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

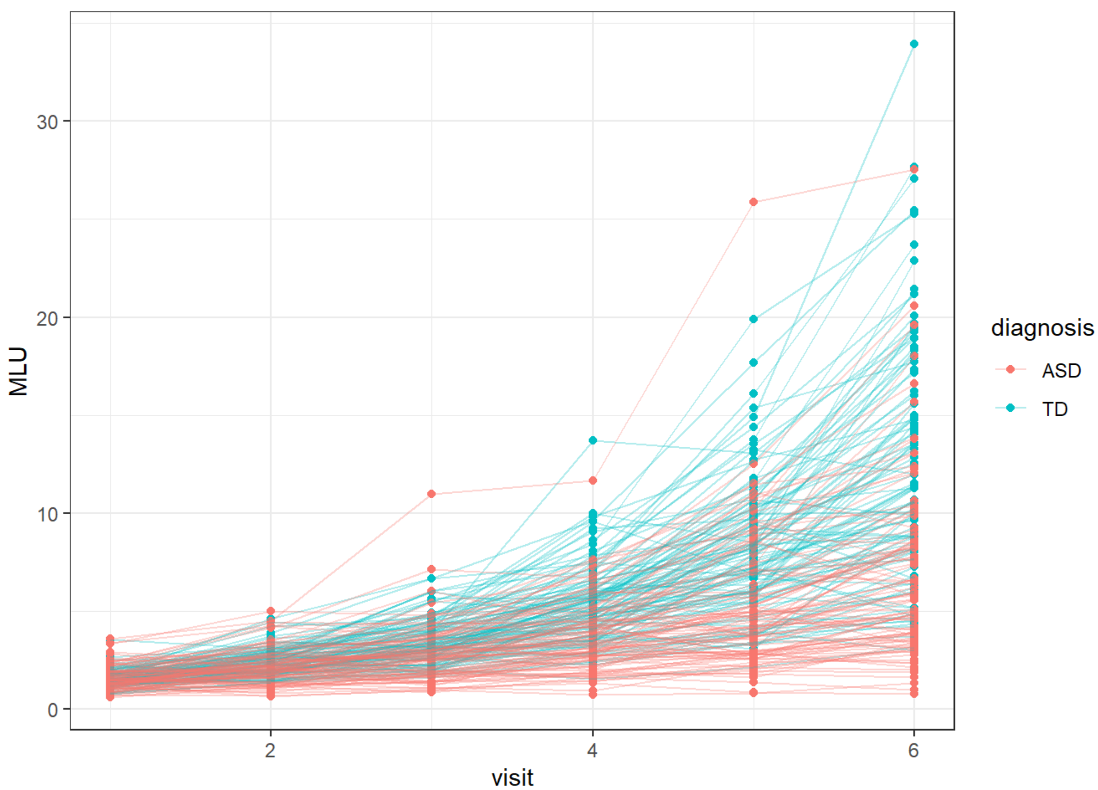
```
average_mlu <- log(1.5)
 sd_mlu_asd <- log(1.5+0.5)-log(1.5)
 sd_mlu_td <- log(1.5+0.3)-log(1.5)

 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)
```

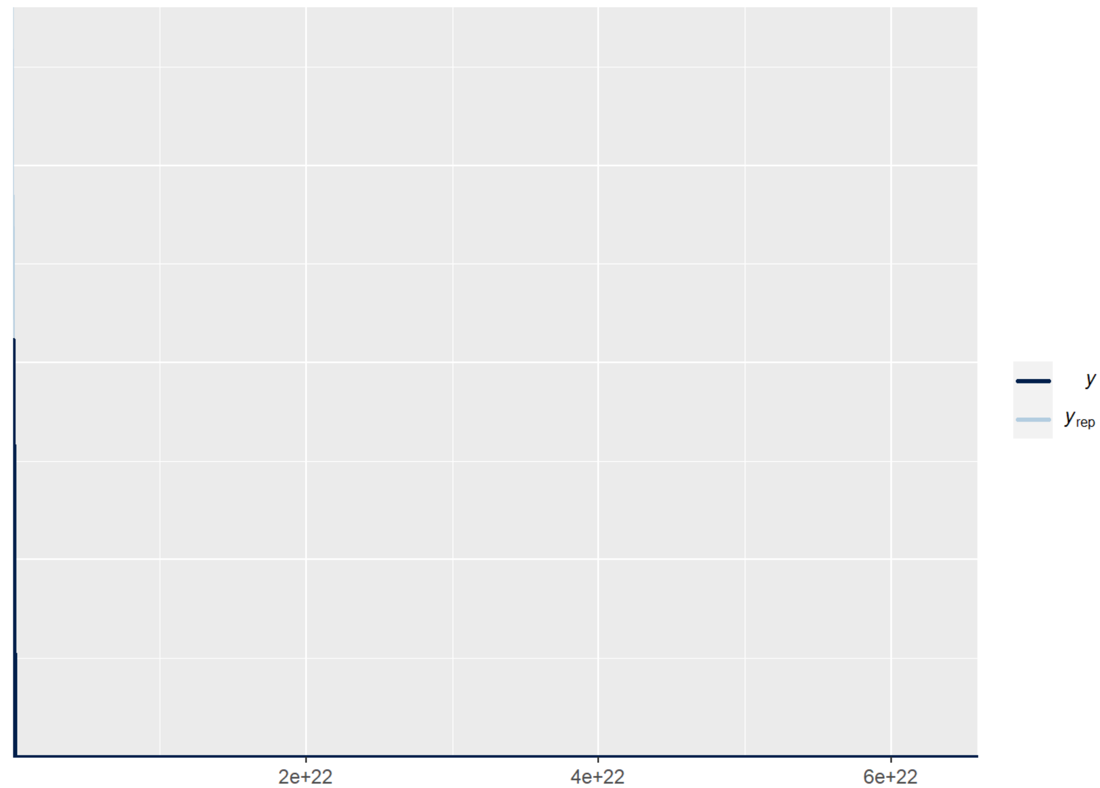
```
priors <- c(
prior(normal(1.5,0.5),class=b,coef="diagnosisASD"),
prior(normal(1.5,0.3),class=b,coef="diagnosisTD"),
prior(normal(0,0.5),class=b),
prior(normal(0,0.5),class=sd),
prior(lkj(2),class=cor))
```

###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)
```



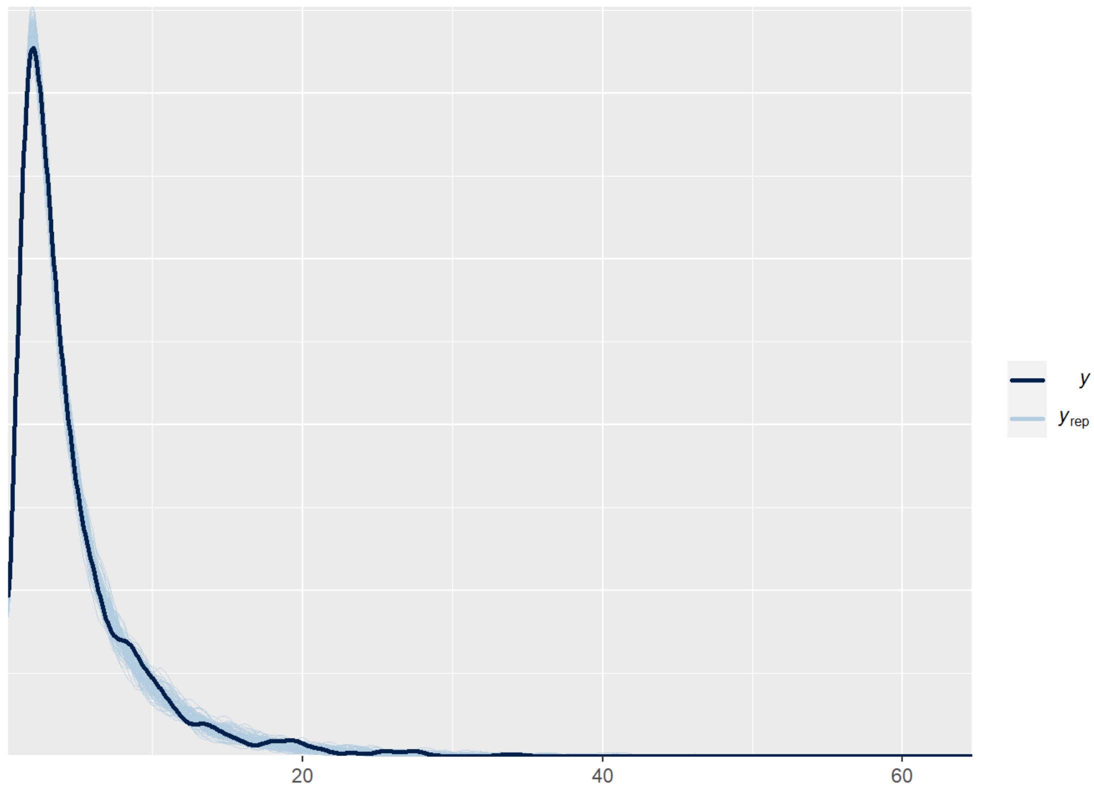
###fit the model

