Assignment 1 - Language development in autistic and neurotypical children

Study group 9

2022-08-15

```
pacman::p_load(tidyverse, brms, cmdstanr, ggpubr)

theme_set(theme_minimal())
```

Part 1

```
# Assign values
n <- 360
visits <- 6
mu_asd <- log(1.5)
sigma_asd <- log(1.5) - log(1.5-0.5)
mu_td <- log(1.5)
sigma_td <- log(1.5) - log(1.5-0.3)
mu_visit_asd <- 0.09
sigma_visit_asd <- 0.035
mu_visit_td <- 0.18
sigma_visit_td <- 0.025
x <- c("ASD", "TD")

# Create data frame
df <- tibble(ID = rep(1:60, each = 6)) %>%
  mutate(Visit = rep(1:6, times = 60)) %>%
  mutate(Diagnosis = rep(x, each = 180))



# loop intercepts and slopes
for (i in seq(n)) {
  df$IndividualIntercept[df$ID == i & df$Diagnosis == "ASD"] <- rnorm(1, mu_asd, sigma_asd)
  df$IndividualIntercept[df$ID == i & df$Diagnosis == "TD"] <- rnorm(1, mu_td, sigma_td)
  df$IndividualSlope[df$ID == i & df$Diagnosis == "ASD"] <- rnorm(1, mu_visit_asd, sigma_visit_asd)
  df$IndividualSlope[df$ID == i & df$Diagnosis == "TD"] <- rnorm(1, mu_visit_td, sigma_visit_td)
}
```

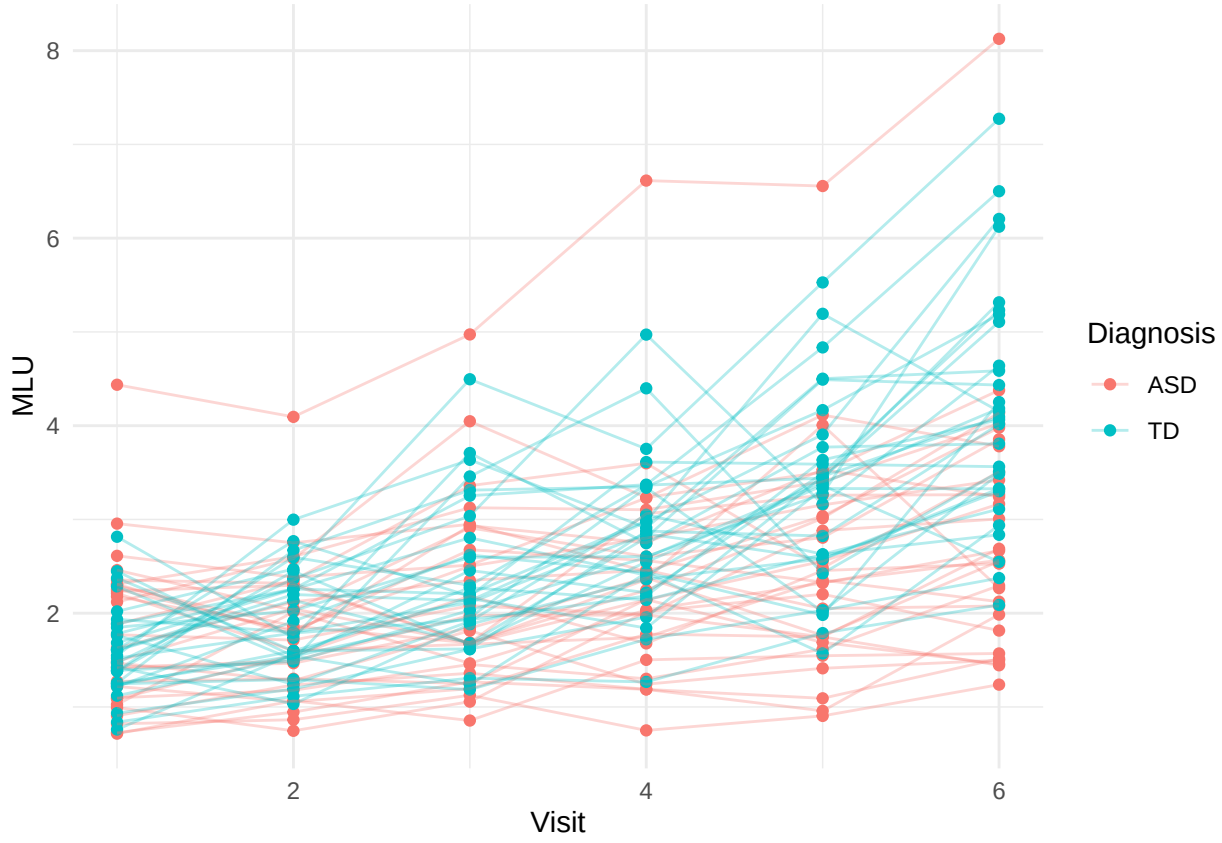
```
## Warning: Unknown or uninitialised column: `IndividualIntercept`.
```

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

```
# Simulate MLU
for (i in seq(n)) {
  df$MLU[i] <- exp(rnorm(1, df$IndividualIntercept[i] + df$IndividualSlope[i] * (df$Visit[i] - 1), sd =
}
```

