S2: analysis of pupil sizes with brms

I. S. Plank

2025-03-21

S2.1 Introduction

This R Markdown script analyses pupil size data from the PAL (probabilistic associative learning) task of the EMBA project. The data was preprocessed and then selected based on a cross-validation procedure before being read into this script.

Some general settings

```
# number of simulations
nsim = 500

# set number of iterations and warmup for models
iter = 6000
warm = 1500

# set the maximum treedepth
max_depth = 15

# set the seed
set.seed(2468)

# and describe the priors we want to use in our sensitivity analysis
ls.priors = c("chosen",
  "sdx2",    "sdx4",   "sdx8",
  "sdx0.5", "sdx0.25", "sdx0.125"
  )
```

Package versions

The following packages are used in this RMarkdown file:

```
## [1] "R version 4.4.3 (2025-02-28)"
```

```
## [1] "knitr version 1.46"
## [1] "ggplot2 version 3.5.1"
## [1] "brms version 2.21.0"
## [1] "designr version 0.1.13"
## [1] "bridgesampling version 1.1.2"
## [1] "tidyverse version 2.0.0"
## [1] "ggpubr version 0.6.0"
## [1] "ggrain version 0.0.4"
## [1] "bayesplot version 1.11.1"
## [1] "SBC version 0.3.0.9000"
```

```
## [1] "rstatix version 0.7.2"
## [1] "R.matlab version 3.7.0"
## [1] "effectsize version 0.8.9"
## [1] "BayesFactor version 0.9.12.4.7"
## [1] "bayestestR version 0.15.0"
```

Introduction

We planned to determine the group-level effect subjects following Barr (2013). For each model, experiment specific priors were set based on previous literature or the task (see comments in the code).

We perform prior predictive checks as proposed in Schad, Betancourt and Vasishth (2020) using the SBC package. To do so, we create 500 simulated datasets where parameters are simulated from the priors. These parameters are used to create one fake dataset. Both the true underlying parameters and the simulated discrimination values are saved.

Then, we create graphs showing the prior predictive distribution of the simulated discrimination threshold to check whether our priors fit our general expectations about the data. Next, we perform checks of computational faithfulness and model sensitivity as proposed by Schad, Betancourt and Vasishth (2020) and implemented in the SBC package. We create models for each of the simulated datasets. Last, we calculate performance metrics for each of these models, focusing on the population-level parameters.

[!EXPLAIN:]CV based on effect of expectancy.

Preparation

First, we load the data and print how many participants in each group.

```
df = read_csv(file.path('data', 'CV_pup_sum', "CV_data_C(expected, Sum)[S.expected].csv"),
              show_col_types = F) %>%
  drop_na() %>%
  merge(., read_csv(file.path('HGF_results', 'main', 'eHGF-C_results.csv'),
                    show_col_types = F),
        all = T) %>%
  mutate_if(is.character, as.factor)

# get rid of NAs
df = df %>% drop_na()

# print the number of participants per group
kable(df %>% select(subID, diagnosis) %>% distinct() %>% group_by(diagnosis) %>% count())
```

diagnosis	n
ADHD	17
ASD	18
BOTH	23
COMP	18

```
# print the contrasts
contrasts(df$diagnosis) = contr.sum(4)
contrasts(df$diagnosis)
```

```
##      [,1] [,2] [,3]
## ADHD    1    0    0
## ASD     0    1    0
```

```
## BOTH    0    0    1
## COMP   -1   -1   -1
```

```
contrasts(df$expected)  = contr.sum(2)
contrasts(df$expected)
```

```
##            [,1]
## expected      1
## unexpected   -1
```

S2.2 Pupil size extracted using CV

Model setup

```
# code for filenames
code = "PAL-pup"

# model formula
f.pup = brms::bf(rel_pupil ~ diagnosis * expected + rts +
                   (1 | subID)
                 )

# set weakly informative priors
priors = c(
  prior(normal(0,    0.10),  class = Intercept),
  prior(normal(0.15, 0.15),  class = sigma),
  prior(normal(0.15, 0.15),  class = sd),
  prior(normal(0,    0.05),  class = b)
)

# change Intercept based on mean in the data > only thing adjusted between models
priors = priors %>%
  mutate(
    prior = if_else(
      class == "Intercept",
      gsub("\\(.*,", paste0("(", round(mean(df$rel_pupil),3), ", "), prior), prior)
  )

kable(priors)
```

prior	class	coef	group	resp	dpar	nlpar	lb	ub	source
normal(0.07, 0.1)	Intercept						NA	NA	user
normal(0.15, 0.15)	sigma						NA	NA	user
normal(0.15, 0.15)	sd						NA	NA	user
normal(0, 0.05)	b						NA	NA	user

Simulation-based calibration

```
# check if the SBC already exists
if (file.exists(file.path(cache_dir, sprintf("df_res_%s.rds", code)))) {
  # load in the results of the SBC
  df.results = readRDS(file.path(cache_dir, sprintf("df_res_%s.rds", code)))
```

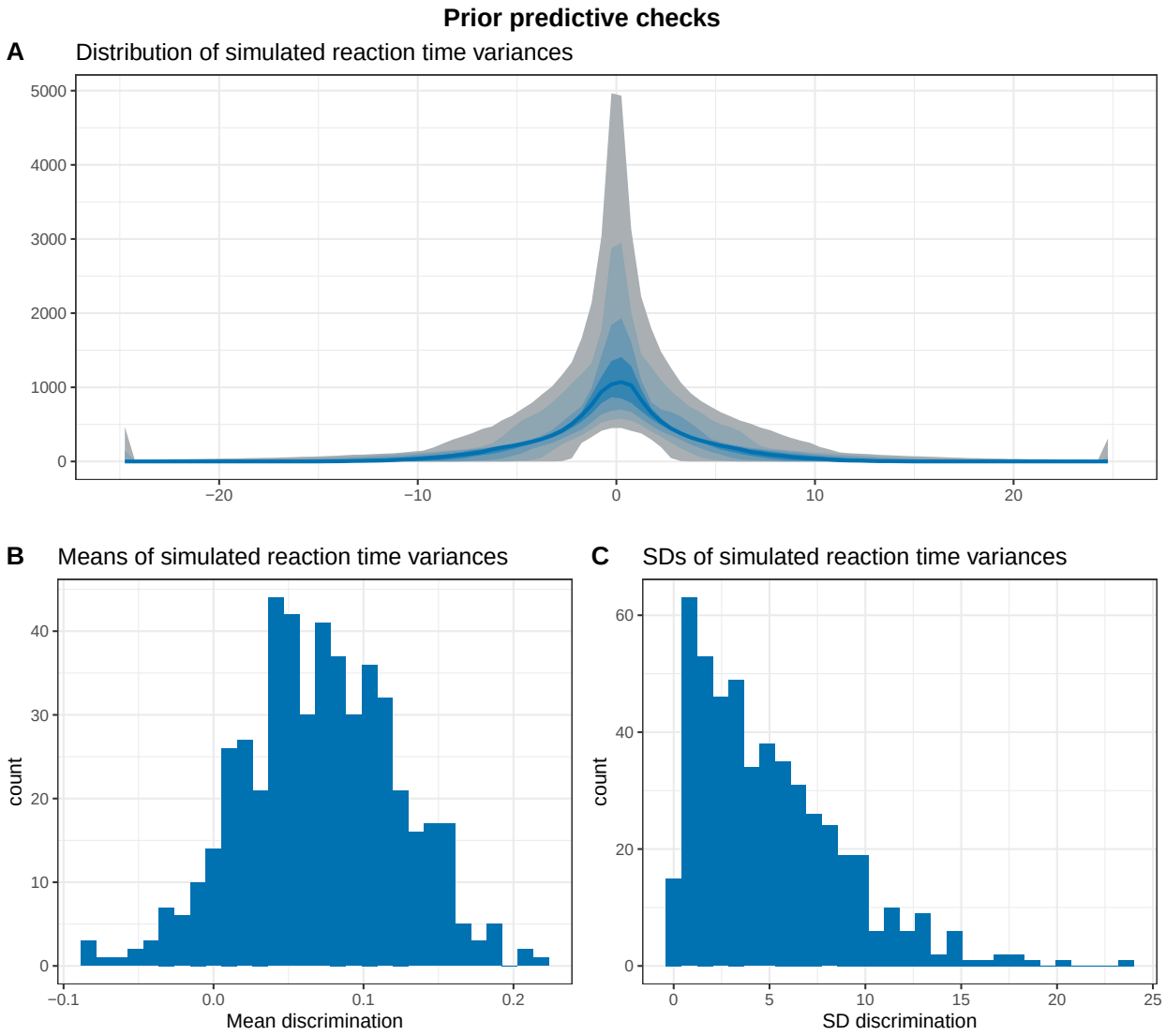
```
  df.backend = readRDS(file.path(cache_dir, sprintf("df_div_%s.rds", code)))
  dat = readRDS(file.path(cache_dir, sprintf("dat_%s.rds", code)))
} else {
  # stimulate some data > this was done in a separate script since it takes so long
  set.seed(2486)
  gen = SBC_generator_brms(f.pup, data = df, prior = priors,
   thin = 50, warmup = 10000, refresh = 2000,
   control = list(max_treedepth = max_depth),
   generate_lp = TRUE, init = 0.1)
  bck = SBC_backend_brms_from_generator(gen, chains = 4, thin = 1,
    warmup = warm, iter = iter)
  dat = generate_datasets(gen, nsim)
  saveRDS(dat, file.path(cache_dir, sprintf("dat_%s.rds", code)))
  # perform the SBC
  res = compute_SBC(dat,
        bck,
        cache_mode     = "results",
        cache_location = file.path(cache_dir, sprintf("res_%s", code)))
  df.results = res$stats
  df.backend = res$backend_diagnostics
  saveRDS(res$stats, file =
            file.path(cache_dir, paste0("df_res_", code, ".rds")))
  saveRDS(res$backend_diagnostics,
          file = file.path(cache_dir, paste0("df_div_", code, ".rds")))
}
```