```
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
```

```
)
)
```

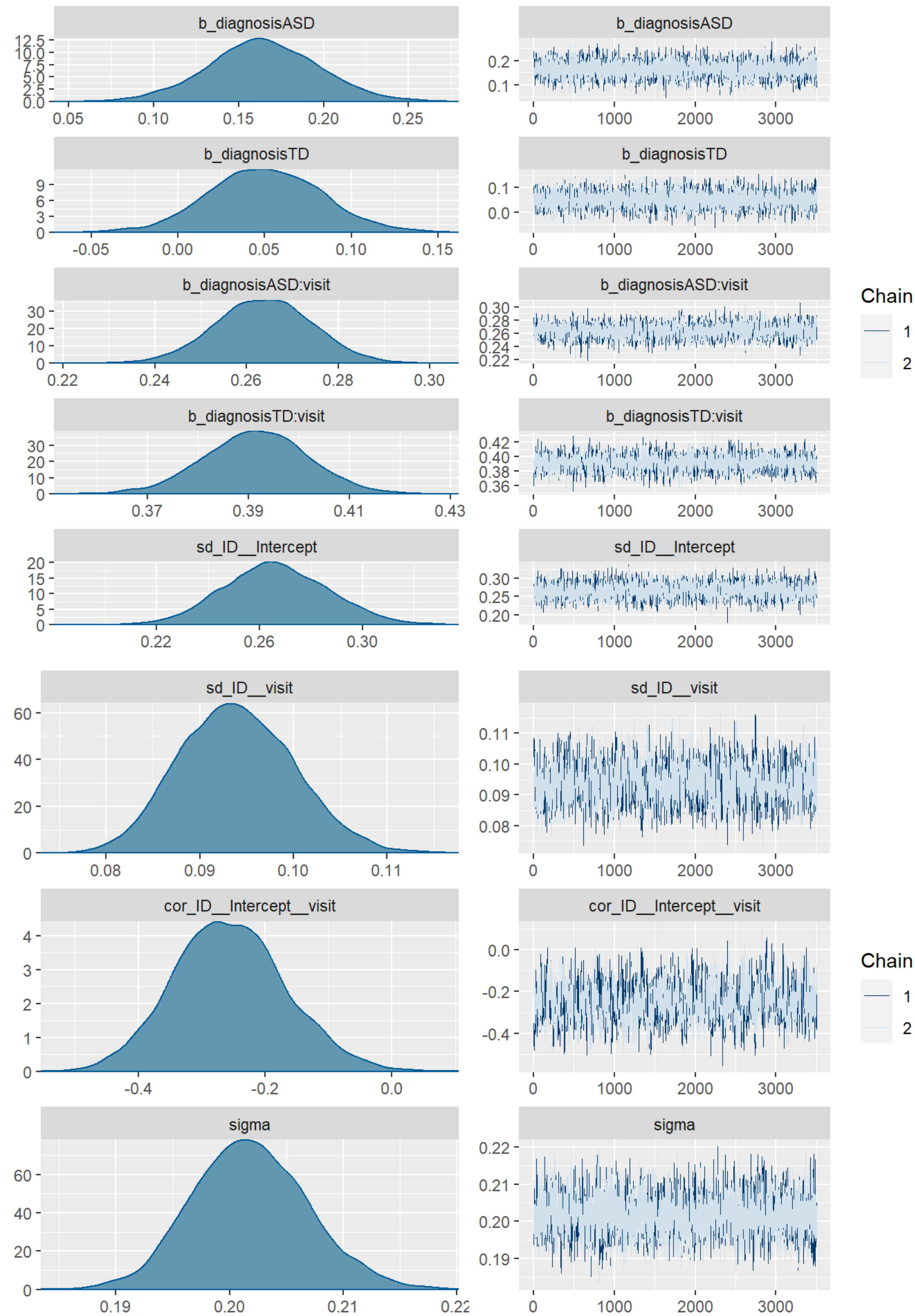
###posterior predictive check

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



###traceplot for fitted model

```
plot(MLU_prior_m1_fit)
```



Ditlev ### parameter recovery from fitted model

```r
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
```

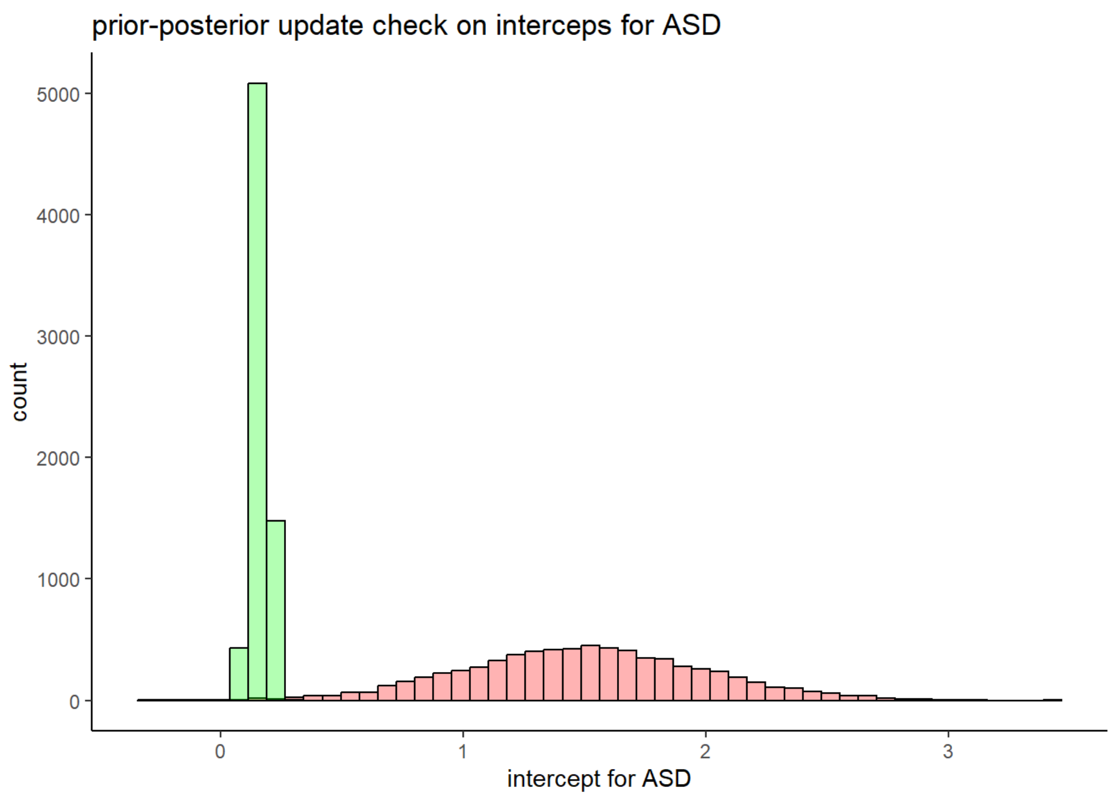
```
0.41 1.00     4086     4820
##
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat
Bulk_ESS Tail_ESS
## sigma     0.20      0.01     0.19     0.21 1.00
4407     4653
##
## Draws were sampled using sample(hmc). For each
parameter, Bulk_ESS
## and Tail_ESS are effective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

prior posterior update check

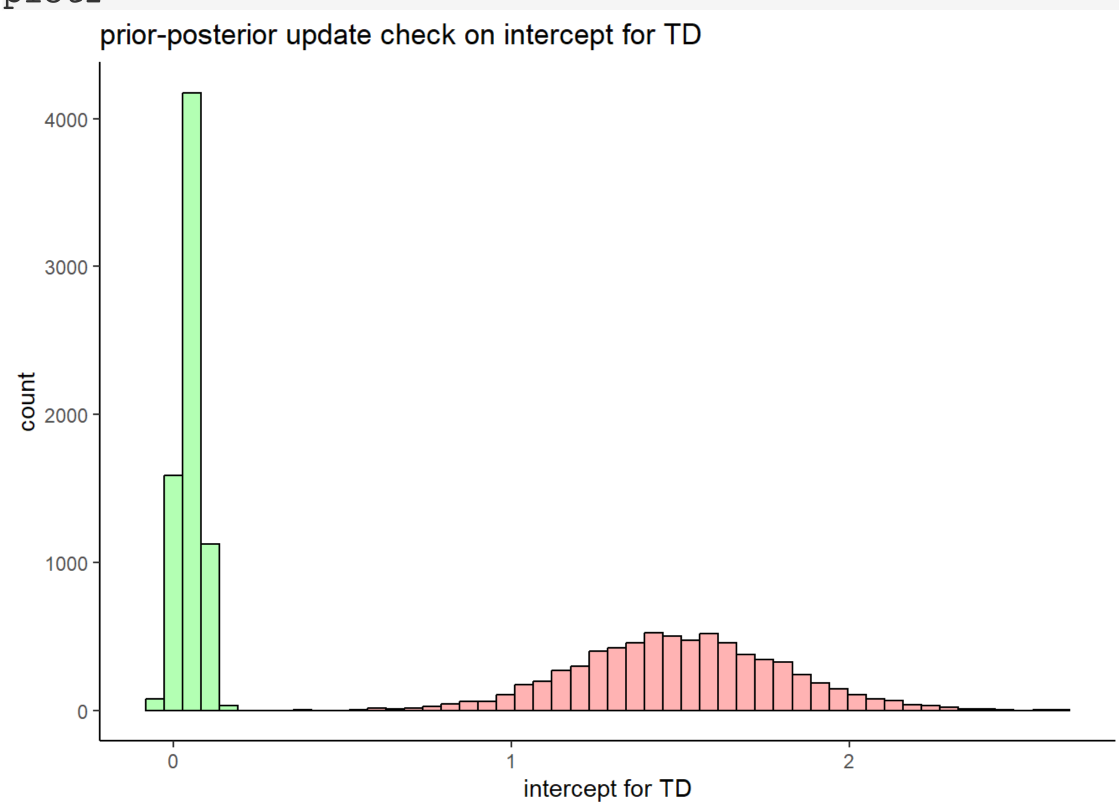
```
posterior <- as_draws_df(MLU_prior_m1_fit)

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)+
```