```
## Warning: Unknown or uninitialised column: 'MLU'.
```

```
# Plot
ggplot(df, aes(Visit, MLU, color = Diagnosis, group = ID)) +
  geom_point() +
  geom_line(alpha = 0.3)
```




```
# Check MLU
max(df$MLU)
```

```
## [1] 8.126249
```

```
min(df$MLU)
```

```
## [1] 0.7153566
```

```
mean(df$MLU)
```

```
## [1] 2.467813
```

```
# Define formulas
m0 <- bf(MLU ~ 1)
m1 <- bf(MLU ~ 0 + Diagnosis)
m2 <- bf(MLU ~ 0 + Diagnosis + Diagnosis:Visit)
m3 <- bf(MLU ~ 0 + Diagnosis + Diagnosis:Visit + (1 + Visit|ID))
m4 <- bf(MLU ~ 0 + Diagnosis + Diagnosis:Visit + (1 + Visit|gr(ID, by = Diagnosis)))

# Define priors
p1 <- c(
  brms::prior(normal(0, 0.1), class = b),
  brms::prior(normal(0.4, 0.1), class = b, coef = "DiagnosisASD"),
  brms::prior(normal(0.4, 0.1), class = b, coef = "DiagnosisTD"),
  brms::prior(normal(0, 0.25), class = sd, coef = Intercept, group = ID),
  brms::prior(normal(0, 0.25), class = sd, coef = Visit, group = ID)
)


get_prior(m1, data = df)
```

```
##                 prior class         coef group resp dpar nlpar lb ub
##                (flat)     b
##                (flat)     b DiagnosisASD
##                (flat)     b  DiagnosisTD
##  student_t(3, 0, 2.5) sigma                                       0
##        source
##       default
##  (vectorized)
##  (vectorized)
##       default
```

```
# Fitting models
prior_fit <- brm(
  m3, data = df, prior = p1, sample_prior = "only", backend = "cmdstanr", chains = 2, cores = 2, contro
)
```

```
## Start sampling
```

```
## Running MCMC with 2 parallel chains...
##
## Chain 1 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1 Iteration:  100 / 2000 [  5%]  (Warmup)
## Chain 1 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1 Iteration:  300 / 2000 [ 15%]  (Warmup)
## Chain 1 Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2 Iteration:  100 / 2000 [  5%]  (Warmup)
## Chain 2 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2 Iteration:  300 / 2000 [ 15%]  (Warmup)
```

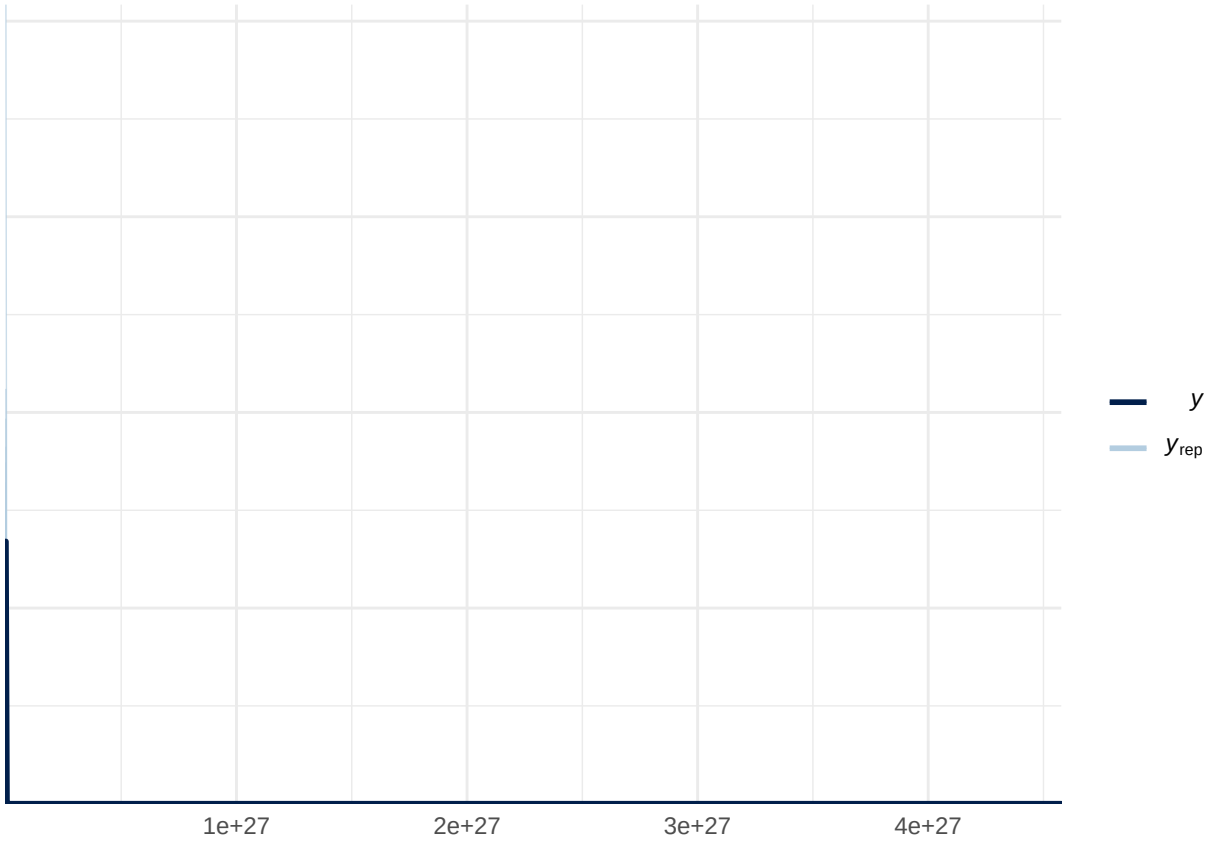
```
## Chain 1 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 1 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2 Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 1 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 1 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 2 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1 Iteration:  900 / 2000 [ 45%]  (Warmup)
## Chain 1 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2 Iteration:  900 / 2000 [ 45%]  (Warmup)
## Chain 2 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 1 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 1 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 2 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 2 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 1 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 1 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 2 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 2 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 2 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 1 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1 finished in 1.5 seconds.
