S4: behavioural, HGF-based analysis with brms

I. S. Plank

2025-03-21

S4.1 Introduction

This R Markdown script analyses data from the PAL (probabilistic associative learning) task of the EMBA project. HGF parameters were extrated based on the subject-specific reaction times beforehand in MATLAB.

Some general settings

```
# number of simulations
nsim = 500

# set number of iterations and warmup for models
iter = 3000
warm = 1000

# 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] "BayesFactor version 0.9.12.4.7"
## [1] "effectsize version 0.8.9"
## [1] "bayestestR version 0.15.0"
```

Preparation

First, we load the parameters from the winning model.

```
# get HGF parameters
df.hgf = read_csv(file.path("HGF_results/main", "eHGF-C_results.csv")) %>%
  mutate_if(is.character, as.factor)

# get belief state trajectories
df.trj = read_csv(file.path("HGF_results/main", "eHGF-C_traj.csv"))

# extract the absolute changes in learning rate for the phases
df.upd = df.trj %>%
  select(subID, diagnosis, trl, alpha2, alpha3) %>%
  mutate(
    phase = case_when(
      trl < 73  ~ "pre",
      trl > 264 ~ "post",
      trl < 145 ~ "vol1",
      trl > 192 ~ "vol2"
    )
  ) %>%
  drop_na() %>%
  group_by(subID, diagnosis, phase) %>%
  summarise(
    alpha2 = median(alpha2),
    alpha3 = median(alpha3)
  ) %>%
  pivot_wider(names_from = phase, id_cols = c(subID, diagnosis), values_from = starts_with("alpha")) %>%
  group_by(subID, diagnosis) %>%
  summarise(
    alpha2_pre2vol  = abs(alpha2_pre  - alpha2_vol1),
    alpha2_vol2post = abs(alpha2_post - alpha2_vol2),
    alpha3_pre2vol  = abs(alpha3_pre  - alpha3_vol1),
    alpha3_vol2post = abs(alpha3_post - alpha3_vol2)
  ) %>%
  pivot_longer(cols = starts_with("alpha")) %>%
  separate(name, into = c("level", "change")) %>%
  mutate_if(is.character, as.factor)

# check whether there are LME differences between the diagnostic groups
kable(df.hgf %>% group_by(diagnosis) %>% shapiro_test(LME)) # all normally distributed
```

diagnosis	variable	statistic	p
ADHD	LME	0.9624505	0.5403480
ASD	LME	0.9654280	0.5810545
BOTH	LME	0.9725506	0.7497527
COMP	LME	0.9722848	0.7633392

```
aov.lme = anovaBF(LME ~ diagnosis, data = df.hgf)
aov.lme@bayesFactor
```

```
##                  bf        error                      time          code
## diagnosis -1.108477 1.324705e-06 Fri Mar 21 15:28:55 2025 3126e74777cf3
```

There is moderate evidence against a difference in LME between diagnostic groups. This suggests that the HGF model fit comparably well to the subjects of the different groups. Therefore, we move on to analyse its parameters.

For this specific model, the following observation model was used:

$$\log RT = \beta_0 + \beta_1 \times surprise_{Shannon} + \beta_2 \times pwPE + \beta_3 \times volatility_{phasic}$$

This model was created based on the model used in Lawson et al. (2021); however, in our case, it was used with the enhanced HGF as the perception model. Next, we use sum contrast coding for all of our categorical predictors.

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

```
##      [,1] [,2] [,3]
## ADHD    1    0    0
## ASD     0    1    0
## BOTH    0    0    1
## COMP   -1   -1   -1
```

```
contrasts(df.upd$diagnosis) = contr.sum(4)
contrasts(df.upd$diagnosis)
```

```
##      [,1] [,2] [,3]
## BOTH    1    0    0
## ADHD    0    1    0
## ASD     0    0    1
## COMP   -1   -1   -1
```

```
contrasts(df.upd$change) = contr.sum(2)
contrasts(df.upd$change)
```

```
##          [,1]
## pre2vol     1
## vol2post   -1
```

```
contrasts(df.upd$level) = contr.sum(2)
contrasts(df.upd$level)
```

```
##        [,1]
## alpha2    1
## alpha3   -1
```

S4.2 H3a: phasic volatility

Model setup

```
# code for filenames
code = "PAL_vol"

# model formula
f.vol = brms::bf( be3 ~ diagnosis )

# set weakly informative priors
priors = c(
```

```
  prior(normal(0, 4),  class = Intercept),
  prior(normal(0, 0.50),  class = sigma),
  prior(normal(0, 0.25),  class = b)
)

# change Intercept based on empirical priors used in the HGF model
priors = priors %>%
  mutate(
    prior = if_else(
      class == "Intercept",
      gsub("\\(.*,", paste0("(", mean(df.hgf$be3mu), ", "), prior), prior),
    prior = if_else(
      class == "Intercept",
      gsub(" .*\\)", paste0(" ", mean(df.hgf$be3sa), ")"), prior), prior)
  )

kable(priors)
```

prior	class	coef	group	resp	dpar	nlpar	lb	ub	source
normal(-0.0574, 0.3976)	Intercept						NA	NA	user
normal(0, 0.5)	sigma						NA	NA	user
normal(0, 0.25)	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
  set.seed(2486)
  gen = SBC_generator_brms(f.vol, data = df.hgf, prior = priors,
   thin = 50, warmup = 10000, refresh = 2000,
   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(df.results, file = file.path(cache_dir, paste0("df_res_", code, ".rds")))
  saveRDS(df.backend,
```

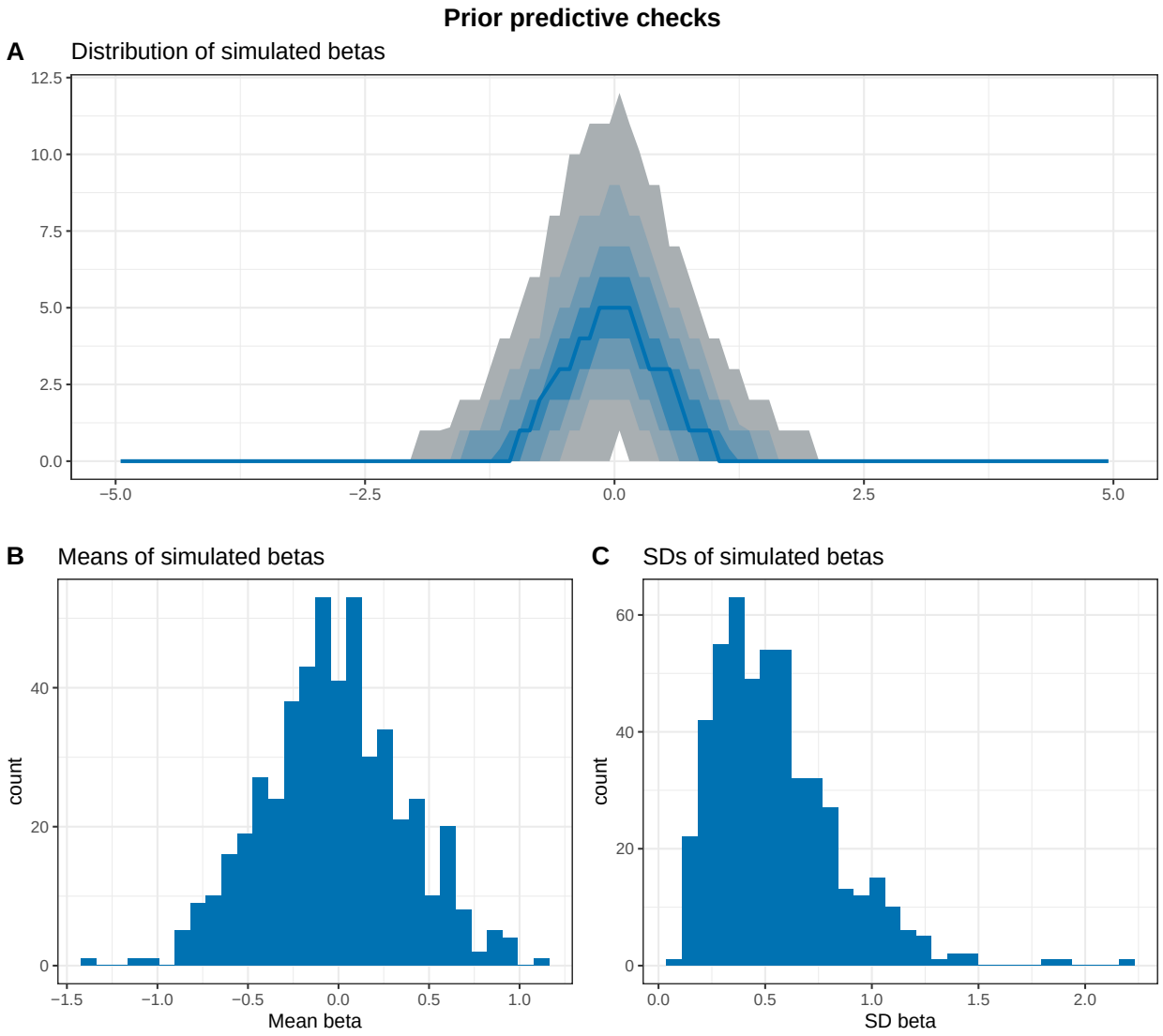
```
          file = file.path(cache_dir, paste0("df_div_", code, ".rds")))
}
```

We start by investigating the rhats and the number of divergent samples. This shows that only 0 of 500 simulations had at least one parameter that had an rhat of at least 1.05 and 0 models had divergent samples. This suggests that this model performs well.

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.vol)[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 > 5]  = 5
dvfakematH[dvfakematH < -5] = -5
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 betas", 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 beta", title = "Means of simulated betas") +
  theme_bw()
p3 = ggplot() +
```

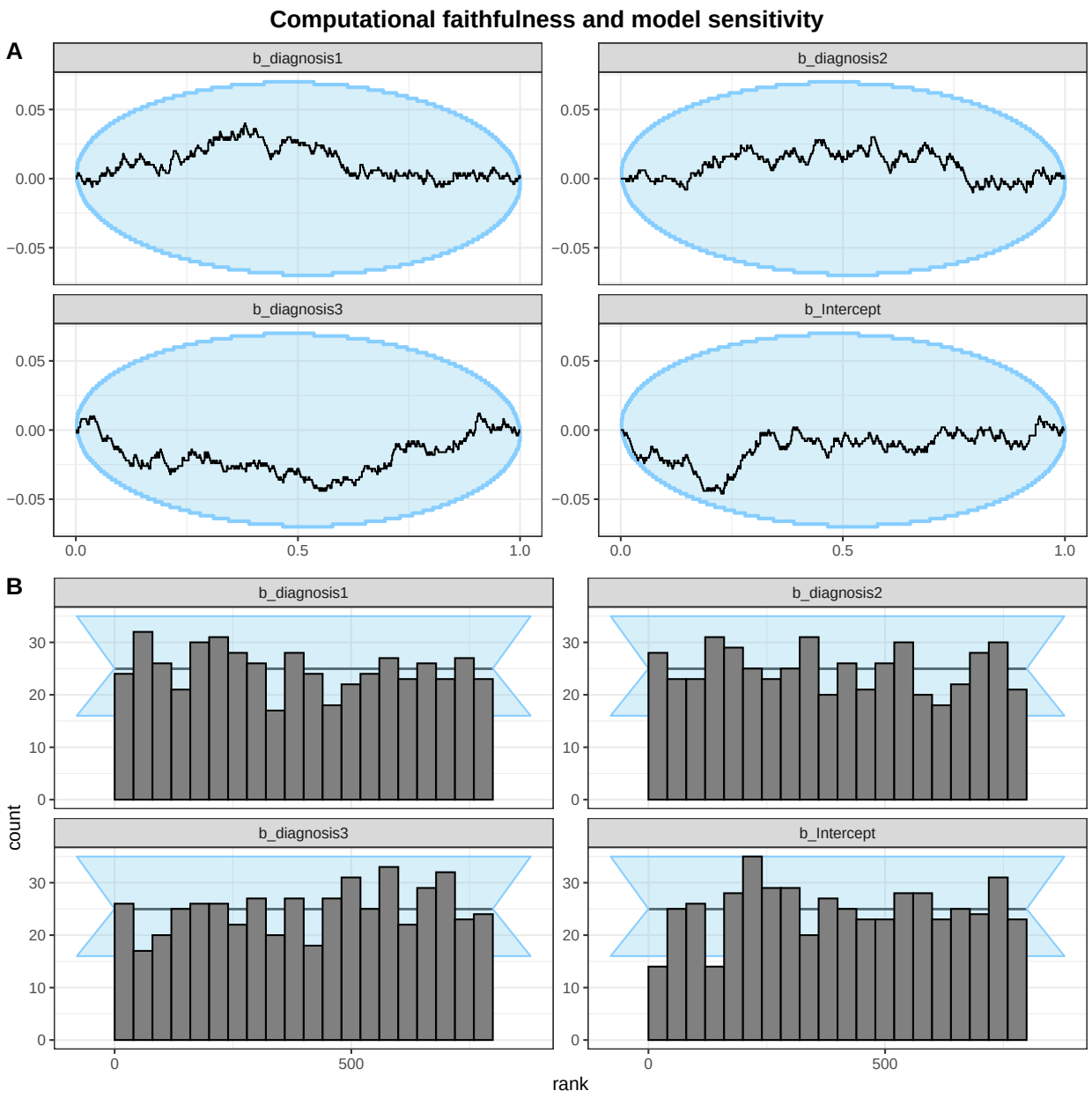
```
  stat_bin(aes(x = tmpSD), fill = c_dark) +
  labs(x = "SD beta", title = "SDs of simulated betas") +
  theme_bw()
p = ggarrange(p1,
  ggarrange(p2, p3, ncol = 2, labels = c("B", "C")),
  nrow = 2, labels = "A")
annotate_figure(p, top = text_grob("Prior predictive checks", face = "bold", size = 14))
```



This all looks good and 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.

```
# 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),
```

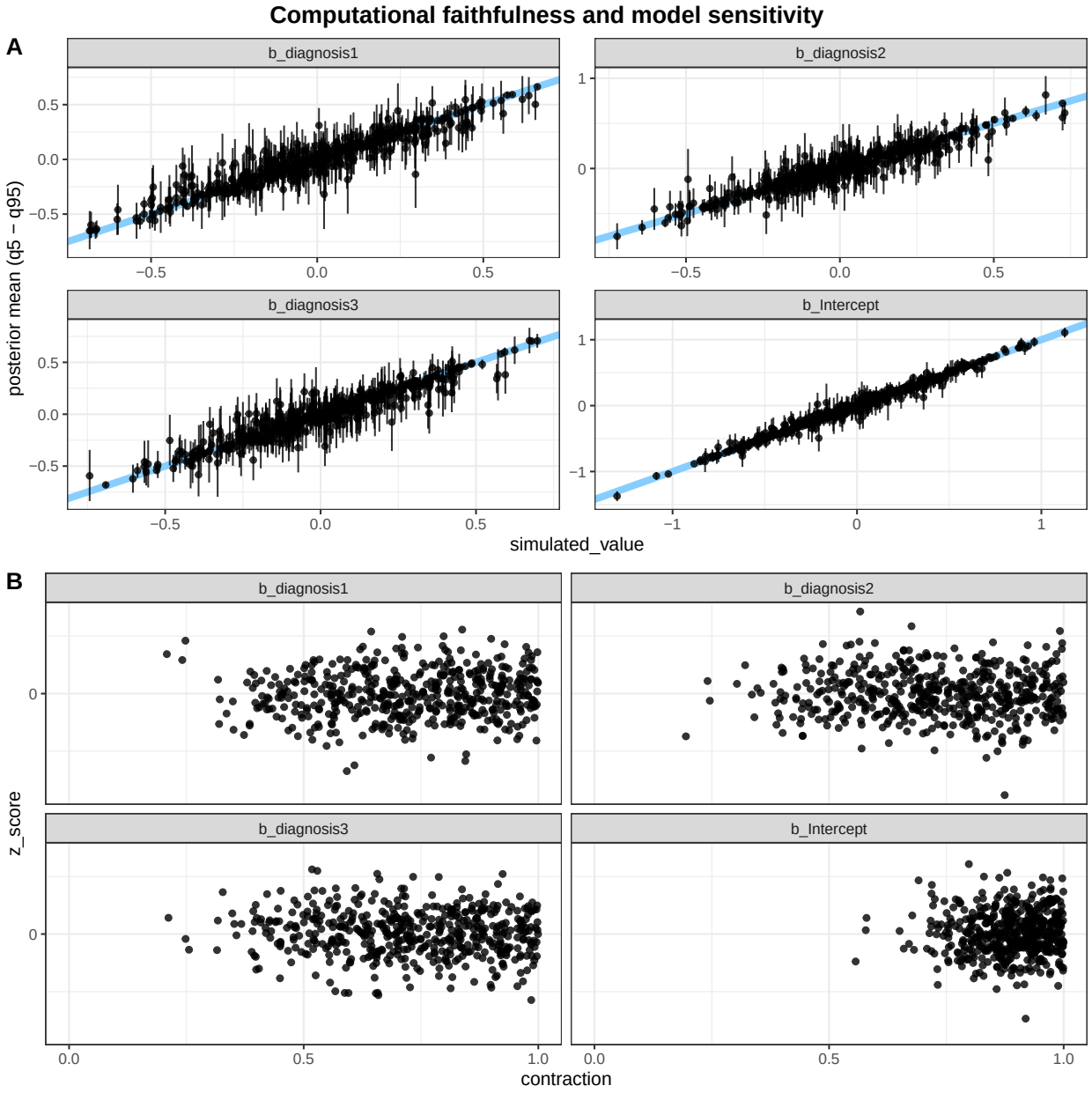
```
  df.backend %>% filter(n_divergent > 0), all = T)

# 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))
```



All looks good and we judge this to be acceptable as we cannot identify clear bias patterns as described in the information accompanying the SBC package: https://hyunjimoon.github.io/SBC/articles/rank_visualizations.html

```
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(mean(df.hgf$be3sa),
                                   rep(0.25,
                                       length(unique(df.results.b$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))
```



Then, we investigated the relationship between the simulated true parameters and the posterior estimates. Most parameters were recovered successfully within an uncertainty interval of alpha = 0.05. Last, z-score and the posterior contraction of our population-level predictors look good as well.

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.vol = brm(f.vol,
            df.hgf, prior = priors,
            iter = iter, warmup = warm,
            backend = "cmdstanr", threads = threading(8),
            file = file.path(brms_dir, "m_hgf_vol"),
            save_pars = save_pars(all = TRUE)
            )
rstan::check_hmc_diagnostics(m.vol$fit)
```

```
##
## Divergences:
```

```
## 0 of 8000 iterations ended with a divergence.
```

```
##
## Tree depth:
```

```
## 0 of 8000 iterations saturated the maximum tree depth of 10.
```

```
##
## Energy:
```

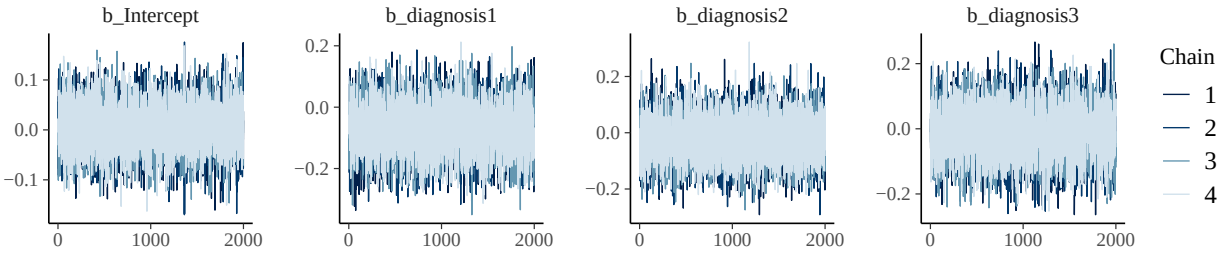
```
## E-BFMI indicated no pathological behavior.
```

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

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

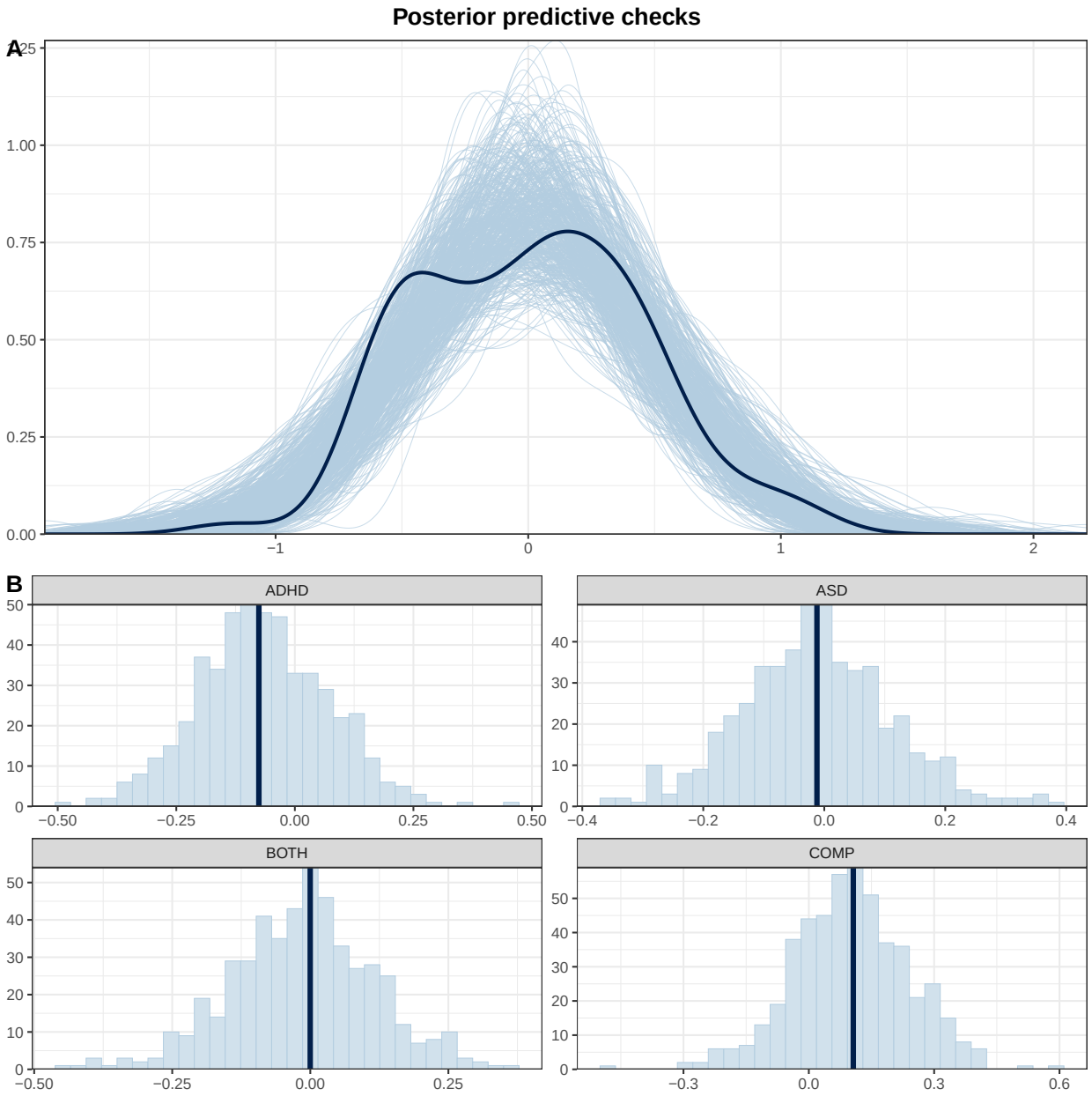
```
# check the trace plots
post.draws = as_draws_df(m.vol)
mcmc_trace(post.draws, regex_pars = "^b_",
           facet_args = list(ncol = 4)) +
  scale_x_continuous(breaks=scales::pretty_breaks(n = 3)) +
  scale_y_continuous(breaks=scales::pretty_breaks(n = 3))
```

```
## 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. Therefore, we go ahead and perform our posterior predictive checks.

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

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

# distributions of means compared to the real values per group
p2 = ppc_stat_grouped(df.hgf$be3, post.pred, df.hgf$diagnosis) +
  theme_bw() + theme(legend.position = "none")

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



The overall simulated data fits reasonably well, even though it does not reproduce the shape completely. The mean simulated data based on the model fits well with the real data.