We start by investigating the rhats and the number of divergent samples. This shows that 0 of 250 models had divergent samples; however, 82 simulations had at least one parameter that had an rhat of at least 1.05. Further inspection shows that even though we increased the maximum treedepth, sometimes the models hit it. Since we already have quite high iterations, maximum treedepth and warm up, we decide to continue, but look out for this issue in the final model.

Next, we can plot the simulated values to perform prior predictive checks.

```
if (!(file.exists(file.path(cache_dir, sprintf("dvfakemat_%s", code))))) {
  # create a matrix out of generated data
  dvname = gsub(" ", "", gsub("[\\|~].*", "", f.pup)[1])
  dvfakemat = matrix(NA, nrow(dat[['generated']][[1]]), length(dat[['generated']]))
  for (i in 1:length(dat[['generated']])) {
    dvfakemat[,i] = dat[['generated']][[i]][[dvname]]
  }
  saveRDS(dvfakemat, file.path(cache_dir, sprintf("dvfakemat_%s", code)))
} else {
  dvfakemat = readRDS(file.path(cache_dir, sprintf("dvfakemat_%s", code)))
}

# compute one histogram per simulated data-set
dvfakematH = dvfakemat
dvfakematH[dvfakematH >  25] =  25
dvfakematH[dvfakematH < -25] = -25
breaks = seq(min(dvfakematH, na.rm=T), max(dvfakematH, na.rm=T), length.out = 101)
binwidth = breaks[2] - breaks[1]
histmat = matrix(NA, ncol = dim(dvfakemat)[2], nrow = length(breaks)-1)
for (i in 1:dim(dvfakemat)[2]) {
  histmat[,i] = hist(dvfakematH[,i], breaks = breaks, plot = F)$counts
```

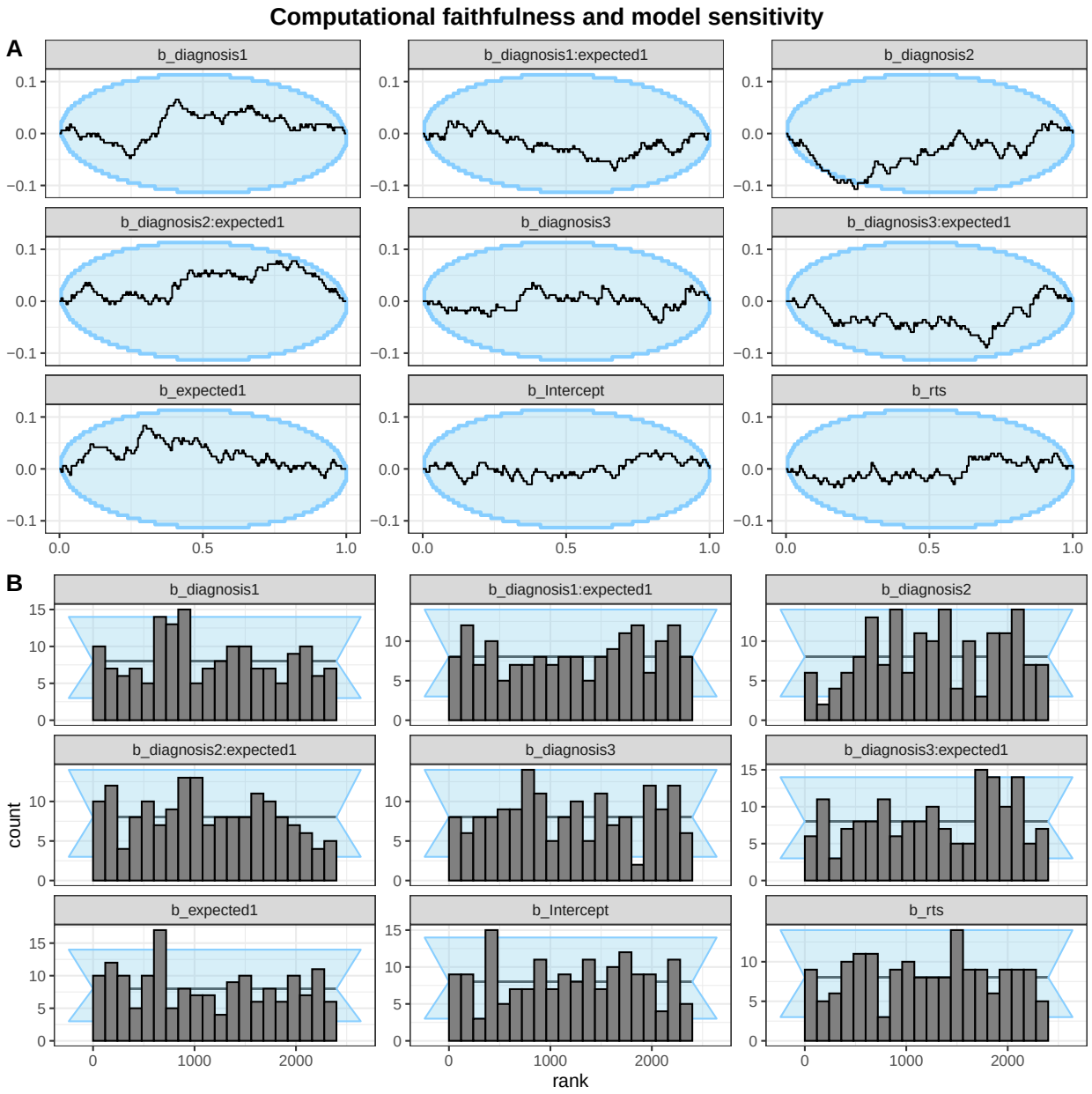
```
}
# for each bin, compute quantiles across histograms
probs = seq(0.1, 0.9, 0.1)
quantmat = as.data.frame(matrix(NA, nrow=dim(histmat)[1], ncol = length(probs)))
names(quantmat) = paste0("p", probs)
for (i in 1:dim(histmat)[1]) {
  quantmat[i,] = quantile(histmat[i,], p = probs)
}
quantmat$x = breaks[2:length(breaks)] - binwidth/2 # add bin mean
p1 = ggplot(data = quantmat, aes(x = x)) +
  geom_ribbon(aes(ymax = p0.9, ymin = p0.1), fill = c_light) +
  geom_ribbon(aes(ymax = p0.8, ymin = p0.2), fill = c_light_highlight) +
  geom_ribbon(aes(ymax = p0.7, ymin = p0.3), fill = c_mid) +
  geom_ribbon(aes(ymax = p0.6, ymin = p0.4), fill = c_mid_highlight) +
  geom_line(aes(y = p0.5), colour = c_dark, linewidth = 1) +
  labs(title = "Distribution of simulated reaction time variances", y = "", x = "") +
  theme_bw()

tmpM = apply(dvfakemat, 2, mean) # mean
tmpSD = apply(dvfakemat, 2, sd)
p2 = ggplot() +
  stat_bin(aes(x = tmpM), fill = c_dark)  +
  labs(x = "Mean discrimination", title = "Means of simulated reaction time variances") +
  theme_bw()
p3 = ggplot() +
  stat_bin(aes(x = tmpSD), fill = c_dark) +
  labs(x = "SD discrimination", title = "SDs of simulated reaction time variances") +
  theme_bw()
p = ggarrange(p1,
  ggarrange(p2, p3, ncol = 2, labels = c("B", "C")),
  nrow = 2, labels = "A")
annotate_figure(p, top = text_grob("Prior predictive checks", face = "bold", size = 14))
```



