visit, Age, A...
##
## ℹ Use `spec()` to retrieve the full column
specification for this data.
## ℹ Specify the column types or set `show_col_types =
FALSE` to quiet this message.
unique(real_data$id)
##  [1]  1 10 11 12 13 14 15 16 17 18 19  2 20 21 22 23
24 25 26 27 28 29  3 30 31
## [26] 32 33 34 35 36 37 38 39  4 40 41 42 43 44 45 46
47 48 49  5 50 51 52 53 54
## [51] 55 56 57 58 59  6 60 61  7  8  9
real_data <- real_data %>%
  mutate(Diagnosis=as.factor(Diagnosis))

real_data %>%
  group_by(Gender) %>%
  filter(visit==1) %>%
  count()

#Counting participants
real_data %>%
  group_by(Diagnosis) %>%
  filter(visit==1) %>%
  count()

##o use lognormal distribution we cannot have negative
MLU, so we filter it
real_data <- real_data %>%
  filter(!CHI_MLU<=0)
ggplot(real_data, aes(visit,CHI_MLU, color=Diagnosis,
group=id))+
  theme_bw()+
```

```
  geom_point()+
  geom_line(alpha=0.3)
```




```
ggplot(real_data, aes(Diagnosis, MOT_MLU,
color=Diagnosis))+
  theme_bw()+
  geom_point()+
  geom_line(alpha=0.3)
```




```
MLU_fit<- bf(CHI_MLU ~ 0 + Diagnosis + Diagnosis:visit
+ (1 + visit|id))

get_prior(data = real_data, family = lognorm_fam,
MLU_fit)
```

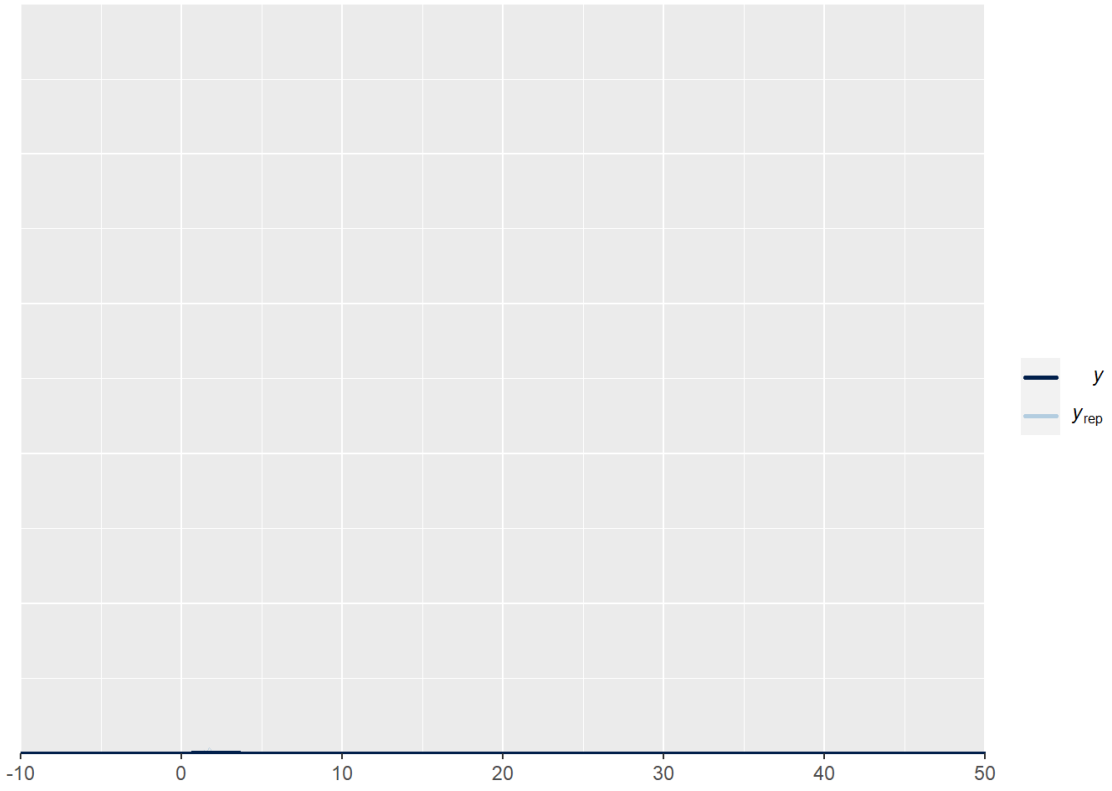
```
#Simulating priors
priors_sim<-c(
prior(normal(0,0.2),class=b),
prior(normal(0.5,0.05),class=b,coef="DiagnosisA"),
prior(normal(0.5,0.02),class=b,coef="DiagnosisB"),
prior(normal(0,0.06),class=b,coef="DiagnosisA:visit"),
prior(normal(0,0.03),class=b,coef="DiagnosisB:visit"),
prior(normal(0,0.2),class=sd,coef=Intercept,group=id),
prior(normal(0,0.1),class=sd,coef=visit,group=id),
prior(normal(0,0.2),class=sigma),
prior(lkj(2),class="cor"))
MLU_prior <- brm(
```

```
  MLU_fit,
  data = real_data,
  prior = priors_sim,
  family = lognorm_fam,
  refresh=0,
  sample_prior = 'only',
  iter=6000,
  warmup = 2500,
  backend = "cmdstanr",
  threads = threading(2),
  chains = 2,
  cores = 2,
  control = list(
    adapt_delta = 0.99,
    max_treedepth = 20
)
)
```

###prior predictive checks

```
pp_check(MLU_prior, ndraws = 100)+
  xlim(-10,50)+
  ylim(0,500)
```

```
## Warning: Removed 9 rows containing non-finite values
(`stat_density()`).
```



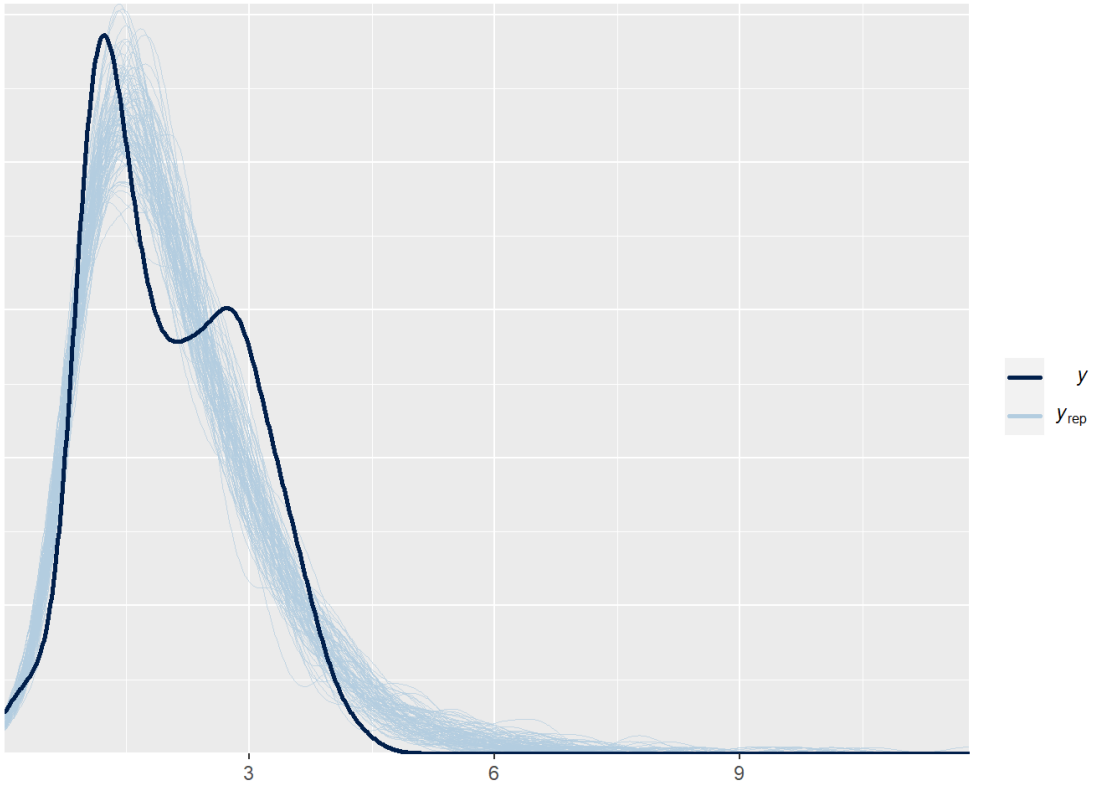
###fit the model

```
MLU_prior_fit <- brm(
  MLU_fit,
  data = real_data,
  prior = priors_sim,
  family = lognorm_fam,
  refresh=0,
  sample_prior = TRUE,
  iter=6000,
  warmup = 2500,
  backend = "cmdstanr",
  threads = threading(2),
  chains = 2,
  cores = 2,
  control = list(
    adapt_delta = 0.99,
    max_treedepth = 20
```

```
)
)
```

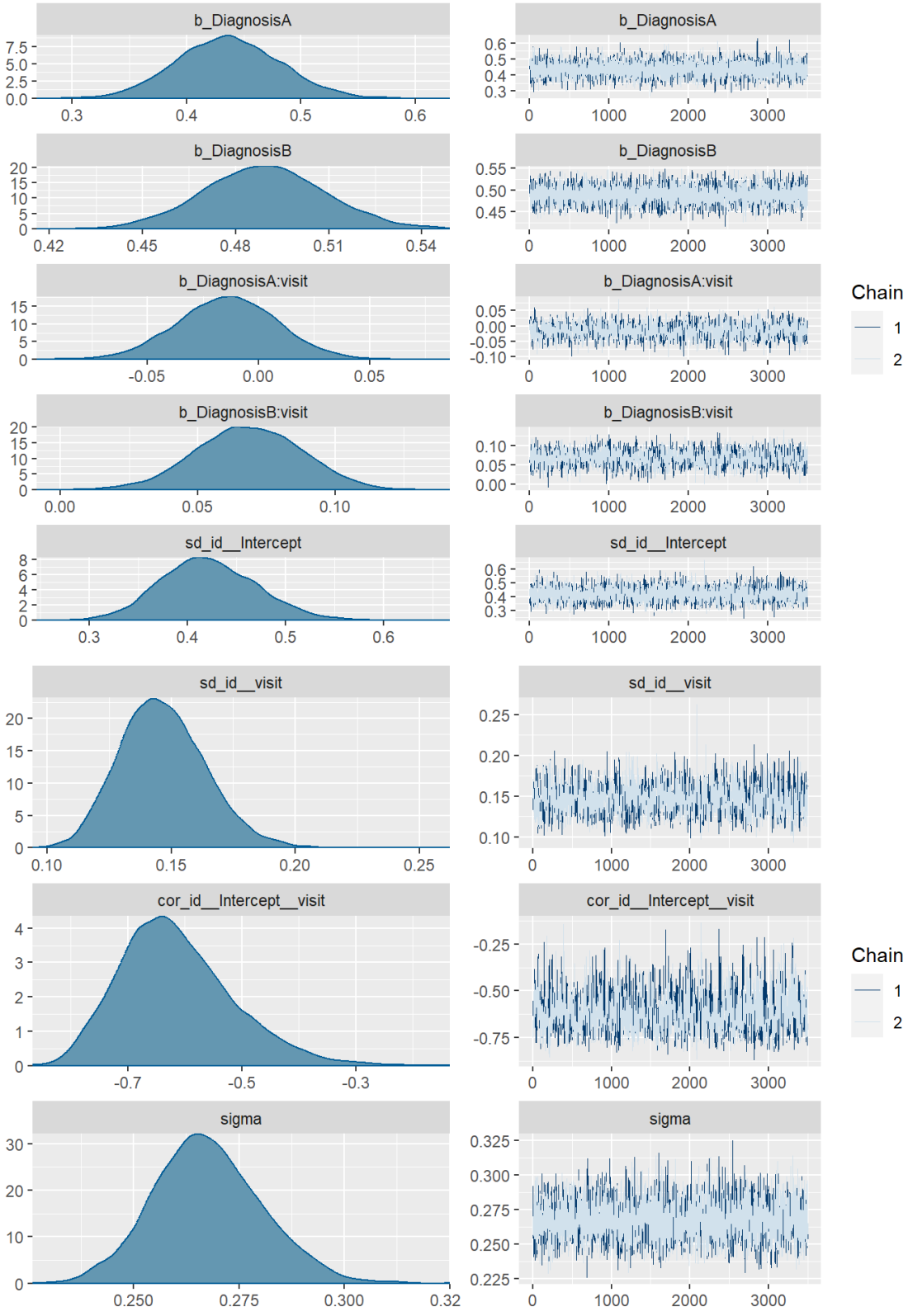
Sara ###posterior predictive check

```
pp_check(MLU_prior_fit, ndraws = 100)
```



###traceplot for fitted model

```
plot(MLU_prior_fit, ndraws = 100)
```


b_DiagnosisA
b_DiagnosisB
b_DiagnosisA:visit
b_DiagnosisB:visit
sd_id__Intercept
sd_id__visit
cor_id__Intercept__visit
sigma
Chain
1
2

parameter recovery from fitted model

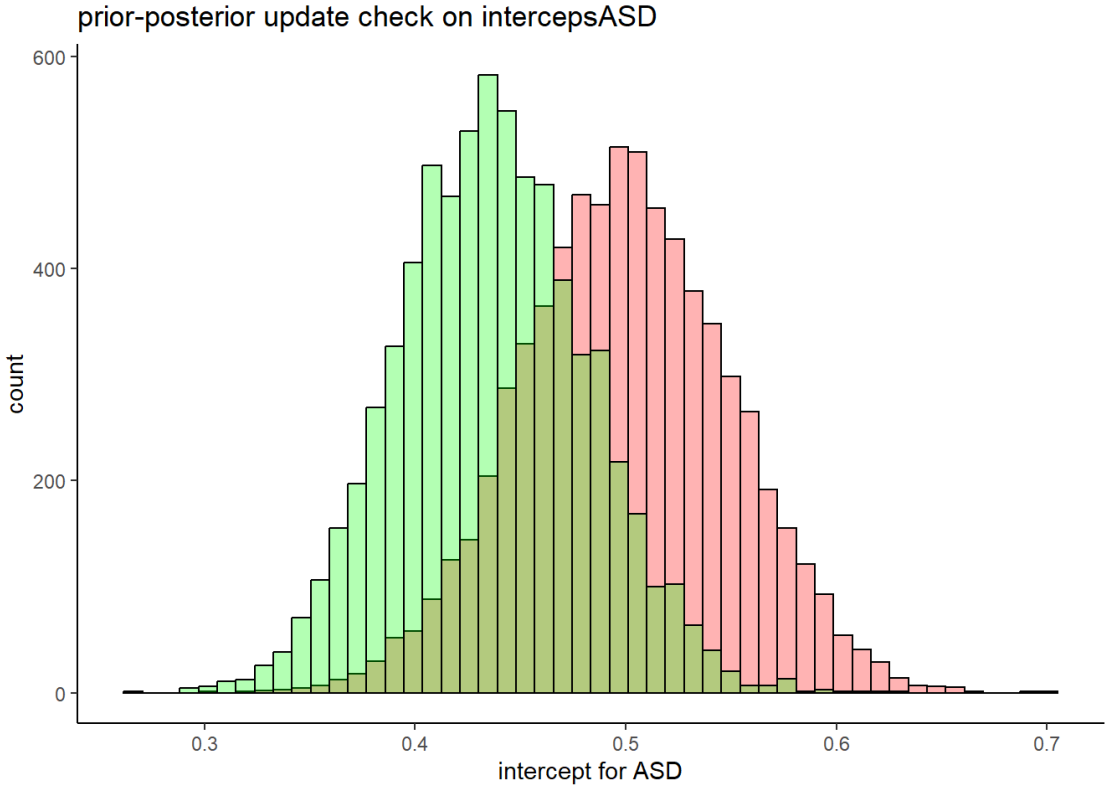
```
print(MLU_prior_fit)
##  Family: lognormal
##   Links: mu = identity; sigma = identity
## Formula: CHI_MLU ~ 0 + Diagnosis + Diagnosis:visit +
(1 + visit | id)
##    Data: real_data (Number of observations: 349)
##   Draws: 2 chains, each with iter = 6000; warmup =
2500; thin = 1;
##          total post-warmup draws = 7000
##
## Group-Level Effects:
## ~id (Number of levels: 61)
##                       Estimate Est.Error l-95% CI
u-95% CI Rhat Bulk_ESS
## sd(Intercept)             0.42      0.05     0.33
0.52 1.00     2532
## sd(visit)                 0.15      0.02     0.12
0.18 1.00      891
## cor(Intercept,visit)    -0.61      0.10    -0.78
-0.39 1.00      770
##                       Tail_ESS
## sd(Intercept)             4522
## sd(visit)                 2270
## cor(Intercept,visit)      1836
##
## Population-Level Effects:
##                   Estimate Est.Error l-95% CI u-95%
CI Rhat Bulk_ESS Tail_ESS
## DiagnosisA           0.44      0.04     0.35
0.53 1.00     6348     5528
## DiagnosisB           0.49      0.02     0.45
0.53 1.00    10730     5376
## DiagnosisA:visit    -0.01      0.02    -0.06
```

```
0.03 1.00     1301     2684
## DiagnosisB:visit     0.07      0.02     0.03
0.11 1.00     1158     2346
##
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat
Bulk_ESS Tail_ESS
## sigma     0.27      0.01     0.24     0.29 1.00
4392     4575
##
## Draws were sampled using sample(hmc). For each
parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and
Rhat is the potential
## scale reduction factor on split chains (at
convergence, Rhat = 1).
posterior <- as_draws_df(MLU_prior_fit)

plot1 <- ggplot(posterior)+
  geom_histogram(aes(prior_b_DiagnosisA), fill='red',
color='black', alpha=0.3, bins=50)+
  geom_histogram(aes(b_DiagnosisA), fill='green',
color='black', alpha=0.3, bins=50)+
  theme_classic()+
  ggtitle('prior-posterior update check on
intercepsASD')+
  xlab('intercept for ASD')

plot2 <- ggplot(posterior)+
  geom_histogram(aes(prior_b_DiagnosisB), fill='red',
color='black', alpha=0.3, bins=50)+
  geom_histogram(aes(b_DiagnosisB), fill='green',
color='black', alpha=0.3, bins=50)+
  theme_classic()+
  ggtitle('prior-posterior update check on intercept
```

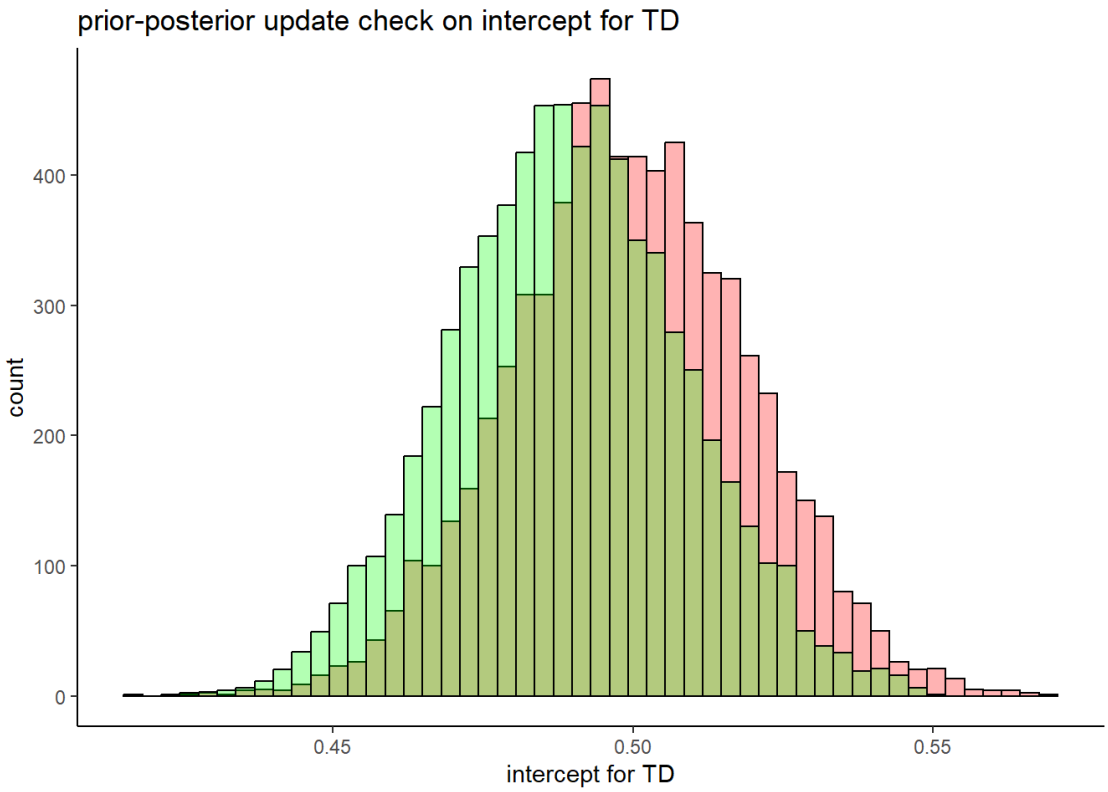
```
for TD')+
  xlab('intercept for TD')

plot3 <- ggplot(posterior)+
  geom_histogram(aes(`prior_b_DiagnosisA:visit`),
fill='red', color='black', alpha=0.3, bins=50)+
  geom_histogram(aes(`b_DiagnosisA:visit`),
fill='green', color='black', alpha=0.3, bins=50)+
  theme_classic()+
  ggtitle('prior-posterior update check on slope for
ASD')+
  xlab("Slope for ASD")

plot4 <- ggplot(posterior)+
  geom_histogram(aes(`prior_b_DiagnosisB:visit`),
fill='red', color='black', alpha=0.3, bins=50)+
  geom_histogram(aes(`b_DiagnosisB:visit`),
fill='green', color='black', alpha=0.3, bins=50)+
  theme_classic()+
  ggtitle('prior-posterior update check on slope for
TD')+
  xlab("slope for TD")


plot5 <- ggplot(posterior)+
  geom_histogram(aes(prior_cor_id), fill='red',
color='black', alpha=0.3, bins=50)+
  geom_histogram(aes(cor_id__Intercept__visit),
fill='green', color='black', alpha=0.3, bins=50)+
  theme_classic()+
  ggtitle('prior-posterior update check on correlation
between varying intercepts and slopes')+
  xlab("Correlation")

plot6 <- ggplot(posterior)+
```

```
  geom_histogram(aes(prior_sd_id__Intercept),
fill='red', color='black', alpha=0.3, bins=50)+
  geom_histogram(aes(sd_id__Intercept), fill='green',
color='black', alpha=0.3, bins=50)+
  theme_classic()+
  ggtitle('Prior-posterior update check, the
variability of the intercept')+
  xlab("Intercept")

plot7 <- ggplot(posterior)+
  geom_histogram(aes(prior_sd_id__visit), fill='red',
color='black', alpha=0.3, bins=50)+
  geom_histogram(aes(sd_id__visit), fill='green',
color='black', alpha=0.3, bins=50)+
  theme_classic()+
  ggtitle('Prior-posterior update check, the
variability of the slopes')+
  xlab("Intercept")

plot8 <- ggplot(posterior)+
  geom_histogram(aes(`prior_b_DiagnosisA:visit`),
fill='red', color='black', alpha=0.3, bins=50)+
  geom_histogram(aes(`b_DiagnosisA:visit`),
fill='green', color='black', alpha=0.3, bins=50)+
  theme_classic()+
  geom_histogram(aes(`b_DiagnosisB:visit`),
fill='yellow', color='black', alpha=0.3, bins=50)+
  theme_classic()+
  ggtitle('prior-posterior update check on slope')+
  xlab("Slope")

plot9 <- ggplot(posterior)+
  geom_histogram(aes(prior_b_DiagnosisA), fill='red',
color='black', alpha=0.3, bins=50)+
  geom_histogram(aes(b_DiagnosisA), fill='green',
```

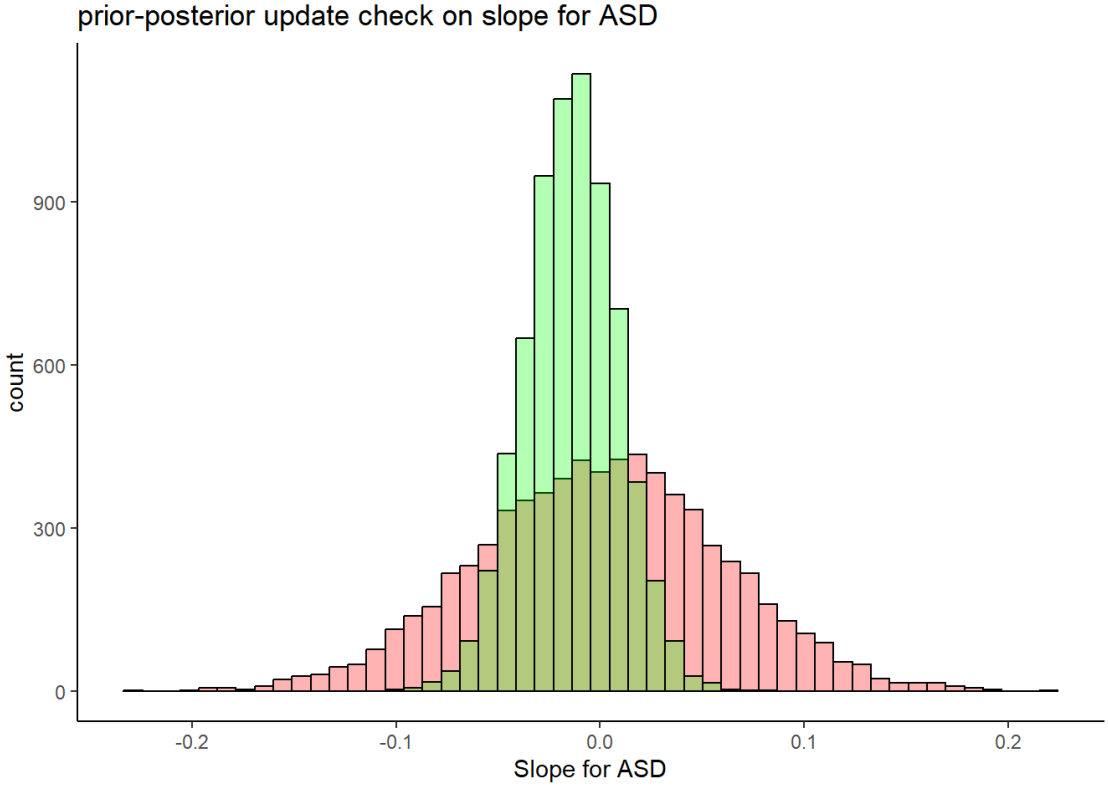
```
color='black', alpha=0.3, bins=50)+
  theme_classic()+
   geom_histogram(aes(b_DiagnosisB), fill='yellow',
color='black', alpha=0.3, bins=50)+
  theme_classic()+
  ggtitle('prior-posterior update check on interceps')+
  xlab('intercept')