Inferences

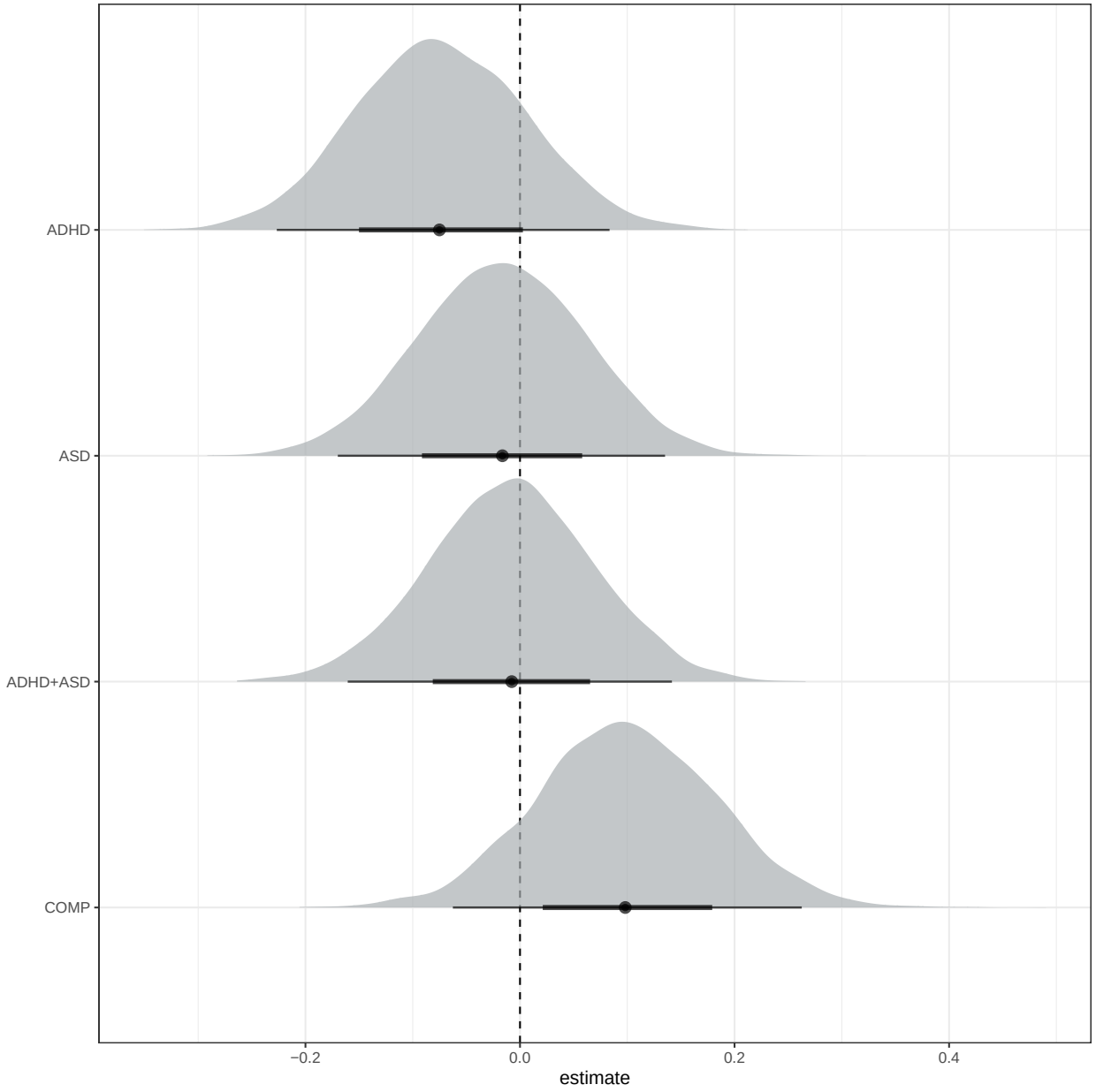
Now that we are convinced that we can trust our model, we have a look at its estimate and use the hypothesis function to assess our hypotheses and perform explorative tests.

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

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: be3 ~ diagnosis
##    Data: df.hgf (Number of observations: 90)
```

```
##   Draws: 4 chains, each with iter = 3000; warmup = 1000; thin = 1;
##          total post-warmup draws = 8000
##
## Regression Coefficients:
##            Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept      0.00      0.05    -0.09     0.10 1.00     9405     5343
## diagnosis1    -0.07      0.08    -0.23     0.08 1.00     8361     6052
## diagnosis2    -0.02      0.08    -0.17     0.14 1.00     8031     6093
## diagnosis3    -0.01      0.08    -0.16     0.14 1.00     8230     6458
##
## Further Distributional Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma     0.46      0.04     0.39     0.53 1.00     9169     5989
##
## 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).
```

```r
# get the estimates and compute group comparisons
df.m.vol = post.draws %>%
  select(starts_with("b_")) %>%
  mutate(
    b_COMP = - b_diagnosis1 - b_diagnosis2 - b_diagnosis3,
    ASD    = b_Intercept + b_diagnosis2,
    ADHD   = b_Intercept + b_diagnosis1,
    BOTH   = b_Intercept + b_diagnosis3,
    COMP   = b_Intercept + b_COMP,
  )

# plot the posterior distributions
df.m.vol %>%
  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, "diagnosis1", "ADHD"),
    coef = str_replace_all(coef, "diagnosis2", "ASD"),
    coef = str_replace_all(coef, "diagnosis3", "ADHD+ASD"),
    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_light, c_dark)) + theme(legend.position = "none")
```



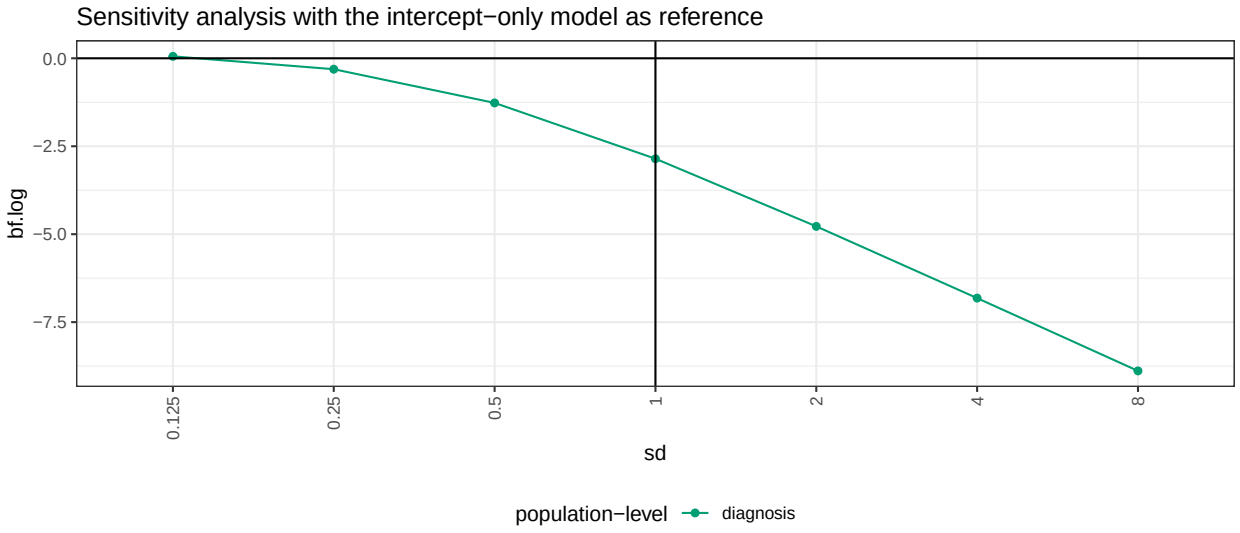

```
# H3a: COMP < ASD
h3a = hypothesis(m.vol, "0 < diagnosis1 + 2*diagnosis2 + diagnosis3")
h3a
```

```
## Hypothesis Tests for class b:
##                 Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(diagnosis1+2... < 0     0.12      0.13     -0.1     0.33       0.24
##   Post.Prob Star
## 1      0.19
## ---
## '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.
```

Bayes factor

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

# rerun the model with more iterations for bridgesampling
set.seed(5544)
m.orig = brm(f.vol,
          df.hgf, prior = priors,
          iter = 40000, warmup = 10000,
          backend = "cmdstanr", threads = threading(8),
          file = file.path(brms_dir, "m_hgf_vol_bf"), silent = 2,
          save_pars = save_pars(all = TRUE)
          )

# check if any priors 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)]
}

# 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_fixed(m.orig, "diagnosis",
                  pr.desc,
                  main.code, # prefix for all models and MLL
                  file.path("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 = "bottom",
        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.int = interpret_bf(bf.log, log = T)
        ))
```

population-level	bf.log	priors	bf.int
diagnosis	-2.8551	chosen	strong evidence against

S4.3 H3b: third level tonic volatility

Model setup

```
# code for filenames
code = "PAL_om3"

# model formula
f.om3 = brms::bf( om3 ~ diagnosis )

# set weakly informative priors
priors = c(
  prior(normal(0, 4),  class = Intercept),
  prior(normal(0, 0.50),  class = sigma),
  prior(normal(0, 0.25),  class = b)
)

# change Intercept based on empirical priors used in the HGF model
priors = priors %>%
  mutate(
    prior = if_else(
      class == "Intercept",
      gsub("\\(.*,", paste0("(", mean(df.hgf$om3mu), ", "), prior), prior),
    prior = if_else(
      class == "Intercept",
      gsub(" .*\\)", paste0(" ", mean(df.hgf$om3sa), ")"), prior), prior)
  )

kable(priors)
```

prior	class	coef	group	resp	dpar	nlpar	lb	ub	source
normal(-6.4754, 7.0733)	Intercept						NA	NA	user
normal(0, 0.5)	sigma						NA	NA	user
normal(0, 0.25)	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
  set.seed(2486)
  gen = SBC_generator_brms(f.om3, data = df.hgf, prior = priors,
   thin = 50, warmup = 10000, refresh = 2000,
   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)
```

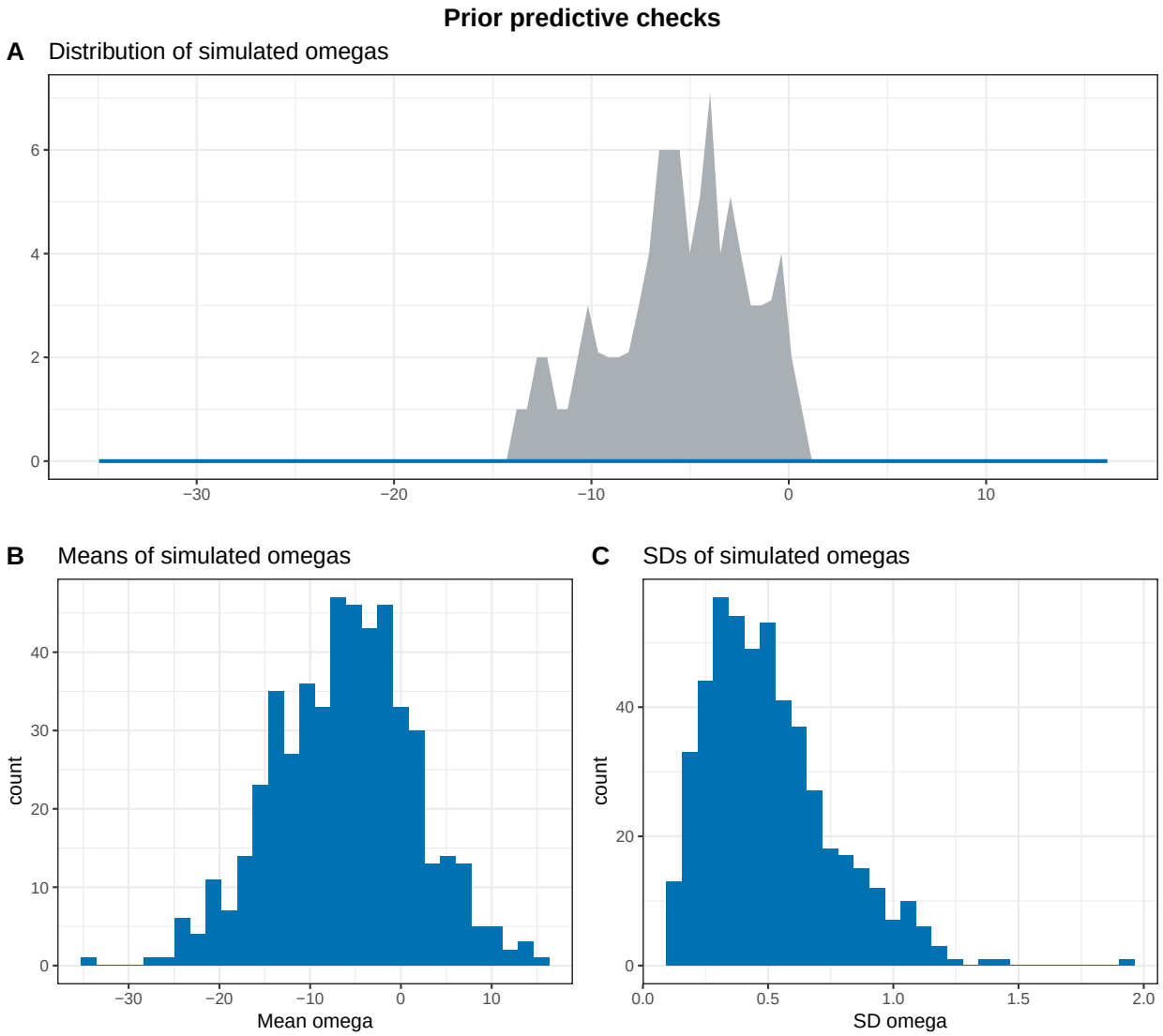
```
  saveRDS(dat, file.path(cache_dir, sprintf("dat_%s.rds", code)))
  # perform the SBC
  res = compute_SBC(
        dat,
        bck,
        cache_mode     = "results",
        cache_location = file.path(cache_dir, sprintf("res_%s", code)))
  df.results = res$stats
  df.backend = res$backend_diagnostics
  saveRDS(df.results, file = file.path(cache_dir, paste0("df_res_", code, ".rds")))
  saveRDS(df.backend,
          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 only 0 of 500 simulations had at least one parameter that had an rhat of at least 1.05 and 0 models had divergent samples. This suggests that this model performs well.

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

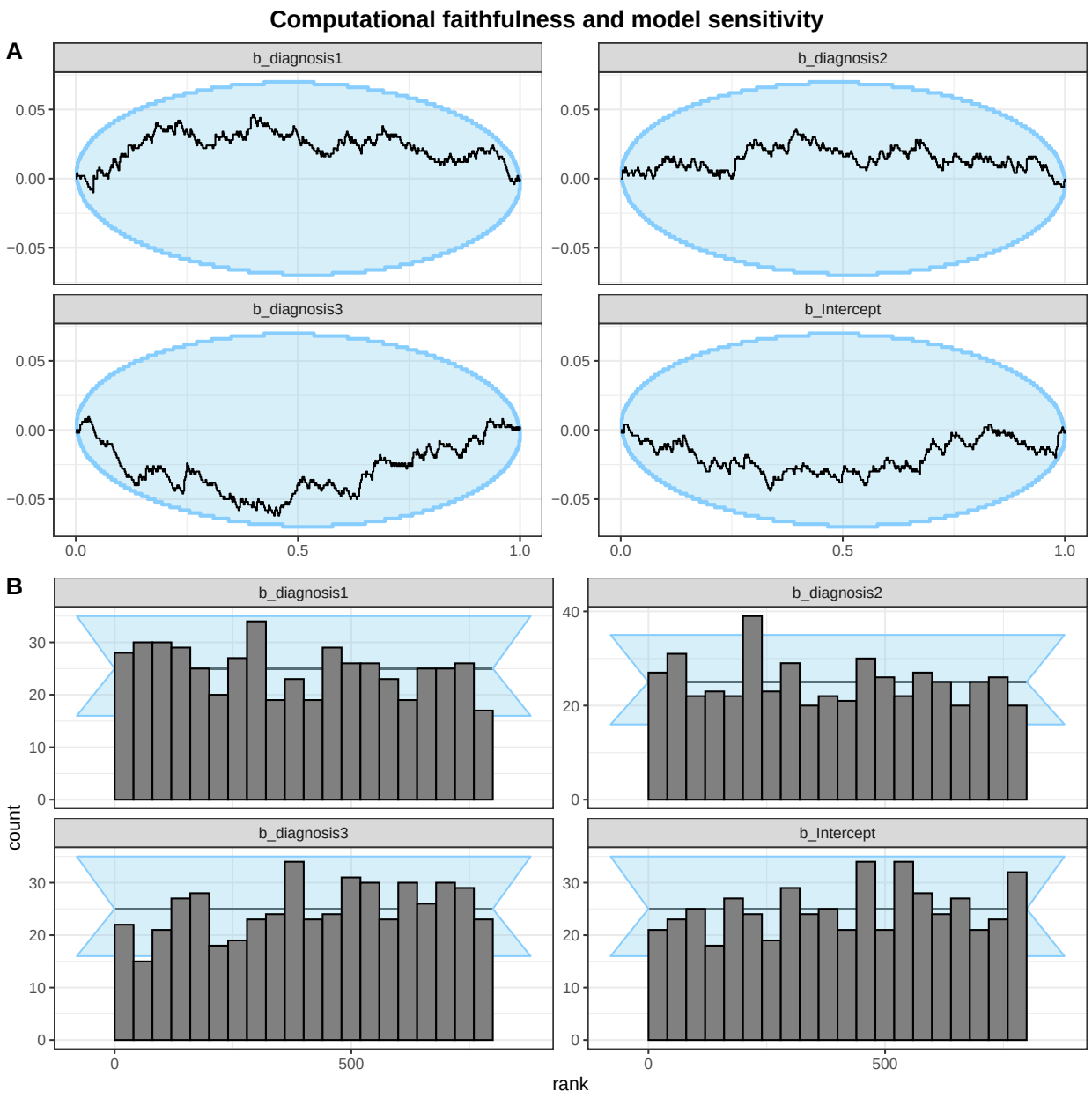
```
if (!(file.exists(file.path(cache_dir, sprintf("dvfakemat_%s", code))))) {
  # create a matrix out of generated data
  dvname = gsub(" ", "", gsub("[\\|~].*", "", f.om3)[1])
  dvfakemat = matrix(NA, nrow(dat[['generated']][[1]]), length(dat[['generated']]))
  for (i in 1:length(dat[['generated']])) {
    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
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 omegas", 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 omega", title = "Means of simulated omegas") +
  theme_bw()
p3 = ggplot() +
  stat_bin(aes(x = tmpSD), fill = c_dark) +
  labs(x = "SD omega", title = "SDs of simulated omegas") +
  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))
```



Looks okay, so 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.

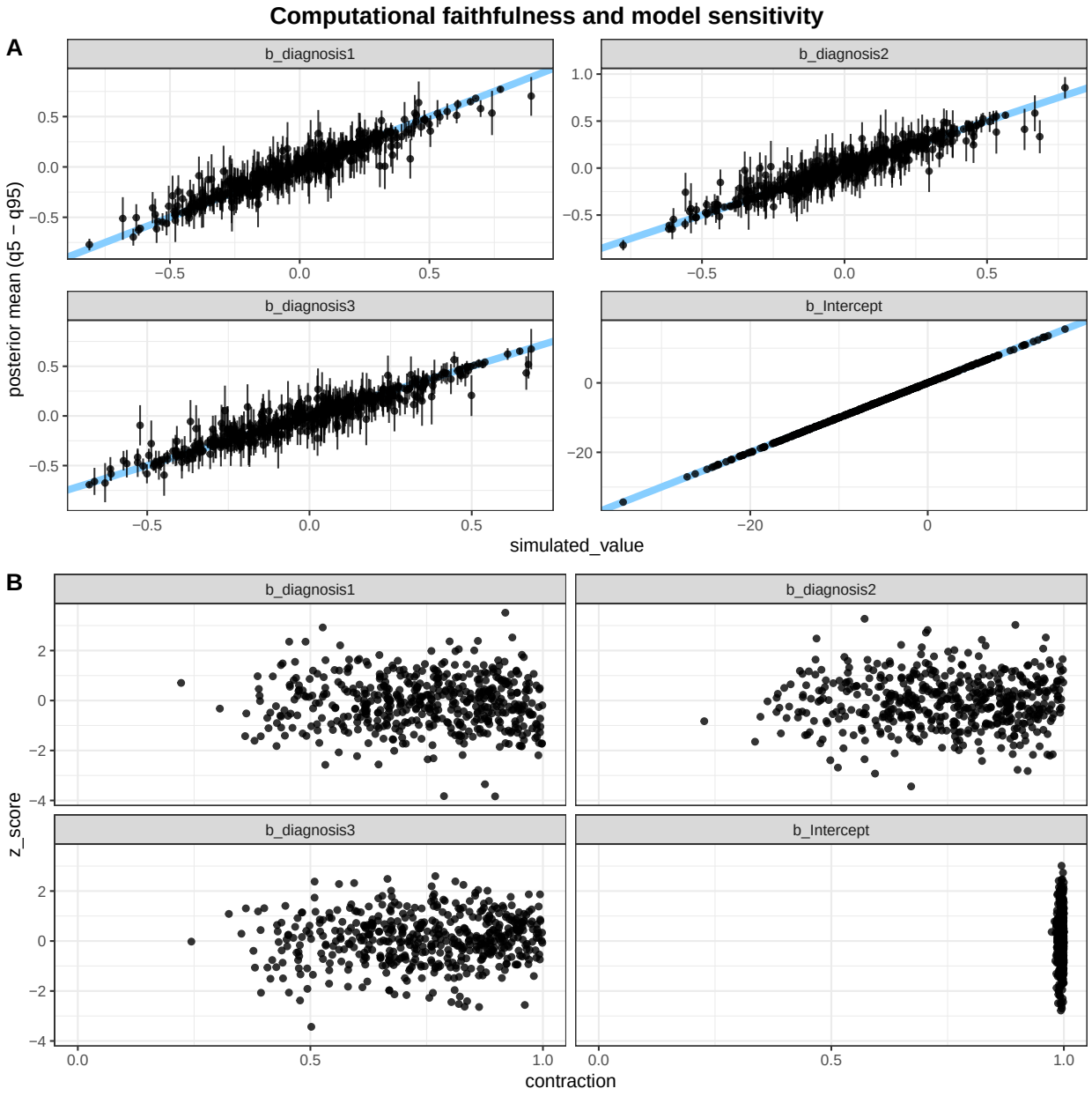
```
# get simulation numbers with issues
mx_rnk = max(df.results$max_rank)
check = merge(df.results %>%
    group_by(sim_id) %>%
      summarise(rhat = max(rhat, na.rm = T),
                max_rank = mean(max_rank)) %>%
    filter(rhat >= 1.05 | max_rank != mx_rnk),
  df.backend %>% filter(n_divergent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



Again all looks good and we judge this to be acceptable as we cannot identify clear bias patterns as described in the information accompanying the SBC package: https://hyunjimoon.github.io/SBC/articles/rank_visualizations.html

```
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(mean(df.hgf$om3sa),
                          rep(0.25, length(unique(df.results.b$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))
```



We judge this as 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(2648)
m.om3 = brm(f.om3,
            df.hgf, prior = priors,
            iter = iter, warmup = warm,
            backend = "cmdstanr", threads = threading(8),
            file = file.path(brms_dir, "m_hgf_om3"),
            save_pars = save_pars(all = TRUE)
            )
rstan::check_hmc_diagnostics(m.om3$fit)
```

```
##
## Divergences:
```

```
## 0 of 8000 iterations ended with a divergence.
```

```
##
## Tree depth:
```

```
## 0 of 8000 iterations saturated the maximum tree depth of 10.
```

```
##
## Energy:
```

