Assignment 2 - Meta-analysis of pitch in schizophrenia

Riccardo Fusaroli

16/8/2022

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 discuss what this implies for your model. remember to use at least one plot to visualize your results. BONUS question: do a power/precision analysis: w this kind of sample sizes (participants) how many studies would you need to acquire good precision (e.g. .1 sd in the pop level estimate)

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 - 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: assess the effect of task on the estimates (model comparison with baseline model)

Question 1

```
# 1. Ground truth level: the real underlying truth  - norm(0.4,0.4)
# 2. Sample level(Study level: 100 samples(studies) - norm(mu drawn from the above, sd drawn from the a
# 3. Participant level: tnorm(20, 10, 10) participants in each study
# 4. Results level: estimates of regressions based on the participants
#        -->(I) Fit no. 1 Bayes model: Given the study results what are the estimated true parameters?
# 5. Published results level: include publication bias filter
#        -->(II) Fit no. 2 Bayes model: Given the **publilshed** results what are the estimated true pa
#       --> Compare the 2 models
```

```
simulate_data <- function(n = 100, gt_mean = 0.4, gt_sd = 0.4, error = 0.8, seed = 1)
  # n - number of studies, gt stands for 'ground-truth'
  {
  set.seed(seed)
```

```
  data <- tibble(study = seq(1, n, by = 1)) %>%
    rowwise %>%
    mutate(t_effect = rnorm(1, gt_mean, gt_sd) %>% round(2),
           sample_size = rtnorm(1, 20, 10, lower = 10) %>% round,
           ind_effects = list(rnorm(sample_size, mean = t_effect, sd = error) %>% round(2)),
           effect = mean(ind_effects),
           effect_sigma = sd(ind_effects) / sqrt(sample_size),
           ci_lower = effect - 1.96 * effect_sigma, # just so it's easier to plot later
           ci_upper = effect + 1.96 * effect_sigma,
           signif = if_else(abs(effect) - 1.96 * effect_sigma > 0, 'yes' , 'no'),
           pub = if_else(signif == 'yes' & effect > 0, rbinom(1, 1, 0.9), rbinom(1, 1, 0.1))) %>%
    ungroup %>%
    relocate(c(t_effect, sample_size, ind_effects), .after = pub)

}
```

Checking if all worked fine

```
check <- simulate_data(n = 10000)

check %>% summarise(mean(t_effect), sd(t_effect))
```

```
## # A tibble: 1 x 2
##   `mean(t_effect)` `sd(t_effect)`
##              <dbl>          <dbl>
## 1            0.397          0.405
```

```
n_tot <- check %>%
  count(signif == 'yes' & effect > 0) %>%
  pull(n)

check %>% filter(pub == 1) %>%
  count(signif == 'yes' & effect > 0, pub) %>%
  mutate(pct = n / n_tot)
```

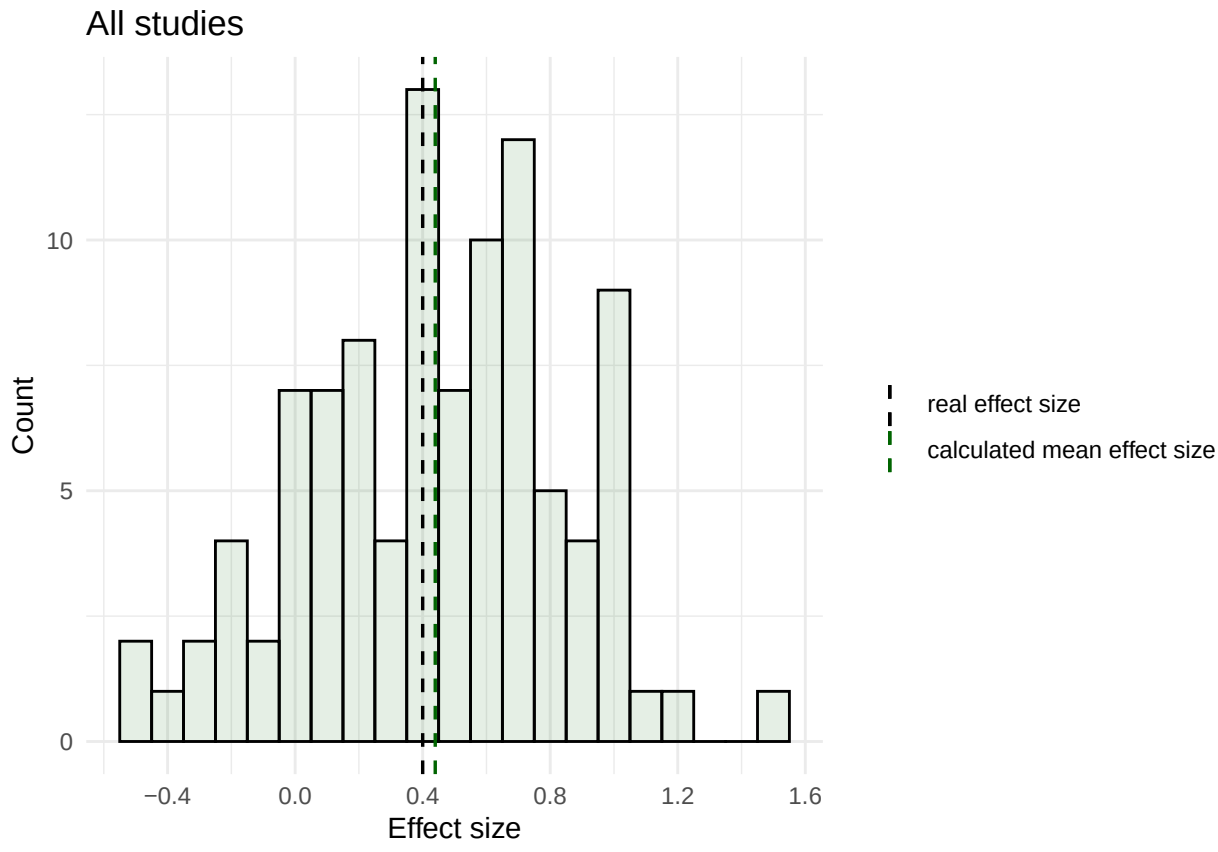
```
## # A tibble: 2 x 4
##   `signif == "yes" & effect > 0`   pub     n    pct
##   <lgl>                          <int> <int>  <dbl>
## 1 FALSE                              1   449 0.0996
## 2 TRUE                               1  4932 0.898
```

```
rm(n_tot, check)
```

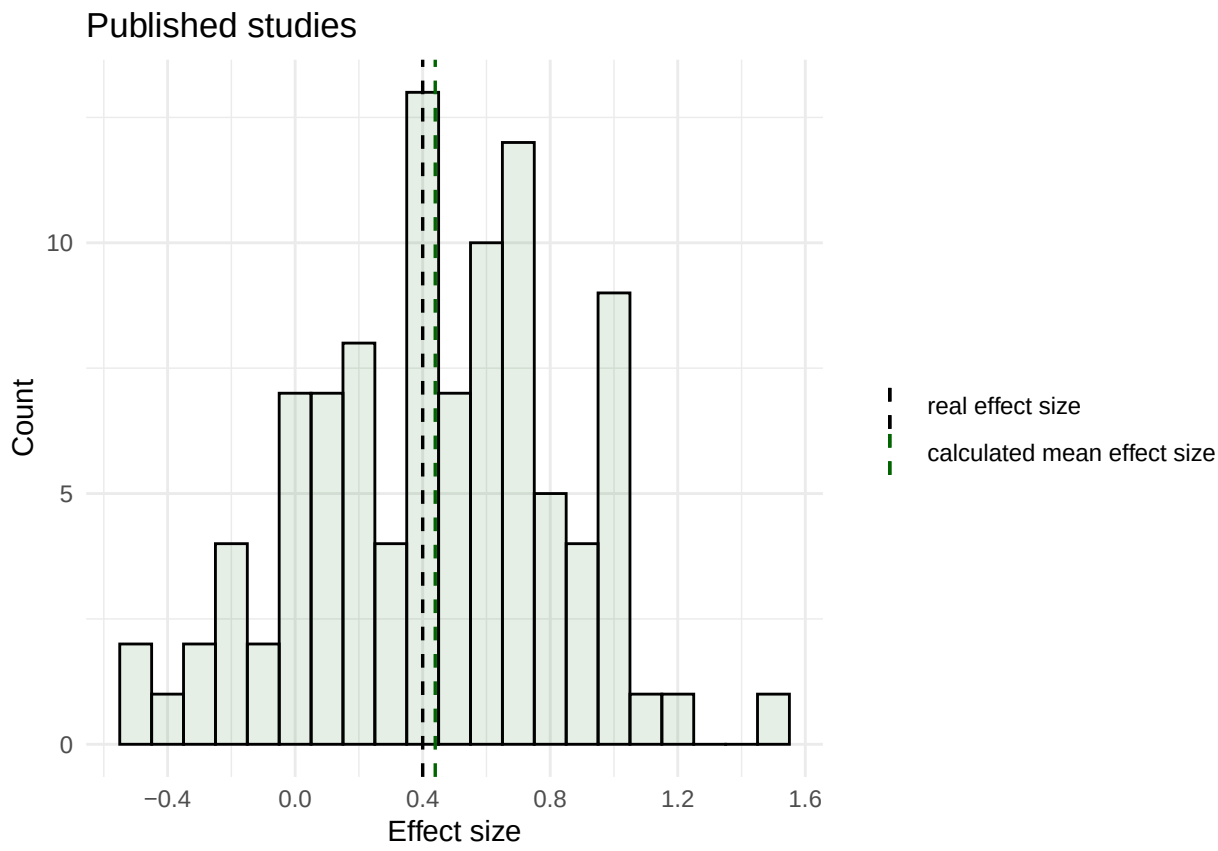
```
data <- simulate_data()
data_pub <- data %>%
  filter(pub == 1)

dfs <- list(data, data_pub)
```

```
map2(
  .x = dfs, .y = c("All studies", "Published studies"),
  .f = function(.x, .y){
  ggplot(data, aes(x = effect)) +
    geom_histogram(binwidth = 0.1, fill = 'darkgreen', color = 'black', alpha = 0.1) +
    geom_vline(aes(xintercept = 0.4, color = 'real effect size'),
               linetype = 'dashed', size = 0.6) +
    geom_vline(aes(xintercept = mean(effect), color = 'calculated mean effect size'),
               linetype = 'dashed', size = 0.6) +
    scale_x_continuous(n.breaks = 8) +
    labs(title = .y,
         x = 'Effect size',
         y = 'Count') +
    scale_color_manual(name = element_blank(), values = c(`real effect size` = "black", `calculated mea
    theme_minimal()
})
```

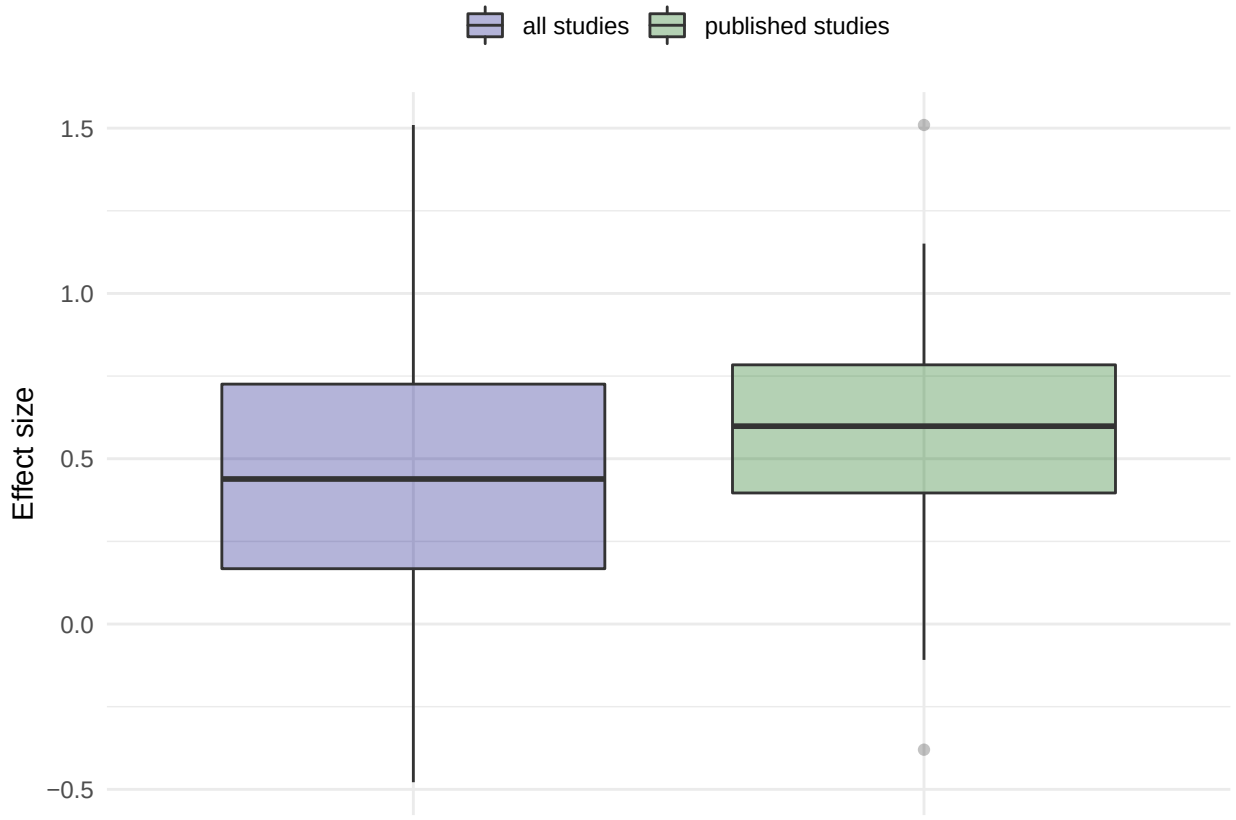
```
## [[1]]
```




```
##
## [[2]]
```




```
rbind(data %>% mutate(index = 'all'),
      data_pub %>% mutate(index = 'published')) %>%
ggplot(aes(x = index, y = effect, fill = index)) +
  geom_boxplot(alpha = 0.3) +
  theme_minimal() +
  guides(alpha = 'none') +
  scale_fill_manual(values = c('navy', 'darkgreen'),
                    labels = c('all studies', 'published studies'),
                    name = element_blank()) +
  ylab('Effect size') +
  theme(axis.text.x = element_blank(),
        axis.title.x = element_blank(),
        legend.position = 'top')
```




```
funnel_plot <- function(data, n_small_big_cutoff = 30, null = 0){
  effect_mean <-  mean(data$effect)

  line_data = tibble(
    se_line =  seq(0, max(data$effect_sigma), by = 0.001),

    line_u95 =  effect_mean + 1.96*se_line,
    line_l95 =  effect_mean - 1.96*se_line,

    line_u99 =  effect_mean + 3.29*se_line,
    line_l99 =  effect_mean -3.29*se_line)

  data %>% mutate(`Sample size` = if_else(sample_size < n_small_big_cutoff, 'small', 'big')) %>%
    ggplot(aes(x = effect, y = effect_sigma)) +
      scale_y_reverse() +
      geom_point(aes(shape = `Sample size` )) +
      geom_line(aes(x = line_u95, y = se_line), linetype = 'dashed', data = line_data) +
      geom_line(aes(x = line_l95, y = se_line), linetype = 'dashed', data = line_data) +
      geom_line(aes(x = line_u99, y = se_line), linetype = 'dotted', data = line_data) +
      geom_line(aes(x = line_l99, y = se_line), linetype = 'dotted', data = line_data) +
      geom_segment(aes(x = null, y = 0, xend = null, yend = max(effect_sigma), colour = 'Null hypothesis
                 linetype = 'dashed',
                 size = 0.6,
                 alpha = 0.1) +
      theme_minimal() +
      scale_shape_manual(values = c('circle', 'circle open')) +
```

```
      scale_color_manual(values = c(`Null hypothesis` = 'darkred')) +
      labs(x = 'Effect size',
           y = 'Standard Error')
}
```

```
map(.x = dfs, .f = function(.x){
  observed_mean <- mean(.x$effect)
  true_mean <- 0.4

  funnel_plot(.x) +
    geom_segment(aes(x = observed_mean,
                     y = 0,
                     xend = observed_mean,
                     yend = max(effect_sigma),
                     colour = 'Mean sample effect size'),
                 linetype = 'solid',
                 size = 1) +
    geom_segment(aes(x = true_mean,
                     y = 0,
                     xend = true_mean,
                     yend = max(effect_sigma),
                     colour = 'True parameter value'),
                 linetype = 'dashed',
                 size = 1)
    scale_colour_manual(values = c(`Mean sample effect size` = 'black',
                                   `True parameter value` = 'grey',
                                   `Null hypothesis` = 'darkred'))
})
```

```
## [[1]]
## <ggproto object: Class ScaleDiscrete, Scale, gg>
##     aesthetics: colour
##     axis_order: function
##     break_info: function
##     break_positions: function
##     breaks: waiver
##     call: call
##     clone: function
##     dimension: function
##     drop: TRUE
##     expand: waiver
##     get_breaks: function
##     get_breaks_minor: function
##     get_labels: function
##     get_limits: function
##     guide: legend
##     is_discrete: function
##     is_empty: function
##     labels: waiver
##     limits: Mean sample effect size True parameter value Null hypothesis
##     make_sec_title: function
##     make_title: function
##     map: function
```

```
##     map_df: function
##     n.breaks.cache: NULL
##     na.translate: TRUE
##     na.value: grey50
##     name: waiver
##     palette: function
##     palette.cache: NULL
##     position: left
##     range: <ggproto object: Class RangeDiscrete, Range, gg>
##         range: NULL
##         reset: function
##         train: function
##         super:  <ggproto object: Class RangeDiscrete, Range, gg>
##     rescale: function
##     reset: function
##     scale_name: manual
##     train: function
##     train_df: function
##     transform: function
##     transform_df: function
##     super:  <ggproto object: Class ScaleDiscrete, Scale, gg>
##
## [[2]]
## <ggproto object: Class ScaleDiscrete, Scale, gg>
##     aesthetics: colour
##     axis_order: function
##     break_info: function
##     break_positions: function
##     breaks: waiver
##     call: call
##     clone: function
##     dimension: function
##     drop: TRUE
##     expand: waiver
##     get_breaks: function
##     get_breaks_minor: function
##     get_labels: function
##     get_limits: function
##     guide: legend
##     is_discrete: function
##     is_empty: function
##     labels: waiver
##     limits: Mean sample effect size True parameter value Null hypothesis
##     make_sec_title: function
##     make_title: function
##     map: function
##     map_df: function
##     n.breaks.cache: NULL
##     na.translate: TRUE
##     na.value: grey50
##     name: waiver
##     palette: function
##     palette.cache: NULL
##     position: left
```

```
##     range: <ggproto object: Class RangeDiscrete, Range, gg>
##         range: NULL
##         reset: function
##         train: function
##         super:  <ggproto object: Class RangeDiscrete, Range, gg>
##     rescale: function
##     reset: function
##     scale_name: manual
##     train: function
##     train_df: function
##     transform: function
##     transform_df: function
##     super:  <ggproto object: Class ScaleDiscrete, Scale, gg>
```

Defining the formula and priors

```
f <- bf(effect | se(effect_sigma) ~ 1 + (1|study))

get_prior(f, data)
```

```
##                   prior     class      coef group resp dpar nlpar lb ub
##  student_t(3, 0.4, 2.5) Intercept
##    student_t(3, 0, 2.5)        sd                                   0
##    student_t(3, 0, 2.5)        sd           study                   0
##    student_t(3, 0, 2.5)        sd Intercept study                   0
##        source
##       default
##       default
##  (vectorized)
##  (vectorized)
```

```
priors <- c(prior(normal(0, 0.5), class = Intercept),
            prior(normal(0, 0.6), class = sd))
```

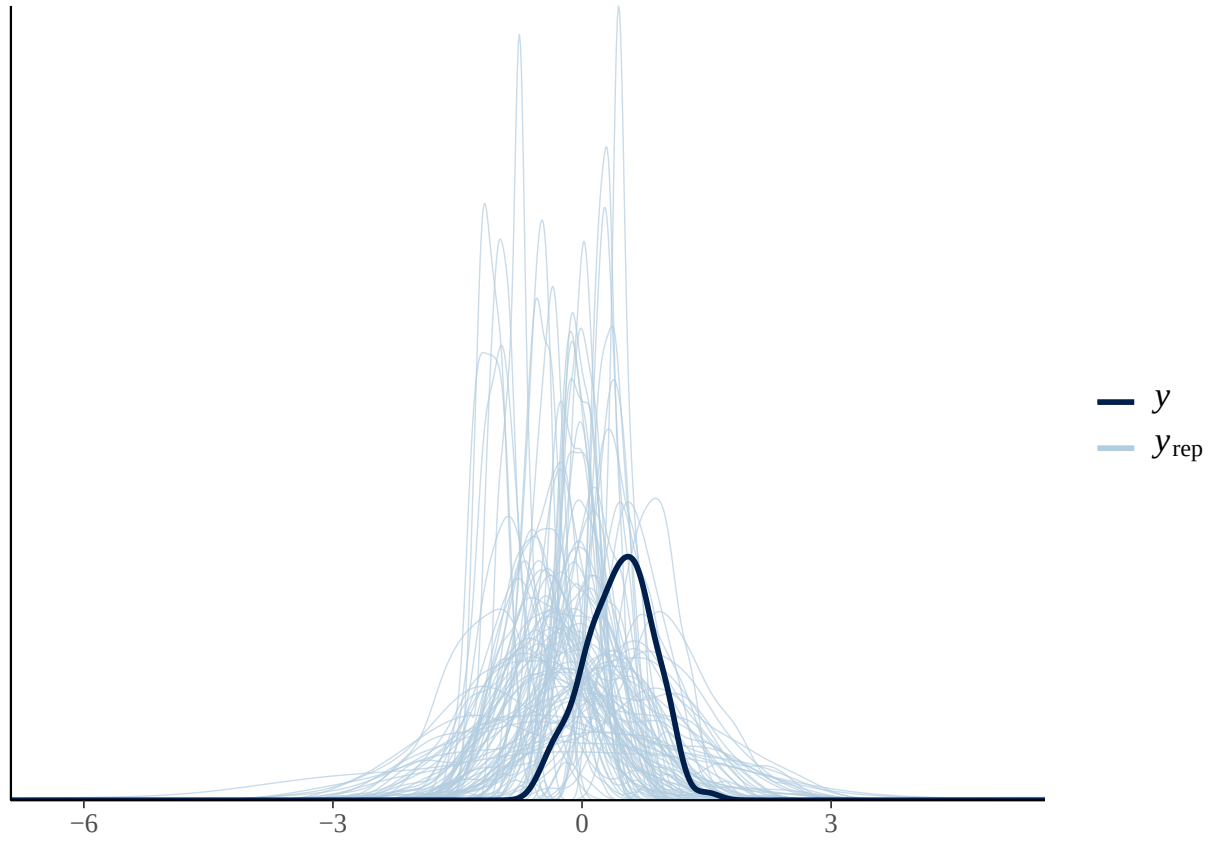
```
prior_m_all <- brm(f,
                   data,
                   family = gaussian,
                   prior = priors,
                   sample_prior = 'only',
                   backend = 'cmdstanr',
                   cores = 3
                   )
```

```
## Start sampling

## Running MCMC with 4 chains, at most 3 in parallel...
##
## Chain 1 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 1 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 1 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 1 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1 Iteration:  900 / 2000 [ 45%]  (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 2 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 2 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 2 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3 Iteration:  100 / 2000 [  5%]  (Warmup)
## Chain 3 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3 Iteration:  300 / 2000 [ 15%]  (Warmup)
## Chain 3 Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 3 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 3 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 1 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 1 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 1 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2 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 2 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 2 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 2 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3 Iteration:  900 / 2000 [ 45%]  (Warmup)
## Chain 3 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 3 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 3 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 3 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 1 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1 Iteration: 1900 / 2000 [ 95%]  (Sampling)
```

```
## Chain 2 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 2 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 3 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 3 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 3 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1 finished in 0.7 seconds.
## Chain 2 finished in 0.6 seconds.
## Chain 3 finished in 0.7 seconds.
## Chain 4 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4 Iteration:  100 / 2000 [  5%]  (Warmup)
## Chain 4 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4 Iteration:  300 / 2000 [ 15%]  (Warmup)
## Chain 4 Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 4 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 4 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4 Iteration:  900 / 2000 [ 45%]  (Warmup)
## Chain 4 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 4 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 4 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 4 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 4 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 4 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4 finished in 0.5 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.6 seconds.
## Total execution time: 1.5 seconds.
```

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




```
summary(prior_m_all)
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: effect | se(effect_sigma) ~ 1 + (1 | study)
##    Data: data (Number of observations: 100)
##   Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup draws = 4000
##
## Group-Level Effects:
## ~study (Number of levels: 100)
##               Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(Intercept)     0.48      0.35     0.02     1.32 1.00     3087     2139
##
## Population-Level Effects:
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept    -0.01      0.49    -0.95     0.95 1.01     7417     2897
##
## 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).
```