## Chain 2 finished in 1.5 seconds.
##
## Both chains finished successfully.
## Mean chain execution time: 1.5 seconds.
## Total execution time: 1.7 seconds.
```

```
model_fit <- brm(
  m3, data = df, prior = p1, sample_prior = T, backend = "cmdstanr", chains = 2, cores = 2, control = li
)
```

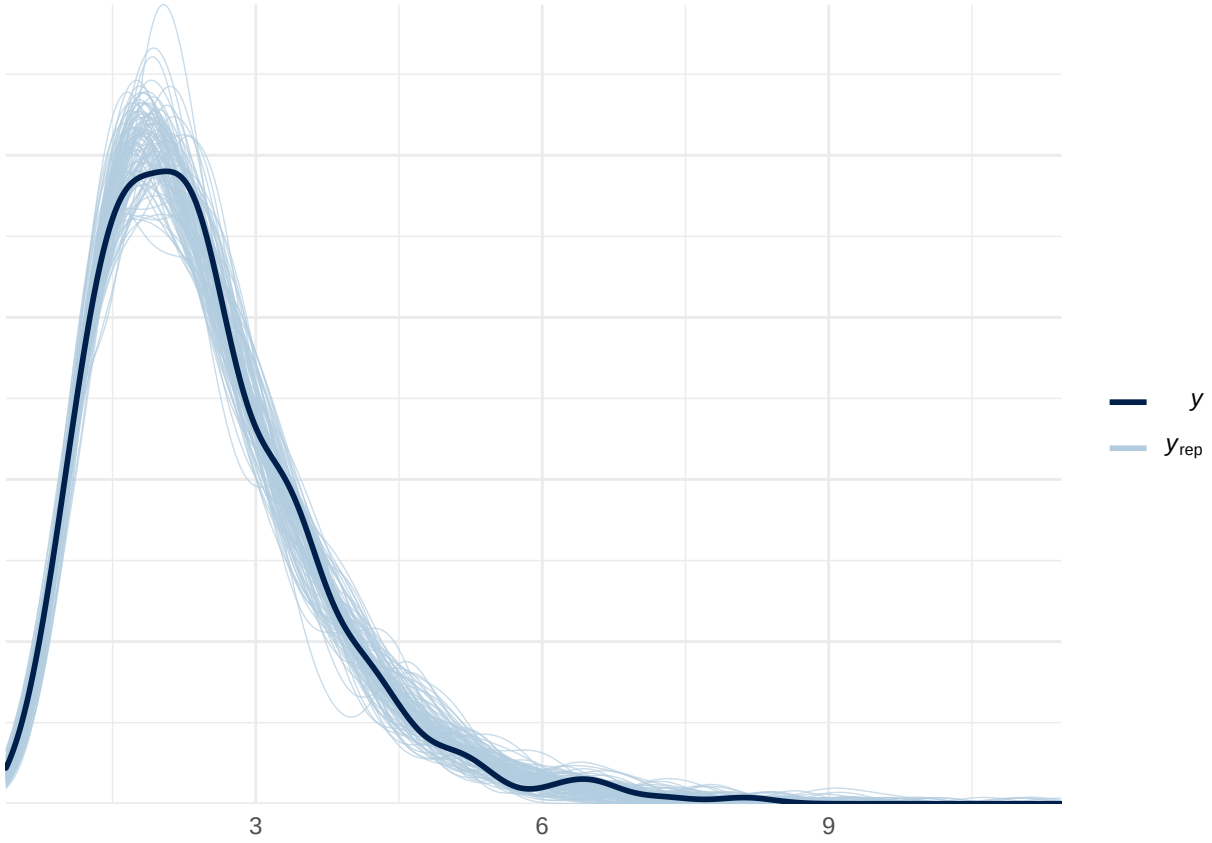
```
## Start sampling

## Running MCMC with 2 parallel chains...
##
## Chain 1 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2 Iteration:  100 / 2000 [  5%]  (Warmup)
```

```
## Chain 1 Iteration:  100 / 2000 [  5%]  (Warmup)
## Chain 2 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2 Iteration:  300 / 2000 [ 15%]  (Warmup)
## Chain 1 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2 Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1 Iteration:  300 / 2000 [ 15%]  (Warmup)
## Chain 1 Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 1 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 2 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 1 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 2 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2 Iteration:  900 / 2000 [ 45%]  (Warmup)
## Chain 1 Iteration:  900 / 2000 [ 45%]  (Warmup)
## Chain 2 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 1 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 1 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 2 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 1 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 2 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 2 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 2 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 1 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2 finished in 19.9 seconds.
## Chain 1 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 1 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 1 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1 finished in 22.5 seconds.