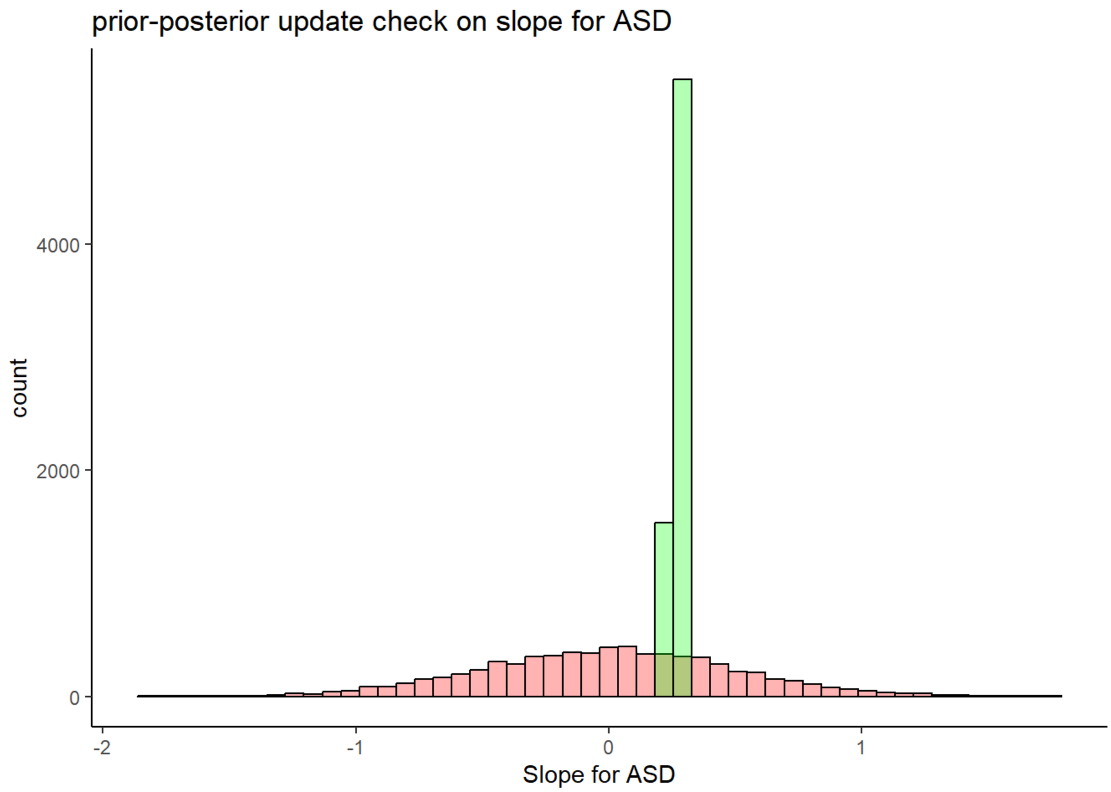
```
  geom_histogram(aes(prior_sd_ID), fill='red',
color='black', alpha=0.3, bins=50)+
  geom_histogram(aes(sd_ID__Intercept), fill='green',
color='black', alpha=0.3, bins=50)+
  theme_classic()+
  ggtitle('Prior-posterior update check, the
variability of the intercept')+
  xlab("Intercept")

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

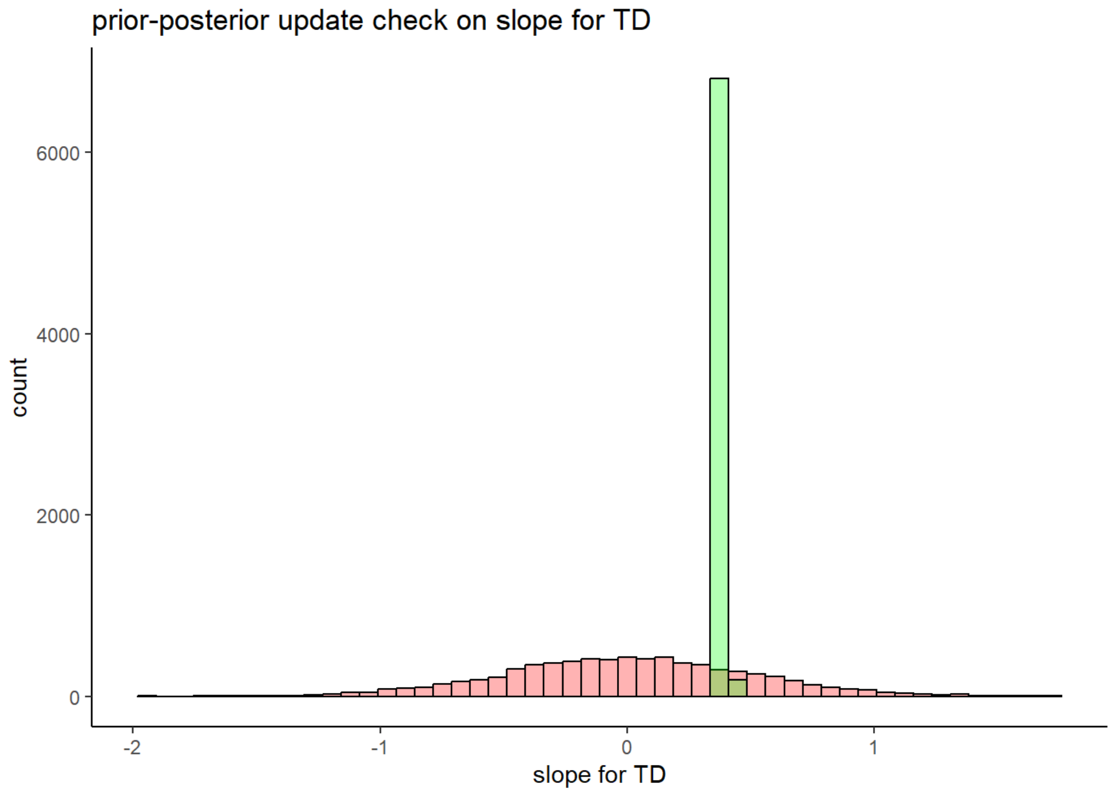
```
plot2
```


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

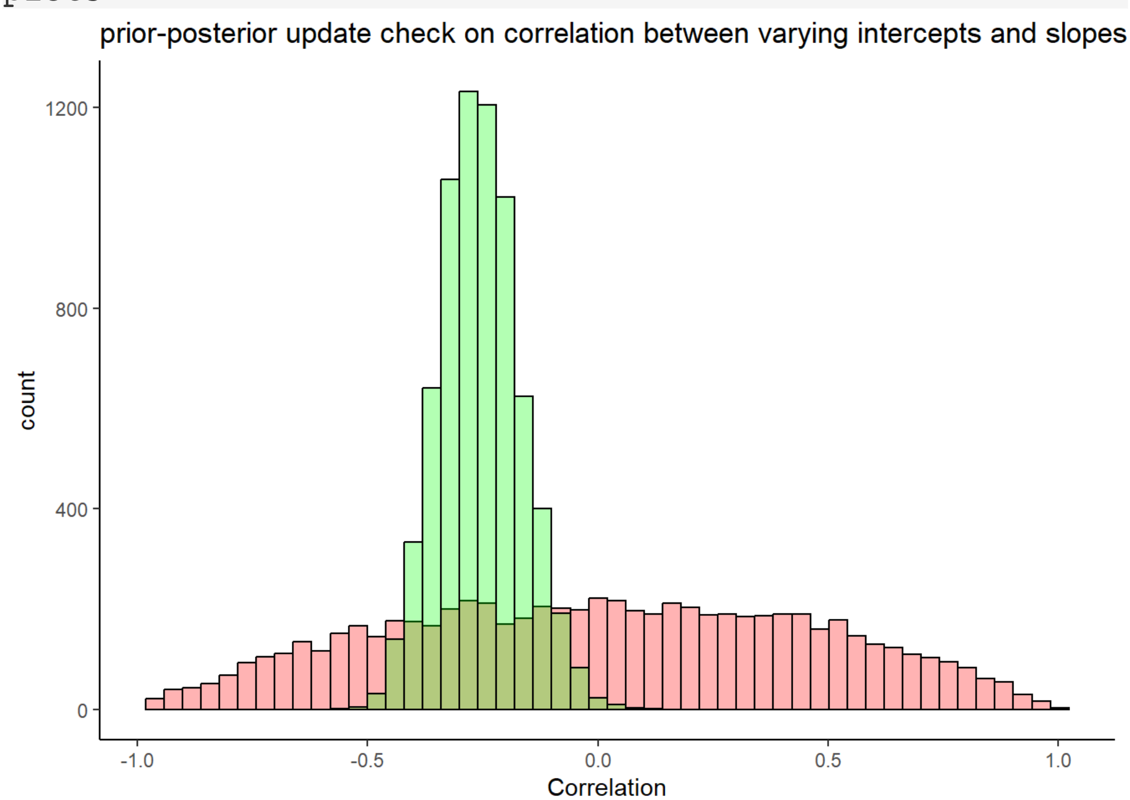
```
plot3
```


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

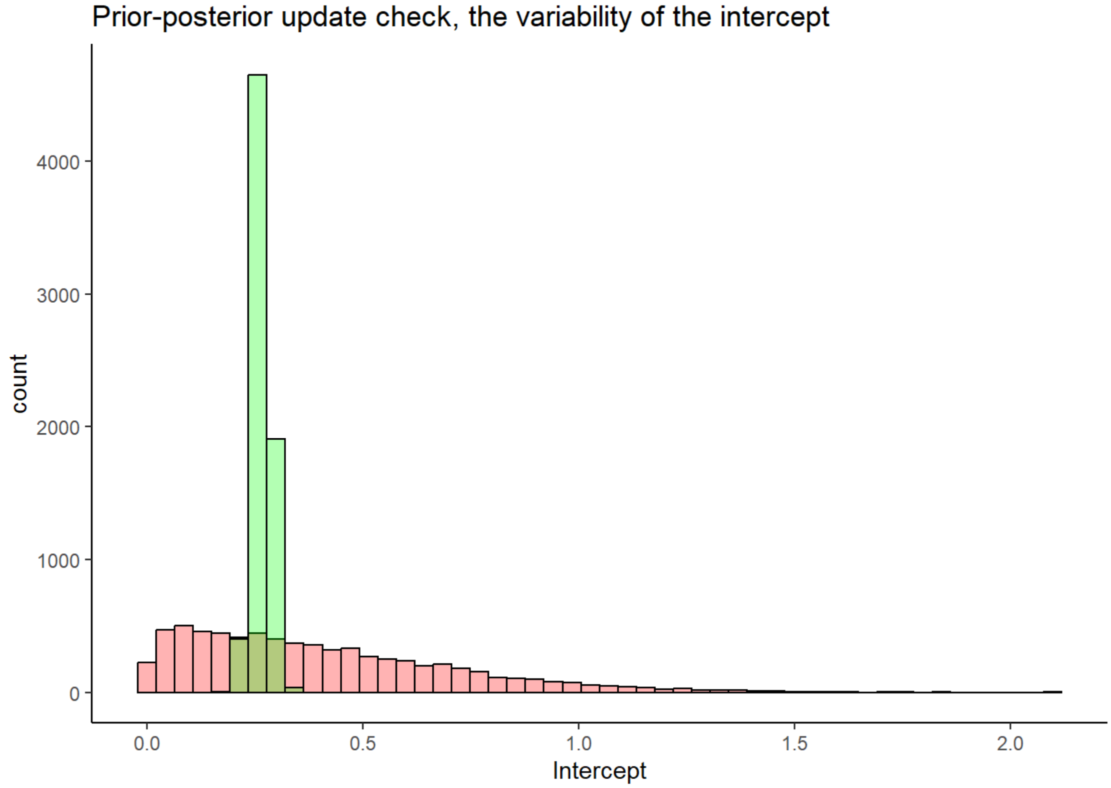
```
plot4
```


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

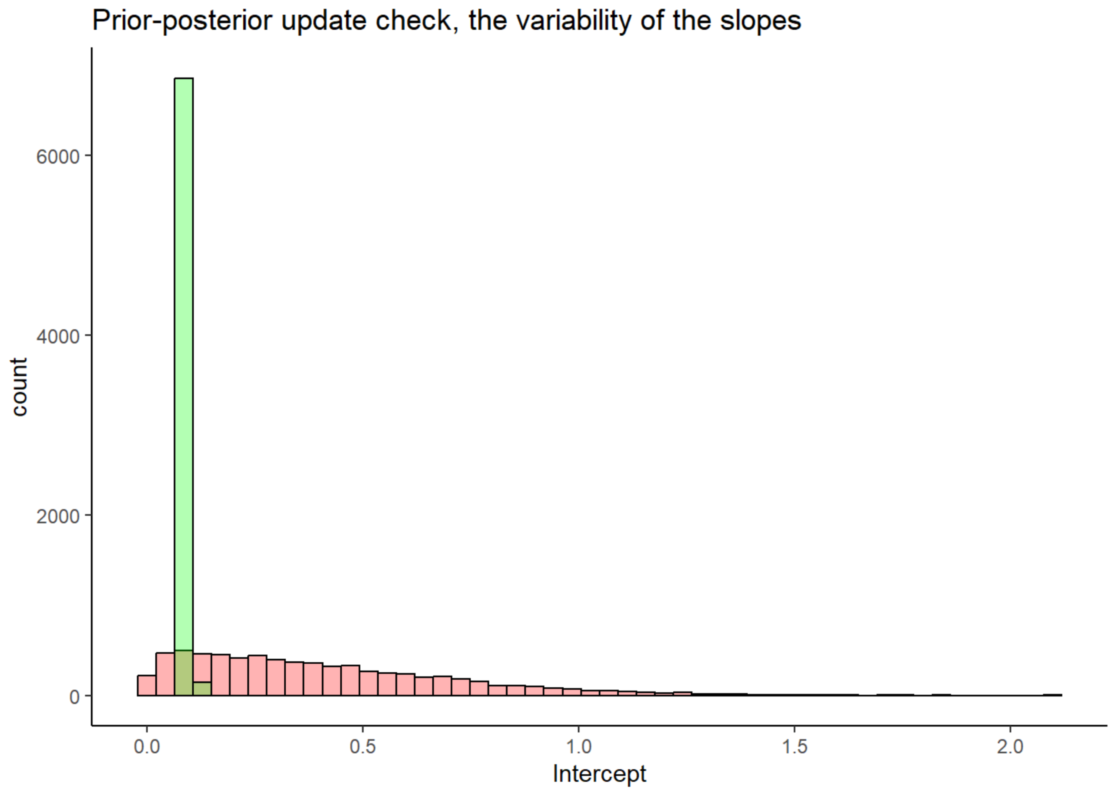
```
plot5
```


prior-posterior update check on correlation between varying intercepts and slopes
count
1200
800
400
0
-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)
```