Fitting the models

```
m_all <- brm(f,
             data,
             family = gaussian,
             prior = priors,
             sample_prior = T,
             backend = 'cmdstanr',
             cores = 3
             )
```

```
## Start sampling

## Running MCMC with 4 chains, at most 3 in parallel...
##
## Chain 1 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3 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 1 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 1 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 1 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1 Iteration:  900 / 2000 [ 45%]  (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 2 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 2 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 2 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3 Iteration:  100 / 2000 [  5%]  (Warmup)
## Chain 3 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3 Iteration:  300 / 2000 [ 15%]  (Warmup)
## Chain 3 Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 3 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 3 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 2 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 3 Iteration:  900 / 2000 [ 45%]  (Warmup)
## Chain 3 Iteration: 1000 / 2000 [ 50%]  (Warmup)
```

```
## Chain 3 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 1 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 2 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 3 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 1 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 2 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 2 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 2 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 3 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 2 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 2 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 3 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 3 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 1 finished in 1.1 seconds.
## Chain 2 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2 finished in 1.1 seconds.
## Chain 3 finished in 1.1 seconds.
## Chain 4 Iteration:  100 / 2000 [  5%]  (Warmup)
## Chain 4 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4 Iteration:  300 / 2000 [ 15%]  (Warmup)
## Chain 4 Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 4 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 4 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4 Iteration:  900 / 2000 [ 45%]  (Warmup)
## Chain 4 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 4 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 4 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 4 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 4 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 4 Iteration: 2000 / 2000 [100%]  (Sampling)
```

```
## Chain 4 finished in 0.9 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 1.0 seconds.
## Total execution time: 2.1 seconds.
```

```
m_pub <- update(m_all,
                newdata = data_pub)
```

```
## Start sampling
```

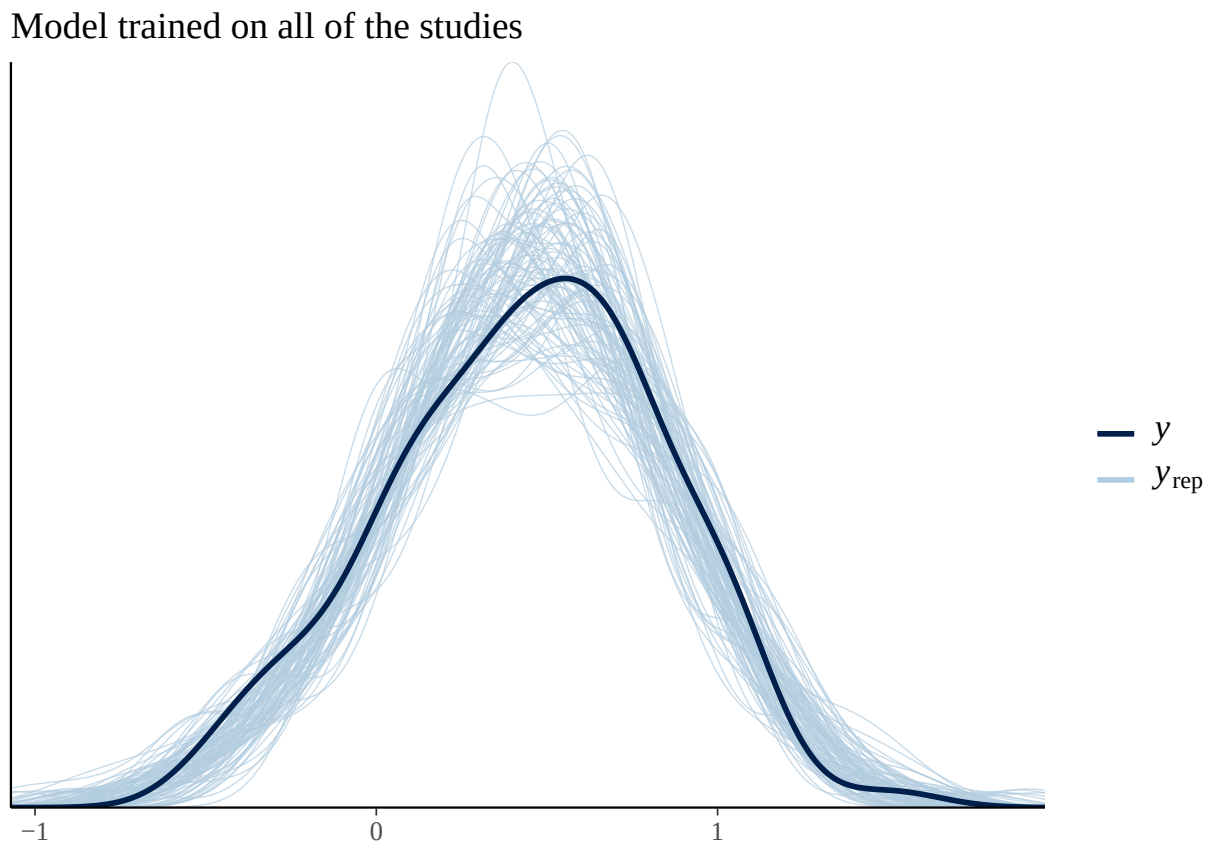
```
## Running MCMC with 4 sequential chains...
##
## Chain 1 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1 Iteration:  100 / 2000 [  5%]  (Warmup)
## Chain 1 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1 Iteration:  300 / 2000 [ 15%]  (Warmup)
## Chain 1 Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 1 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1 Iteration:  700 / 2000 [ 35%]  (Warmup)
## 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 1 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 1 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 1 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 1 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## 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 0.6 seconds.
## 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 2 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 2 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2 Iteration:  700 / 2000 [ 35%]  (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 2 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 2 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 2 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2 Iteration: 1500 / 2000 [ 75%]  (Sampling)
```

```
## Chain 2 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 2 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 2 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2 finished in 0.6 seconds.
## Chain 3 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3 Iteration:  100 / 2000 [  5%]  (Warmup)
## Chain 3 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3 Iteration:  300 / 2000 [ 15%]  (Warmup)
## Chain 3 Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 3 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 3 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3 Iteration:  900 / 2000 [ 45%]  (Warmup)
## Chain 3 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 3 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 3 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 3 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 3 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 3 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3 finished in 0.6 seconds.
## Chain 4 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4 Iteration:  100 / 2000 [  5%]  (Warmup)
## Chain 4 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4 Iteration:  300 / 2000 [ 15%]  (Warmup)
## Chain 4 Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 4 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 4 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4 Iteration:  900 / 2000 [ 45%]  (Warmup)
## Chain 4 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 4 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 4 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 4 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 4 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 4 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4 finished in 0.7 seconds.
##
## All 4 chains finished successfully.
```

```
## Mean chain execution time: 0.6 seconds.
## Total execution time: 2.9 seconds.
```

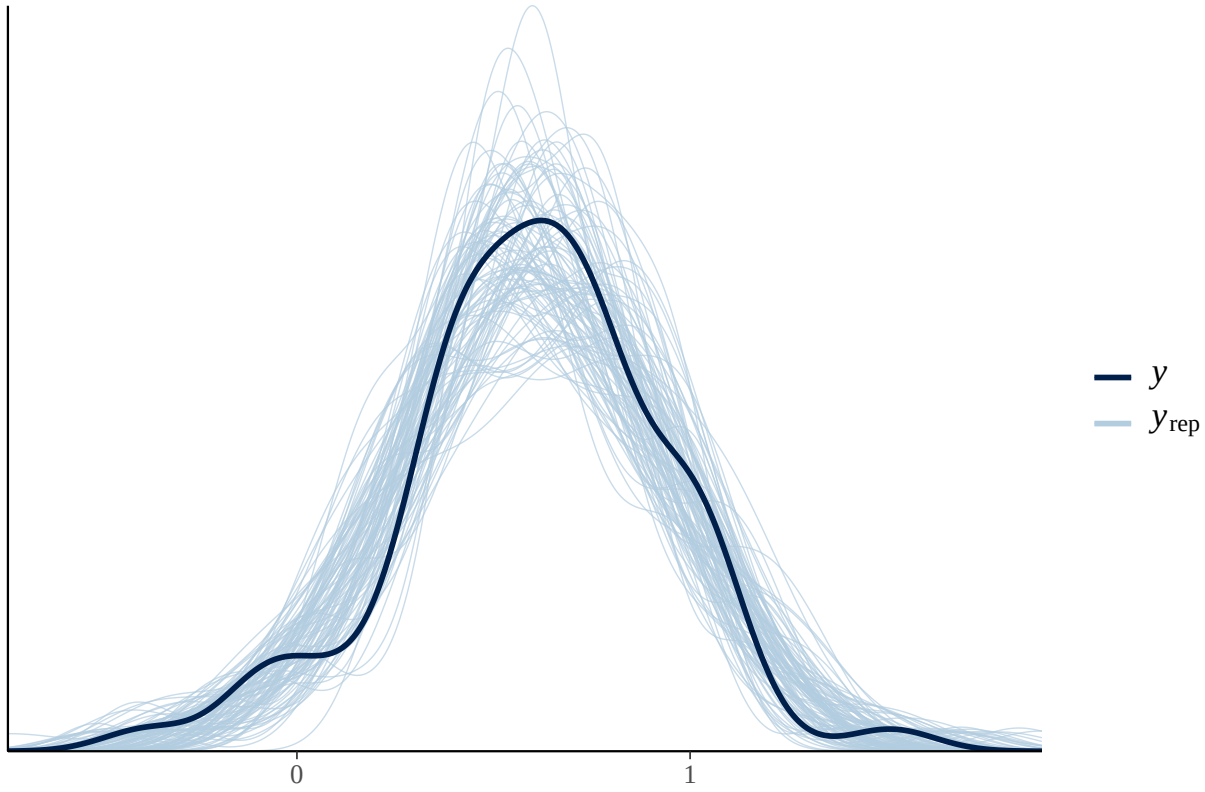
```
models <- list(m_all, m_pub)
```

```
pp_check(m_all, ndraws = 100)+
  ggtitle("Model trained on all of the studies")
```




```
pp_check(m_pub, ndraws = 100)+
  ggtitle("Model trained on published studies")
```

Model trained on published studies




```
summary(m_all)
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: effect | se(effect_sigma) ~ 1 + (1 | study)
##    Data: data (Number of observations: 100)
##   Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup draws = 4000
##
## Group-Level Effects:
## ~study (Number of levels: 100)
##               Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(Intercept)     0.36      0.03     0.30     0.43 1.00     1424     2052
##
## Population-Level Effects:
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept     0.44      0.04     0.36     0.52 1.00     1390     1649
##
## 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).
```

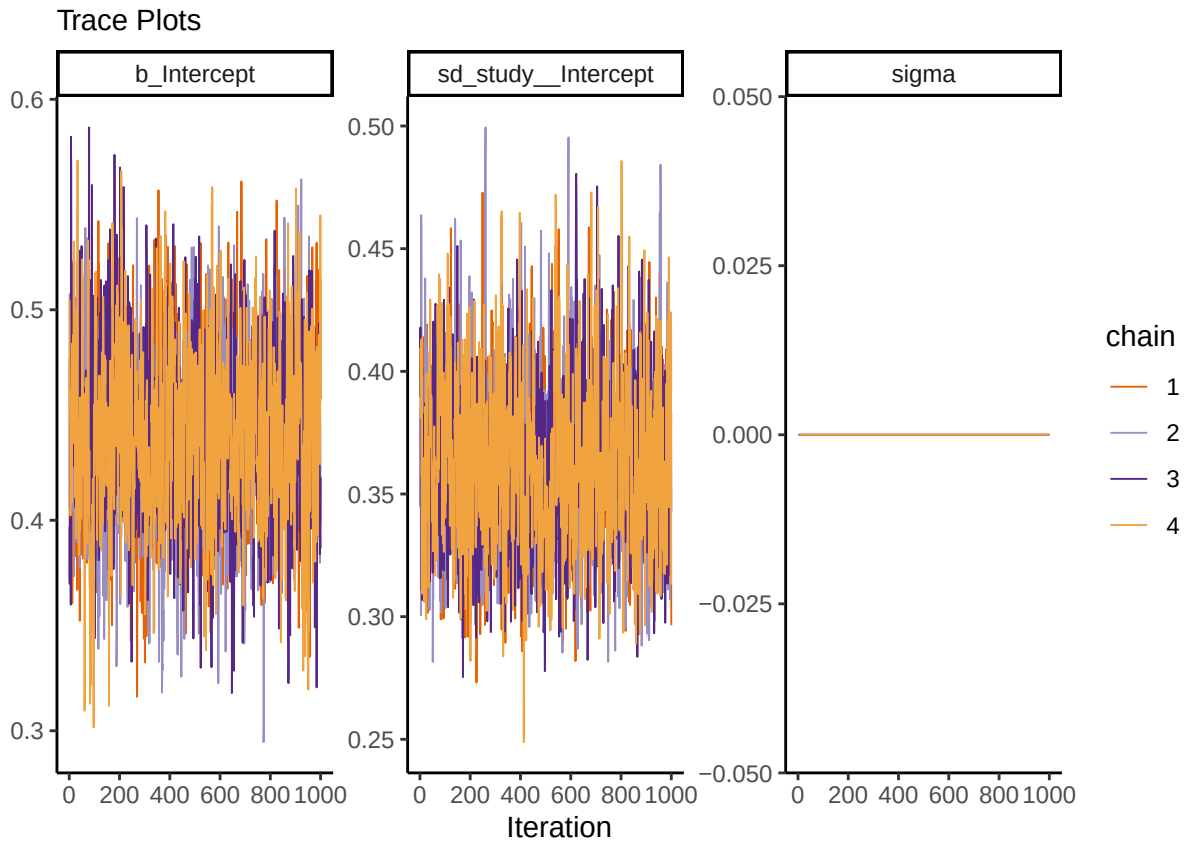
```
summary(m_pub)
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: effect | se(effect_sigma) ~ 1 + (1 | study)
##    Data: data_pub (Number of observations: 68)
##   Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup draws = 4000
##
## Group-Level Effects:
## ~study (Number of levels: 68)
##               Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(Intercept)     0.27      0.04     0.21     0.35 1.00     1419     2056
##
## Population-Level Effects:
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept     0.60      0.04     0.52     0.68 1.00     2346     2443
##
## 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).
```

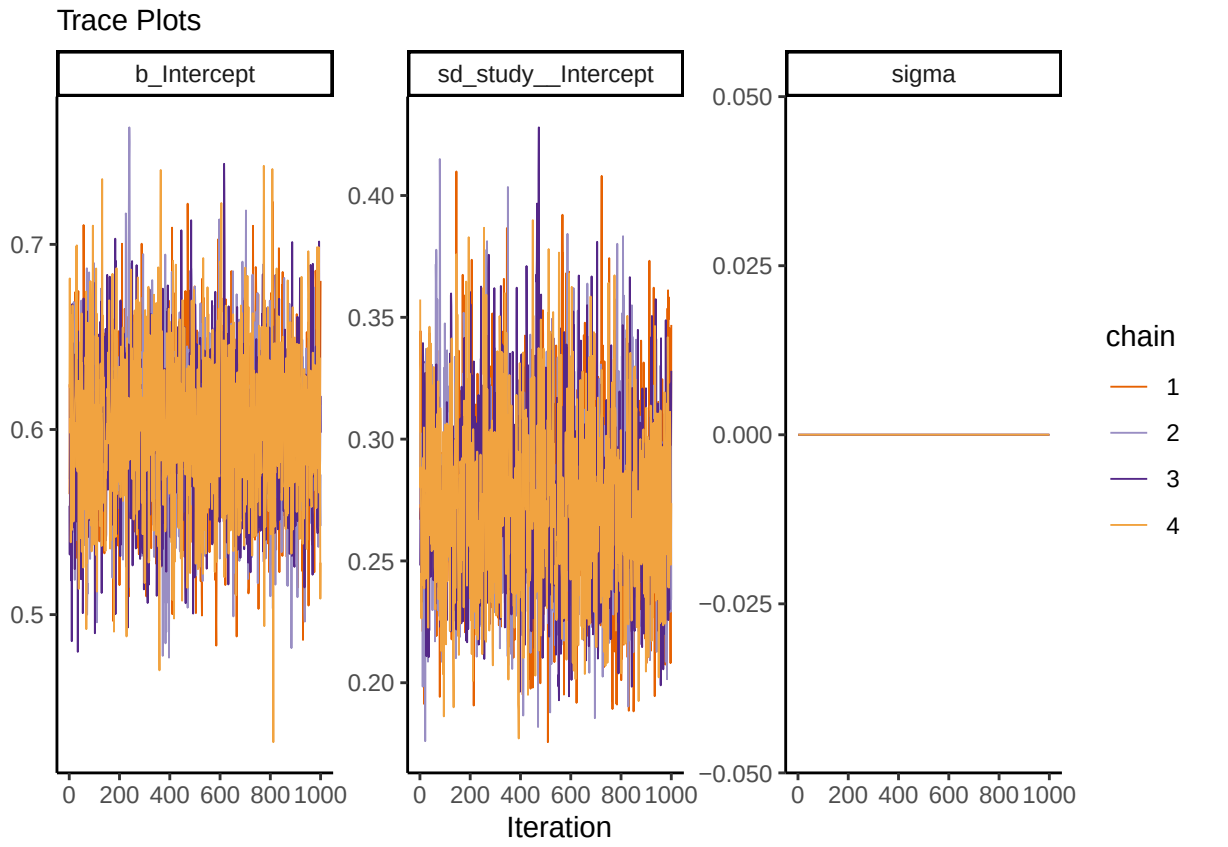
Convergance tests

```
# launch_shinystan(f_m) # - very nice for exploring and diagnosing the model, but opens up in a new win

map(.x = models, ~ mcmc_plot(.x, type = 'trace') +
    theme_classic() +
    scale_color_manual(values=c("#E66101", "#998EC3", "#542788", "#F1A340")) +
    ylab("") +
    xlab("Iteration") +
    labs(subtitle = 'Trace Plots'))
```

```
## No divergences to plot.
## Scale for 'colour' is already present. Adding another scale for 'colour',
## which will replace the existing scale.
##
## No divergences to plot.
## Scale for 'colour' is already present. Adding another scale for 'colour',
## which will replace the existing scale.

## [[1]]
```

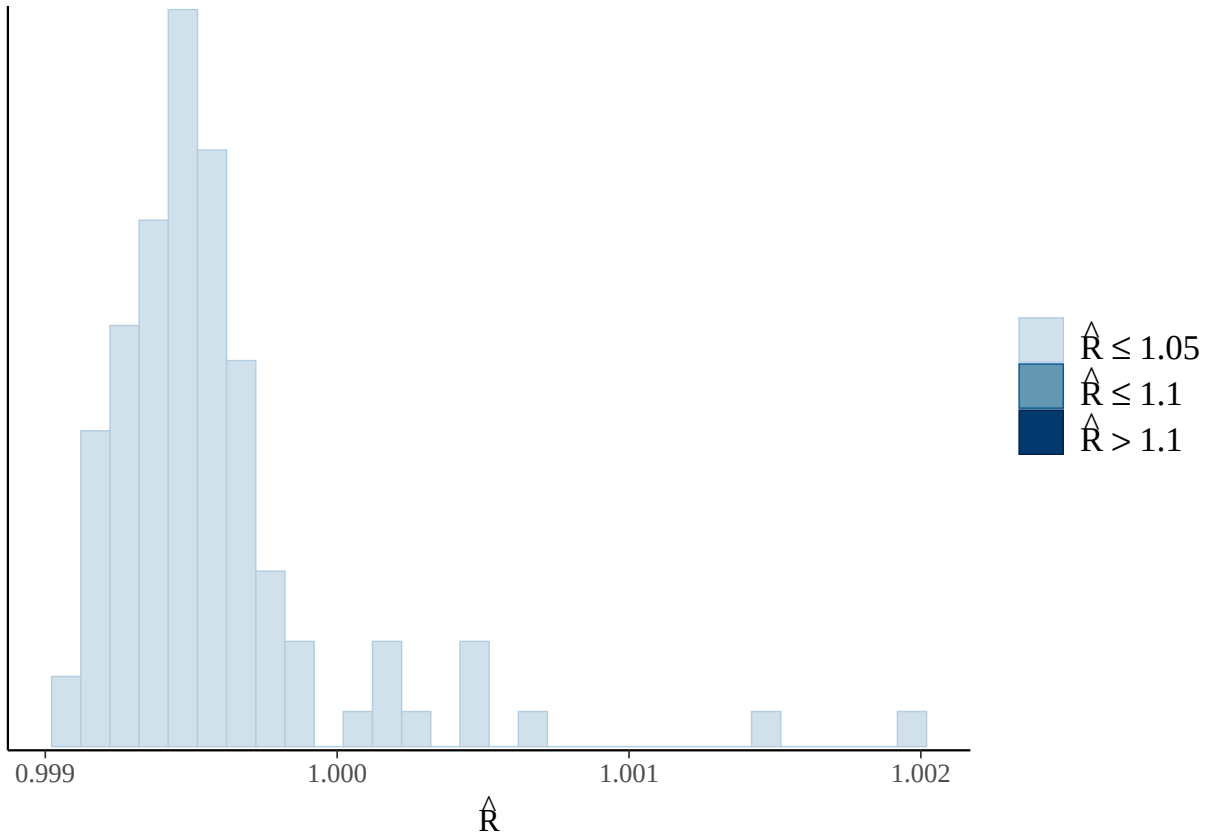

Trace Plots
b_Intercept
sd_study__Intercept
sigma
0.6
0.5
0.4
0.3
0.50
0.45
0.40
0.35
0.30
0.25
0.050
0.025
0.000
−0.025
−0.050
0
200
400
600
800
1000
Iteration
chain
1
2
3
4

```
##
## [[2]]
```

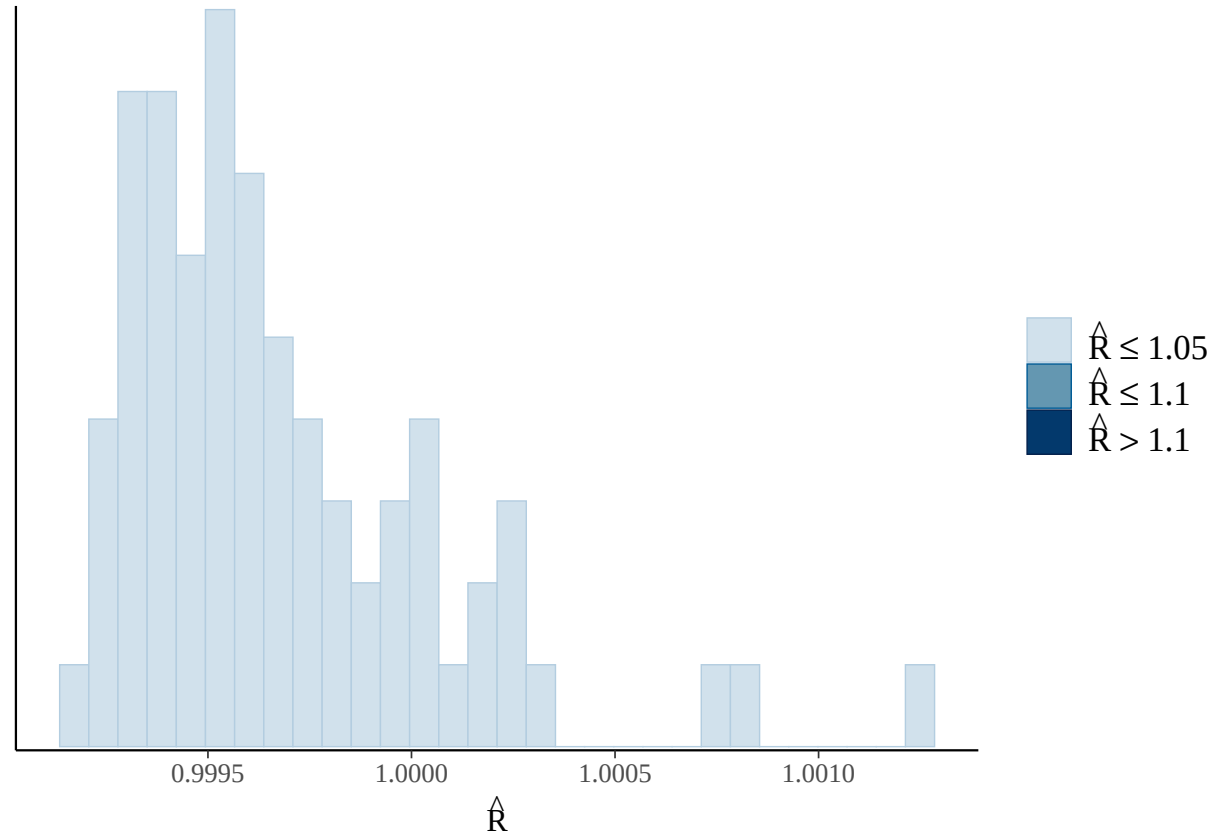



```
map(.x = models, ~ mcmc_plot(.x, type = 'rhat_hist'))
```

```
## Warning: Dropped 1 NAs from 'new_rhat(rhat)'.
## Dropped 1 NAs from 'new_rhat(rhat)'.

## [[1]]

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
##
## [[2]]

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

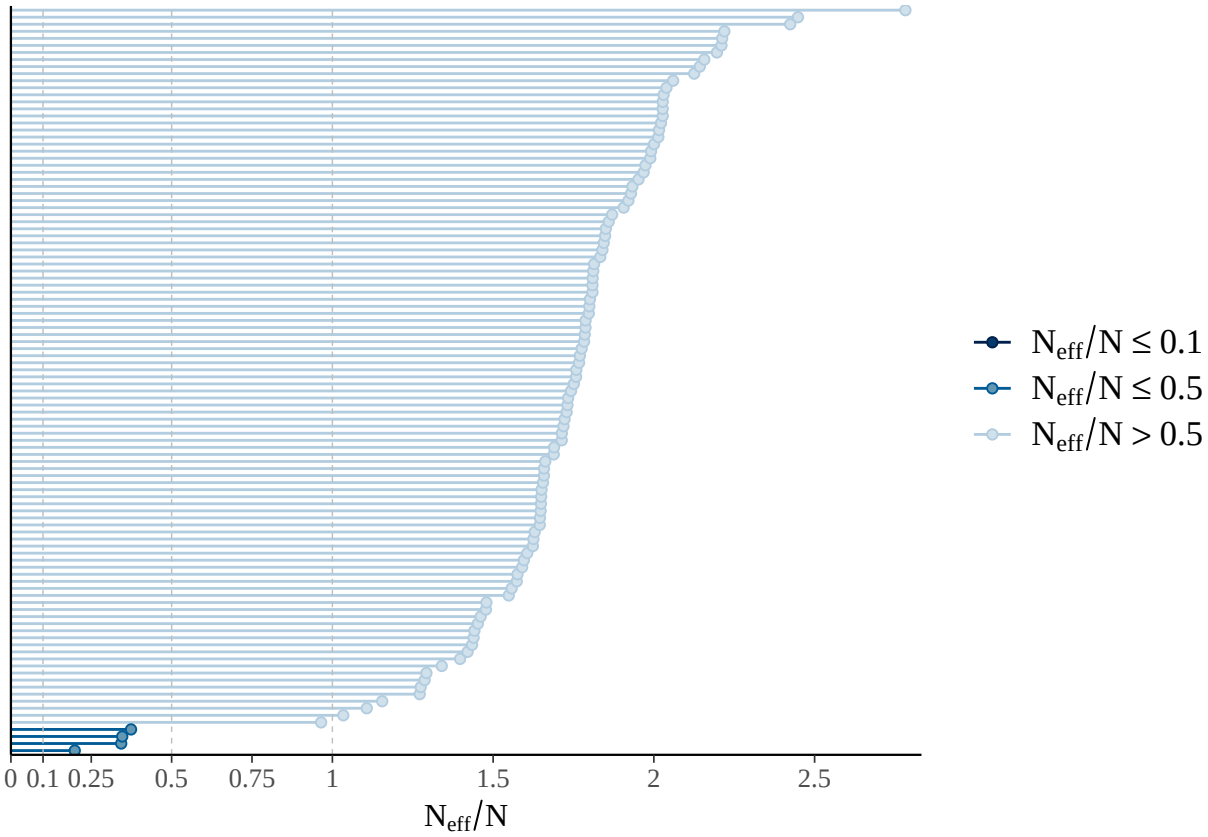