plot1
```



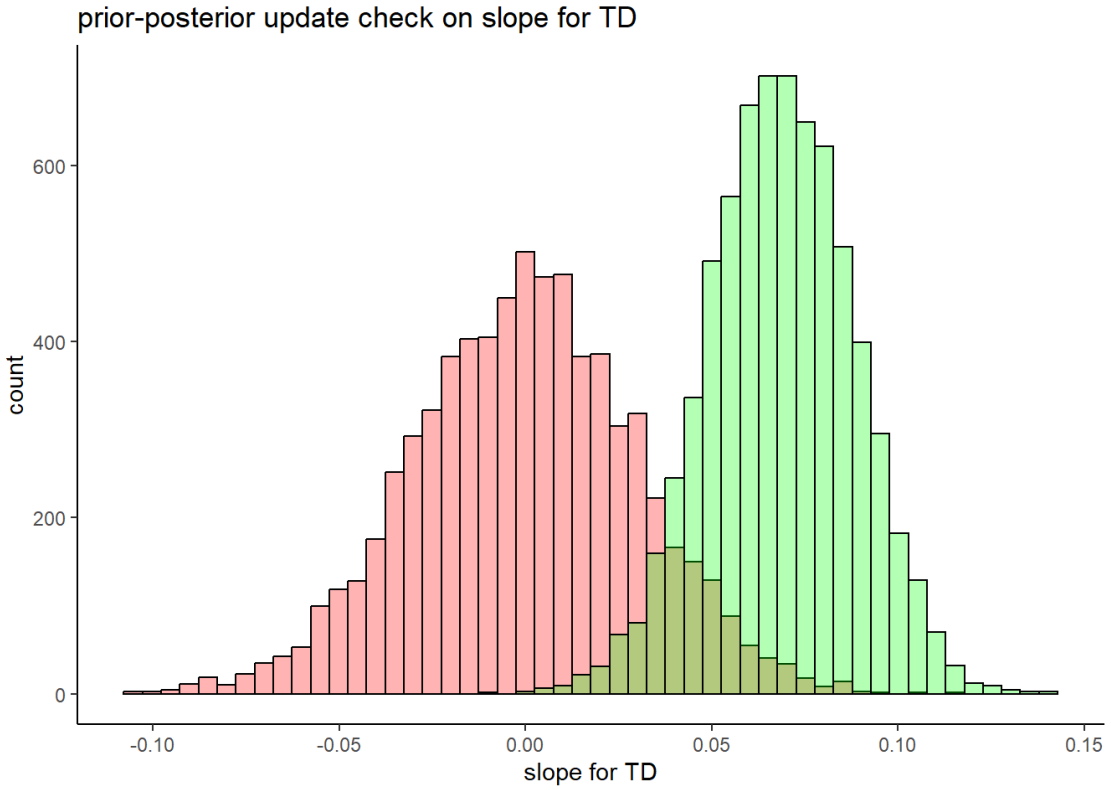

```
plot2
```


prior-posterior update check on intercept for TD
count
400
300
200
100
0
0.45
0.50
0.55
intercept for TD

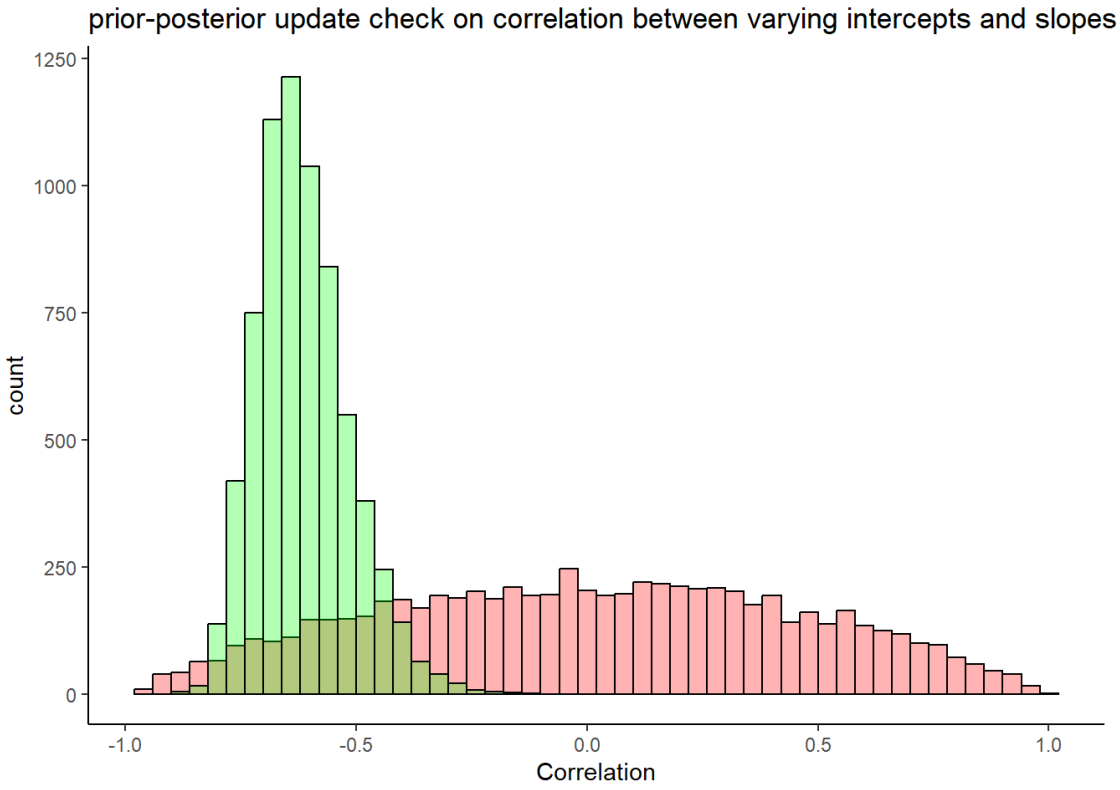
```
plot3
```


prior-posterior update check on slope for ASD
count
900
600
300
0
-0.2
-0.1
0.0
0.1
0.2
Slope for ASD

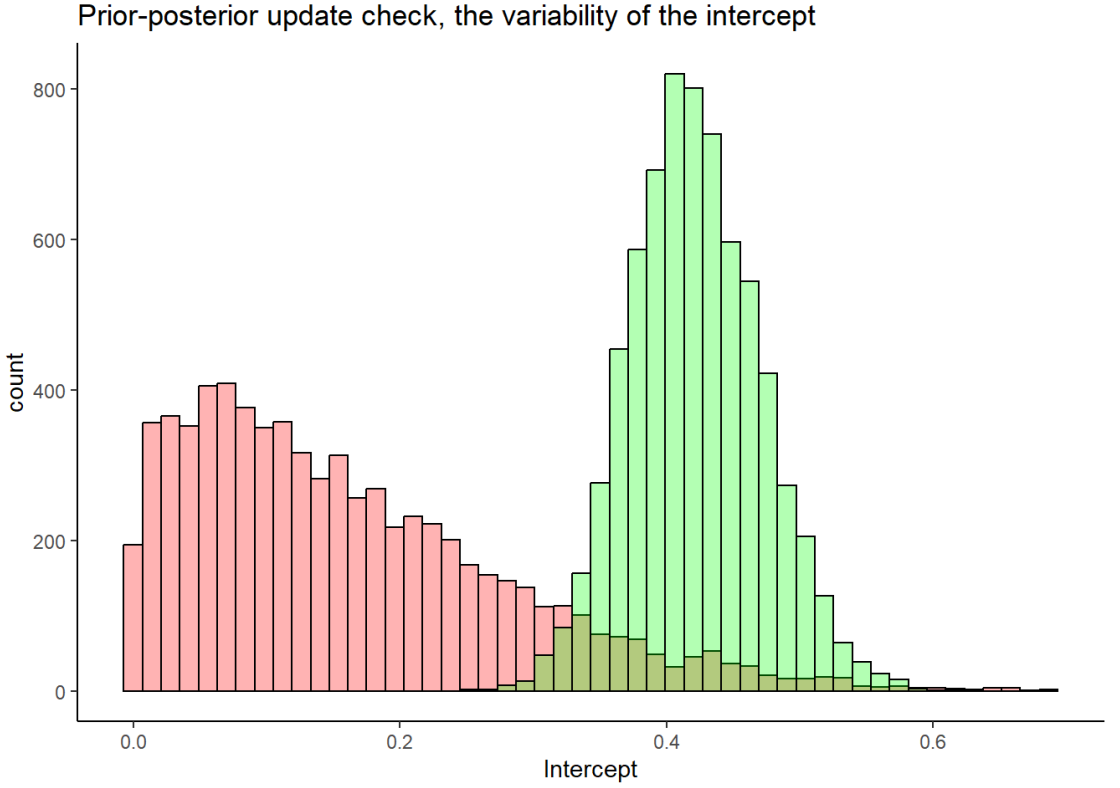
```
plot4
```


prior-posterior update check on slope for TD
count
600
400
200
0
-0.10
-0.05
0.00
0.05
0.10
0.15
slope for TD

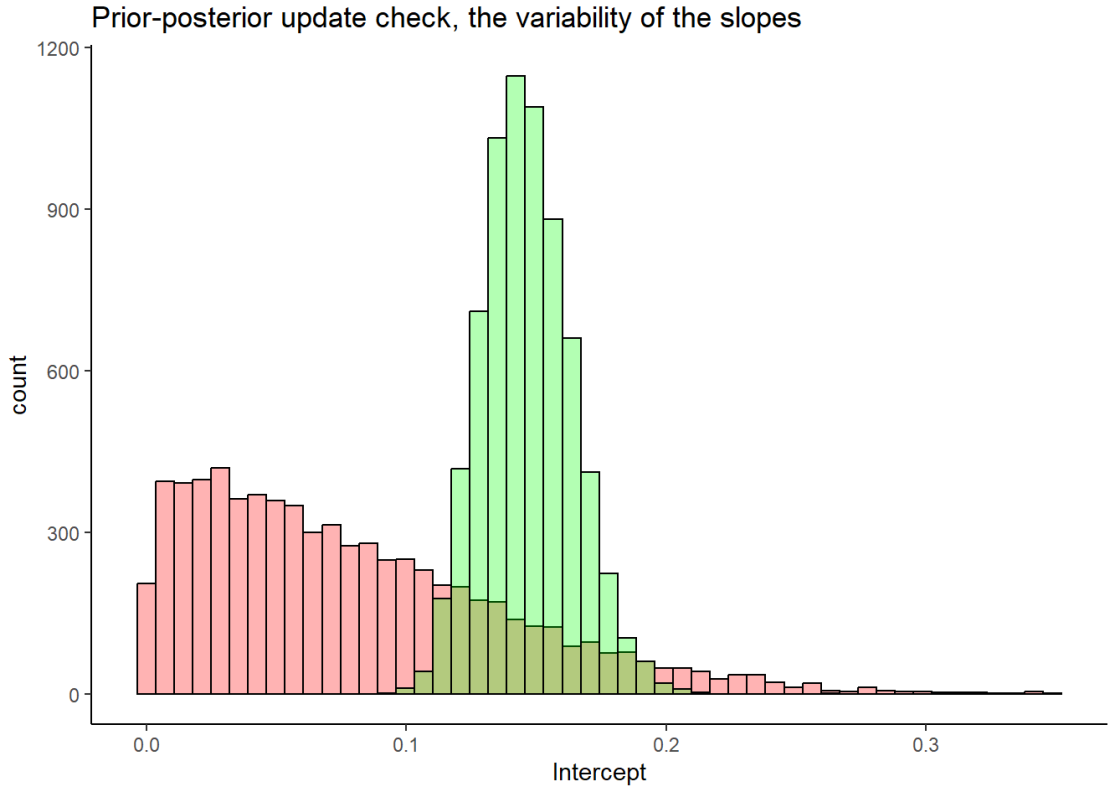
```
plot5
```


prior-posterior update check on correlation between varying intercepts and slopes
1250
1000
750
500
250
0
count
-1.0
-0.5
0.0
0.5
1.0
Correlation

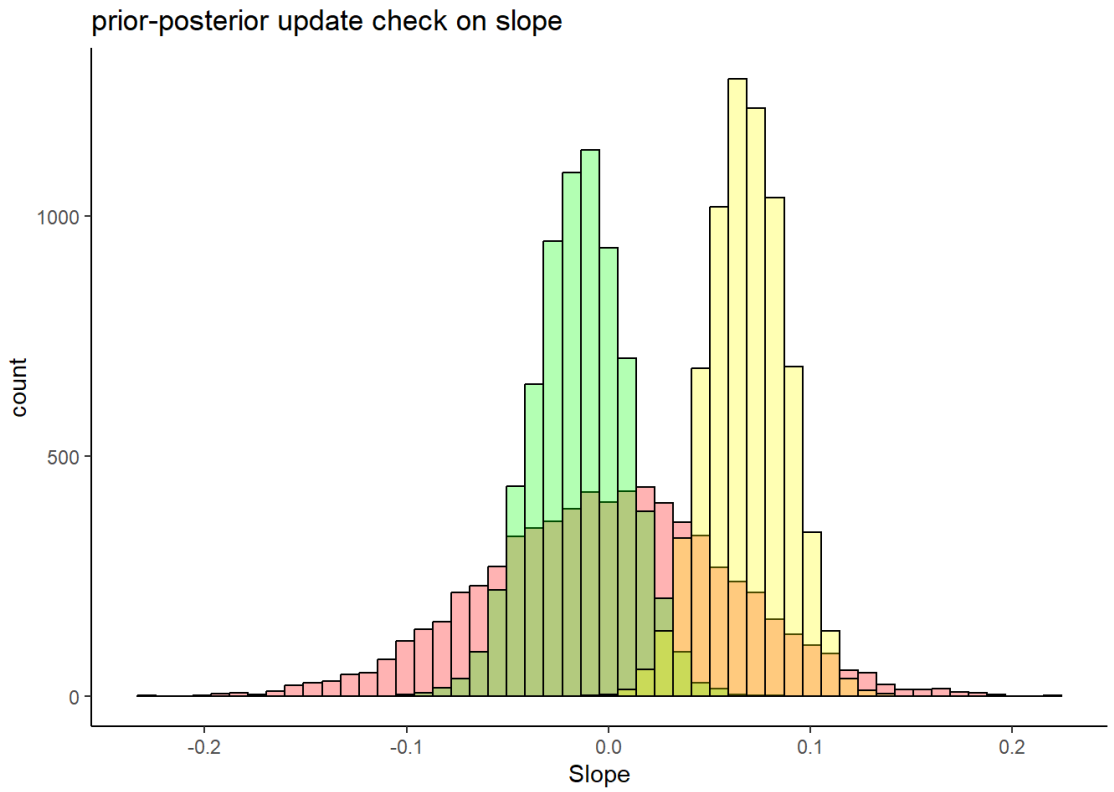
```
plot6
```


Prior-posterior update check, the variability of the intercept
800
600
400
200
0
count
0.0
0.2
0.4
0.6
Intercept

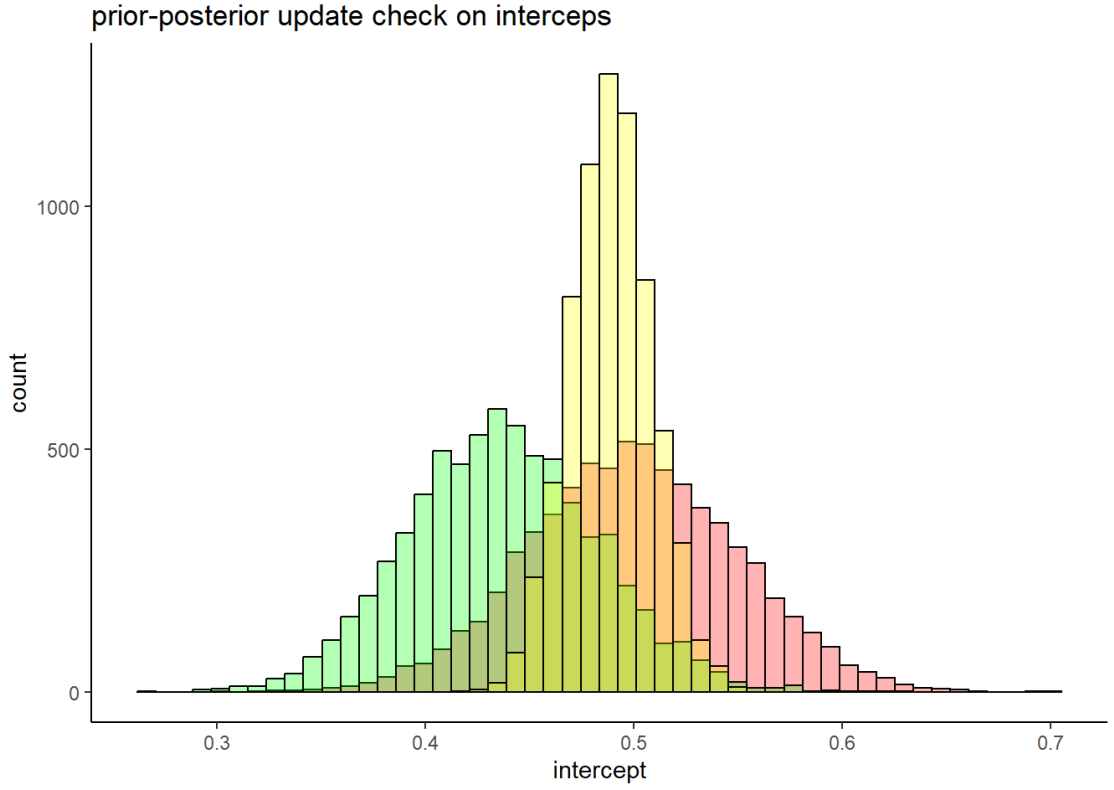
```
plot7
```


Prior-posterior update check, the variability of the slopes
1200
900
600
300
0
count
0.0
0.1
0.2
0.3
Intercept

```
plot8
```


prior-posterior update check on slope
count
1000
500
0
-0.2
-0.1
0.0
0.1
0.2
Slope

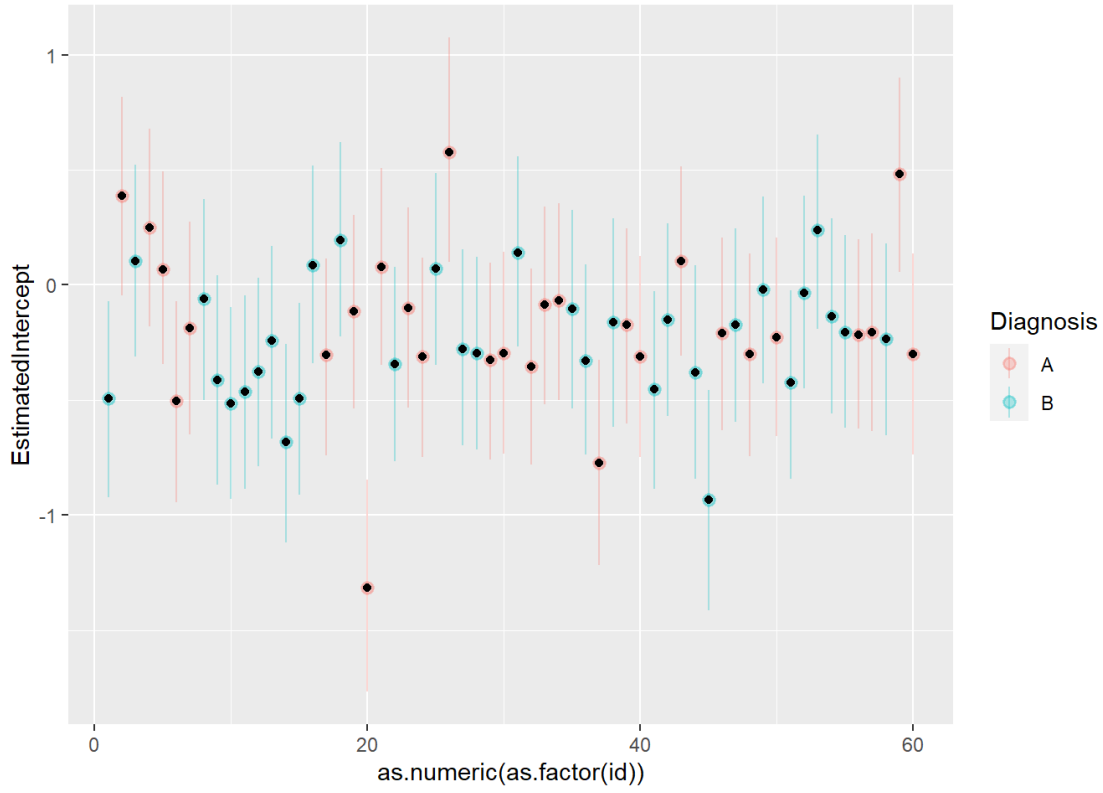
```
plot9
```


prior-posterior update check on interceps
count
1000
500
0
0.3
0.4
0.5
0.6
0.7
intercept

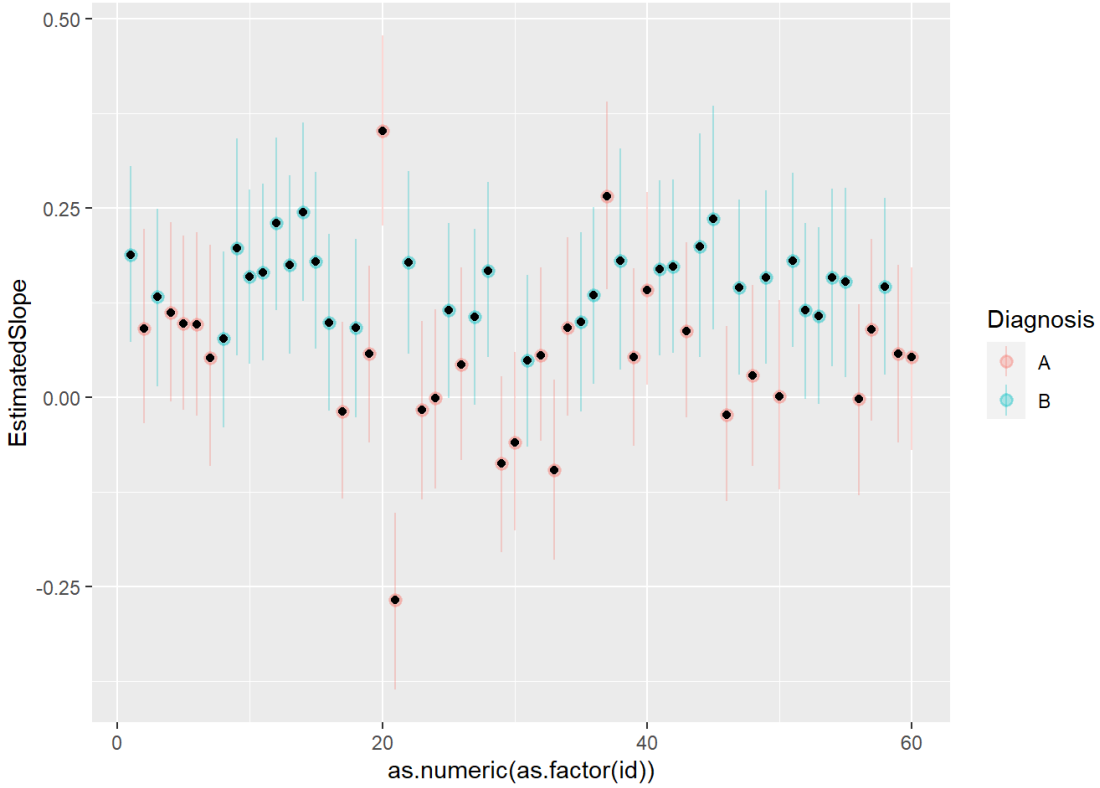
```
temp_re <- ranef(MLU_prior_fit)$id
for (i in unique(real_data$id)) {
  temp <- as.character(i)
  real_data$EstimatedIntercept[real_data$id == i] <-
temp_re[,,'Intercept'][temp,1]
  real_data$EstimatedIntercept_low[real_data$id == i]
<- temp_re[,,'Intercept'][temp,3]
  real_data$EstimatedIntercept_high[real_data$id == i]
<- temp_re[,,'Intercept'][temp,4]
  real_data$EstimatedSlope[real_data$id == i] <-
temp_re[,,'visit'][temp,1]
  real_data$EstimatedSlope_low[real_data$id == i] <-
temp_re[,,'visit'][temp,3]
  real_data$EstimatedSlope_high[real_data$id == i] <-
temp_re[,,'visit'][temp,4]
}
## Warning: Unknown or uninitialised column:
`EstimatedIntercept`.
## Warning: Unknown or uninitialised column:
`EstimatedIntercept_low`.
## Warning: Unknown or uninitialised column:
`EstimatedIntercept_high`.
## Warning: Unknown or uninitialised column:
`EstimatedSlope`.
## Warning: Unknown or uninitialised column:
`EstimatedSlope_low`.
## Warning: Unknown or uninitialised column:
`EstimatedSlope_high`.
d <- real_data %>% subset(visit == 1) %>%
  mutate(
    EstimatedIntercept = ifelse(Diagnosis == 'A',
                                EstimatedIntercept
+ 0.44,
                                EstimatedIntercept
+ 0.49),
```

```
    EstimatedIntercept_low = ifelse(Diagnosis == 'A',

EstimatedIntercept_low + 0.44,

EstimatedIntercept_low + 0.49),
    EstimatedIntercept_high = ifelse(Diagnosis == 'A',

EstimatedIntercept_high + 0.44,

EstimatedIntercept_high + 0.49)
)


d <- real_data %>% subset(visit == 1) %>%
  mutate(
    EstimatedSlope = ifelse(Diagnosis == 'A',
                                     EstimatedSlope -
0.01,
                                     EstimatedSlope +
0.07),
    EstimatedSlope_low = ifelse(Diagnosis == 'A',
                                    EstimatedSlope_low -
0.01,
                                    EstimatedSlope_low +
0.07),
    EstimatedSlope_high = ifelse(Diagnosis == 'A',

EstimatedSlope_high - 0.01,

EstimatedSlope_high + 0.07)
)

estimates_intercept <- ggplot(d) +
  geom_pointrange(aes( x = as.numeric(as.factor(id)), y
= EstimatedIntercept,
```

```
                        ymin = EstimatedIntercept_low,
ymax = EstimatedIntercept_high,
                        color = Diagnosis), alpha = 0.3)
+
  geom_point(aes( x = as.numeric(as.factor(id)), y =
EstimatedIntercept))

estimates_slopes <- ggplot(d) +
  geom_pointrange(aes( x = as.numeric(as.factor(id)), y
= EstimatedSlope,
                        ymin = EstimatedSlope_low, ymax
= EstimatedSlope_high,
                        color = Diagnosis), alpha = 0.3)
+
  geom_point(aes( x = as.numeric(as.factor(id)), y =
EstimatedSlope))

estimates_intercept
```



```
estimates_slopes
```



A_2.final

Study group 3

2022-11-04

Loading required packages

```
pacman::p_load(glue,
       tidyr,
       data.table,
       moments,
       tidybayes,
       tibble,
       cowplot,
       viridis,
       brms,
       stringr,
       rstan,
       cmdstanr,
       magrittr,
       gridExtra,
       grid,
       lattice,
       ggplot2,
       ggridges,
       ellipse,
       Rmisc,
       dplyr,
       "rmarkdown",
       knitr)
pacman::p_load(tidyverse)
```

```
## Error in completeSubclasses(classDef2, class1, obj, where) :
##   trying to get slot "subclasses" from an object of a basic class ("NULL") with
no slots
##
## The downloaded binary packages are in
##  /var/folders/hf/2cjx77xd24b01rtsfd8v4mmh0000gn/T//RtmpGE2FYm/downloaded_packag
es
## Error in completeSubclasses(classDef2, class1, obj, where) :
##   trying to get slot "subclasses" from an object of a basic class ("NULL") with
no slots
```

```
pacman::p_load(purrr)
pacman::p_load(MCMCglmm)
pacman::p_load(readxl)
pacman::p_load(metafor)
```

Assignment 2: meta-analysis

Questions to be answered

1. Simulate data to setup the analysis and gain insight on the structure of the problem. Simulate one dataset of 100 studies (n of participants should follow a normal distribution with mean of 20, sd of 10, but no fewer than 10 participants), with a mean effect size of 0.4, average deviation by study of .4 and measurement error of .8. The data you get should have one row per study, with an effect size mean and standard error. Build a proper bayesian model to analyze the simulated data. Then simulate publication bias (only some of the studies you simulate are likely to be published, which?), the effect of publication bias on your estimates (re-run the model on published studies, assess the difference), and use at least one technique to assess publication bias. remember to use at least one plot to visualize your results. BONUS question: do a power/precision analysis.
2. What is the current evidence for distinctive vocal patterns in schizophrenia? Use the data from Parola et al (2020) - https://www.dropbox.com/s/0l9ur0gaabr80a8/Matrix_MetaAnalysis_Diagnosis_updated290719.xlsx?dl=0 (https://www.dropbox.com/s/0l9ur0gaabr80a8/Matrix_MetaAnalysis_Diagnosis_updated290719.xlsx?dl=0) - focusing on pitch variability (PITCH_F0SD). Describe the data available (studies, participants). Using the model from question 1 analyze the data, visualize and report the findings: population level effect size; how well studies reflect it; influential studies, publication bias. BONUS question: add the findings from https://www.medrxiv.org/content/10.1101/2022.04.03.22273354v2 (https://www.medrxiv.org/content/10.1101/2022.04.03.22273354v2). BONUS question: assess the effect of task on the estimates (model comparison with baseline model)

Sara

Question 1

Simulation

Outlining prior parameter provided by the assignment description

```
mean_effect <- 0.4
effect_sd <- 0.4
meas_error <- 0.8
par_mean <- 20
par_sd <- 10
n <- 100
```

A simulation of participant data of multiple visits using the provided data

```
set.seed(954)

sim_studies <-
  tibble(
    study_ID = seq(1:n),
    n_participants =
        round(rtnorm(n, mean=par_mean, sd=par_sd, lower=10))
    )


for (i in seq(nrow(sim_studies))){
    sim_studies$study_effect[i] <-
      rnorm(1,mean_effect,effect_sd)
    temp <-
      rnorm(sim_studies$n_participants[i],sim_studies$study_effect[i], meas_error)
    sim_studies$mean_effect_size[i] <-
      mean(temp)
    sim_studies$sd_effect[i] <-
      sd(temp)
    sim_studies$standard_error[i] <-
      sim_studies$sd_effect[i]/sqrt(sim_studies$n_participants[i])
  }
```

Bayesian model

A Bayesian model illustrating potential effect sizes on individual participants

```
model_study <- bf(mean_effect_size|se(standard_error) ~1 + (1|study_ID))
```

Priors

Generating prior data simulations to model, using parameters provided in class

```
get_prior(data = sim_studies, family = gaussian, model_study)
```

```
##                   prior     class      coef    group resp dpar nlpar lb ub
##  student_t(3, 0.3, 2.5) Intercept
##    student_t(3, 0, 2.5)        sd                                    0
##    student_t(3, 0, 2.5)        sd           study_ID                 0
##    student_t(3, 0, 2.5)        sd Intercept study_ID                 0
##        source
##       default
##       default
##  (vectorized)
##  (vectorized)
```

```
priors <- c(
  prior(normal(0, 0.3), class=Intercept),
  prior(normal(0, 0.3), class=sd))
```

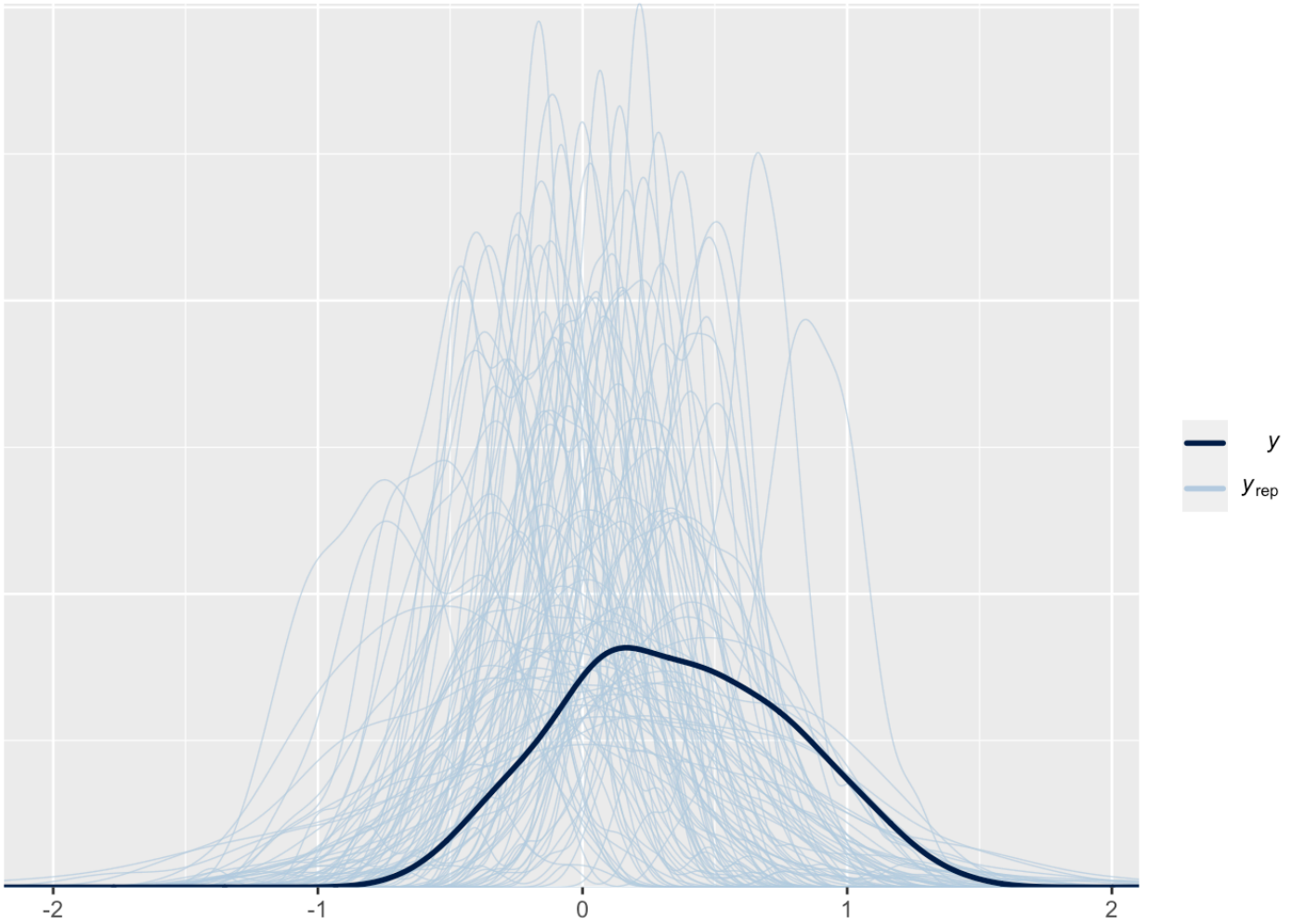
Model

Modeling using sample_prior = 'only'

```
model_prior <- brm(
  model_study,
  data = sim_studies,
  prior = priors,
  family = gaussian,
  refresh=0,
  sample_prior = 'only',
  iter=10000,
  warmup = 1000,
  backend = "cmdstanr",
  threads = threading(2),
  chains = 2,
  cores = 2,
  control = list(
    adapt_delta = 0.99,
    max_treedepth = 20
)
)
```

```
## Running MCMC with 2 parallel chains, with 2 thread(s) per chain...
##
## Chain 1 finished in 1.6 seconds.
## Chain 2 finished in 1.6 seconds.
##
## Both chains finished successfully.
## Mean chain execution time: 1.6 seconds.
## Total execution time: 1.6 seconds.
```

```
pp_check(model_prior, ndraws=100)
```



Manuela

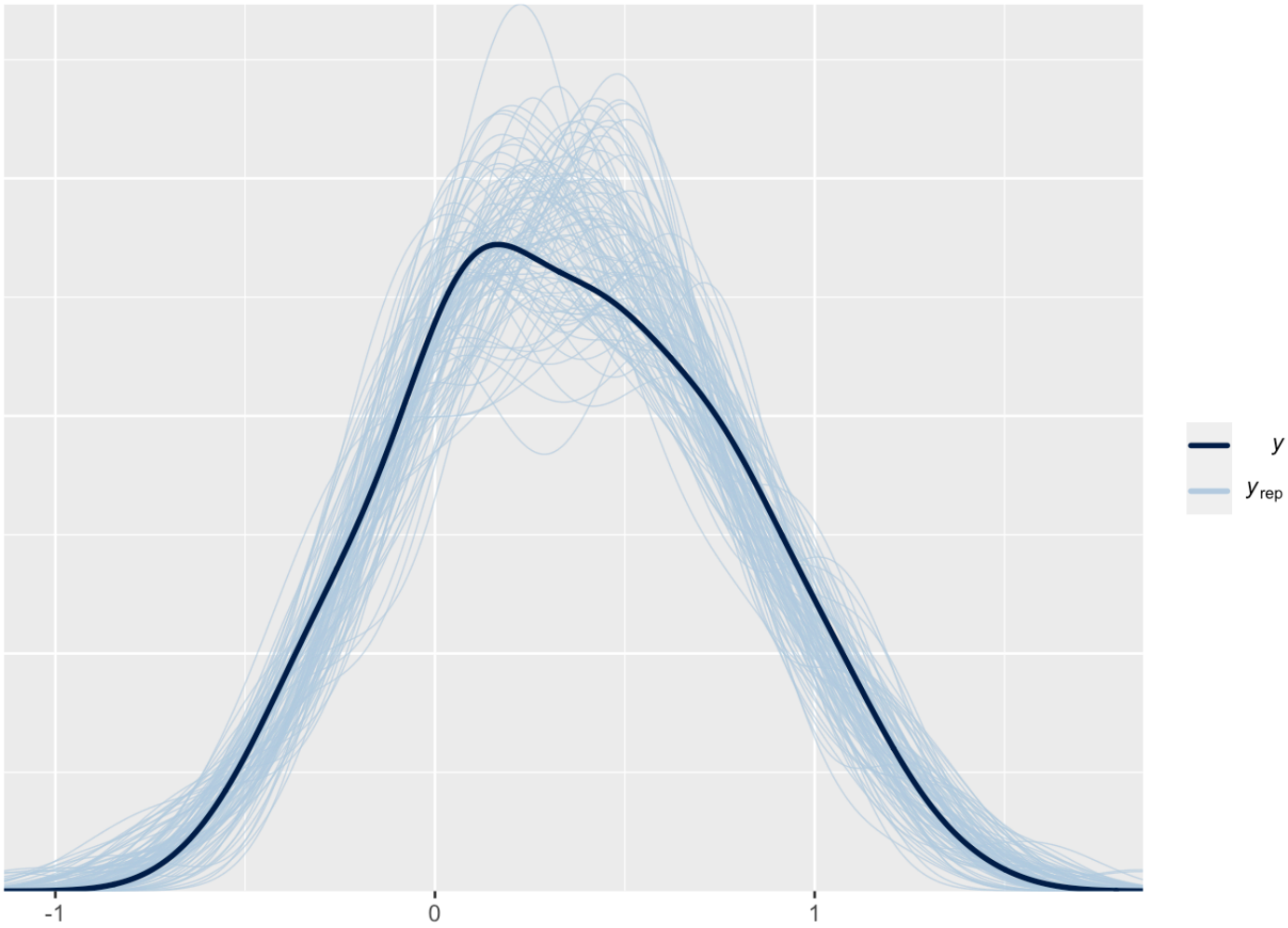
Fitting model

Modeling the sampled priors along with the simulation