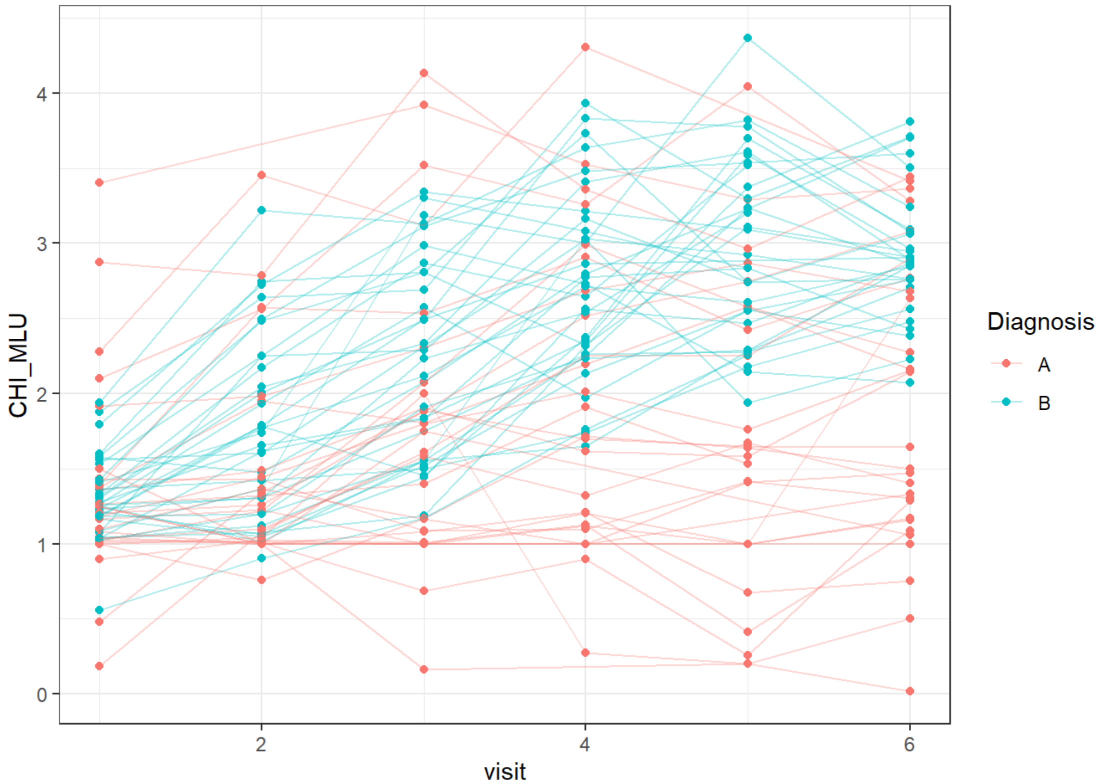
```
##  [1]  1 10 11 12 13 14 15 16 17 18 19  2 20 21 22 23
24 25 26 27 28 29  3 30 31
## [26] 32 33 34 35 36 37 38 39  4 40 41 42 43 44 45 46
47 48 49  5 50 51 52 53 54
## [51] 55 56 57 58 59  6 60 61  7  8  9
```

```
real_data <- real_data %>%
  mutate(Diagnosis=as.factor(Diagnosis))

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

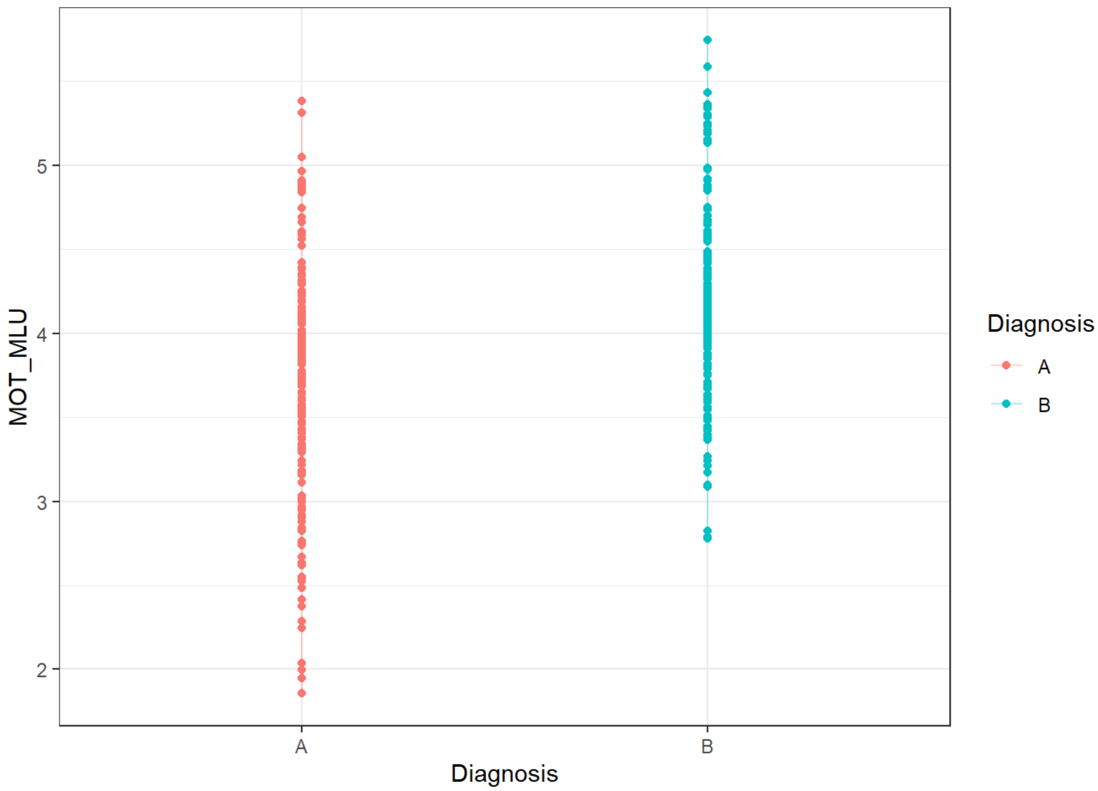
```
#Counting participants
real_data %>%
  group_by(Diagnosis) %>%
  filter(visit==1) %>%
  count()
```