The simulated distributions fit our expectations. We go ahead with these priors and check the results of the SBC. We only plot the results from the models that had no divergence issues and also no large rhats.

```
# get simulation numbers with issues
mx_rnk = max(df.results$max_rank)
check = merge(df.results %>%
    group_by(sim_id) %>%
      summarise(rhat = max(rhat, na.rm = T),
                max_rank = mean(max_rank)) %>%
    filter(rhat >= 1.05 | max_rank != mx_rnk),
  df.backend %>% filter(n_divergent > 0), all = T)

# plot SBC with functions from the SBC package focusing on population-level parameters
a = 0.5
df.results.b = df.results %>%
  filter(substr(variable, 1, 2) == "b_") %>%
  filter(!(sim_id %in% check$sim_id))
p1 = plot_ecdf_diff(df.results.b) + theme_bw() + theme(legend.position = "none") +
  scale_x_continuous(breaks=scales::pretty_breaks(n = 3)) +
```

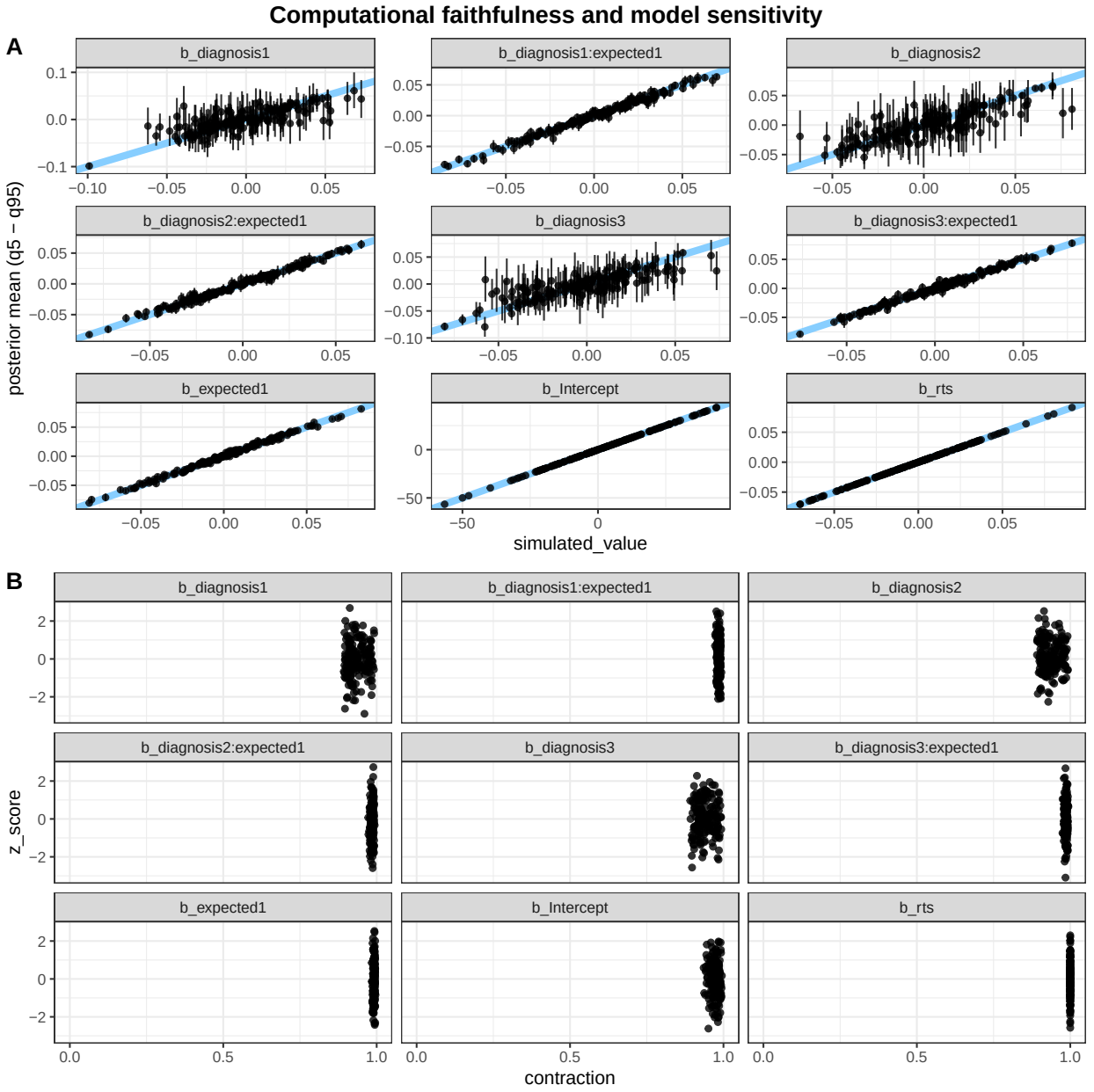
```
  scale_y_continuous(breaks=scales::pretty_breaks(n = 3))
p2 = plot_rank_hist(df.results.b, bins = 20) + theme_bw() +
  scale_x_continuous(breaks=scales::pretty_breaks(n = 3)) +
  scale_y_continuous(breaks=scales::pretty_breaks(n = 3))

p = ggarrange(p1, p2, labels = "AUTO", ncol = 1, nrow = 2)
annotate_figure(p, top = text_grob("Computational faithfulness and model sensitivity",
                                   face = "bold", size = 14))
```