##
## Both chains finished successfully.
## Mean chain execution time: 21.2 seconds.
## Total execution time: 22.6 seconds.
```

```
# Checking priors
pp_check(prior_fit, ndraws = 100)
```



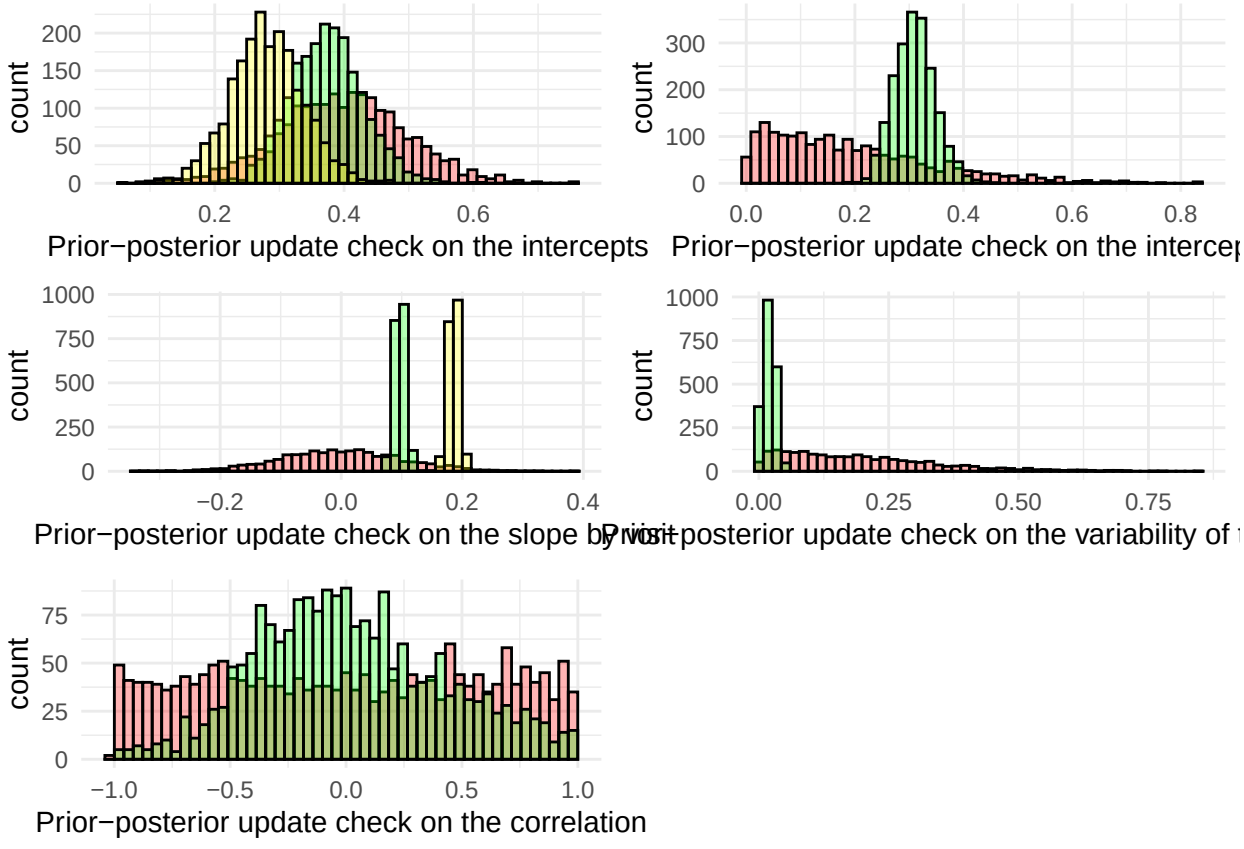

```
# Cheking fit
pp_check(model_fit, ndraws = 100)
```




```
posterior <- as_draws_df(model_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) +
  geom_histogram(aes(b_DiagnosisTD), fill = "yellow", color = "black", alpha = 0.3, bins = 50) +
  xlab("Prior-posterior update check on the intercepts")



plot2 <- 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) +
  xlab("Prior-posterior update check on the intercepts")



plot3 <- ggplot(posterior) +
  geom_histogram(aes(`prior_b_DiagnosisASD:Visit`), fill = "red", color = "black", alpha = 0.3, bins = 5



plot4 <- 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) +
```

```
  xlab("Prior-posterior update check on the variability of the slope")



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)
  xlab("Prior-posterior update check on the correlation")


# Wrapping them plots
ggarrange(plot1, plot2, plot3, plot4, plot5, rremove("x.text"), ncol = 2, nrow = 3)
```

```
## Warning in as_grob.default(plot): Cannot convert object of class themegg into a
## grob.
```




```
# Model output
summary(model_fit)
```

```
##  Family: lognormal
##   Links: mu = identity; sigma = identity
## Formula: MLU ~ 0 + Diagnosis + Diagnosis:Visit + (1 + Visit | ID)
##    Data: df (Number of observations: 360)
##   Draws: 2 chains, each with iter = 2000; warmup = 1000; thin = 1;
```

```
##          total post-warmup draws = 2000
##
## Group-Level Effects:
## ~ID (Number of levels: 60)
##                       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS
## sd(Intercept)             0.31      0.04     0.24     0.39 1.00      523
## sd(Visit)                 0.02      0.01     0.00     0.04 1.01      175
## cor(Intercept,Visit)      0.02      0.40    -0.70     0.85 1.00      773
##                       Tail_ESS
## sd(Intercept)             1186
## sd(Visit)                  456
## cor(Intercept,Visit)       780
##
## Population-Level Effects:
##                     Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## DiagnosisASD            0.37      0.06     0.27     0.49 1.01      542      944
## DiagnosisTD             0.28      0.06     0.17     0.39 1.00      657      965
## DiagnosisASD:Visit      0.10      0.01     0.08     0.12 1.00     2724     1456
## DiagnosisTD:Visit       0.19      0.01     0.17     0.20 1.00     2676     1359
##
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma     0.19      0.01     0.18     0.21 1.00      686     1373
##
## 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).
```

```
# n_sim <- 100
#
# # this will help us track time
# t1 <- Sys.time()
#
# # here's the main event!
# s <-
#   tibble(seed = 1:n_sim) %>%
#   mutate(d = map(seed, generateFunc, n = 60)) %>%
#   mutate(model_fit = map2(d, seed, ~update(fit, newdata = .x, seed = .y)))
#
# t2 <- Sys.time()
#
# head(s)
#
#
# parameters <-
#   s %>%
#   mutate(treatment = map(model_fit, ~ fixef(.) %>%
#                            data.frame() %>%
#                            rownames_to_column("parameter"))) %>%
#   unnest(treatment)
#
# head(parameters)
#
#
```

```
#
# parameters %>%
#   filter(parameter == "DiagnosisASD") %>%
#
#   ggplot(aes(x = seed, y = Estimate, ymin = Q2.5, ymax = Q97.5)) +
#   geom_hline(yintercept = c(0, .5), color = "white") +
#   geom_pointrange(fatten = 1/2) +
#   labs(x = "seed (i.e., simulation index)",
#        y = expression(beta[1]))
#
# parameters %>%
#   ggplot(aes(x = reorder(seed, Q2.5), y = Estimate, ymin = Q2.5, ymax = Q97.5)) +
#   geom_hline(yintercept = c(0, .5), color = "white") +
#   geom_pointrange(fatten = 1/2) +
#   scale_x_discrete("reordered by the lower level of the 95% intervals", breaks = NULL) +
#   ylab(expression(beta[1])) +
#   coord_cartesian(ylim = c(-.5, 1.3))
```

```
# parameters %>%
#   filter(parameter == "DiagnosisASD") %>%
#   mutate(check = ifelse(Q2.5 > 0, 1, 0)) %>%
#   summarise(power = mean(check))
#
#
# s %>%
#   mutate(rhat = map(model_fit, rhat)) %>%
#   unnest(rhat) %>%
#
#   ggplot(aes(x = rhat)) +
#   geom_histogram(bins = 20)
#
# parameters <- parameters %>%
#   mutate(width = Q97.5 - Q2.5)
#
# parameters %>%
#   ggplot(aes(x = width)) +
#   geom_histogram(binwidth = .01)
```