```
##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)
```

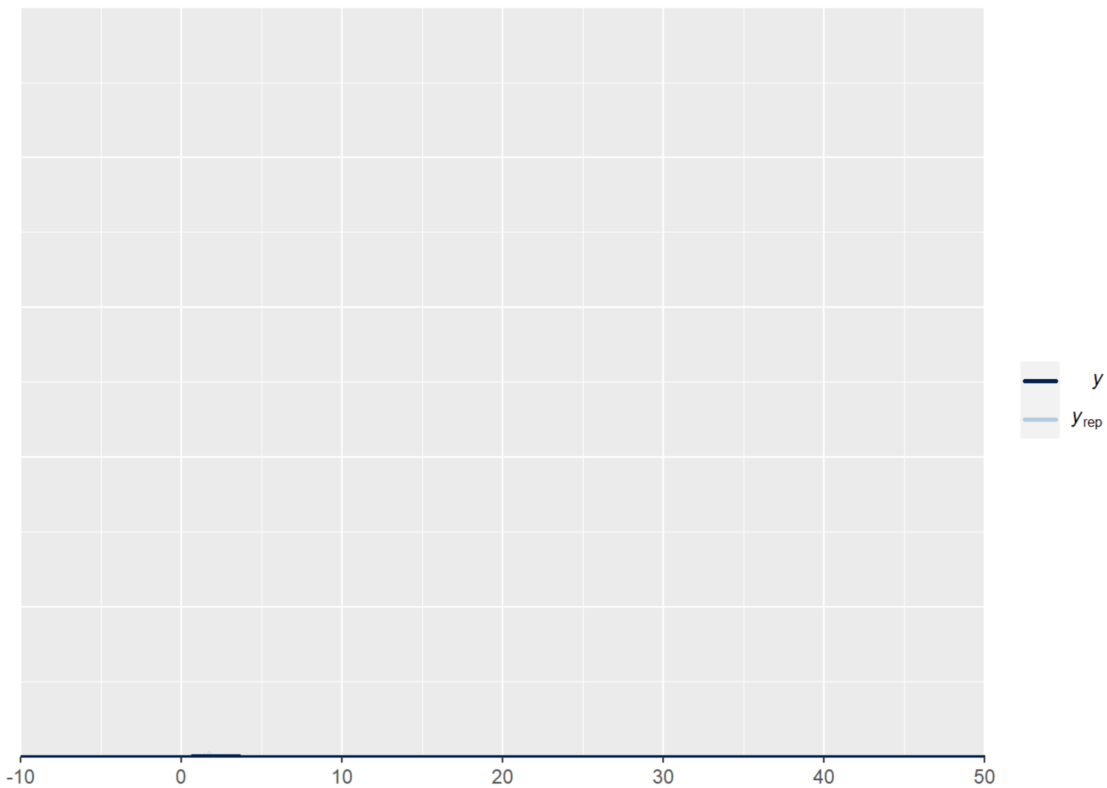
```
#Simulating priors
priors_sim<-c(
prior(normal(0,0.2),class=b),
prior(normal(0.5,0.05),class=b,coef="DiagnosisA"),
prior(normal(0.5,0.02),class=b,coef="DiagnosisB"),
prior(normal(0,0.06),class=b,coef="DiagnosisA:visit"),
prior(normal(0,0.03),class=b,coef="DiagnosisB:visit"),
prior(normal(0,0.2),class=sd,coef=Intercept,group=id),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



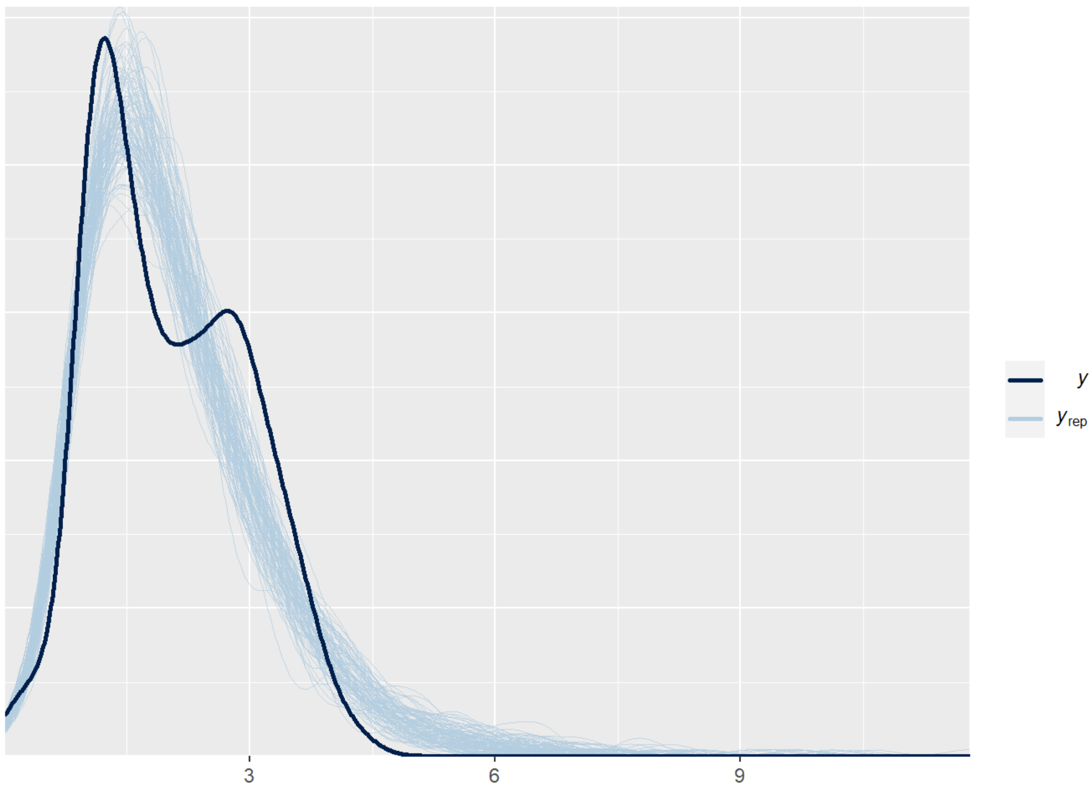
###fit the model

```
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
```

```
)
)
```

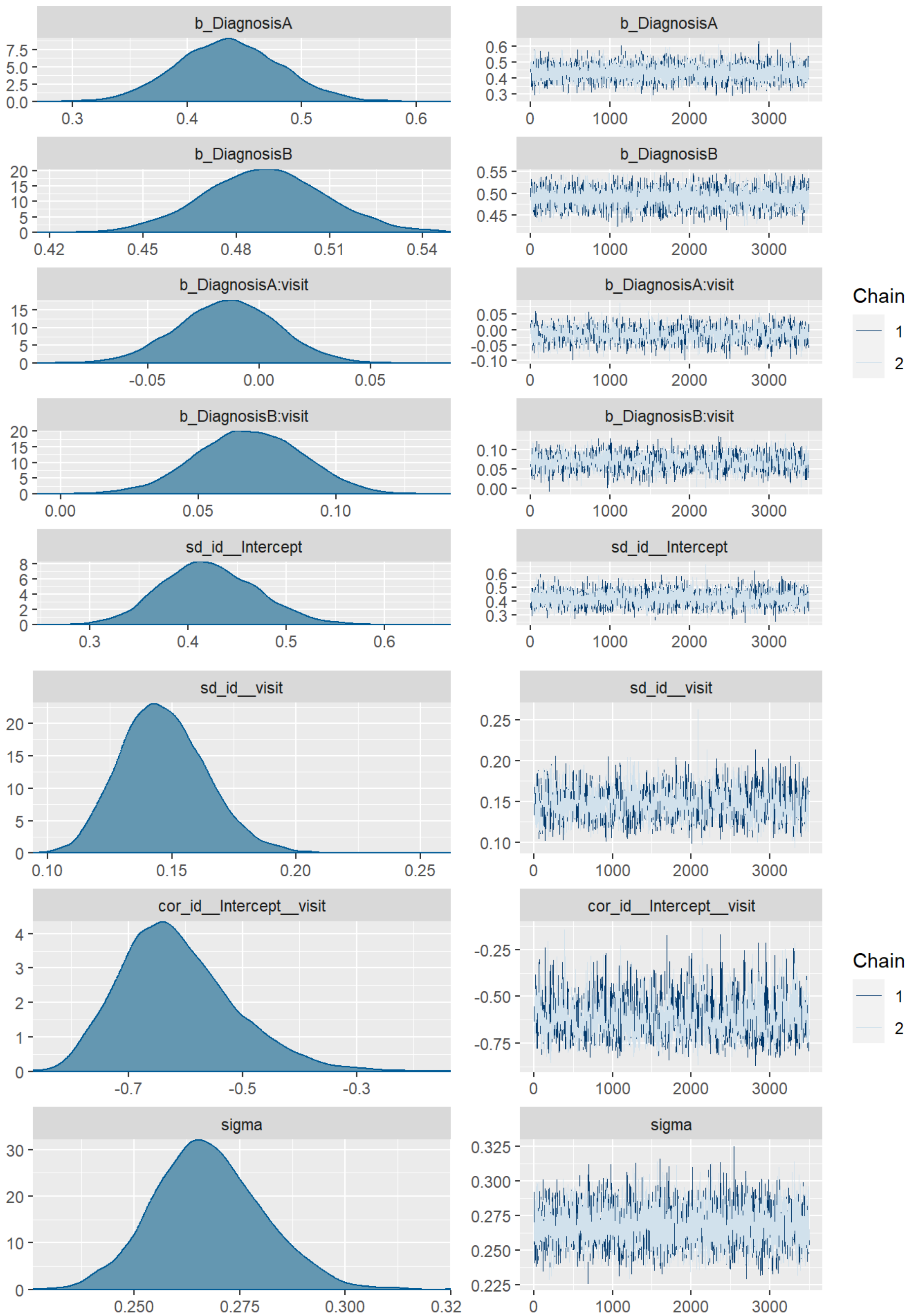
Sara ###posterior predictive check

```
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
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
```