There are some deviations, but nothing too concerning and no apparent biases.

```
p3 = plot_sim_estimated(df.results.b, alpha = .8) + theme_bw() +
  scale_x_continuous(breaks=scales::pretty_breaks(n = 3)) +
```

```
  scale_y_continuous(breaks=scales::pretty_breaks(n = 3))
p4 = plot_contraction(
  df.results.b,
  prior_sd = setNames(c(0.50, rep(0.25, length(unique(df.results.b$variable))-1)),
                      unique(df.results.b$variable))) +
  theme_bw() +
  scale_x_continuous(breaks=scales::pretty_breaks(n = 3)) +
  scale_y_continuous(breaks=scales::pretty_breaks(n = 3))

p = ggarrange(p3, p4, labels = "AUTO", ncol = 1, nrow = 2)
annotate_figure(p, top = text_grob("Computational faithfulness and model sensitivity",
                                   face = "bold", size = 14))
```



All looks good, the priors could have even been a bit narrower, but we find this to be acceptable.

Posterior predictive checks

As the next step, we fit the model, check whether there are divergence or rhat issues, and then check whether the chains have converged.

```
# fit the final model
set.seed(2684)
m.pup = brm(f.pup,
            df, prior = priors,
            iter = iter, warmup = warm,
            init = 0.1,
            control = list(max_treedepth = max_depth),
            backend = "cmdstanr", threads = threading(8),
            file = file.path(brms_dir, "m_pup"),
            save_pars = save_pars(all = TRUE)
            )
rstan::check_divergences(m.pup$fit)
```

```
## 0 of 18000 iterations ended with a divergence.
```

```
rstan::check_energy(m.pup$fit)
```

```
## E-BFMI indicated no pathological behavior.
```

```
# check_treedepth takes treedepth 10 which we already increased,
# so we check this manually
sparams  = rstan::get_sampler_params(m.pup$fit, inc_warmup = FALSE)
exceedtd = do.call(rbind, sparams)[, "treedepth__"] >= max_depth
message(
      sprintf('%s of %s iterations saturated the maximum tree depth of %s (%.3f%%).',
              sum(exceedtd),
              length(exceedtd),
              max_depth,
              100 * sum(exceedtd) / length(exceedtd))
    )
```

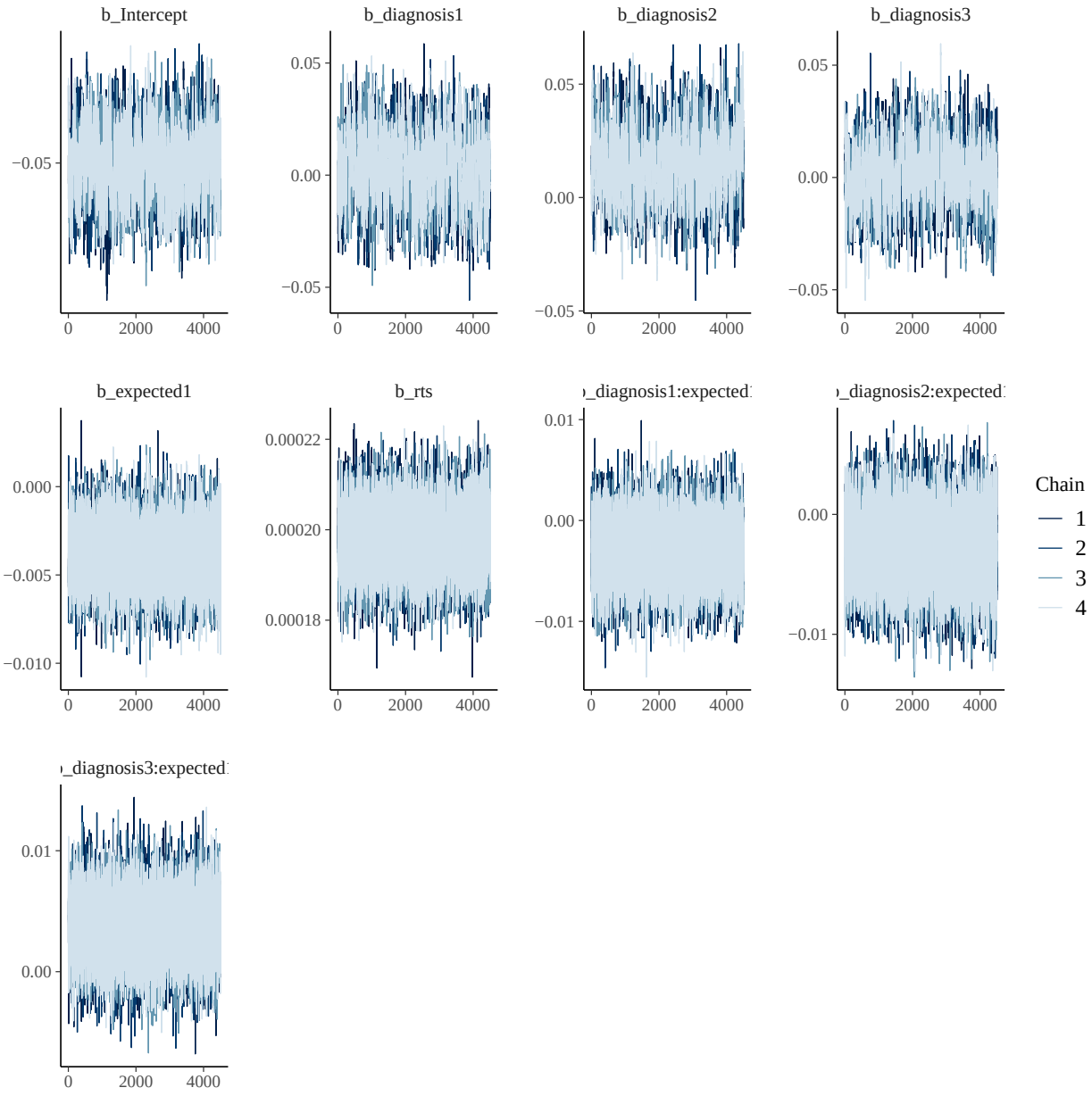
```
## 0 of 18000 iterations saturated the maximum tree depth of 15 (0.000%).
```

```
# check that rhats are below 1.01
sum(brms::rhat(m.pup) >= 1.01, na.rm = T)
```

```
## [1] 0
```

```
# check the trace plots
post.draws = as_draws_df(m.pup)
mcmc_trace(post.draws, regex_pars = "^b_",
           facet_args = list(ncol = 4)) +
  scale_x_continuous(breaks=scales::pretty_breaks(n = 3)) +
  scale_y_continuous(breaks=scales::pretty_breaks(n = 3))
```