```
model_prior_fit <- brm(
  model_study,
  data = sim_studies,
  prior = priors,
  family = gaussian,
  refresh=0,
  sample_prior = TRUE,
  iter=10000,
  warmup = 1000,
  backend = "cmdstanr",
  threads = threading(2),
  chains = 2,
  cores = 2,
  control = list(
    adapt_delta = 0.99,
    max_treedepth = 20
)
)
```

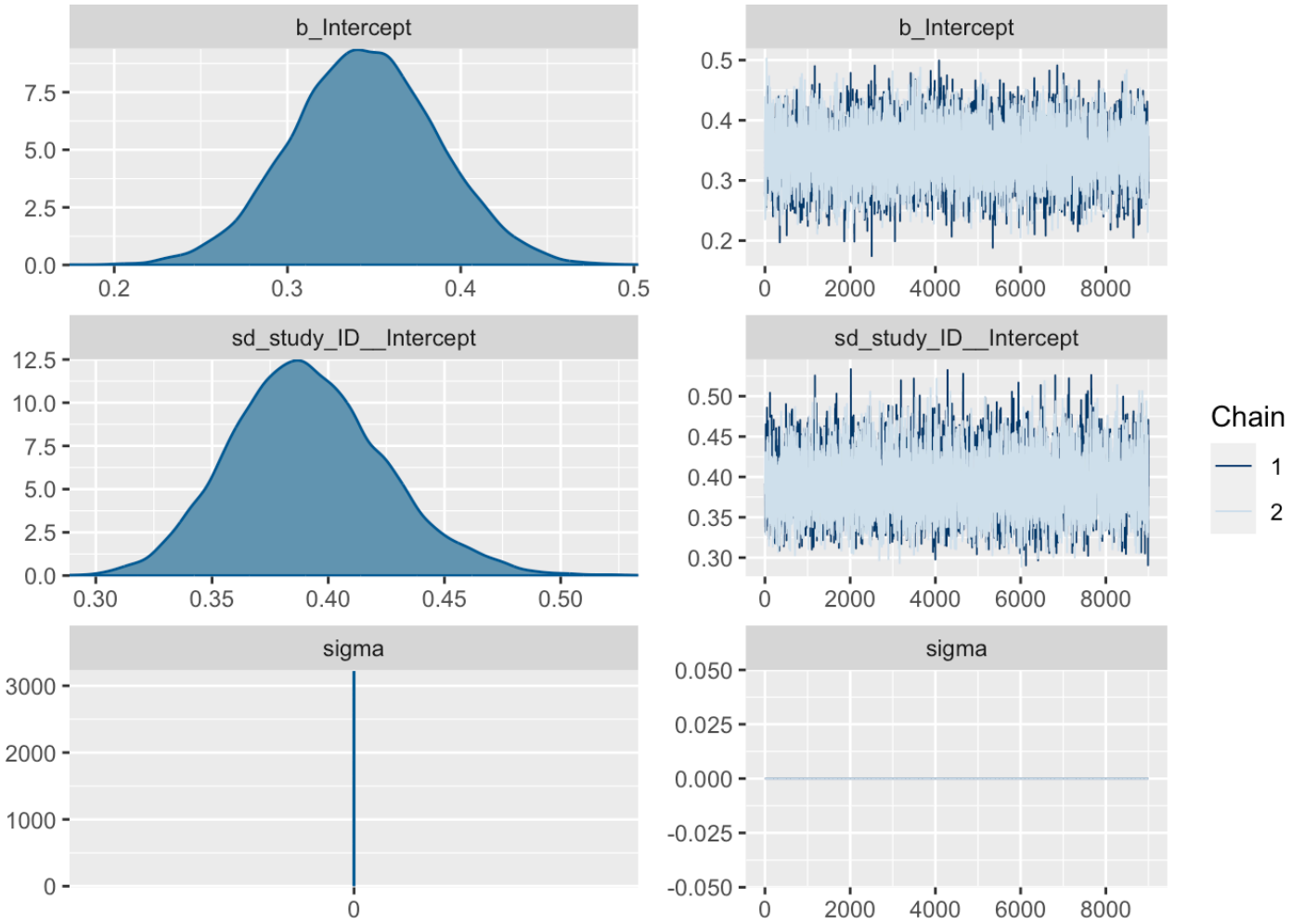
```
## Running MCMC with 2 parallel chains, with 2 thread(s) per chain...
##
## Chain 1 finished in 3.0 seconds.
## Chain 2 finished in 4.7 seconds.
##
## Both chains finished successfully.
## Mean chain execution time: 3.9 seconds.
## Total execution time: 4.7 seconds.
```

```
pp_check(model_prior_fit, ndraws=100)
```



Plotting and visualizing

```
plot(model_prior_fit)
```



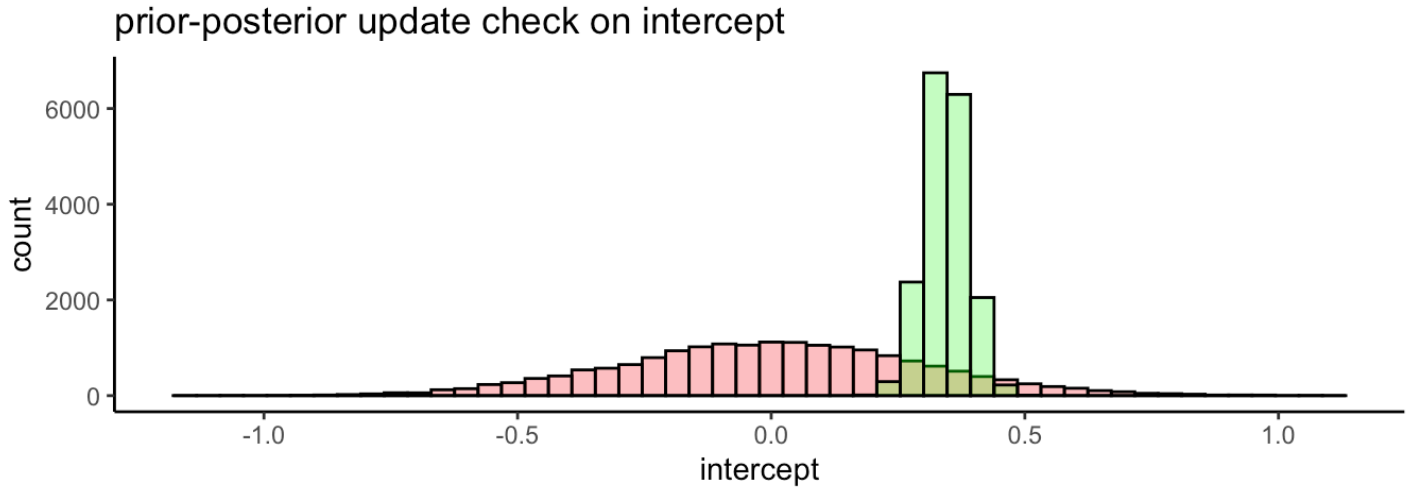

```
summary(model_prior_fit)
```

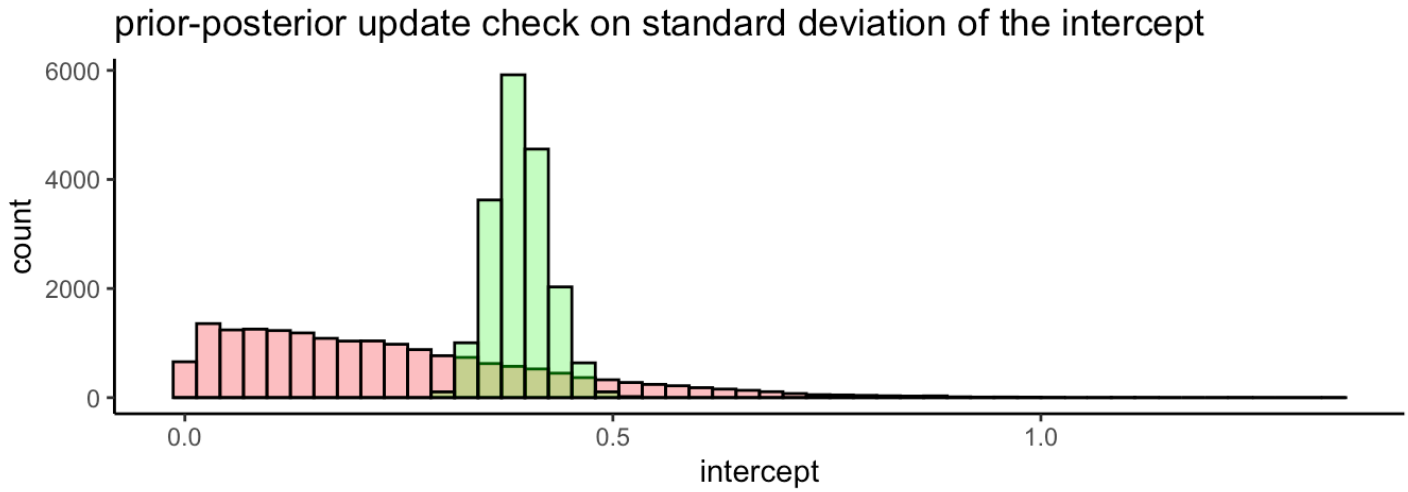
```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: mean_effect_size | se(standard_error) ~ 1 + (1 | study_ID)
##    Data: sim_studies (Number of observations: 100)
##   Draws: 2 chains, each with iter = 10000; warmup = 1000; thin = 1;
##          total post-warmup draws = 18000
##
## Group-Level Effects:
## ~study_ID (Number of levels: 100)
##               Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(Intercept)     0.39      0.03     0.33     0.46 1.00     5268     8430
##
## Population-Level Effects:
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept     0.34      0.04     0.26     0.43 1.00     4104     7005
##
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma     0.00      0.00     0.00     0.00   NA       NA       NA
##
## Draws were sampled using sample(hmc). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

Prior posterior update check

Plotting “prior-posterior update check on intercept” and “prior-posterior update check on standard deviation of the intercept”

```
model_posterior <- as_draws_df(model_prior_fit)

plot1 <- ggplot(model_posterior)+
  geom_histogram(aes(prior_Intercept), fill='red', color='black', alpha=0.3, bins=
50)+
  geom_histogram(aes(Intercept), fill='green', color='black', alpha=0.3, bins=50)+
  theme_classic()+
  ggtitle('prior-posterior update check on intercept')+
  xlab('intercept')
plot2 <- ggplot(model_posterior)+
  geom_histogram(aes(prior_sd_study_ID), fill='red', color='black', alpha=0.3, bin
s=50)+
  geom_histogram(aes(sd_study_ID__Intercept), fill='green', color='black', alpha=0
.3, bins=50)+
  theme_classic()+
  ggtitle('prior-posterior update check on standard deviation of the intercept')+
  xlab('intercept')
grid.arrange(plot1, plot2)
```





Simulation of publication bias, the effect of publication bias on our estimate and asses the publication bias (remember to visualize our results)

Simulating the effect size of the publication factors and filtering data for only published studies

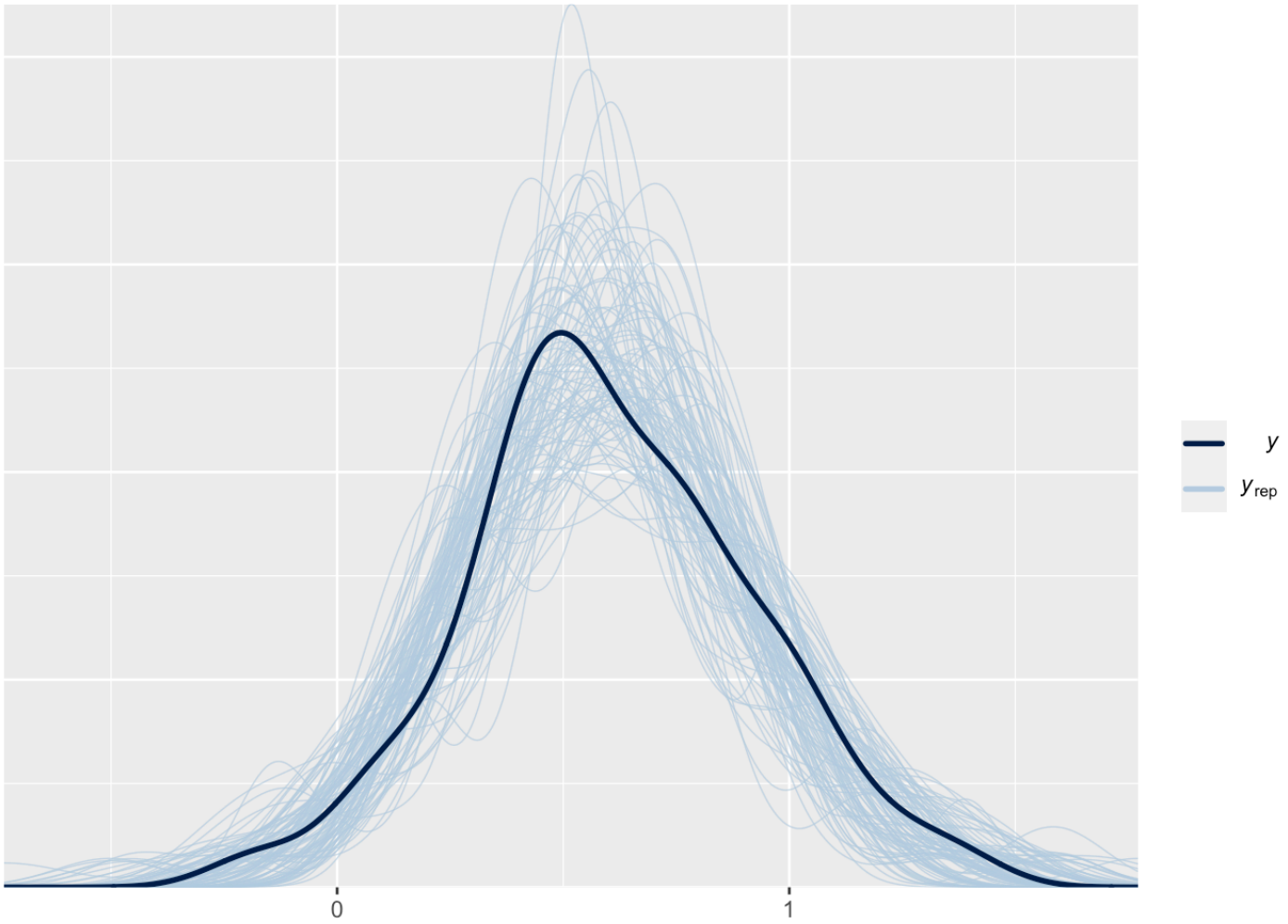
```
set.seed(843)

for (i in seq(nrow(sim_studies))){
  sim_studies$published[i] <-
    ifelse(abs(
      sim_studies$mean_effect_size[i])-(2*sim_studies$standard_error[i])>0
      & sim_studies$mean_effect_size[i]>0,
      rbinom(1,1,0.9), rbinom(1,1,0.1))}

sim_studies <- sim_studies %>%
  mutate(published=as.factor(published))

pub_sim_studies <- dplyr::filter(sim_studies, published==1)
```

Modeling using sample_prior = ‘only’

```
pub_model_prior_fit <- brm(
  model_study,
  data = pub_sim_studies,
  prior = priors,
  family = gaussian,
  refresh=0,
  sample_prior = TRUE,
  iter=10000,
  warmup = 1000,
  backend = "cmdstanr",
  threads = threading(2),
  chains = 2,
  cores = 2,
  control = list(
    adapt_delta = 0.99,
    max_treedepth = 20
)
)
```

```
## Running MCMC with 2 parallel chains, with 2 thread(s) per chain...
##
## Chain 2 finished in 2.1 seconds.
## Chain 1 finished in 2.1 seconds.
##
## Both chains finished successfully.
## Mean chain execution time: 2.1 seconds.
## Total execution time: 2.2 seconds.
```

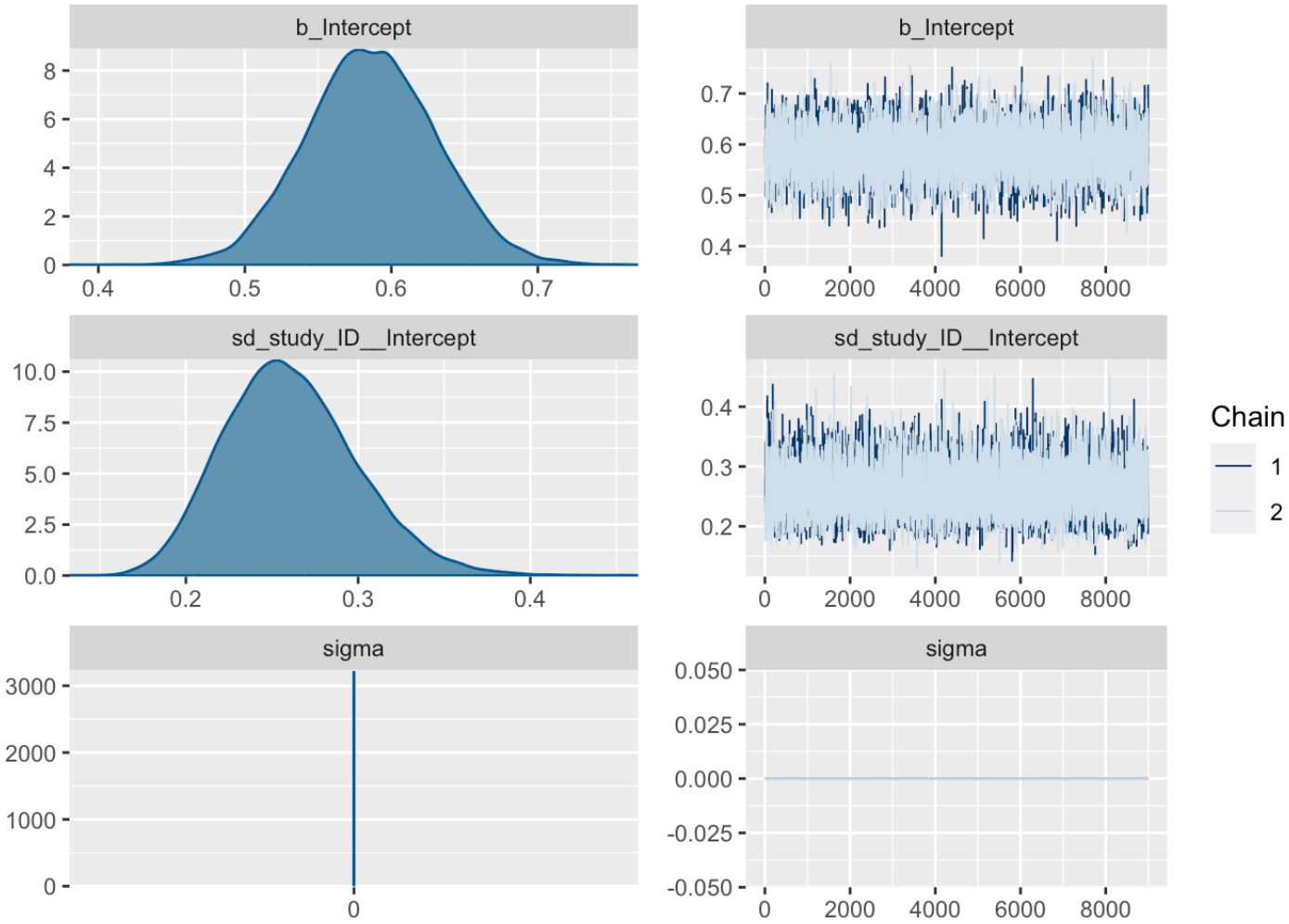
```
pp_check(pub_model_prior_fit, ndraws=100)
```



Ditlev

Potting and assesing and transforming the brmsfit to draws

```
plot(pub_model_prior_fit)
```



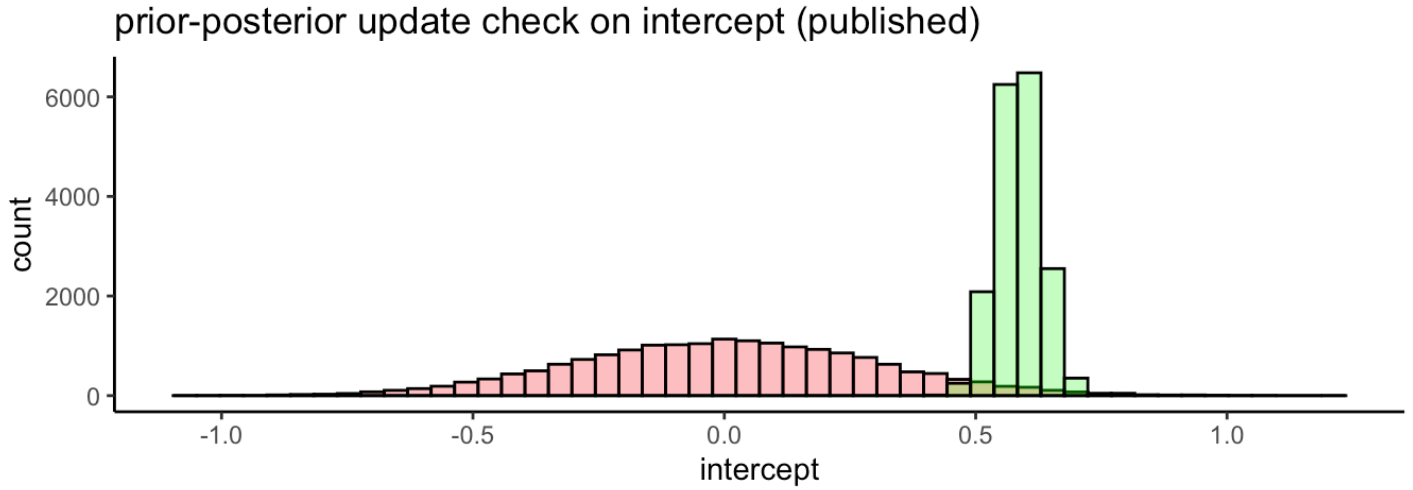

```
summary(pub_model_prior_fit)
```

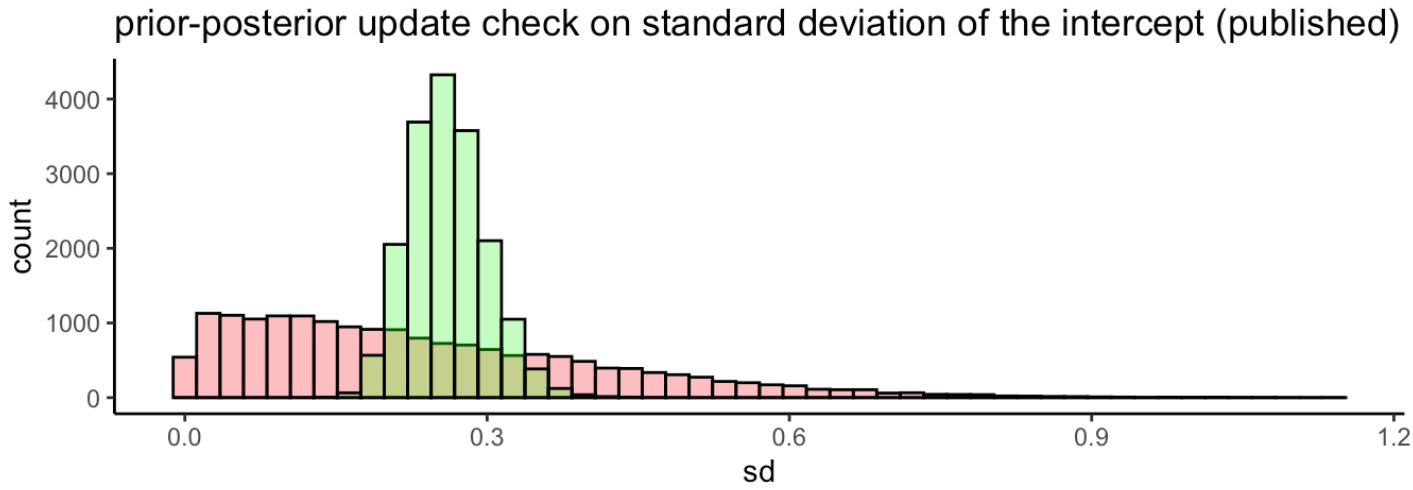
```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: mean_effect_size | se(standard_error) ~ 1 + (1 | study_ID)
##    Data: pub_sim_studies (Number of observations: 48)
##   Draws: 2 chains, each with iter = 10000; warmup = 1000; thin = 1;
##          total post-warmup draws = 18000
##
## Group-Level Effects:
## ~study_ID (Number of levels: 48)
##               Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(Intercept)     0.26      0.04     0.19     0.34 1.00     5900     9863
##
## Population-Level Effects:
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept     0.59      0.04     0.50     0.67 1.00     4538     8095
##
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma     0.00      0.00     0.00     0.00   NA       NA       NA
##
## Draws were sampled using sample(hmc). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

```
pub_posterior <- as_draws_df(pub_model_prior_fit)
```

Plotting “prior-posterior update check on intercept” and “prior-posterior update check on standard deviation of the intercept”

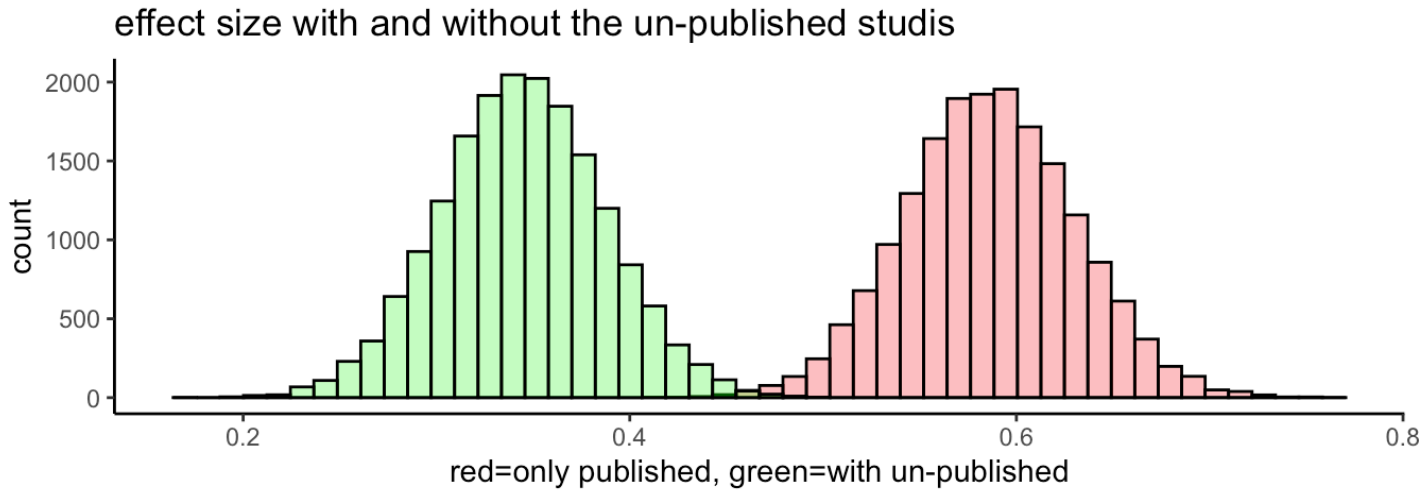
```
pub_plot1 <- ggplot(pub_posterior)+
  geom_histogram(aes(prior_Intercept), fill='red', color='black', alpha=0.3, bins=
50)+
  geom_histogram(aes(Intercept), fill='green', color='black', alpha=0.3, bins=50)+
  theme_classic()+
  ggtitle('prior-posterior update check on intercept (published)')+
  xlab('intercept')
pub_plot2 <- ggplot(pub_posterior)+
  geom_histogram(aes(prior_sd_study_ID), fill='red', color='black', alpha=0.3, bin
s=50)+
  geom_histogram(aes(sd_study_ID__Intercept), fill='green', color='black', alpha=0
.3, bins=50)+
  theme_classic()+
  ggtitle('prior-posterior update check on standard deviation of the intercept (pu
blished)')+
  xlab('sd')
grid.arrange(pub_plot1, pub_plot2)
```

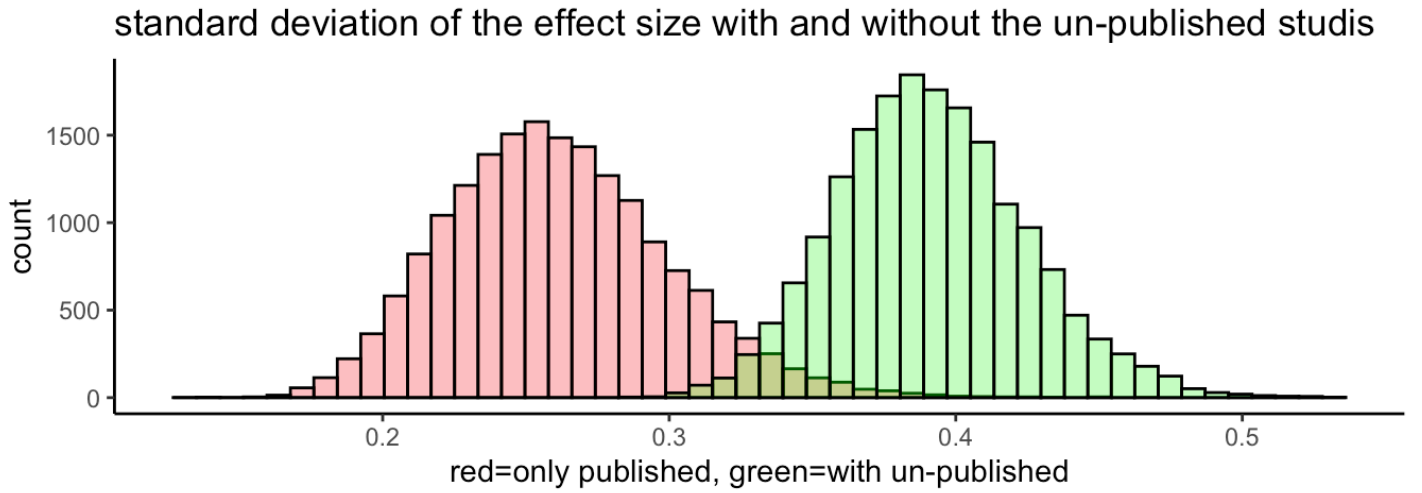




Plotting “effect size with and without the un-published studis” and “tandard deviation of the effect size with and without the un-published studis”

```
plot3 <- ggplot()+
  geom_histogram(aes(pub_posterior$Intercept), fill='red', color='black', alpha=0.
3, bins=50)+
  geom_histogram(aes(model_posterior$Intercept), fill='green', color='black', alph
a=0.3, bins=50)+
  theme_classic()+
  ggtitle('effect size with and without the un-published studis')+
  xlab('red=only published, green=with un-published')
plot4 <- ggplot()+
  geom_histogram(aes(pub_posterior$sd_study_ID__Intercept), fill='red', color='bla
ck', alpha=0.3, bins=50)+
  geom_histogram(aes(model_posterior$sd_study_ID__Intercept), fill='green', color=
'black', alpha=0.3, bins=50)+
  theme_classic()+
  ggtitle('standard deviation of the effect size with and without the un-published
studis')+
  xlab('red=only published, green=with un-published')
grid.arrange(plot3, plot4)
```





Question 2

Loading the data

```
matrix_ma <- read_excel("/Users/patrikmolnar/Desktop/Cognitive Science/Semester 3/
Methods3/Matrix_MetaAnalysis.xlsx")
```

Describing the data

Filtering out NA & NR for SZ

```
matrix_ma_filter_for_analysis <- matrix_ma %>%
  dplyr::filter(AGE_M_SZ!="NR") %>%
  dplyr::filter(AGE_M_SZ!="NA")
matrix_ma_filter_for_analysis <- matrix_ma_filter_for_analysis %>%
  dplyr::filter(AGE_SD_SZ!="NR") %>%
  dplyr::filter(AGE_SD_SZ!="NA")
matrix_ma_filter_for_analysis <- matrix_ma_filter_for_analysis %>%
  dplyr::filter(MALE_SZ!="NR") %>%
  dplyr::filter(MALE_SZ!="NA")
matrix_ma_filter_for_analysis <- matrix_ma_filter_for_analysis %>%
  dplyr::filter(FEMALE_SZ!="NR") %>%
  dplyr::filter(FEMALE_SZ!="NA")
```

Filtering out NA & NR for HC

```
matrix_ma_filter_for_analysis <- matrix_ma_filter_for_analysis %>%
  dplyr::filter(AGE_M_HC!="NR") %>%
  dplyr::filter(AGE_M_HC!="NA")
matrix_ma_filter_for_analysis <- matrix_ma_filter_for_analysis %>%
  dplyr::filter(AGE_SD_HC!="NR") %>%
  dplyr::filter(AGE_SD_HC!="NA")
matrix_ma_filter_for_analysis <- matrix_ma_filter_for_analysis %>%
  dplyr::filter(MALE_HC!="NR") %>%
  dplyr::filter(MALE_HC!="NA")
matrix_ma_filter_for_analysis <- matrix_ma_filter_for_analysis %>%
  dplyr::filter(FEMALE_HC!="NR") %>%
  dplyr::filter(FEMALE_HC!="NA")
```

Making the variables numeric for SZ

```
matrix_ma_filter_for_analysis$AGE_M_SZ <- as.numeric(matrix_ma_filter_for_analysis
$AGE_M_SZ)
matrix_ma_filter_for_analysis$AGE_SD_SZ <- as.numeric(matrix_ma_filter_for_analysi
s$AGE_SD_SZ)
matrix_ma_filter_for_analysis$MALE_SZ <- as.numeric(matrix_ma_filter_for_analysis$
MALE_SZ)
matrix_ma_filter_for_analysis$FEMALE_SZ <- as.numeric(matrix_ma_filter_for_analysi
s$FEMALE_SZ)
```

Making the variables numeric for HC

```
matrix_ma_filter_for_analysis$AGE_M_HC <- as.numeric(matrix_ma_filter_for_analysis
$AGE_M_HC)
matrix_ma_filter_for_analysis$AGE_SD_HC <- as.numeric(matrix_ma_filter_for_analysi
s$AGE_SD_HC)
matrix_ma_filter_for_analysis$MALE_HC <- as.numeric(matrix_ma_filter_for_analysis$
MALE_HC)
matrix_ma_filter_for_analysis$FEMALE_HC <- as.numeric(matrix_ma_filter_for_analysi
s$FEMALE_HC)
```

Patrik

Making both tibbles in order to combine them and make them easier for the eye

```
a <- tibble(diagnosis = "SZ",
         mean_sample_size=mean(matrix_ma_filter_for_analysis$SAMPLE_SIZE_SZ),
         mean_numer_of_males=mean(matrix_ma_filter_for_analysis$MALE_SZ),
         mean_number_of_females=mean(matrix_ma_filter_for_analysis$FEMALE_SZ),
         mean_age=mean(matrix_ma_filter_for_analysis$AGE_M_SZ),
         mean_sd_age=mean(matrix_ma_filter_for_analysis$AGE_SD_SZ)
         )

b <- tibble(diagnosis = "HC",
         mean_sample_size=mean(matrix_ma_filter_for_analysis$SAMPLE_SIZE_HC),
         mean_numer_of_males=mean(matrix_ma_filter_for_analysis$MALE_HC),
         mean_number_of_females=mean(matrix_ma_filter_for_analysis$FEMALE_HC),
         mean_age=mean(matrix_ma_filter_for_analysis$AGE_M_HC),
         mean_sd_age=mean(matrix_ma_filter_for_analysis$AGE_SD_HC)
         )
```

Binding the rows together

```
Demographic_overview <- bind_rows(a,b)
```

Showing the tibble

```
Demographic_overview
```

```
## # A tibble: 2 × 6
##   diagnosis mean_sample_size mean_numer_of_males mean_number_o…¹ mean_…² mean_…
³
##   <chr>                <dbl>               <dbl>           <dbl>   <dbl>   <dbl
>
## 1 SZ                    40.5                27.3            14.3    35.9    8.3
9
## 2 HC                    31.2                17.7            14.7    34.9    8.9
2
## # … with abbreviated variable names ¹mean_number_of_females, ²mean_age,
## #   ³mean_sd_age
```

Selceting the relevant variables