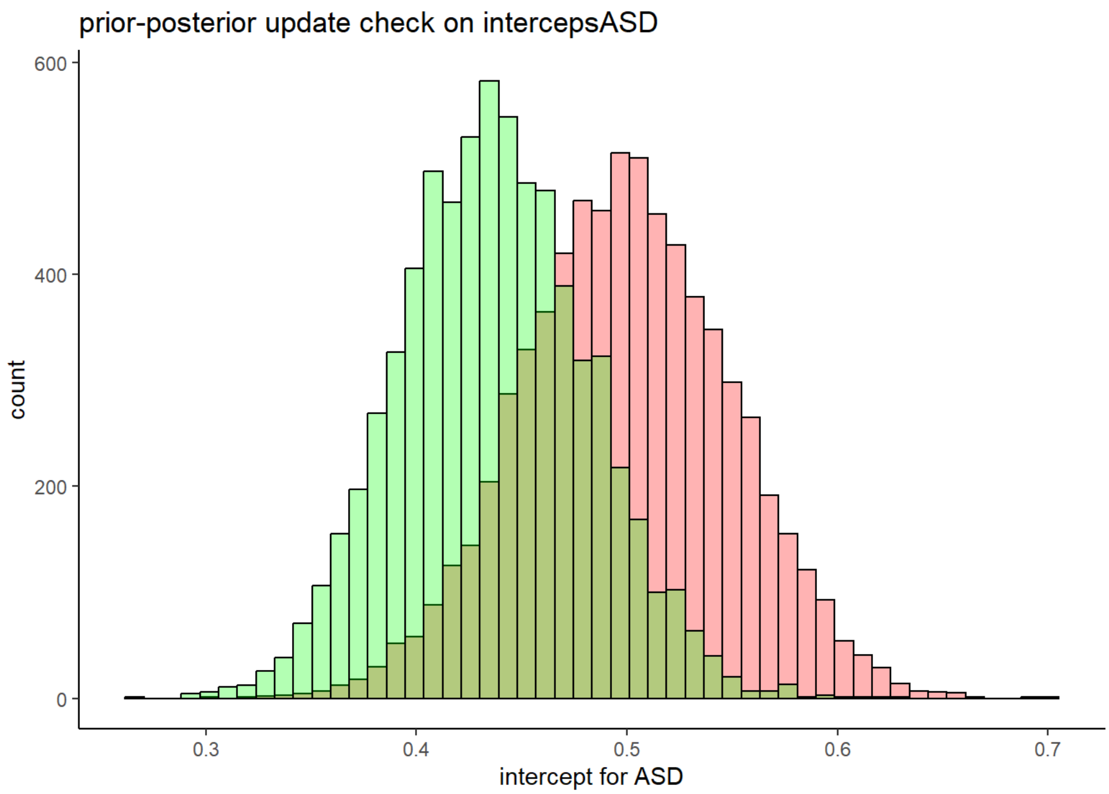
```
0.03 1.00     1301     2684
## DiagnosisB:visit     0.07      0.02     0.03
0.11 1.00     1158     2346
##
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat
Bulk_ESS Tail_ESS
## sigma     0.27      0.01     0.24     0.29 1.00
4392     4575
##
## Draws were sampled 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)+
```

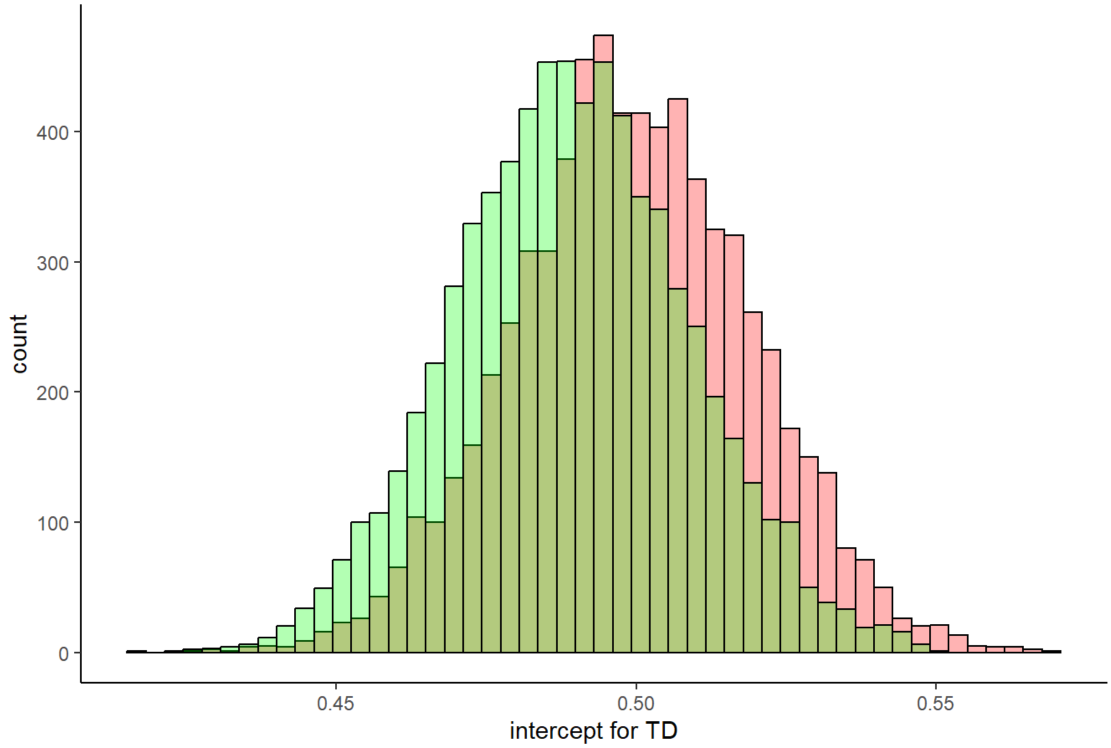
```
  geom_histogram(aes(prior_sd_id__Intercept),
fill='red', color='black', alpha=0.3, bins=50)+
  geom_histogram(aes(sd_id__Intercept), fill='green',
color='black', alpha=0.3, bins=50)+
  theme_classic()+
  ggtitle('Prior-posterior update check, the
variability of the intercept')+
  xlab("Intercept")

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
```