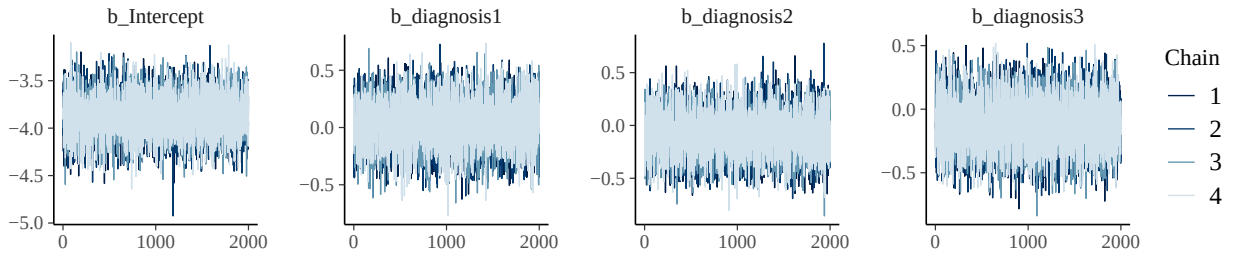
```
## E-BFMI indicated no pathological behavior.
```

```
# check that rhats are below 1.01
sum(brms::rhat(m.om3) >= 1.01, na.rm = T)
```

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

```
# check the trace plots
post.draws = as_draws_df(m.om3)
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))
```

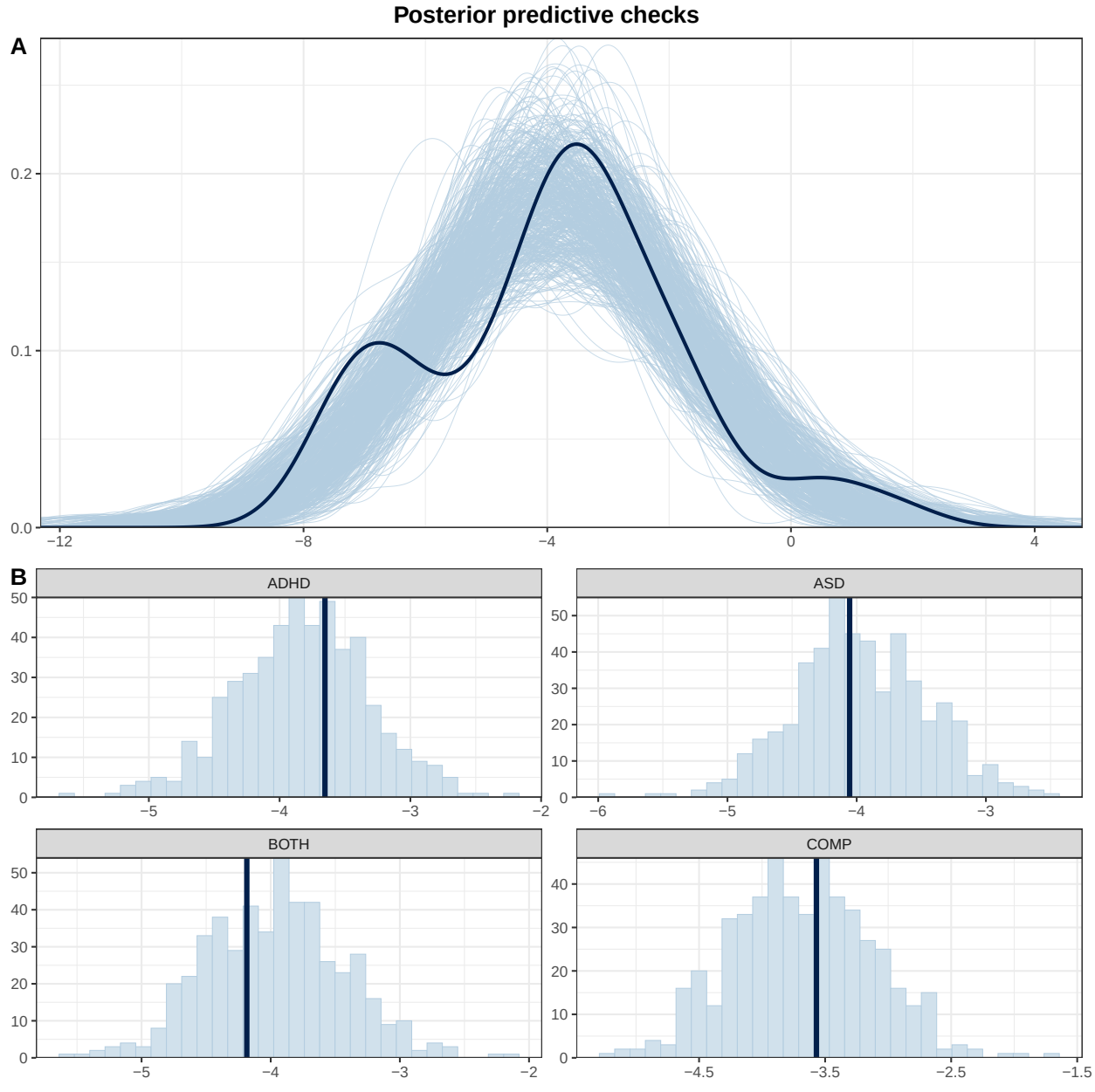
```
## Scale for x is already present.
## Adding another scale for x, which will replace the existing scale.
```



This model has no pathological behaviour with E-BFMI, no divergent sample and no rhats that are higher or equal to 1.01. Therefore, we go ahead and perform our posterior predictive checks.

```
# get posterior predictions
post.pred = posterior_predict(m.om3, ndraws = nsim)

# check the fit of the predicted data compared to the real data
p1 = pp_check(m.om3, ndraws = nsim) +
```

```
  theme_bw() + theme(legend.position = "none")

# distributions of means compared to the real values per group
p2 = ppc_stat_grouped(df.hgf$om3, post.pred, df.hgf$diagnosis) +
  theme_bw() + theme(legend.position = "none")

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



Similar to above, the simulated data based on the model fits well with the real data, altough it doesn't reproduce the smaller peak.

Inferences

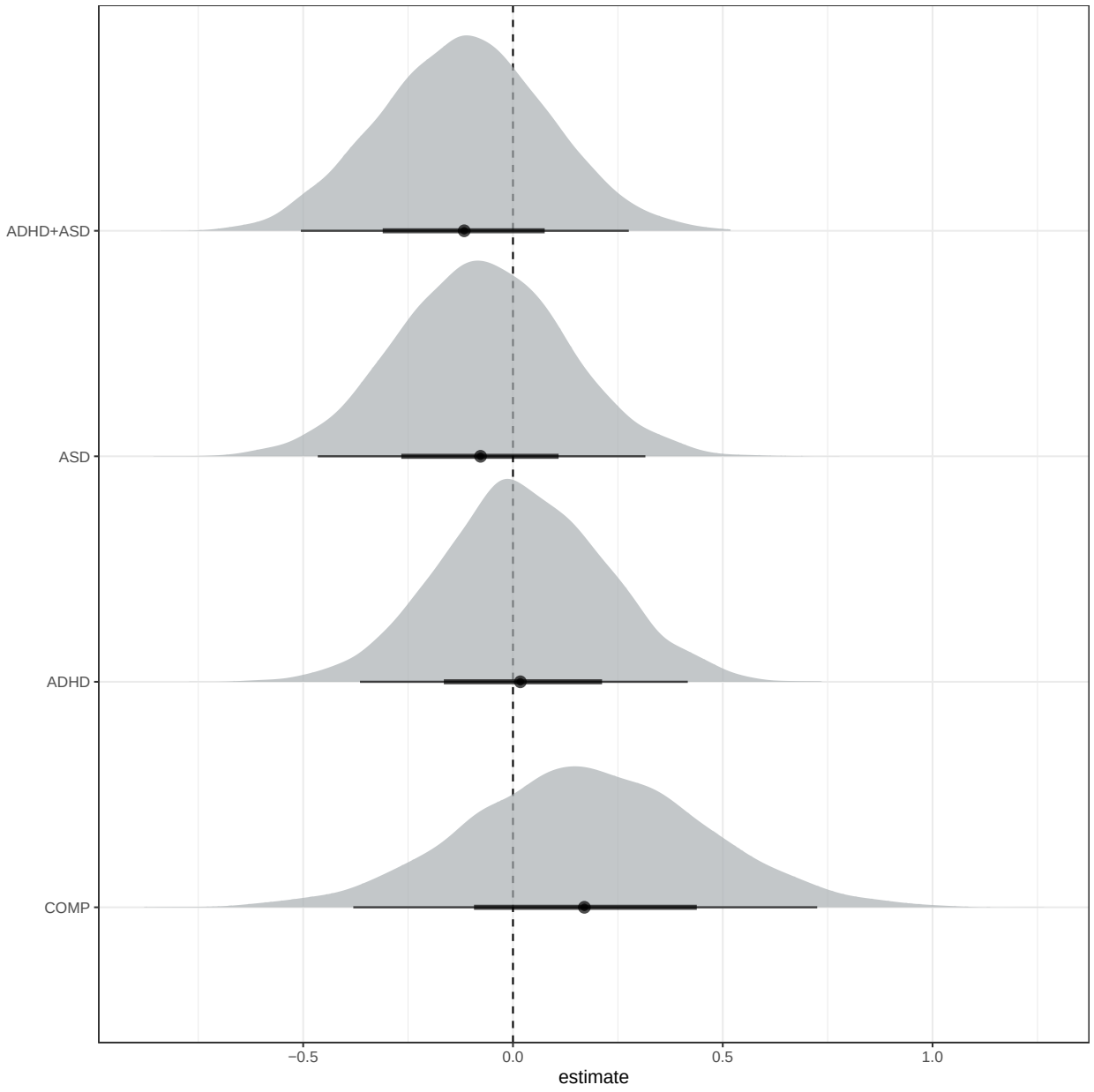
Now that we are convinced that we can trust our model, we have a look at its estimate and use the hypothesis function to assess our hypotheses and perform explorative tests.

```
# print a summary
summary(m.om3)
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: om3 ~ diagnosis
##    Data: df.hgf (Number of observations: 90)
##   Draws: 4 chains, each with iter = 3000; warmup = 1000; thin = 1;
##          total post-warmup draws = 8000
##
## Regression Coefficients:
##            Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept     -3.87      0.21    -4.29    -3.45 1.00     8653     5601
## diagnosis1     0.02      0.20    -0.36     0.42 1.00     8043     5985
## diagnosis2    -0.08      0.20    -0.47     0.32 1.00     8213     6004
## diagnosis3    -0.12      0.20    -0.51     0.28 1.00     9135     5669
##
## Further Distributional Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma     2.01      0.13     1.78     2.28 1.00     8946     6030
##
## 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).
```

```
# get the estimates and compute group comparisons
df.m.om3 = post.draws %>%
  select(starts_with("b_")) %>%
  mutate(
    b_COMP = - b_diagnosis1 - b_diagnosis2 - b_diagnosis3,
    ASD    = b_Intercept + b_diagnosis2,
    ADHD   = b_Intercept + b_diagnosis1,
    BOTH   = b_Intercept + b_diagnosis3,
    COMP   = b_Intercept + b_COMP,
  )

# plot the posterior distributions
df.m.om3 %>%
  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, "diagnosis1", "ADHD"),
    coef = str_replace_all(coef, "diagnosis2", "ASD"),
    coef = str_replace_all(coef, "diagnosis3", "ADHD+ASD"),
    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_light, c_dark)) + theme(legend.position = "none")
```



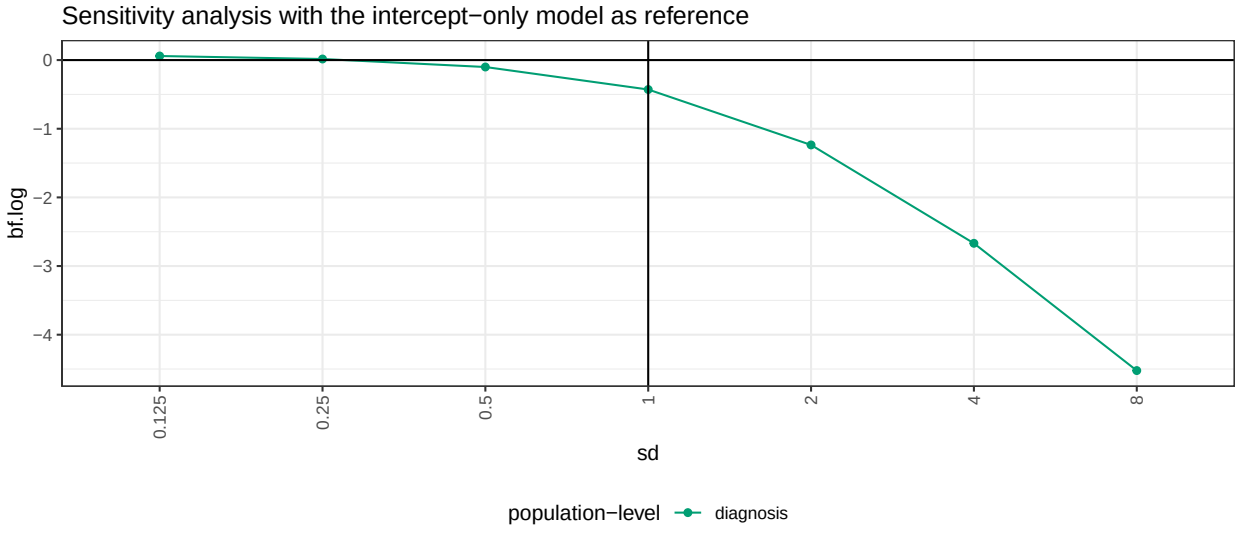

```
# H3b: COMP < ASD
h3b = hypothesis(m.om3, "0 < diagnosis1 + 2*diagnosis2 + diagnosis3")
h3b
```

```
## Hypothesis Tests for class b:
##                 Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(diagnosis1+2... < 0     0.25      0.41    -0.43     0.93       0.37
##   Post.Prob Star
## 1      0.27
## ---
## '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.
```

Bayes factor

```
# set the directory in which to save results
sense_dir = file.path(getwd(), "_brms_sens_cache")
main.code = "hgf_om3"

# rerun the model with more iterations for bridgesampling
set.seed(5544)
m.orig = brm(f.om3,
          df.hgf, prior = priors,
          iter = 40000, warmup = 10000,
          backend = "cmdstanr", threads = threading(8),
          file = file.path(brms_dir, "m_hgf_om3_bf"), silent = 2,
          save_pars = save_pars(all = TRUE)
          )

# 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)]
}

# 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_fixed(m.orig, "diagnosis",
                  pr.desc,
                  main.code, # prefix for all models and MLL
                  file.path("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))
  }
  )
}
```

```r
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 = "bottom",
        axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1))
```




```r
# print the results for the chosen priors
kable(df.pal.bf %>%
```

```
        filter(priors == "chosen" & `population-level` != "1") %>%
        arrange(desc(bf.log)) %>%
        mutate(
          bf.int = interpret_bf(bf.log, log = T)
        ))
```

population-level	bf.log	priors	bf.int
diagnosis	-0.4286	chosen	anecdotal evidence against

S4.3 H3c: second level tonic volatility

Model setup

```
# code for filenames
code = "PAL_om2"

# model formula
f.om2 = brms::bf( om2 ~ diagnosis )

# set weakly informative priors
priors = c(
  prior(normal(0, 4),  class = Intercept),
  prior(normal(0, 0.50),  class = sigma),
  prior(normal(0, 0.25),  class = b)
)

# change Intercept based on empirical priors used in the HGF model
priors = priors %>%
  mutate(
    prior = if_else(
      class == "Intercept",
      gsub("\\(.*,", paste0("(", mean(df.hgf$om2mu), ", "), prior), prior),
    prior = if_else(
      class == "Intercept",
      gsub(" .*\\)", paste0(" ", mean(df.hgf$om2sa), ")"), prior), prior)
  )

kable(priors)
```

prior	class	coef	group	resp	dpar	nlpar	lb	ub	source
normal(-6.921, 8.7788)	Intercept						NA	NA	user
normal(0, 0.5)	sigma						NA	NA	user
normal(0, 0.25)	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)))
```

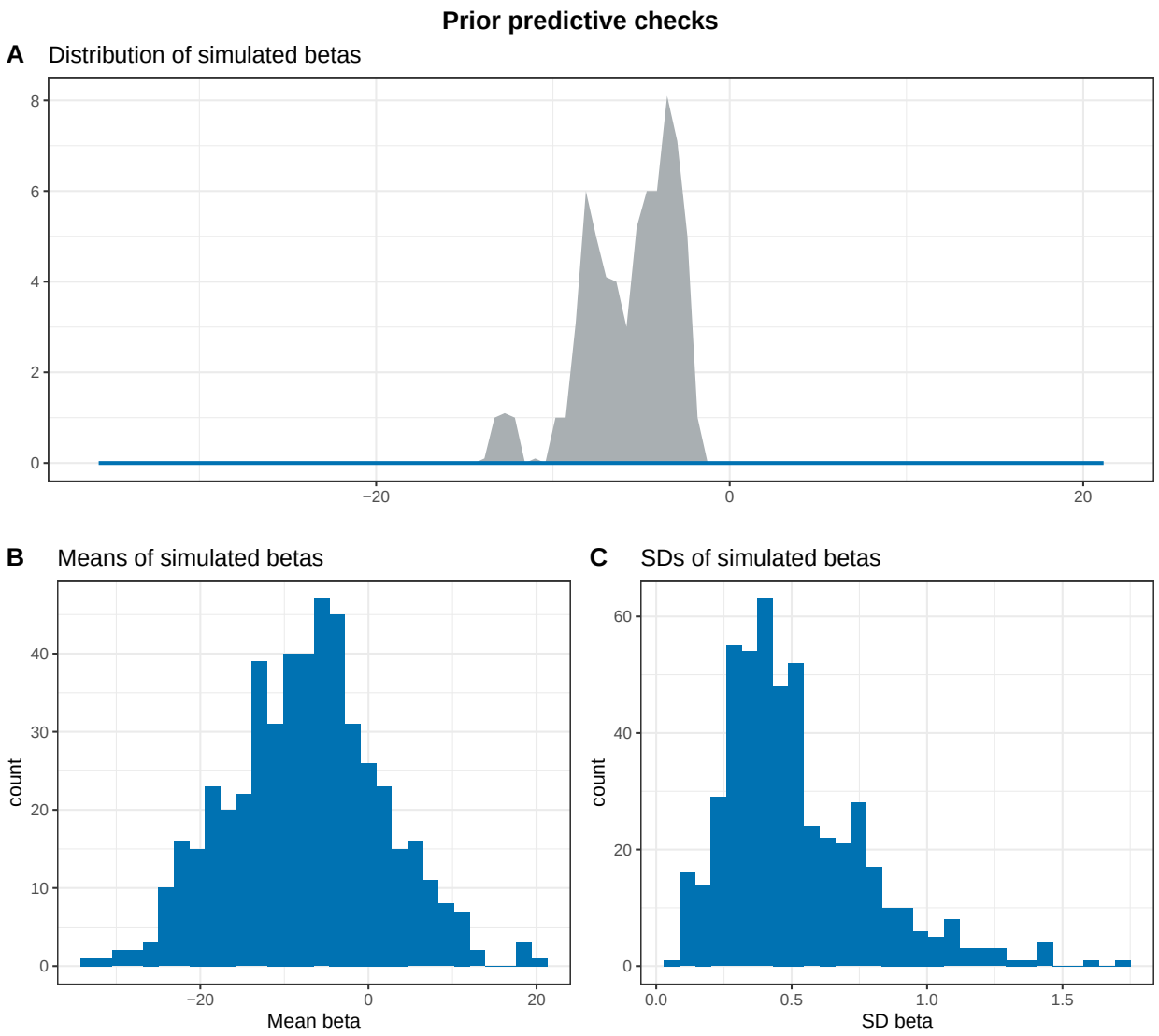
```
  df.backend = readRDS(file.path(cache_dir, sprintf("df_div_%s.rds", code)))
  dat        = readRDS(file.path(cache_dir, sprintf("dat_%s.rds", code)))
} else {
  # stimulate some data
  set.seed(2486)
  gen = SBC_generator_brms(f.om2, data = df.hgf, prior = priors,
   thin = 50, warmup = 10000, refresh = 2000,
   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(df.results, file = file.path(cache_dir, paste0("df_res_", code, ".rds")))
  saveRDS(df.backend,
          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 only 0 of 500 simulations had at least one parameter that had an rhat of at least 1.05 and 0 models had divergent samples. This suggests that this model performs well.

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

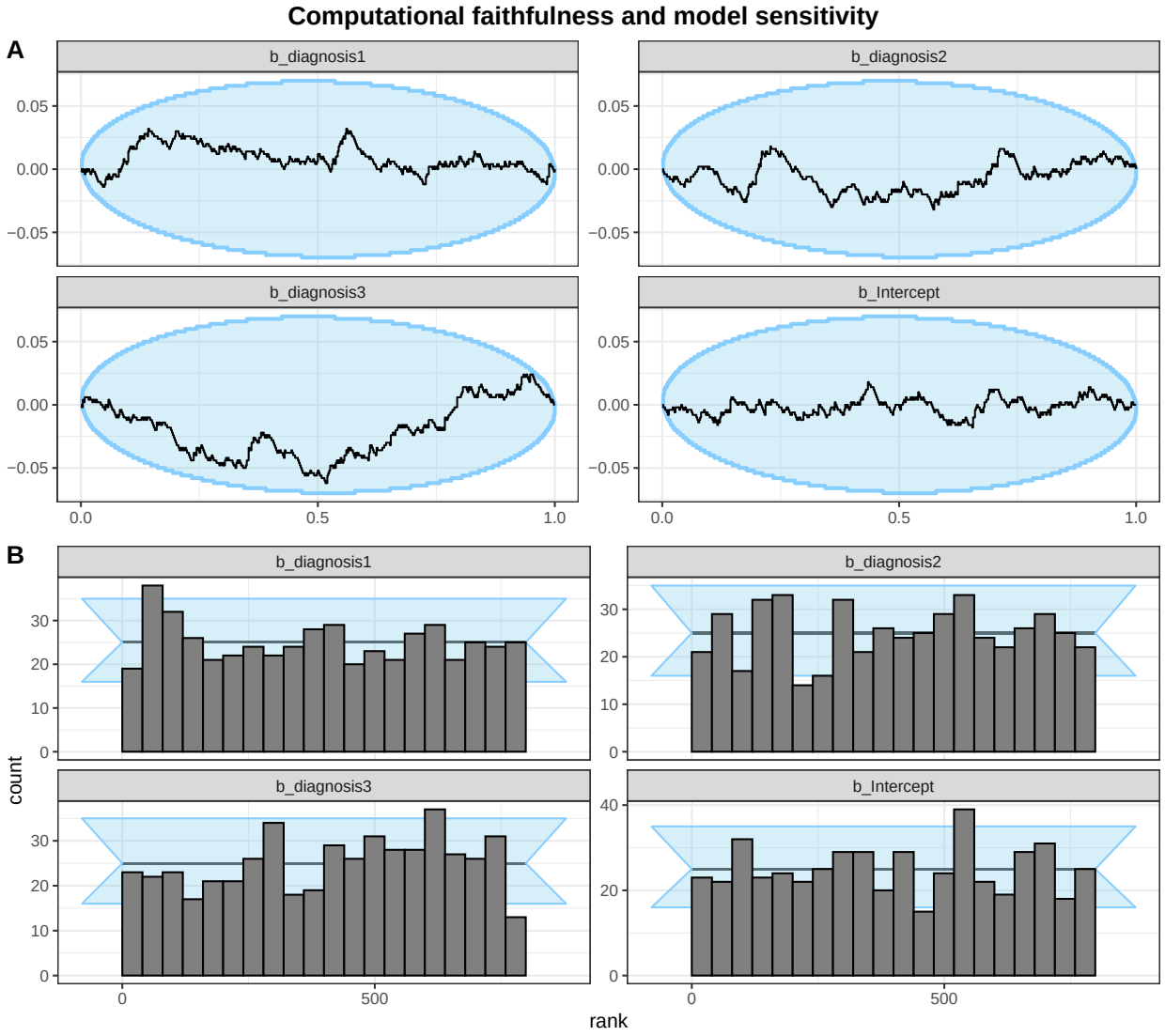
```
if (!(file.exists(file.path(cache_dir, sprintf("dvfakemat_%s", code))))) {
  # create a matrix out of generated data
  dvname = gsub(" ", "", gsub("[\\|~].*", "", f.om2)[1])
  dvfakemat = matrix(NA, nrow(dat[['generated']][[1]]), length(dat[['generated']]))
  for (i in 1:length(dat[['generated']])) {
    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
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)
```