```
matrix_pitch <- matrix_ma %>%
  select('StudyID','Article','SAMPLE_SIZE_SZ','SAMPLE_SIZE_HC', 'PITCH_F0SD_HC_M',
'PITCH_F0SD_HC_SD','PITCH_F0SD_SZ_M','PITCH_F0SD_SZ_SD')
```

Filtering out the NA

```
matrix_pitch <- matrix_pitch %>%
  na.omit()
```

Merging diagnosis into one variable

```
matrix_pitch <- matrix_pitch %>%
  mutate(sample_size=(SAMPLE_SIZE_SZ+SAMPLE_SIZE_HC))
```

Creating IDs for the studies

```
matrix_pitch <- matrix_pitch %>%
  mutate(StudyID=as.factor(StudyID))
matrix_pitch <- matrix_pitch %>%
  mutate(StudyID=as.numeric(StudyID))
matrix_pitch <- matrix_pitch %>%
  mutate(StudyID=as.factor(StudyID))
```

Getting normalized results

```
matrix_pitch <- escalc('SMD',
n1i=SAMPLE_SIZE_HC,
n2i=SAMPLE_SIZE_SZ,
m1i = PITCH_F0SD_HC_M,
m2i=PITCH_F0SD_SZ_M,
sd1i = PITCH_F0SD_HC_SD,
sd2i = PITCH_F0SD_SZ_SD,
data = matrix_pitch)
matrix_pitch <- matrix_pitch %>%
  rename(effect_size=yi)
```

Creating a loop to calculate sd effect size and se

```
for (i in seq(nrow(matrix_pitch))){
  matrix_pitch$sd_effect[i] <- sqrt((sum((matrix_pitch$effect_size[i] - mean(matri
x_pitch$effect_size))^2))/length(matrix_pitch))
  matrix_pitch$standard_error[i] <- matrix_pitch$sd_effect[i]/sqrt(matrix_pitch$sa
mple_size)
}
```

Setting model

```
model_matrix <- bf(effect_size|se(standard_error) ~1 + (1|StudyID))
```

Getting priors

```
get_prior(data = matrix_pitch, family = gaussian, model_matrix)
```

```
##                   prior     class      coef   group resp dpar nlpar lb ub
##  student_t(3, 0.3, 2.5) Intercept
##    student_t(3, 0, 2.5)        sd                                    0
##    student_t(3, 0, 2.5)        sd           StudyID                  0
##    student_t(3, 0, 2.5)        sd Intercept StudyID                  0
##        source
##       default
##       default
##  (vectorized)
##  (vectorized)
```

Setting priors

```
matrix_priors <- c(
  prior(normal( .3, 2.5), class=Intercept),
  prior(normal( 0, 2.5), class=sd))
```

Priors

```
matrix_prior_fit <- brm(
  model_matrix,
  data = matrix_pitch,
  prior = matrix_priors,
  family = gaussian,
  refresh=0,
  sample_prior = 'only',
  iter=10000,
  warmup = 1000,
  backend = "cmdstanr",
  threads = threading(2),
  chains = 2,
  cores = 2,
  control = list(
    adapt_delta = 0.99,
    max_treedepth = 20
)
)
```

```
## Running MCMC with 2 parallel chains, with 2 thread(s) per chain...
##
## Chain 2 finished in 0.4 seconds.
## Chain 1 finished in 0.5 seconds.
##
## Both chains finished successfully.
## Mean chain execution time: 0.4 seconds.
## Total execution time: 0.5 seconds.
```

Bryan

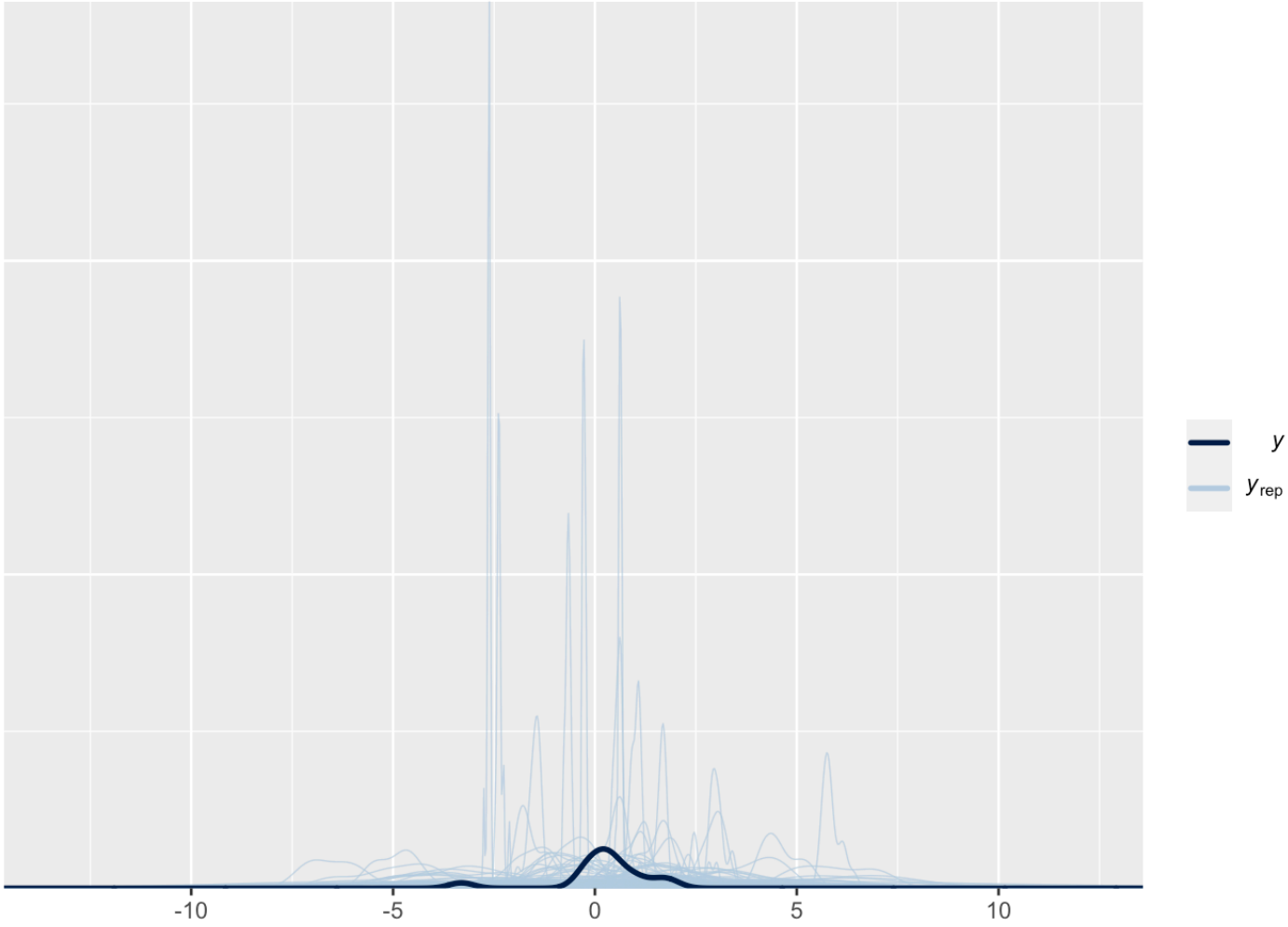
Assesing results

```
matrix_prior_fit
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: effect_size | se(standard_error) ~ 1 + (1 | StudyID)
##    Data: matrix_pitch (Number of observations: 15)
##   Draws: 2 chains, each with iter = 10000; warmup = 1000; thin = 1;
##          total post-warmup draws = 18000
##
## Group-Level Effects:
## ~StudyID (Number of levels: 12)
##               Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(Intercept)     1.98      1.52     0.08     5.62 1.00    14754     8411
##
## Population-Level Effects:
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept     0.29      2.51    -4.62     5.09 1.00    26582    13577
##
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma     0.00      0.00     0.00     0.00   NA       NA       NA
##
## Draws were sampled using sample(hmc). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

PP check

```
pp_check(matrix_prior_fit, ndraws=100)
```



Both data and priors

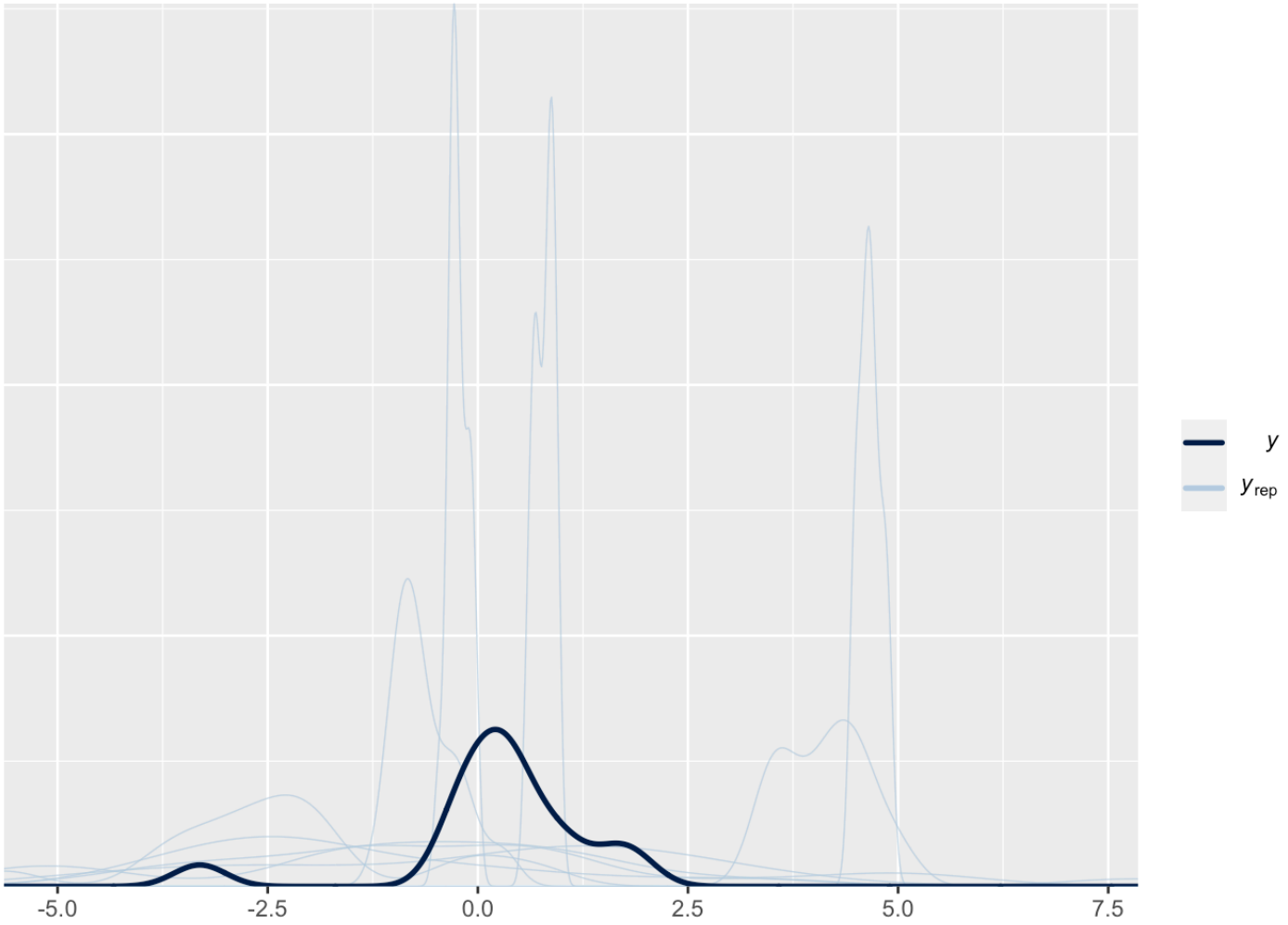
Including both data and priors

```
matrix_fit <- brm(
  model_matrix,
  data = matrix_pitch,
  prior = matrix_priors,
  family = gaussian,
  refresh=0,
  sample_prior = 'only',
  iter=10000,
  warmup = 1000,
  backend = "cmdstanr",
  threads = threading(2),
  chains = 2,
  cores = 2,
  control = list(
    adapt_delta = 0.99,
    max_treedepth = 20
)
)
```

```
## Running MCMC with 2 parallel chains, with 2 thread(s) per chain...
##
## Chain 1 finished in 0.4 seconds.
## Chain 2 finished in 0.6 seconds.
##
## Both chains finished successfully.
## Mean chain execution time: 0.5 seconds.
## Total execution time: 0.7 seconds.
```

PP check

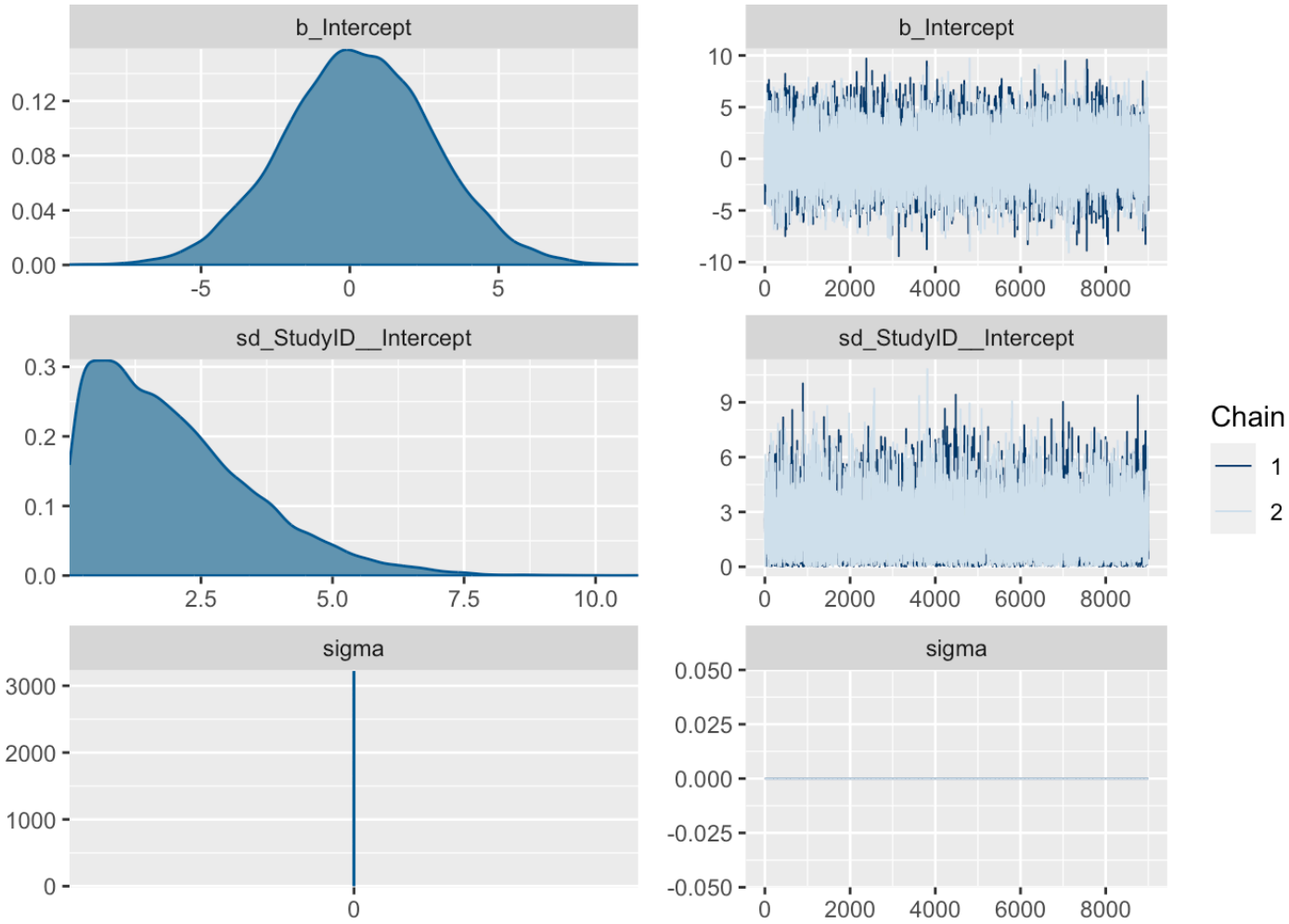
```
pp_check(matrix_fit)
```



Visualize and report

Plotting and assesing the results

```
plot(matrix_fit)
```




```
summary(matrix_fit)
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: effect_size | se(standard_error) ~ 1 + (1 | StudyID)
##    Data: matrix_pitch (Number of observations: 15)
##   Draws: 2 chains, each with iter = 10000; warmup = 1000; thin = 1;
##          total post-warmup draws = 18000
##
## Group-Level Effects:
## ~StudyID (Number of levels: 12)
##               Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(Intercept)     1.99      1.51     0.08     5.65 1.00    13897     7399
##
## Population-Level Effects:
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept     0.29      2.52    -4.62     5.20 1.00    24126    12836
##
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma     0.00      0.00     0.00     0.00   NA       NA       NA
##
## Draws were sampled using sample(hmc). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

Visualizing and assesing intercepts

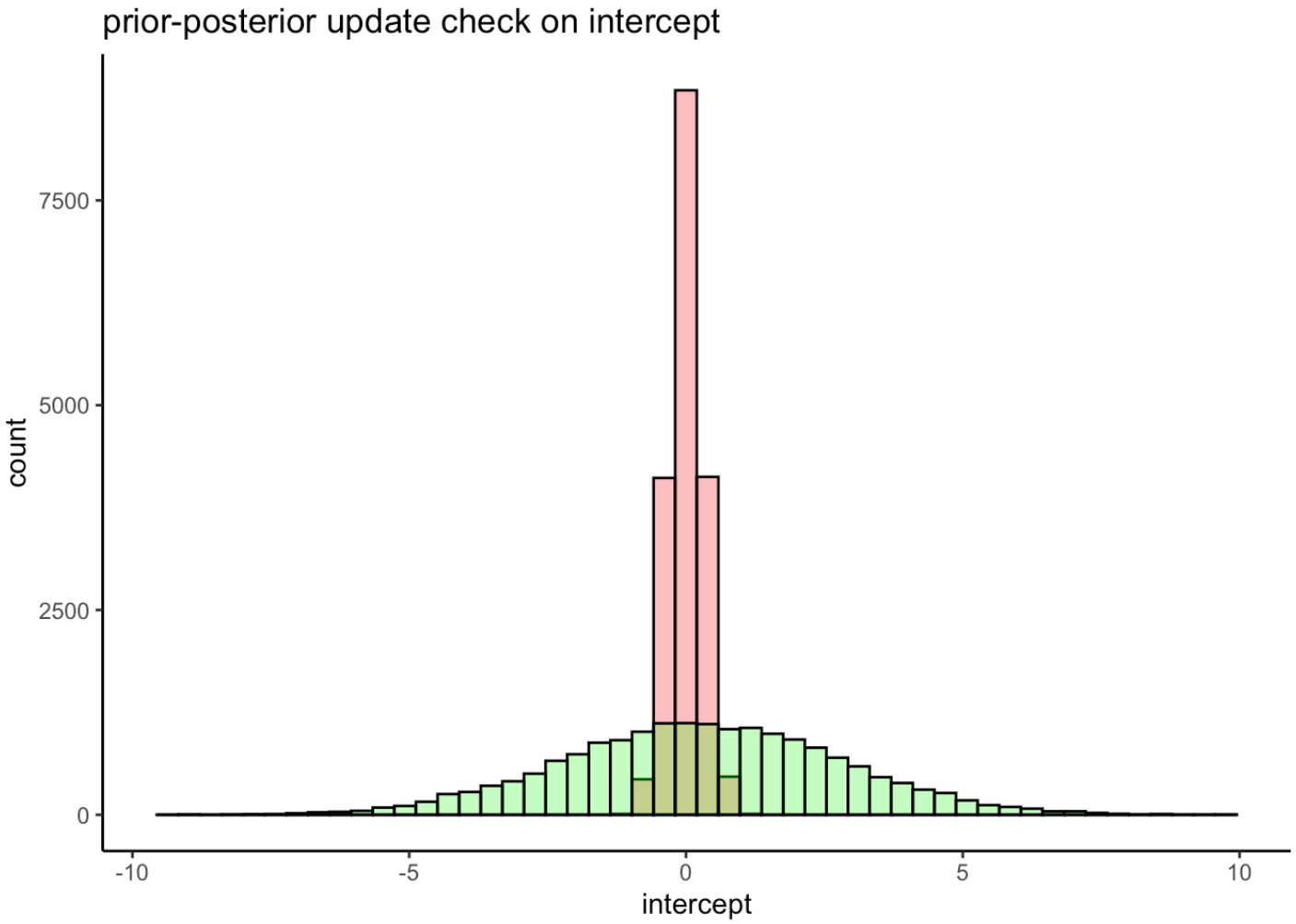
```
matrix_posterior <- as_draws_df(matrix_fit)
plot1 <- ggplot(matrix_posterior)+
  geom_histogram(aes(model_posterior$prior_Intercept), fill='red', color='black',
alpha=0.3, bins=50)+
  geom_histogram(aes(Intercept), fill='green', color='black', alpha=0.3, bins=50)+
  theme_classic()+
  ggtitle('prior-posterior update check on intercept')+
  xlab('intercept')
```

Visualizing standard deviation

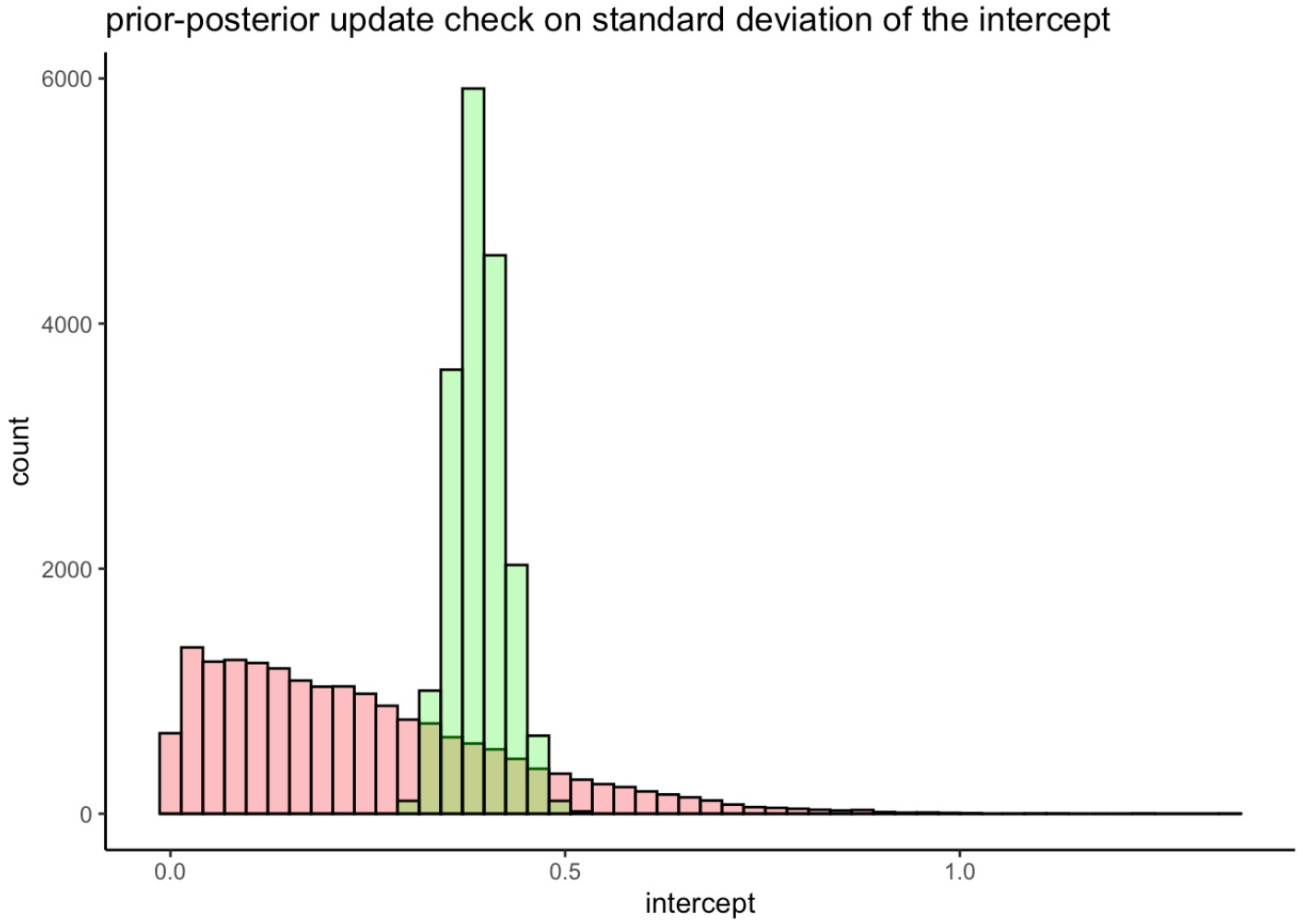
```
plot2 <- ggplot(matrix_posterior)+
  geom_histogram(aes(model_posterior$prior_sd_study_ID), fill='red', color='black'
, alpha=0.3, bins=50)+
  geom_histogram(aes(model_posterior$sd_study_ID__Intercept), fill='green', color=
'black', alpha=0.3, bins=50)+
  theme_classic()+
  ggtitle('prior-posterior update check on standard deviation of the intercept')+
  xlab('intercept')
```

Printing plots

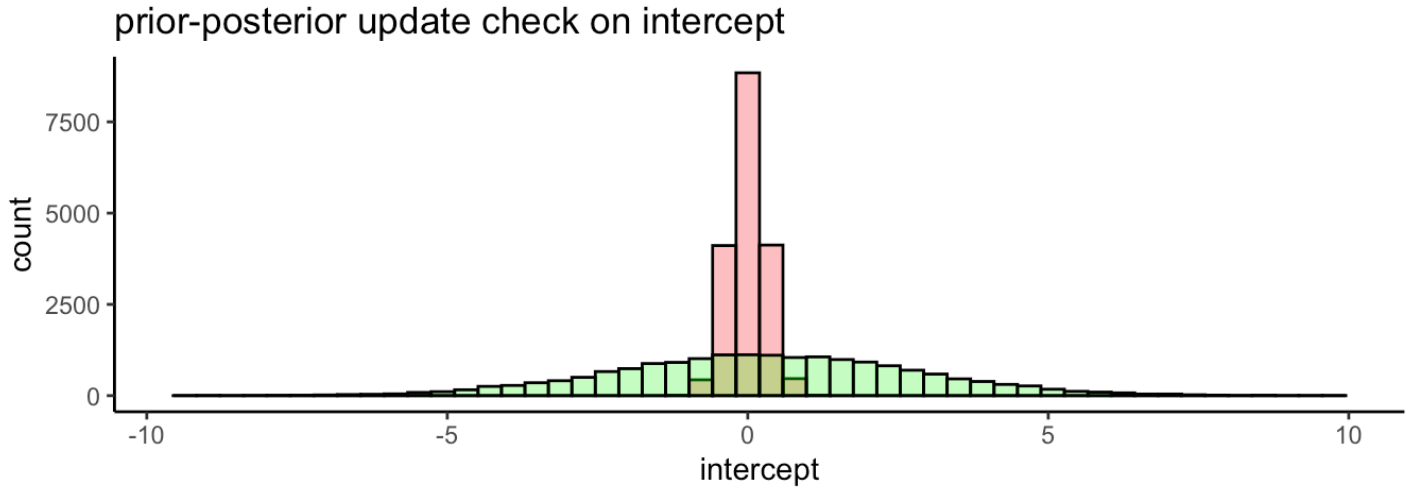
```
plot1
```

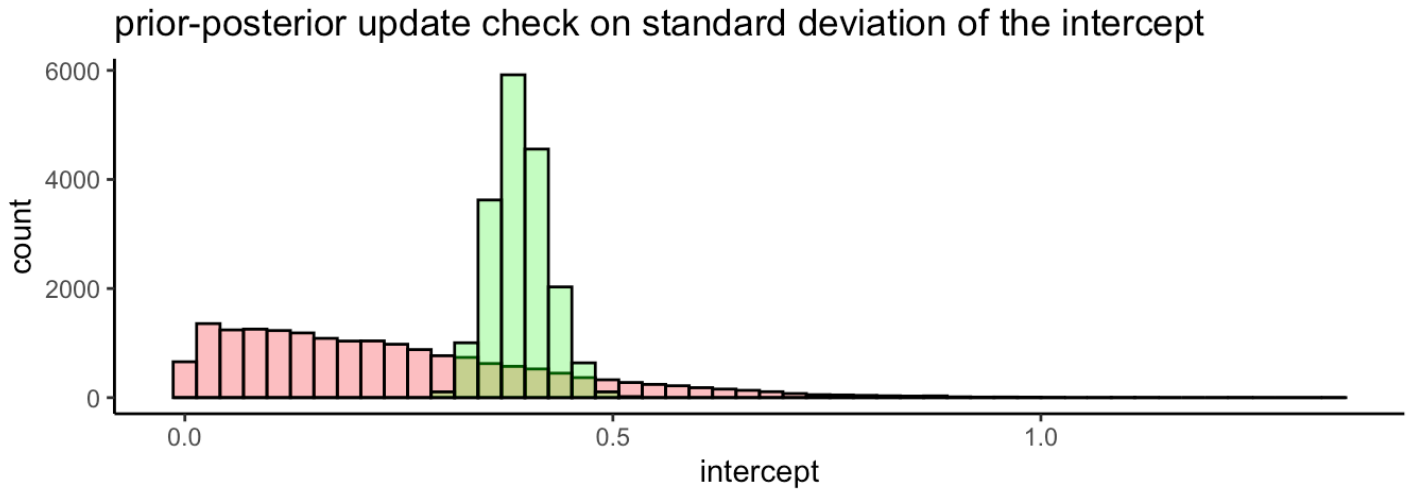

prior-posterior update check on intercept
7500
5000
2500
0
count
-10
-5
0
5
10
intercept

```
plot2
```




```
grid.arrange(plot1, plot2)
```





Influencial studies

Excluding the Cohen et al. (2014) by indexing

```
excluded_matrix <- matrix_pitch %>%
  dplyr::filter(StudyID!=6)
```

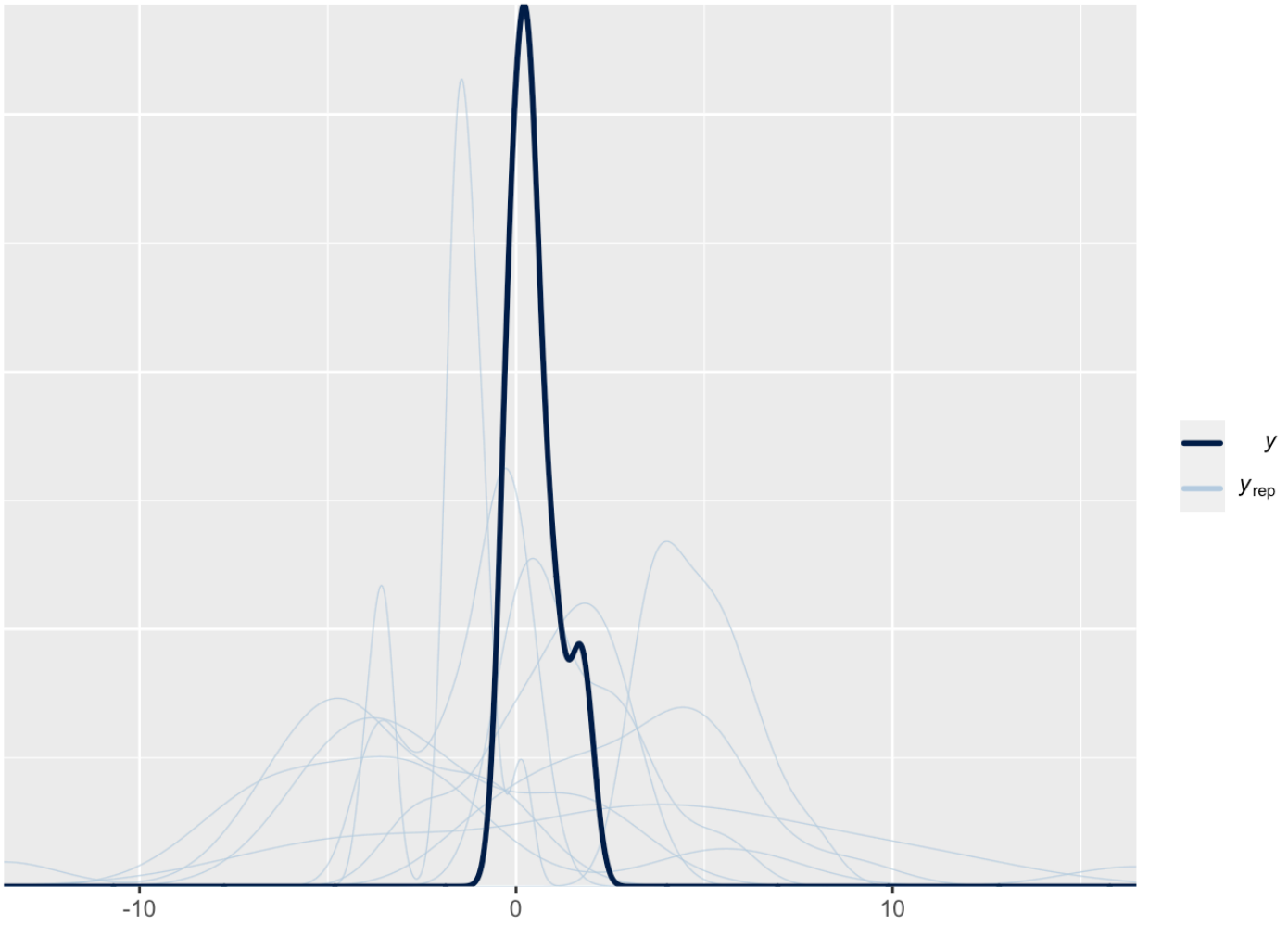
Running the model