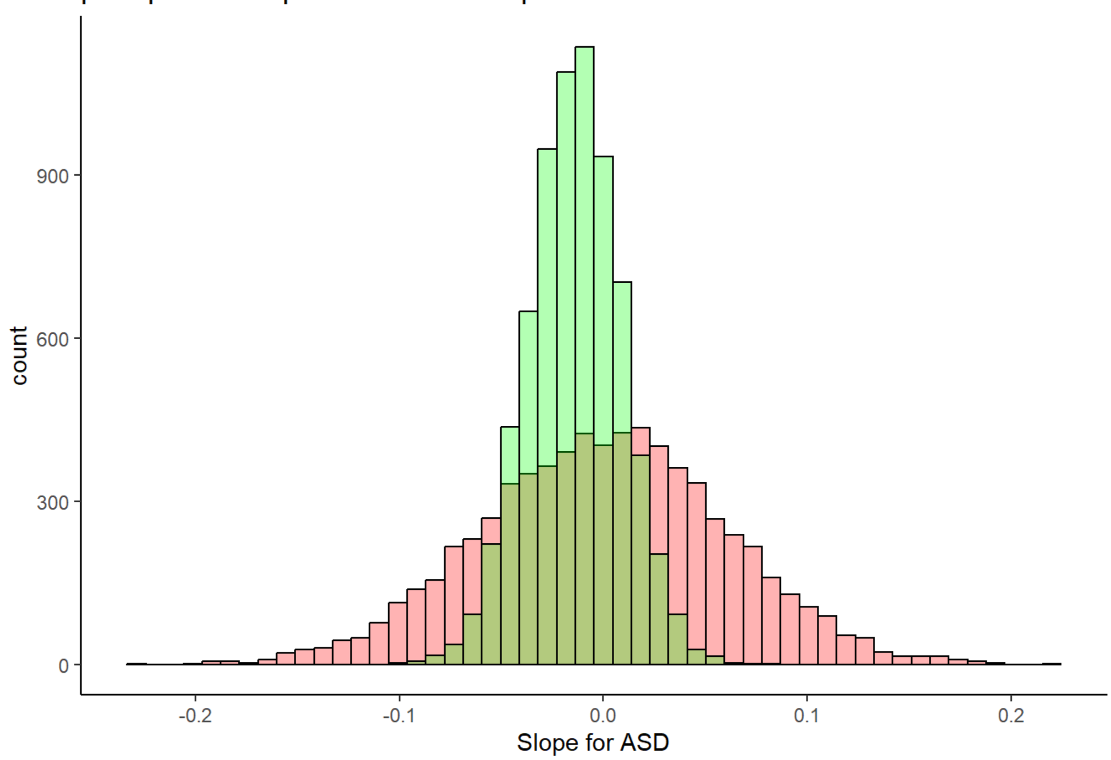

```
plot2
```

prior-posterior update check on intercept for TD

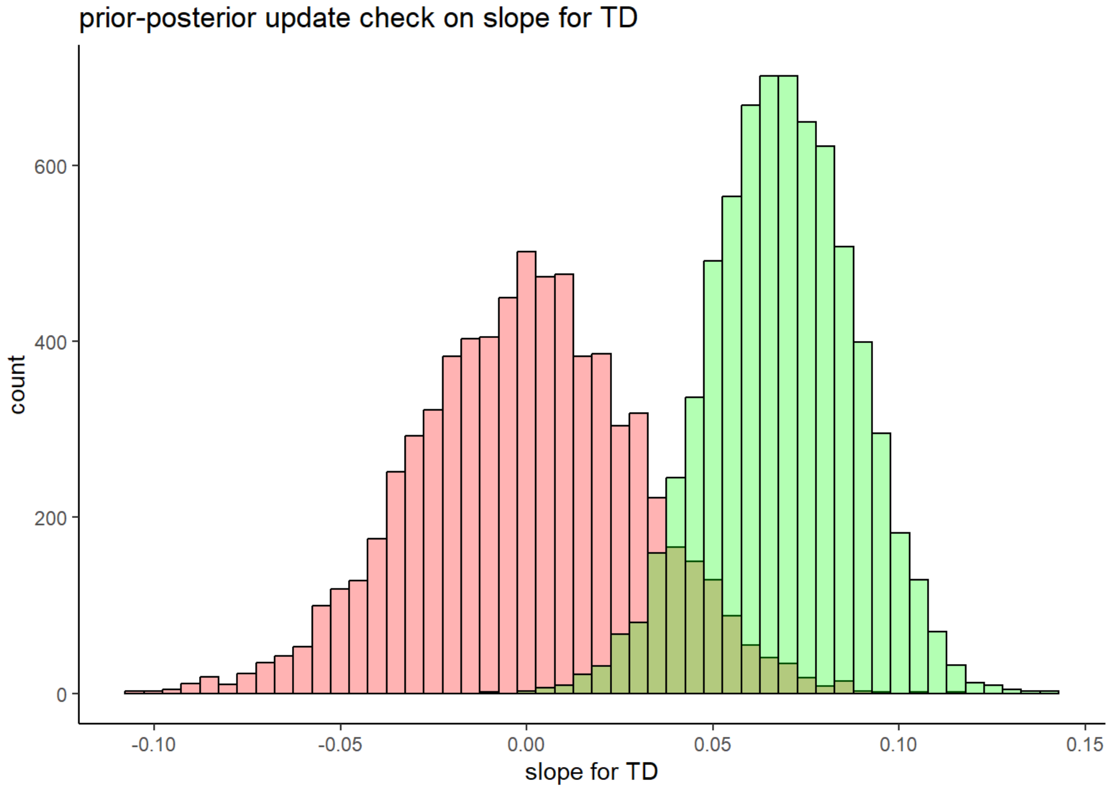



```
plot3
```

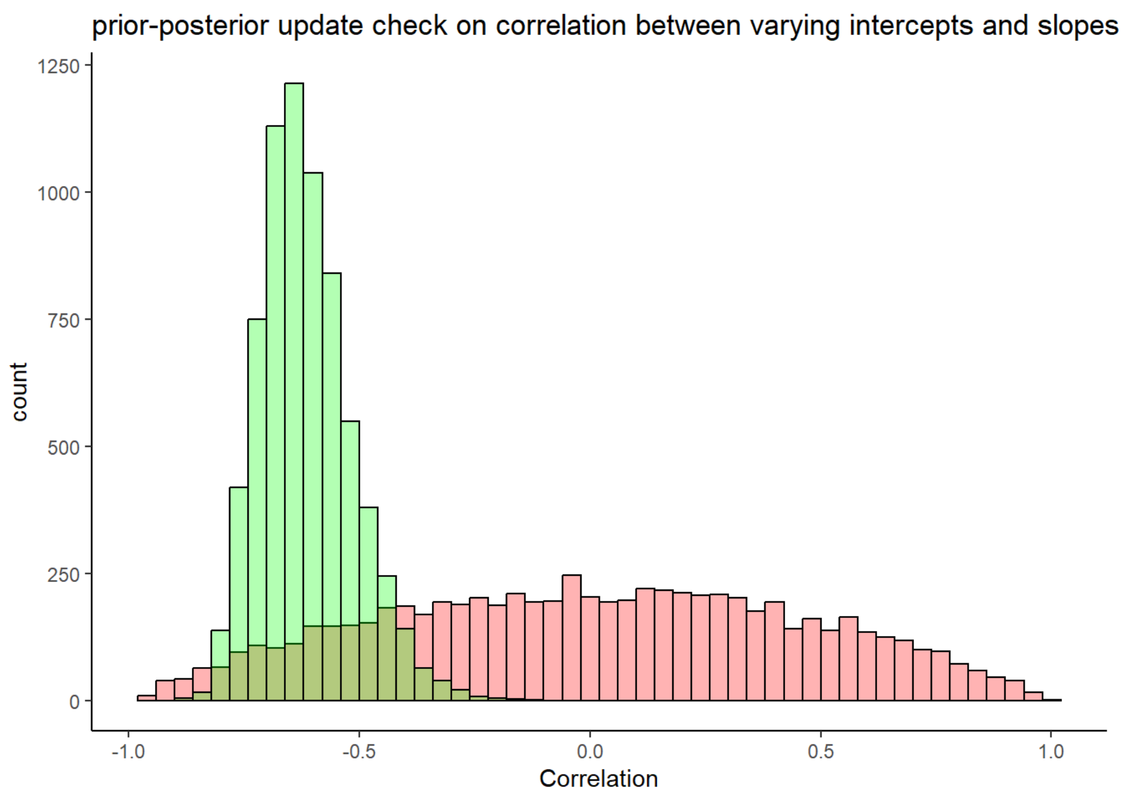
prior-posterior update check on slope for ASD