```r
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 betas", 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 beta", title = "Means of simulated betas") +
  theme_bw()
p3 = ggplot() +
  stat_bin(aes(x = tmpSD), fill = c_dark) +
  labs(x = "SD beta", title = "SDs of simulated betas") +
  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))
```



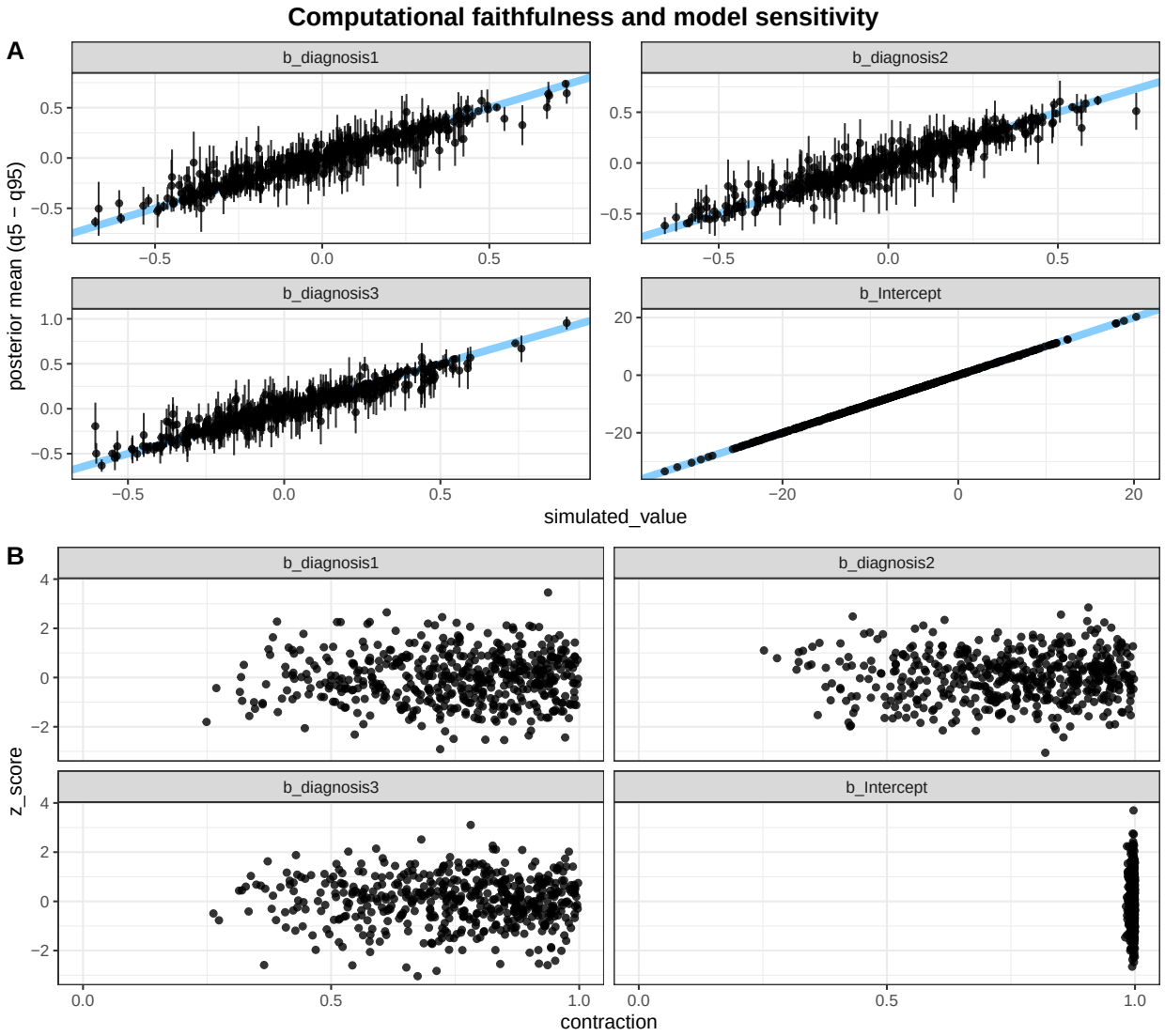

```
# get simulation numbers with issues
mx_rnk = max(df.results$max_rank)
check = merge(df.results %>%
    group_by(sim_id) %>%
      summarise(rhat = max(rhat, na.rm = T),
                max_rank = mean(max_rank)) %>%
    filter(rhat >= 1.05 | max_rank != mx_rnk),
  df.backend %>% filter(n_divergent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




```
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(mean(df.hgf$om2sa),
                                   rep(0.25, length(unique(df.results.b$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 so we move on and run the model on the actual data.

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(2486)
m.om2 = brm(f.om2,
            df.hgf, prior = priors,
            iter = iter, warmup = warm,
```

```
            backend = "cmdstanr", threads = threading(8),
            file = file.path(brms_dir, "m_hgf_om2"),
            save_pars = save_pars(all = TRUE)
            )
rstan::check_hmc_diagnostics(m.om2$fit)
```

```
##
## Divergences:
```

```
## 0 of 8000 iterations ended with a divergence.
```

```
##
## Tree depth:
```

```
## 0 of 8000 iterations saturated the maximum tree depth of 10.
```

```
##
## Energy:
```

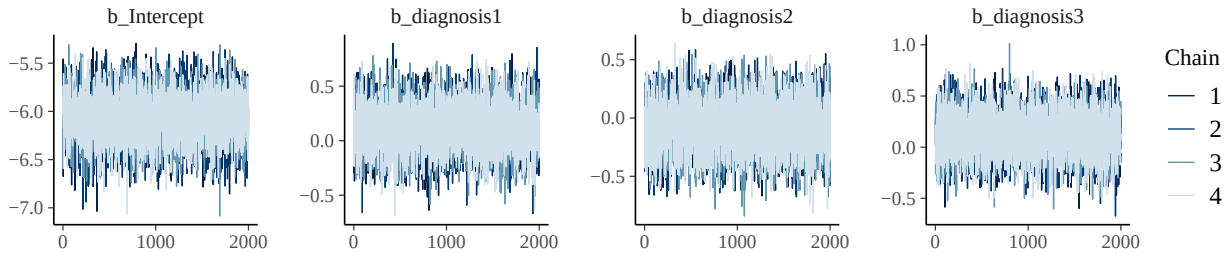
```
## E-BFMI indicated no pathological behavior.
```

```
# check that rhats are below 1.01
sum(brms::rhat(m.om2) >= 1.01, na.rm = T)
```

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

```
# check the trace plots
post.draws = as_draws_df(m.om2)
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))
```

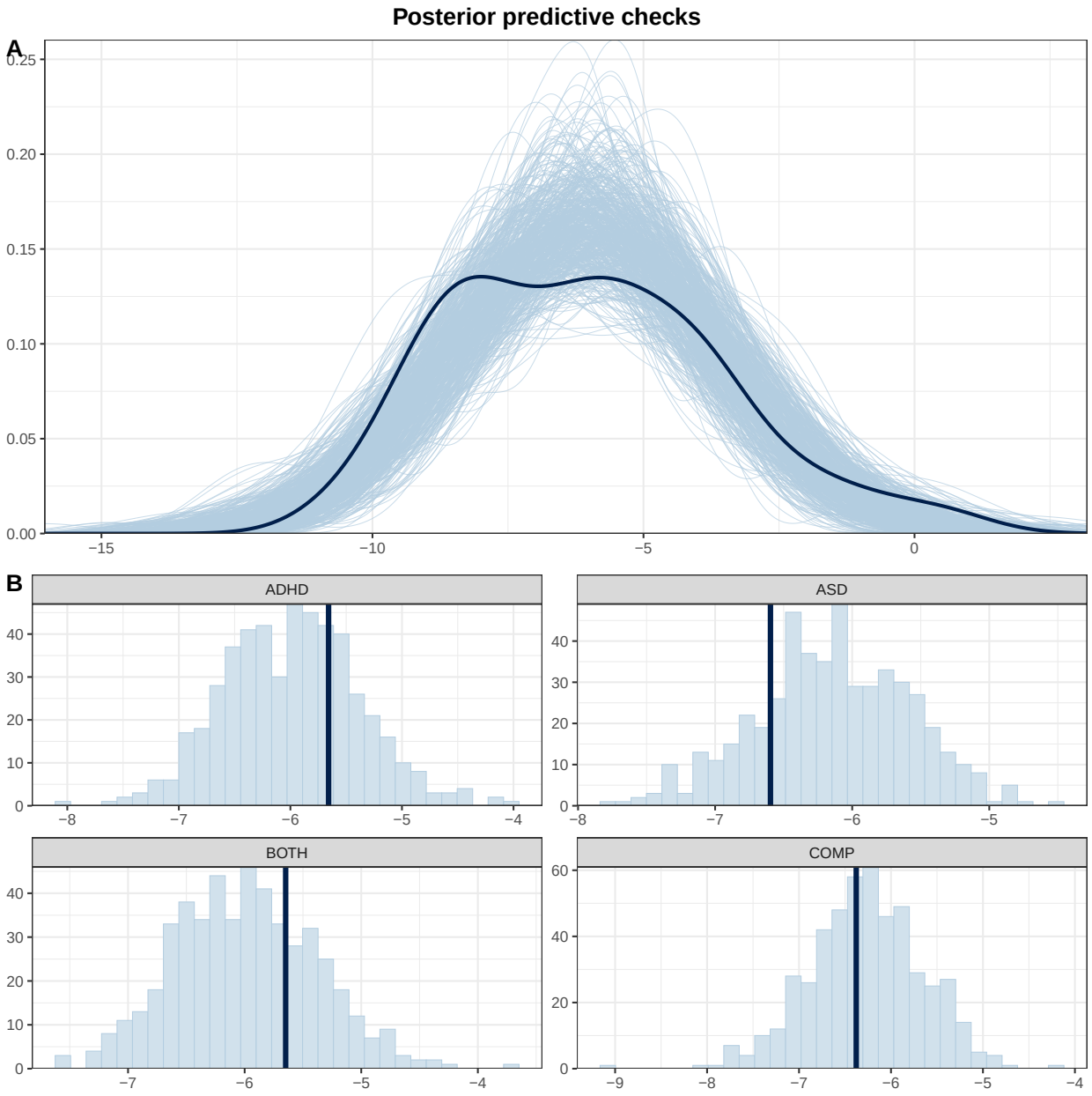
```
## Scale for x is already present.
## Adding another scale for x, which will replace the existing scale.
```



This model has no pathological behaviour with E-BFMI, no divergent sample and no rhats that are higher or equal to 1.01. Therefore, we go ahead and perform our posterior predictive checks.

```
# get posterior predictions
post.pred = posterior_predict(m.om2, ndraws = nsim)

# check the fit of the predicted data compared to the real data
p1 = pp_check(m.om2, ndraws = nsim) +
  theme_bw() + theme(legend.position = "none")

# distributions of means compared to the real values per group
p2 = ppc_stat_grouped(df.hgf$om2, post.pred, df.hgf$diagnosis) +
  theme_bw() + theme(legend.position = "none")
```

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



The simulated data based on the model fits reasonably well with the real data, although there seems to be a slight overestimation of the values in the ASD group.

Inferences

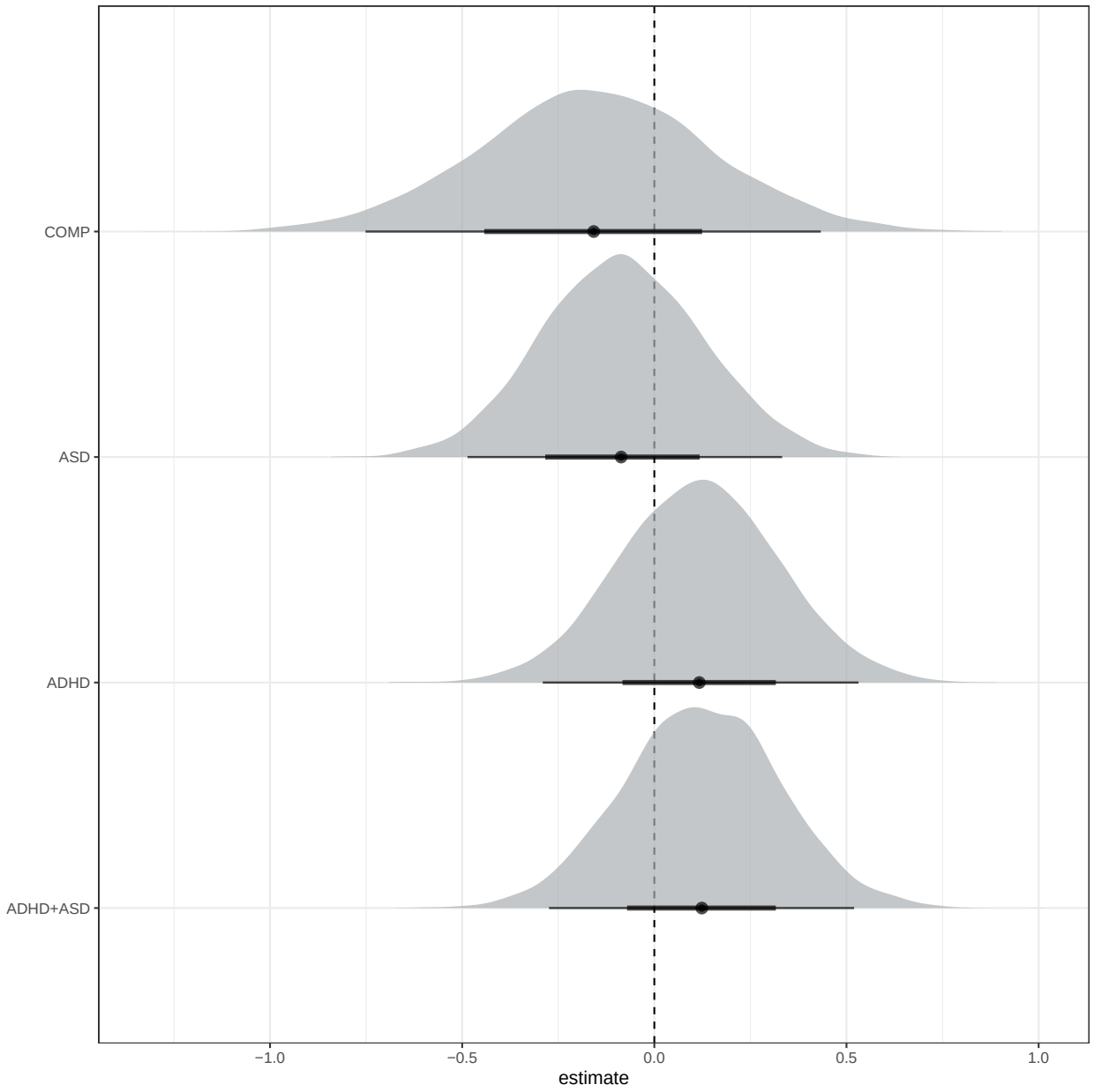
Now that we are convinced that we can trust our model, we have a look at its estimate and use the hypothesis function to assess our hypotheses and perform explorative tests.

```r
# print a summary
summary(m.om2)
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: om2 ~ diagnosis
##    Data: df.hgf (Number of observations: 90)
##   Draws: 4 chains, each with iter = 3000; warmup = 1000; thin = 1;
##          total post-warmup draws = 8000
##
## Regression Coefficients:
##            Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept     -6.08      0.24    -6.55    -5.60 1.00     8843     5713
## diagnosis1     0.12      0.21    -0.29     0.53 1.00     9188     6250
## diagnosis2    -0.08      0.21    -0.49     0.33 1.00     9215     6225
## diagnosis3     0.12      0.20    -0.27     0.52 1.00     9688     6290
##
## Further Distributional Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma     2.29      0.14     2.03     2.59 1.00     9029     6364
##
## 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).
```

```r
# get the estimates and compute group comparisons
df.m.om2 = post.draws %>%
  select(starts_with("b_")) %>%
  mutate(
    b_COMP         = - b_diagnosis1 - b_diagnosis2 - b_diagnosis3
  )

# plot the posterior distributions
df.m.om2 %>%
  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, "diagnosis1", "ADHD"),
    coef = str_replace_all(coef, "diagnosis2", "ASD"),
    coef = str_replace_all(coef, "diagnosis3", "ADHD+ASD"),
    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_light, c_dark)) + theme(legend.position = "none")
```




```
# H3b: COMP != ADHD
h3c = hypothesis(m.om2, "0 < 2*diagnosis1 + diagnosis2 + diagnosis3", alpha = 0.025)
h3c
```

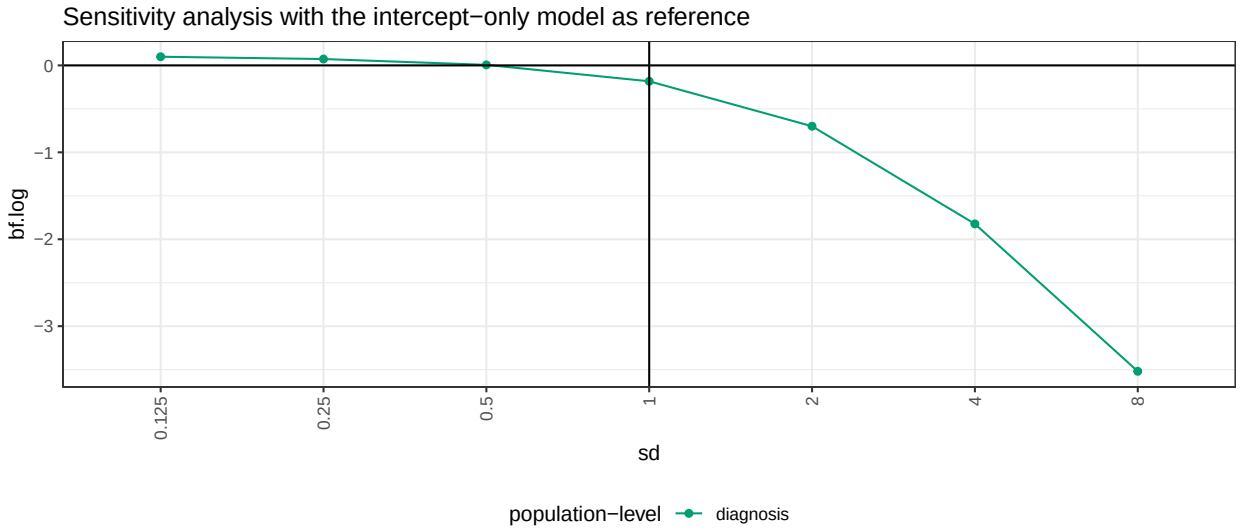
```
## Hypothesis Tests for class b:
##                 Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(2*diagnosis1... < 0    -0.27      0.44    -1.15      0.6       2.73
##   Post.Prob Star
## 1      0.73
## ---
```

```
## '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.
```

Bayes factor

```
# set the directory in which to save results
sense_dir = file.path(getwd(), "_brms_sens_cache")
main.code = "hgf_om2"

# rerun the model with more iterations for bridgesampling
set.seed(5544)
m.orig = brm(f.om2,
          df.hgf, prior = priors,
          iter = 40000, warmup = 10000,
          backend = "cmdstanr", threads = threading(8),
          file = file.path(brms_dir, "m_hgf_om2_bf"), silent = 2,
          save_pars = save_pars(all = TRUE)
          )

# 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)]
}

# 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_fixed(m.orig, "diagnosis",
                  pr.desc,
                  main.code, # prefix for all models and MLL
                  file.path("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 = "bottom",
        axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1))
```

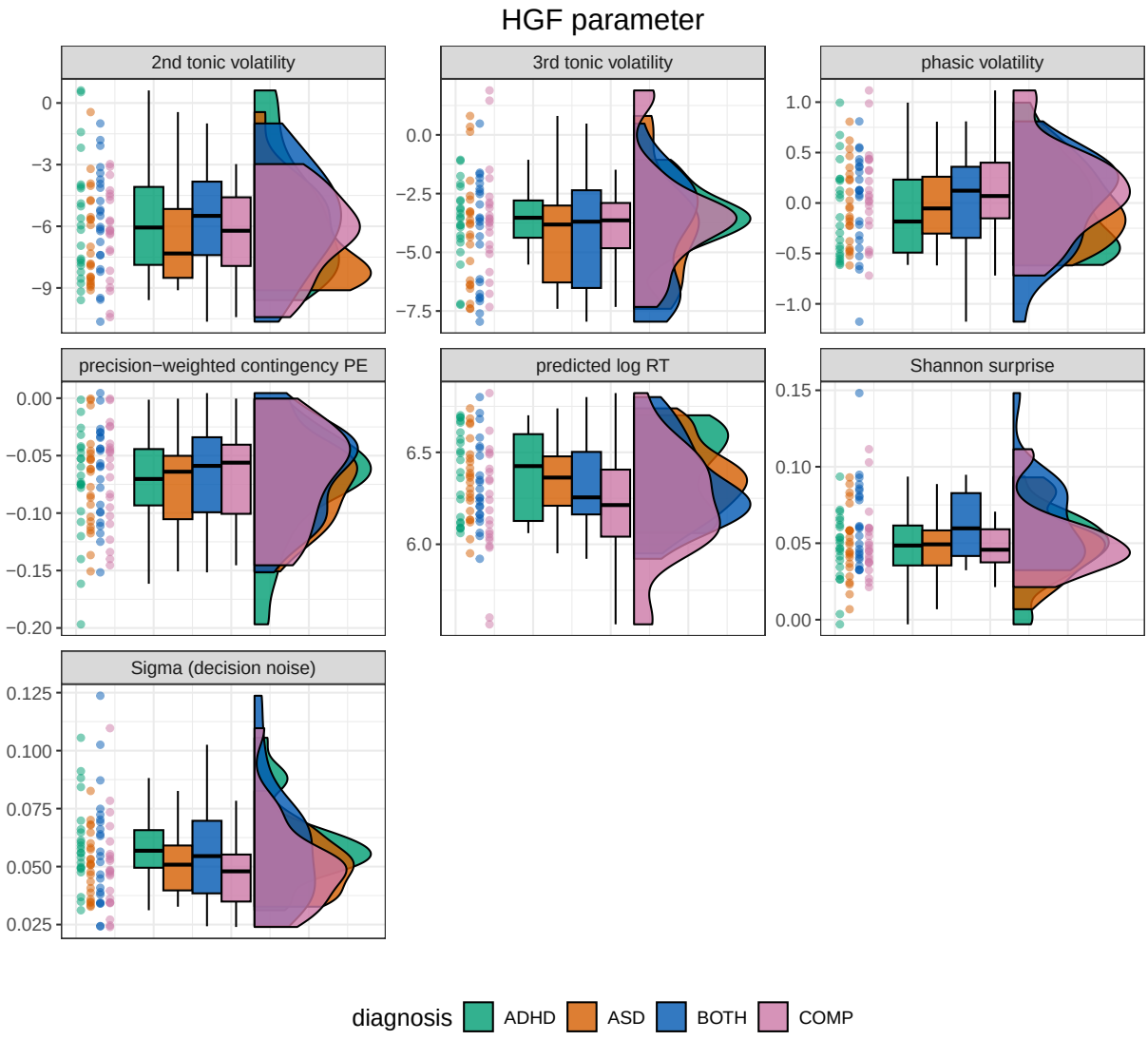



```
# print the results for the chosen priors
kable(df.pal.bf %>%
        filter(priors == "chosen" & `population-level` != "1") %>%
        arrange(desc(bf.log)) %>%
        mutate(
          bf.int = interpret_bf(bf.log, log = T)
        ))
```

population-level	bf.log	priors	bf.int
diagnosis	-0.1826	chosen	anecdotal evidence against

S4.4: Plots for all HGF parameters

```
# rain cloud plot

df.hgf %>%
  select(subID, diagnosis, be0, be1, be2, be3, ze, om2, om3) %>% #
  pivot_longer(cols = c(be0, be1, be2, be3, ze, om2, om3),
               names_to = "parameter") %>%
  mutate(
    parameter = case_match(parameter,
                           "be0" ~ "predicted log RT",
                           "be1" ~ "Shannon surprise",
                           "be2" ~ "precision-weighted contingency PE",
                           "be3" ~ "phasic volatility",
                           "ze"  ~ "Sigma (decision noise)",
                           "om2" ~ "2nd tonic volatility",
                           "om3" ~ "3rd tonic volatility"
                           )
  ) %>%
  ggplot(aes(x = 1, y = value, 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))) +
  scale_fill_manual(values = custom.col) +
  scale_color_manual(values = custom.col) +
  facet_wrap(. ~ parameter, scales = "free", ncol = 3) +
  labs(title = "HGF parameter", x = "", y = "") +
  theme_bw() +
  theme(legend.position = "bottom", plot.title = element_text(hjust = 0.5),
        text = element_text(size = 13), axis.text.x=element_blank(),
        axis.ticks.x=element_blank(), legend.direction = "horizontal")
```