```
# t1 <- Sys.time()
#
# # here's the main event!
# s2 <-
#   tibble(seed = 1:n_sim) %>%
#   mutate(d = map(seed, generateFunc, n = 5)) %>%
#   mutate(model_fit = map2(d, seed, ~update(fit, newdata = .x, seed = .y)))
#
# t2 <- Sys.time()
#
# parameters2 <-
#   s2 %>%
#   mutate(treatment = map(model_fit, ~ fixef(.) %>%
#                            data.frame() %>%
#                            rownames_to_column("parameter"))) %>%
```

```
#   unnest(treatment)
#
# parameters2 %>%
#   ggplot(aes(x = reorder(seed, Q2.5), y = Estimate, ymin = Q2.5, ymax = Q97.5)) +
#   geom_hline(yintercept = c(0, .5), color = "white") +
#   geom_pointrange(fatten = 1/2) +
#   scale_x_discrete("reordered by the lower level of the 95% intervals", breaks = NULL) +
#   ylab(expression(beta[1])) +
#   coord_cartesian(ylim = c(-.5, 1.3))
#
# parameters2 %>%
#   filter(parameter == "DiagnosisASD") %>%
#   mutate(check = ifelse(Q2.5 > 0, 1, 0)) %>%
#   summarise(power = mean(check))
```

Part 2

```
# Load data
real_df <- read_csv("data_clean.csv")
```

```
## Rows: 372 Columns: 22
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (3): Ethnicity, Diagnosis, Gender
## dbl (19): Child.ID, Visit, Age, ADOS, MullenRaw, ExpressiveLangRaw, Socializ...
##
## 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.
```

```
# Run through analysis pipeline

# Filter CHI_MLU
real_df <- real_df %>%
  filter(CHI_MLU != 0) %>%
  filter(CHI_MLU != "na")

# Define formula
r_m0 <- bf(CHI_MLU ~ 0 + Diagnosis + Diagnosis:Visit + (1 + Visit|Child.ID))

# Define priors
p2 <- c(
  brms::prior(normal(0, 0.1), class = b),
  brms::prior(normal(0.4, 0.1), class = b, coef = "DiagnosisASD"),
  brms::prior(normal(0.4, 0.1), class = b, coef = "DiagnosisTD"),
  brms::prior(normal(0, 0.25), class = sd, coef = Intercept, group = Child.ID),
  brms::prior(normal(0, 0.25), class = sd, coef = Visit, group = Child.ID)
)

# Fitting priors
real_prior_fit <- brm(
```

```
  r_m0, data = real_df, prior = p2, sample_prior = "only", backend = "cmdstanr", chains = 2, cores = 2,
)
```

```
## Start sampling

## Running MCMC with 2 parallel chains...
##
## Chain 1 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2 Iteration:  100 / 2000 [  5%]  (Warmup)
## Chain 2 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2 Iteration:  300 / 2000 [ 15%]  (Warmup)
## Chain 2 Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1 Iteration:  100 / 2000 [  5%]  (Warmup)
## Chain 2 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 2 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 1 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1 Iteration:  300 / 2000 [ 15%]  (Warmup)
## Chain 1 Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2 Iteration:  900 / 2000 [ 45%]  (Warmup)
## Chain 2 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 1 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 2 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 1 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1 Iteration:  900 / 2000 [ 45%]  (Warmup)
## Chain 1 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 2 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 1 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 1 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 1 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 2 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 1 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 2 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 1 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 1 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1 finished in 1.7 seconds.