```
exclude_matrix_fit <- brm(
  model_matrix,
  data = excluded_matrix,
  prior = matrix_priors,
  family = gaussian,
  refresh=0,
  sample_prior = 'only',
  iter=10000,
  warmup = 1000,
  backend = "cmdstanr",
  threads = threading(2),
  chains = 2,
  cores = 2,
  control = list(
    adapt_delta = 0.99,
    max_treedepth = 20
)
)
```

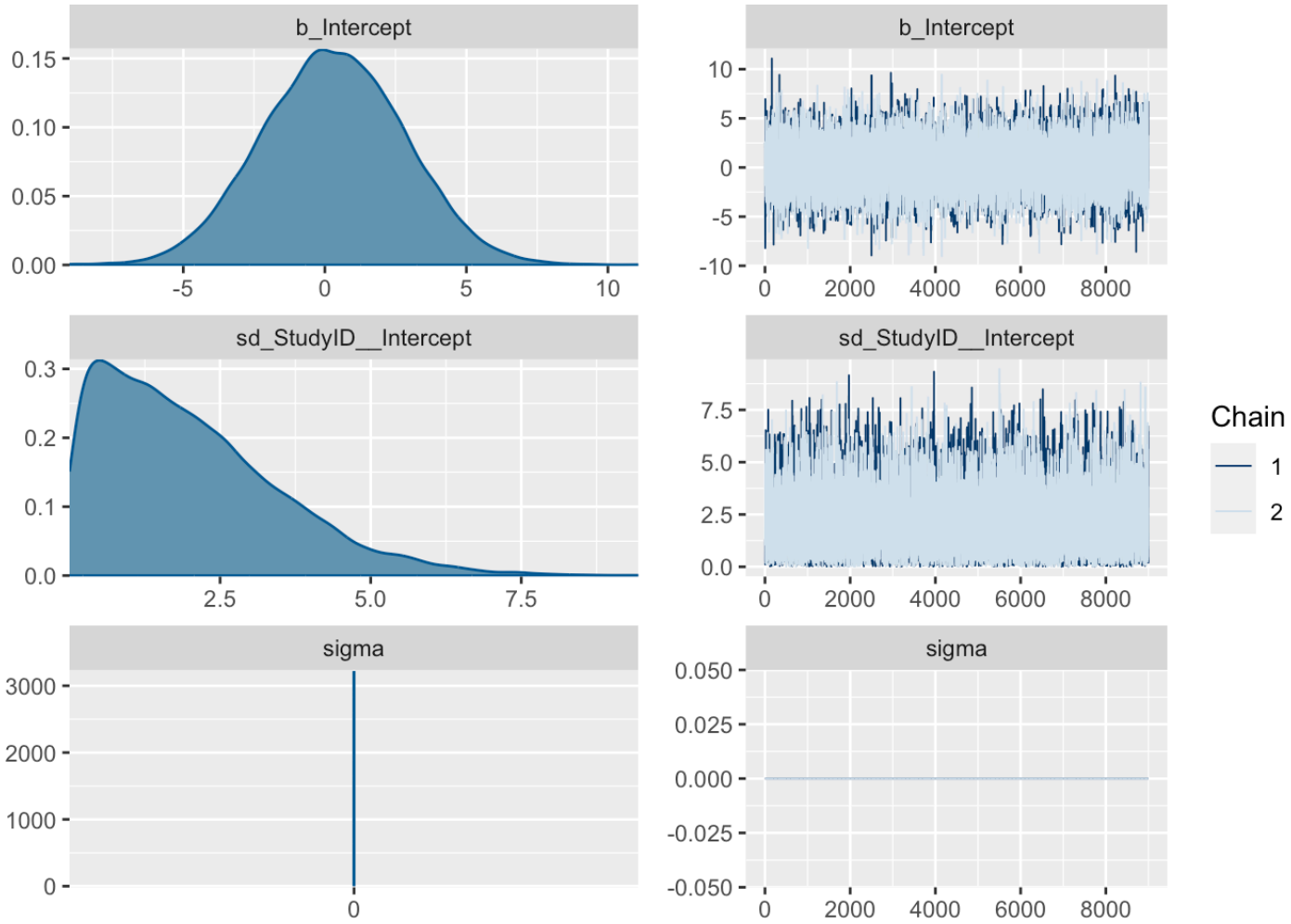
```
## Running MCMC with 2 parallel chains, with 2 thread(s) per chain...
##
## Chain 2 finished in 0.4 seconds.
## Chain 1 finished in 0.5 seconds.
##
## Both chains finished successfully.
## Mean chain execution time: 0.5 seconds.
## Total execution time: 0.6 seconds.
```

Visualizing and plotting

```
pp_check(exclude_matrix_fit)
```




```
plot(exclude_matrix_fit)
```




```
summary(exclude_matrix_fit)
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: effect_size | se(standard_error) ~ 1 + (1 | StudyID)
##    Data: excluded_matrix (Number of observations: 14)
##   Draws: 2 chains, each with iter = 10000; warmup = 1000; thin = 1;
##          total post-warmup draws = 18000
##
## Group-Level Effects:
## ~StudyID (Number of levels: 11)
##               Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(Intercept)     2.00      1.49     0.09     5.60 1.00    14300     7570
##
## Population-Level Effects:
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept     0.31      2.52    -4.57     5.20 1.00    22115    13548
##
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma     0.00      0.00     0.00     0.00   NA       NA       NA
##
## Draws were sampled using sample(hmc). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

Assignment_3

Patrik Molnar, Ditlev Kræn, Bryan Roemelt, Sara Szabo, Manuela Skov Thomasen

2022-11-09

#Assignment 3

##Part 1 - Simulating data

Use meta analysis reported in Parola et al (2020) to create informed simulated data - 100 pairs of schizophrenia and controls, each participant producing 10 repeated measures (10 trials with their speech recorded), for each recording produce 10 acoustic measures (6 from meta analysis and 4 with random noise)

- Do the same for a baseline data set including only 10 noise variables ##Sara ### seeting up variables

```
###Data simulation
n <- 100
trials <- 10

#Effect sizes definition = Informed effect mean and Skeptic effect mean
IEM <- c(-0.5,-1.26,-.74,1.89,0.25,1.3,0,0,0,0)
SEM <- rep(0,10)

#Defining individual vairability from populationd accross trials measurement error
ISD <- 1
TSD <- 0.5
E <- 0.2
```

Simulating the true effect size for each varibale for all pairs of participants

```
for (i in seq(10)){
  temp_informed <- tibble(
    ID=seq(n),
    TrueEffect = rnorm(n, IEM[i], ISD),
    Variable = paste0("V",i))
  temp_skeptic <- tibble(
    ID=seq(n),
    TrueEffect = rnorm(n, SEM[i], ISD),
    Variable = paste0("V",i))

  if(i==1){
    d_informed_true <- temp_informed
    d_skeptic_true <- temp_skeptic
  } else {
    d_informed_true <-  rbind(d_informed_true, temp_informed)
    d_skeptic_true <- rbind(d_skeptic_true, temp_skeptic)
  }
}
```

Creating one row per trial

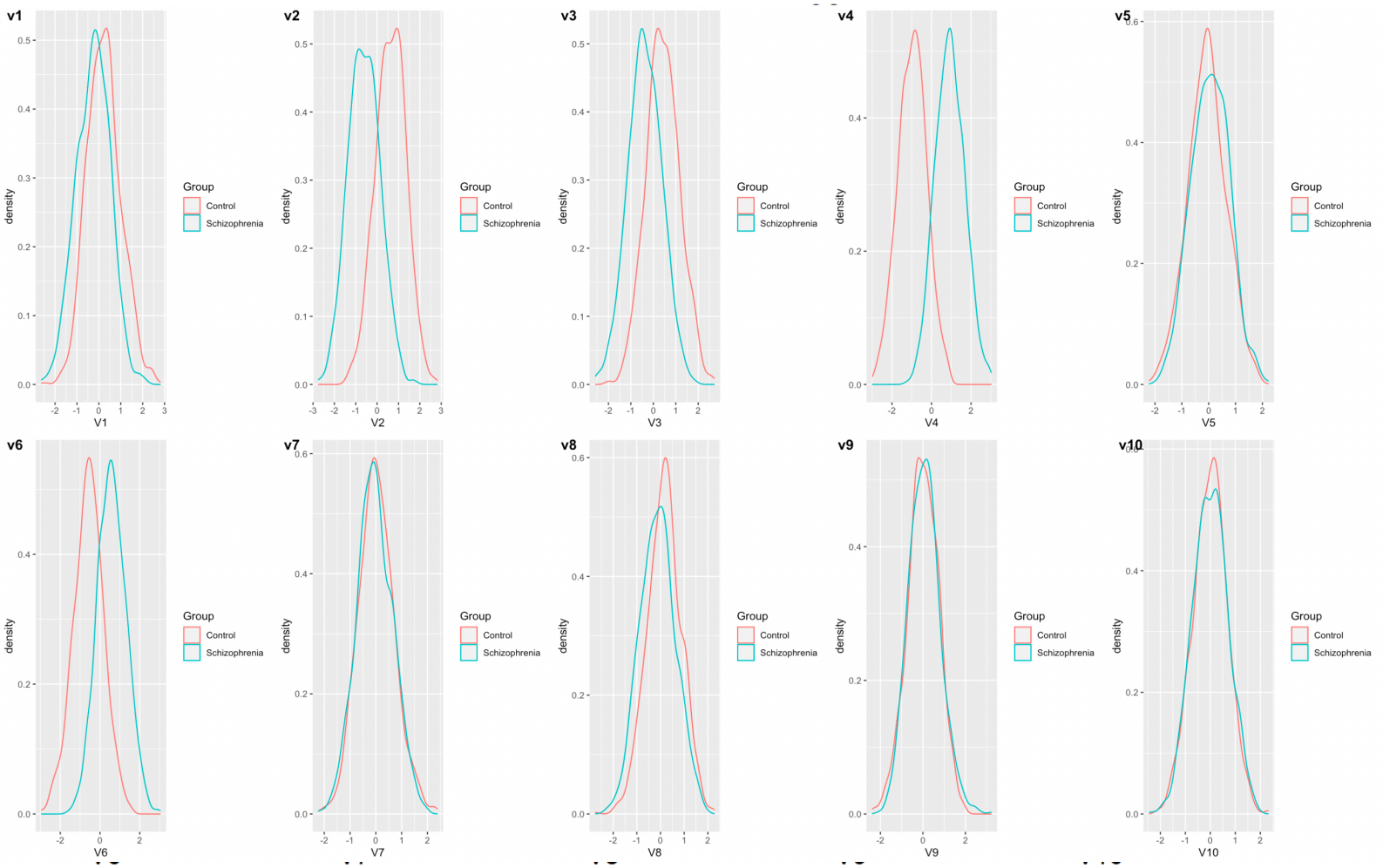
```
d_trial <- tibble(expand_grid(ID=seq(n), Trial = seq(trials), Group = c("Schizophr
enia", "Control")))

d_informed <- merge(d_informed_true, d_trial)
d_skeptic <- merge(d_skeptic_true, d_trial)

for ( i in seq(nrow(d_informed))){
  d_informed$measurement[i] <- ifelse(d_informed$Group[i]=="Schizophrenia",
                                      rnorm(1, rnorm(1, d_informed$TrueEffect[i]/2
, TSD), E),
                                      rnorm(1, rnorm(1, (-d_informed$TrueEffect[i]
)/2, TSD), E))

  d_skeptic$measurement[i] <- ifelse(d_skeptic$Group[i]=="Schizophrenia",
                                      rnorm(1, rnorm(1, d_skeptic$TrueEffect[i]/2,
TSD), E),
                                      rnorm(1, rnorm(1, (-d_skeptic$TrueEffect[i])
/2, TSD), E))
}
```

Transforming the dataframe to a wide format based on the variable

```
d_informed_wide <- d_informed %>%
  mutate(TrueEffect=NULL) %>%
  pivot_wider(names_from = Variable,
              values_from = measurement)
d_skeptic_wide <- d_skeptic %>%
  mutate(TrueEffect=NULL) %>%
  pivot_wider(names_from = Variable,
              values_from = measurement)

Schizo_ID <- d_informed_wide %>%
  filter(Group == 'Schizophrenia')

 control_ID <- d_informed_wide %>%
  filter(Group == 'Control')

control_ID[, 1] <- control_ID[,1] + 100

 d_informed_wide <- rbind(control_ID,Schizo_ID)


 Schizo_ID_skeptic <- d_skeptic_wide %>%
  filter(Group == 'Schizophrenia')

 control_ID_skeptic <- d_skeptic_wide %>%
  filter(Group == 'Control')

 control_ID_skeptic[,1] <- control_ID_skeptic[,1] + 100

 d_skeptic_wide <- rbind(control_ID_skeptic,Schizo_ID_skeptic)
```

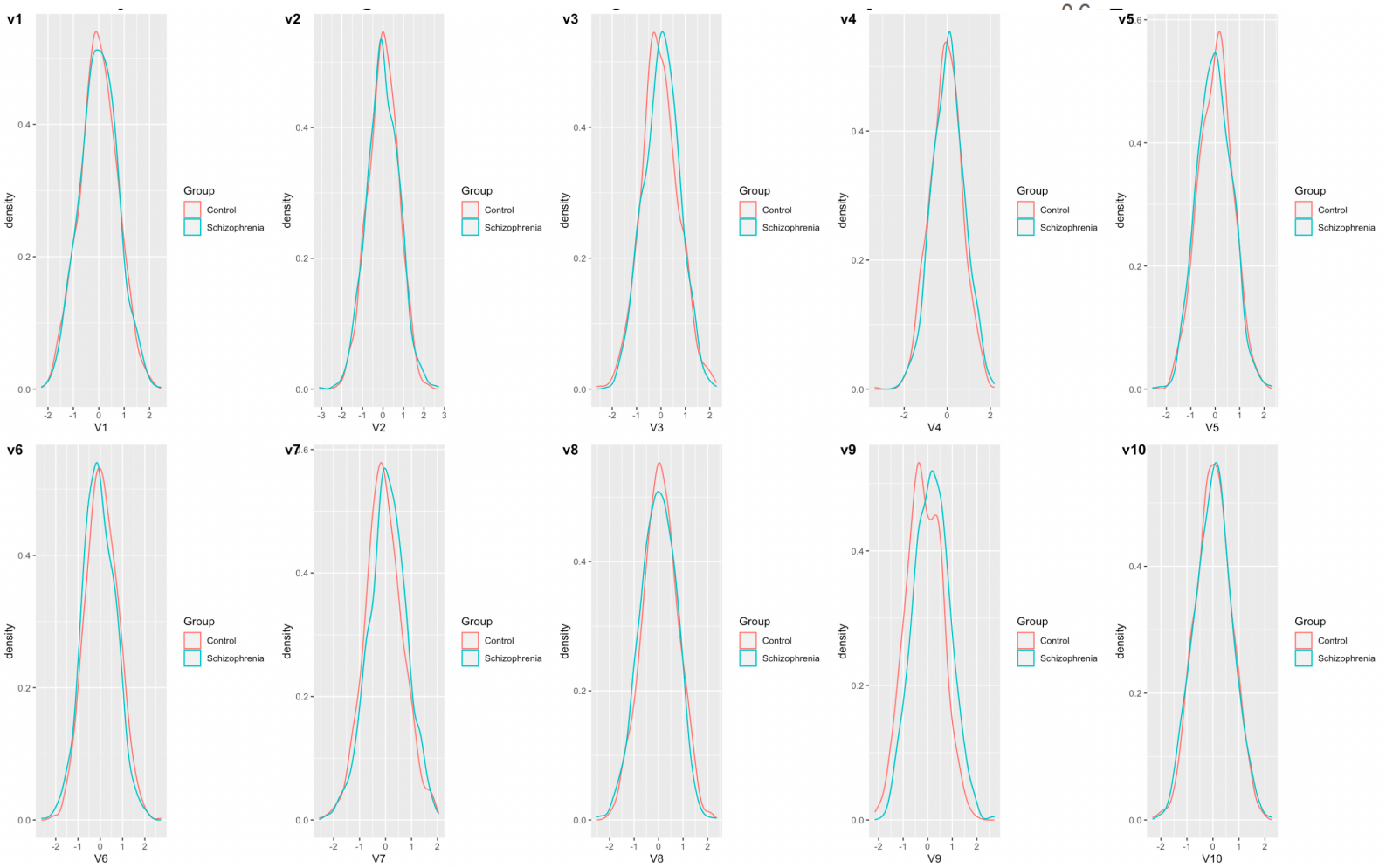
Visualizing the simulated informed data

```
plot1 <- d_informed_wide %>%
  ggplot(aes(x = V1, color = Group))+
  geom_density()

plot2 <- d_informed_wide %>%
  ggplot(aes(x = V2, color = Group))+
  geom_density()

plot3 <- d_informed_wide %>%
  ggplot(aes(x = V3, color = Group))+
  geom_density()

plot4 <- d_informed_wide %>%
  ggplot(aes(x = V4, color = Group))+
  geom_density()

plot5 <- d_informed_wide %>%
  ggplot(aes(x = V5, color = Group))+
  geom_density()

plot6 <- d_informed_wide %>%
  ggplot(aes(x = V6, color = Group))+
  geom_density()

plot7 <- d_informed_wide %>%
  ggplot(aes(x = V7, color = Group))+
  geom_density()

plot8 <- d_informed_wide %>%
  ggplot(aes(x = V8, color = Group))+
  geom_density()

plot9 <- d_informed_wide %>%
  ggplot(aes(x = V9, color = Group))+
  geom_density()

plot10 <- d_informed_wide %>%
  ggplot(aes(x = V10, color = Group))+
  geom_density()

cowplot::plot_grid(plot1, plot2, plot3, plot4, plot5, plot6, plot7, plot8, plot9,
plot10,
          labels = c("v1", "v2", "v3",'v4','v5','v6','v7','v8','v9','v10'),
          ncol = 5, nrow = 2)
```



Visualizing the simulated data for skeptical data frame

```
plot1_s <- d_skeptic_wide %>%
  ggplot(aes(x = V1, color = Group))+
  geom_density()

plot2_s <- d_skeptic_wide %>%
  ggplot(aes(x = V2, color = Group))+
  geom_density()

plot3_s <- d_skeptic_wide %>%
  ggplot(aes(x = V3, color = Group))+
  geom_density()

plot4_s <- d_skeptic_wide %>%
  ggplot(aes(x = V4, color = Group))+
  geom_density()

plot5_s <- d_skeptic_wide %>%
  ggplot(aes(x = V5, color = Group))+
  geom_density()

plot6_s <- d_skeptic_wide %>%
  ggplot(aes(x = V6, color = Group))+
  geom_density()

plot7_s <- d_skeptic_wide %>%
  ggplot(aes(x = V7, color = Group))+
  geom_density()

plot8_s <- d_skeptic_wide %>%
  ggplot(aes(x = V8, color = Group))+
  geom_density()

plot9_s <- d_skeptic_wide %>%
  ggplot(aes(x = V9, color = Group))+
  geom_density()

plot10_s <- d_skeptic_wide %>%
  ggplot(aes(x = V10, color = Group))+
  geom_density()

cowplot::plot_grid(plot1_s, plot2_s, plot3_s, plot4_s, plot5_s, plot6_s, plot7_s,
plot8_s, plot9_s, plot10_s,
          labels = c("v1", "v2", "v3",'v4','v5','v6','v7','v8','v9','v10'),
          ncol = 5, nrow = 2)
```



##Patrik ##Part 2 - Machine larning pipeline on simulated data Build a machine leaning pipeline (separately on the 2 datasets) - create a data budget (e.g., balanced training and test sets) pree-process the data (e.g., scaling the features) - fit and assess a classification algorithm on the training data (e.g., bayesian multilevel logistic regression) - assess performance on the test set - discuss whether performance and feature importance is as expected

Data budget (splitting) and pree-processing (scaling)

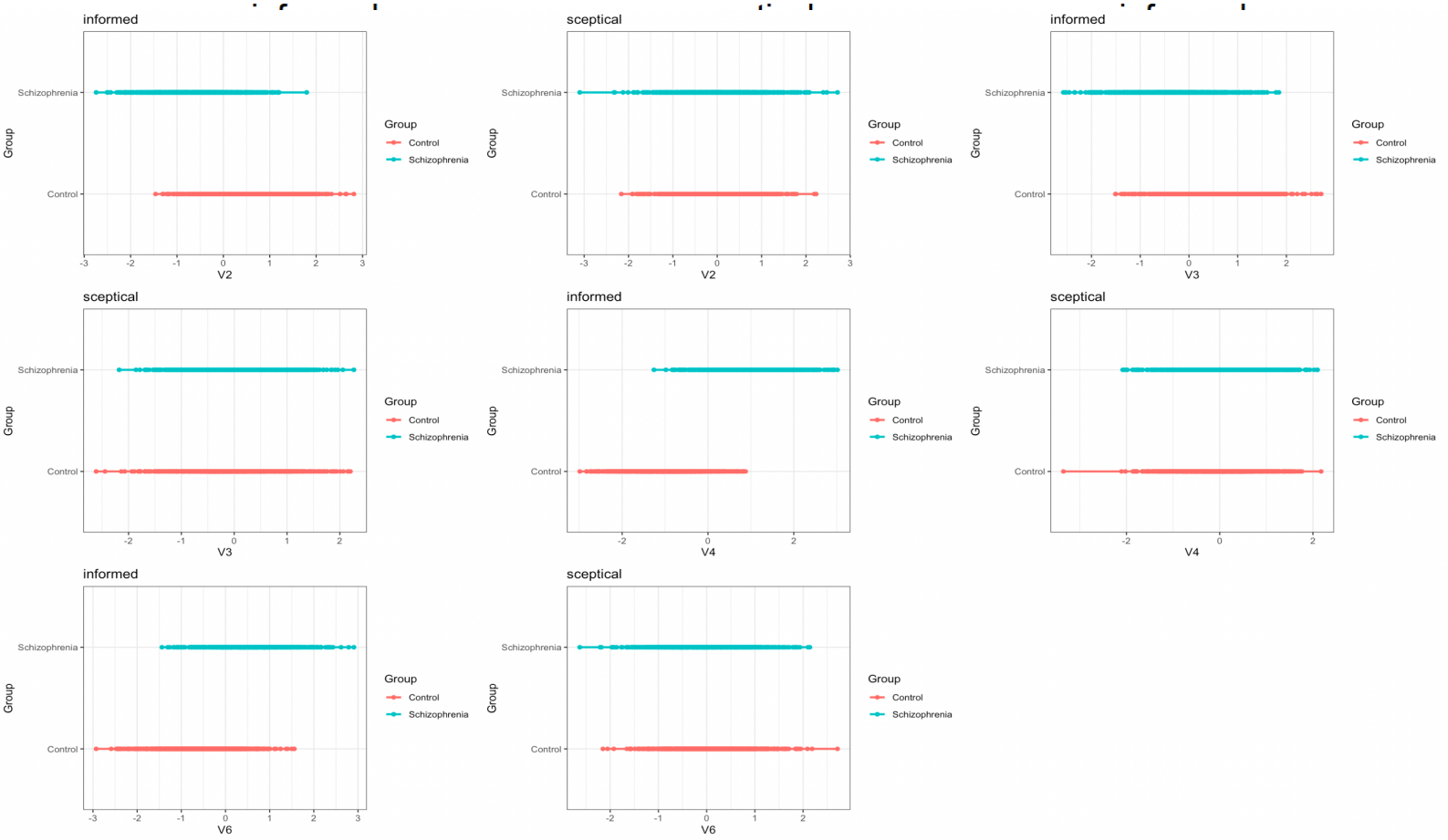
```
set.seed(260)


d_informed_wide <- d_informed_wide %>%
  mutate(pair_ID=ID) %>%
  mutate(pair_ID= ifelse(ID>100, ID-100, ID))

d_skeptic_wide <- d_skeptic_wide %>%
  mutate(pair_ID=ID) %>%
  mutate(pair_ID= ifelse(ID>100, ID-100, ID))

split_inf <- initial_split(d_informed_wide, prop = 4/5)

train_informed <- training(split_inf)
test_informed <- testing(split_inf)
```

```
split_skep <- initial_split(d_skeptic_wide, prop = 4/5)
train_skeptic <- training(split_skep)
test_skeptic <- testing(split_skep)

train_informed$ID <- as.factor(train_informed$ID)
train_skeptic$ID <- as.factor(train_skeptic$ID)

test_informed$ID <- as.factor(test_informed$ID)
test_skeptic$ID <- as.factor(test_skeptic$ID)

train_informed$pair_ID <- as.factor(train_informed$pair_ID)
train_skeptic$pair_ID <- as.factor(train_skeptic$pair_ID)

test_informed$pair_ID <- as.factor(test_informed$pair_ID)
test_skeptic$pair_ID <- as.factor(test_skeptic$pair_ID)

train_informed$Trial <- as.factor(train_informed$Trial)
train_skeptic$Trial <- as.factor(train_skeptic$Trial)

test_informed$Trial <- as.factor(test_informed$Trial)
test_skeptic$Trial <- as.factor(test_skeptic$Trial)


rec_informed <- train_informed %>%
  recipe(Group~.) %>%
  update_role(ID, new_role = 'ID') %>%
  step_scale(all_numeric()) %>%
  step_center(all_numeric()) %>%
  prep(training=train_informed, retain=TRUE)

rec_skeptic <- train_skeptic %>%
  recipe(Group~.) %>%
  update_role(ID, new_role = 'ID') %>%
  step_scale(all_numeric()) %>%
  step_center(all_numeric()) %>%
  prep(training=train_informed, retain=TRUE)


train_informed_s <- juice(rec_informed)
test_informed_s <- bake(rec_informed, new_data = test_informed)

train_skeptic_s <- juice(rec_skeptic)
test_skeptic_s <- bake(rec_skeptic, new_data = test_skeptic)
```

Visual inspection of the data

```
plot_i2 <- ggplot(train_informed, aes(V2, Group, colour=Group))+
  geom_point()+
  geom_smooth(method = "glm", se=FALSE)+
  theme_bw()+
  ggtitle("informed")

plot_s2 <- ggplot(train_skeptic, aes(V2, Group, colour=Group))+
  geom_point()+
  geom_smooth(method = "glm", se=FALSE)+
  theme_bw()+
  ggtitle("sceptical")

plot_i3 <- ggplot(train_informed, aes(V3, Group, colour=Group))+
  geom_point()+
  geom_smooth(method = "glm", se=FALSE)+
  theme_bw()+
  ggtitle("informed")

plot_s3 <- ggplot(train_skeptic, aes(V3, Group, colour=Group))+
  geom_point()+
  geom_smooth(method = "glm", se=FALSE)+
  theme_bw()+
  ggtitle("sceptical")

plot_i4 <- ggplot(train_informed, aes(V4, Group, colour=Group))+
  geom_point()+
  geom_smooth(method = "glm", se=FALSE)+
  theme_bw()+
  ggtitle("informed")

plot_s4 <- ggplot(train_skeptic, aes(V4, Group, colour=Group))+
  geom_point()+
  geom_smooth(method = "glm", se=FALSE)+
  theme_bw()+
  ggtitle("sceptical")

plot_i6 <- ggplot(train_informed, aes(V6, Group, colour=Group))+
  geom_point()+
  geom_smooth(method = "glm", se=FALSE)+
  theme_bw()+
  ggtitle("informed")

plot_s6 <- ggplot(train_skeptic, aes(V6, Group, colour=Group))+
  geom_point()+
  geom_smooth(method = "glm", se=FALSE)+
  theme_bw()+
  ggtitle("sceptical")

plot_grid(plot_i2, plot_s2, plot_i3, plot_s3, plot_i4, plot_s4, plot_i6, plot_s6)
```

```
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
```



fit and asses a classification algorithm on training data (Baysian)

```
##Setting up the model
PR_f0 <-bf(Group~1+V1+V2+V3+V4+V5+V6+V7+V8+V9+V10)

PR_f1 <-bf(Group~1+V1+V2+V3+V4+V5+V6+V7+V8+V9+V10+(1|ID))

get_prior(PR_f0, train_informed_s, family = bernoulli)
```

```
##                 prior     class coef group resp dpar nlpar lb ub       source
##                (flat)         b                                       default
##                (flat)         b   V1                             (vectorized)
##                (flat)         b  V10                             (vectorized)
##                (flat)         b   V2                             (vectorized)
##                (flat)         b   V3                             (vectorized)
##                (flat)         b   V4                             (vectorized)
##                (flat)         b   V5                             (vectorized)
##                (flat)         b   V6                             (vectorized)
##                (flat)         b   V7                             (vectorized)
##                (flat)         b   V8                             (vectorized)
##                (flat)         b   V9                             (vectorized)
##  student_t(3, 0, 2.5) Intercept                                        default
```

```
get_prior(PR_f1, train_informed_s, family = bernoulli)
```

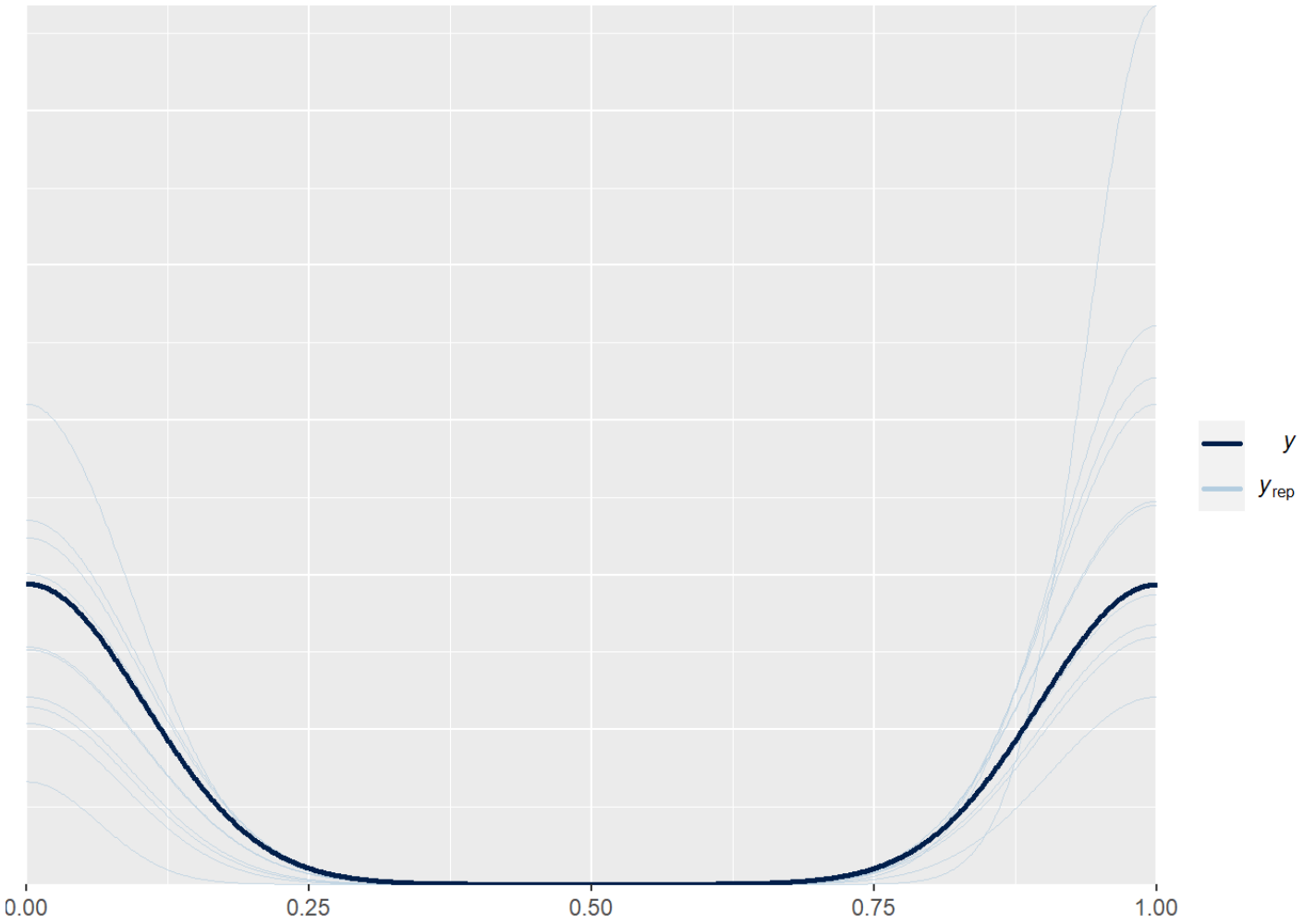
```
##                 prior     class      coef group resp dpar nlpar lb ub
##                (flat)         b
##                (flat)         b        V1
##                (flat)         b       V10
##                (flat)         b        V2
##                (flat)         b        V3
##                (flat)         b        V4
##                (flat)         b        V5
##                (flat)         b        V6
##                (flat)         b        V7
##                (flat)         b        V8
##                (flat)         b        V9
##  student_t(3, 0, 2.5) Intercept
##  student_t(3, 0, 2.5)        sd                                   0
##  student_t(3, 0, 2.5)        sd              ID                   0
##  student_t(3, 0, 2.5)        sd Intercept    ID                   0
##       source
##      default
##  (vectorized)
##  (vectorized)
##  (vectorized)
##  (vectorized)
##  (vectorized)
##  (vectorized)
##  (vectorized)
##  (vectorized)
##  (vectorized)
##  (vectorized)
##  (vectorized)
##      default
##      default
##  (vectorized)
##  (vectorized)
```

Setting priors

```
PR_p0 <- c(
  prior(normal(0, 1), class=Intercept),
  prior(normal(0, 0.3), class=b)
)

PR_p1 <- c(
  prior(normal(0, 1), class=Intercept),
  prior(normal(0, 0.3), class=sd),
  prior(normal(0, 0.3), class=b)
)
```