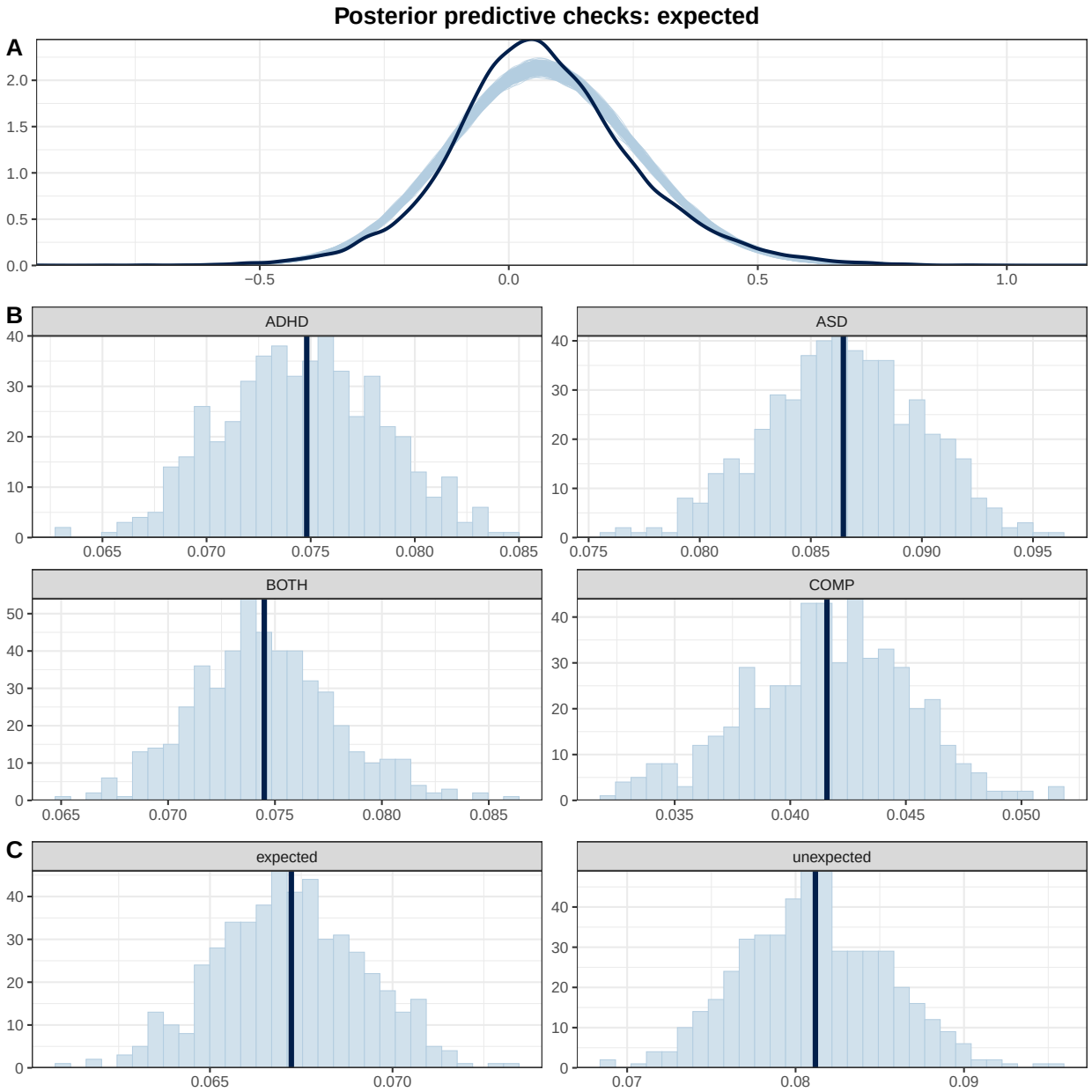
```
## Scale for x is already present.
## Adding another scale for x, which will replace the existing scale.
```



This model has no pathological behaviour with E-BFMI, no divergent sample and no rhats that are higher or equal to 1.01. Furthermore, there are no iterations where the maximum treedepth is exceeded. Therefore, we go ahead and perform our posterior predictive checks.

```
# get posterior predictions
post.pred = posterior_predict(m.pup, ndraws = nsim)

# check the fit of the predicted data compared to the real data
p1 = pp_check(m.pup, ndraws = nsim) +
  theme_bw() + theme(legend.position = "none")

# distributions of means compared to the real values per group
p2 = ppc_stat_grouped(df$rel_pupil, post.pred, df$diagnosis) +
  theme_bw() + theme(legend.position = "none")
```

```
# distributions of means compared to the real values per group
p3 = ppc_stat_grouped(df$rel_pupil, post.pred, df$expected) +
  theme_bw() + theme(legend.position = "none")

p = ggarrange(p1, p2, p3, heights = c(1,2,1),
          nrow = 3, ncol = 1, labels = "AUTO")
annotate_figure(p, top = text_grob("Posterior predictive checks: expected",
                                   face = "bold", size = 14))
```



The simulated data based on the model fits well with the real data.