## Chain 2 finished in 1.7 seconds.
```

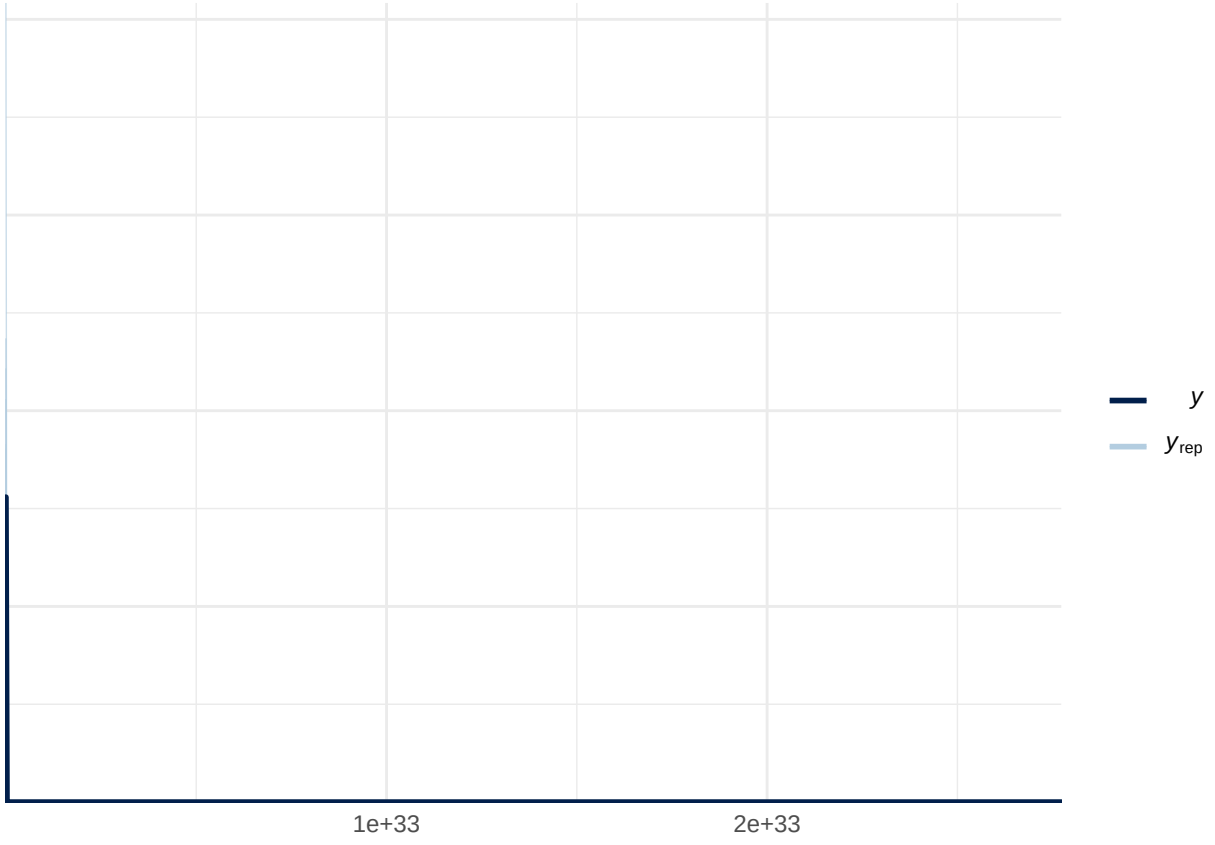
```
##
## Both chains finished successfully.
## Mean chain execution time: 1.7 seconds.
## Total execution time: 1.8 seconds.
```

```
# Fitting model
real_model_fit1 <- brm(
  r_m0, data = real_df, prior = p2, sample_prior = T, backend = "cmdstanr", chains = 2, cores = 2, conti
)
```

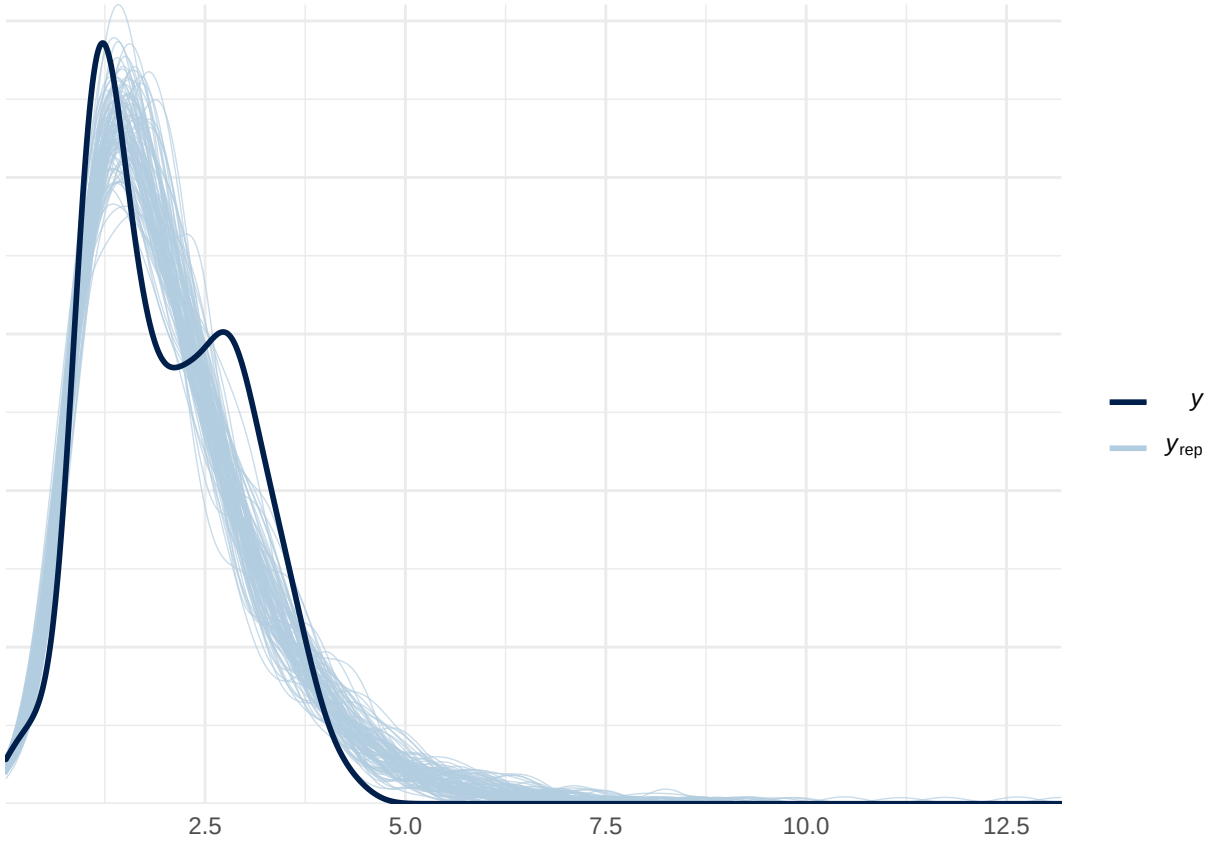
```
## Start sampling

## Running MCMC with 2 parallel chains...
##
## Chain 1 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2 Iteration:  100 / 2000 [  5%]  (Warmup)
## Chain 1 Iteration:  100 / 2000 [  5%]  (Warmup)
## Chain 2 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2 Iteration:  300 / 2000 [ 15%]  (Warmup)
## Chain 1 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2 Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1 Iteration:  300 / 2000 [ 15%]  (Warmup)
## Chain 2 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 1 Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 1 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 2 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2 Iteration:  900 / 2000 [ 45%]  (Warmup)
## Chain 1 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 2 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 1 Iteration:  900 / 2000 [ 45%]  (Warmup)
## Chain 1 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 1 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 1 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 2 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 1 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 2 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 1 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1 Iteration: 1700 / 2000 [ 85%]  (Sampling)
```

```
## Chain 2 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 1 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2 finished in 11.4 seconds.