S4.5 H4: Learning rate update - volatile to stable

Model setup

```
# code for filenames
code = "PAL_alpha"

# model formula
f.alpha = brms::bf( value ~ diagnosis * level * change + (level + change | subID) )

# set weakly informative priors taking Lawson 2017 into consideration
priors = c(
  prior(normal(-5, 2),    class = Intercept),
  prior(normal(0.5, 0.5), class = sigma),
  prior(normal(0.5, 0.5), class = sd),
  prior(lkj(2),           class = cor),
  prior(normal(0, 0.5),   class = b)
```

```
)
```

Simulation-based calibration

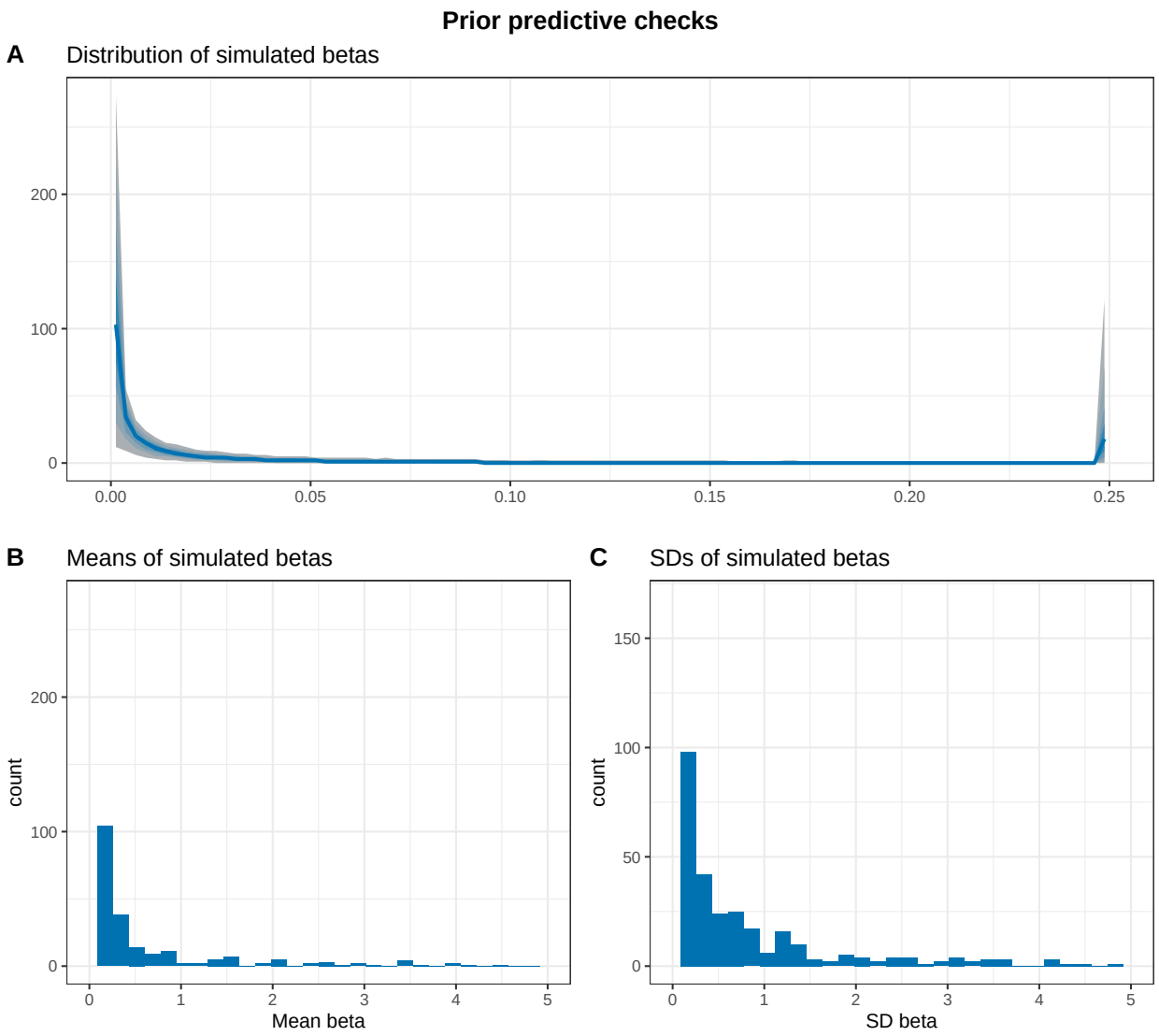
```
# check if the SBC already exists
if (file.exists(file.path(cache_dir, sprintf("df_res_%s.rds", code)))) {
  # load in the results of the SBC
  df.results = readRDS(file.path(cache_dir, sprintf("df_res_%s.rds", code)))
  df.backend = readRDS(file.path(cache_dir, sprintf("df_div_%s.rds", code)))
  dat        = readRDS(file.path(cache_dir, sprintf("dat_%s.rds", code)))
} else {
  # stimulate some data
  set.seed(2486)
  gen = SBC_generator_brms(f.alpha, data = df.upd,
   prior = priors, family = lognormal,
   thin = 50, warmup = 10000, refresh = 2000,
   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(df.results, file = file.path(cache_dir, paste0("df_res_", code, ".rds")))
  saveRDS(df.backend,
          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 only 0 of 500 simulations had at least one parameter that had an rhat of at least 1.05 and 12 models had divergent samples. This suggests that while there are some divergence issues this model performs well. We are going to keep an eye out for divergence issues 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.alpha)[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
```

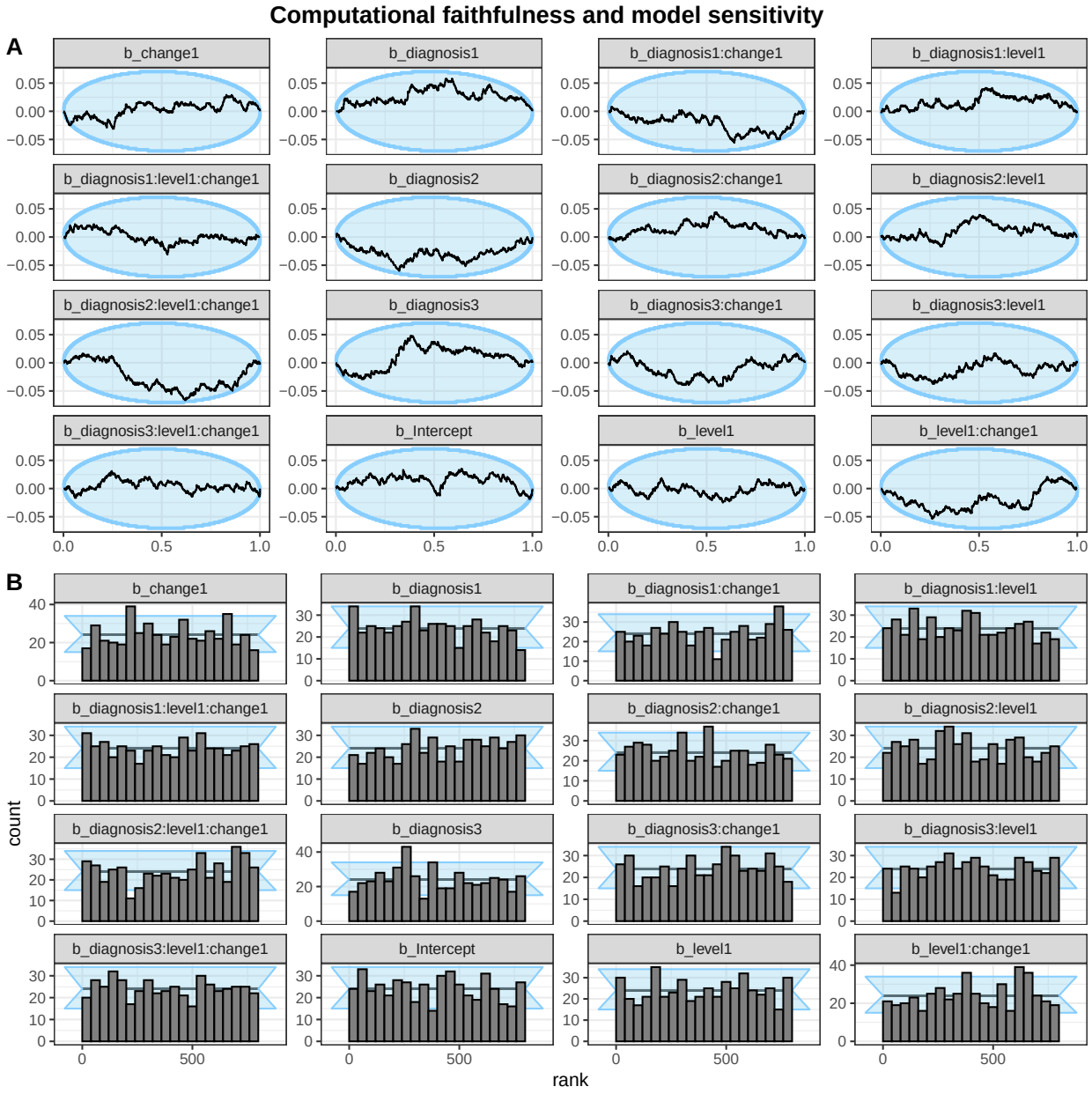
```r
dvfakematH = dvfakemat
dvfakematH[dvfakematH > 0.25] = 0.25
breaks = seq(min(dvfakematH, na.rm=T), max(dvfakematH, na.rm=T), length.out = 101)
binwidth = breaks[2] - breaks[1]
histmat = matrix(NA, ncol = dim(dvfakemat)[2], nrow = length(breaks)-1)
for (i in 1:dim(dvfakemat)[2]) {
  histmat[,i] = 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 betas", 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 beta", title = "Means of simulated betas") +
  theme_bw() + xlim(0, 5)
p3 = ggplot() +
  stat_bin(aes(x = tmpSD), fill = c_dark) +
  labs(x = "SD beta", title = "SDs of simulated betas") +
  theme_bw() + xlim(0, 5)
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))
```



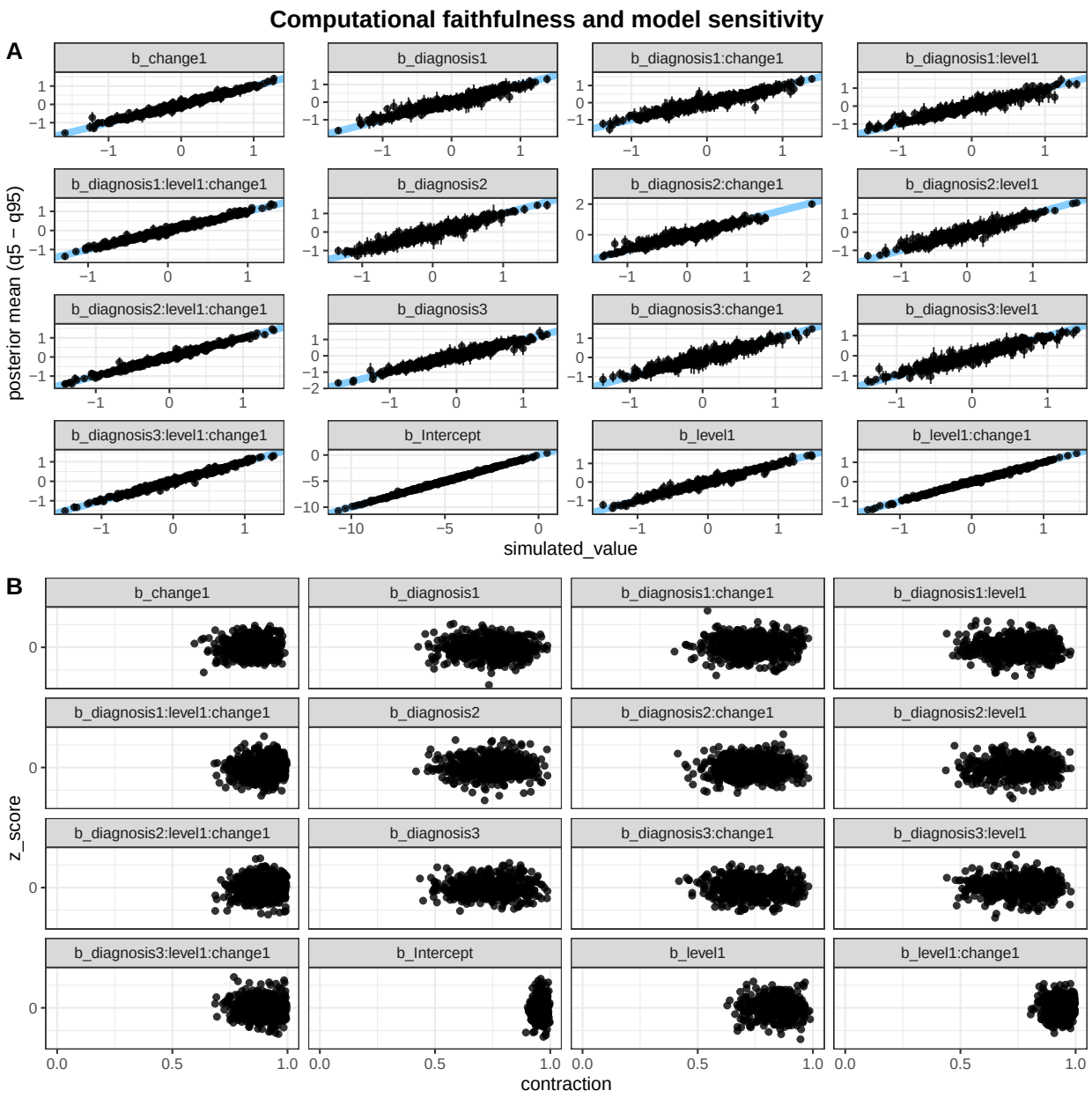
Subfigure B shows the distribution of the simulated data with bluer bands being more likely than greyer bands. It shows a general distribution that fits our expectations, even though there are quite a few values that are unreastically large. This is because we had to increase the standard deviations in the priors to achieve appropriate contraction values. The same applies to the distribution of the means and standard deviations in the simulated datasets. 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.

```
# 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))
```



```
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(2,
                                   rep(0.5, 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 and we move on to running the model on the data.

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(2486)
m.alpha = brm(f.alpha, family = lognormal,
              df.upd, prior = priors,
              iter = iter, warmup = warm,
              backend = "cmdstanr", threads = threading(8),
              file = file.path(brms_dir, "m_hgf_alpha"),
              save_pars = save_pars(all = TRUE)
```

```
            )
rstan::check_hmc_diagnostics(m.alpha$fit)
```

```
##
## Divergences:
```

```
## 0 of 8000 iterations ended with a divergence.
```

```
##
## Tree depth:
```

```
## 0 of 8000 iterations saturated the maximum tree depth of 10.
```

```
##
## Energy:
```

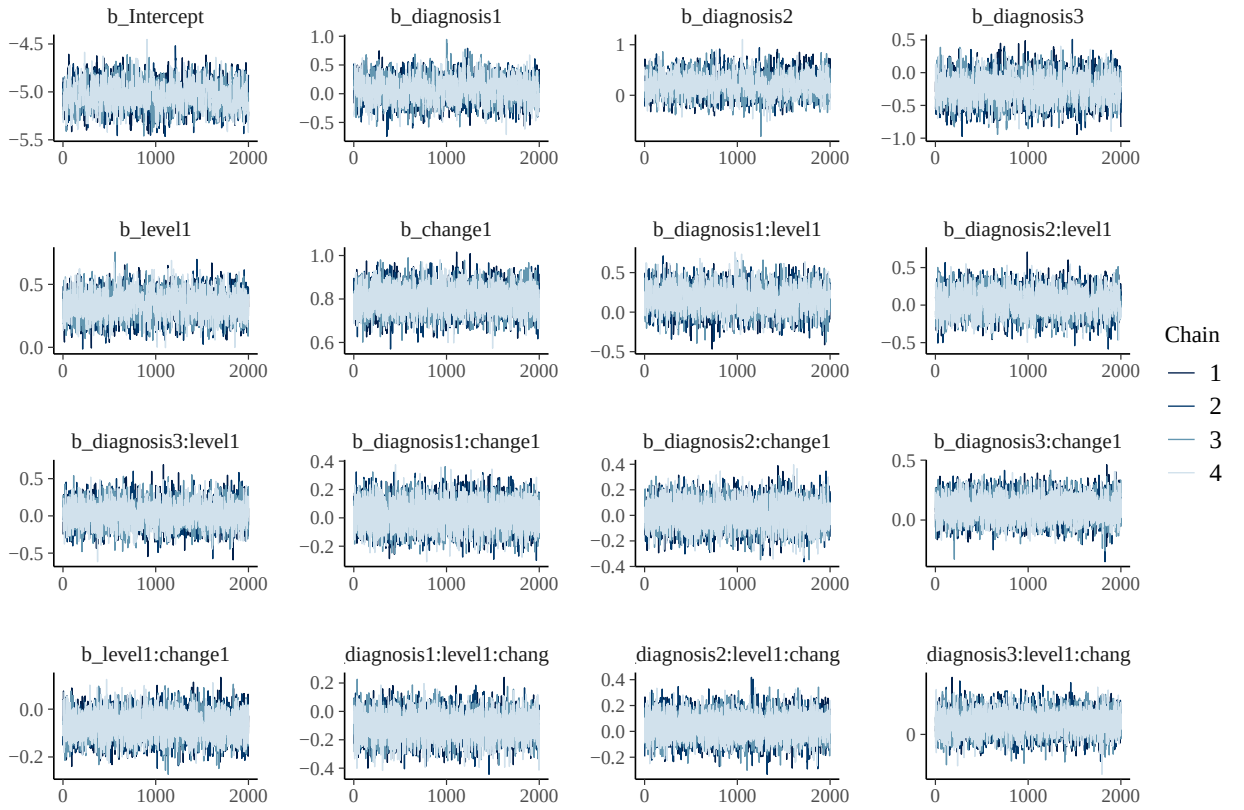
```
## E-BFMI indicated no pathological behavior.
```

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

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

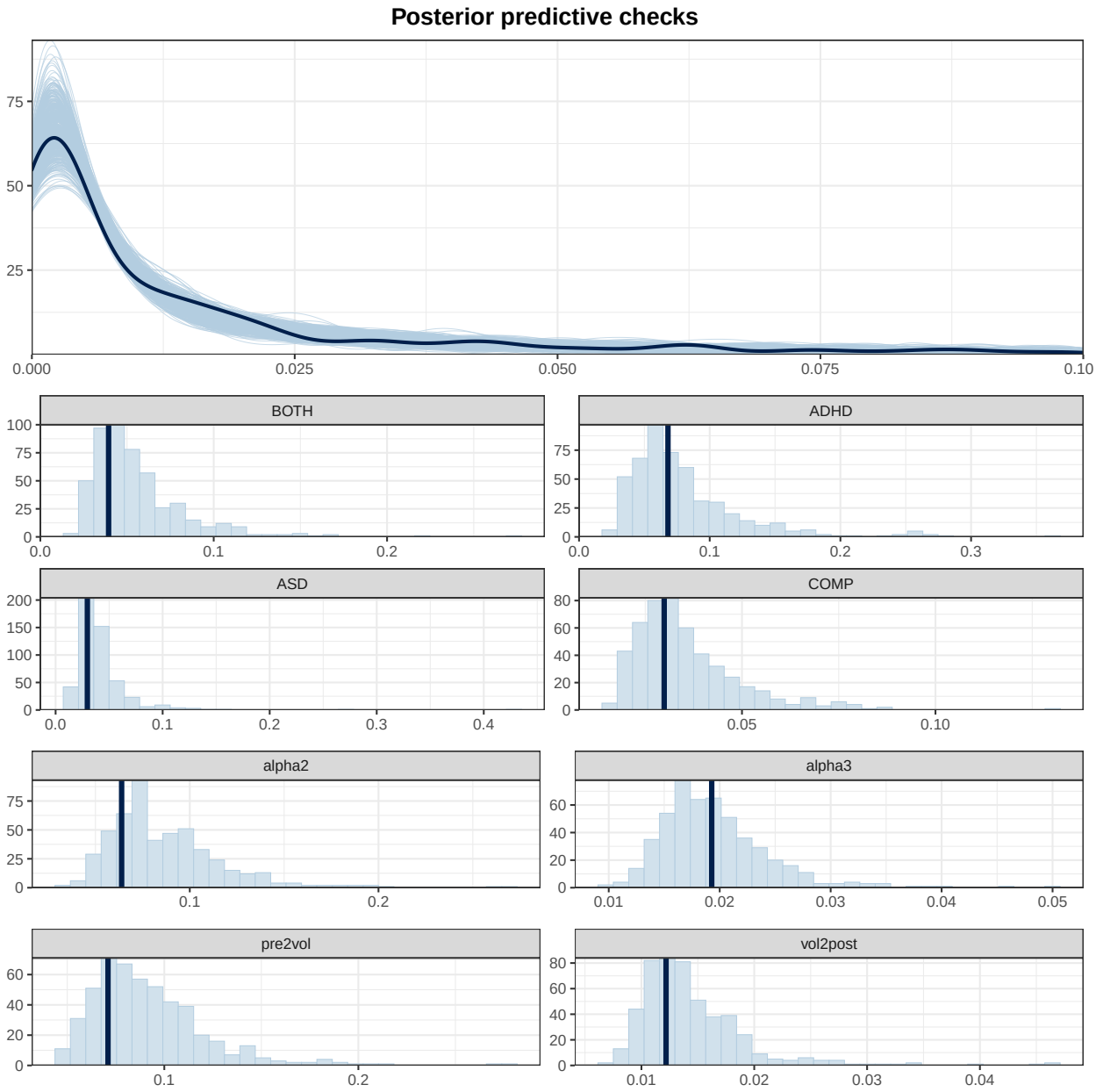
```
# check the trace plots
post.draws = as_draws_df(m.alpha)
mcmc_trace(post.draws, regex_pars = "^b_",
           facet_args = list(ncol = 4)) +
  scale_x_continuous(breaks=scales::pretty_breaks(n = 3)) +
  scale_y_continuous(breaks=scales::pretty_breaks(n = 3))
```

```
## 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. Therefore, we go ahead and perform our posterior predictive checks.

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

# check the fit of the predicted data compared to the real data
p1 = pp_check(m.alpha, ndraws = nsim) +
  theme_bw() + theme(legend.position = "none") + xlim(0, 0.10)

# distributions of means compared to the real values per group
p2 = ppc_stat_grouped(df.upd$value, post.pred, df.upd$diagnosis) +
  theme_bw() + theme(legend.position = "none")
p3 = ppc_stat_grouped(df.upd$value, post.pred, df.upd$level) +
  theme_bw() + theme(legend.position = "none")
p4 = ppc_stat_grouped(df.upd$value, post.pred, df.upd$change) +
  theme_bw() + theme(legend.position = "none")

p = ggarrange(p1, p2, ggarrange(p3, p4, nrow = 2),
          nrow = 3, ncol = 1)
annotate_figure(p, top = text_grob("Posterior predictive checks",
                                   face = "bold", size = 14))
```



The posterior predictive checks look fine.

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 hypotheses and perform explorative tests.