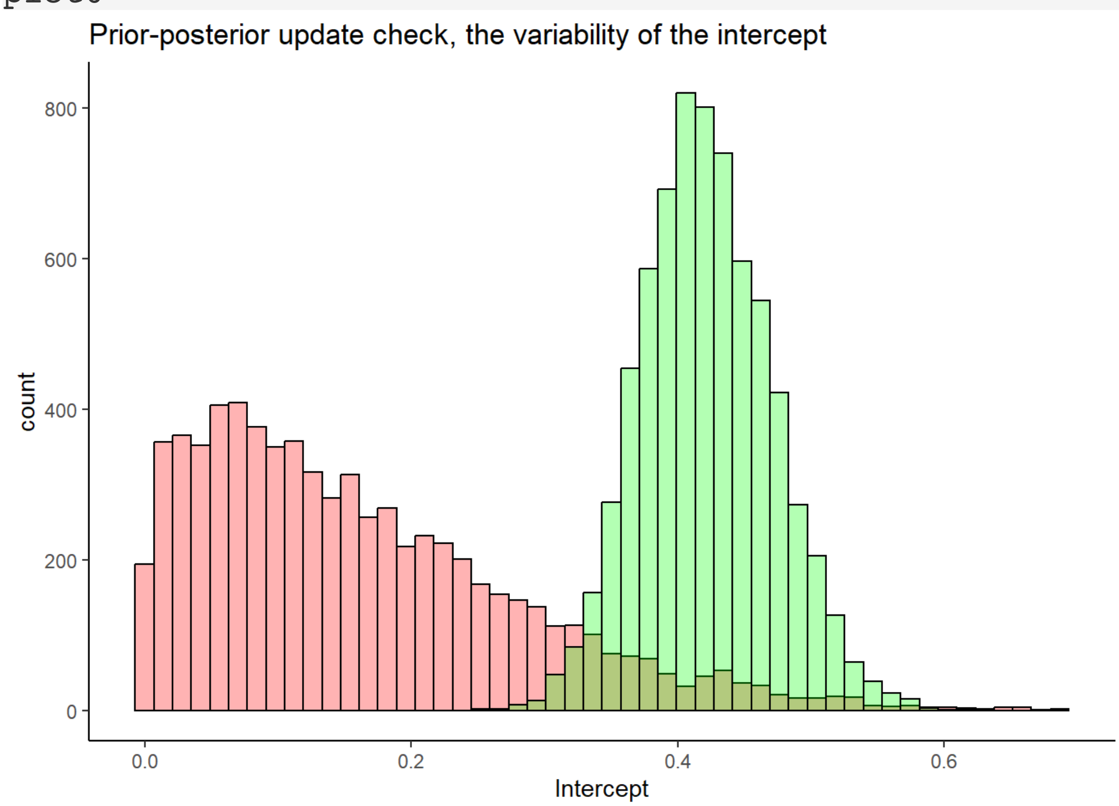
```
plot4
```


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

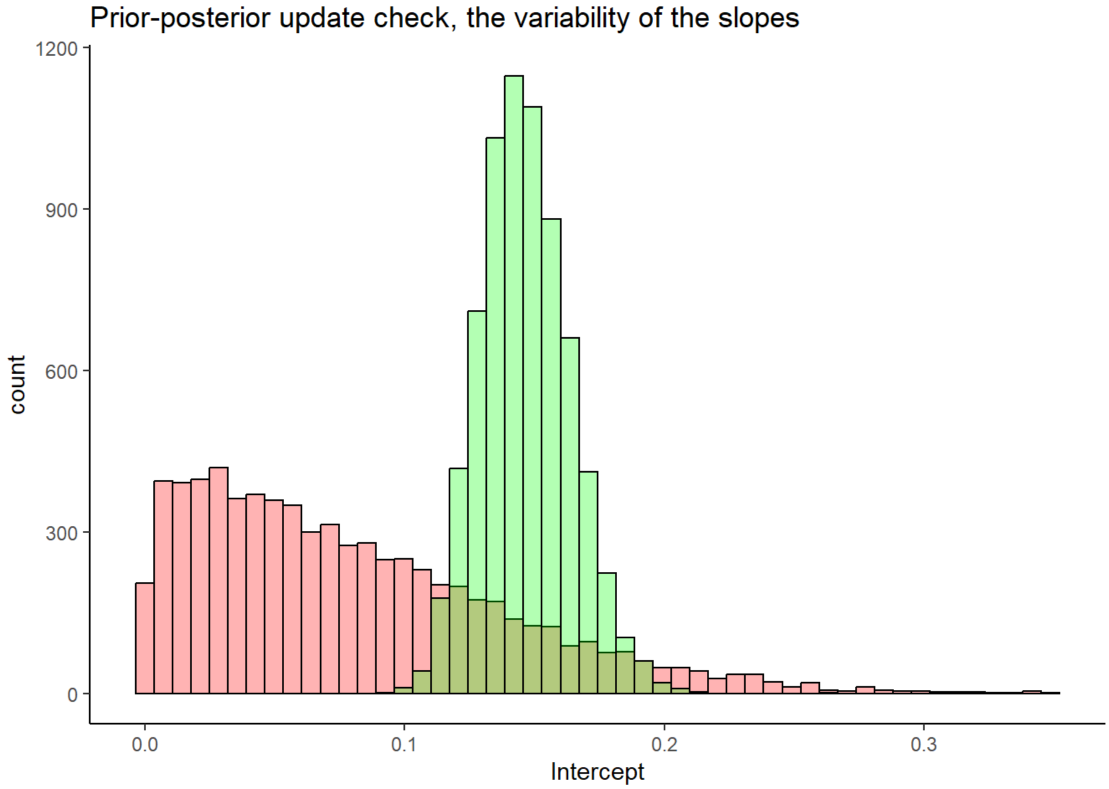
```
plot5
```


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

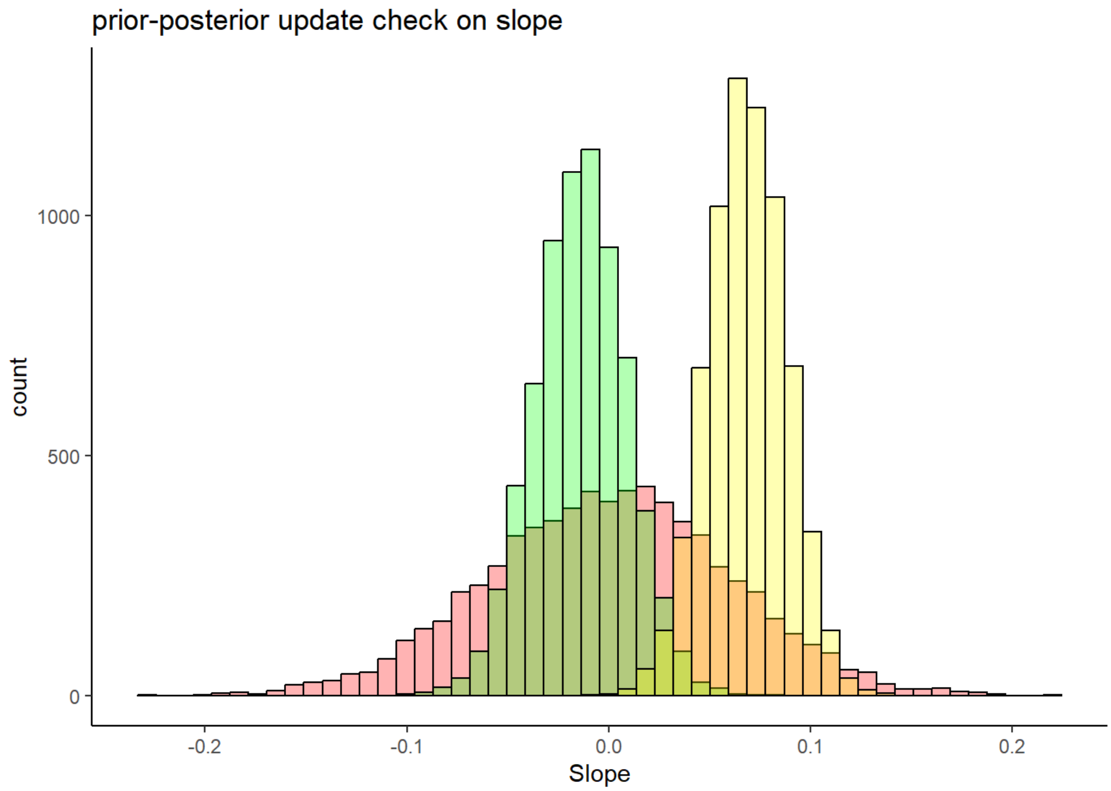
```
plot6
```


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

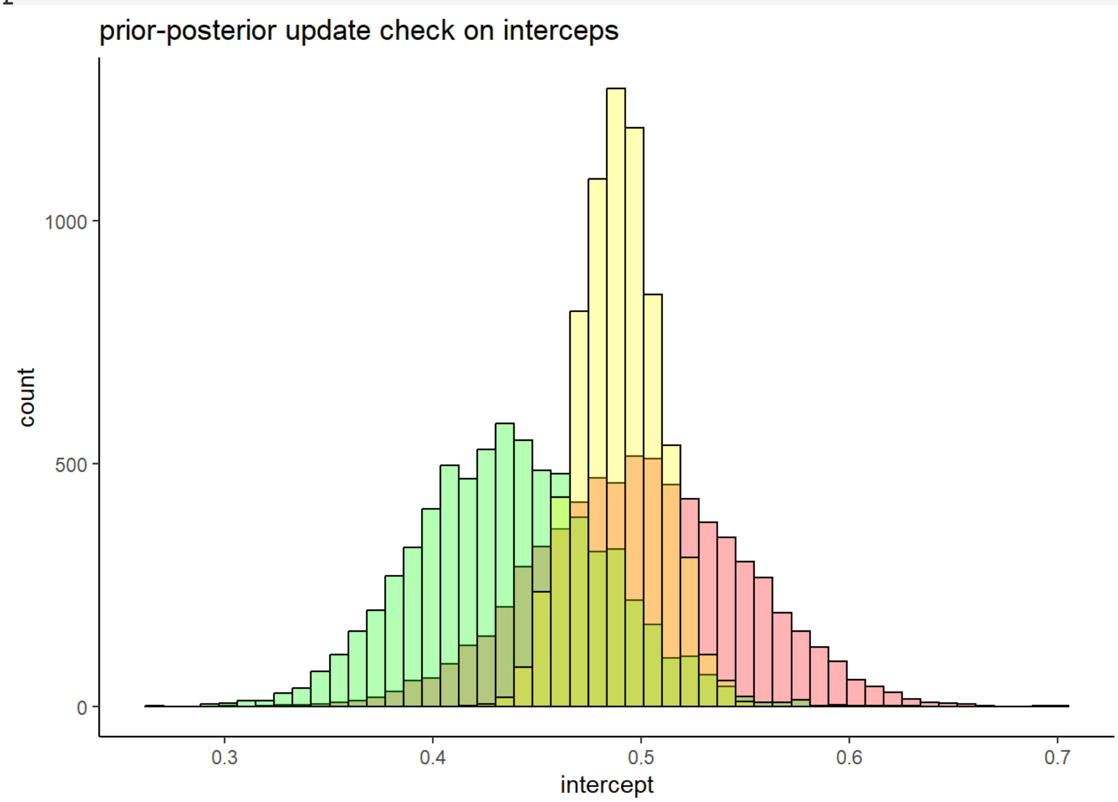
```
plot7
```


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

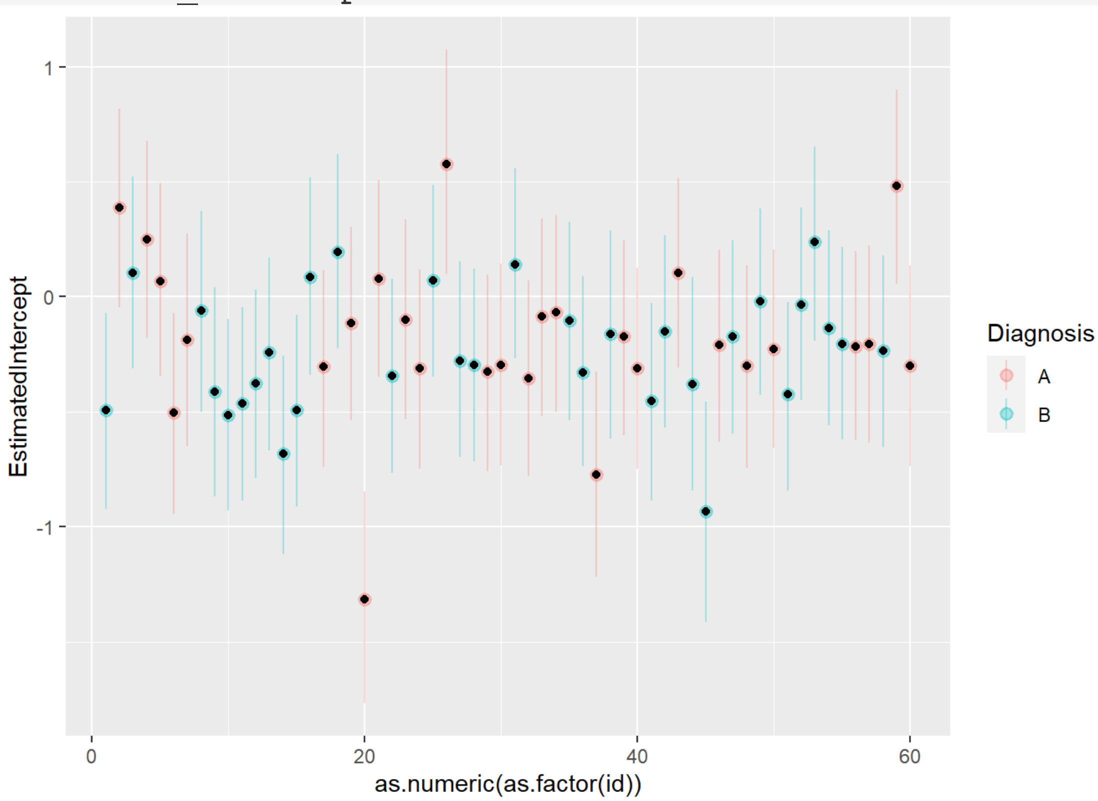
```
temp_re <- ranef(MLU_prior_fit)$id
for (i in unique(real_data$id)) {
  temp <- as.character(i)
  real_data$EstimatedIntercept[real_data$id == i] <-
temp_re[,,'Intercept'][temp,1]
  real_data$EstimatedIntercept_low[real_data$id == i]
<- temp_re[,,'Intercept'][temp,3]
  real_data$EstimatedIntercept_high[real_data$id == 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
```