```
map(.x = models, ~ mcmc_plot(.x, type = 'neff'))
```

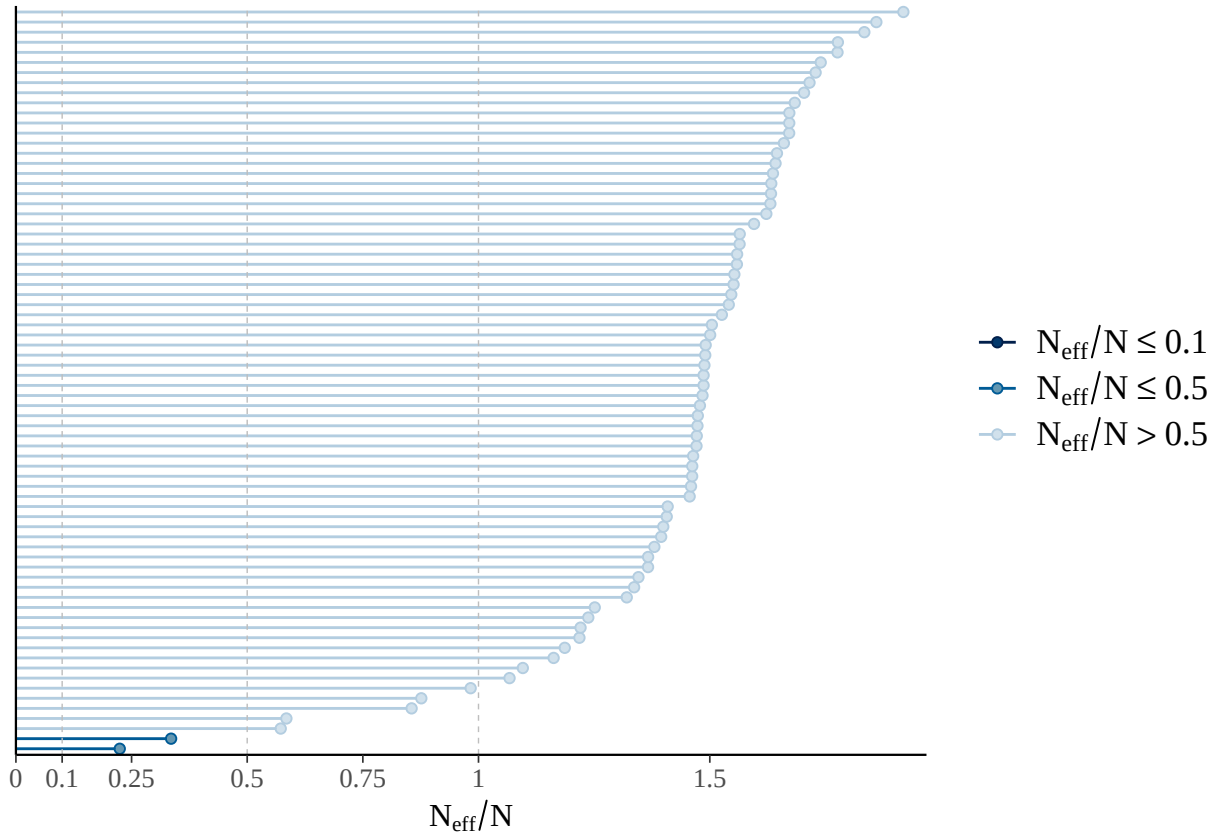
```
## Warning: Dropped 1 NAs from 'new_neff_ratio(ratio)'.
```

```
## Warning: Dropped 1 NAs from 'new_neff_ratio(ratio)'.
```

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




```
##
## [[2]]
```



Posterior-prior checks

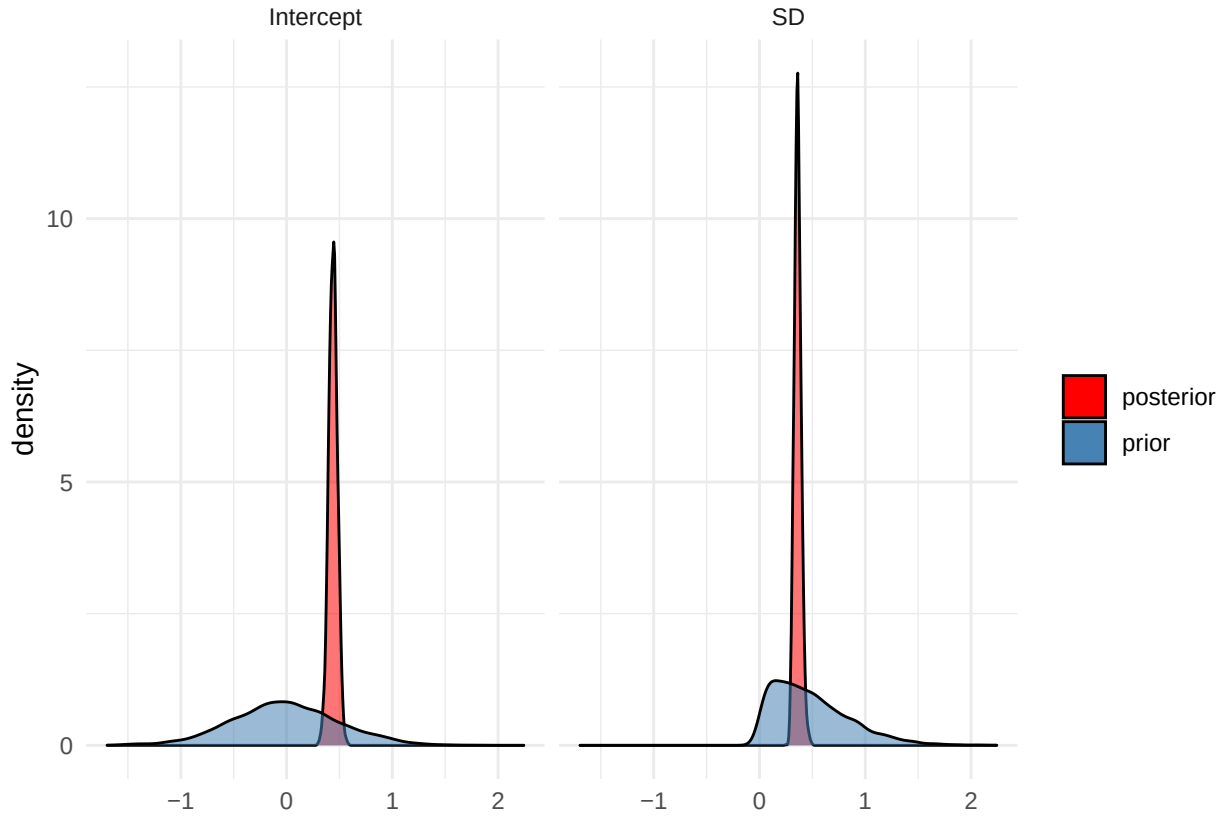
```
get_variables(m_all)
```

```
##   [1] "b_Intercept"            "sd_study__Intercept"    "sigma"
##   [4] "r_study[1,Intercept]"   "r_study[2,Intercept]"   "r_study[3,Intercept]"
##   [7] "r_study[4,Intercept]"   "r_study[5,Intercept]"   "r_study[6,Intercept]"
##  [10] "r_study[7,Intercept]"   "r_study[8,Intercept]"   "r_study[9,Intercept]"
##  [13] "r_study[10,Intercept]"  "r_study[11,Intercept]"  "r_study[12,Intercept]"
##  [16] "r_study[13,Intercept]"  "r_study[14,Intercept]"  "r_study[15,Intercept]"
##  [19] "r_study[16,Intercept]"  "r_study[17,Intercept]"  "r_study[18,Intercept]"
##  [22] "r_study[19,Intercept]"  "r_study[20,Intercept]"  "r_study[21,Intercept]"
##  [25] "r_study[22,Intercept]"  "r_study[23,Intercept]"  "r_study[24,Intercept]"
##  [28] "r_study[25,Intercept]"  "r_study[26,Intercept]"  "r_study[27,Intercept]"
##  [31] "r_study[28,Intercept]"  "r_study[29,Intercept]"  "r_study[30,Intercept]"
##  [34] "r_study[31,Intercept]"  "r_study[32,Intercept]"  "r_study[33,Intercept]"
##  [37] "r_study[34,Intercept]"  "r_study[35,Intercept]"  "r_study[36,Intercept]"
##  [40] "r_study[37,Intercept]"  "r_study[38,Intercept]"  "r_study[39,Intercept]"
##  [43] "r_study[40,Intercept]"  "r_study[41,Intercept]"  "r_study[42,Intercept]"
##  [46] "r_study[43,Intercept]"  "r_study[44,Intercept]"  "r_study[45,Intercept]"
##  [49] "r_study[46,Intercept]"  "r_study[47,Intercept]"  "r_study[48,Intercept]"
##  [52] "r_study[49,Intercept]"  "r_study[50,Intercept]"  "r_study[51,Intercept]"
##  [55] "r_study[52,Intercept]"  "r_study[53,Intercept]"  "r_study[54,Intercept]"
##  [58] "r_study[55,Intercept]"  "r_study[56,Intercept]"  "r_study[57,Intercept]"
##  [61] "r_study[58,Intercept]"  "r_study[59,Intercept]"  "r_study[60,Intercept]"
##  [64] "r_study[61,Intercept]"  "r_study[62,Intercept]"  "r_study[63,Intercept]"
```

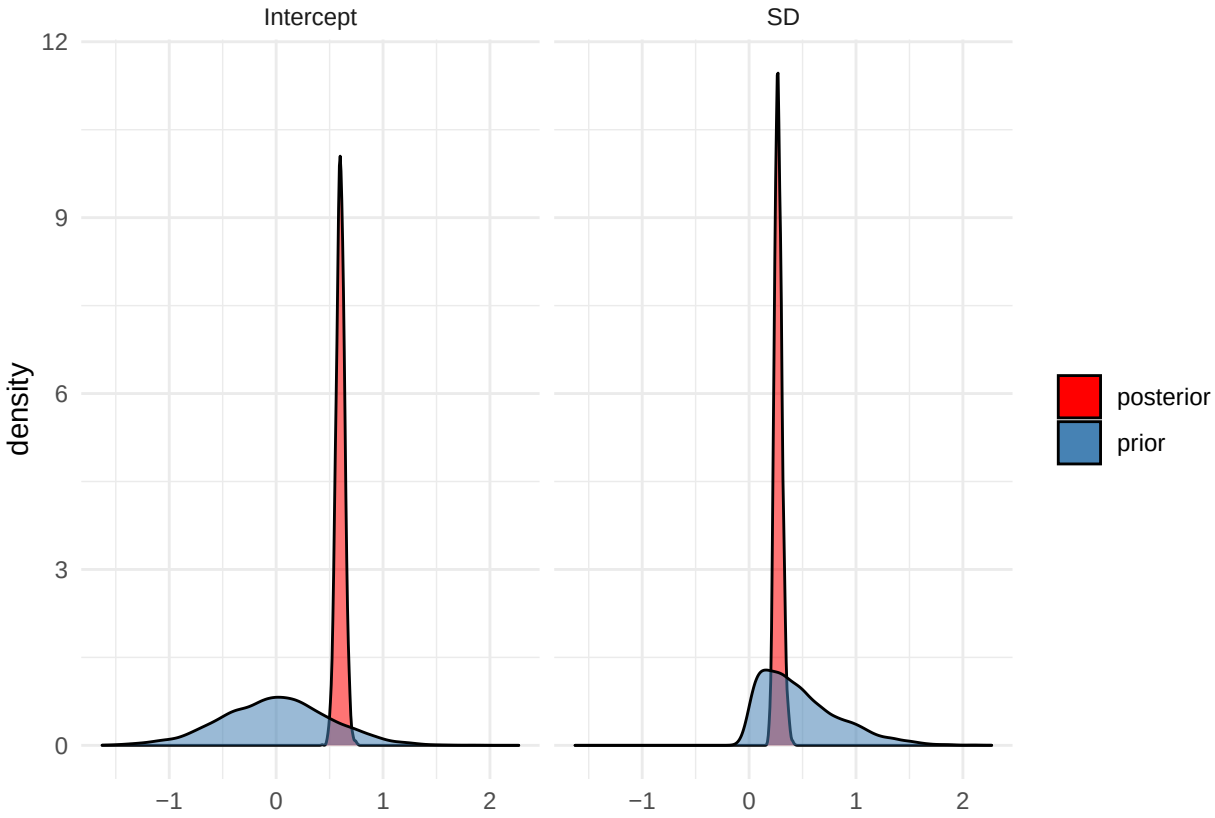
```
##  [67] "r_study[64,Intercept]"  "r_study[65,Intercept]"  "r_study[66,Intercept]"
##  [70] "r_study[67,Intercept]"  "r_study[68,Intercept]"  "r_study[69,Intercept]"
##  [73] "r_study[70,Intercept]"  "r_study[71,Intercept]"  "r_study[72,Intercept]"
##  [76] "r_study[73,Intercept]"  "r_study[74,Intercept]"  "r_study[75,Intercept]"
##  [79] "r_study[76,Intercept]"  "r_study[77,Intercept]"  "r_study[78,Intercept]"
##  [82] "r_study[79,Intercept]"  "r_study[80,Intercept]"  "r_study[81,Intercept]"
##  [85] "r_study[82,Intercept]"  "r_study[83,Intercept]"  "r_study[84,Intercept]"
##  [88] "r_study[85,Intercept]"  "r_study[86,Intercept]"  "r_study[87,Intercept]"
##  [91] "r_study[88,Intercept]"  "r_study[89,Intercept]"  "r_study[90,Intercept]"
##  [94] "r_study[91,Intercept]"  "r_study[92,Intercept]"  "r_study[93,Intercept]"
##  [97] "r_study[94,Intercept]"  "r_study[95,Intercept]"  "r_study[96,Intercept]"
## [100] "r_study[97,Intercept]"  "r_study[98,Intercept]"  "r_study[99,Intercept]"
## [103] "r_study[100,Intercept]" "prior_Intercept"        "prior_sd_study"
## [106] "lprior"                 "lp__"                   "accept_stat__"
## [109] "treedepth__"            "stepsize__"             "divergent__"
## [112] "n_leapfrog__"           "energy__"
```

```
pp_update_plot <- function(model){

bind_rows(
  gather_draws(model, c(b_Intercept, sd_study__Intercept)) %>%
    mutate(index = 'posterior') %>%
    mutate(.variable = if_else(.variable == 'b_Intercept', 'Intercept', 'SD')),
  gather_draws(model, `prior.*`, regex = T) %>%
    mutate(index = 'prior') %>%
    mutate(.variable = if_else(.variable == 'prior_Intercept', 'Intercept', 'SD'))
  ) %>%
    ggplot(aes(x = .value, fill = index, alpha = 0.3)) +
      geom_density() +
    facet_grid(~ .variable) +
    theme_minimal() +
    guides(alpha = 'none') +
    scale_fill_manual(name = element_blank(), values = c('red', 'steelblue')) +
    labs(x = element_blank())
}

map(models, pp_update_plot)
```

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




```
##
## [[2]]
```

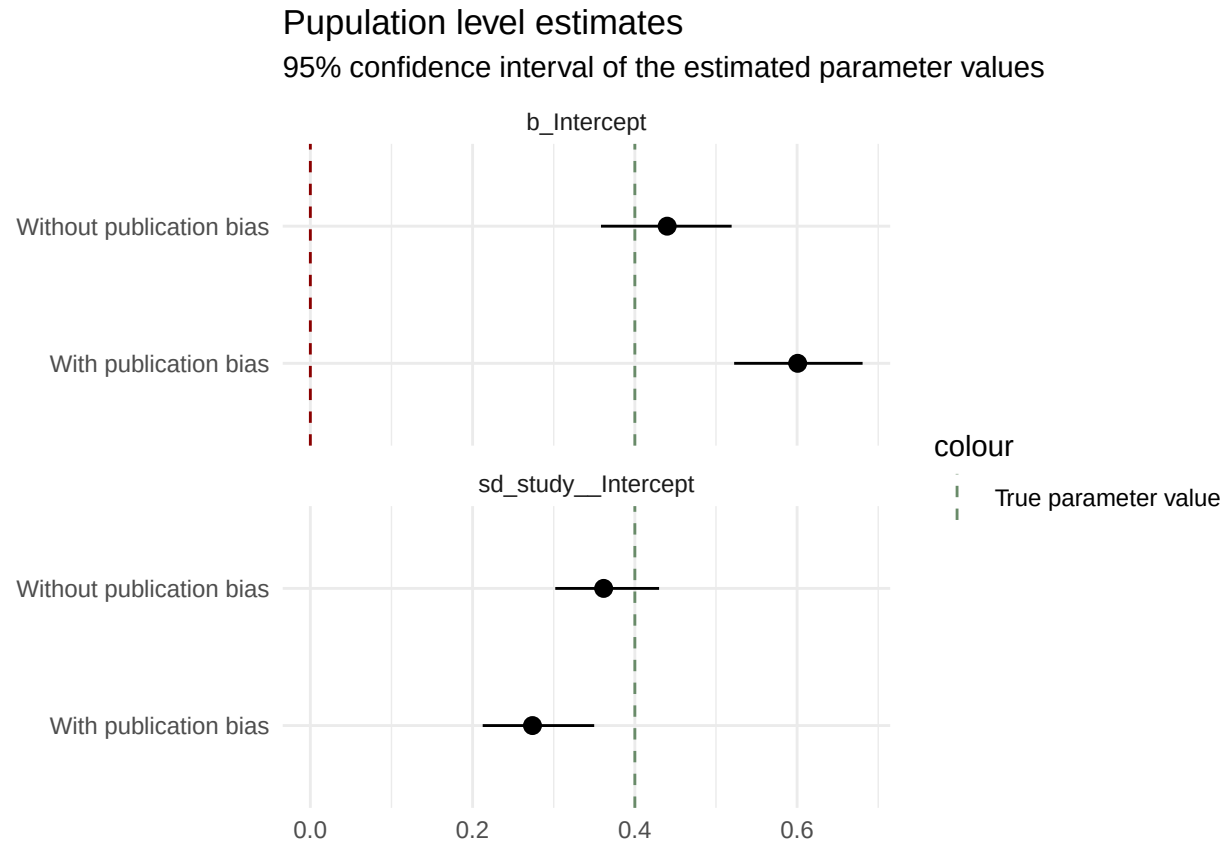


Comparing estimated and true values:

```
pop_effects_plot <- function(models, model_names, null = 0)
  # models must be a list (but not a vector) of models
  {

  plot_data <- tibble()

  plot_data <- map2_df(
    .x = models, .y = model_names,
    .f = function(.x, .y){
        bind_rows(plot_data,
                gather_draws(.x, c(b_Intercept, sd_study__Intercept)) %>%
                  mean_qi %>%
                  mutate(model = .y)
                )
    })

  plot_data %>%
    ggplot(aes(x = .value, y = model, xmin = .lower, xmax = .upper)) +
      geom_pointrange() +
      theme_minimal() +
      labs(x = NULL,
           y = NULL,
           title = "Pupulation level estimates",
           subtitle = "95% confidence interval of the estimated parameter values") +
      facet_wrap(vars(.variable), nrow = 2) +
```

```
      geom_vline(data = filter(plot_data, .variable == 'b_Intercept'),
                 aes(xintercept = null), linetype = 'dashed', color = 'darkred', size = 0.5)

}
```

```
pop_effects_plot(models = models, model_names = c('Without publication bias', 'With publication bias'))
  geom_vline(aes(xintercept = 0.4, colour = 'True parameter value'), linetype = 'dashed',) +
  scale_colour_manual(values = c(`True parameter value` = 'darkseagreen4'))
```

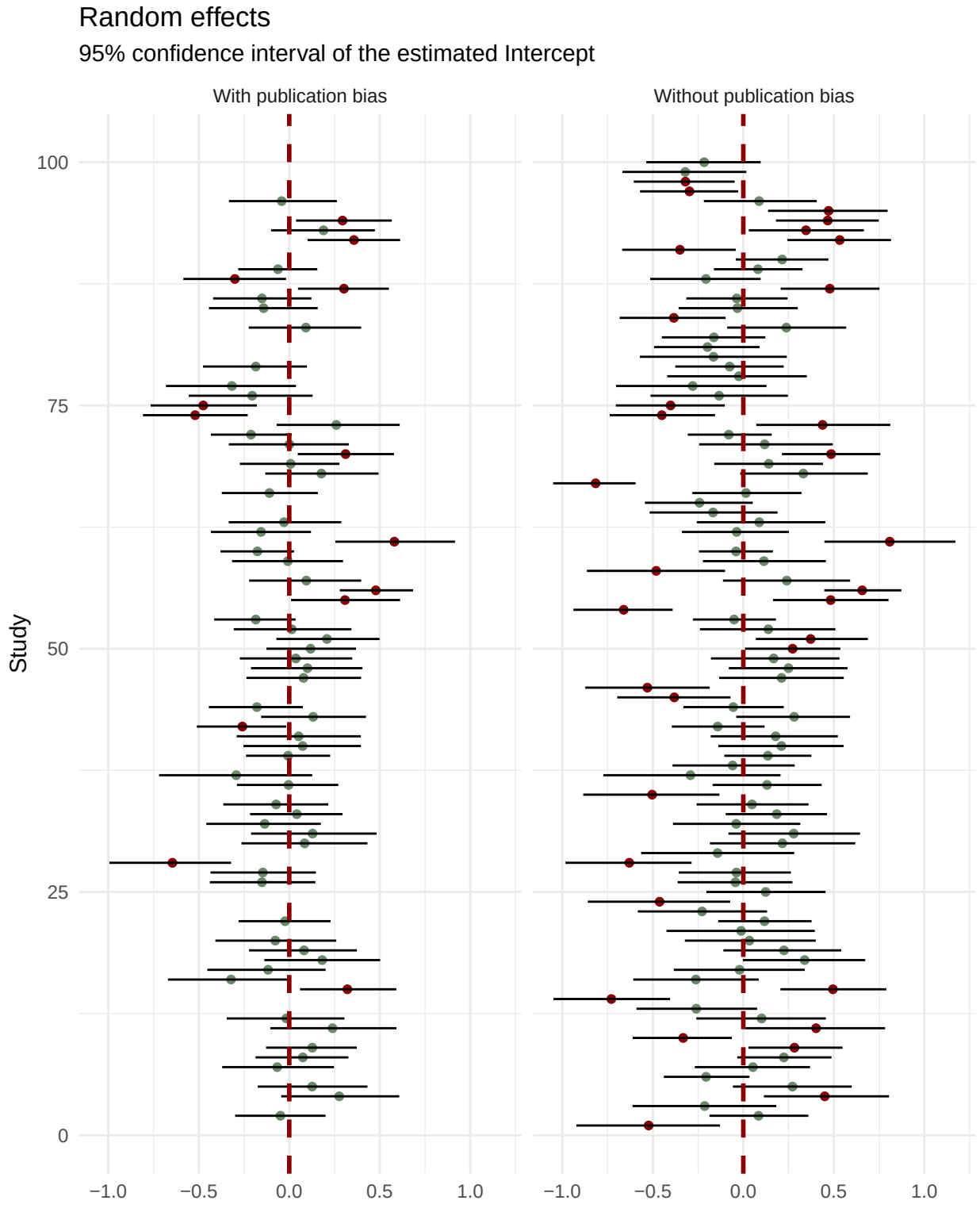



```
rand_effects_plot <- function(models, model_names, null = 0)
  # models must be a list (but not a vector) of models
  {

  plot_data <- tibble()

  plot_data <- map2_df(
    .x = models, .y = model_names,
    ~ bind_rows(plot_data,
                gather_draws(.x, r_study[study, parameter]) %>% mean_qi %>% mutate(model = .y)
                ) %>%
      mutate(colour = if_else(null < .upper & null > .lower, 'darkseagreen4', 'darkred'))
)

  plot_data %>%
    ggplot(aes(x = .value, y = study, xmin = .lower, xmax = .upper)) +
```

```
      geom_point(aes(colour = colour)) +
      geom_linerange() +
      theme_minimal() +
      labs(x = NULL,
           y = 'Study',
           title = "Random effects",
           subtitle = "95% confidence interval of the estimated Intercept") +
    geom_vline(xintercept = null, linetype = 'dashed', color = 'darkred', size = 1) +
    scale_colour_identity() +
    facet_wrap(vars(model))

}
```

```
rand_effects_plot(models = models, model_names = c('Without publication bias', 'With publication bias'))
```

```
## Warning: `gather_()` was deprecated in tidyr 1.2.0.
## i Please use `gather()` instead.
## i The deprecated feature was likely used in the tidybayes package.
##   Please report the issue at <https://github.com/mjskay/tidybayes/issues/new>.
```



Conclusions - the effect of publication bias: Introducing publication bias resulted in higher estimated mean of the intercept of the effect size, and lower estimated sd of the intercept of the effect size. However, the confidence intervals of the two models considerably overlap for both estimated parameters. In a NHST type approach the difference between the estimates would be considered non-significant. On the other hand, the estimates of the published studies only model do not include the true parameter values, while the model fitted on all of the studies does.

Since we only have access to published studies in the real data, we can expect the estimated mean of the intercept to be slightly higher and the standard deviation of the Intercept to be slighly lower then

```
save.image('a2_part1.Rdata')
# saving the object from part 1. In case i have to use them we won't need to rerun the whole thing
```

Question 2

```
#What is the current evidence for distinctive vocal patterns in schizophrenia?
#       - focusing on pitch variability (PITCH_FOSD).