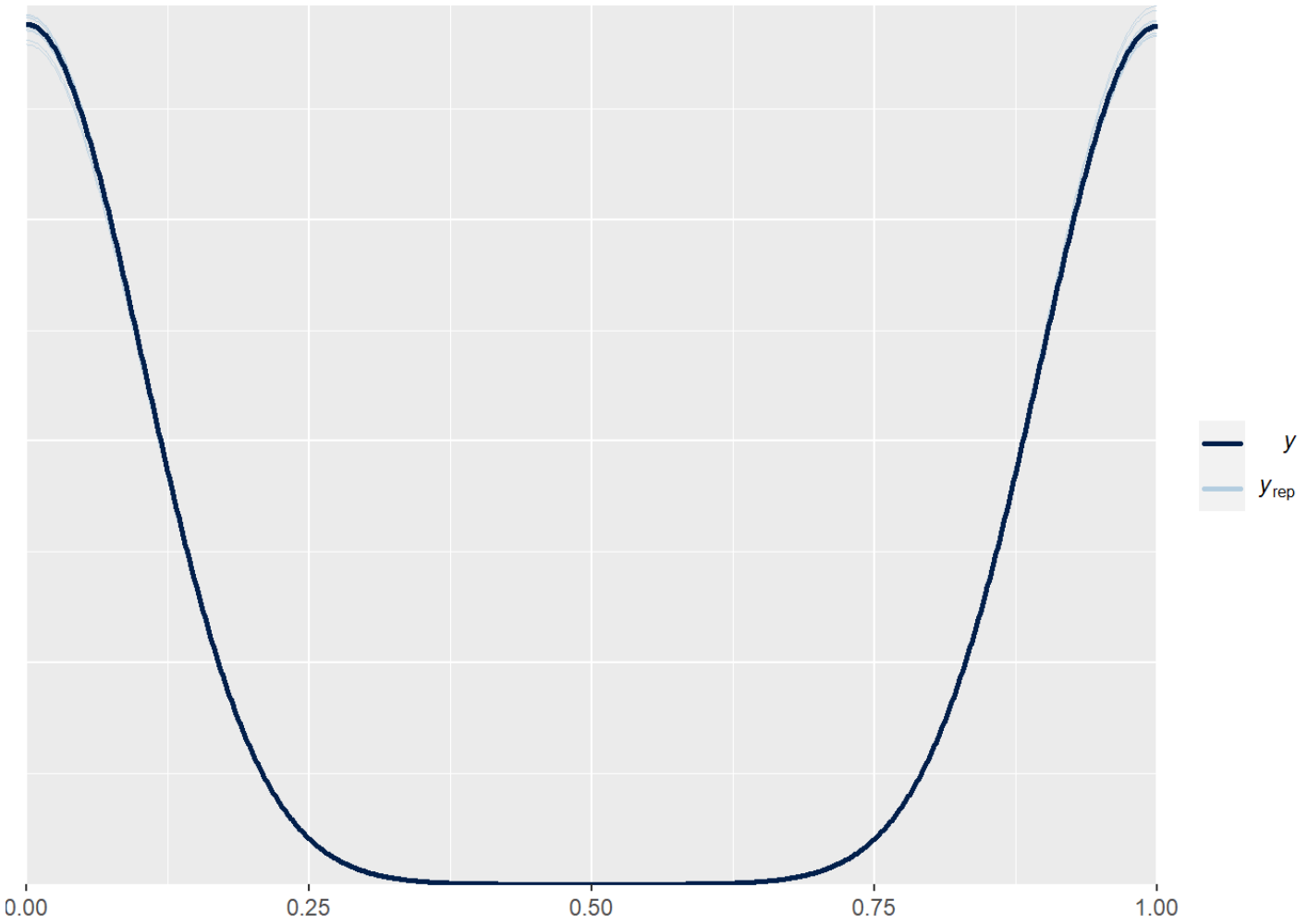
Fitting the first model on both the skeptical and informed data

```
##Model fit on priors
pr_m0_inf <- brm(
  PR_f0,
  data = train_informed_s,
  prior = PR_p0,
  family = bernoulli,
  refresh=0,
  sample_prior = 'only',
  iter=6000,
  warmup = 2500,
  backend = "cmdstanr",
  threads = threading(2),
  chains = 4,
  cores = 4,
  control = list(
    adapt_delta = 0.9,
    max_treedepth = 20)
)


#pp_check(pr_m0_inf)

pr_m0_skep <- brm(
  PR_f0,
  data = train_skeptic_s,
  prior = PR_p0,
  family = bernoulli,
  refresh=0,
  sample_prior = 'only',
  iter=6000,
  warmup = 2500,
  backend = "cmdstanr",
  threads = threading(2),
  chains = 4,
  cores = 4,
  control = list(
    adapt_delta = 0.9,
    max_treedepth = 20)
)


pp_check(pr_m0_skep)
```

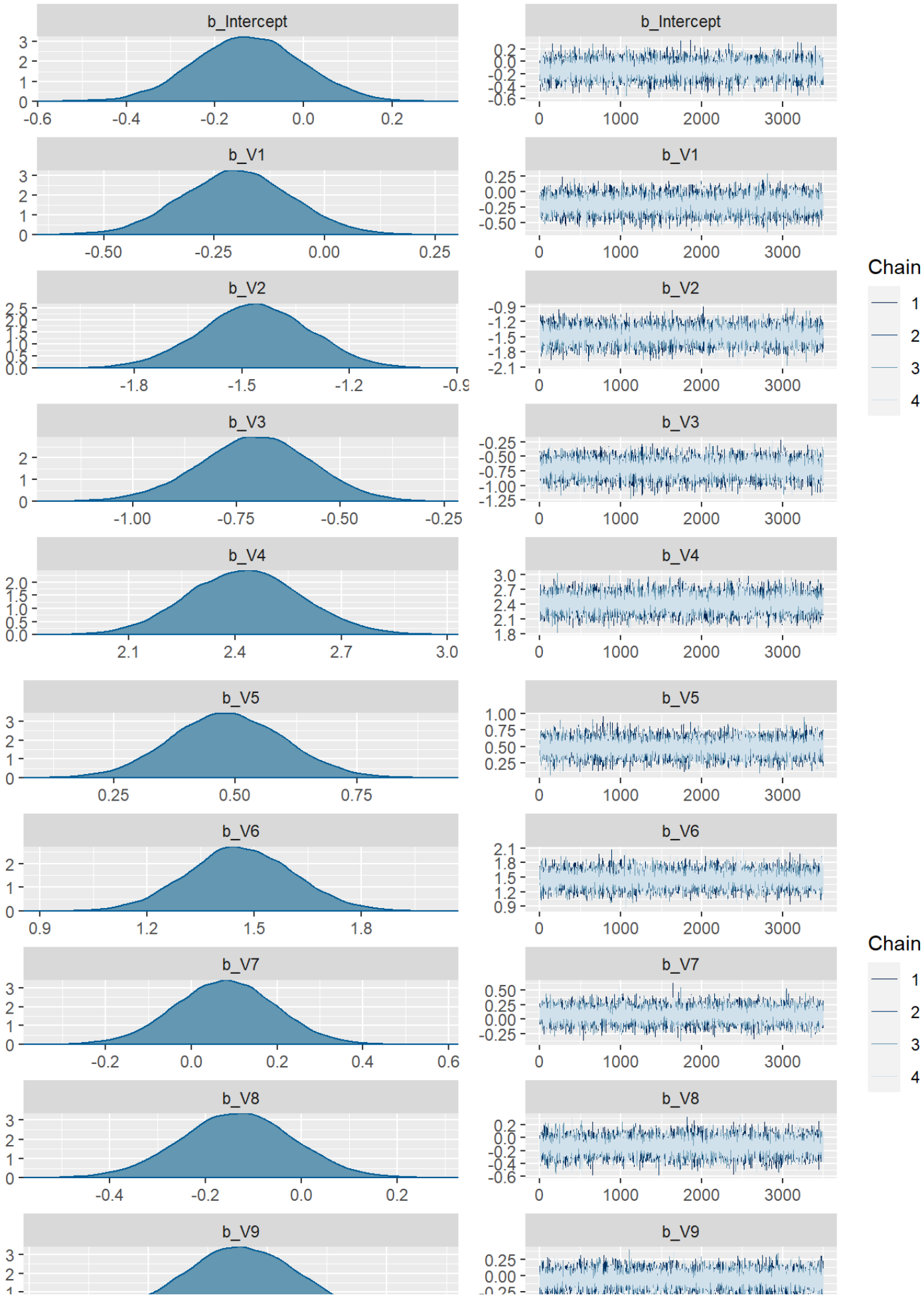


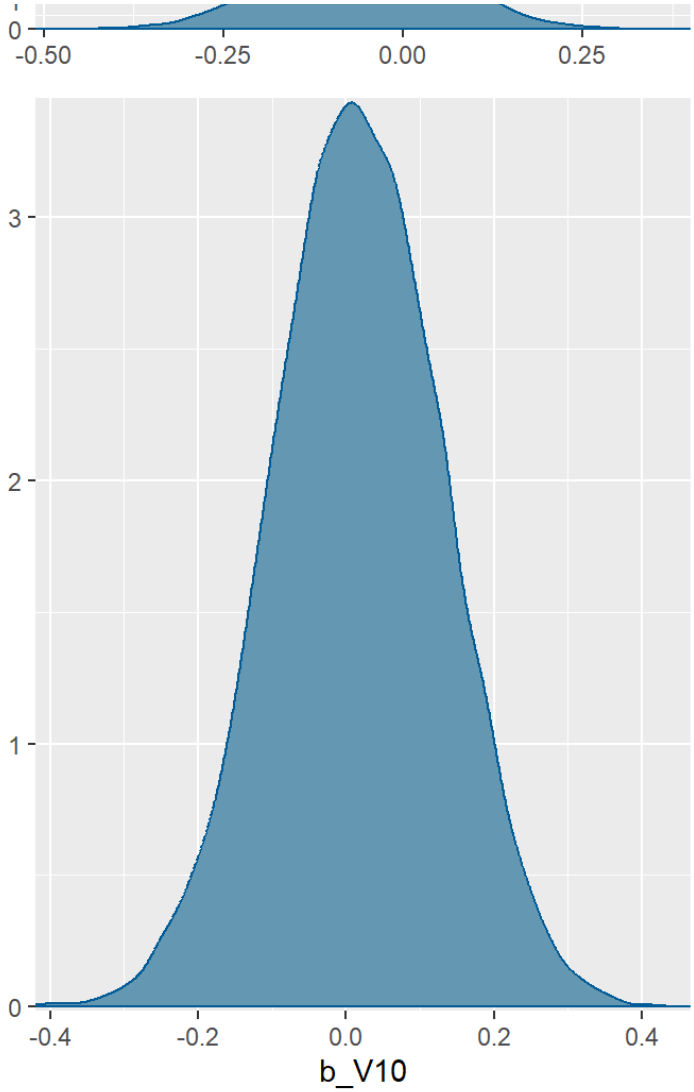
##Bryan

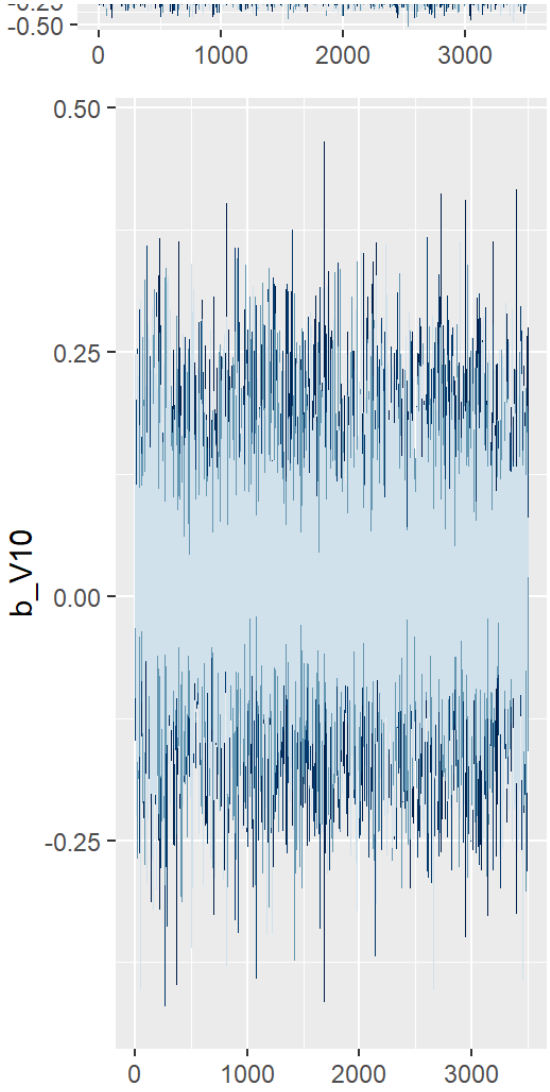
```
##Model fit on informed data
pr_m0_fit_inf <- brm(
  PR_f0,
  data = train_informed_s,
  prior = PR_p0,
  family = bernoulli,
  refresh=0,
  sample_prior = TRUE,
  iter=6000,
  warmup = 2500,
  backend = "cmdstanr",
  threads = threading(2),
  chains = 4,
  cores = 4,
  control = list(
    adapt_delta = 0.9,
    max_treedepth = 20)
)

pp_check(pr_m0_fit_inf)
```




```
plot(pr_m0_fit_inf)
```

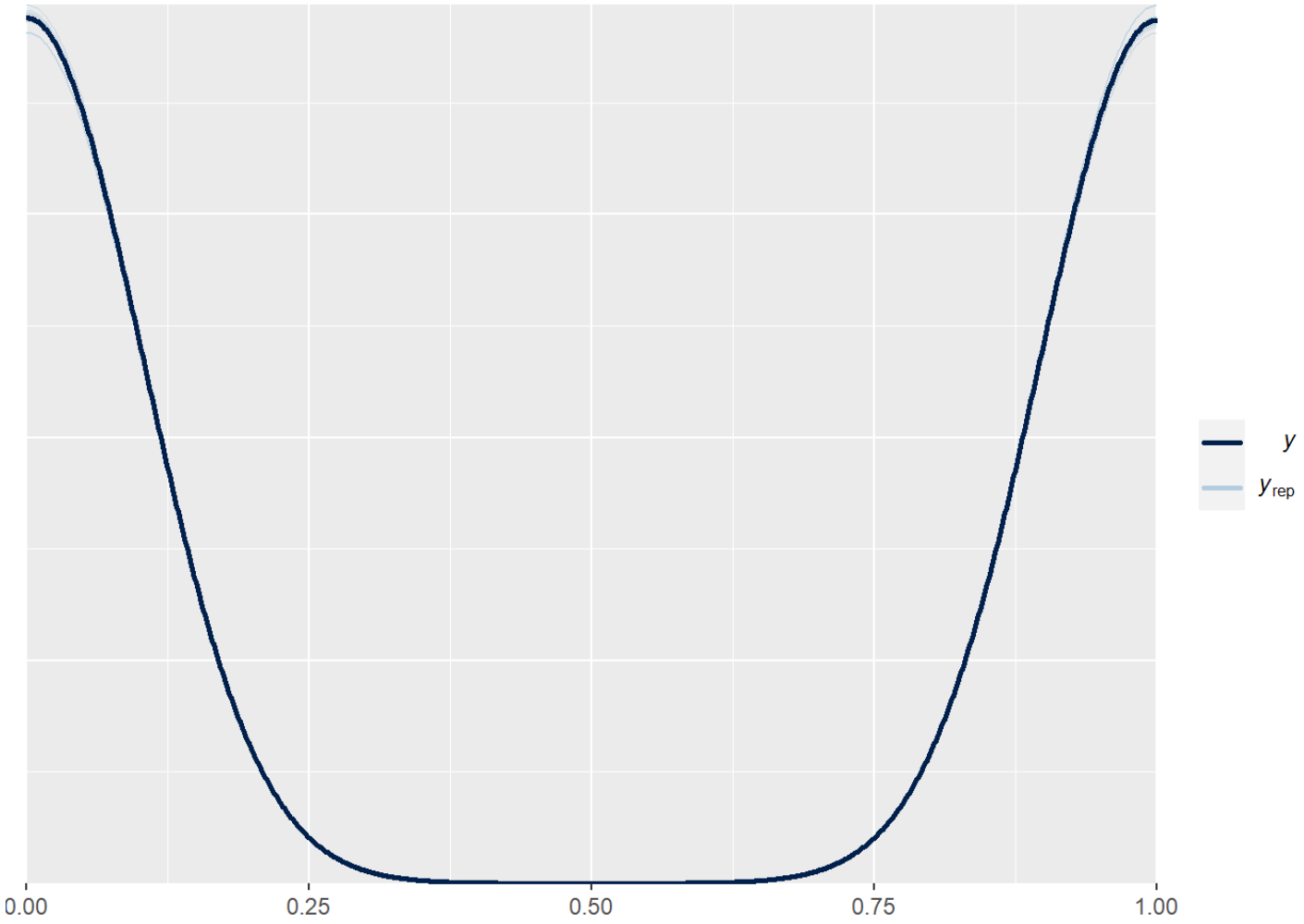

b_Intercept
b_V1
b_V2
b_V3
b_V4
b_V5
b_V6
b_V7
b_V8
b_V9
Chain
1
2
3
4



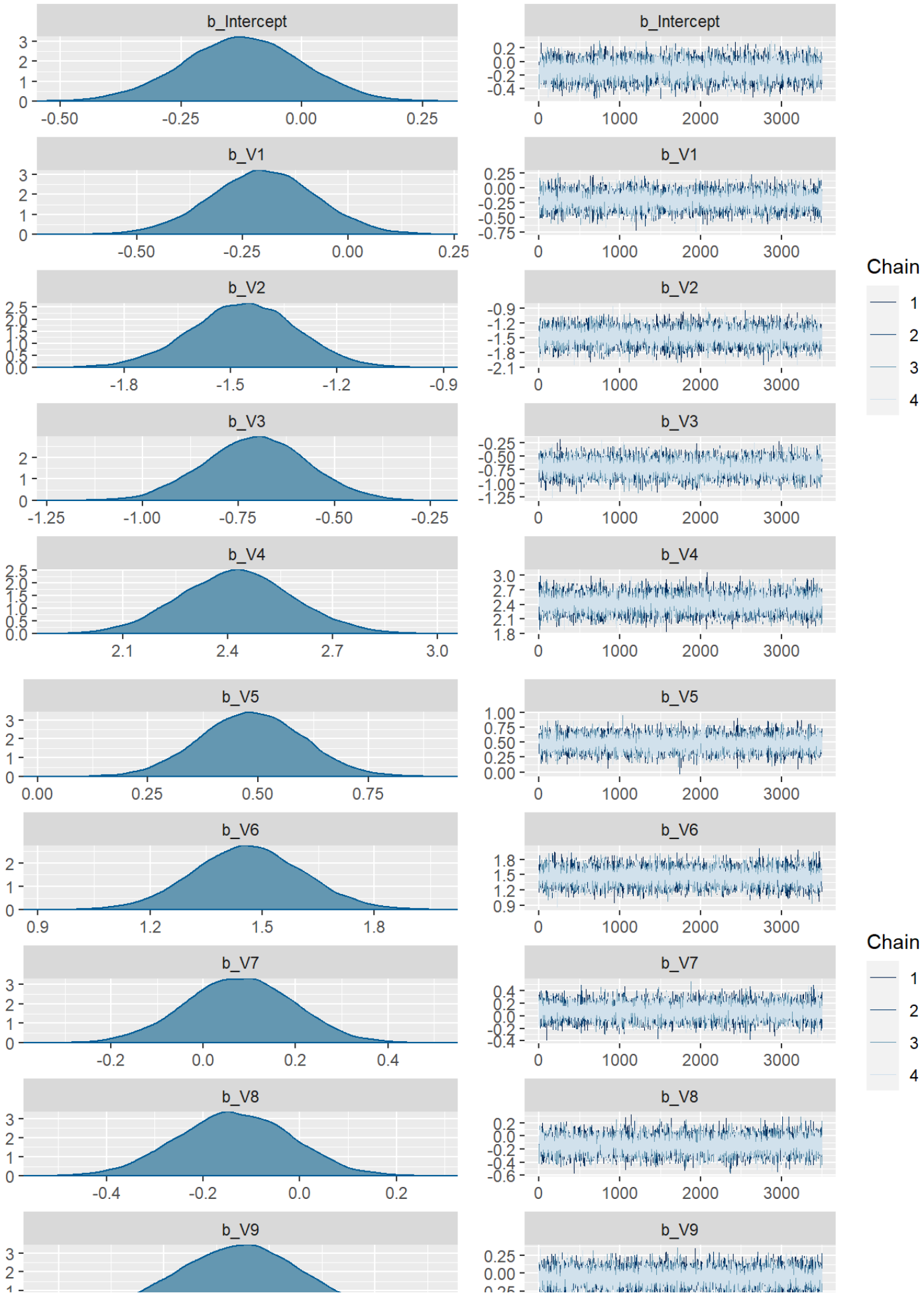


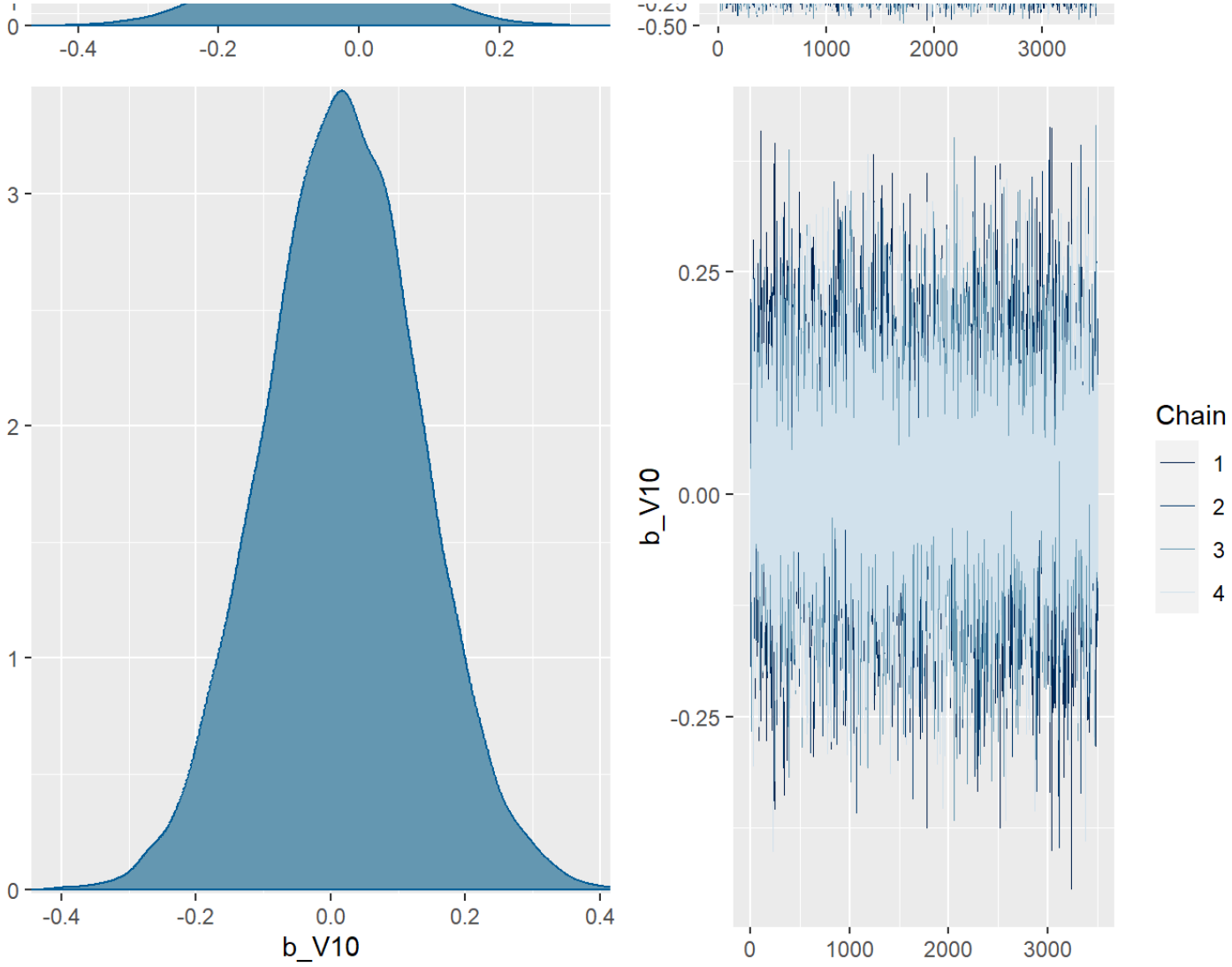

```
summary(pr_m0_fit_inf)
```

```
##Model fit on skeptical data
pr_m0_fit_skep <- brm(
  PR_f0,
  data = train_skeptic_s,
  prior = PR_p0,
  family = bernoulli,
  refresh=0,
  sample_prior = TRUE,
  iter=6000,
  warmup = 2500,
  backend = "cmdstanr",
  threads = threading(2),
  chains = 4,
  cores = 4,
  control = list(
    adapt_delta = 0.9,
    max_treedepth = 20)
)

pp_check(pr_m0_fit_skep)
```




```
plot(pr_m0_fit_skep)
```

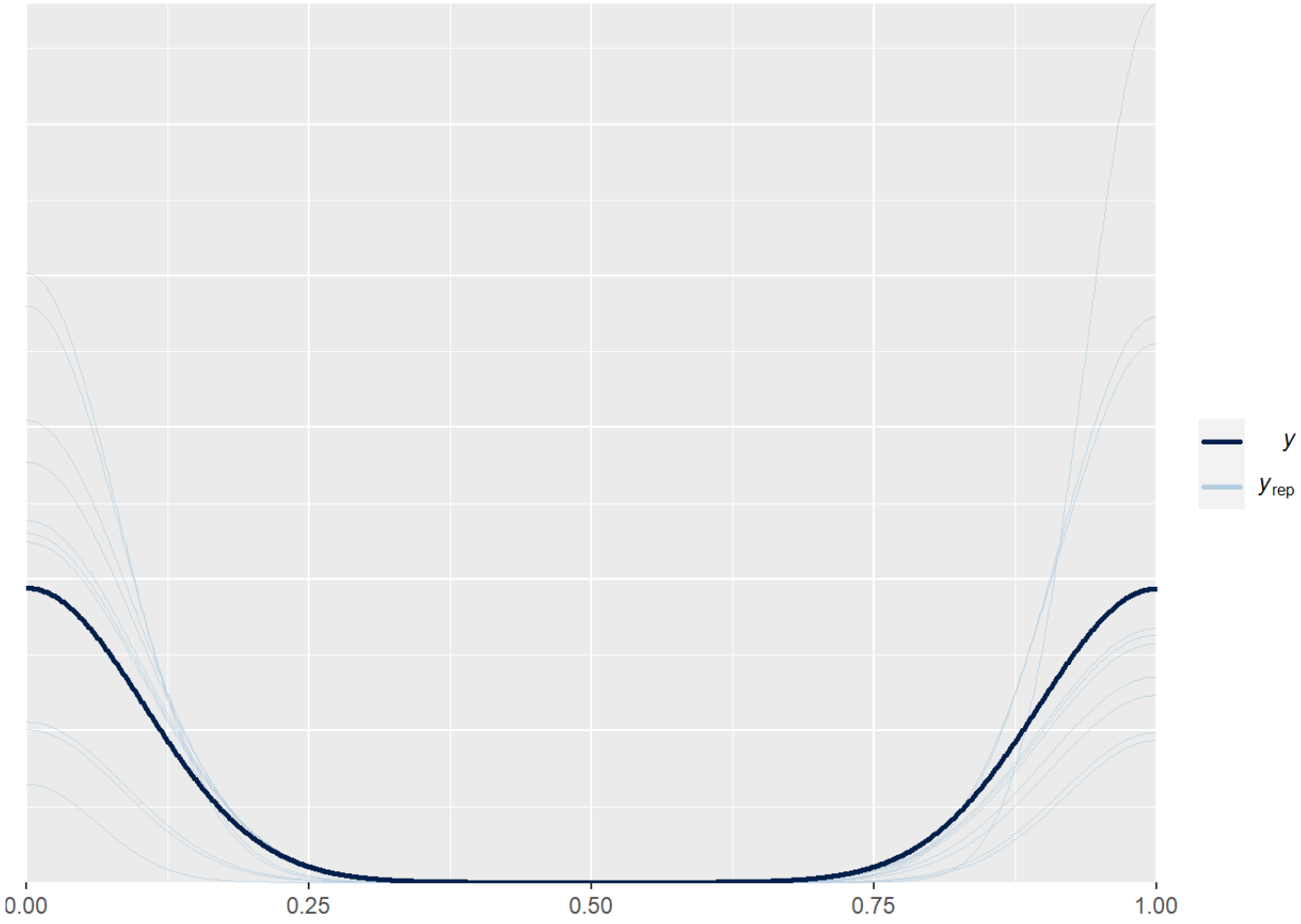

b_Intercept
b_V1
b_V2
b_V3
b_V4
b_V5
b_V6
b_V7
b_V8
b_V9
Chain
1
2
3
4



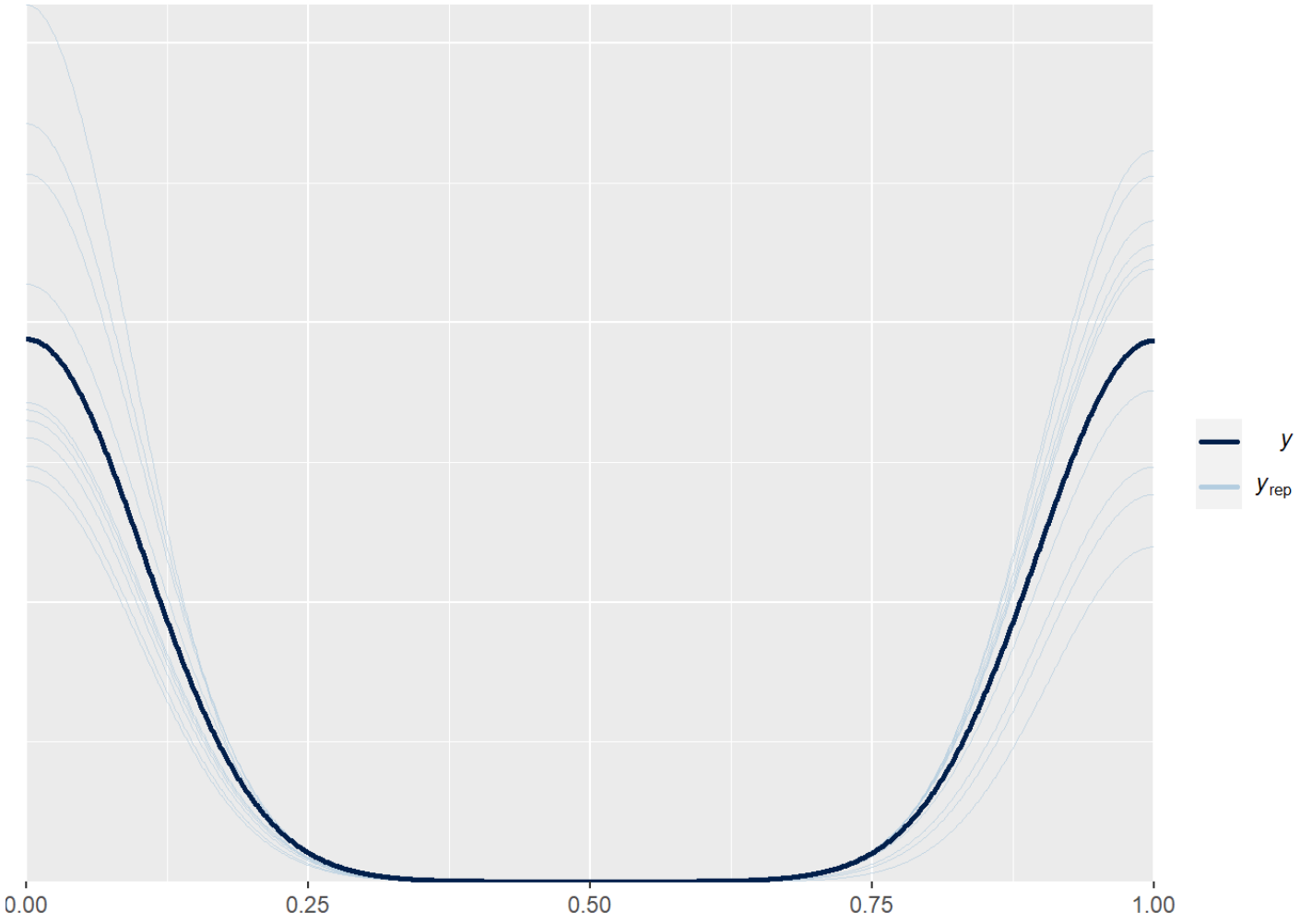

```
summary(pr_m0_fit_skep)
```

Fitting the second model on skeptical and informed data

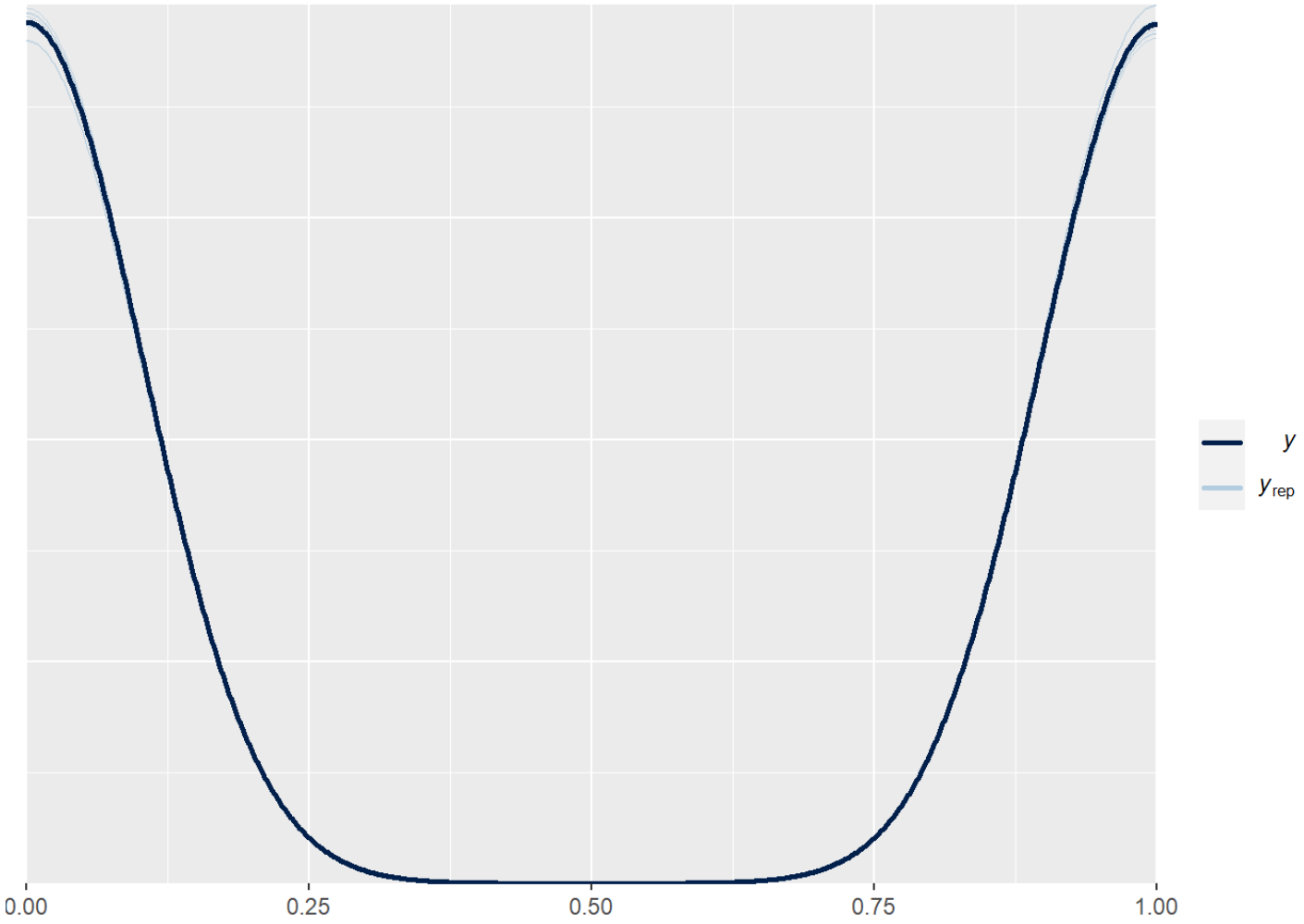
```
##Second model fit on priors
pr_m1_inf <- brm(
  PR_f1,
  data = train_informed_s,
  prior = PR_p1,
  family = bernoulli,
  refresh=0,
  sample_prior = 'only',
  iter=6000,
  warmup = 2500,
  backend = "cmdstanr",
  threads = threading(2),
  chains = 4,
  cores = 4,
  control = list(
    adapt_delta = 0.9,
    max_treedepth = 20)
)


pp_check(pr_m1_inf)
```