Inferences

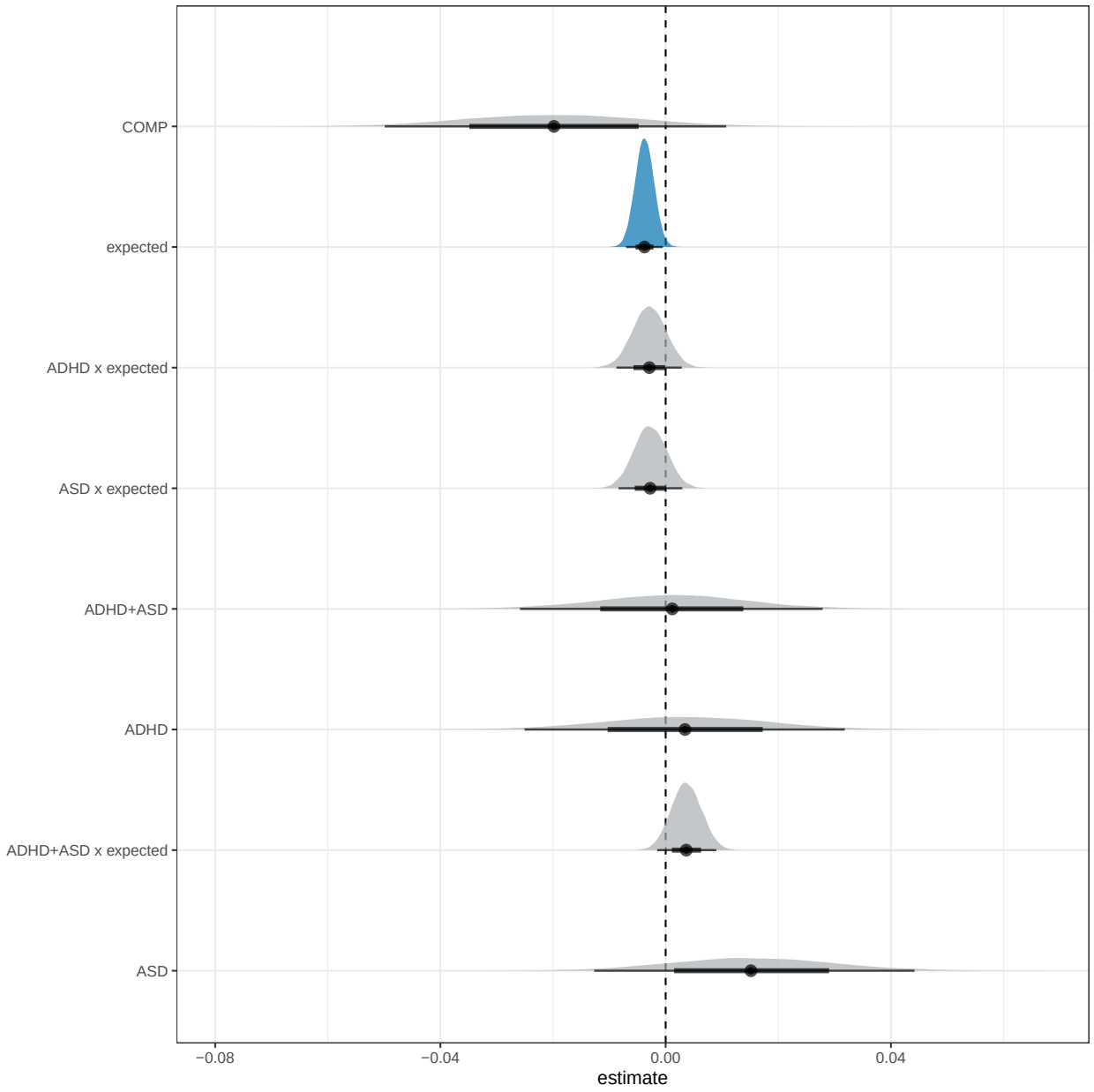
Now that we are convinced that we can trust our model, we have a look at its estimate and use the hypothesis function to assess our hypothesis and explore the data.

```
# print a summary
summary(m.pup)
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: rel_pupil ~ diagnosis * expected + rts + (1 | subID)
##    Data: df (Number of observations: 19618)
##   Draws: 4 chains, each with iter = 6000; warmup = 1500; thin = 1;
##          total post-warmup draws = 18000
##
## Multilevel Hyperparameters:
## ~subID (Number of levels: 76)
##               Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(Intercept)     0.07      0.01     0.06     0.09 1.00     1747     3835
##
## Regression Coefficients:
##                      Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS
## Intercept               -0.05      0.01    -0.07    -0.03 1.00     1087
## diagnosis1               0.00      0.01    -0.03     0.03 1.00     1155
## diagnosis2               0.02      0.01    -0.01     0.04 1.00      965
## diagnosis3               0.00      0.01    -0.03     0.03 1.00      943
## expected1               -0.00      0.00    -0.01    -0.00 1.00    36207
## rts                      0.00      0.00     0.00     0.00 1.00    19660
## diagnosis1:expected1    -0.00      0.00    -0.01     0.00 1.00    17173
## diagnosis2:expected1    -0.00      0.00    -0.01     0.00 1.00    17966
## diagnosis3:expected1     0.00      0.00    -0.00     0.01 1.00    17886
##                      Tail_ESS
## Intercept                2694
## diagnosis1               2360
## diagnosis2               2246
## diagnosis3               1966
## expected1               12748
## rts                     14488
## diagnosis1:expected1    13963
## diagnosis2:expected1    14535
## diagnosis3:expected1    12762
##
## Further Distributional Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma     0.17      0.00     0.17     0.17 1.00    15694    13282
##
## 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).
```

```
# plot the posterior distributions
post.draws %>%
  select(starts_with("b_")) %>%
  select(-b_rts) %>%
  mutate(
    b_COMP         = - b_diagnosis1 - b_diagnosis2 - b_diagnosis3
```

```
) %>%
select(starts_with("b_")) %>%
pivot_longer(cols = starts_with("b_"), names_to = "coef", values_to = "estimate") %>%
subset(!startsWith(coef, "b_Int")) %>%
mutate(
  coef = substr(coef, 3, nchar(coef)),
  coef = str_replace_all(coef, ":", " x "),
  coef = str_replace_all(coef, "diagnosis1", "ADHD"),
  coef = str_replace_all(coef, "diagnosis2", "ASD"),
  coef = str_replace_all(coef, "diagnosis3", "ADHD+ASD"),
  coef = str_replace_all(coef, "expected1", "expected"),
  coef = fct_reorder(coef, desc(estimate))
) %>%
group_by(coef) %>%
mutate(
  cred = case_when(
    (mean(estimate) < 0 & quantile(estimate, probs = 0.975) < 0) |
      (mean(estimate) > 0 & quantile(estimate, probs = 0.025) > 0) ~ "credible",
    T ~ "not credible"
  )
) %>% ungroup() %>%
ggplot(aes(x = estimate, y = coef, fill = cred)) +
geom_vline(xintercept = 0, linetype = 'dashed') +
ggdist::stat_halfeye(alpha = 0.7) + ylab(NULL) + theme_bw() +
scale_fill_manual(values = c(c_dark, c_light)) + theme(legend.position = "none")
```




```
# get the design matrix to figure out how to set the contrasts
df.des = cbind(df, model.matrix(~ diagnosis * expected, data = df)) %>%
  select(starts_with("diagn") | starts_with("expect")) %>% distinct() %>%
  relocate(diagnosis, expected)

# explore expectancy effect
e.exp = hypothesis(m.pup, "0 > 2*expected1", alpha = 0.025)
e.exp
```

```
## Hypothesis Tests for class b:
##              Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(2*expected1) > 0     0.01         0        0     0.01         79
##   Post.Prob Star
## 1      0.99    *
```

```
## ---
## 'CI': 95%-CI for one-sided and 97.5%-CI for two-sided hypotheses.
## '*': For one-sided hypotheses, the posterior probability exceeds 97.5%;
## for two-sided hypotheses, the value tested against lies outside the 97.5%-CI.
## Posterior probabilities of point hypotheses assume equal prior probabilities.
```

```
# H2a: COMP(unexp - exp) > ASD(unexp - exp)
t(df.des %>%
    filter((diagnosis == "ASD" | diagnosis == "COMP")) %>%
    group_by(diagnosis) %>%
    summarise(across(where(is.numeric), ~ diff(.x))) %>% # unexpected - expected
    arrange(diagnosis) %>%
    select(where(is.numeric)) %>%
    map_df(~ diff(.x))) # COMP(unexpected - expected) - ASD(unexpected - expected)
```

```
##                       [,1]
## diagnosis1               0
## diagnosis2               0
## diagnosis3               0
## diagnosis1:expected1     2
## diagnosis2:expected1     4
## diagnosis3:expected1     2
## expected1                0
```

```
h2a = hypothesis(m.pup, "0 <
                 2*(diagnosis1:expected1 + 2*diagnosis2:expected1 + diagnosis3:expected1)")
h2a
```

```
## Hypothesis Tests for class b:
##                 Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(2*(diagnosis... < 0     0.01      0.01    -0.01     0.02       0.19
##   Post.Prob Star
## 1      0.16
## ---
## 'CI': 90%-CI for one-sided and 95%-CI for two-sided hypotheses.
## '*': For one-sided hypotheses, the posterior probability exceeds 95%;
## for two-sided hypotheses, the value tested against lies outside the 95%-CI.
## Posterior probabilities of point hypotheses assume equal prior probabilities.
```

```
# H2b: COMP(unexp - exp) != ADHD(unexp - exp)
h2b = hypothesis(m.pup, "0 >
                 2*(2*diagnosis1:expected1 + diagnosis2:expected1 + diagnosis3:expected1)",
                 alpha = 0.025)
h2b
```