# 1. Describe the data available (studies, participants).
# 2. Fit the models
# 3. visualize and report the findings:
    # 3.1 population level effect size;
    # 3.2 how well studies reflect it;
    # 3.3 influential studies,
    # 3.4 publication bias.
# BONUS question: assess the effect of task on the estimates (model comparison with baseline model)
```

```
rm(list = setdiff(ls(), lsf.str()))
# clearing the whole environment except the functions
```

```
data_raw <- read_excel('Matrix_MetaAnalysis_Diagnosis_updated290719.xlsx')
```

```
## New names:
## * `frequency` -> `frequency...68`
## * `frequency` -> `frequency...73`
## * `frequency` -> `frequency...78`
## * `frequency` -> `frequency...83`
## * `frequency` -> `frequency...88`
## * `frequency` -> `frequency...93`
## * `frequency` -> `frequency...98`
## * `variability` -> `variability...108`
## * `variability` -> `variability...113`
## * `variability` -> `variability...118`
## * `variability` -> `variability...123`
## * `variability` -> `variability...128`
```

```
glimpse(data_raw)
```

```
## Rows: 57
## Columns: 147
## $ ArticleID                <dbl> 1, 2, 3, 3, 4, 5, 6, 7, 8, 8, 9, 9, 10, 11, ~
## $ StudyID                  <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 9, 10, 10, 11, 12~
## $ Specification            <chr> NA, NA, "_1_it", "_2_us", NA, NA, NA, NA, "p~
## $ Title                    <chr> "Emotional self-other voice processing in sc~
## $ Authors                  <chr> "Pinheiro AP, Rezaii N, Rauber A, Nestor PG,~
## $ Year_publication         <dbl> 2016, 2016, 2016, 2016, 2015, 2014, 2002, 20~
```

```
## $ Article                    <chr> "Pinheiro et al. (2016)", "Zhang et al. (201~
## $ DIAGNOSIS                  <chr> "SZ", "SZ", "SZ", "SZ", "SZ", "ST", "SZ", "S~
## $ MALE_SZ                    <chr> "12", "16", "13", "13", "32", "4", "30", "27~
## $ FEMALE_SZ                  <chr> "5", "10", "7", "7", "13", "4", "0", "12", "~
## $ MALE_HC                    <chr> "13", "16", NA, NA, "22", "18", NA, "8", "NR~
## $ FEMALE_HC                  <chr> "5", "14", NA, NA, "13", "18", NA, "10", "NR~
## $ AGE_M_SZ                   <chr> "48.29", "43.3", "35.4", "33.6", "39.49", "2~
## $ AGE_SD_SZ                  <chr> "8.49", "10.9", "11.2", "10.1", "10.89", "1.~
## $ AGE_M_HC                   <chr> "49.53", "37", NA, NA, "35.340000000000003",~
## $ AGE_SD_HC                  <chr> "4.4800000000000004", "14.3", NA, NA, "10.48~
## $ EDUCATION_M_SZ             <chr> "13.69", "9.5", "12.6", "12.2", "10.4", "NR"~
## $ EDUCATION_SD_SZ            <chr> "2.33", "3", "3.8", "2.4", "2,63197\r\n \r\n~
## $ EDUCATION_M_HC             <chr> "15.06", "11.6", NA, NA, "11.6571", "NR", NA~
## $ EDUCATION_SD_HC            <chr> "1.91", "2.5", NA, NA, "3.3426200000000001",~
## $ SAMPLE_SIZE_SZ             <dbl> 17, 26, 20, 20, 45, 8, 30, 39, 28, 28, 13, 1~
## $ SAMPLE_SIZE_HC             <dbl> 18, 30, NA, NA, 35, 36, NA, 18, 27, 27, 6, 6~
## $ TASK_TOTAL_DURATION_HC_M   <chr> "NR", NA, NA, NA, "83.85", NA, NA, "194.58",~
## $ TASK_TOTAL_DURATION_HC_SD  <chr> "NR", NA, NA, NA, "15.65", NA, NA, "30.64", ~
## $ TASK_TOTAL_DURATION_SZ_M   <chr> "NR", NA, "240", NA, "124.12", "100", NA, "2~
## $ TASK_TOTAL_DURATION_SZ_SD  <chr> "NR", NA, NA, NA, "43.15", NA, NA, "42.8", N~
## $ TYPE_OF_TASK               <chr> "CONSTR", "SOCIAL", "FREE", "FREE", "CONSTR"~
## $ MEAN_DURATION              <chr> "Mean words duration (ms)", NA, "2 min + 2 m~
## $ SP_DUR_HC_M                <dbl> 625.4967, NA, NA, NA, NA, 1170.0740, NA, 302~
## $ SP_DUR_HC_SD               <dbl> 62.11333, NA, NA, NA, NA, 213.21500, NA, 480~
## $ SP_DUR_SZ_M                <dbl> 666.4167, NA, NA, NA, NA, 1126.6600, NA, 269~
## $ SP_DUR_SZ_SD               <dbl> 64.46667, NA, NA, NA, NA, 136.84000, NA, 630~
## $ SPEECH_RATE                <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ SP_RAT_HC_M                <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ SP_RAT_HC_SD               <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ SP_RAT_SZ_M                <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ SP_RAT_SZ_SD               <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ SPEECH_PERCENTAGE          <chr> NA, NA, NA, NA, "Non-Pause rate % > 300 ms",~
## $ SP_PER_HC_M                <dbl> NA, NA, NA, NA, 71.42000, 29.90000, NA, 84.2~
## $ SP_PER_HC_SD               <dbl> NA, NA, NA, NA, 5.98000, 9.60000, NA, 13.084~
## $ SP_PER_SZ_M                <dbl> NA, NA, 56.00000, 42.00000, 62.36000, 25.200~
## $ SP_PER_SZ_SD               <dbl> NA, NA, 12.000000, 17.000000, 10.690000, 10.~
## $ PAUSE_DURATION             <chr> NA, NA, NA, NA, NA, NA, NA, "Mean pause dura~
## $ PA_DUR_HC_M                <dbl> NA, NA, NA, NA, NA, NA, NA, 570.00, NA, NA, ~
## $ PA_DUR__HC_SD              <dbl> NA, NA, NA, NA, NA, NA, NA, 80.0000, NA, NA,~
## $ PA_DUR_SZ_M                <dbl> NA, NA, NA, NA, NA, NA, NA, 630.00, NA, NA, ~
## $ PA_DUR_SZ_SD               <dbl> NA, NA, NA, NA, NA, NA, NA, 80.000, NA, NA, ~
## $ Number_of_pauses           <chr> NA, NA, NA, NA, NA, "> 10 msec", NA, "Number~
## $ PA_NUM_HC_M                <dbl> NA, NA, NA, NA, NA, 177.7770, NA, 54.0500, N~
## $ PA_NUM_HC_SD               <dbl> NA, NA, NA, NA, NA, 40.383000, NA, 8.940000,~
## $ PA_NUM_SZ_M                <dbl> NA, NA, NA, NA, NA, 178.8750, NA, 68.7200, N~
## $ PA_NUM_SZ_SD               <dbl> NA, NA, NA, NA, NA, 34.411000, NA, 19.770000~
## $ `Total length of pauses`   <chr> NA, NA, NA, NA, NA, NA, NA, "ms", NA, NA, "P~
## $ PA_TLE_HC_M                <dbl> NA, NA, NA, NA, NA, NA, NA, 30560.00, NA, NA~
## $ PA_TLE_HC_SD               <dbl> NA, NA, NA, NA, NA, NA, NA, 6100.00, NA, NA,~
## $ PA_TLE_SZ_M                <dbl> NA, NA, NA, NA, NA, NA, NA, 42800.00, NA, NA~
## $ PA_TLE_SZ_SD               <dbl> NA, NA, NA, NA, NA, NA, NA, 12420.00, NA, NA~
## $ `Response latency`         <chr> NA, NA, NA, NA, NA, NA, "sec", NA, NA, NA, N~
## $ PA_RL_HC_M                 <dbl> NA, NA, NA, NA, NA, NA, NA, NA, 87.75, NA, N~
## $ PA_RL_HC_SD                <dbl> NA, NA, NA, NA, NA, NA, NA, NA, 28.3, NA, NA~
```

```
## $ PA_RL_SZ_M               <dbl> NA, NA, NA, NA, NA, NA, 1100.00, NA, 95.75, ~
## $ PA_RL_SZ_SD              <dbl> NA, NA, NA, NA, NA, NA, 560.0, NA, 31.7, NA,~
## $ `Pause SD`               <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, "Pau~
## $ PA_SD_HC_M               <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, 130,~
## $ PA_SD_HC_SD              <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, 50, ~
## $ PA_SD_SZ_M               <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, 250,~
## $ PA_SD_SZ_SD              <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, 170,~
## $ frequency...68           <chr> "hz", NA, "hz", "hz", "hz", NA, NA, NA, NA, ~
## $ PITCH_F0_HC_M            <dbl> 137.2467, NA, NA, NA, 156.5700, NA, NA, NA, ~
## $ PITCH_F0_HC_SD           <dbl> 29.98333, NA, NA, NA, 43.89000, NA, NA, NA, ~
## $ PITCH_F0_SZ_M            <dbl> 148.0433, NA, 136.4409, 118.9344, 143.0200, ~
## $ PITCH_F0_SZ_SD           <dbl> 23.89667, NA, 31.93596, 30.24209, 40.33000, ~
## $ frequency...73           <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ PITCH_F1_HC_M            <dbl> NA, 0.04, NA, NA, NA, NA, NA, NA, NA, NA, NA~
## $ PITCH_F1_HC_SD           <dbl> NA, 0.005, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ PITCH_F1_SZ_M            <dbl> NA, 0.1365, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ PITCH_F1_SZ_SD           <dbl> NA, 0.0175, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ frequency...78           <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ PITCH_F2_HC_M            <dbl> NA, 0.083, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ PITCH_F2_HC_SD           <dbl> NA, 0.009, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ PITCH_F2_SZ_M            <dbl> NA, 0.062, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ PITCH_F2_SZ_SD           <dbl> NA, 0.0305, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ frequency...83           <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ PITCH_F3_HC_M            <dbl> NA, 0.182, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ PITCH_F3_HC_SD           <dbl> NA, 0.012, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ PITCH_F3_SZ_M            <dbl> NA, 0.1765, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ PITCH_F3_SZ_SD           <dbl> NA, 0.0145, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ frequency...88           <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ PITCH_F4_HC_M            <dbl> NA, 0.257, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ PITCH_F4_HC_SD           <dbl> NA, 0.019, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ PITCH_F4_SZ_M            <dbl> NA, 0.2645, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ PITCH_F4_SZ_SD           <dbl> NA, 0.021, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ frequency...93           <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ PITCH_F5_HC_M            <dbl> NA, 0.357, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ PITCH_F5_HC_SD           <dbl> NA, 0.013, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ PITCH_F5_SZ_M            <dbl> NA, 0.359, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ PITCH_F5_SZ_SD           <dbl> NA, 0.017, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ frequency...98           <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ PITCH_F6_HC_M            <dbl> NA, 0.426, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ PITCH_F6_HC_SD           <dbl> NA, 0.015, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ PITCH_F6_SZ_M            <dbl> NA, 0.426, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ PITCH_F6_SZ_SD           <dbl> NA, 0.0115, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ pitch_f0_variability     <chr> NA, NA, "Log10  Hz", "Log10  Hz", "hz", "not~
## $ PITCH_F0SD_HC_M          <dbl> NA, NA, NA, NA, 29.670000, 0.034800, NA, 0.2~
## $ PITCH_F0SD_HC_SD         <dbl> NA, NA, NA, NA, 8.32000000, 0.01600000, NA, ~
## $ PITCH_F0SD_SZ_M          <dbl> NA, NA, 0.0875000, 0.0834000, 27.5000000, 0.~
## $ PITCH_F0SD_SZ_SD         <dbl> NA, NA, 0.0394000, 0.0242000, 10.3600000, 0.~
## $ variability...108        <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ PITCH_F0_ENTROPY_HC_M    <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ PITCH_F0_ENTROPY__HC_SD  <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ PITCH_F0_ENTROPY_SZ_M    <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ PITCH_F0_ENTROPY__SZ_SD  <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ variability...113        <chr> NA, NA, "Log10  Hz", "Log10  Hz", NA, NA, NA~
## $ PITCH_F1SD_HC_M          <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
```

```
## $ PITCH_F1SD_HC_SD          <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ PITCH_F1SD_SZ_M           <dbl> NA, NA, 134.3, 149.4, NA, NA, NA, NA, NA, NA~
## $ PITCH_F1SD_SZ_SD          <dbl> NA, NA, 30.4, 62.7, NA, NA, NA, NA, NA, NA, ~
## $ variability...118         <chr> NA, NA, "Log10  Hz", "Log10  Hz", NA, NA, NA~
## $ PITCH_F2SD_HC_M           <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ PITCH_F2SD_HC_SD          <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ PITCH_F2SD_SZ_M           <dbl> NA, NA, 411.8, 397.7, NA, NA, NA, NA, NA, NA~
## $ PITCH_F2SD_SZ_SD          <dbl> NA, NA, 66.9, 53.5, NA, NA, NA, NA, NA, NA, ~
## $ variability...123         <chr> NA, "Hz", NA, NA, NA, NA, NA, NA, NA, NA, NA~
## $ PITCH_BDWTH_HC_M          <dbl> NA, 18.1, NA, NA, NA, NA, NA, NA, NA, NA, NA~
## $ PITCH_BDWTH_HC_SD         <dbl> NA, 9.81, NA, NA, NA, NA, NA, NA, NA, NA, NA~
## $ PITCH_BDWTH_SZ_M          <dbl> NA, 20.045, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ PITCH_BDWTH_SZ_SD         <dbl> NA, 11.73, NA, NA, NA, NA, NA, NA, NA, NA, N~
## $ variability...128         <chr> NA, NA, NA, NA, "Semitones/s", NA, NA, NA, N~
## $ PITCH_FORAN_HC_M          <dbl> NA, NA, NA, NA, 11, NA, NA, NA, NA, NA, NA, ~
## $ PITCH_FORAN_HC_SD         <dbl> NA, NA, NA, NA, 2.9, NA, NA, NA, NA, NA, NA,~
## $ PITCH_FORAN_SZ_M          <dbl> NA, NA, NA, NA, 11.27, NA, NA, NA, NA, NA, N~
## $ PITCH_FORAN_SZ_SD         <dbl> NA, NA, NA, NA, 3.84, NA, NA, NA, NA, NA, NA~
## $ intensity_mean            <chr> "db", NA, NA, NA, "db", NA, NA, NA, NA, NA, ~
## $ INT_MEAN_HC_M             <dbl> 73.95667, NA, NA, NA, 73.79000, NA, NA, NA, ~
## $ INT_MEAN_HC_SD            <dbl> 8.2966667, NA, NA, NA, 1.4000000, NA, NA, NA~
## $ INT_MEAN_SZ_M             <dbl> 74.52667, NA, NA, NA, 71.08000, NA, NA, NA, ~
## $ INT_MEAN_SZ_SD            <dbl> 2.3866667, NA, NA, NA, 1.5000000, NA, NA, NA~
## $ intensity_variability     <chr> NA, NA, NA, NA, NA, "not specified, db?", "a~
## $ INT_VAR_HC_M              <dbl> NA, NA, NA, NA, NA, 8.1420, NA, NA, NA, NA, ~
## $ INT_VAR_HC_SD             <dbl> NA, NA, NA, NA, NA, 1.201000, NA, NA, NA, NA~
## $ INT_VAR_SZ_M              <dbl> NA, NA, NA, NA, NA, 7.975000, 18.900000, NA,~
## $ INT_VAR_SZ_SD             <dbl> NA, NA, NA, NA, NA, 1.1950000, 5.3400000, NA~
## $ variability_entropy       <chr> NA, "db_formant_amplitude (MEAN between (t1_~
## $ INT_VAR_ENTR_HC_M         <dbl> NA, 0.0410, NA, NA, NA, NA, NA, NA, NA, NA, ~
## $ INT_VAR_ENTR_HC_SD        <dbl> NA, 0.0090000, NA, NA, NA, NA, NA, NA, NA, N~
## $ INT_VAR_ENTR_SZ_M         <dbl> NA, 0.04250, NA, NA, NA, NA, NA, NA, NA, NA,~
## $ INT_VAR_ENTR_SZ_SD        <dbl> NA, 0.0045000, NA, NA, NA, NA, NA, NA, NA, N~
```

```
head(data_raw)
```

```
## # A tibble: 6 x 147
##   Articl~1 StudyID Speci~2 Title Authors Year_~3 Article DIAGN~4 MALE_SZ FEMAL~5
##      <dbl>   <dbl> <chr>   <chr> <chr>     <dbl> <chr>   <chr>   <chr>   <chr>
## 1        1       1 <NA>    "Emo~ Pinhei~    2016 Pinhei~ SZ      12      5
## 2        2       2 <NA>    "Cli~ Zhang ~    2016 Zhang ~ SZ      16      10
## 3        3       3 _1_it   "Ass~ Bernar~    2016 Bernar~ SZ      13      7
## 4        3       4 _2_us   "Ass~ Bernar~    2016 Bernar~ SZ      13      7
## 5        4       5 <NA>    "Can~ Martín~    2015 Martín~ SZ      32      13
## 6        5       6 <NA>    "Spe~ Bedwel~    2014 Bedwel~ ST      4       4
## # ... with 137 more variables: MALE_HC <chr>, FEMALE_HC <chr>, AGE_M_SZ <chr>,
## #   AGE_SD_SZ <chr>, AGE_M_HC <chr>, AGE_SD_HC <chr>, EDUCATION_M_SZ <chr>,
## #   EDUCATION_SD_SZ <chr>, EDUCATION_M_HC <chr>, EDUCATION_SD_HC <chr>,
## #   SAMPLE_SIZE_SZ <dbl>, SAMPLE_SIZE_HC <dbl>, TASK_TOTAL_DURATION_HC_M <chr>,
## #   TASK_TOTAL_DURATION_HC_SD <chr>, TASK_TOTAL_DURATION_SZ_M <chr>,
## #   TASK_TOTAL_DURATION_SZ_SD <chr>, TYPE_OF_TASK <chr>, MEAN_DURATION <chr>,
## #   SP_DUR_HC_M <dbl>, SP_DUR_HC_SD <dbl>, SP_DUR_SZ_M <dbl>, ...
```