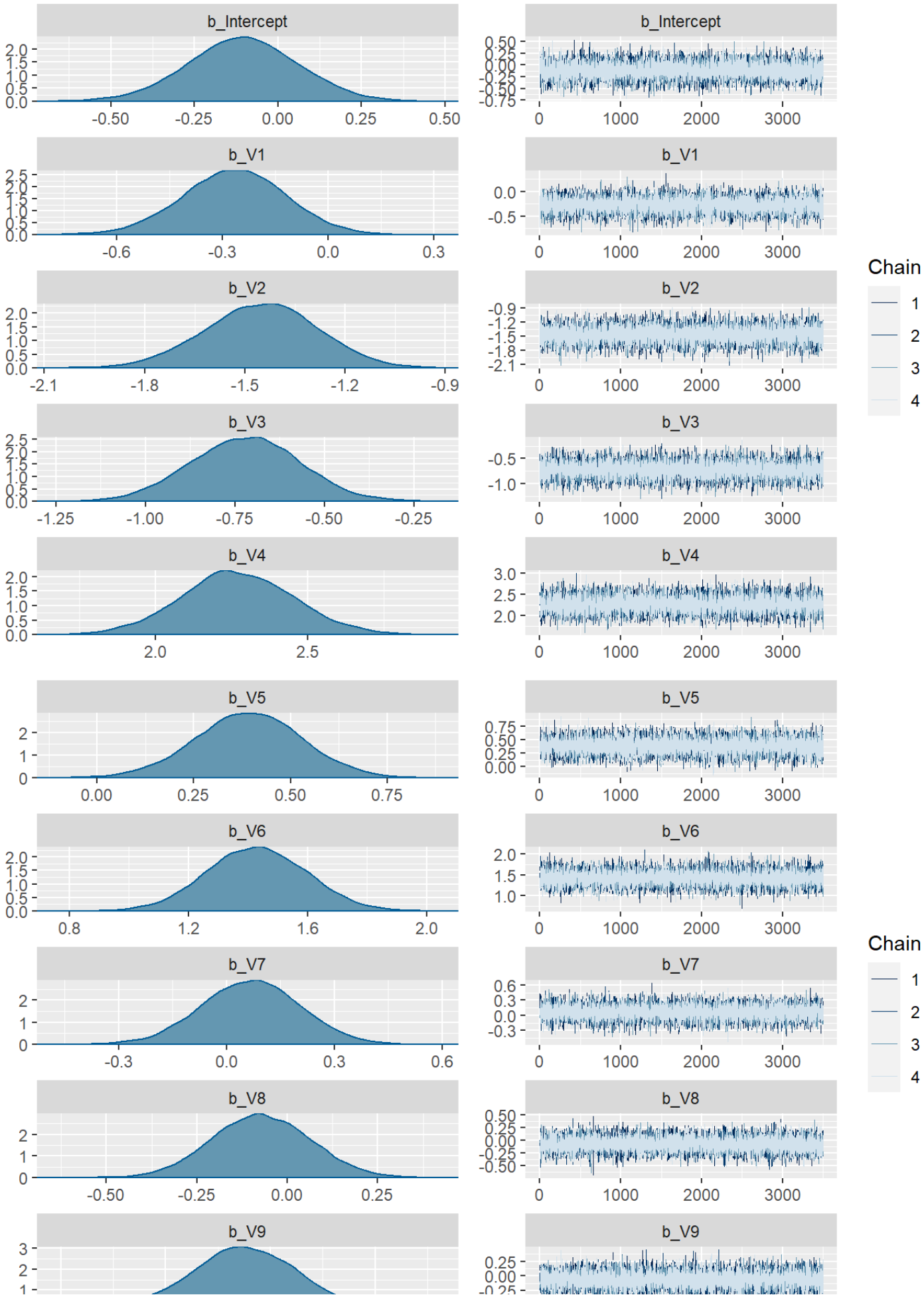
```
pr_m1_skep <- brm(
  PR_f1,
  data = train_skeptic_s,
  prior = PR_p1,
  family = bernoulli,
  refresh=0,
  sample_prior = 'only',
  iter=6000,
  warmup = 2500,
  backend = "cmdstanr",
  threads = threading(2),
  chains = 4,
  cores = 4,
  control = list(
    adapt_delta = 0.9,
    max_treedepth = 20)
)


pp_check(pr_m1_skep)
```

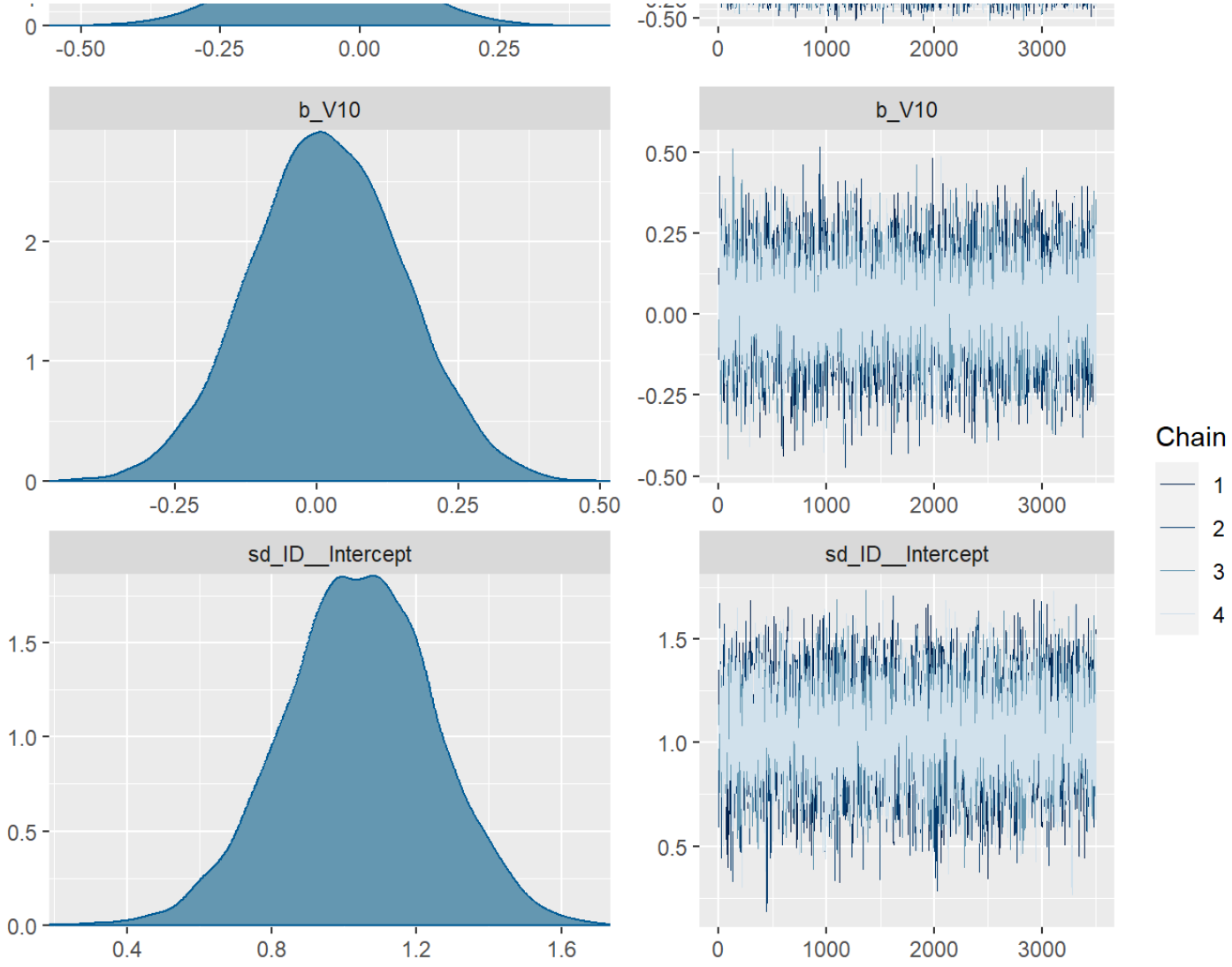


```
##second model fit on informed data
pr_m1_fit_inf <- brm(
  PR_f1,
  data = train_informed_s,
  prior = PR_p1,
  family = bernoulli,
  refresh=0,
  sample_prior = TRUE,
  iter=6000,
  warmup = 2500,
  backend = "cmdstanr",
  threads = threading(2),
  chains = 4,
  cores = 4,
  control = list(
    adapt_delta = 0.9,
    max_treedepth = 20)
)

pp_check(pr_m1_fit_inf)
```



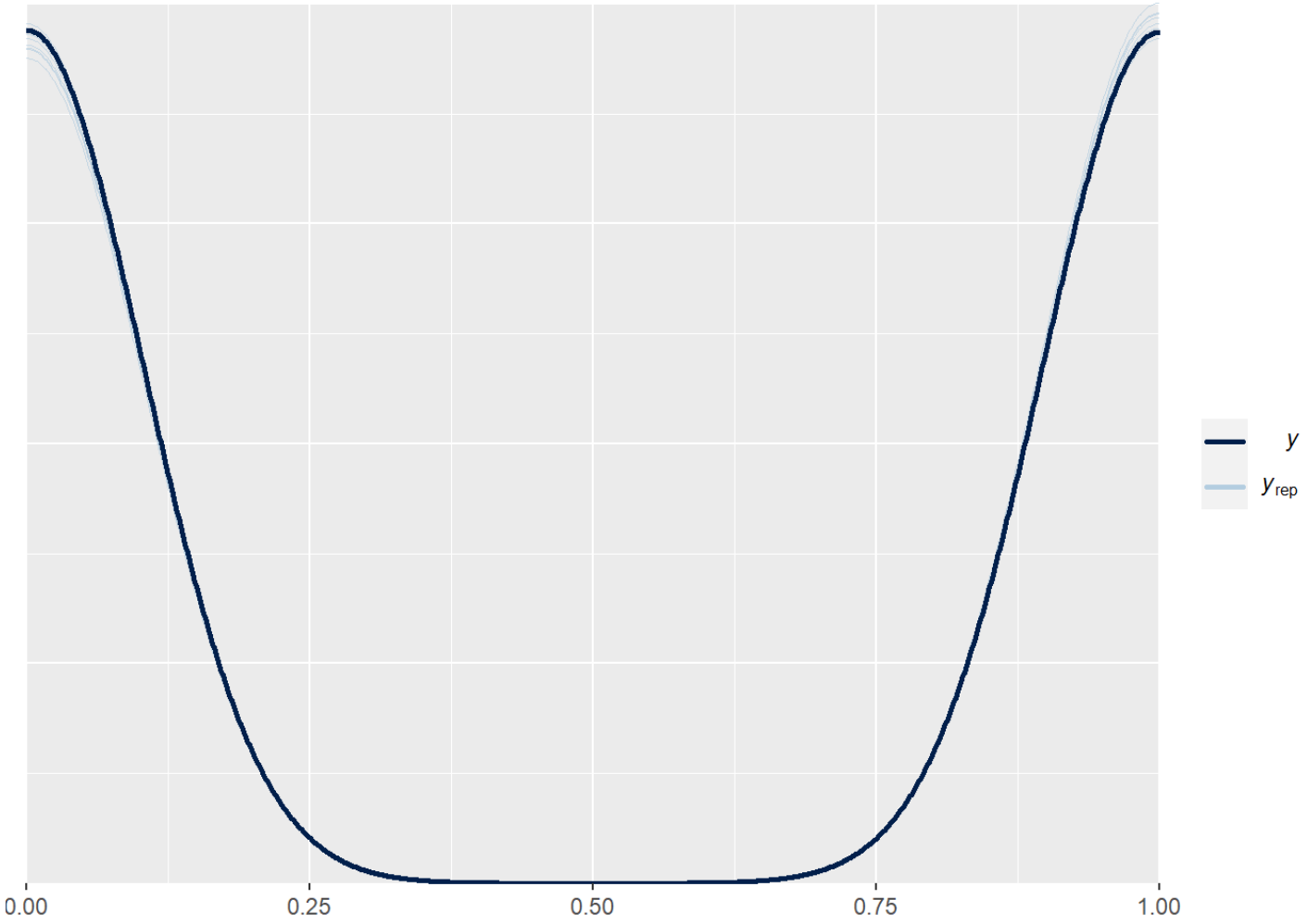

```
plot(pr_m1_fit_inf)
```


b_Intercept
b_V1
b_V2
b_V3
b_V4
b_V5
b_V6
b_V7
b_V8
b_V9
Chain
1
2
3
4

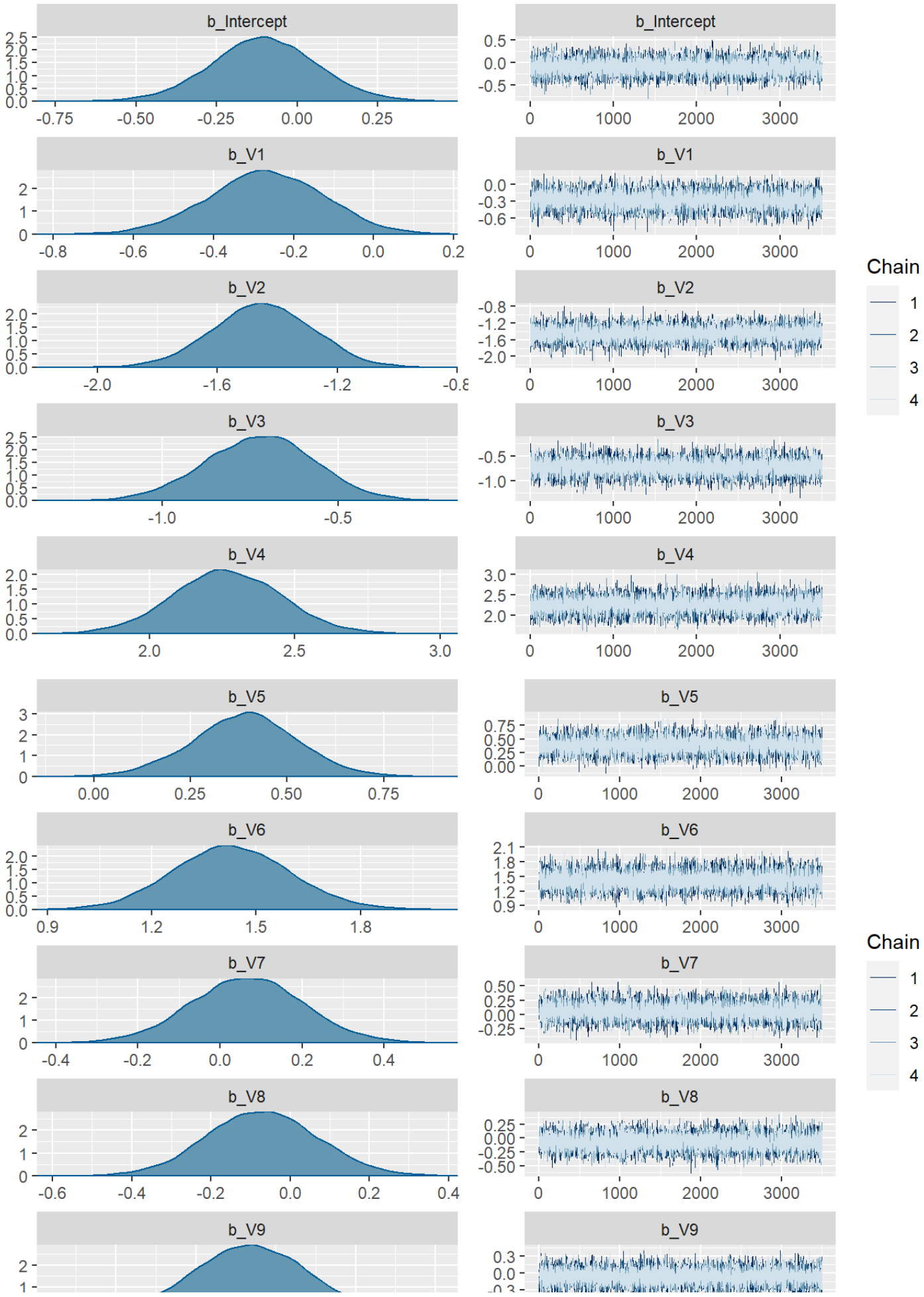


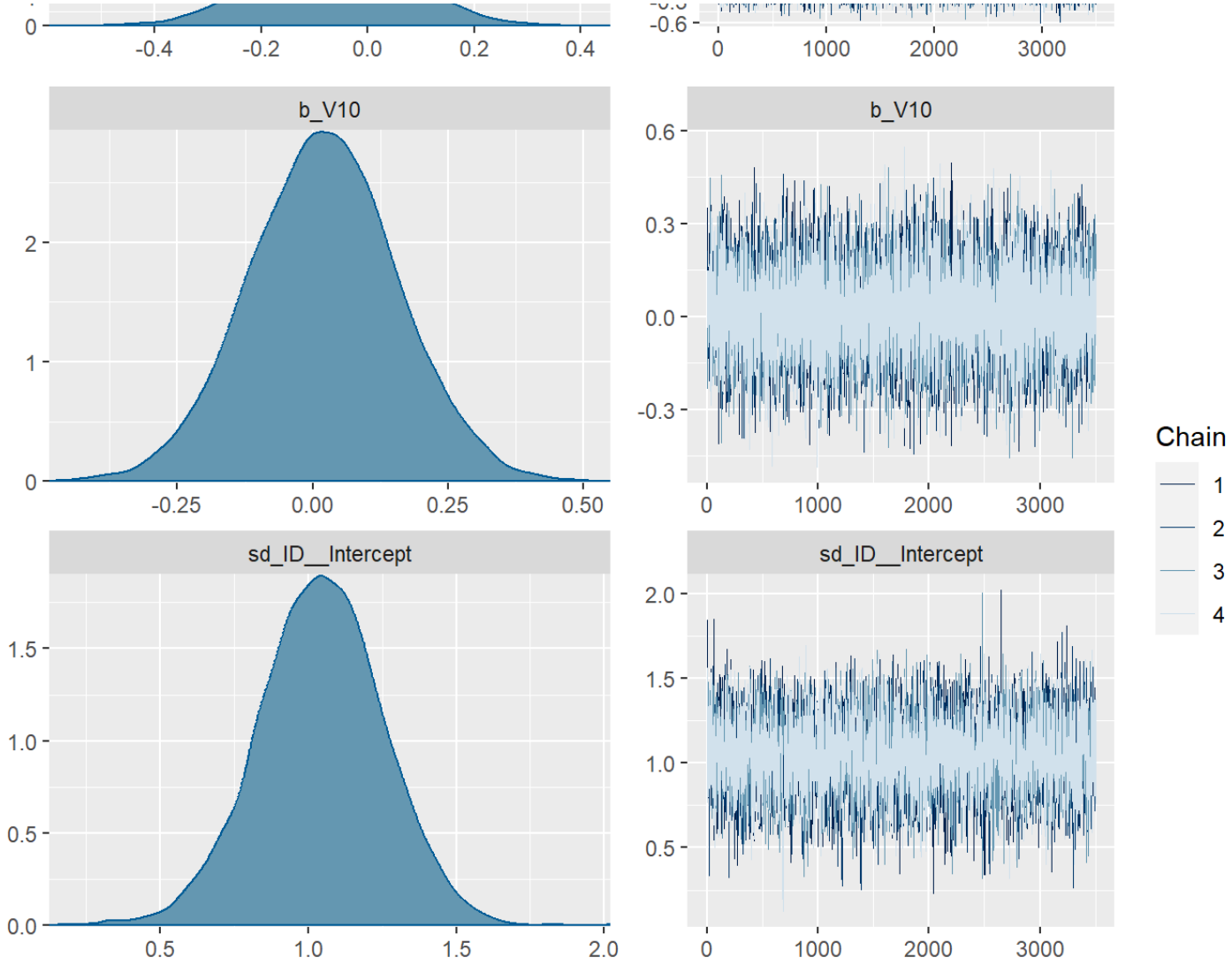

```
summary(pr_m1_fit_inf)
```

```
## second model fit on skeptical data
pr_m1_fit_skep <- brm(
  PR_f1,
  data = train_skeptic_s,
  prior = PR_p1,
  family = bernoulli,
  refresh=0,
  sample_prior = TRUE,
  iter=6000,
  warmup = 2500,
  backend = "cmdstanr",
  threads = threading(2),
  chains = 4,
  cores = 4,
  control = list(
    adapt_delta = 0.9,
    max_treedepth = 20)
)

pp_check(pr_m1_fit_skep)
```




```
plot(pr_m1_fit_skep)
```


b_Intercept
b_V1
b_V2
b_V3
b_V4
b_V5
b_V6
b_V7
b_V8
b_V9
Chain
1
2
3
4




```
summary(pr_m1_fit_skep)
```

##Ditlev ### Asses performance on test data

```
train_informed_s$PredictionsPerc0 <- predict(pr_m0_fit_inf)[, 1]
train_informed_s$Prediction0[train_informed_s$PredictionsPerc0 > 0.5] <- "Schizoph
renia"
```

```
## Warning: Unknown or uninitialised column: `Prediction0`.
```

```
train_informed_s$Prediction0[train_informed_s$PredictionsPerc0 <= 0.5] <- "Control
"


train_informed_s$PredictionsPerc1 <- predict(pr_m1_fit_inf)[, 1]
train_informed_s$Prediction1[train_informed_s$PredictionsPerc1 > 0.5] <- "Schizoph
renia"
```

```
## Warning: Unknown or uninitialised column: `Prediction1`.
```

```
train_informed_s$Prediction1[train_informed_s$PredictionsPerc1 <= 0.5] <- "Control
"


train_skeptic_s$PredictionsPerc0 <- predict(pr_m0_fit_skep)[, 1]
train_skeptic_s$Prediction0[train_skeptic_s$PredictionsPerc0 > 0.5] <- "Schizophre
nia"
```

```
## Warning: Unknown or uninitialised column: `Prediction0`.
```

```
train_skeptic_s$Prediction0[train_skeptic_s$PredictionsPerc0 <= 0.5] <- "Control"

train_skeptic_s$PredictionsPerc1 <- predict(pr_m1_fit_skep)[, 1]
train_skeptic_s$Prediction1[train_skeptic_s$PredictionsPerc1 > 0.5] <- "Schizophre
nia"
```

```
## Warning: Unknown or uninitialised column: `Prediction1`.
```

```
train_skeptic_s$Prediction1[train_skeptic_s$PredictionsPerc1 <= 0.5] <- "Control"

train_informed_s <- train_informed_s %>%
  mutate(
    Group = as.factor(Group),
    Prediction0 = as.factor(Prediction0),
    Prediction1 = as.factor(Prediction1)
  )

train_skeptic_s <- train_skeptic_s %>%
  mutate(
    Group = as.factor(Group),
    Prediction0 = as.factor(Prediction0),
    Prediction1 = as.factor(Prediction1)
  )


test_informed_s$PredictionsPerc0 <- predict(pr_m0_fit_inf, newdata = test_informed
_s, allow_new_levels = T)[, 1]
test_informed_s$Prediction0[test_informed_s$PredictionsPerc0 > 0.5] <- "Schizophre
nia"
```

```
## Warning: Unknown or uninitialised column: `Prediction0`.
```

```
test_informed_s$Prediction0[test_informed_s$PredictionsPerc0 <= 0.5] <- "Control"


test_informed_s$PredictionsPerc1 <- predict(pr_m1_fit_inf, newdata = test_informed
_s, allow_new_levels = T)[, 1]
test_informed_s$Prediction1[test_informed_s$PredictionsPerc1 > 0.5] <- "Schizophre
nia"
```

```
## Warning: Unknown or uninitialised column: `Prediction1`.
```

```
test_informed_s$Prediction1[test_informed_s$PredictionsPerc1 <= 0.5] <- "Control"


test_skeptic_s$PredictionsPerc0 <- predict(pr_m0_fit_skep, newdata = test_informed
_s, allow_new_levels = T)[, 1]
test_skeptic_s$Prediction0[test_skeptic_s$PredictionsPerc0 > 0.5] <- "Schizophreni
a"
```

```
## Warning: Unknown or uninitialised column: `Prediction0`.
```

```
test_skeptic_s$Prediction0[test_skeptic_s$PredictionsPerc0 <= 0.5] <- "Control"

test_skeptic_s$PredictionsPerc1 <- predict(pr_m1_fit_skep, newdata = test_informed
_s, allow_new_levels = T)[, 1]
test_skeptic_s$Prediction1[test_skeptic_s$PredictionsPerc1 > 0.5] <- "Schizophreni
a"
```

```
## Warning: Unknown or uninitialised column: `Prediction1`.
```

```
test_skeptic_s$Prediction1[test_skeptic_s$PredictionsPerc1 <= 0.5] <- "Control"

test_informed_s <- test_informed_s %>%
  mutate(
    Group = as.factor(Group),
    Prediction0 = as.factor(Prediction0),
    Prediction1 = as.factor(Prediction1)
  )

test_skeptic_s <- test_skeptic_s %>%
  mutate(
    Group = as.factor(Group),
    Prediction0 = as.factor(Prediction0),
    Prediction1 = as.factor(Prediction1)
  )
```

```
confusionMatrix(train_skeptic_s$Group, train_skeptic_s$Prediction0)
```

```
## Confusion Matrix and Statistics
##
##                Reference
## Prediction      Control Schizophrenia
##   Control           778            23
##   Schizophrenia      15           784
##
##                Accuracy : 0.9762
##                  95% CI : (0.9675, 0.9831)
##     No Information Rate : 0.5044
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9525
##
##  Mcnemar's Test P-Value : 0.2561
##
##             Sensitivity : 0.9811
##             Specificity : 0.9715
##          Pos Pred Value : 0.9713
##          Neg Pred Value : 0.9812
##              Prevalence : 0.4956
##          Detection Rate : 0.4863
##    Detection Prevalence : 0.5006
##       Balanced Accuracy : 0.9763
##
##        'Positive' Class : Control
##
```

```
confusionMatrix(test_skeptic_s$Group, test_skeptic_s$Prediction0)
```

```
## Confusion Matrix and Statistics
##
##                Reference
## Prediction      Control Schizophrenia
##   Control           198              9
##   Schizophrenia       5            188
##
##                Accuracy : 0.965
##                  95% CI : (0.942, 0.9807)
##     No Information Rate : 0.5075
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.93
##
##  Mcnemar's Test P-Value : 0.4227
##
##             Sensitivity : 0.9754
##             Specificity : 0.9543
##          Pos Pred Value : 0.9565
##          Neg Pred Value : 0.9741
##              Prevalence : 0.5075
##          Detection Rate : 0.4950
##    Detection Prevalence : 0.5175
##       Balanced Accuracy : 0.9648
##
##        'Positive' Class : Control
##
```

```
confusionMatrix(train_skeptic_s$Group, train_skeptic_s$Prediction1)
```

```
## Confusion Matrix and Statistics
##
##                Reference
## Prediction      Control Schizophrenia
##   Control           797              4
##   Schizophrenia       2            797
##
##                Accuracy : 0.9962
##                  95% CI : (0.9919, 0.9986)
##     No Information Rate : 0.5006
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9925
##
##  Mcnemar's Test P-Value : 0.6831
##
##             Sensitivity : 0.9975
##             Specificity : 0.9950
##          Pos Pred Value : 0.9950
##          Neg Pred Value : 0.9975
##              Prevalence : 0.4994
##          Detection Rate : 0.4981
##    Detection Prevalence : 0.5006
##       Balanced Accuracy : 0.9963
##
##        'Positive' Class : Control
##
```

```
confusionMatrix(test_skeptic_s$Group, test_skeptic_s$Prediction1)
```

```
## Confusion Matrix and Statistics
##
##                Reference
## Prediction      Control Schizophrenia
##   Control           199              8
##   Schizophrenia       2            191
##
##                Accuracy : 0.975
##                  95% CI : (0.9545, 0.9879)
##     No Information Rate : 0.5025
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.95
##
##  Mcnemar's Test P-Value : 0.1138
##
##             Sensitivity : 0.9900
##             Specificity : 0.9598
##          Pos Pred Value : 0.9614
##          Neg Pred Value : 0.9896
##              Prevalence : 0.5025
##          Detection Rate : 0.4975
##    Detection Prevalence : 0.5175
##       Balanced Accuracy : 0.9749
##
##        'Positive' Class : Control
##
```

```
confusionMatrix(train_informed_s$Group, train_informed_s$Prediction0)
```

```
## Confusion Matrix and Statistics
##
##                Reference
## Prediction      Control Schizophrenia
##   Control           777            24
##   Schizophrenia      15           784
##
##                Accuracy : 0.9756
##                  95% CI : (0.9668, 0.9826)
##     No Information Rate : 0.505
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9513
##
##  Mcnemar's Test P-Value : 0.2002
##
##             Sensitivity : 0.9811
##             Specificity : 0.9703
##          Pos Pred Value : 0.9700
##          Neg Pred Value : 0.9812
##              Prevalence : 0.4950
##          Detection Rate : 0.4856
##    Detection Prevalence : 0.5006
##       Balanced Accuracy : 0.9757
##
##        'Positive' Class : Control
##
```

```
confusionMatrix(test_informed_s$Group, test_informed_s$Prediction0)
```

```
## Confusion Matrix and Statistics
##
##                Reference
## Prediction      Control Schizophrenia
##   Control           198             1
##   Schizophrenia       5           196
##
##                Accuracy : 0.985
##                  95% CI : (0.9676, 0.9945)
##     No Information Rate : 0.5075
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.97
##
##  Mcnemar's Test P-Value : 0.2207
##
##             Sensitivity : 0.9754
##             Specificity : 0.9949
##          Pos Pred Value : 0.9950
##          Neg Pred Value : 0.9751
##              Prevalence : 0.5075
##          Detection Rate : 0.4950
##    Detection Prevalence : 0.4975
##       Balanced Accuracy : 0.9851
##
##        'Positive' Class : Control
##
```

```
confusionMatrix(train_informed_s$Group, train_informed_s$Prediction1)
```

```
## Confusion Matrix and Statistics
##
##                Reference
## Prediction      Control Schizophrenia
##   Control           797              4
##   Schizophrenia       2            797
##
##                Accuracy : 0.9962
##                  95% CI : (0.9919, 0.9986)
##     No Information Rate : 0.5006
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9925
##
##  Mcnemar's Test P-Value : 0.6831
##
##             Sensitivity : 0.9975
##             Specificity : 0.9950
##          Pos Pred Value : 0.9950
##          Neg Pred Value : 0.9975
##              Prevalence : 0.4994
##          Detection Rate : 0.4981
##    Detection Prevalence : 0.5006
##       Balanced Accuracy : 0.9963
##
##        'Positive' Class : Control
##
```

```
confusionMatrix(test_informed_s$Group, test_informed_s$Prediction1)
```

```
## Confusion Matrix and Statistics
##
##                Reference
## Prediction      Control Schizophrenia
##   Control           199              0
##   Schizophrenia       2            199
##
##                Accuracy : 0.995
##                  95% CI : (0.9821, 0.9994)
##     No Information Rate : 0.5025
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.99
##
##  Mcnemar's Test P-Value : 0.4795
##
##             Sensitivity : 0.9900
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.9900
##              Prevalence : 0.5025
##          Detection Rate : 0.4975
##    Detection Prevalence : 0.4975
##       Balanced Accuracy : 0.9950
##
##        'Positive' Class : Control
##
```

Discuss whether performance and feature importance is as expected

##Part 3 - Applying the machine learning pipeline to empirical data

```
real_data <- read_csv("assignment_3_data.csv")
```

```
## Rows: 1889 Columns: 398
## ── Column specification ────────────────────────────────────────────────────
─
## Delimiter: ","
## chr   (5): NewID, Diagnosis, Language, Gender, Trial
## dbl (393): PatID, Corpus, Duration_Praat, F0_Mean_Praat, F0_SD_Praat, Intens...
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this messag
e.
```

```
real_data <- real_data %>%
  select(-Language) %>%
  select(-Corpus) %>%
  select(-NewID)
```

Apply your machine learning pipeline to the empirical data

Warning: in simulated data you only have 10 features, now you have many more - Consider the impact a higher number of features will have on your ML inference and decide if you want to cut down the number of features before running the pipeline (alternatibvely expand the pipeline to add feature selection)