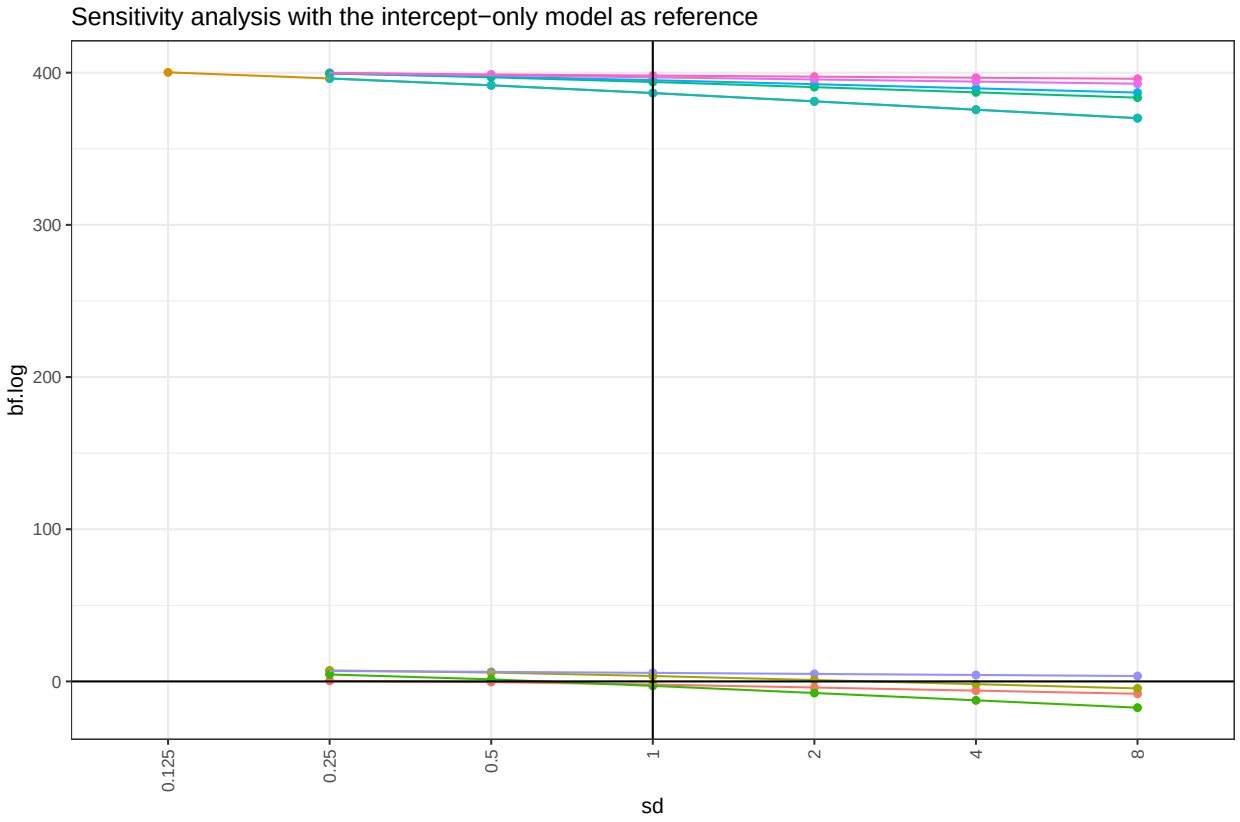
```
## Hypothesis Tests for class b:
##                 Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(2*(2*diagnos... > 0     0.01      0.01    -0.01     0.03       5.34
##   Post.Prob Star
## 1      0.84
## ---
## 'CI': 95%-CI for one-sided and 97.5%-CI for two-sided hypotheses.
## '*': For one-sided hypotheses, the posterior probability exceeds 97.5%;
## for two-sided hypotheses, the value tested against lies outside the 97.5%-CI.
## Posterior probabilities of point hypotheses assume equal prior probabilities.
```

[!MISSING]

Bayes factor

```
# set the directory in which to save results
sense_dir = file.path(getwd(), "_brms_sens_cache")
main.code = "pal_pup"

# rerun the model with more iterations for bridgesampling
set.seed(5544)
m.orig = brm(f.pup,
             df, prior = priors,
             iter = 40000, warmup = 10000,
             init = 0.1,
             control = list(max_treedepth = max_depth),
             backend = "cmdstanr", threads = threading(8),
             file = file.path(brms_dir, "m_pup_bf"),
             save_pars = save_pars(all = TRUE), silent = 2
             )

# check if they have been run already
pr.descriptions = ls.priors
if (file.exists(file.path(sense_dir,sprintf("df_%s_bf.csv", main.code)))) {
  pr.done = read_csv(file.path(sense_dir,
                               sprintf("df_%s_bf.csv", main.code)),
                     show_col_types = F) %>%
    select(priors) %>% distinct()
  pr.descriptions = pr.descriptions[!(pr.descriptions %in% pr.done$priors)]
}

pr.descriptions = c()

# loop through the priors that have not been used before
for (pr.desc in pr.descriptions) {
  tryCatch({
    # use function to compute BF with the described priors
    bf_sens_2int1cov(m.orig, "diagnosis", "expected", "rts",
                 pr.desc,
                 main.code, # prefix for all models and MLL
                 file.path("/home/emba/pCloudDrive/data/NEVIA/logfiles",
                           "log_PAL_bf.txt"), # log file
                 sense_dir # where to save the models and MLL
    )
  },
  error = function(err) {
    message(sprintf("Error for %s: %s", pr.desc, err))
  }
  )
}

df.pal.bf = read_csv(file.path(sense_dir,
                               sprintf("df_%s_bf.csv", main.code)),
                     show_col_types = F)

# check the sensitivity analysis result per model
df.pal.bf %>%
```

```
  filter(`population-level` != "1") %>%
  mutate(
    sd = as.factor(case_when(
      priors == "chosen" ~ "1",
      substr(priors, 1, 3) == "sdx" ~ gsub("sdx", "", priors),
      T ~ priors)
    ),
    order = case_when(
      priors == "chosen" ~ 1,
      substr(priors, 1, 3) == "sdx" ~ as.numeric(gsub("sdx", "", priors)),
      T ~ 999),
    sd = fct_reorder(sd, order)
  ) %>%
  ggplot(aes(y = bf.log,
             x = sd,
             group = `population-level`,
             colour = `population-level`)) +
  geom_point() +
  geom_line() +
  geom_vline(xintercept = "1") +
  geom_hline(yintercept = 0) +
  ggtitle("Sensitivity analysis with the intercept-only model as reference") +
  #scale_colour_manual(values = custom.col) +
  theme_bw() +
  theme(legend.position = "none",
        axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1))
```



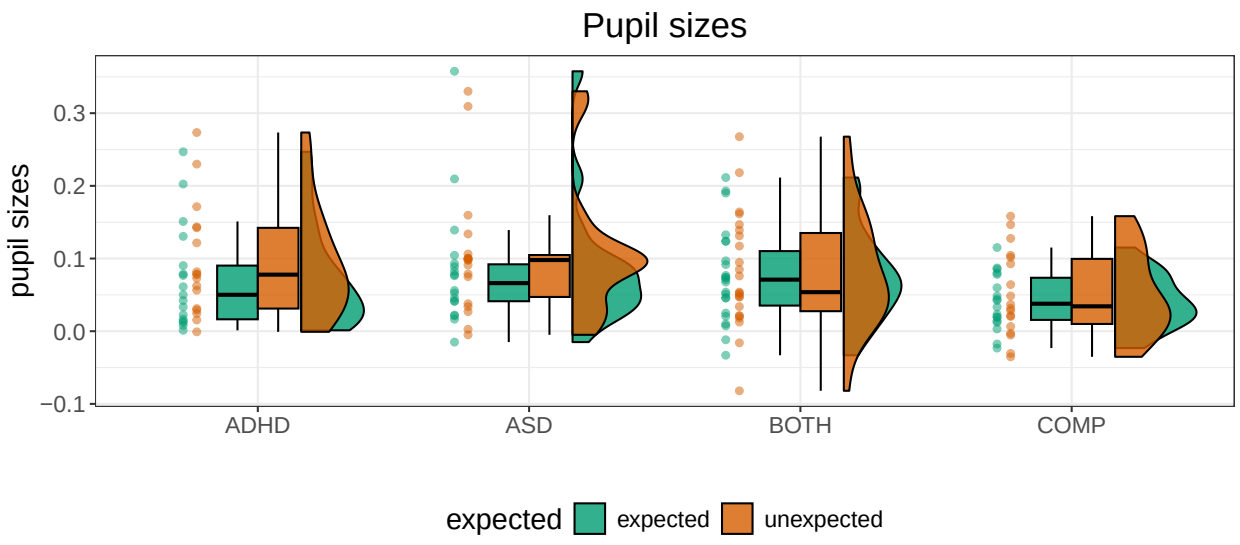

```
# print the results for the chosen priors
kable(df.pal.bf %>%
        filter(priors == "chosen" & `population-level` != "1") %>%
        arrange(desc(bf.log)) %>%
        mutate(
          bf.log.diff = bf.log - lead(bf.log),
          bf.int = interpret_bf(bf.log.diff, log = T)
        ))
```