```r
cohens_d <- function(x1, x2, sd1, sd2, n1, n2){

  mean_diff <- x1 - x2
  pooled_sd <- sqrt((sd1^2 + sd2^2) / 2)

  return(mean_diff / pooled_sd)
}



cohens_d_se <- function(d, sd1, sd2, n1, n2){

  a1 <- (n1 + n2) / (n1*n2)
  a2 <- d^2 / (2*(n1 + n2 - 2))
  b <- (n1 + n2) / (n1 + n2 - 2)

return((a1 + a2)*b)
}
# I know i'll call these function just once, but i thought it will make the code easier to read



data <- data_raw %>%
  select(1:2 | TYPE_OF_TASK | 9:22 | starts_with('PITCH_F0SD')) %>%
  filter(!(is.na(PITCH_F0SD_HC_M) | is.na(PITCH_F0SD_SZ_M))) %>%
  rename_with(~ str_to_lower(.x) %>%
                str_replace_all(c('_sz_sd' = '_sd_sz', '_sz_m' = '_m_sz',
                                  '_hc_sd' = '_sd_hc', '_hc_m' = '_m_hc')) %>%
                str_replace(fixed('_hc'), '__hc') %>%
                str_replace(fixed('_sz'), '__sz')
              ) %>%
  #this makes pivot_longer() (later in the pipeline) easier
  add_count(studyid) %>%
  # useful when dealing with repeated studyids (later in the pipeline)
  mutate(across(everything(), ~ str_to_lower(.x) %>% na_if('nr')),
         studyid = as.character(studyid),
         studyid = case_when(
                     n == 1                              ~ studyid,
                     n == 2 & lag(studyid) == studyid ~ paste0(studyid, 'b'),
                     TRUE                                ~ paste0(studyid, 'a')
                     ),
         # dealing with repeated studyids
         n = NULL,
         #deleting the now useless column created by add_count(studyid)
         across(1:3, as_factor),
         across(!1:3,
                ~ str_replace_all(.x, ',', '.') %>%
                  str_remove_all("[^0-9.]") %>%
                  as.numeric),
          #there were weird cells like '2,63197\r\n \r\n    ' that needed to be fixed before converting
        effect = cohens_d(x1 = pitch_f0sd_m__hc, x2 = pitch_f0sd_m__sz,
                          sd1 = pitch_f0sd_sd__hc, sd2 = pitch_f0sd_sd__sz,
                          n1 = sample_size__hc, n2 = sample_size__sz),
```

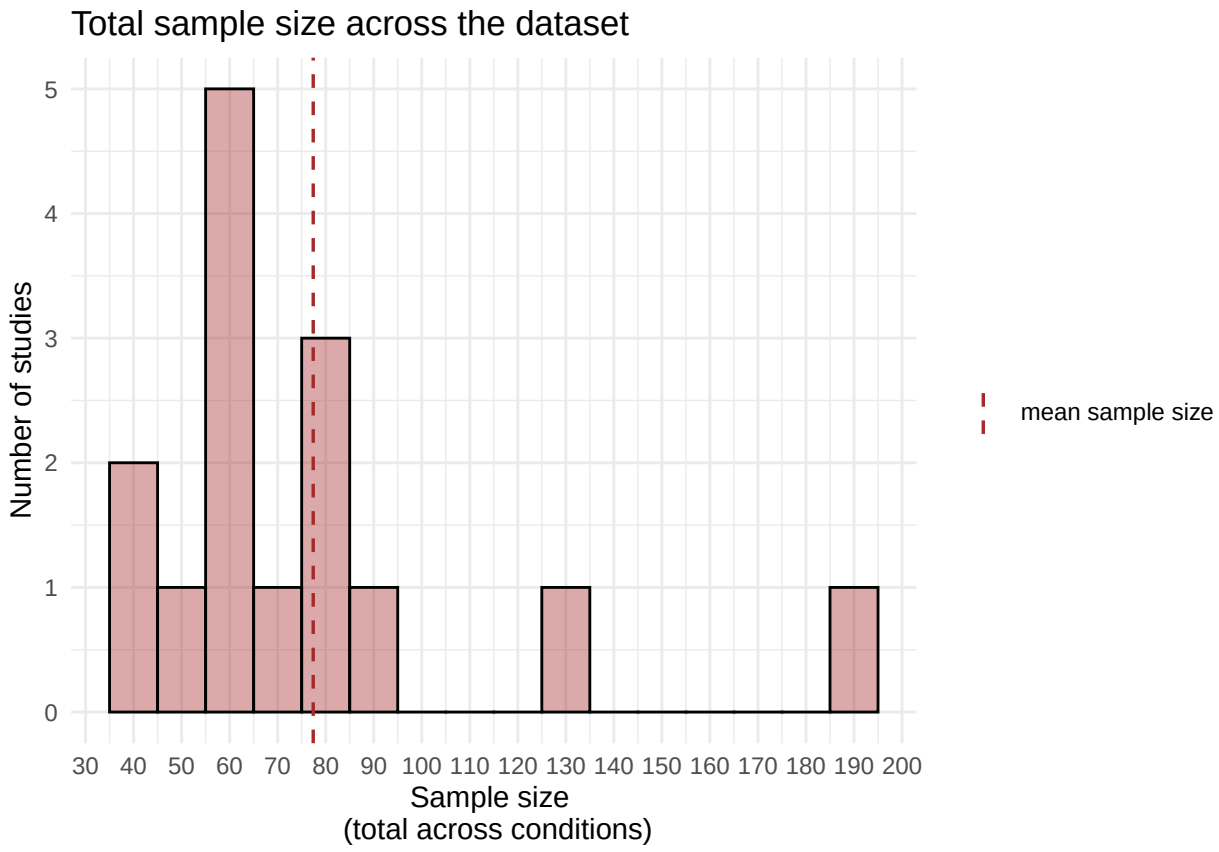
```
        effect_sigma = cohens_d_se(d = effect,
                                   sd1 = pitch_f0sd_sd__hc, sd2 = pitch_f0sd_sd__sz,
                                   n1 = sample_size__hc, n2 = sample_size__sz),
        .after = type_of_task
        ) %>%
  pivot_longer(cols = !1:5,
               names_to = c('.value', 'diagnosis'),
               names_sep= '__') %>%
  mutate(diagnosis = diagnosis %>% as_factor) %>%
  rename('n_diagnosis' = 'sample_size') %>%
  group_by(studyid) %>%
  mutate(sample_size = sum(n_diagnosis, na.rm = T), .after = n_diagnosis) %>%
  ungroup

head(data)
```

```
## # A tibble: 6 x 16
##   articleid studyid type_of_t~1 effect effec~2 diagn~3  male female age_m age_sd
##   <fct>     <fct>   <fct>        <dbl>   <dbl> <fct>   <dbl>  <dbl> <dbl>  <dbl>
## 1 4         5       constr       0.231  0.0524 sz         32     13  39.5  10.9
## 2 4         5       constr       0.231  0.0524 hc         22     13  35.3  10.5
## 3 5         6       free        -0.219  0.161  sz          4      4  20     1.2
## 4 5         6       free        -0.219  0.161  hc         18     18  20     4.94
## 5 7         8       constr       0.300  0.0850 sz         27     12  42.3  13.5
## 6 7         8       constr       0.300  0.0850 hc          8     10  40.5  12.9
## # ... with 6 more variables: education_m <dbl>, education_sd <dbl>,
## #   n_diagnosis <dbl>, sample_size <dbl>, pitch_f0sd_m <dbl>,
## #   pitch_f0sd_sd <dbl>, and abbreviated variable names 1: type_of_task,
## #   2: effect_sigma, 3: diagnosis
```

```
rm(cohens_d, cohens_d_se)
```

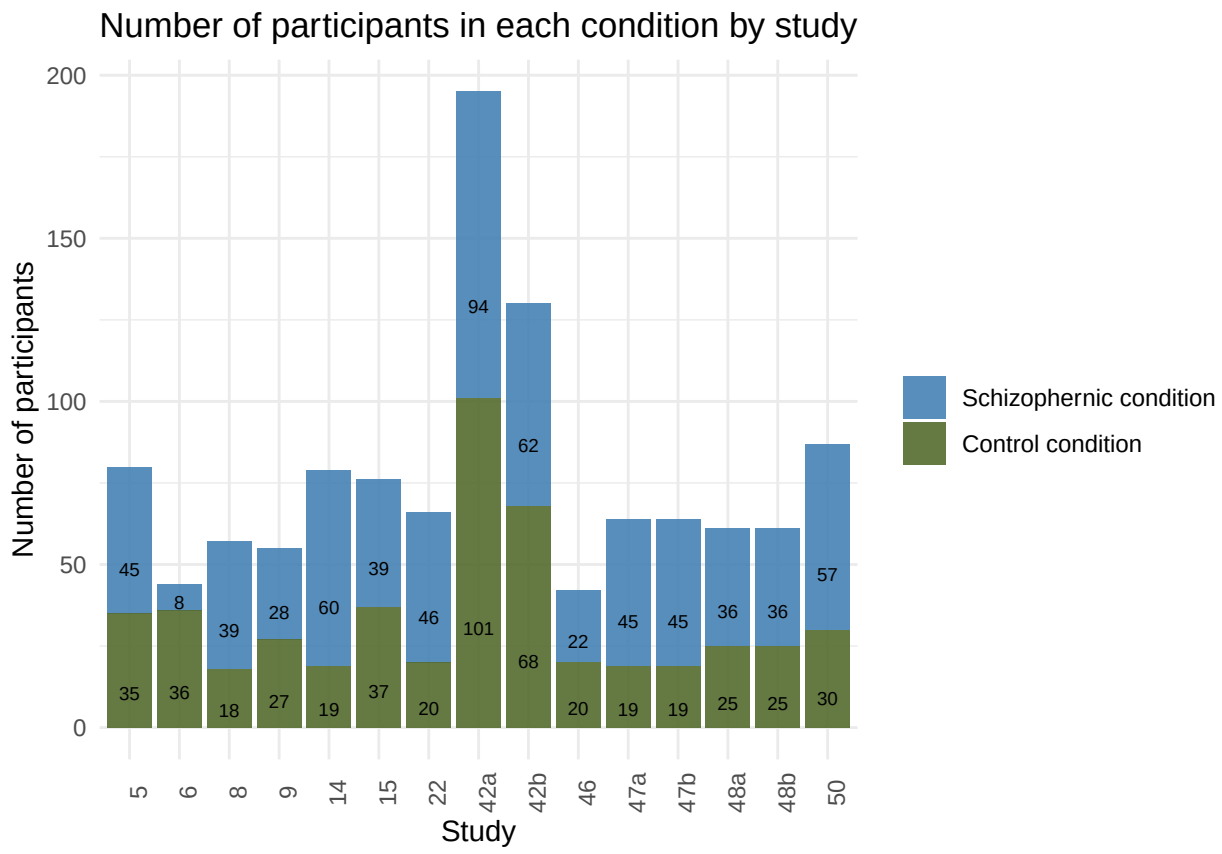
Describing the data

```
# Mean total sample size
data %>% filter(diagnosis == 'sz') %>%
  ggplot(aes(x = sample_size)) +
    geom_histogram(fill = 'brown', color = 'black', alpha = 0.4, binwidth = 10) +
    geom_vline(aes(xintercept = mean(sample_size, na.rm = T), color = 'mean sample size'),
                linetype = 'dashed', size = 0.6) +
    theme_minimal() +
    labs(title = "Total sample size across the dataset",
         x = "Sample size \n (total across conditions)",
         y = "Number of studies") +
    scale_x_continuous(n.breaks = 14) +
    scale_color_manual(name = element_blank(), values = c(`mean sample size` = 'brown'))
```



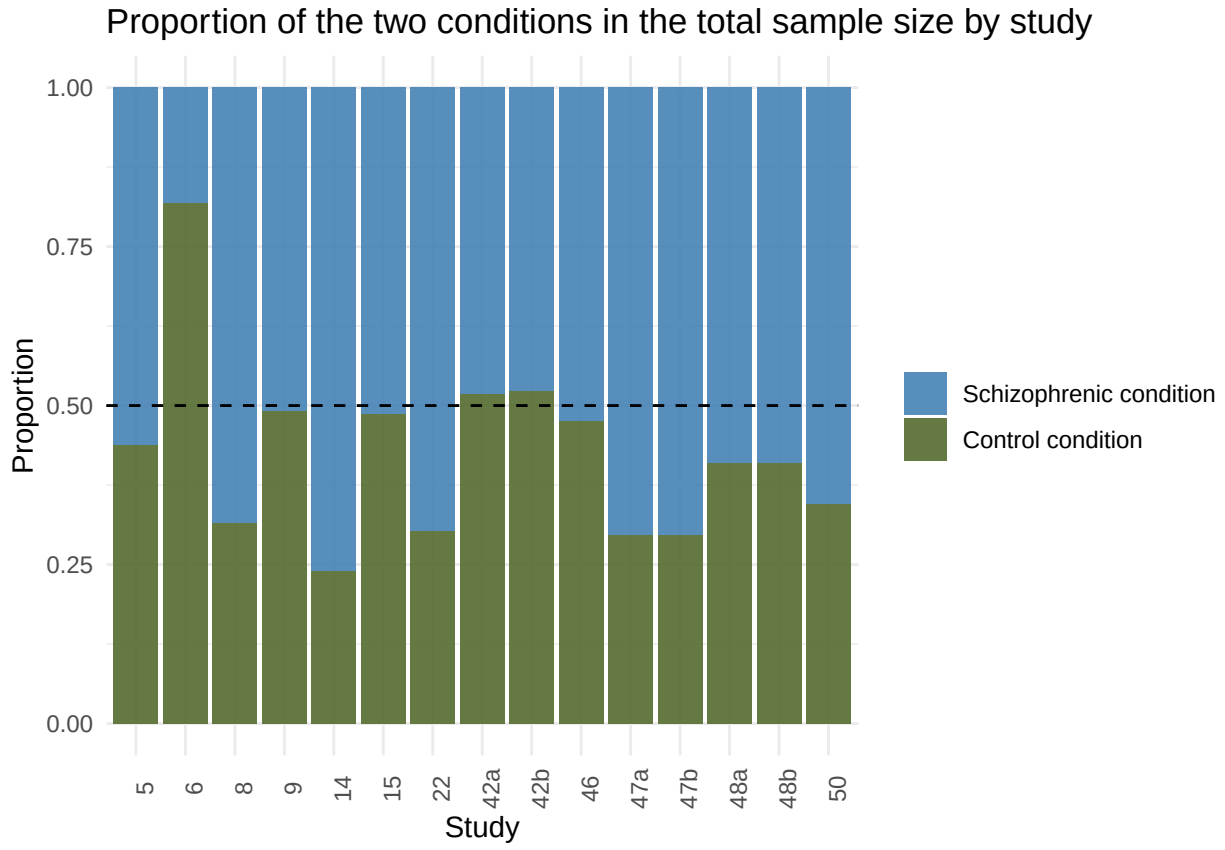
Sample size

```
#where the conditions balanced in terms of size?
data %>%
  ggplot(aes(x = studyid, y = n_diagnosis, fill = diagnosis)) +
    geom_bar(stat = 'identity', alpha = 0.9) +
    geom_text(aes(label = n_diagnosis),
              size = 2.5,
              position = position_stack(vjust = 0.3)) +
    theme_minimal() +
      labs(title = "Number of participants in each condition by study",
           x = "Study",
           y = "Number of participants") +
      scale_fill_manual(name = element_blank(),
                        labels = c('Schizophernic condition', 'Control condition'),
                        values = c('steelblue', 'darkolivegreen')) +
      theme(axis.text.x = element_text(angle = 90))
```




```
data %>%
  group_by(studyid, diagnosis) %>%
  summarise(n_diagnosis = n_diagnosis, sample_size = sample_size, pct = n_diagnosis / sample_size) %>%
  ggplot(aes(x = studyid, fill = diagnosis)) +
    geom_bar(aes(y = pct), stat = 'identity', alpha = 0.9) +
    geom_hline(aes(yintercept = 0.5), linetype = 'dashed', size = 0.5) +
    theme_minimal() +
    labs(title = "Proportion of the two conditions in the total sample size by study",
         x = "Study",
         y = "Proportion") +
    scale_fill_manual(name = element_blank(),
                      labels = c('Schizophrenic condition', 'Control condition'),
                      values = c('steelblue', 'darkolivegreen')) +
    theme(axis.text.x = element_text(angle = 90))
```

```
## `summarise()` has grouped output by 'studyid'. You can override using the
## `.groups` argument.
```




```
#sum of the samples of each condition
data %>%
  group_by(diagnosis) %>%
  summarise(mean = mean(n_diagnosis) %>% round(2),
            n = sum(n_diagnosis, na.rm = T)) %>%
  mutate(pct = (n / sum(n)) %>% round(2))
```

```
## # A tibble: 2 x 4
##   diagnosis  mean     n   pct
##   <fct>     <dbl> <dbl> <dbl>
## 1 sz         44.1   662  0.57
## 2 hc         33.3   499  0.43
```

```
data %>% filter(diagnosis == 'sz') %>% summarise(mean = mean(sample_size))
```

```
## # A tibble: 1 x 1
##    mean
##   <dbl>
## 1  77.4
```

After filtering for studies that measured the pitch variability(PITCH_F0SD) for both healthy and schizophrenic conditions the number of studies got reduced to 15. Among the studies the average number of participants in total (both conditions summed) was 77.4, the mean sample size for each condition was 44.13 and 33.27 for schizophrenic and healthy conditions respectively.

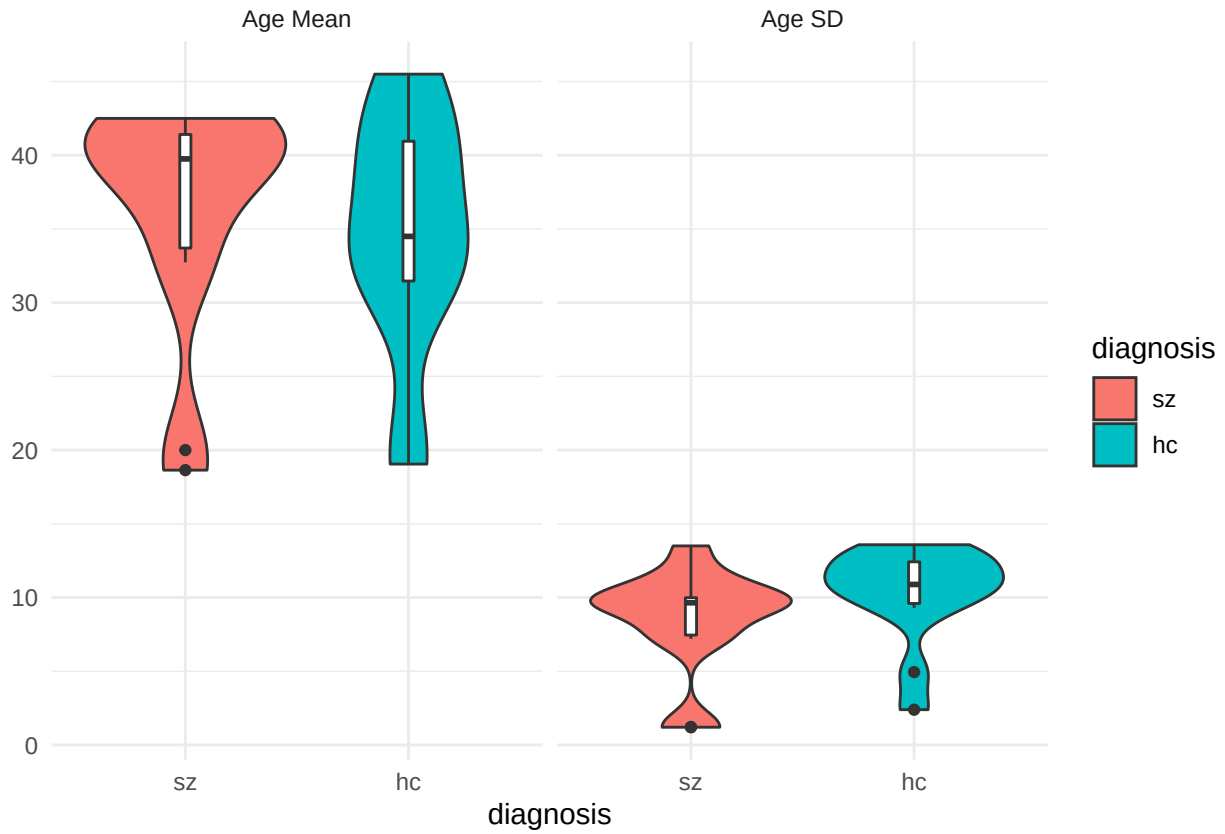
Overall, the sizes of the two conditions were roughly balanced. However, some of the studies had considerably different sample sizes (e.g. study 6, 14, 47a, 47b), which might be considered problematic. The total number of participants in each condition across all of the studies was 662 (57% of all participants) and 499 (43% of all participants) for schizophrenic and healthy conditions respectively.

```
age_data <- data %>%
  select(studyid, diagnosis, age_m, age_sd) %>%
  pivot_longer(c(age_m, age_sd),
               names_to = c(".value","age_parameter"),
               names_sep = "_") %>%
              mutate(age_parameter = ifelse(age_parameter =="m", "Age Mean", "Age SD"))

age_data %>%
  ggplot(aes(x = diagnosis, y = age, fill = diagnosis))+
    geom_violin()+
    geom_boxplot(width = 0.05, fill = "white") +
    facet_wrap(~age_parameter) +
    labs(y = NULL) +
    theme_minimal()
```

Age

```
## Warning: Removed 4 rows containing non-finite values (stat_ydensity).
```

```
## Warning: Removed 4 rows containing non-finite values (stat_boxplot).
```




```
age_data %>%
  group_by(diagnosis, age_parameter) %>%
  summarise(mean = mean(age, na.rm = T) %>% round(2))
```

```
## `summarise()` has grouped output by 'diagnosis'. You can override using the
## `.groups` argument.
```

```
## # A tibble: 4 x 3
## # Groups:   diagnosis [2]
##   diagnosis age_parameter  mean
##   <fct>     <chr>         <dbl>
## 1 sz        Age Mean      36.4
## 2 sz        Age SD         8.53
## 3 hc        Age Mean      34.9
## 4 hc        Age SD        10.4
```

```
rm(age_data)
```

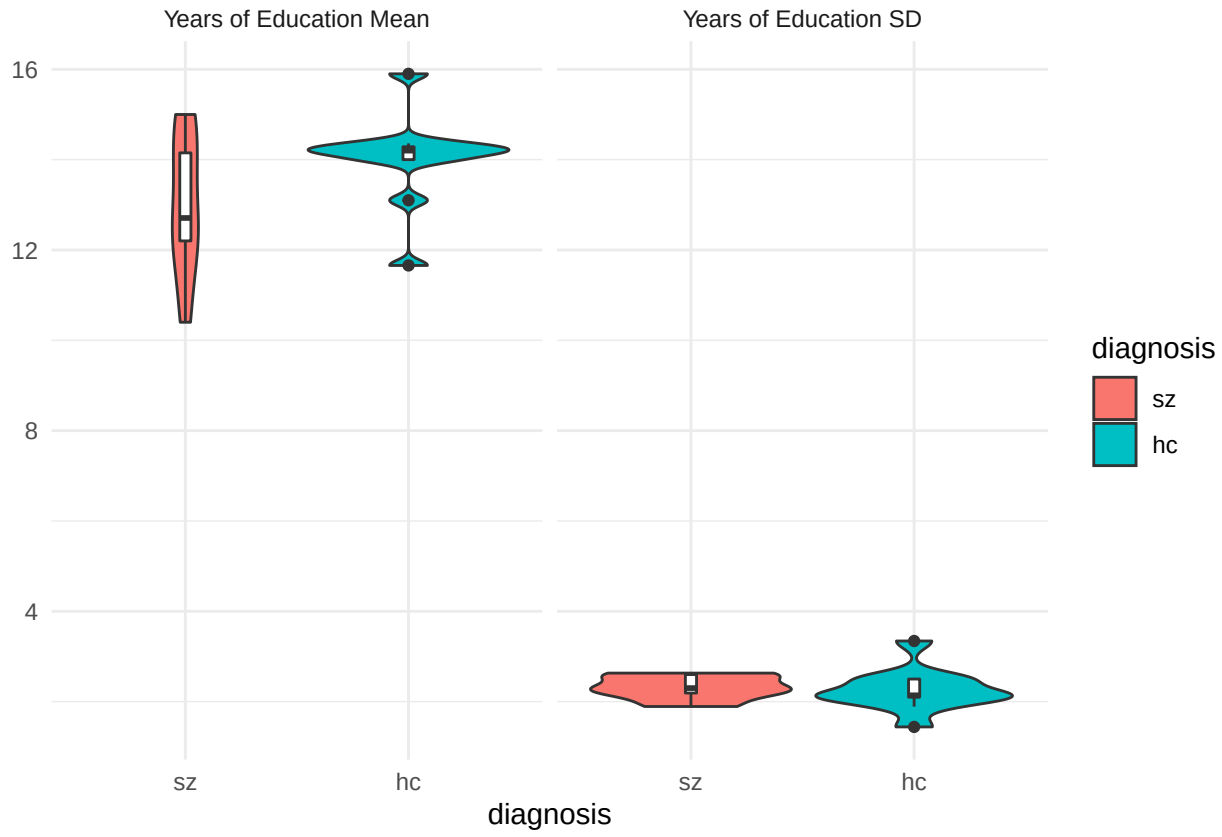
The two groups seems to be balanced in terms of participant's age. The means of the mean age equal to 36.44 and 34.89 for schizophrenic and healthy conditions respectively. The means of the standard deviations of age equal 8.53 and 10.35.

However, as the violin plot shows, the mean age seems to be distributed differently for the two conditions.

```r
edu_data <- data %>%
  select(studyid, diagnosis, education_m, education_sd) %>%
  pivot_longer(c(education_m, education_sd),
               names_to = c(".value","edu_parameter"),
               names_sep = "_") %>%
  mutate(edu_parameter = ifelse(edu_parameter == "m",
                                "Years of Education Mean",
                                "Years of Education SD"))

edu_data %>%
  ggplot(aes(x = diagnosis, y = education, fill = diagnosis)) +
    geom_violin() +
    geom_boxplot(width = 0.05, fill = "white") +
    facet_wrap(~edu_parameter)+
    labs(y = NULL) +
    theme_minimal()
```

Education

```
## Warning: Removed 16 rows containing non-finite values (stat_ydensity).

## Warning: Removed 16 rows containing non-finite values (stat_boxplot).
```



```
edu_data %>%
  group_by(diagnosis, edu_parameter) %>%
  summarise(mean = mean(education, na.rm = T) %>% round(2))
```

```
## `summarise()` has grouped output by 'diagnosis'. You can override using the
## `.groups` argument.
```

```
## # A tibble: 4 x 3
## # Groups:   diagnosis [2]
##   diagnosis edu_parameter            mean
##   <fct>     <chr>                   <dbl>
## 1 sz        Years of Education Mean 13.0
## 2 sz        Years of Education SD    2.32
## 3 hc        Years of Education Mean 14.0
## 4 hc        Years of Education SD    2.26
```

```
rm(edu_data)
```

(We didn't manage to find the correct interpretation of the Education variable. We for now just assumed it refers to the number of years spend in Education)

The two groups seems to be balanced in terms of the length of their education. The means of the mean number of years spend in education equal to 13.02 and 14.02 for schizophrenic and healthy conditions respectively. The means of the standard deviations of number of years spend in education equal 2.32 and 2.26.

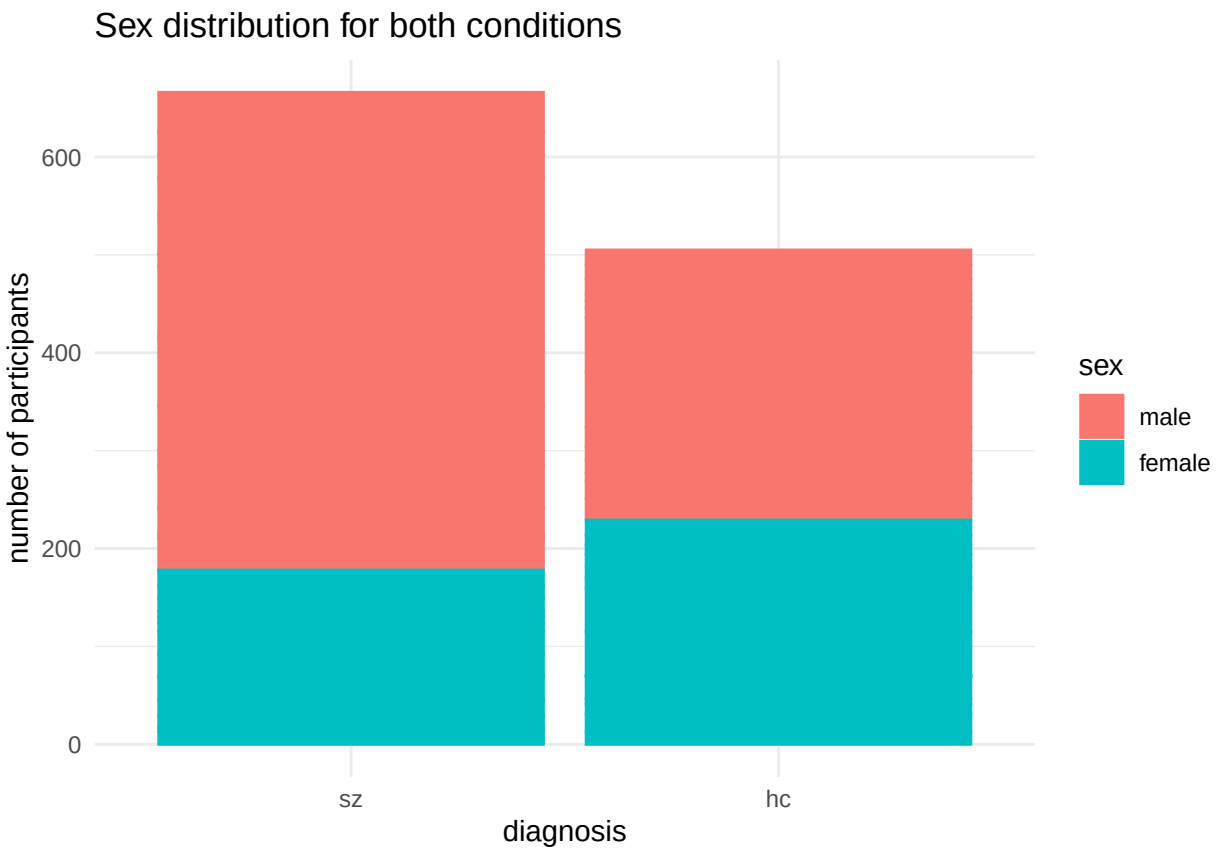
However, as the violin plot shows, the mean number of years in education seems to be distributed differently for the two conditions.