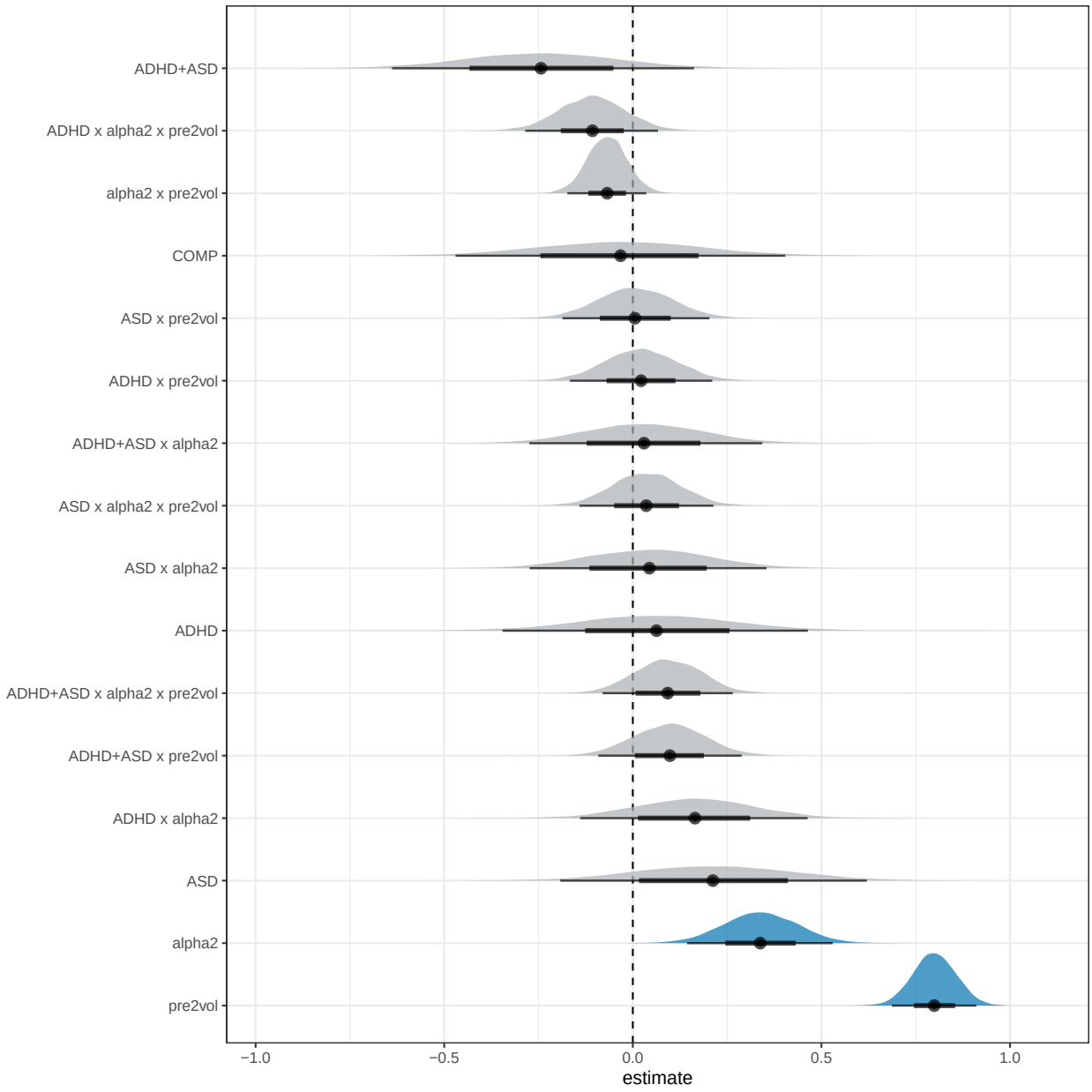
```
# print a summary
summary(m.alpha)
```

```
##  Family: lognormal
##   Links: mu = identity; sigma = identity
## Formula: value ~ diagnosis * level * change + (level + change | subID)
##    Data: df.upd (Number of observations: 360)
##   Draws: 4 chains, each with iter = 3000; warmup = 1000; thin = 1;
```

```
##          total post-warmup draws = 8000
##
## Multilevel Hyperparameters:
## ~subID (Number of levels: 90)
##                         Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS
## sd(Intercept)               1.13      0.10     0.94     1.33 1.00     1858
## sd(level1)                  0.78      0.09     0.62     0.95 1.00     2958
## sd(change1)                 0.19      0.07     0.05     0.32 1.00     2809
## cor(Intercept,level1)       0.38      0.12     0.13     0.60 1.00     2115
## cor(Intercept,change1)      0.61      0.21     0.10     0.91 1.00     4176
## cor(level1,change1)         0.64      0.22     0.12     0.93 1.00     4658
##                         Tail_ESS
## sd(Intercept)               3383
## sd(level1)                  5000
## sd(change1)                 1578
## cor(Intercept,level1)       4092
## cor(Intercept,change1)      3995
## cor(level1,change1)         3931
##
## Regression Coefficients:
##                             Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS
## Intercept                      -5.04      0.13    -5.29    -4.78 1.00     1305
## diagnosis1                      0.06      0.20    -0.35     0.46 1.00     1569
## diagnosis2                      0.21      0.21    -0.19     0.62 1.01     1414
## diagnosis3                     -0.24      0.20    -0.64     0.16 1.00     1883
## level1                          0.34      0.10     0.14     0.53 1.00     2216
## change1                         0.80      0.06     0.69     0.91 1.00     5210
## diagnosis1:level1               0.16      0.16    -0.14     0.46 1.00     2437
## diagnosis2:level1               0.04      0.16    -0.27     0.35 1.00     2420
## diagnosis3:level1               0.03      0.16    -0.27     0.34 1.00     2769
## diagnosis1:change1              0.02      0.10    -0.17     0.21 1.00     5374
## diagnosis2:change1              0.01      0.10    -0.19     0.20 1.00     5901
## diagnosis3:change1              0.10      0.10    -0.09     0.29 1.00     6371
## level1:change1                 -0.07      0.05    -0.17     0.04 1.00     9393
## diagnosis1:level1:change1      -0.11      0.09    -0.29     0.07 1.00     7480
## diagnosis2:level1:change1       0.04      0.09    -0.14     0.21 1.00     7647
## diagnosis3:level1:change1       0.09      0.09    -0.08     0.26 1.00     7161
##                             Tail_ESS
## Intercept                       3005
## diagnosis1                      2977
## diagnosis2                      3001
## diagnosis3                      3319
## level1                          3765
## change1                         5819
## diagnosis1:level1               4088
## diagnosis2:level1               4033
## diagnosis3:level1               4428
## diagnosis1:change1              5890
## diagnosis2:change1              5691
## diagnosis3:change1              5701
## level1:change1                  5962
## diagnosis1:level1:change1       5489
## diagnosis2:level1:change1       5751
## diagnosis3:level1:change1       5548
```

```
##
## Further Distributional Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma     1.01      0.06     0.91     1.13 1.00     2662     4254
##
## 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).
```

```r
# get the estimates and compute group comparisons
df.m.alpha = post.draws %>%
  select(starts_with("b_")) %>%
  mutate(
    b_COMP         = - b_diagnosis1 - b_diagnosis2 - b_diagnosis3
  )

# plot the posterior distributions
df.m.alpha %>%
  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, "level1", "alpha2"),
    coef = str_replace_all(coef, "change1", "pre2vol"),
    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.upd,
               model.matrix(~ diagnosis * level * change, data = df.upd)) %>%
  ungroup() %>%
  select(-subID, -value) %>% distinct()

t(df.des %>%
    filter(level == "alpha3" &
             (diagnosis == "ASD" | diagnosis == "COMP")) %>%
    group_by(diagnosis) %>%
    summarise(across(where(is.numeric), ~ mean(.x))) %>%
    arrange(diagnosis) %>%
    select(where(is.numeric)) %>%
    map_df(~ diff(.x))) # COMP - ASD
```

```
##                            [,1]
## (Intercept)                   0
## diagnosis1                   -1
## diagnosis2                   -1
## diagnosis3                   -2
## level1                        0
## change1                       0
## diagnosis1:level1             1
## diagnosis2:level1             1
## diagnosis3:level1             2
## diagnosis1:change1            0
## diagnosis2:change1            0
## diagnosis3:change1            0
## level1:change1                0
## diagnosis1:level1:change1     0
## diagnosis2:level1:change1     0
## diagnosis3:level1:change1     0
```

```
h4a = hypothesis(m.alpha, "0 > -(diagnosis1 + diagnosis2 + 2*diagnosis3) +
                 (diagnosis1:level1 + diagnosis2:level1 + 2*diagnosis3:level1)")
h4a
```

```
## Hypothesis Tests for class b:
##                 Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(-(diagnosis1... > 0    -0.48      0.38    -1.09     0.15       0.11
##   Post.Prob Star
## 1       0.1
## ---
## '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.
```

```
# H4b: alpha2 ASD - COMP < 0
t(df.des %>%
    filter(level == "alpha2" &
             (diagnosis == "ASD" | diagnosis == "COMP")) %>%
    group_by(diagnosis) %>%
    summarise(across(where(is.numeric), ~ mean(.x))) %>%
    arrange(diagnosis) %>%
    select(where(is.numeric)) %>%
    map_df(~ diff(.x))) # COMP - ASD
```

```
##                            [,1]
## (Intercept)                   0
## diagnosis1                   -1
## diagnosis2                   -1
## diagnosis3                   -2
## level1                        0
## change1                       0
## diagnosis1:level1            -1
## diagnosis2:level1            -1
## diagnosis3:level1            -2
## diagnosis1:change1            0
## diagnosis2:change1            0
## diagnosis3:change1            0
```

```
## level1:change1               0
## diagnosis1:level1:change1    0
## diagnosis2:level1:change1    0
## diagnosis3:level1:change1    0
```

```
h4b = hypothesis(m.alpha, "0 < -(diagnosis1 + diagnosis2 + 2*diagnosis3) +
                 -(diagnosis1:level1 + diagnosis2:level1 + 2*diagnosis3:level1)")
h4b
```

```
## Hypothesis Tests for class b:
##                 Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(-(diagnosis1... < 0     0.06      0.49    -0.74     0.88       0.84
##   Post.Prob Star
## 1      0.46
## ---
## '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.
```

```
# Explore: ASD generally lower alpha?
t(df.des %>%
    filter(diagnosis == "ASD" | diagnosis == "COMP") %>%
    select(-change, -level) %>%
    group_by(diagnosis) %>%
    summarise(across(where(is.numeric), ~ mean(.x, na.rm = TRUE))) %>%
    arrange(diagnosis) %>%
    select(where(is.numeric)) %>%
    map_df(~ diff(.x))) # COMP - ASD
```

```
##                            [,1]
## (Intercept)                   0
## diagnosis1                   -1
## diagnosis2                   -1
## diagnosis3                   -2
## level1                        0
## change1                       0
## diagnosis1:level1             0
## diagnosis2:level1             0
## diagnosis3:level1             0
## diagnosis1:change1            0
## diagnosis2:change1            0
## diagnosis3:change1            0
## level1:change1                0
## diagnosis1:level1:change1     0
## diagnosis2:level1:change1     0
## diagnosis3:level1:change1     0
```

```
e1  = hypothesis(m.alpha, "0 < -(diagnosis1 + diagnosis2 + 2*diagnosis3)",
                 alpha = 0.025)
e1
```

```
## Hypothesis Tests for class b:
##                 Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(-(diagnosis1... < 0    -0.21      0.35    -0.89     0.46       2.66
##   Post.Prob Star
## 1      0.73
```

```
## ---
## '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.
```

```
# Explore: interaction effect between alpha level and ASD vs. COMP
t(df.des %>%
    filter((diagnosis == "ASD" | diagnosis == "COMP") & change == "pre2vol") %>%
    select(-change) %>%
    group_by(diagnosis, level) %>%
    summarise(across(where(is.numeric), ~ mean(.x, na.rm = TRUE))) %>%
    group_by(diagnosis) %>%
    summarise(across(where(is.numeric), ~ diff(.x))) %>%
    select(where(is.numeric)) %>%
    map_df(~ diff(.x))) # COMP - ASD
```

```
##                            [,1]
## (Intercept)                   0
## diagnosis1                    0
## diagnosis2                    0
## diagnosis3                    0
## level1                        0
## change1                       0
## diagnosis1:level1             2
## diagnosis2:level1             2
## diagnosis3:level1             4
## diagnosis1:change1            0
## diagnosis2:change1            0
## diagnosis3:change1            0
## level1:change1                0
## diagnosis1:level1:change1     2
## diagnosis2:level1:change1     2
## diagnosis3:level1:change1     4
```

```
e2  = hypothesis(m.alpha, "0 <
                 2*(diagnosis1:level1 + diagnosis2:level1 + 2*diagnosis3:level1) +
                 2*(diagnosis1:level1:change1 +
                    diagnosis2:level1:change1 +
                    2*diagnosis3:level1:change1)",
                 alpha = 0.025)
e2
```

```
## Hypothesis Tests for class b:
##                 Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(2*(diagnosis... < 0    -0.76      0.61    -1.95     0.43       8.77
##   Post.Prob Star
## 1       0.9
## ---
## '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.
```

```
# H4c: alpha2 COMP != ADHD
h4c.2 = hypothesis(m.alpha, "0 > -(diagnosis1 + 2*diagnosis2 + diagnosis3) +
```

```
                -(diagnosis1:level1 + 2*diagnosis2:level1 + diagnosis3:level1)",
                alpha = 0.025)
h4c.2
```

```
## Hypothesis Tests for class b:
##                 Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(-(diagnosis1... > 0     0.52      0.51    -0.48     1.51       5.76
##   Post.Prob Star
## 1      0.85
## ---
## '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.
```

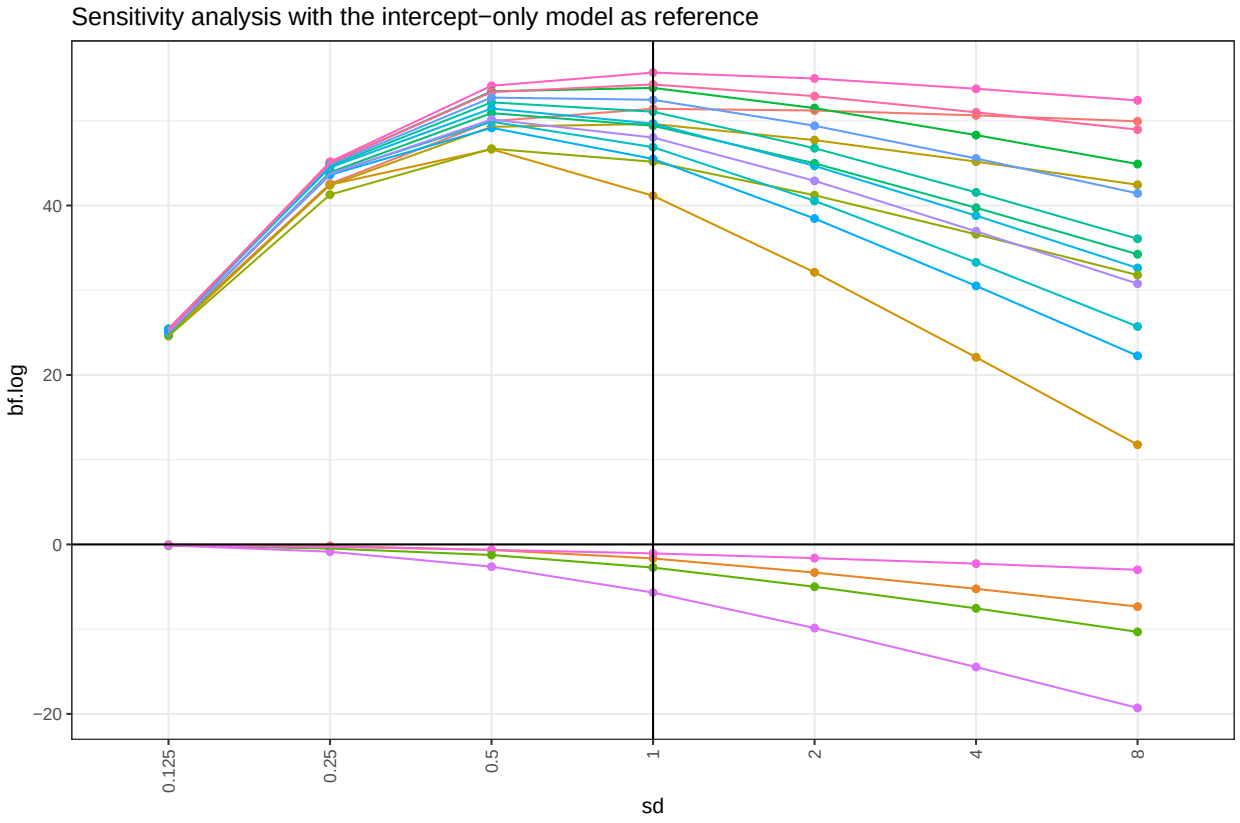
```
# H4c: alpha3 COMP != ADHD
h4c.3 = hypothesis(m.alpha, "0 < -(diagnosis1 + 2*diagnosis2 + diagnosis3) +
                (diagnosis1:level1 + 2*diagnosis2:level1 + diagnosis3:level1)",
                alpha = 0.025)
h4c.3
```

```
## Hypothesis Tests for class b:
##                 Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(-(diagnosis1... < 0    -0.03      0.38    -0.77     0.73       1.13
##   Post.Prob Star
## 1      0.53
## ---
## '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.
```

Bayes factor

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

# rerun the model with more iterations for bridgesampling
set.seed(5544)
m.orig = brm(f.alpha, family = lognormal,
          df.upd, prior = priors,
          iter = 40000, warmup = 10000,
          backend = "cmdstanr", threads = threading(8),
          file = file.path(brms_dir, "m_hgf_alpha_bf"), silent = 2,
          save_pars = save_pars(all = TRUE)
          )

# 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)]
}

# 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_3int(m.orig, "diagnosis", "level", "change",
                 pr.desc,
                 main.code, # prefix for all models and MLL
                 file.path("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))
```



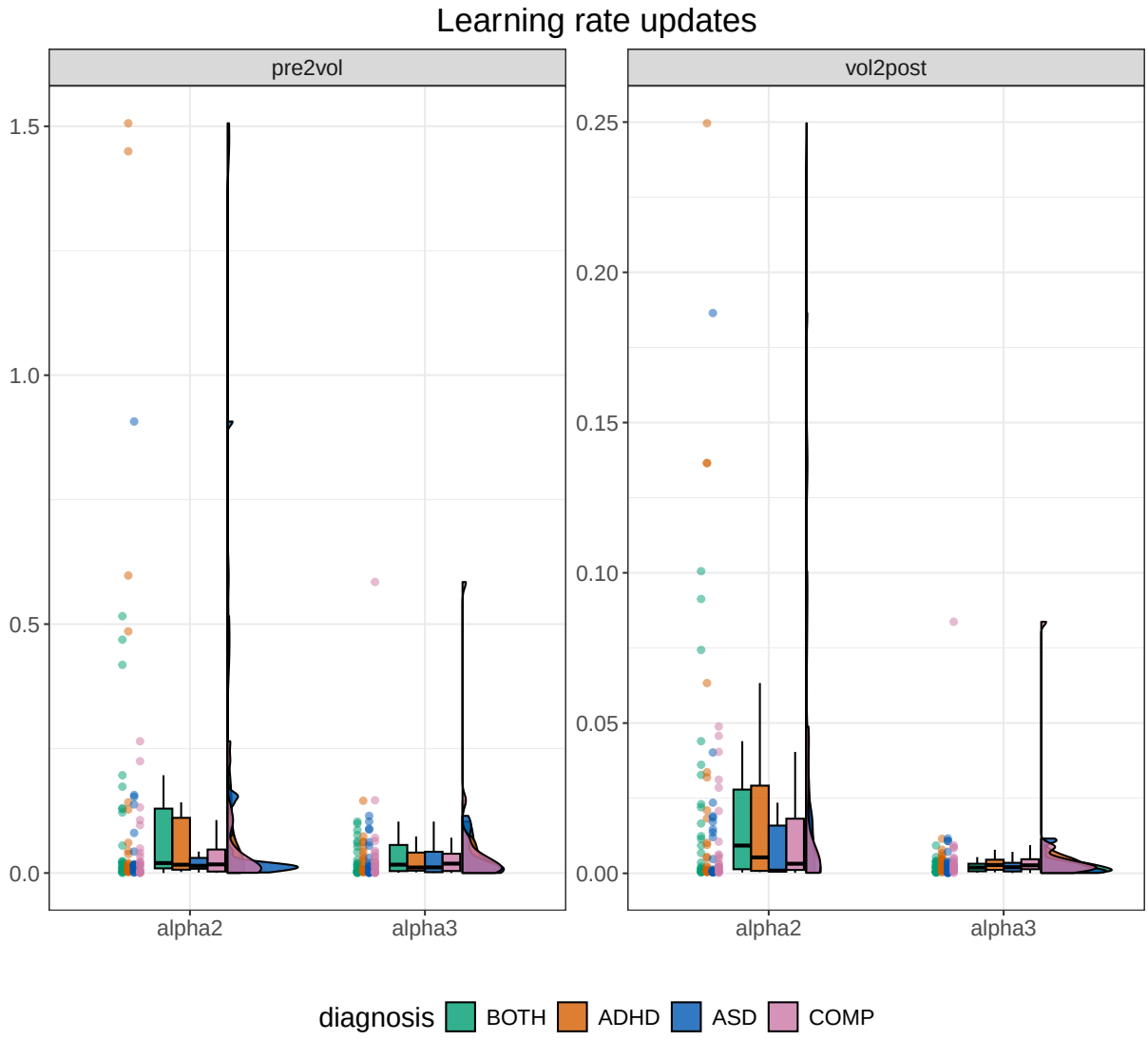

```
# print the results for the chosen priors
kable(df.pal.bf %>%
        filter(priors == "chosen" & `population-level` != "1") %>%
        arrange(desc(bf.log)) %>%
        mutate(
          bf.log.diff = bf.log - lead(bf.log),
          bf.int = interpret_bf(bf.log.diff, log = T)
        ))
```

population-level	bf.log	priors	bf.log.diff	bf.int
level + change	55.6853	chosen	1.4004	moderate evidence in favour of
level + change + level:change	54.2849	chosen	0.4041	anecdotal evidence in favour of
diagnosis + level + change	53.8808	chosen	1.4063	moderate evidence in favour of
diagnosis + level + change + level:change	52.4745	chosen	1.0601	anecdotal evidence in favour of
change	51.4144	chosen	0.3486	anecdotal evidence in favour of
diagnosis + level + change + diagnosis:level	51.0658	chosen	1.3922	moderate evidence in favour of
diagnosis + level + change + diagnosis:level + level:change	49.6736	chosen	0.0521	anecdotal evidence in favour of
diagnosis + change	49.6215	chosen	0.2092	anecdotal evidence in favour of

population-level	bf.log	priors	bf.log.diff	bf.int
diagnosis + level + change + diagnosis:change	49.4123	chosen	1.3900	moderate evidence in favour of
diagnosis + level + change + level:change + diagnosis:change	48.0223	chosen	1.1388	moderate evidence in favour of
diagnosis + level + change + diagnosis:level + diagnosis:change	46.8835	chosen	1.3890	moderate evidence in favour of
diagnosis + level + change + diagnosis:level + level:change + diagnosis:change	45.4945	chosen	0.3333	anecdotal evidence in favour of
diagnosis + change + diagnosis:change	45.1612	chosen	4.0120	very strong evidence in favour of
diagnosis * level * change	41.1492	chosen	42.2100	extreme evidence in favour of
level	-1.0608	chosen	0.5959	anecdotal evidence in favour of
diagnosis	-1.6567	chosen	1.0734	anecdotal evidence in favour of
diagnosis + level	-2.7301	chosen	2.9538	strong evidence in favour of
diagnosis + level + diagnosis:level	-5.6839	chosen	NA	

S4.6: Plots for learning rate updates

```
# rain cloud plot

df.upd %>%
  ggplot(aes(level, value, 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))) +
  scale_fill_manual(values = custom.col) +
  scale_color_manual(values = custom.col) +
  facet_wrap(. ~ change, scales = "free") +
  labs(title = "Learning rate updates", x = "", y = "") +
  theme_bw() +
  theme(legend.position = "bottom", plot.title = element_text(hjust = 0.5),
        legend.direction = "horizontal", text = element_text(size = 15))
```



S4.7: HGF model after Lawson et al. (2017)

Preparation

Again, we start by loading the extracted parameters. We will not redo the SBCs, since we will use the same models just with slightly different priors.