population-level	bf.log	priors	bf.log.diff	bf.int
rts	398.1030	chosen	1.1782	moderate evidence in favour of
expected + rts	396.9248	chosen	1.8702	moderate evidence in favour of
diagnosis + rts	395.0546	chosen	1.1584	moderate evidence in favour of
diagnosis + expected + rts	393.8962	chosen	7.2780	extreme evidence in favour of
diagnosis * expected + rts	386.6182	chosen	0.0022	anecdotal evidence in favour of
diagnosis + expected + rts + diagnosis:expected	386.6160	chosen	380.9993	extreme evidence in favour of
expected	5.6167	chosen	2.0514	moderate evidence in favour of
diagnosis + expected	3.5653	chosen	5.6262	extreme evidence in favour of
diagnosis	-2.0609	chosen	0.8625	anecdotal evidence in favour of
diagnosis + expected + diagnosis:expected	-2.9234	chosen	NA	

population-level	bf.log	priors	bf.log.diff	bf.int

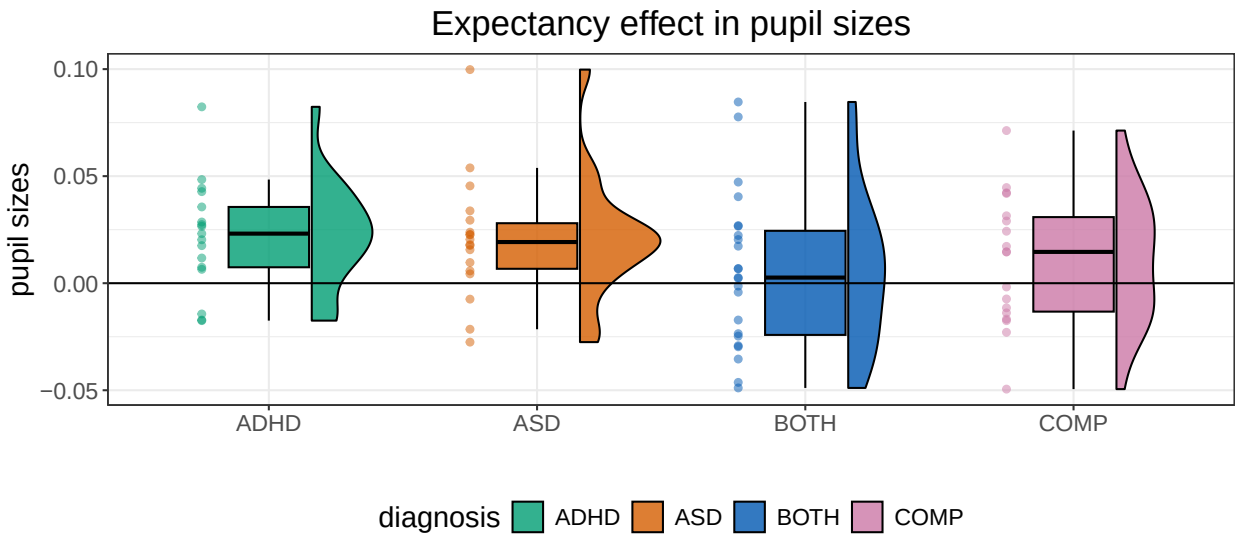
Plots

```
# rain cloud plot
df %>%
  group_by(subID, diagnosis, expected) %>%
  summarise(
    rel_pupil = mean(rel_pupil)
  ) %>%
  ggplot(aes(diagnosis, rel_pupil, fill = expected, colour = expected)) + #
  geom_rain(rain.side = 'r',
boxplot.args = list(color = "black", outlier.shape = NA, show.legend = FALSE, alpha = .8),
violin.args = list(color = "black", outlier.shape = NA, alpha = .8),
boxplot.args.pos = list(
  position = ggpp::position_dodgenudge(x = 0, width = 0.3), width = 0.3
),
point.args = list(show_guide = FALSE, alpha = .5),
violin.args.pos = list(
  width = 0.6, position = position_nudge(x = 0.16)),
point.args.pos = list(position = ggpp::position_dodgenudge(x = -0.25, width = 0.1))) +
  scale_fill_manual(values = custom.col) +
  scale_color_manual(values = custom.col) +
  labs(title = "Pupil sizes", x = "", y = "pupil sizes") +
  theme_bw() +
  theme(legend.position = "bottom", plot.title = element_text(hjust = 0.5),
        legend.direction = "horizontal", text = element_text(size = 15))
```




```
df %>%
  group_by(subID, diagnosis, expected) %>%
  summarise(
    rel_pupil = mean(rel_pupil)
  ) %>%
  group_by(subID, diagnosis) %>%
```

```
  summarise(
    rel_pupil = diff(rel_pupil)
  ) %>%
  ggplot(aes(diagnosis, rel_pupil, fill = diagnosis, colour = diagnosis)) + #
  geom_rain(rain.side = 'r',
boxplot.args = list(color = "black", outlier.shape = NA, show.legend = FALSE, alpha = .8),
violin.args = list(color = "black", outlier.shape = NA, alpha = .8),
boxplot.args.pos = list(
  position = ggpp::position_dodgenudge(x = 0, width = 0.3), width = 0.3
),
point.args = list(show_guide = FALSE, alpha = .5),
violin.args.pos = list(
  width = 0.6, position = position_nudge(x = 0.16)),
point.args.pos = list(position = ggpp::position_dodgenudge(x = -0.25, width = 0.1))) +
  geom_hline(yintercept = 0) +
  scale_fill_manual(values = custom.col) +
  scale_color_manual(values = custom.col) +
  labs(title = "Expectancy effect in pupil sizes", x = "", y = "pupil sizes") +
  theme_bw() +
  theme(legend.position = "bottom", plot.title = element_text(hjust = 0.5),
        legend.direction = "horizontal", text = element_text(size = 15))
```




```
# two-way interaction
df %>%
  group_by(subID, diagnosis, expected) %>%
  summarise(
    rel_pupil = mean(rel_pupil)
  ) %>%
  group_by(diagnosis, expected) %>%
  summarise(
    rel_pupil.mn = mean(rel_pupil),
    rel_pupil.se = sd(rel_pupil) / sqrt(n())
  ) %>%
  ggplot(aes(y = rel_pupil.mn, x = diagnosis, group = expected, colour = expected)) +
  geom_line(position = position_dodge(0.4), linewidth = 1) +
```

```
geom_errorbar(aes(ymin = rel_pupil.mn - rel_pupil.se,
                  ymax = rel_pupil.mn + rel_pupil.se), linewidth = 1, width = 0.5,
              position = position_dodge(0.4)) +
geom_point(position = position_dodge(0.4), size = 5) +
labs (x = "diagnosis", y = "pupil size") +
ggtitle("Diagnosis x expectancy") +
scale_fill_manual(values = custom.col) +
scale_color_manual(values = custom.col) +
theme_bw() +
theme(legend.position = "bottom", plot.title = element_text(hjust = 0.5))
```