```
#were the conditions balanced in terms of sex?

sex_data <- data %>%
  select(male, female, diagnosis, n_diagnosis, studyid) %>%
    pivot_longer(c(male, female),
      names_to = "sex",
      values_to = "participants") %>%
  mutate(sex = sex %>% as_factor)

sex_data %>%
  ggplot(aes(x = diagnosis, y = participants, fill = sex, colour = sex)) +
    geom_col(stat = "identity") +
    ylab("number of participants")+
    theme_minimal() +
      labs(title = "Sex distribution for both conditions")
```

Sex

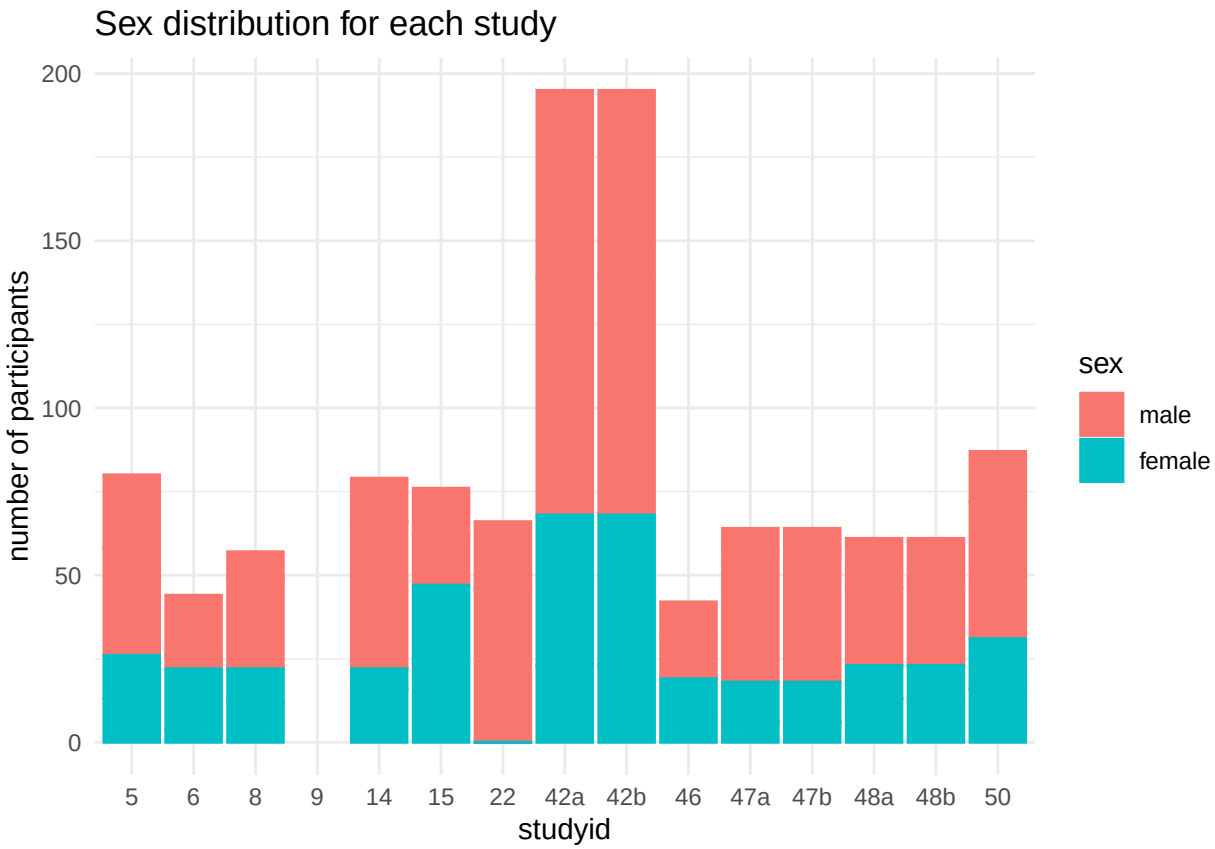
```
## Warning: Ignoring unknown parameters: stat
```

```
## Warning: Removed 4 rows containing missing values (position_stack).
```



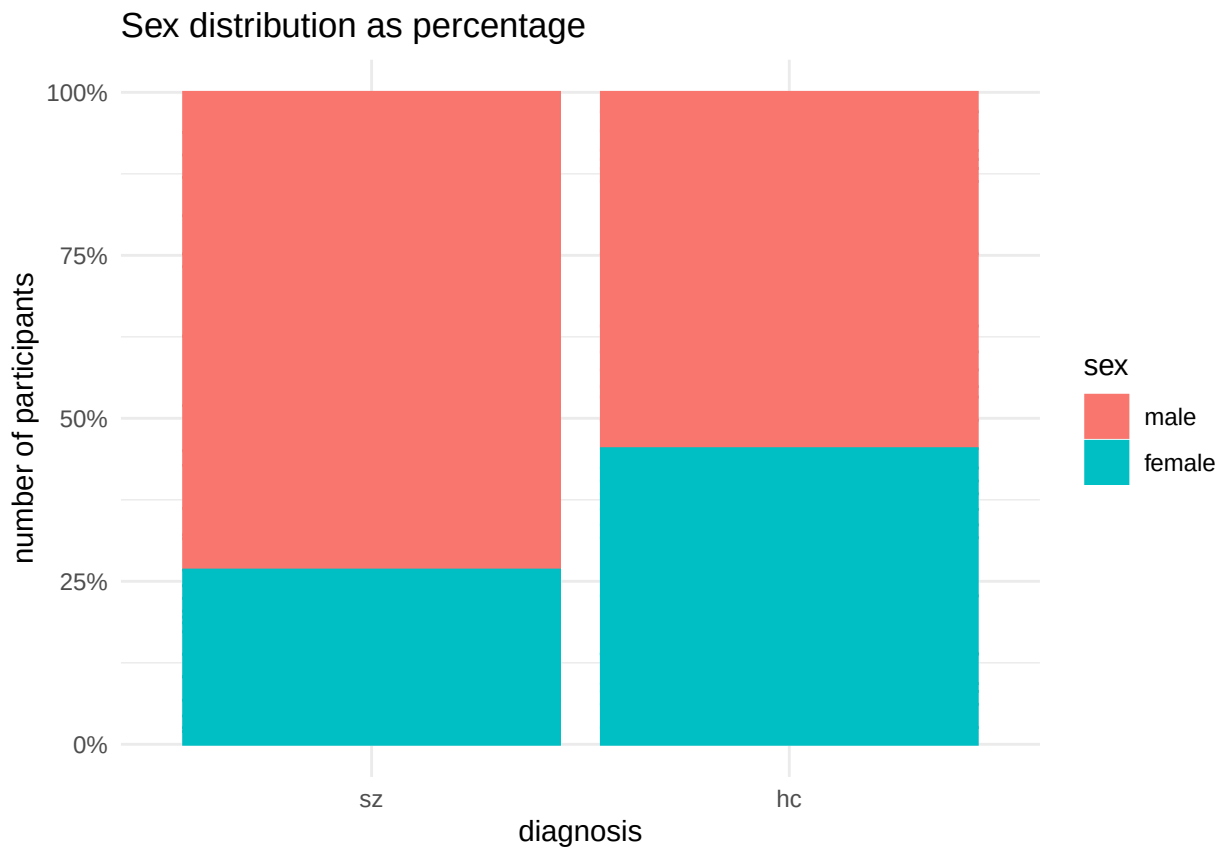

```
sex_data %>%
  ggplot(aes(x = studyid, y = participants, fill = sex, colour = sex)) +
    geom_col(stat = "identity") +
    ylab("number of participants")+
    theme_minimal() +
      labs(title = "Sex distribution for each study")
```

```
## Warning: Ignoring unknown parameters: stat
## Removed 4 rows containing missing values (position_stack).
```




```
sex_data %>%
  ggplot(aes(x = diagnosis, y = participants, fill = sex, colour = sex)) +
    geom_col(stat = 'identity', position = "fill") +
    scale_y_continuous(labels = scales::percent)+
    ylab("number of participants")+
    theme_minimal() +
    labs(title = "Sex distribution as percentage")
```

```
## Warning: Ignoring unknown parameters: stat
## Removed 4 rows containing missing values (position_stack).
```




```
sex_data %>%
  group_by(diagnosis, sex) %>%
  summarise(n = sum(participants, na.rm = TRUE)) %>%
  mutate(pct = n /sum(n)*100)
```

```
## `summarise()` has grouped output by 'diagnosis'. You can override using the
## `.groups` argument.

## # A tibble: 4 x 4
## # Groups:   diagnosis [2]
##   diagnosis sex        n   pct
##   <fct>     <fct>  <dbl> <dbl>
## 1 sz        male     488  73.3
## 2 sz        female   178  26.7
## 3 hc        male     276  54.7
## 4 hc        female   229  45.3
```

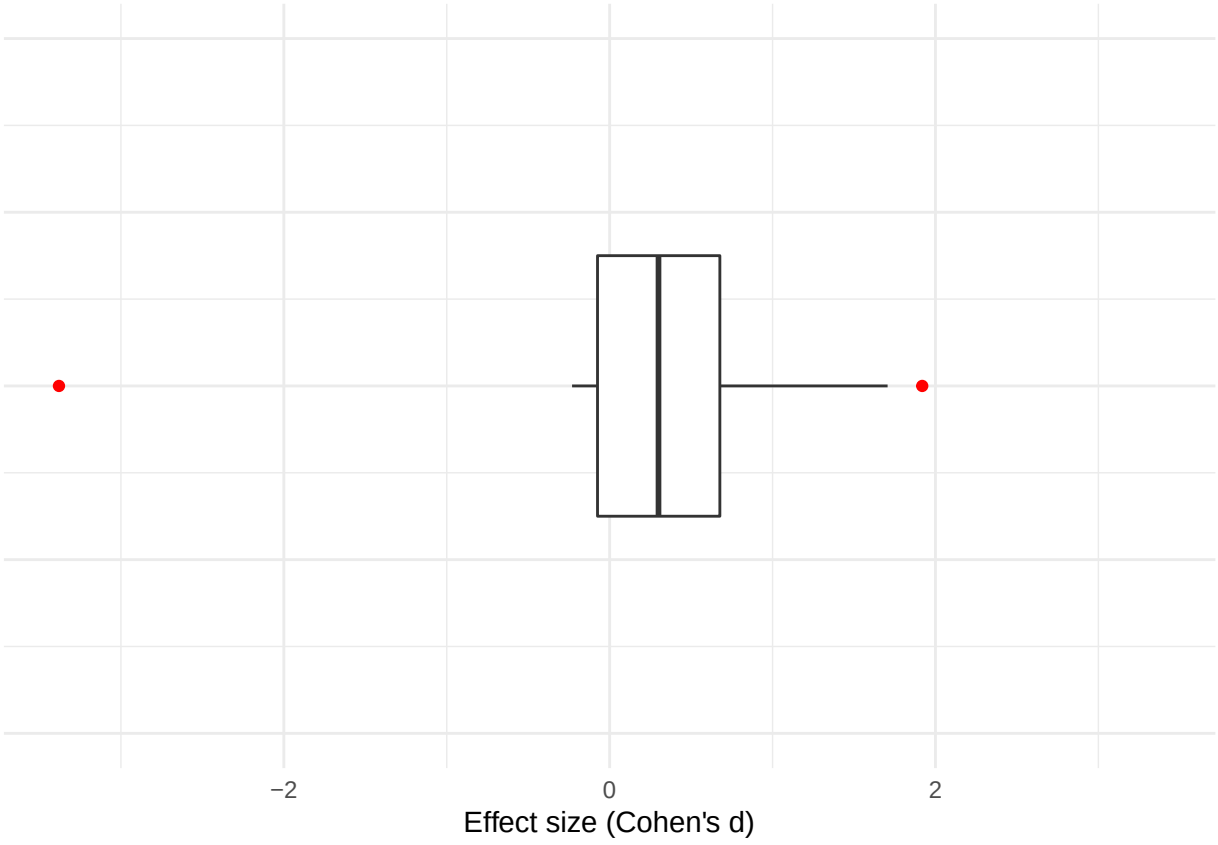
It appears that the distribution of sex is not equal. In the healthy control population, the ratio is lightly skewed, with an absolute number of 229 females to 276 males, equal to 45.3% to 54.7% of that population. Across the schizophrenic population, the values are more imbalanced with absolute numbers of 178 females and 488 males, or 26.7% to 73.3%.

The distribution also varies across studies, with almost all studies (excluding nr. 6 and 15) sampling visibly more male than female participants. Study 22 seems not include any female participants (perhaps by design, we were unable to find further information)

NOTE: as study 9 doesn't include a gender distribution, these values do not entirely represent the sample.

Influential studies

```
data_all <- data %>%
  distinct(studyid, effect, effect_sigma, sample_size) %>%
  rename(study = studyid)

data_all %>%
  ggplot() +
    geom_boxplot(aes(x = effect), outlier.color = 'red') +
    ylim(-1, 1) +
    xlim(min(data_all$effect), - min(data_all$effect)) +
    xlab("Effect size (Cohen's d)") +
    theme_minimal() +
    theme(axis.title.y = element_blank(),
          axis.text.y = element_blank(),
          axis.ticks.y = element_blank())
```




```
data_trimmed <- data_all %>%
  filter(!(effect > quantile(effect, 0.75) + 1.5*IQR(effect) |
         effect < quantile(effect, 0.25) - 1.5*IQR(effect)
          ))


dfs <- list(data_all, data_trimmed)
```

```r
map(dfs, ~ mean(.x$effect))
```

```
## [[1]]
## [1] 0.2438389
##
## [[2]]
## [1] 0.3937825
```

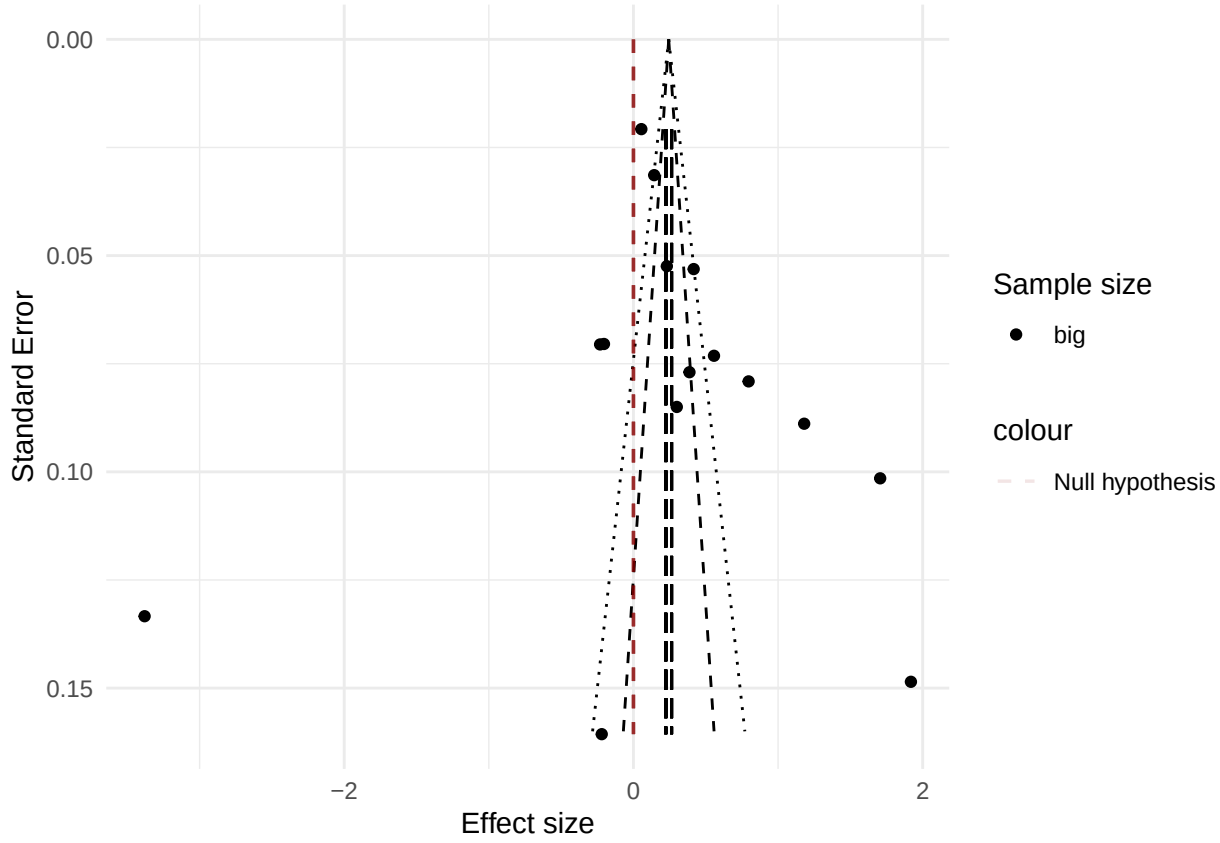
There were two outliers present in the data in terms of effect size. Both values were much larger than what can be usually expected from effect sizes in psychology (-3.38 and 1.92), and substantially impacted the $\bar{x}$ of our small sample.

We decided to create two datasets, one with and one without the recognised outliers to see how big of an impact they would have on the regression estimates.

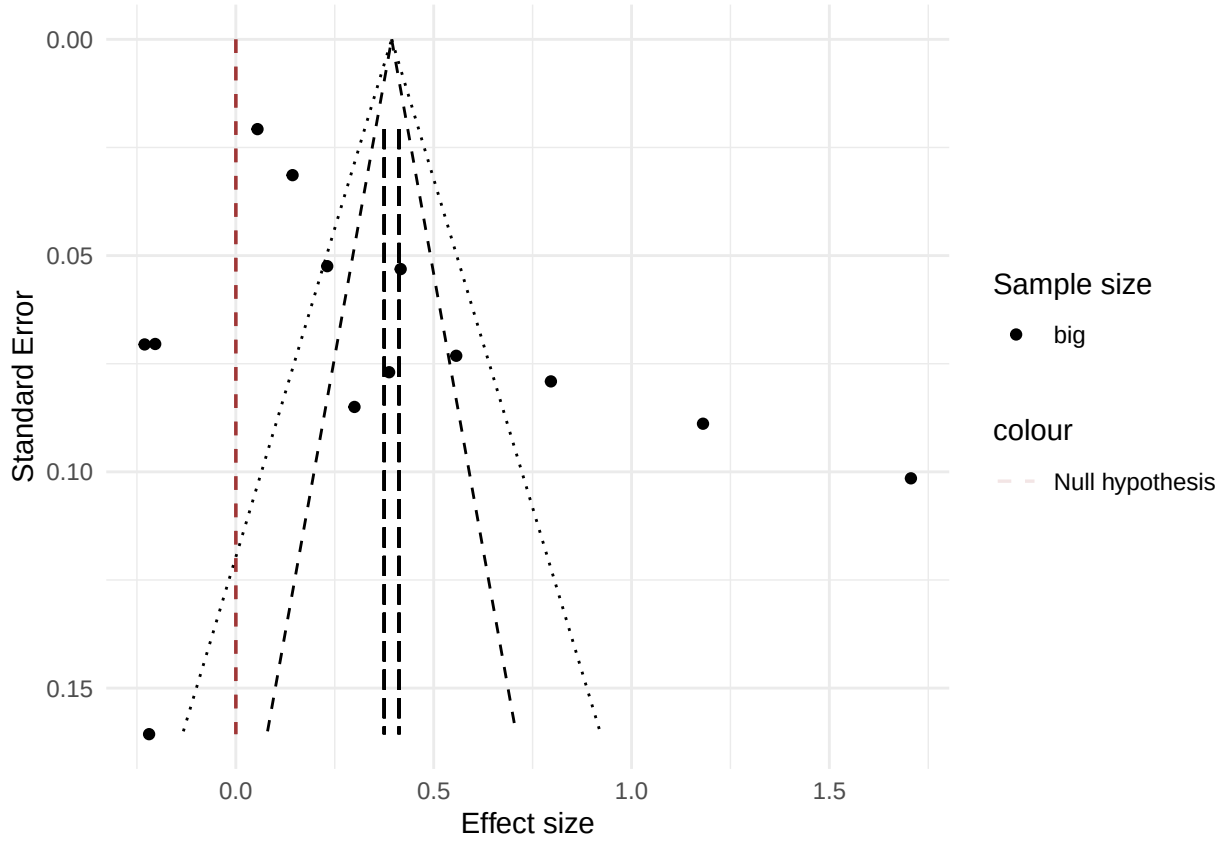
Publication bias

```r
map(.x = dfs, function(.x){

 effect_mean <- mean(.x$effect)
 effect_se <- sd(.x$effect_sigma) / sqrt(length(.x$study))

  uci <- effect_mean + 1.96*effect_se
  lci <-  effect_mean - 1.96*effect_se

  funnel_plot(.x) +
    geom_segment(aes(x = uci, y = min(effect_sigma), xend = uci, yend = max(effect_sigma)), linetype =
    geom_segment(aes(x = lci, y = min(effect_sigma), xend = lci, yend = max(effect_sigma)), linetype =
})
```

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


0.00
0.05
0.10
0.15
Standard Error
−2
0
2
Effect size
Sample size
big
colour
Null hypothesis

```
##
## [[2]]
```



There isn't a single study that would have less then 30 participants, and the plot doesn't seem to be symmetrical - much more studies appear over the right side of the triangle (bigger positive effect sizes). This might suggest the presence of publication bias. Smaller studies tend to have bigger standard errors which makes them more likely to be deemed not significant, and a positive effect might be expected because of previous literature. Alternatively, the asymmetry might be explained by some methodological differences between small and big sample studies (different measuring technologies and techniques, differences in the analysis process, etc.). More importantly, it has to be noted that the number of studies included is very low (n = 15), and so the influence of random noise in the sample might be substantial.

Building the models

```
f <- bf(effect | se(effect_sigma) ~ 1 + (1|study))
```

Defining the formula

```
get_prior(f, data_all)
```

Prior only

```
##                 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                  0
##    student_t(3, 0, 2.5)        sd Intercept study                  0
##        source
##       default
##       default
##  (vectorized)
##  (vectorized)
```