```
# get data
df.hgf = read_csv(file.path("HGF_results/main", "Lawson2017_results.csv")) %>%
  mutate_if(is.character, as.factor)

# extract the absolute changes in learning rate for the phases
df.upd = read_csv(file.path("HGF_results/main", "Lawson2017_traj.csv")) %>%
  select(subID, diagnosis, trl, alpha2, alpha3) %>%
  mutate(
    phase = case_when(
      trl < 73  ~ "pre",
      trl > 264 ~ "post",
```

```
      trl < 145 ~ "vol1",
      trl > 192 ~ "vol2"
    )
  ) %>%
  drop_na() %>%
  group_by(subID, diagnosis, phase) %>%
  summarise(
    alpha2 = median(alpha2),
    alpha3 = median(alpha3)
  ) %>%
  pivot_wider(names_from = phase, id_cols = c(subID, diagnosis),
              values_from = starts_with("alpha")) %>%
  group_by(subID, diagnosis) %>%
  summarise(
    alpha2_pre2vol  = abs(alpha2_pre  - alpha2_vol1),
    alpha2_vol2post = abs(alpha2_post - alpha2_vol2),
    alpha3_pre2vol  = abs(alpha3_pre  - alpha3_vol1),
    alpha3_vol2post = abs(alpha3_post - alpha3_vol2)
  ) %>%
  pivot_longer(cols = starts_with("alpha")) %>%
  separate(name, into = c("level", "change")) %>%
  mutate_if(is.character, as.factor)

# check whether there are LME differences between the diagnostic groups
kable(df.hgf %>% group_by(diagnosis) %>% shapiro_test(LME)) # all normally distributed
```

diagnosis	variable	statistic	p
ADHD	LME	0.9655285	0.6082406
ASD	LME	0.9566490	0.3991500
BOTH	LME	0.9706004	0.7033793
COMP	LME	0.9871230	0.9884383

```
aov.lme = anovaBF(LME ~ diagnosis, data = df.hgf)
aov.lme@bayesFactor
```

```
##                    bf        error                      time          code
## diagnosis -0.9246943 7.220503e-07 Fri Mar 21 15:29:25 2025 3126e126abc71
```

There is anecdotal evidence against a difference in LME between diagnostic groups. This suggests that the HGF model fit comparably well to the subjects of the different groups. Therefore, we move on to analyse its parameters.

For this specific model, the following observation model was used:

$$\log RT = \beta_0 + \beta_1 \times surprise_{Shannon} + \beta_2 \times uncertainty_{outcome} + \beta_3 \times uncertainty_{probability} + \beta_4 \times volatility_{phasic}$$

This model is deployed in the TAPAS toolbox and, to our knowledge, was used in Lawson et al. (2021) with the HGF (not eHGF) as the perception model.

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

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

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

```
contrasts(df.upd$diagnosis) = contr.sum(4)
contrasts(df.upd$diagnosis)
```

```
##      [,1] [,2] [,3]
## BOTH    1    0    0
## ADHD    0    1    0
## ASD     0    0    1
## COMP   -1   -1   -1
```

```
contrasts(df.upd$change) = contr.sum(2)
contrasts(df.upd$change)
```

```
##          [,1]
## pre2vol     1
## vol2post   -1
```

```
contrasts(df.upd$level) = contr.sum(2)
contrasts(df.upd$level)
```

```
##        [,1]
## alpha2    1
## alpha3   -1
```

Phasic volatility

```
# model formula
f.vol = brms::bf( be4 ~ diagnosis )

# set weakly informative priors
priors = c(
  prior(normal(0, 4),  class = Intercept),
  prior(normal(0, 0.50),  class = sigma),
  prior(normal(0, 0.25),  class = b)
)

# change Intercept based on empirical priors used in the HGF model
priors = priors %>%
  mutate(
    prior = if_else(
      class == "Intercept",
      gsub("\\(.*,", paste0("(", mean(df.hgf$be4mu), ", "), prior), prior),
    prior = if_else(
      class == "Intercept",
      gsub(" .*\\)", paste0(" ", mean(df.hgf$be4sa), ")"), prior), prior)
  )

kable(priors)
```

prior	class	coef	group	resp	dpar	nlpar	lb	ub	source
normal(0, 4)	Intercept						NA	NA	user
normal(0, 0.5)	sigma						NA	NA	user

prior	class	coef	group	resp	dpar	nlpar	lb	ub	source
normal(0, 0.25)	b						NA	NA	user

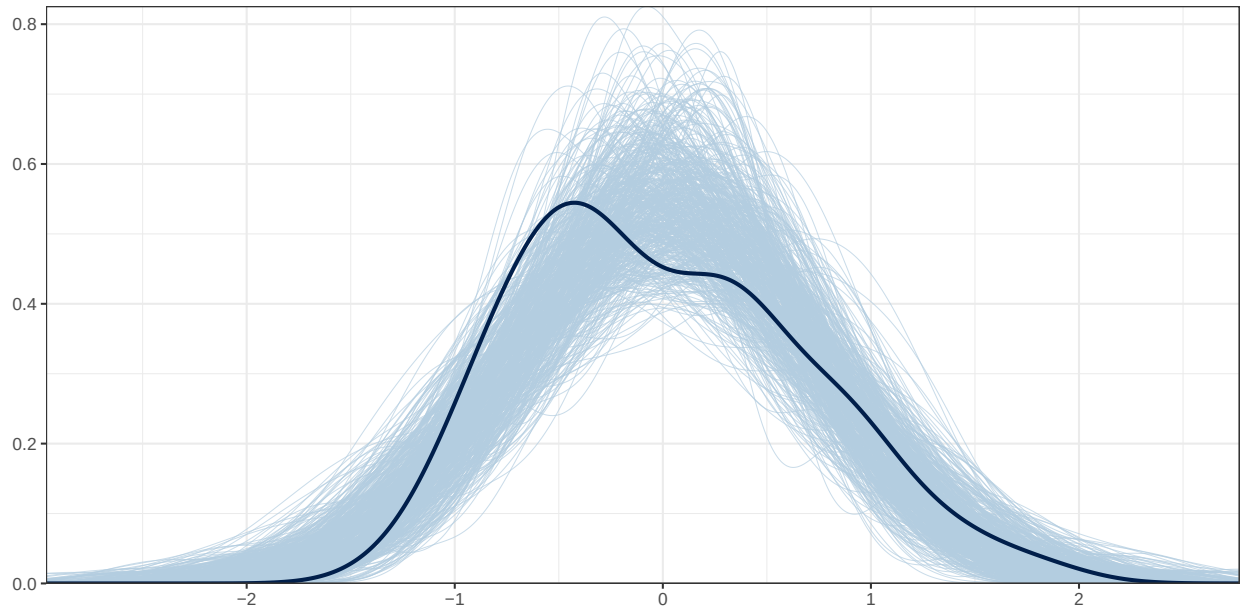
```
# fit the final model
set.seed(2684)
m.vol = brm(f.vol,
            df.hgf, prior = priors,
            iter = iter, warmup = warm,
            backend = "cmdstanr", threads = threading(8),
            file = file.path(brms_dir, "m_hgf_vol-Law17"),
            save_pars = save_pars(all = TRUE)
            )
rstan::check_hmc_diagnostics(m.vol$fit)
```

```
##
## Divergences:
##
## Tree depth:
##
## Energy:
```

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

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

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



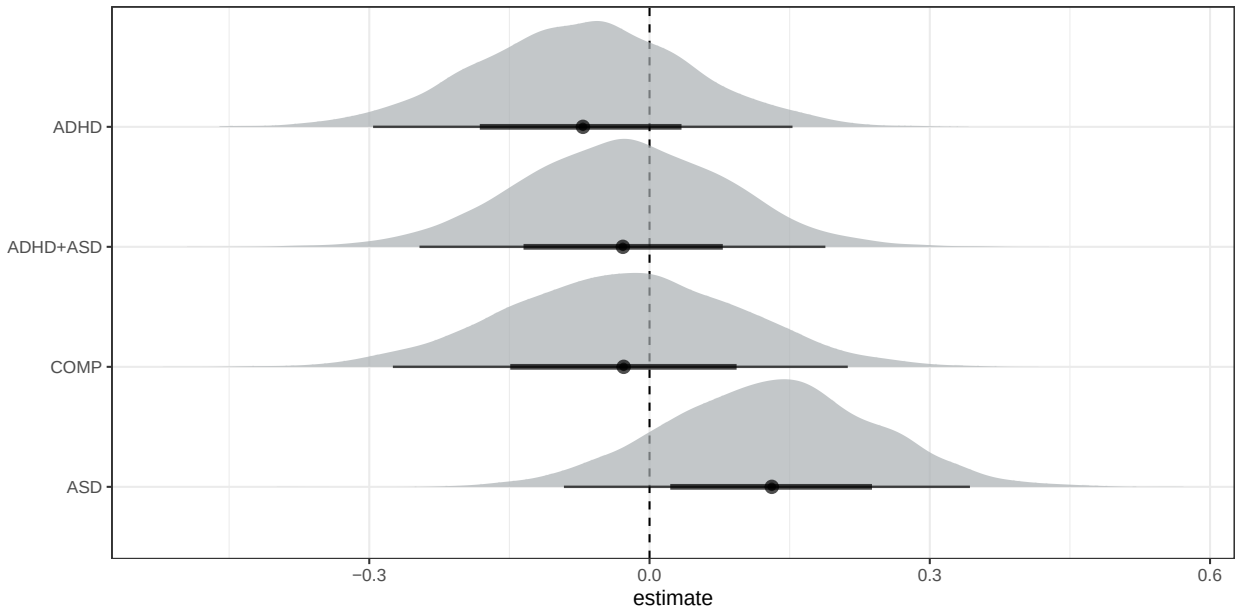

```
# print a summary
summary(m.vol)
```

```
##  Family: gaussian
```

```
##   Links: mu = identity; sigma = identity
## Formula: be4 ~ diagnosis
##    Data: df.hgf (Number of observations: 90)
##   Draws: 4 chains, each with iter = 3000; warmup = 1000; thin = 1;
##          total post-warmup draws = 8000
##
## Regression Coefficients:
##            Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept      0.01      0.07    -0.13     0.15 1.00    10120     6100
## diagnosis1    -0.07      0.11    -0.30     0.15 1.00     8764     5813
## diagnosis2     0.13      0.11    -0.09     0.34 1.00     9110     6265
## diagnosis3    -0.03      0.11    -0.25     0.19 1.00     9178     5964
##
## Further Distributional Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma     0.69      0.05     0.59     0.80 1.00     9770     5841
##
## 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).
```

```
# get the estimates and compute group comparisons
df.m.vol = as_draws_df(m.vol) %>%
  select(starts_with("b_")) %>%
  mutate(
    b_COMP = - b_diagnosis1 - b_diagnosis2 - b_diagnosis3,
    ASD    = b_Intercept + b_diagnosis2,
    ADHD   = b_Intercept + b_diagnosis1,
    BOTH   = b_Intercept + b_diagnosis3,
    COMP   = b_Intercept + b_COMP,
  )

# plot the posterior distributions
df.m.vol %>%
  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, "diagnosis1", "ADHD"),
    coef = str_replace_all(coef, "diagnosis2", "ASD"),
    coef = str_replace_all(coef, "diagnosis3", "ADHD+ASD"),
    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_light, c_dark)) + theme(legend.position = "none")
```




```
# COMP < ASD
hypothesis(m.vol, "0 < diagnosis1 + 2*diagnosis2 + diagnosis3",
                  alpha = 0.025)
```

```
## Hypothesis Tests for class b:
##                 Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(diagnosis1+2... < 0    -0.16       0.2    -0.55     0.23       3.74
##   Post.Prob Star
## 1      0.79
## ---
## '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.
```

Third-level tonic volatility

```
# model formula
f.om3 = brms::bf( om3 ~ diagnosis )

# set weakly informative priors
priors = c(
  prior(normal(0, 4),  class = Intercept),
  prior(normal(0, 0.50),  class = sigma),
  prior(normal(0, 0.25),  class = b)
)

# change Intercept based on empirical priors used in the HGF model
priors = priors %>%
```

```
  mutate(
    prior = if_else(
      class == "Intercept",
      gsub("\\(.*,", paste0("(", mean(df.hgf$om3mu), ", "), prior), prior),
    prior = if_else(
      class == "Intercept",
      gsub(" .*\\)", paste0(" ", mean(df.hgf$om3sa), ")"), prior), prior)
  )

kable(priors)
```

prior	class	coef	group	resp	dpar	nlpar	lb	ub	source
normal(-6, 16)	Intercept						NA	NA	user
normal(0, 0.5)	sigma						NA	NA	user
normal(0, 0.25)	b						NA	NA	user

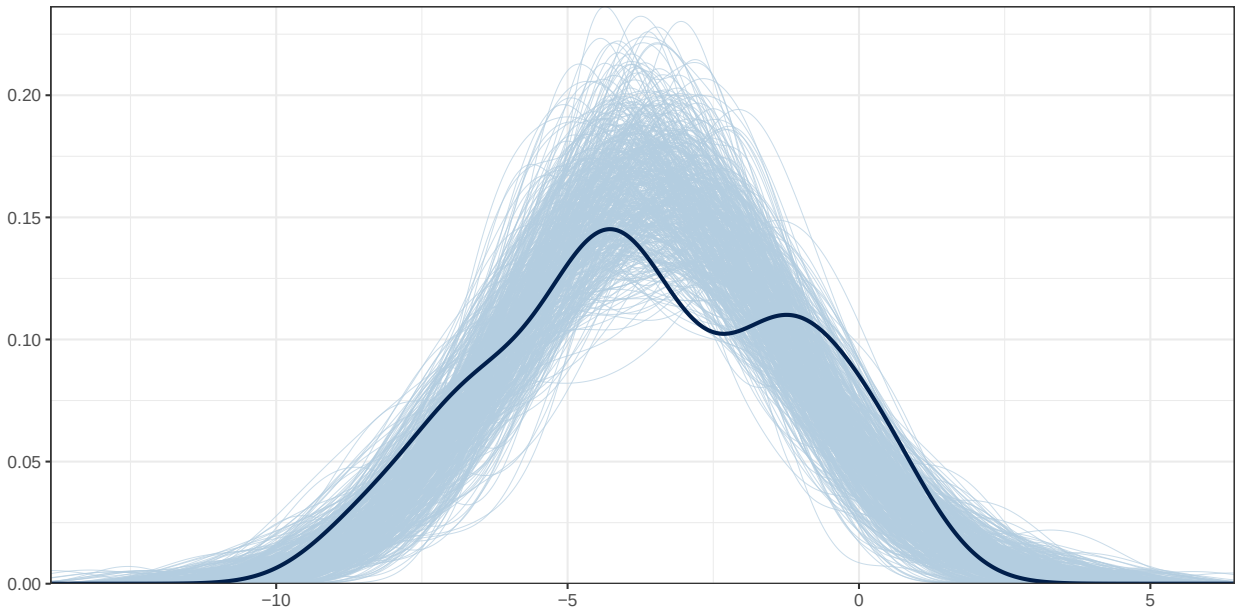
```
# fit the final model
set.seed(2684)
m.om3 = brm(f.om3,
            df.hgf, prior = priors,
            iter = iter, warmup = warm,
            backend = "cmdstanr", threads = threading(8),
            file = file.path(brms_dir, "m_hgf_om3-Law17"),
            save_pars = save_pars(all = TRUE)
            )
rstan::check_hmc_diagnostics(m.om3$fit)
```

```
##
## Divergences:
##
## Tree depth:
##
## Energy:
```

```
# check that rhats are below 1.01
sum(brms::rhat(m.om3) >= 1.01, na.rm = T)
```

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

```
# check the fit of the predicted data compared to the real data
pp_check(m.om3, ndraws = nsim) +
  theme_bw() + theme(legend.position = "none")
```

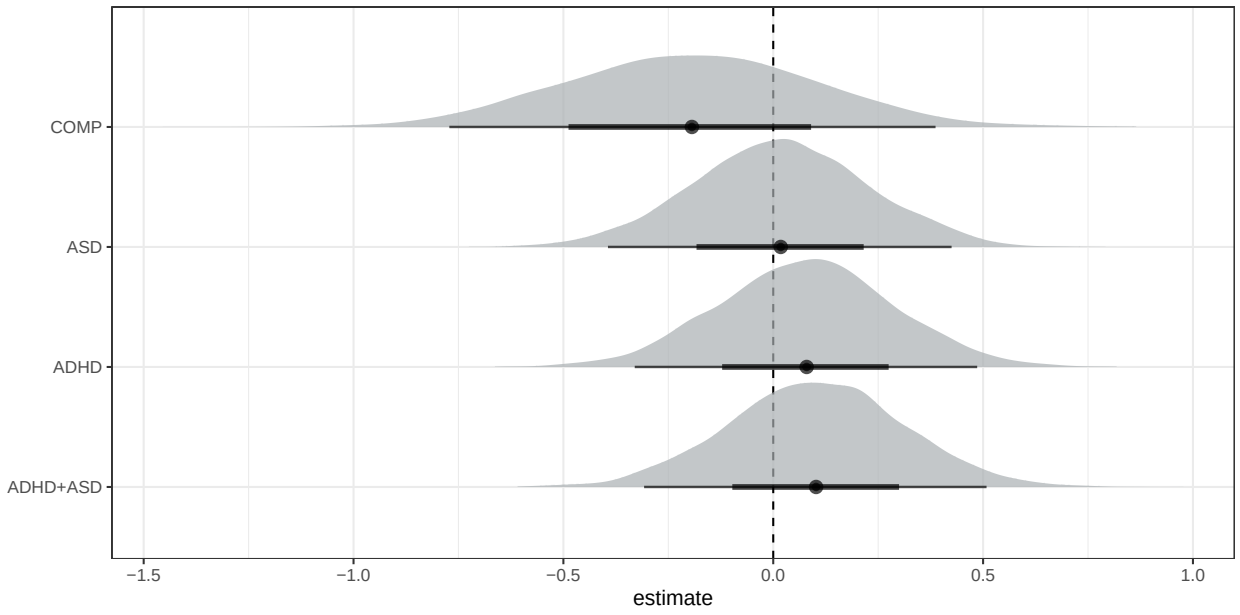



```
# print a summary
summary(m.om3)
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: om3 ~ diagnosis
##    Data: df.hgf (Number of observations: 90)
##   Draws: 4 chains, each with iter = 3000; warmup = 1000; thin = 1;
##          total post-warmup draws = 8000
##
## Regression Coefficients:
##            Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept     -3.63      0.24    -4.11    -3.17 1.00     9176     5449
## diagnosis1     0.08      0.21    -0.33     0.49 1.00     9103     6247
## diagnosis2     0.02      0.21    -0.39     0.42 1.00     9931     6392
## diagnosis3     0.10      0.21    -0.31     0.51 1.00     9525     6418
##
## Further Distributional Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma     2.31      0.14     2.05     2.61 1.00     9088     6247
##
## 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).
```

```
# get the estimates and compute group comparisons
df.m.om3 = as_draws_df(m.om3) %>%
  select(starts_with("b_")) %>%
  mutate(
    b_COMP = - b_diagnosis1 - b_diagnosis2 - b_diagnosis3,
    ASD    = b_Intercept + b_diagnosis2,
    ADHD   = b_Intercept + b_diagnosis1,
    BOTH   = b_Intercept + b_diagnosis3,
    COMP   = b_Intercept + b_COMP,
```

```
  )

# plot the posterior distributions
df.m.om3 %>%
  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, "diagnosis1", "ADHD"),
    coef = str_replace_all(coef, "diagnosis2", "ASD"),
    coef = str_replace_all(coef, "diagnosis3", "ADHD+ASD"),
    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_light, c_dark)) + theme(legend.position = "none")
```




```
# COMP < ASD
hypothesis(m.om3, "0 < diagnosis1 + 2*diagnosis2 + diagnosis3",
                  alpha = 0.025)
```

```
## Hypothesis Tests for class b:
##                 Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
```

```
## 1 (0)-(diagnosis1+2... < 0    -0.21      0.44    -1.06     0.65       2.18
##   Post.Prob Star
## 1      0.69
## ---
## '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.
```

Second-level tonic volatility

```
# model formula
f.om2 = brms::bf( om2 ~ diagnosis )

# set weakly informative priors
priors = c(
  prior(normal(0, 4),  class = Intercept),
  prior(normal(0, 0.50),  class = sigma),
  prior(normal(0, 0.25),  class = b)
)

# change Intercept based on empirical priors used in the HGF model
priors = priors %>%
  mutate(
    prior = if_else(
      class == "Intercept",
      gsub("\\(.*,", paste0("(", mean(df.hgf$om2mu), ", "), prior), prior),
    prior = if_else(
      class == "Intercept",
      gsub(" .*\\)", paste0(" ", mean(df.hgf$om2sa), ")"), prior), prior)
  )

kable(priors)
```

prior	class	coef	group	resp	dpar	nlpar	lb	ub	source
normal(-5, 16)	Intercept						NA	NA	user
normal(0, 0.5)	sigma						NA	NA	user
normal(0, 0.25)	b						NA	NA	user

```
# fit the final model
set.seed(2684)
m.om2 = brm(f.om2,
            df.hgf, prior = priors,
            iter = iter, warmup = warm,
            backend = "cmdstanr", threads = threading(8),
            file = file.path(brms_dir, "m_hgf_om2-Law17"),
            save_pars = save_pars(all = TRUE)
            )
rstan::check_hmc_diagnostics(m.om2$fit)
```

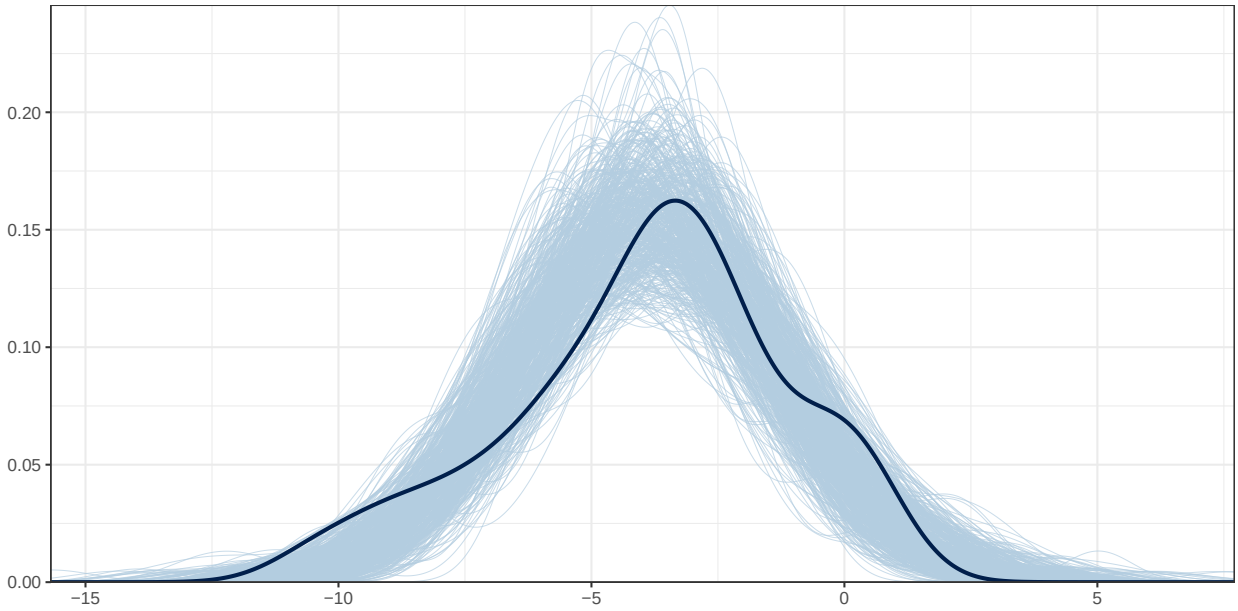
```
##
## Divergences:
##
```

```
## Tree depth:
##
## Energy:
```

```
# check that rhats are below 1.01
sum(brms::rhat(m.om2) >= 1.01, na.rm = T)
```

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