## Chain 1 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 1 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1 finished in 12.1 seconds.
##
## Both chains finished successfully.
## Mean chain execution time: 11.7 seconds.
## Total execution time: 12.1 seconds.
```

```r
# Check priors
pp_check(real_prior_fit, ndraws = 100)
```



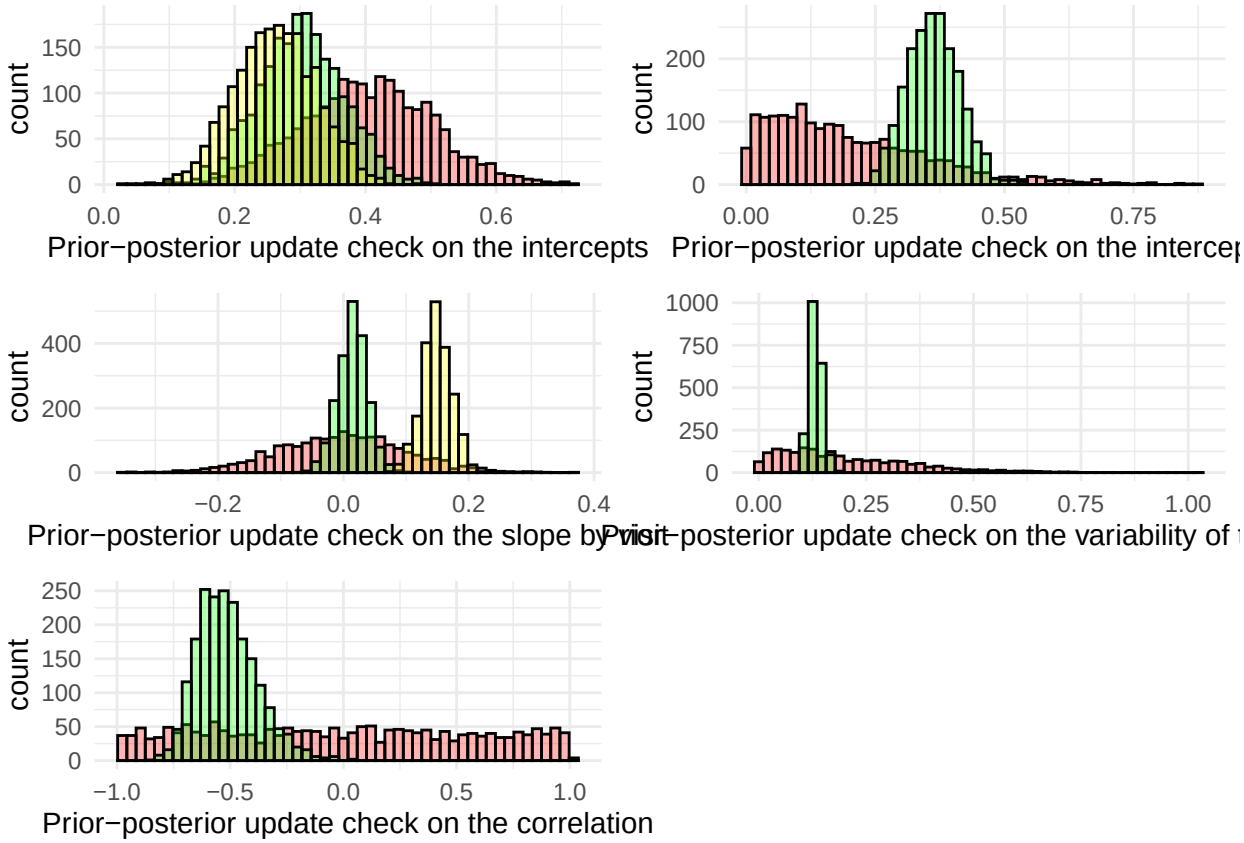

```r
# Check model
pp_check(real_model_fit1, ndraws = 100)
```




```
posterior2 <- as_draws_df(real_model_fit1)


real_plot1 <- ggplot(posterior2) + geom_histogram(aes(prior_b_DiagnosisASD), fill = "red", color = "bla




real_plot2 <- ggplot(posterior2) + geom_histogram(aes(prior_sd_Child.ID__Intercept), fill = "red", colo



real_plot3 <- ggplot(posterior2) +
  geom_histogram(aes(`prior_b_DiagnosisASD:Visit`), fill = "red", color = "black", alpha = 0.3, bins = 5
  geom_histogram(aes(`b_DiagnosisASD:Visit`), fill = "green", color = "black", alpha = 0.3, bins = 50) +
  geom_histogram(aes(`b_DiagnosisTD:Visit`), fill = "yellow", color = "black", alpha = 0.3, bins = 50) +
  xlab("Prior-posterior update check on the slope by visit")




real_plot4 <- ggplot(posterior2) +
  geom_histogram(aes(prior_sd_Child.ID__Visit), fill = "red", color = "black", alpha = 0.3, bins = 50) +
  geom_histogram(aes(sd_Child.ID__Visit), fill = "green", color = "black", alpha = 0.3, bins = 50) +
  xlab("Prior-posterior update check on the variability of the slope")
```

```
real_plot5 <- ggplot(posterior2) +
  geom_histogram(aes(prior_cor_Child.ID), fill = "red", color = "black", alpha = 0.3, bins = 50) +
  geom_histogram(aes(cor_Child.ID__Intercept__Visit), fill = "green", color = "black", alpha = 0.3, bins
  xlab("Prior-posterior update check on the correlation")


# Wrapping plots
ggarrange(real_plot1, real_plot2, real_plot3, real_plot4, real_plot5, rremove("x.text"), ncol = 2, nrow
```