```
priors <- c(prior(normal(0, 0.6), class = Intercept),
            prior(normal(0, 1), class = sd))
```

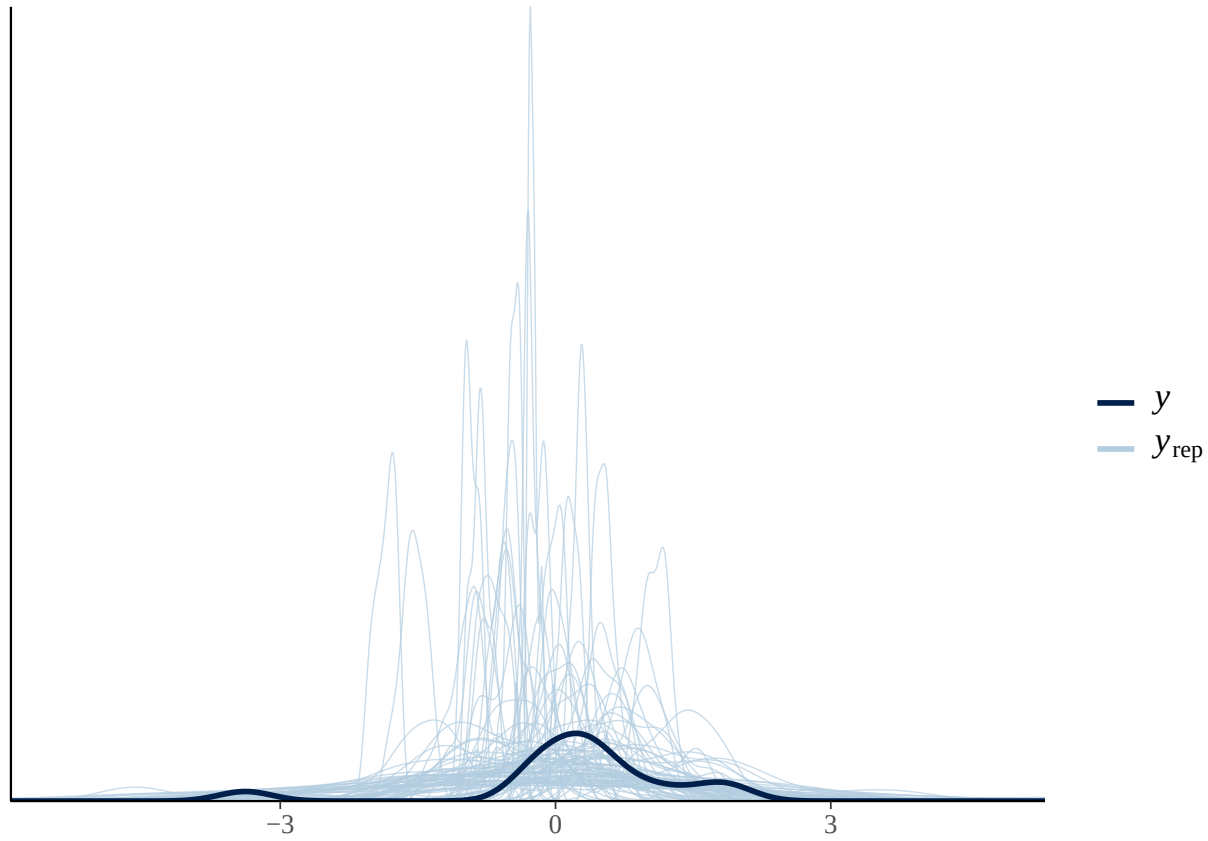
```
prior_m_all <- brm(f,
                   data_all,
                   family = gaussian,
                   prior = priors,
                   sample_prior = 'only',
                   backend = 'cmdstanr',
                   cores = 3,
                   control = list(
                     adapt_delta = 0.9,
                     max_treedepth = 20))
```

```
## Start sampling

## Running MCMC with 4 chains, at most 3 in parallel...
##
## Chain 1 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 1 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 1 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1 Iteration:  700 / 2000 [ 35%]  (Warmup)
## 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 1 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 1 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## 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 2 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 2 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2 Iteration:  700 / 2000 [ 35%]  (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 2 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 2 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3 Iteration:  100 / 2000 [  5%]  (Warmup)
## Chain 3 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3 Iteration:  300 / 2000 [ 15%]  (Warmup)
## Chain 3 Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 3 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 3 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3 Iteration:  900 / 2000 [ 45%]  (Warmup)
## Chain 3 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 3 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 1 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 1 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## 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 0.2 seconds.
## Chain 2 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 2 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 2 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 2 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 2 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2 finished in 0.2 seconds.
## Chain 3 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 3 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 3 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 3 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 3 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3 finished in 0.2 seconds.
## Chain 4 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4 Iteration:  100 / 2000 [  5%]  (Warmup)
## Chain 4 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4 Iteration:  300 / 2000 [ 15%]  (Warmup)
## Chain 4 Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 4 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 4 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4 Iteration:  900 / 2000 [ 45%]  (Warmup)
```

```
## Chain 4 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 4 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 4 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 4 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 4 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 4 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4 finished in 0.2 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.2 seconds.
## Total execution time: 1.4 seconds.
```

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




```
summary(prior_m_all)
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: effect | se(effect_sigma) ~ 1 + (1 | study)
```

```
##    Data: data_all (Number of observations: 15)
##   Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup draws = 4000
##
## Group-Level Effects:
## ~study (Number of levels: 15)
##               Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(Intercept)     0.79      0.60     0.03     2.23 1.00     3216     1992
##
## Population-Level Effects:
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept    -0.00      0.58    -1.13     1.13 1.01     6859     2259
##
## 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).
```

```
m_all <- brm(f,
             data_all,
             family = gaussian,
             prior = priors,
             sample_prior = T,
             backend = 'cmdstanr',
             cores = 3,
             control = list(
               adapt_delta = 0.9,
               max_treedepth = 20))
```

Fitting the models

```
## Start sampling

## Running MCMC with 4 chains, at most 3 in parallel...
##
## Chain 1 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1 Iteration:  100 / 2000 [  5%]  (Warmup)
## Chain 1 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2 Iteration:  100 / 2000 [  5%]  (Warmup)
## Chain 2 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3 Iteration:  100 / 2000 [  5%]  (Warmup)
## Chain 1 Iteration:  300 / 2000 [ 15%]  (Warmup)
## Chain 1 Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 1 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1 Iteration:  700 / 2000 [ 35%]  (Warmup)
```

```
## Chain 1 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1 Iteration:  900 / 2000 [ 45%]  (Warmup)
## Chain 2 Iteration:  300 / 2000 [ 15%]  (Warmup)
## Chain 2 Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 2 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2 Iteration:  700 / 2000 [ 35%]  (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 3 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3 Iteration:  300 / 2000 [ 15%]  (Warmup)
## Chain 3 Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 3 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 3 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3 Iteration:  900 / 2000 [ 45%]  (Warmup)
## Chain 1 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1 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 2 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 3 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 1 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 2 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 3 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 3 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 1 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 3 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 3 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 1 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 1 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 3 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3 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 3 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1 finished in 1.0 seconds.
```

```
## Chain 2 finished in 0.9 seconds.
## Chain 3 finished in 0.9 seconds.
## Chain 4 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4 Iteration:  100 / 2000 [  5%]  (Warmup)
## Chain 4 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4 Iteration:  300 / 2000 [ 15%]  (Warmup)
## Chain 4 Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 4 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 4 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4 Iteration:  900 / 2000 [ 45%]  (Warmup)
## Chain 4 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 4 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 4 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 4 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 4 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 4 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4 finished in 0.8 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.9 seconds.
## Total execution time: 2.4 seconds.
```

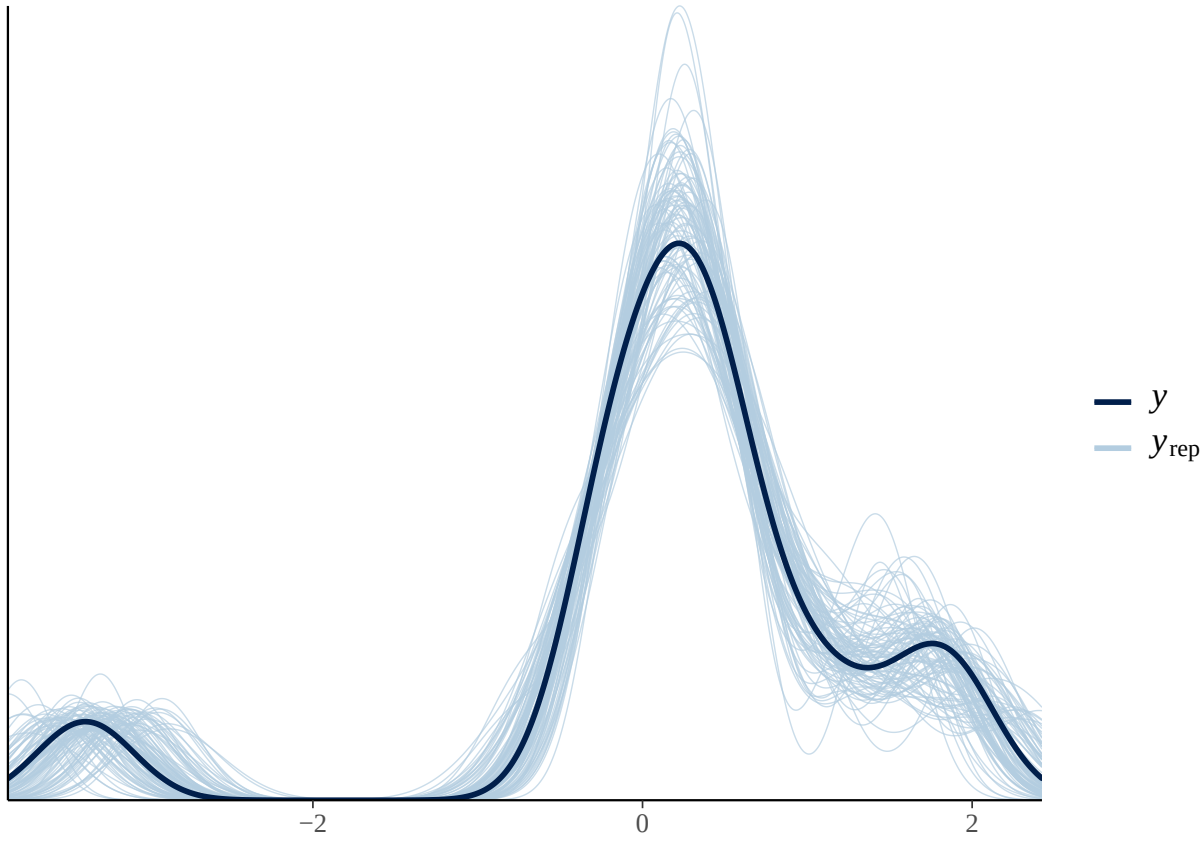
```
m_trimmed <- update(m_all,
                    newdata = data_trimmed)
```

```
## Start sampling

## Running MCMC with 4 sequential chains...
##
## Chain 1 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 1 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 1 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1 Iteration:  700 / 2000 [ 35%]  (Warmup)
## 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 1 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 1 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 1 Iteration: 1400 / 2000 [ 70%]  (Sampling)
```

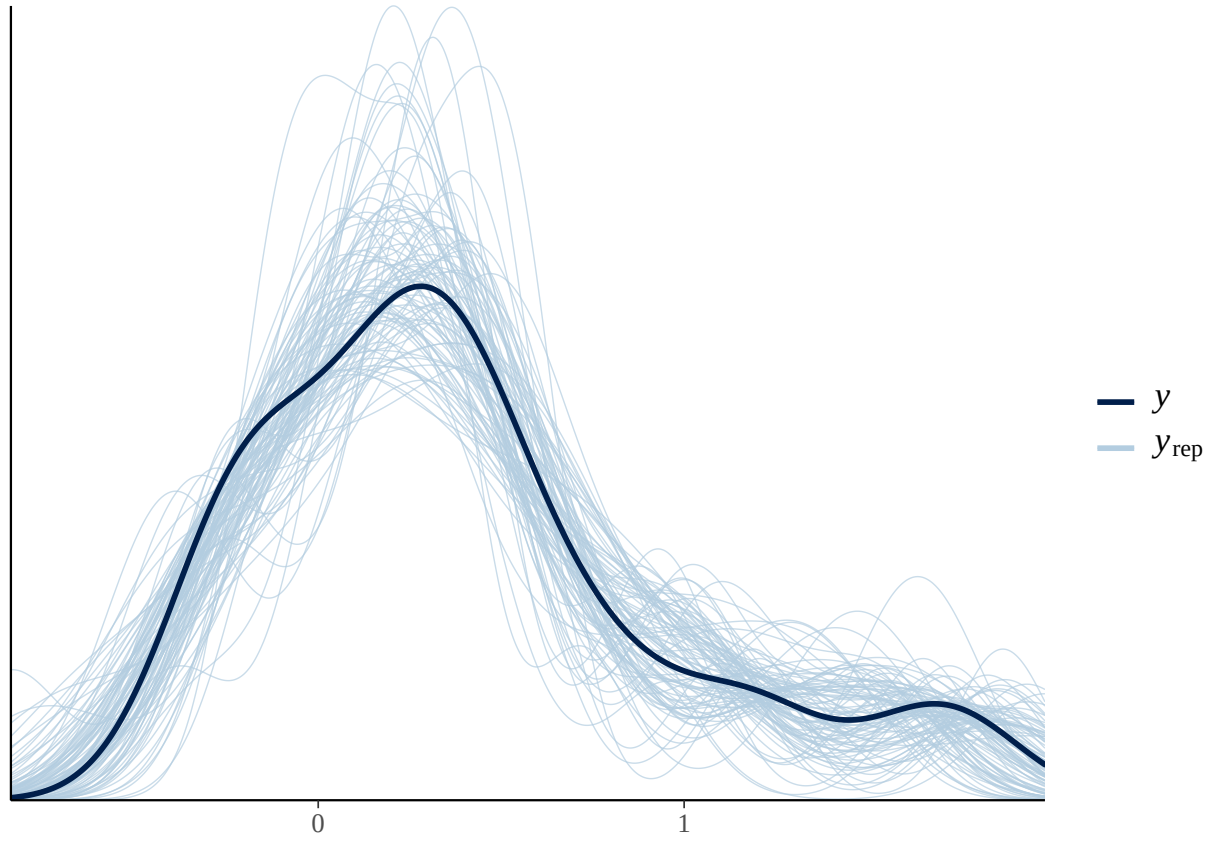
```
## Chain 1 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 1 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 1 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 1 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1 finished in 0.6 seconds.
## 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 2 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 2 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2 Iteration:  700 / 2000 [ 35%]  (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 2 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 2 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 2 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 2 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 2 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 2 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2 finished in 0.6 seconds.
## Chain 3 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3 Iteration:  100 / 2000 [  5%]  (Warmup)
## Chain 3 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3 Iteration:  300 / 2000 [ 15%]  (Warmup)
## Chain 3 Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 3 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 3 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3 Iteration:  900 / 2000 [ 45%]  (Warmup)
## Chain 3 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 3 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 3 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 3 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 3 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 3 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3 finished in 0.6 seconds.
## Chain 4 Iteration:    1 / 2000 [  0%]  (Warmup)
```

```
## Chain 4 Iteration:  100 / 2000 [  5%]  (Warmup)
## Chain 4 Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4 Iteration:  300 / 2000 [ 15%]  (Warmup)
## Chain 4 Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4 Iteration:  500 / 2000 [ 25%]  (Warmup)
## Chain 4 Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4 Iteration:  700 / 2000 [ 35%]  (Warmup)
## Chain 4 Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4 Iteration:  900 / 2000 [ 45%]  (Warmup)
## Chain 4 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4 Iteration: 1100 / 2000 [ 55%]  (Sampling)
## Chain 4 Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4 Iteration: 1300 / 2000 [ 65%]  (Sampling)
## Chain 4 Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4 Iteration: 1500 / 2000 [ 75%]  (Sampling)
## Chain 4 Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4 Iteration: 1700 / 2000 [ 85%]  (Sampling)
## Chain 4 Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4 Iteration: 1900 / 2000 [ 95%]  (Sampling)
## Chain 4 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4 finished in 0.6 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 0.6 seconds.
## Total execution time: 2.7 seconds.
```

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




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




```
summary(m_all)
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: effect | se(effect_sigma) ~ 1 + (1 | study)
##    Data: data_all (Number of observations: 15)
##   Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup draws = 4000
##
## Group-Level Effects:
## ~study (Number of levels: 15)
##               Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(Intercept)     1.21      0.23     0.86     1.78 1.01      335      606
##
## Population-Level Effects:
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept     0.19      0.30    -0.39     0.79 1.02      301      208
##
## 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).
```

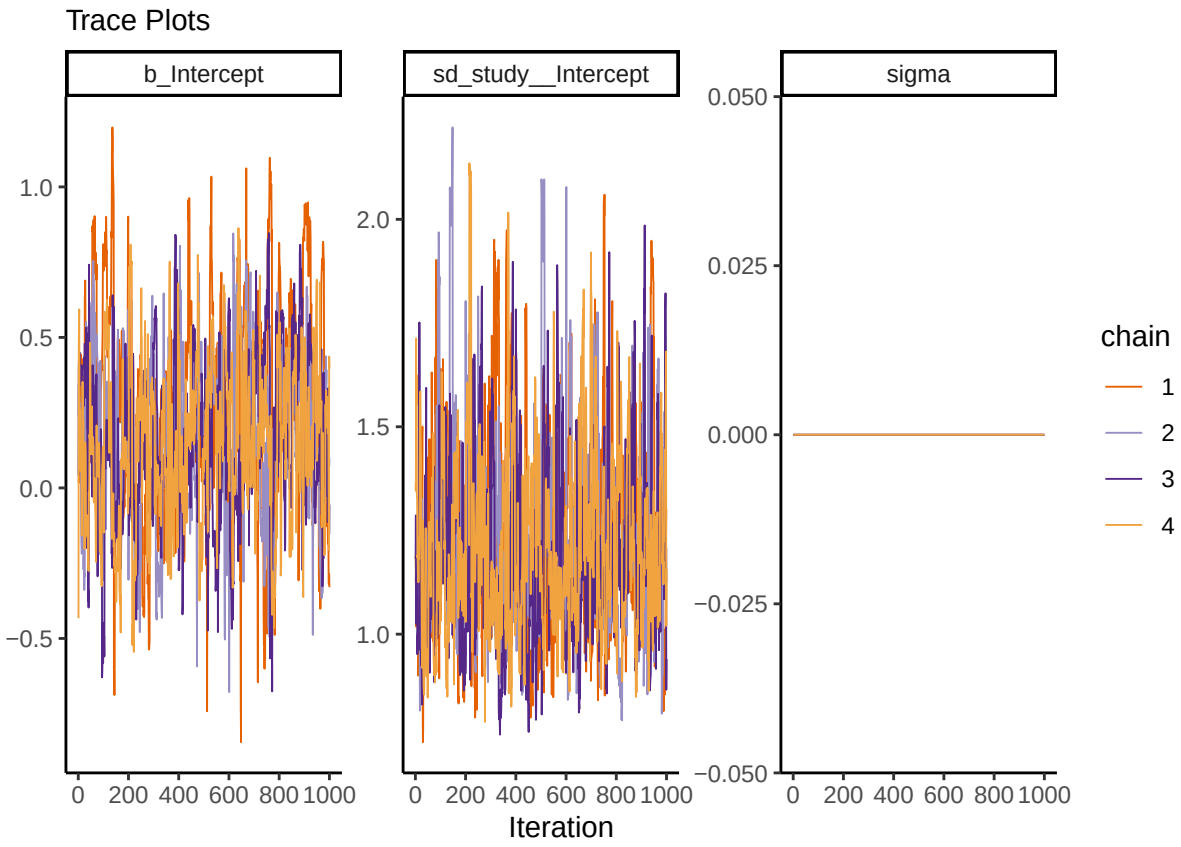
```
summary(m_trimmed)
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: effect | se(effect_sigma) ~ 1 + (1 | study)
##    Data: data_trimmed (Number of observations: 13)
##   Draws: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup draws = 4000
##
## Group-Level Effects:
## ~study (Number of levels: 13)
##               Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(Intercept)     0.62      0.14     0.40     0.95 1.00      501      871
##
## Population-Level Effects:
##           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept     0.37      0.18     0.01     0.73 1.01      320      470
##
## 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).
```

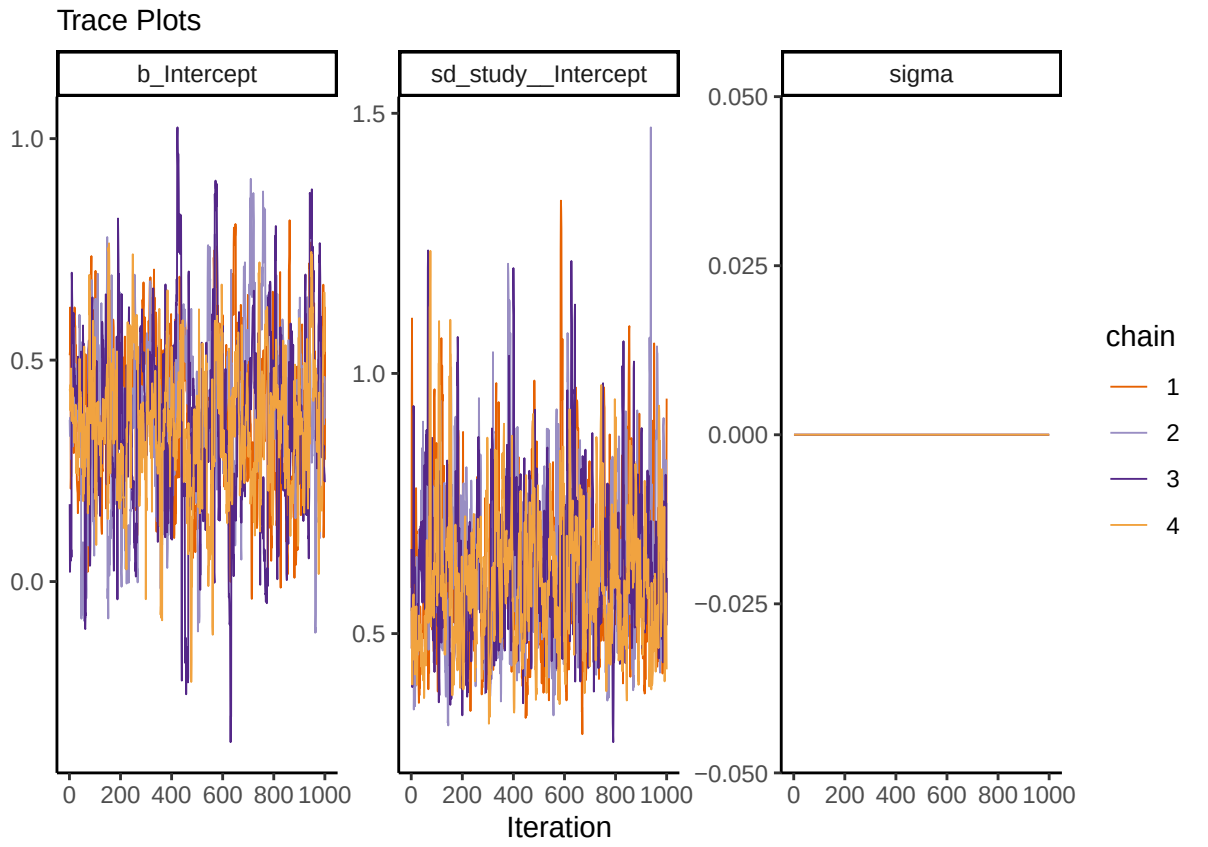
```
models <- list(m_all, m_trimmed)

# launch_shinystan(f_m) # - very nice for exploring and diagnosing the model, but opens up in a new win

map(models, ~ mcmc_plot(.x, type = 'trace') +
    theme_classic() +
    scale_color_manual(values=c("#E66101", "#998EC3", "#542788", "#F1A340")) +
    ylab("") +
    xlab("Iteration") +
    labs(subtitle = 'Trace Plots'))
```

Convergenece checks

```
## No divergences to plot.
## Scale for 'colour' is already present. Adding another scale for 'colour',
## which will replace the existing scale.
##
## No divergences to plot.
## Scale for 'colour' is already present. Adding another scale for 'colour',
## which will replace the existing scale.