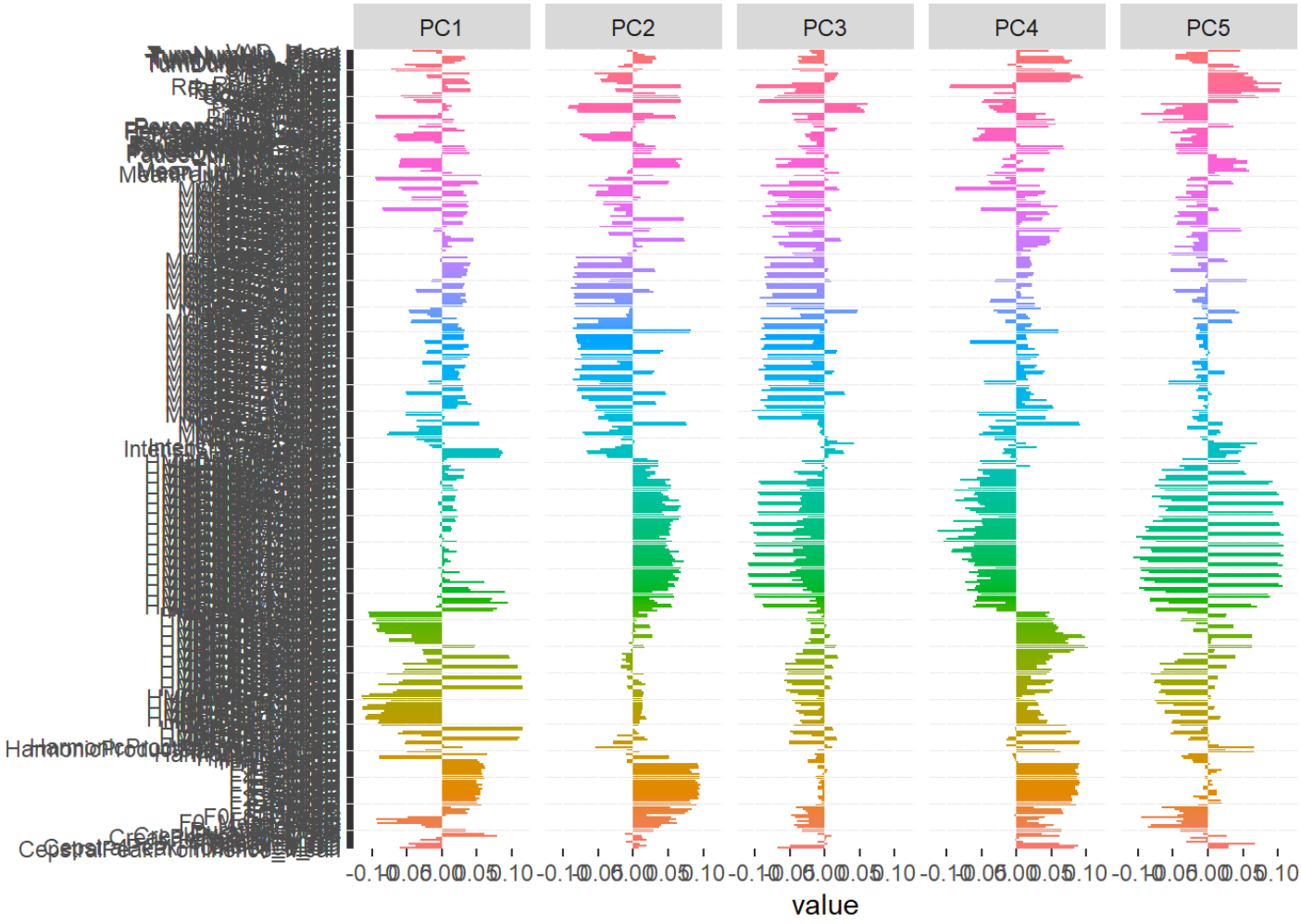
data budgeting and pree-processing of the data

```
set.seed(304)

split_real_data <- initial_split(real_data, prop = 4/5, strata = Gender)

train_data <- training(split_real_data)
test_data <- testing(split_real_data)

train_data <- train_data %>%
  select(-Gender) %>%
  select(-Trial)

test_data <- test_data %>%
  select(-Gender) %>%
  select(-Trial)

train_data$PatID <- as.factor(train_data$PatID)

test_data$PatID <- as.factor(test_data$PatID)
```

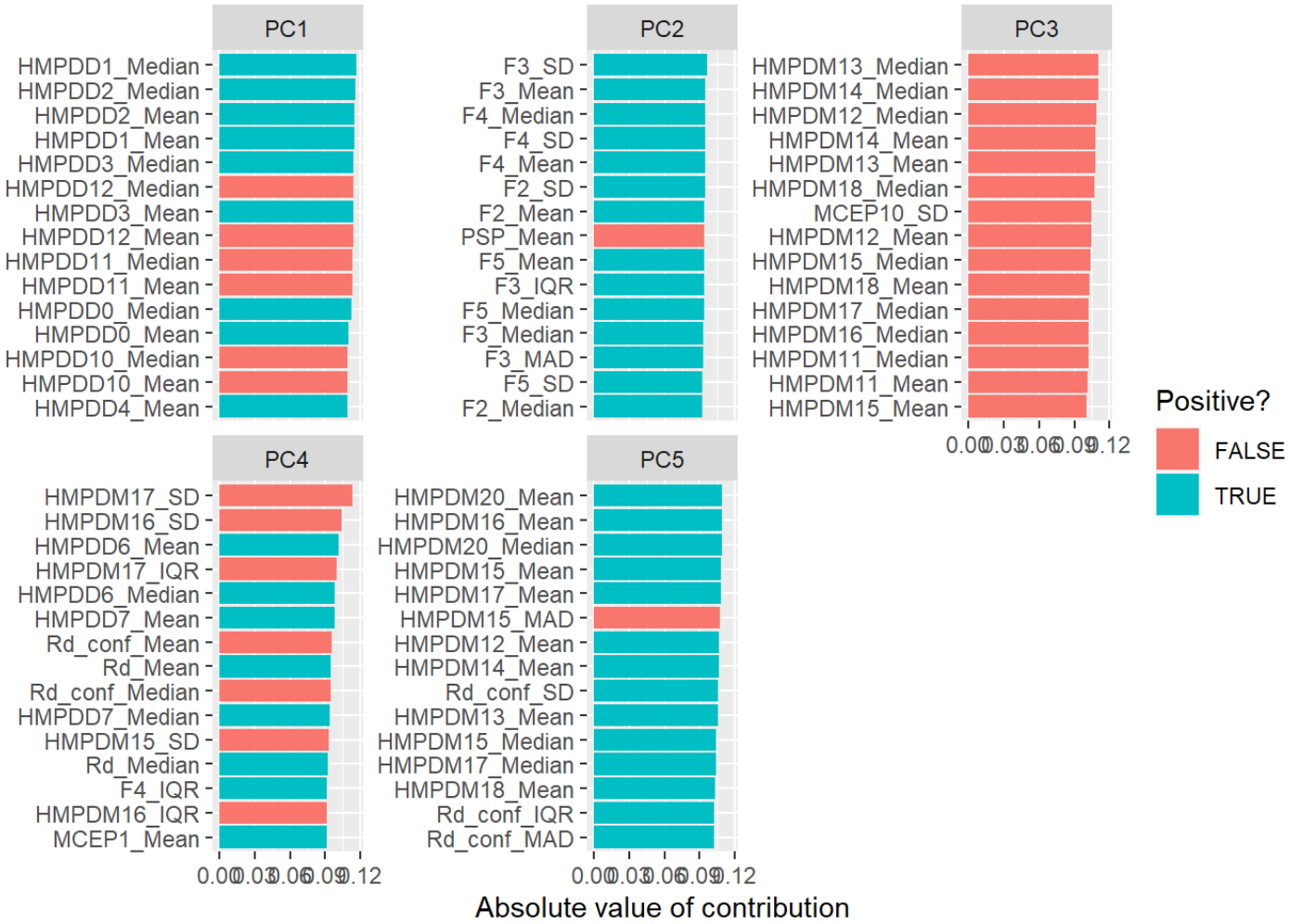
```
recipe_real <- train_data %>%
  recipe(Diagnosis~.) %>%
  update_role(PatID, new_role = 'PatID') %>%
  step_scale(all_numeric()) %>%
  step_center(all_numeric()) %>%
  prep(training=train_data, retain=TRUE)

train_data_s <- juice(recipe_real)
test_data_s <- bake(recipe_real, new_data = test_data)
```

Principle component analysis

```
pca_recipe <- recipe(Diagnosis~., data = train_data_s) %>%
  update_role(PatID, new_role = "id") %>%
  step_pca(all_numeric(), id = "pca")%>%
  prep()

tidy_pca <- tidy(pca_recipe, 1)

bake_pca <- bake(pca_recipe, train_data_s)

pca_b_test <- bake(pca_recipe, test_data_s)

tidy_pca %>%
  filter(component %in% paste0("PC", 1:5)) %>%
  mutate(component = fct_inorder(component)) %>%
  ggplot(aes(value, terms, fill = terms)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~component, nrow = 1) +
  labs(y = NULL)
```



```
tidy_pca %>%
  filter(component %in% paste0("PC", 1:5)) %>%
  group_by(component) %>%
  top_n(15, abs(value)) %>%
  ungroup() %>%
  mutate(terms = reorder_within(terms, abs(value), component)) %>%
  ggplot(aes(abs(value), terms, fill = value > 0)) +
  geom_col() +
  facet_wrap(~component, scales = "free_y") +
  scale_y_reordered() +
  labs(
    x = "Absolute value of contribution",
    y = NULL, fill = "Positive?"
  )
```



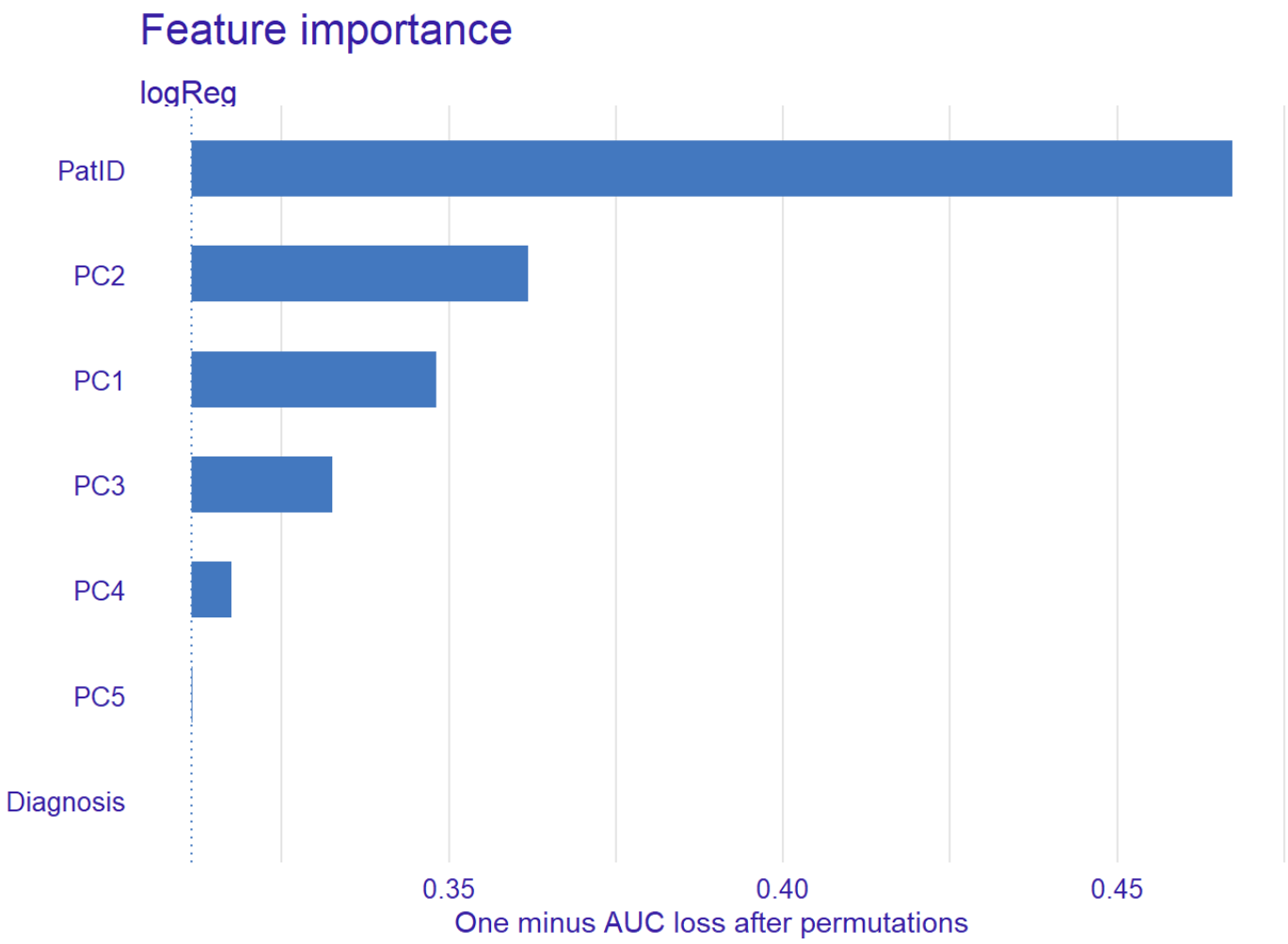

```
??reorder_within
```

```
## starting httpd help server ... done
```

##Manuela ### feature selection

```
d_data <- bake_pca %>%
  mutate(PatID = NULL, Trail = NULL, NewID = NULL, Gender = NULL, Language = NULL,
Corpus = NULL)

LogisticRegression_train<- logistic_reg() %>%
  set_mode('classification') %>%
  set_engine('glm') %>%
  fit(Diagnosis ~ . , data = bake_pca)
```

```
explainer_lm <-
  explain_tidymodels(
    LogisticRegression_train,
    data = bake_pca,
    y = as.numeric(d_data$Diagnosis) -1,
    label = 'logReg',
    verbose = FALSE
  )

explainer_lm %>% model_parts() %>%  plot(show_boxplots = FALSE) + ggtitle('Feature
importance', '')
```



fit and assess a classification algorithm on the training data

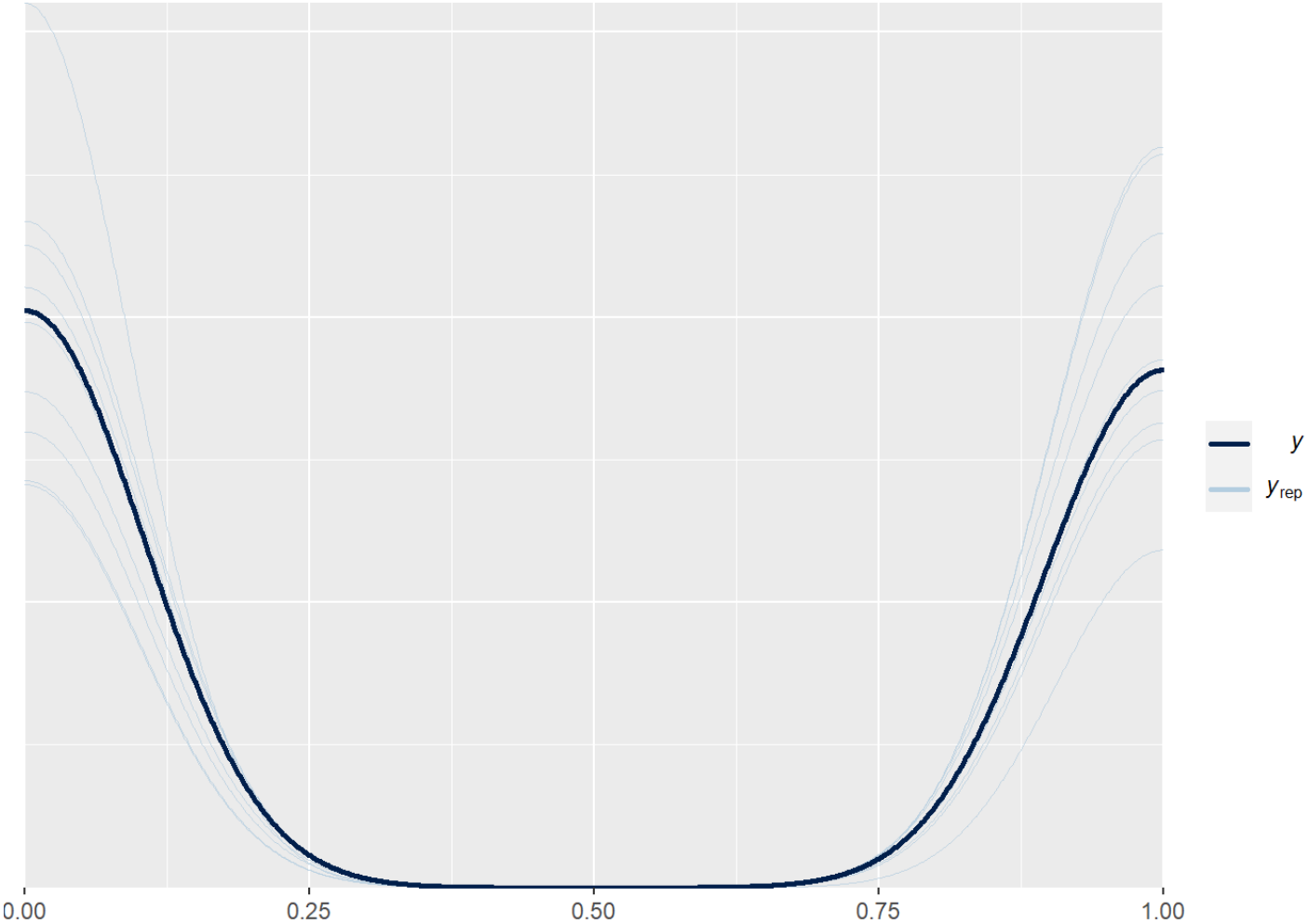
```
form1 <- bf(Diagnosis~1+PC2+PC1+PC3+PC4+(1|PatID))
get_prior(form1, bake_pca, family = bernoulli)
```

```
##                 prior     class      coef group resp dpar nlpar lb ub
##                (flat)         b
##                (flat)         b       PC1
##                (flat)         b       PC2
##                (flat)         b       PC3
##                (flat)         b       PC4
##  student_t(3, 0, 2.5) Intercept
##  student_t(3, 0, 2.5)        sd                                   0
##  student_t(3, 0, 2.5)        sd           PatID                   0
##  student_t(3, 0, 2.5)        sd Intercept PatID                   0
##        source
##       default
##  (vectorized)
##  (vectorized)
##  (vectorized)
##  (vectorized)
##       default
##       default
##  (vectorized)
##  (vectorized)
```

```
prior_f1 <- c(
  prior(normal(0, 1), class=Intercept),
  prior(normal(0, 1), class=sd),
  prior(normal(0, 0.3), class=b)
)
```

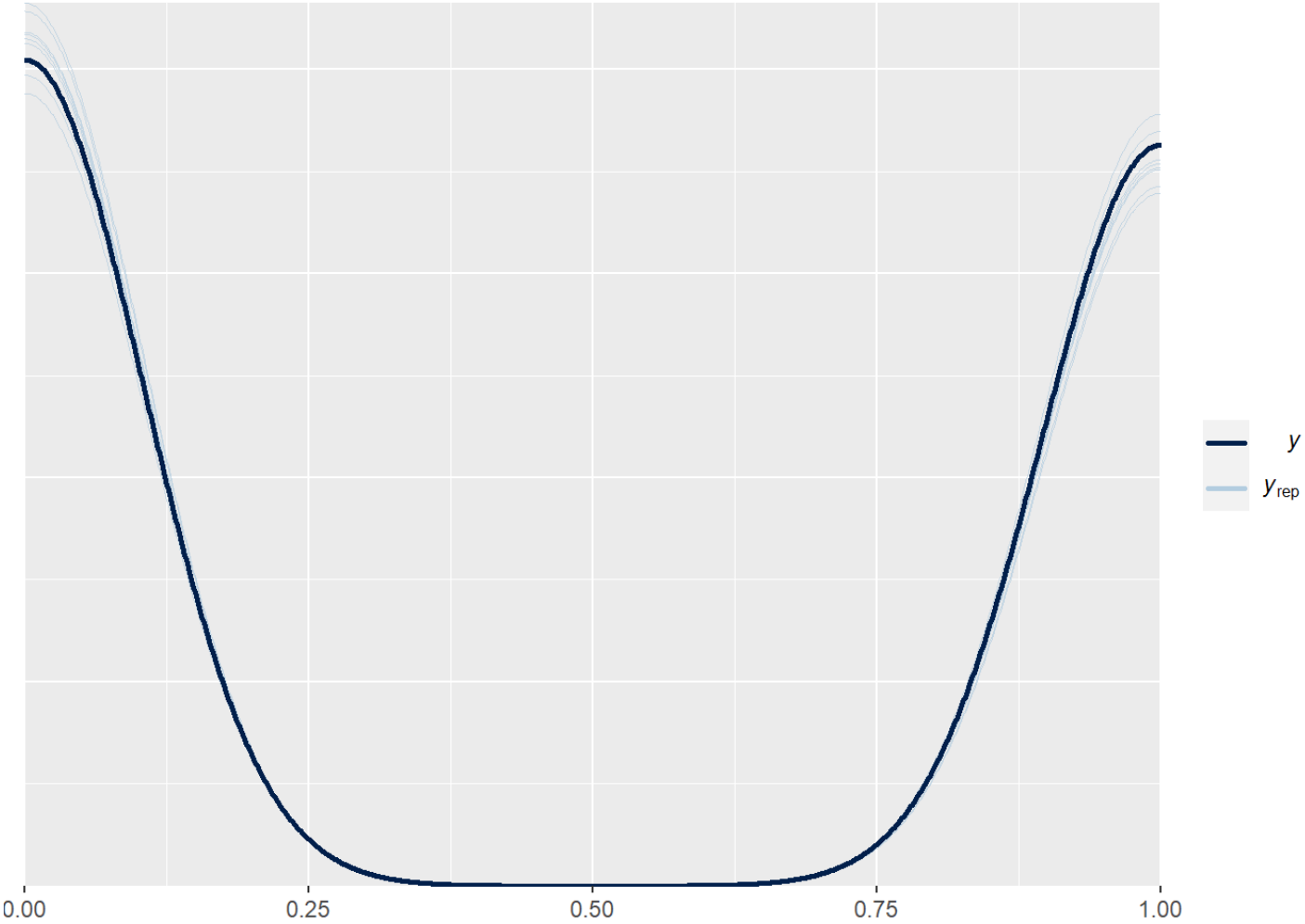
Fitting the model with only the priors

```
pr_m1 <- brm(
  form1,
  data = bake_pca,
  prior = prior_f1,
  family = bernoulli,
  refresh=0,
  sample_prior = 'only',
  iter=6000,
  warmup = 2500,
  backend = "cmdstanr",
  threads = threading(2),
  chains = 4,
  cores = 4,
  control = list(
    adapt_delta = 0.9,
    max_treedepth = 20)
)

pp_check(pr_m1)
```

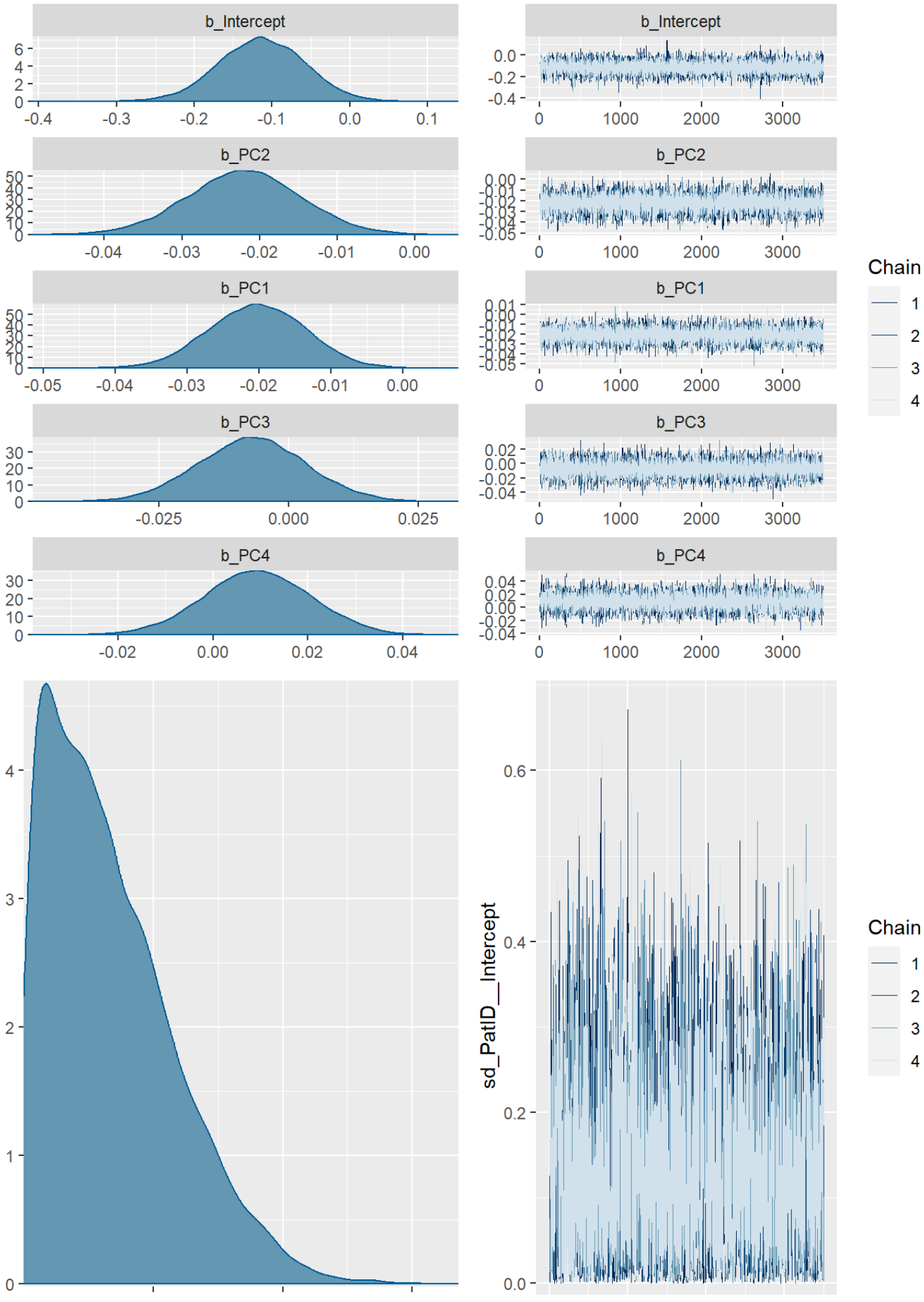


Fitting model on the data

```
pr_m1_fit <- brm(
  form1,
  data = bake_pca,
  prior = prior_f1,
  family = bernoulli,
  refresh=0,
  sample_prior = TRUE,
  iter=6000,
  warmup = 2500,
  backend = "cmdstanr",
  threads = threading(2),
  chains = 4,
  cores = 4,
  control = list(
    adapt_delta = 0.9,
    max_treedepth = 20)
)

pp_check(pr_m1_fit)
```



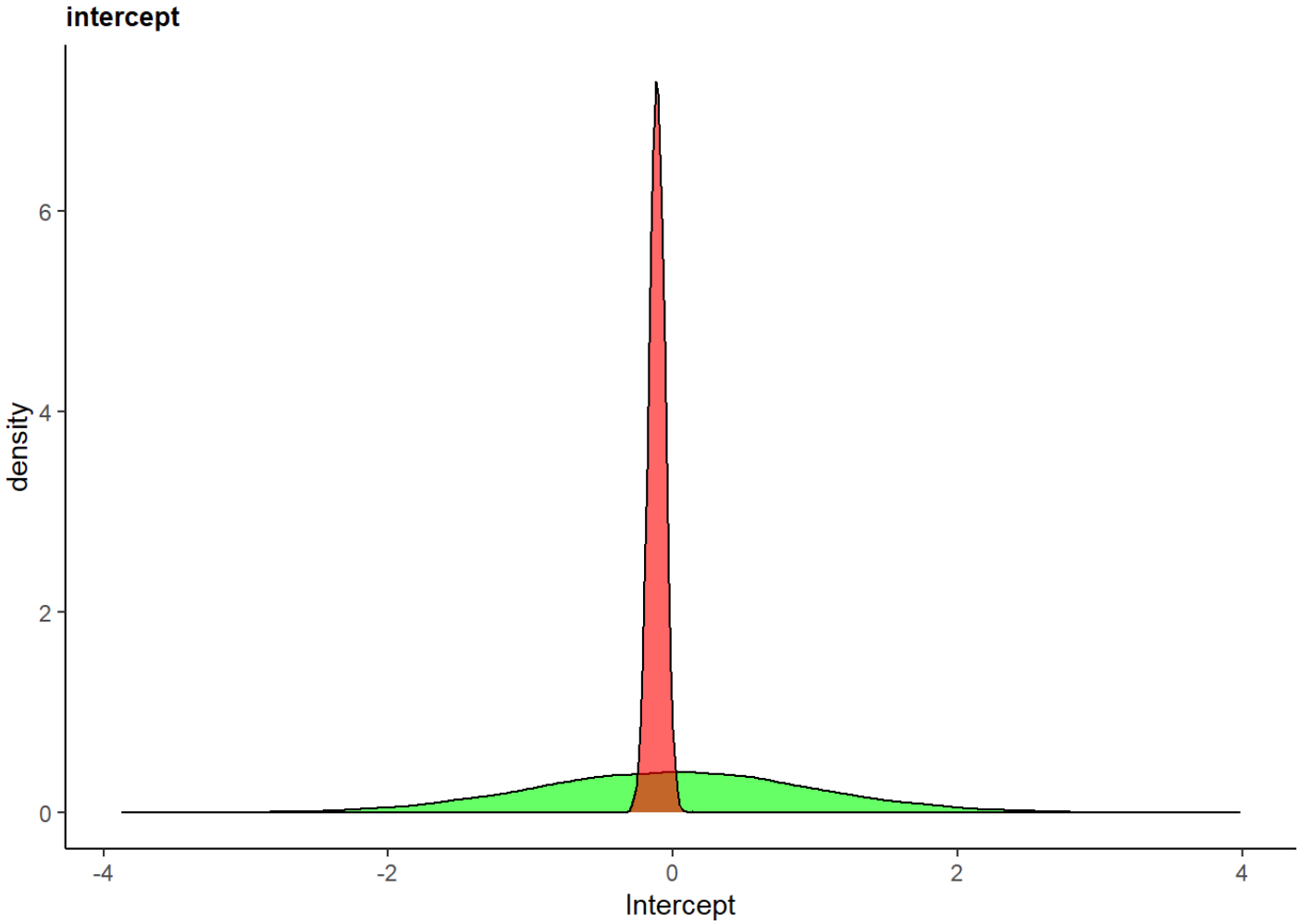

```
plot(pr_m1_fit)
```


b_Intercept
b_PC2
b_PC1
b_PC3
b_PC4
Chain
1
2
3
4
sd_PatID__Intercept


```
summary(pr_m1_fit)
```

Prior posterior update check

```
Posterior_f1 <- as_draws_df(pr_m1_fit)

ggplot(Posterior_f1) +
  geom_density(aes(prior_Intercept), fill="green", color="black",alpha=0.6) +
  geom_density(aes(b_Intercept), fill="red", color="black",alpha=0.6) +
  xlab('Intercept') +
  theme_classic()+
  ggtitle("intercept")+
      theme(plot.title = element_text(size = 10, face = "bold"))
```



assess performance on the test set

```
bake_pca$PredictionsPerc1 <- predict(pr_m1_fit)[, 1]

bake_pca$Prediction1[bake_pca$PredictionsPerc1 > 0.5] <- "SCZ"
```

```
## Warning: Unknown or uninitialised column: `Prediction1`.
```

```
bake_pca$Prediction1[bake_pca$PredictionsPerc1 <= 0.5] <- "CT"

bake_pca <- bake_pca %>%
  mutate(
    Diagnosis = as.factor(Diagnosis),
    Prediction1 = as.factor(Prediction1))

pca_b_test$PredictionsPerc1 <- predict(pr_m1_fit, newdata = pca_b_test, allow_new_
levels = T)[, 1]



pca_b_test$Prediction1[pca_b_test$PredictionsPerc1 > 0.5] <- "SCZ"
```

```
## Warning: Unknown or uninitialised column: `Prediction1`.
```

```
pca_b_test$Prediction1[pca_b_test$PredictionsPerc1 <= 0.5] <- "CT"

pca_b_test <- pca_b_test %>%
  mutate(
    Diagnosis = as.factor(Diagnosis),
    Prediction1 = as.factor(Prediction1)
  )
```

Accuracy

```
confusionMatrix(bake_pca$Diagnosis, bake_pca$Prediction1)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  CT SCZ
##        CT  545 251
##        SCZ 444 270
##
##                Accuracy : 0.5397
##                  95% CI : (0.5142, 0.5651)
##     No Information Rate : 0.655
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0637
##
##  Mcnemar's Test P-Value : 3.265e-13
##
##             Sensitivity : 0.5511
##             Specificity : 0.5182
##          Pos Pred Value : 0.6847
##          Neg Pred Value : 0.3782
##              Prevalence : 0.6550
##          Detection Rate : 0.3609
##    Detection Prevalence : 0.5272
##       Balanced Accuracy : 0.5346
##
##        'Positive' Class : CT
##
```

```
confusionMatrix(pca_b_test$Diagnosis, pca_b_test$Prediction1)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  CT SCZ
##        CT  115  78
##        SCZ 104  82
##
##                Accuracy : 0.5198
##                  95% CI : (0.4682, 0.5711)
##     No Information Rate : 0.5778
##     P-Value [Acc > NIR] : 0.99011
##
##                   Kappa : 0.0368
##
##  Mcnemar's Test P-Value : 0.06386
##
##             Sensitivity : 0.5251
##             Specificity : 0.5125
##          Pos Pred Value : 0.5959
##          Neg Pred Value : 0.4409
##              Prevalence : 0.5778
##          Detection Rate : 0.3034
##    Detection Prevalence : 0.5092
##       Balanced Accuracy : 0.5188
##
##        'Positive' Class : CT
##
```

###discuss whether performance and feature importance is as expected