```
## Warning in as_grob.default(plot): Cannot convert object of class themegg into a
## grob.
```




```
# Model output
summary(real_model_fit1)
```

```
##  Family: lognormal
##   Links: mu = identity; sigma = identity
## Formula: CHI_MLU ~ 0 + Diagnosis + Diagnosis:Visit + (1 + Visit | Child.ID)
##    Data: real_df (Number of observations: 349)
##   Draws: 2 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup draws = 2000
##
```

```
## Group-Level Effects:
## ~Child.ID (Number of levels: 61)
##                       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS
## sd(Intercept)             0.36      0.05     0.27     0.46 1.00      658
## sd(Visit)                 0.13      0.01     0.11     0.16 1.01      211
## cor(Intercept,Visit)     -0.51      0.13    -0.72    -0.22 1.01      117
##                       Tail_ESS
## sd(Intercept)             1251
## sd(Visit)                  444
## cor(Intercept,Visit)       413
##
## Population-Level Effects:
##                     Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## DiagnosisASD            0.30      0.06     0.18     0.43 1.00      856     1353
## DiagnosisTD             0.26      0.06     0.14     0.38 1.01      908     1358
## DiagnosisASD:Visit      0.01      0.02    -0.03     0.06 1.00      348      678
## DiagnosisTD:Visit       0.15      0.02     0.10     0.19 1.00      355      703
##
## Family Specific Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma     0.27      0.01     0.25     0.30 1.00      774     1269
##
## 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).
```

```
# Defining formulas
# ind_m1 <- bf(CHI_MLU ~ 0 + Diagnosis + Diagnosis:Visit + Diagnosis:verbalIQ1 + Diagnosis:nonVerbalIQ1
#
# env_m2 <- bf(CHI_MLU ~ 0 + Diagnosis + Diagnosis:Visit + Diagnosis:MOT_MLU + Diagnosis:types_MOT + Di
#
# ind_env_m3 <- bf(CHI_MLU ~ 0 + Diagnosis + Diagnosis:Visit + Diagnosis:verbalIQ1 + Diagnosis:nonVerba
#
#
#
# # Fitting models
# real_model_fit2 <- brm(
#   ind_m1, data = real_df, prior = p2, sample_prior = T, backend = "cmdstanr", chains = 2, cores = 2,
#
# pp_check(real_model_fit2, ndraws = 100)
#
#
# real_model_fit3 <- brm(
#   env_m2, data = real_df, prior = p2, sample_prior = T, backend = "cmdstanr", chains = 2, cores = 2,
#
# pp_check(real_model_fit3, ndraws = 100)
#
# real_model_fit4 <- brm(
#   ind_env_m3, data = real_df, prior = p2, sample_prior = T, backend = "cmdstanr", chains = 2, cores =
#
# pp_check(real_model_fit4, ndraws = 100)
```

```
# # Add criterions to models
#
# real_model_fit_loo1 <- add_criterion(real_model_fit1, criterion = "loo")
#
# real_model_fit_loo2 <- add_criterion(real_model_fit2, criterion = "loo")
#
# real_model_fit_loo3 <- add_criterion(real_model_fit3, criterion = "loo")
#
# real_model_fit_loo4 <- add_criterion(real_model_fit4, criterion = "loo")
#
# # Compare models on information criterion
#
# loo_compare(real_model_fit_loo1, real_model_fit_loo2, real_model_fit_loo3, real_model_fit_loo4)
#
# loo_model_weights(real_model_fit_loo1, real_model_fit_loo2, real_model_fit_loo3, real_model_fit_loo4)
```

```
# # Split in k folds
# kfold1 <- kfold(real_model_fit1, folds = "stratified", group = "Child.ID", K = 5,save_fits = TRUE)
#
# kfold2 <- kfold(real_model_fit2, folds = "stratified", group = "Child.ID", K = 5,save_fits = TRUE)
#
# kfold3 <- kfold(real_model_fit3, folds = "stratified", group = "Child.ID", K = 5,save_fits = TRUE)
#
# kfold4 <- kfold(real_model_fit4, folds = "stratified", group = "Child.ID", K = 5,save_fits = TRUE)
#
#
#
# # Define loss function
# rmse <- function(y,yrep) {
#   yrep_mean <- colMeans(yrep)
#   sqrt(mean((yrep_mean-y)^2))
# }
#
# # Predict responses and evaluate the loss
# kfp1 <- kfold_predict(kfold1)
# rmse1 <- rmse(kfp1$y, kfp1$yrep)
#
# kfp2 <- kfold_predict(kfold2)
# rmse2 <- rmse(kfp2$y, kfp2$yrep)
#
# kfp3 <- kfold_predict(kfold3)
# rmse3 <- rmse(kfp3$y, kfp3$yrep)
#
# kfp4 <- kfold_predict(kfold4)
# rmse4 <- rmse(kfp4$y, kfp4$yrep)
#
# # Plotting rmse for cross-validation
#
# rmse_plot <- tibble(
#   model = c("fit1", "fit2", "fit3", "fit4"),
#   rmse = c(rmse1, rmse2, rmse3, rmse4)
# )
#
```

```
# ggplot(rmse_plot, aes(model, rmse)) +
#   geom_point(size = 5)
```