## [[1]]
```

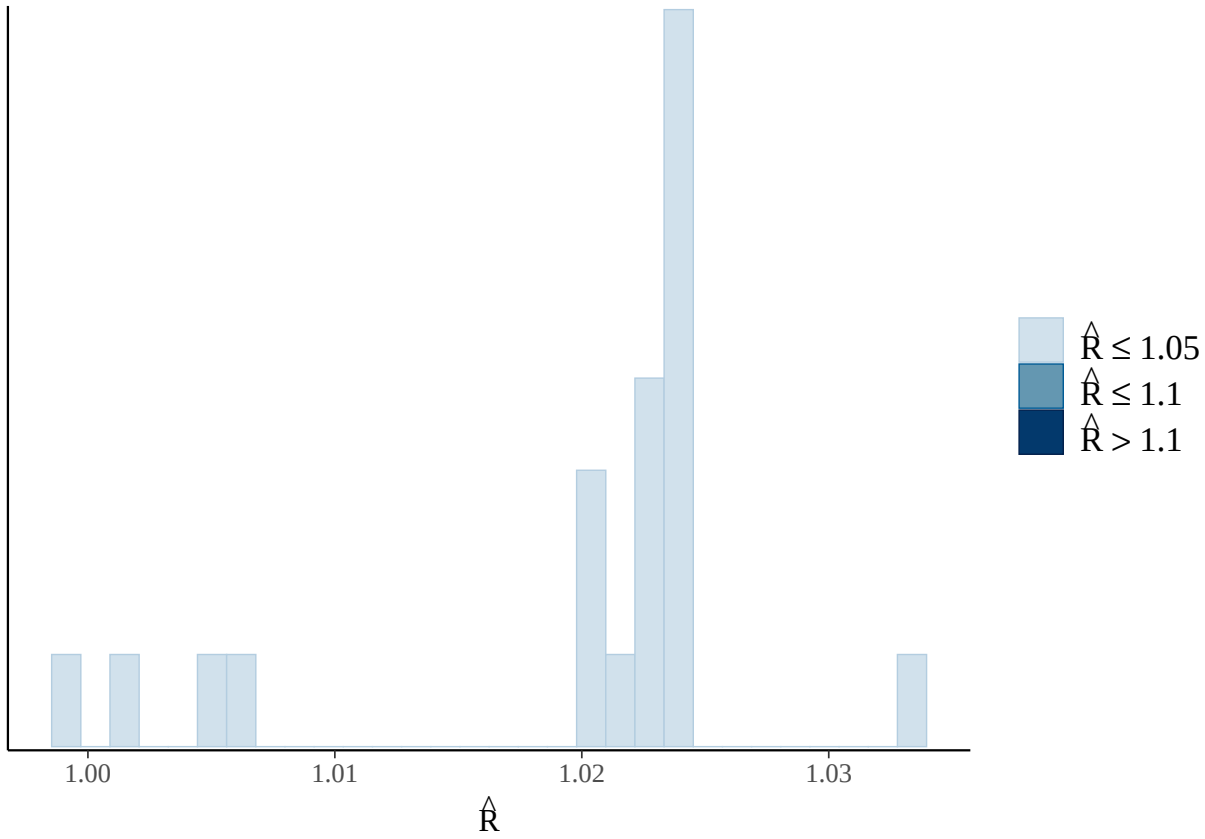

Trace Plots
b_Intercept
sd_study__Intercept
sigma
1.0
0.5
0.0
−0.5
2.0
1.5
1.0
0.050
0.025
0.000
−0.025
−0.050
0
200
400
600
800
1000
Iteration
chain
1
2
3
4

```
##
## [[2]]
```

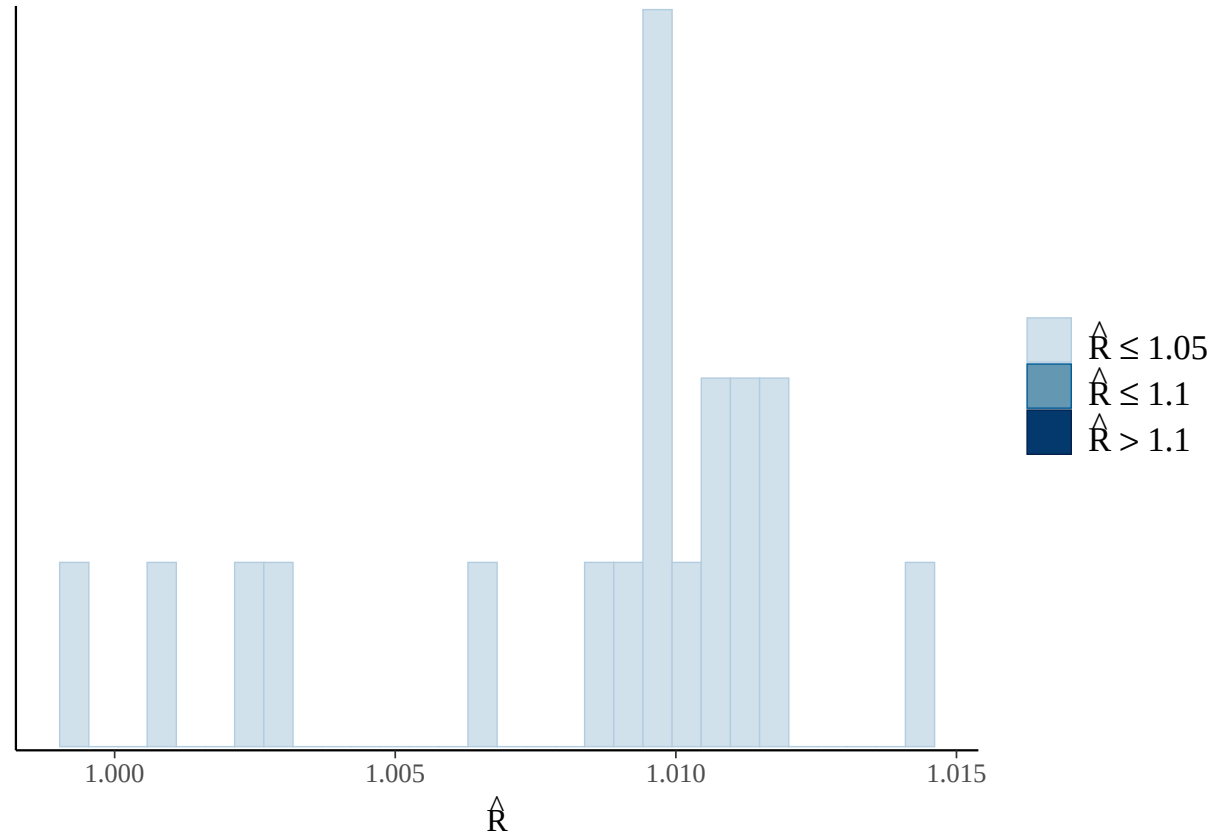



```
map(models, ~ mcmc_plot(.x, type = 'rhat_hist'))
```

```
## Warning: Dropped 1 NAs from 'new_rhat(rhat)'.
## Dropped 1 NAs from 'new_rhat(rhat)'.

## [[1]]

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
##
## [[2]]

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

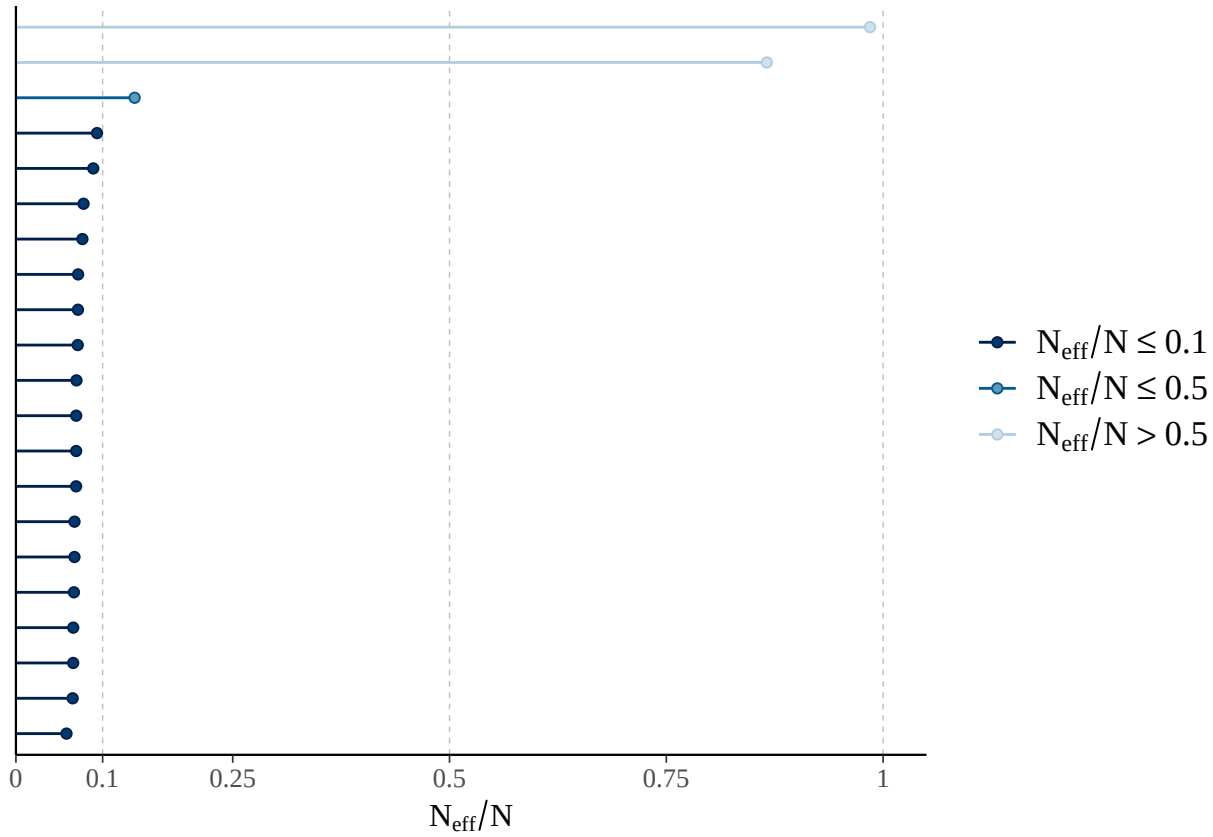



```
map(models, ~ mcmc_plot(.x, type = 'neff'))
```

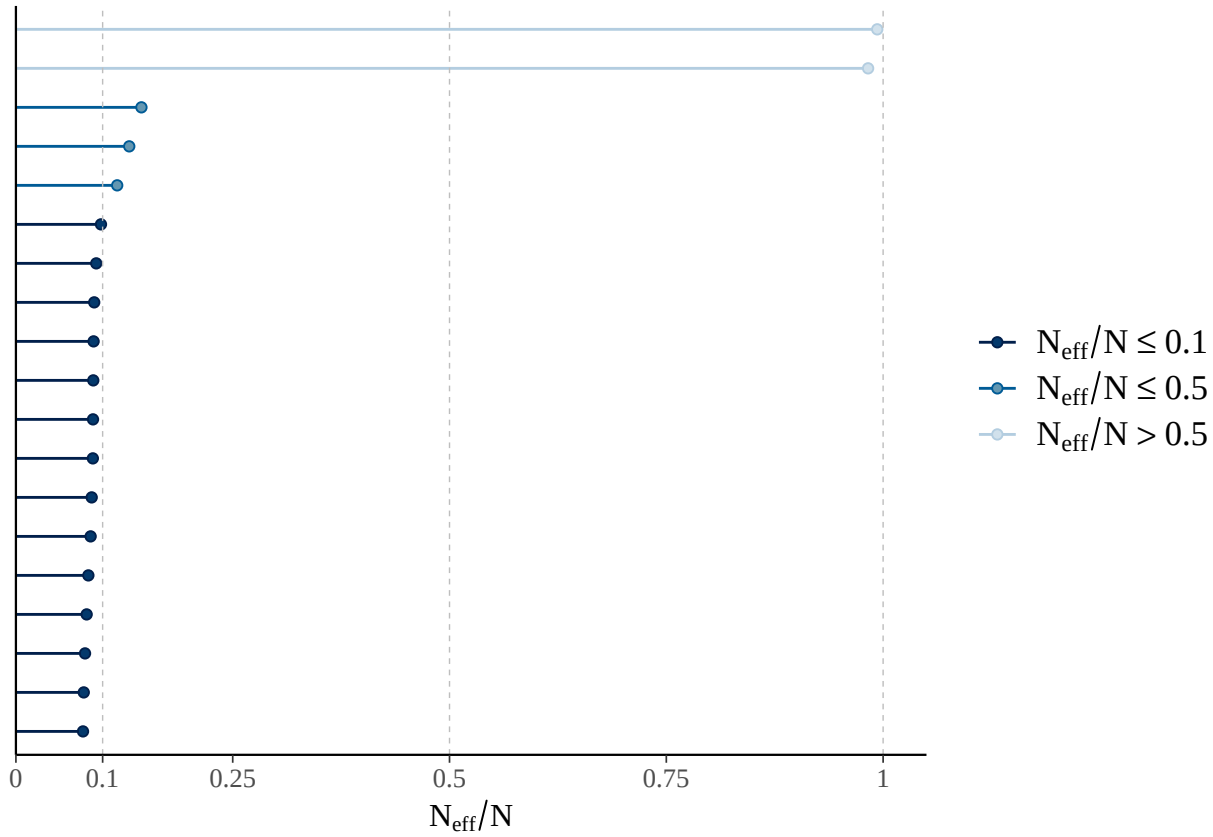
```
## Warning: Dropped 1 NAs from 'new_neff_ratio(ratio)'.
```

```
## Warning: Dropped 1 NAs from 'new_neff_ratio(ratio)'.
```

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



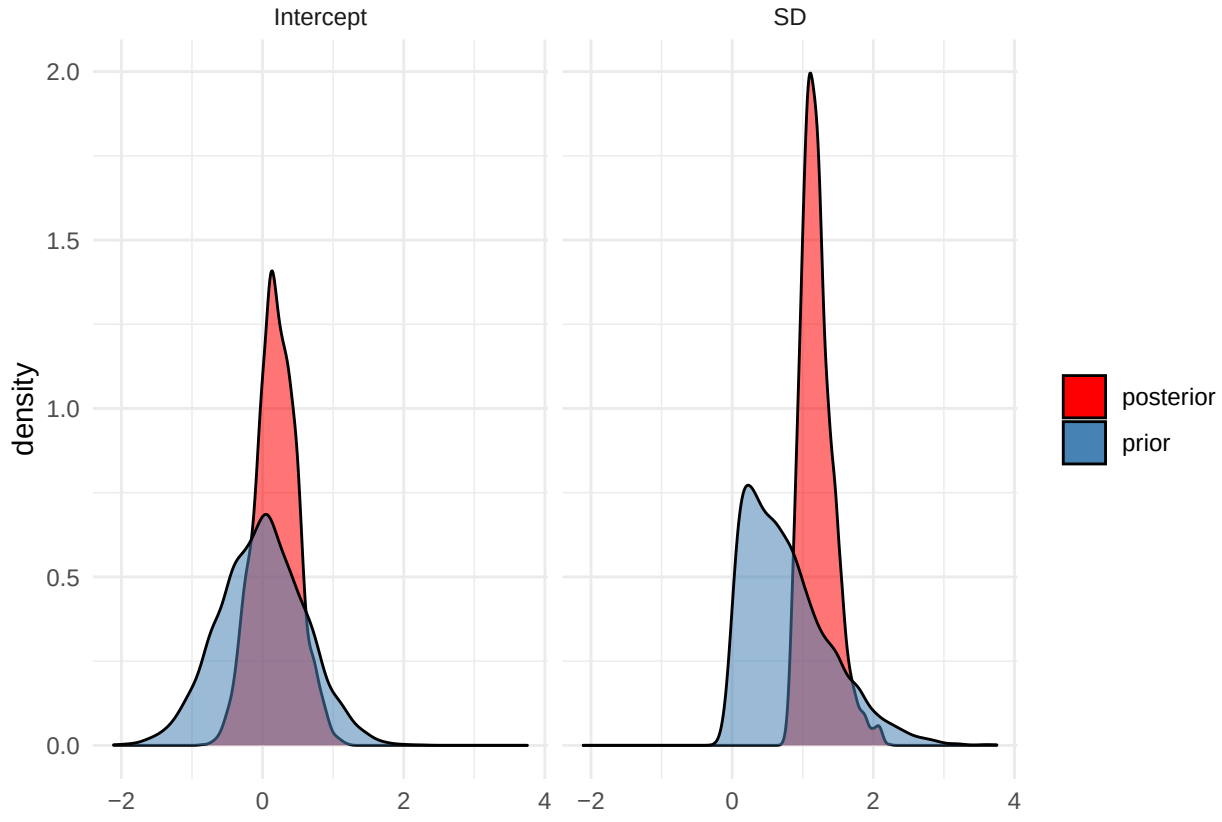

```
##
## [[2]]
```



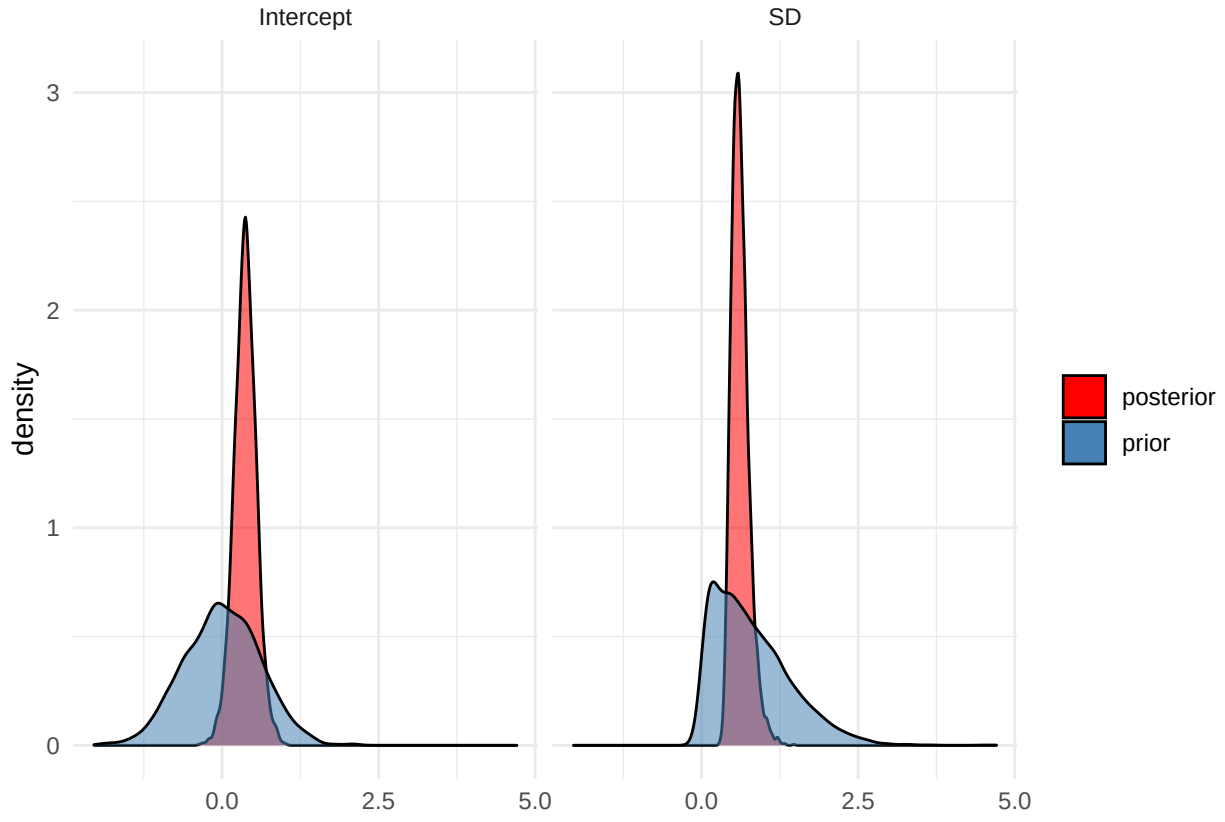
Posterior - prior update checks

```
map(models, pp_update_plot)
```

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

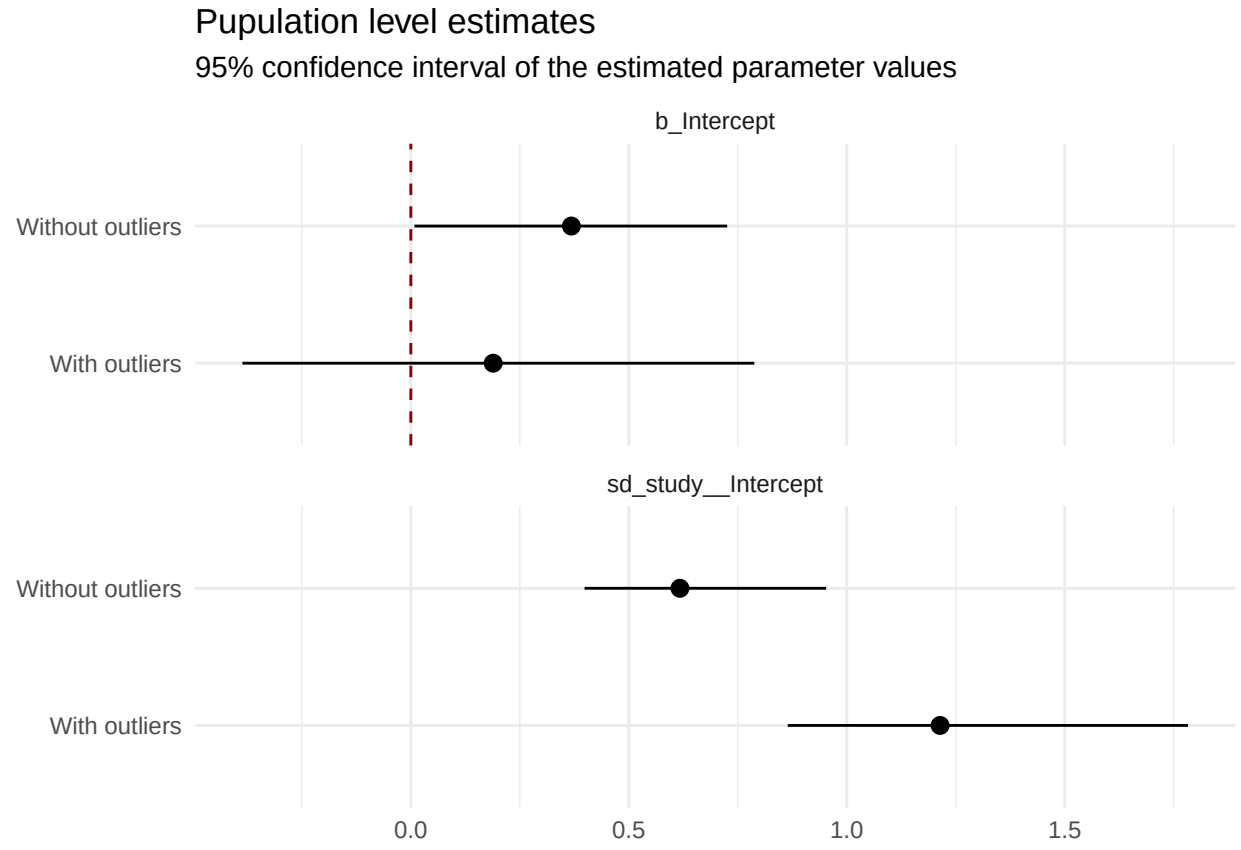



```
##
## [[2]]
```



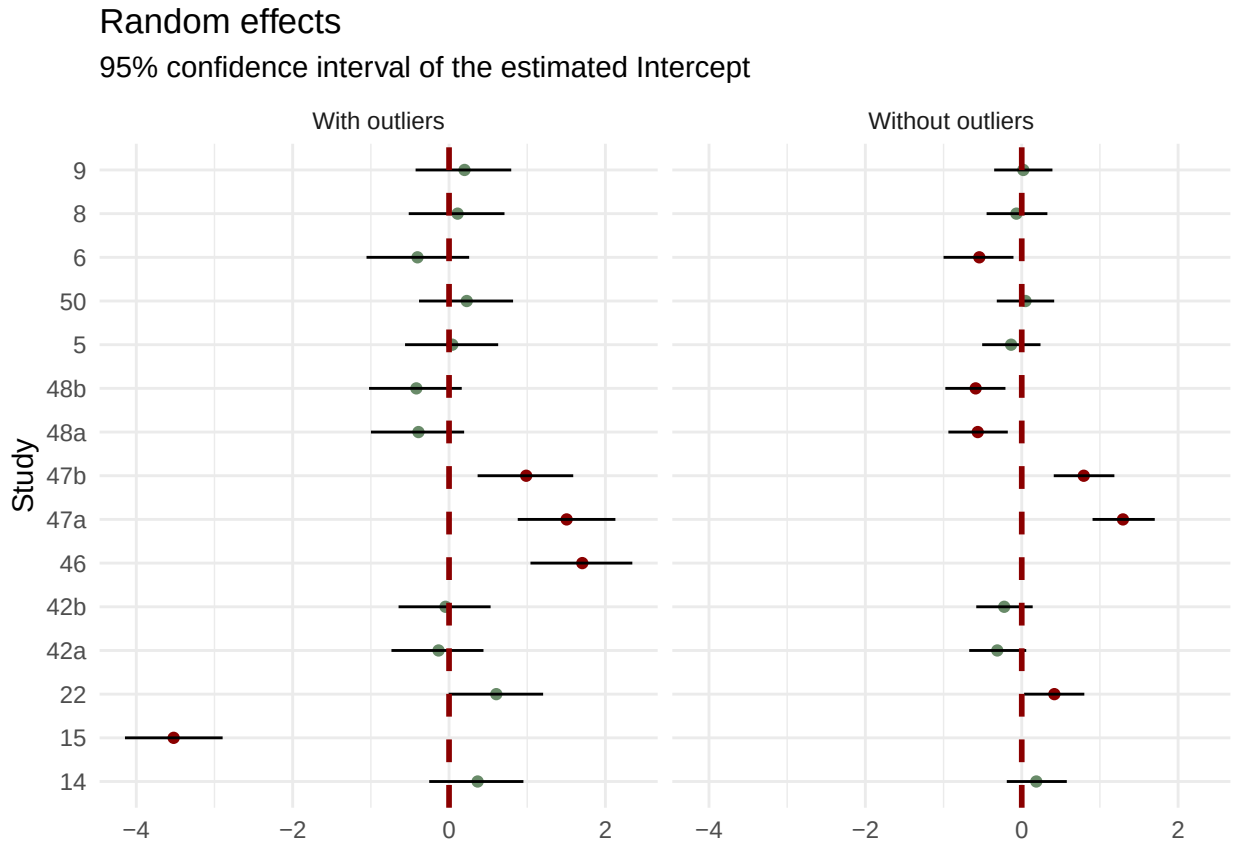
Comparing the models #### Population effects

```
model_names <- c('With outliers', 'Without outliers')

pop_effects_plot(models, model_names)
```



Random effects

```
rand_effects_plot(models, model_names)
```



Conclusions