```
# check the fit of the predicted data compared to the real data
pp_check(m.om2, ndraws = nsim) +
  theme_bw() + theme(legend.position = "none")
```

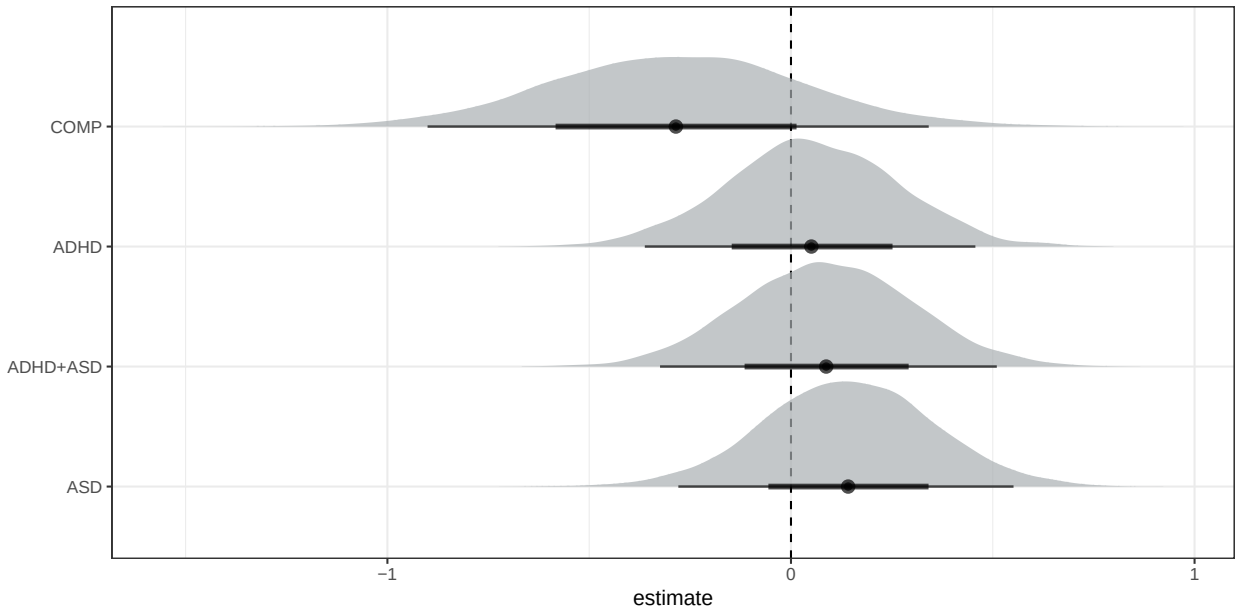



```
# print a summary
summary(m.om2)
```

```
##  Family: gaussian
##   Links: mu = identity; sigma = identity
## Formula: om2 ~ diagnosis
##    Data: df.hgf (Number of observations: 90)
##   Draws: 4 chains, each with iter = 3000; warmup = 1000; thin = 1;
##          total post-warmup draws = 8000
##
## Regression Coefficients:
##            Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept     -3.90      0.26    -4.41    -3.39 1.00     9509     6224
## diagnosis1     0.05      0.21    -0.36     0.46 1.00     9254     6606
## diagnosis2     0.14      0.21    -0.28     0.55 1.00     9668     6705
## diagnosis3     0.09      0.21    -0.32     0.51 1.00     9404     6283
##
## Further Distributional Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma     2.44      0.15     2.17     2.74 1.00     8853     6277
##
## 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).
```

```r
# get the estimates and compute group comparisons
df.m.om2 = as_draws_df(m.om2) %>%
  select(starts_with("b_")) %>%
  mutate(
    b_COMP = - b_diagnosis1 - b_diagnosis2 - b_diagnosis3,
    ASD    = b_Intercept + b_diagnosis2,
    ADHD   = b_Intercept + b_diagnosis1,
    BOTH   = b_Intercept + b_diagnosis3,
    COMP   = b_Intercept + b_COMP,
  )

# plot the posterior distributions
df.m.om2 %>%
  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, "diagnosis1", "ADHD"),
    coef = str_replace_all(coef, "diagnosis2", "ASD"),
    coef = str_replace_all(coef, "diagnosis3", "ADHD+ASD"),
    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_light, c_dark)) + theme(legend.position = "none")
```



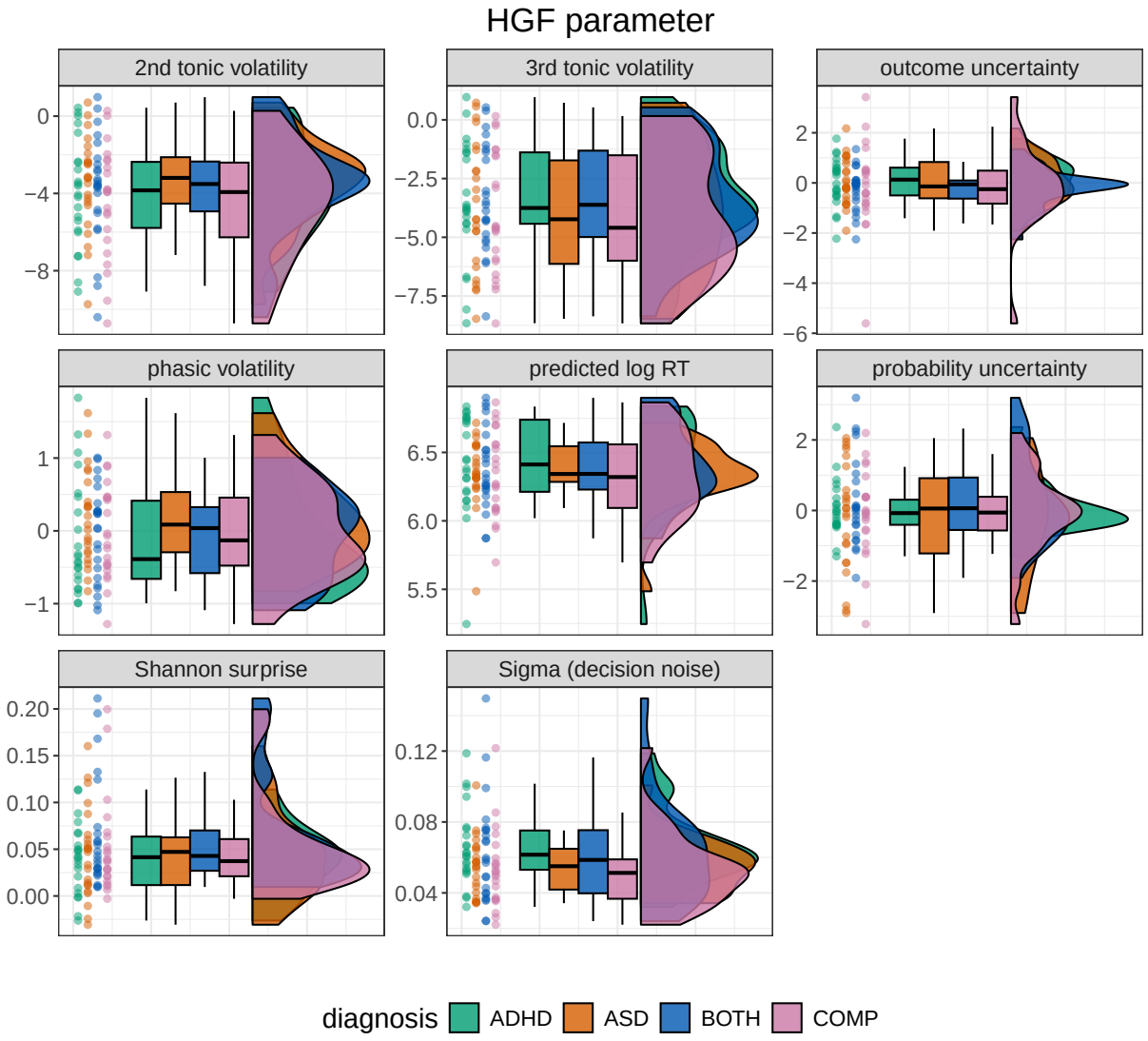

```
# COMP != ADHD
hypothesis(m.om2, "0 < 2*diagnosis1 + diagnosis2 + diagnosis3", alpha = 0.025)
```

```
## Hypothesis Tests for class b:
##                 Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(2*diagnosis1... < 0    -0.34      0.46    -1.24     0.57       3.35
##   Post.Prob Star
## 1      0.77
## ---
## '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.
```

Plots for all HGF parameters

```
# rain cloud plot

df.hgf %>%
  select(subID, diagnosis, be0, be1, be2, be3, be4, ze, om2, om3) %>% #
  pivot_longer(cols = c(be0, be1, be2, be3, be4, ze, om2, om3),
               names_to = "parameter") %>%
  mutate(
    parameter = case_match(parameter,
                           "be0" ~ "predicted log RT",
                           "be1" ~ "Shannon surprise",
                           "be2" ~ "outcome uncertainty",
                           "be3" ~ "probability uncertainty",
                           "be4" ~ "phasic volatility",
                           "ze"  ~ "Sigma (decision noise)",
                           "om2" ~ "2nd tonic volatility",
                           "om3" ~ "3rd tonic volatility"
                           )
  ) %>%
```

```
  ggplot(aes(x = 1, y = value, 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))) +
  scale_fill_manual(values = custom.col) +
  scale_color_manual(values = custom.col) +
  facet_wrap(. ~ parameter, scales = "free", ncol = 3) +
  labs(title = "HGF parameter", x = "", y = "") +
  theme_bw() +
  theme(legend.position = "bottom", plot.title = element_text(hjust = 0.5),
        text = element_text(size = 15), axis.text.x=element_blank(),
        axis.ticks.x=element_blank(), legend.direction = "horizontal")
```



Learning rate update

```
# model formula
f.alpha = brms::bf( value ~ diagnosis * level * change + (level + change | subID) )

# set weakly informative priors taking Lawson 2017 into consideration
priors = c(
  prior(normal(-5, 2),    class = Intercept),
  prior(normal(0.5, 0.5), class = sigma),
  prior(normal(0.5, 0.5), class = sd),
  prior(lkj(2),           class = cor),
  prior(normal(0, 0.5),   class = b)
)

# fit the final model
set.seed(2684)
m.alpha = brm(f.alpha, family = lognormal,
              df.upd, prior = priors,
```

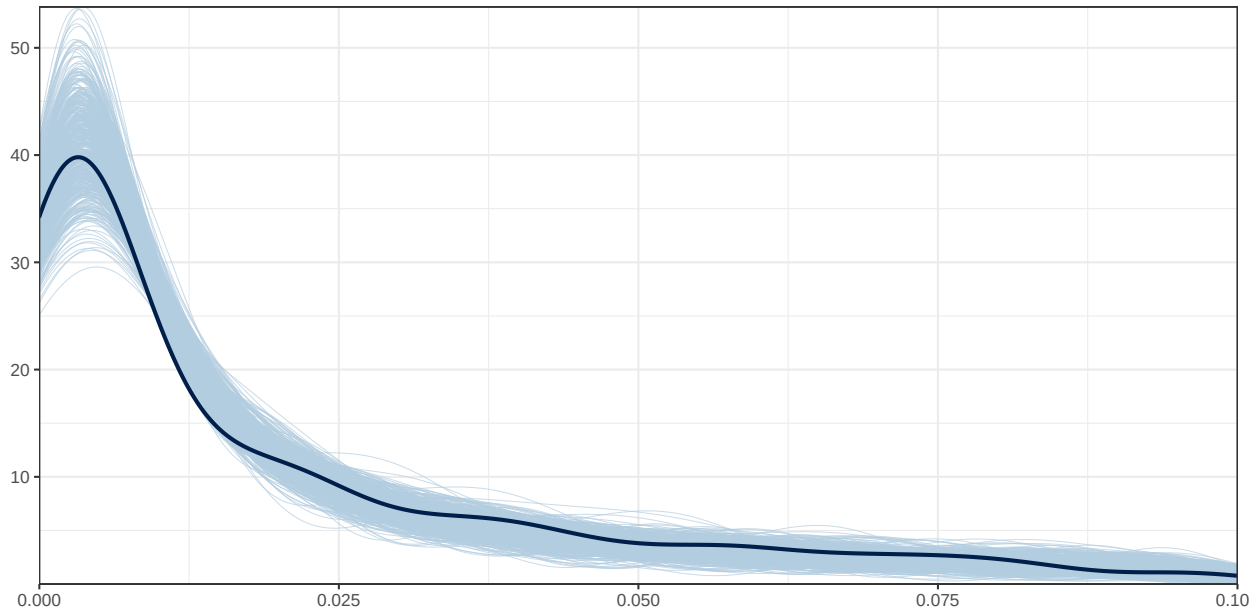
```
            iter = iter, warmup = warm,
            backend = "cmdstanr", threads = threading(8),
            file = file.path(brms_dir, "m_hgf_alpha-Law17"),
            save_pars = save_pars(all = TRUE)
            )
rstan::check_hmc_diagnostics(m.alpha$fit)
```

```
##
## Divergences:
##
## Tree depth:
##
## Energy:
```

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

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

```
# check the fit of the predicted data compared to the real data
pp_check(m.alpha, ndraws = nsim) +
  theme_bw() + theme(legend.position = "none") + xlim(0, 0.10)
```



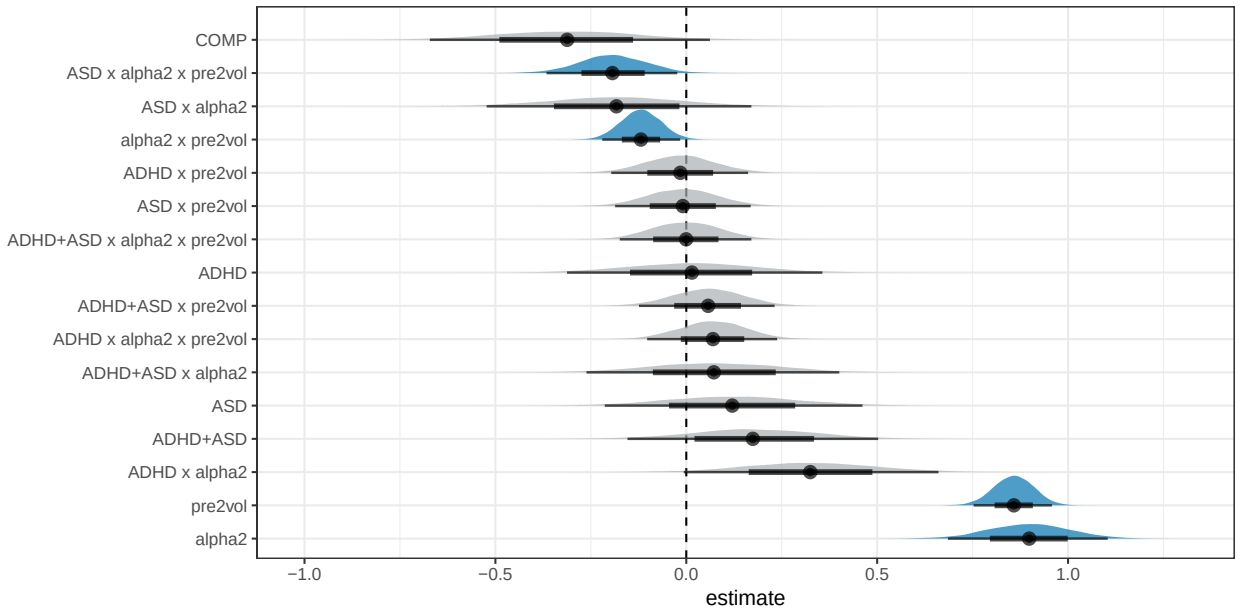

```
# print a summary
summary(m.alpha)
```

```
##  Family: lognormal
##   Links: mu = identity; sigma = identity
## Formula: value ~ diagnosis * level * change + (level + change | subID)
##    Data: df.upd (Number of observations: 360)
##   Draws: 4 chains, each with iter = 3000; warmup = 1000; thin = 1;
##          total post-warmup draws = 8000
##
## Multilevel Hyperparameters:
```

```
## ~subID (Number of levels: 90)
##                           Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS
## sd(Intercept)                 0.90      0.09     0.74     1.08 1.00     2616
## sd(level1)                    0.91      0.09     0.75     1.09 1.00     2837
## sd(change1)                   0.12      0.07     0.01     0.25 1.00     3098
## cor(Intercept,level1)         0.57      0.10     0.35     0.75 1.00     2015
## cor(Intercept,change1)        0.46      0.33    -0.37     0.90 1.00     7006
## cor(level1,change1)           0.23      0.33    -0.48     0.80 1.00     8614
##                           Tail_ESS
## sd(Intercept)                 3770
## sd(level1)                    4255
## sd(change1)                   3106
## cor(Intercept,level1)         3292
## cor(Intercept,change1)        5074
## cor(level1,change1)           6113
##
## Regression Coefficients:
##                              Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS
## Intercept                       -4.22      0.11    -4.42    -4.00 1.00     2078
## diagnosis1                       0.01      0.17    -0.31     0.36 1.00     2402
## diagnosis2                       0.12      0.17    -0.21     0.46 1.00     2198
## diagnosis3                       0.18      0.17    -0.15     0.50 1.00     2196
## level1                           0.90      0.11     0.69     1.10 1.00     2846
## change1                          0.86      0.05     0.75     0.96 1.00     8982
## diagnosis1:level1                0.33      0.17    -0.01     0.66 1.00     2841
## diagnosis2:level1               -0.18      0.17    -0.52     0.17 1.00     3016
## diagnosis3:level1                0.07      0.17    -0.26     0.40 1.00     2955
## diagnosis1:change1              -0.02      0.09    -0.20     0.16 1.00     7021
## diagnosis2:change1              -0.01      0.09    -0.19     0.17 1.00     7984
## diagnosis3:change1               0.06      0.09    -0.12     0.23 1.00     6875
## level1:change1                  -0.12      0.05    -0.22    -0.02 1.00    12526
## diagnosis1:level1:change1        0.07      0.09    -0.10     0.24 1.00     8610
## diagnosis2:level1:change1       -0.19      0.09    -0.37    -0.02 1.00     8680
## diagnosis3:level1:change1       -0.00      0.09    -0.17     0.17 1.00     9253
##                              Tail_ESS
## Intercept                        3509
## diagnosis1                       4105
## diagnosis2                       3894
## diagnosis3                       3657
## level1                           4032
## change1                          5754
## diagnosis1:level1                3874
## diagnosis2:level1                4359
## diagnosis3:level1                4139
## diagnosis1:change1               6081
## diagnosis2:change1               6279
## diagnosis3:change1               5846
## level1:change1                   5824
## diagnosis1:level1:change1        6662
## diagnosis2:level1:change1        5567
## diagnosis3:level1:change1        6568
##
## Further Distributional Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
```

```
## sigma     0.98      0.05     0.88     1.09 1.00     2628     3639
##
## 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).
```

```
# get the estimates and compute group comparisons
df.m.alpha = as_draws_df(m.alpha) %>%
  select(starts_with("b_")) %>%
  mutate(
    b_COMP = - b_diagnosis1 - b_diagnosis2 - b_diagnosis3,
    ASD    = b_Intercept + b_diagnosis2,
    ADHD   = b_Intercept + b_diagnosis1,
    BOTH   = b_Intercept + b_diagnosis3,
    COMP   = b_Intercept + b_COMP,
  )

# plot the posterior distributions
df.m.alpha %>%
  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, "level1", "alpha2"),
    coef = str_replace_all(coef, "change1", "pre2vol"),
    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")
```




```
# alpha3 ASD > COMP
hypothesis(m.alpha, "0 > -(diagnosis1 + diagnosis2 + 2*diagnosis3) +
           (diagnosis1:level1 + diagnosis2:level1 + 2*diagnosis3:level1)",
           alpha = 0.025)
```

```
## Hypothesis Tests for class b:
##                 Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(-(diagnosis1... > 0      0.2      0.31    -0.41      0.8       2.86
##   Post.Prob Star
## 1      0.74
## ---
## '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.
```

```
# alpha2 COMP > ASD
hypothesis(m.alpha, "0 < -(diagnosis1 + diagnosis2 + 2*diagnosis3) +
           -(diagnosis1:level1 + diagnosis2:level1 + 2*diagnosis3:level1)",
           alpha = 0.025)
```

```
## Hypothesis Tests for class b:
##                 Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(-(diagnosis1... < 0     0.78      0.49    -0.18     1.72       0.06
##   Post.Prob Star
## 1      0.05
## ---
## '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.
```

```
# alpha2 COMP != ADHD
hypothesis(m.alpha, "0 > -(diagnosis1 + 2*diagnosis2 + diagnosis3) +
           -(diagnosis1:level1 + 2*diagnosis2:level1 + diagnosis3:level1)",
```

```
          alpha = 0.025)
```

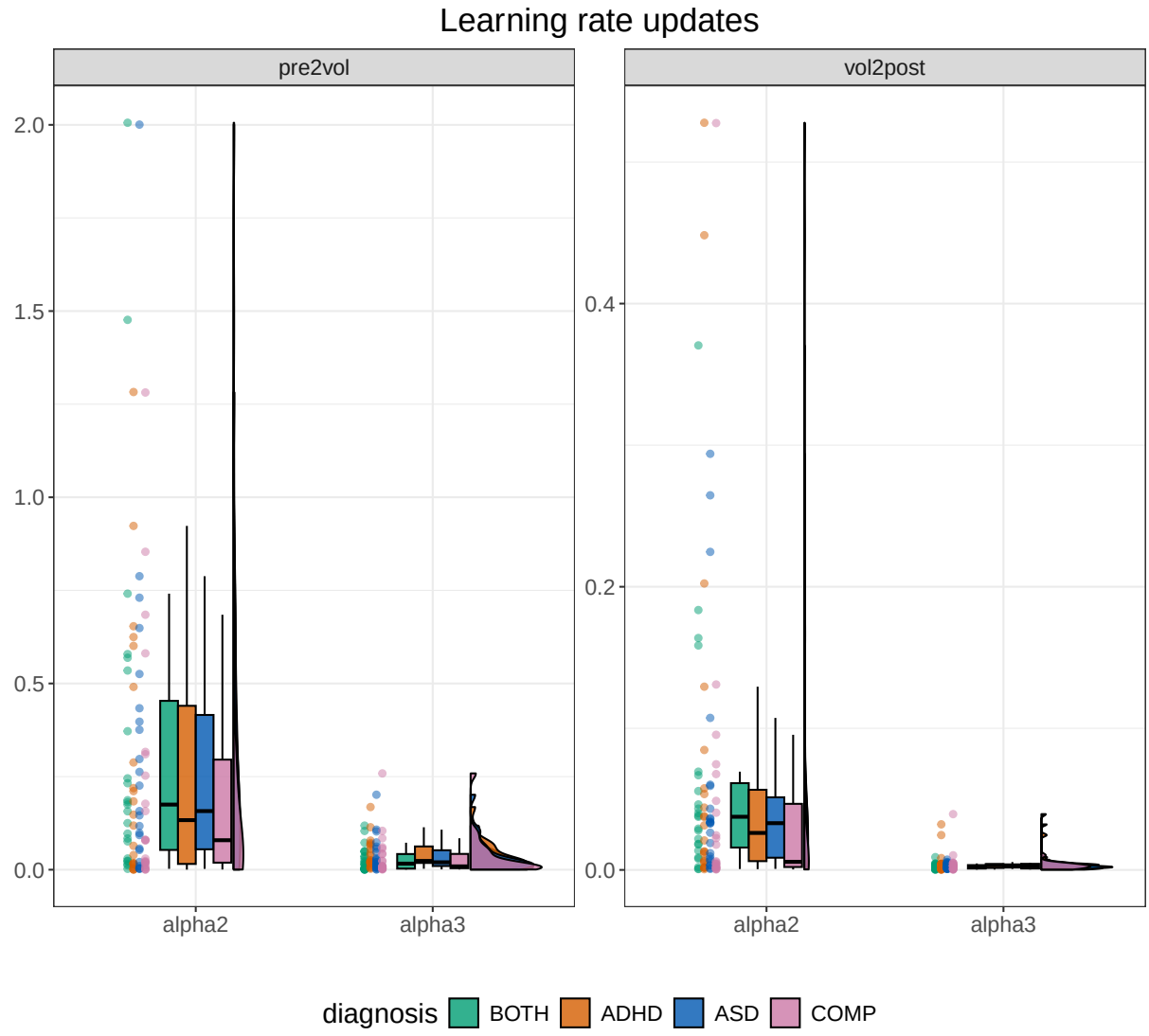
```
## Hypothesis Tests for class b:
##                 Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(-(diagnosis1... > 0     0.47       0.5    -0.53     1.43       4.85
##   Post.Prob Star
## 1      0.83
## ---
## 'CI': 95%-CI for one-sided and 97.5%-CI for two-sided hypotheses.
## '*': 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.
```

```
# alpha3 COMP != ADHD
hypothesis(m.alpha, "0 > -(diagnosis1 + 2*diagnosis2 + diagnosis3) +
          (diagnosis1:level1 + 2*diagnosis2:level1 + diagnosis3:level1)",
          alpha = 0.025)
```

```
## Hypothesis Tests for class b:
##                 Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(-(diagnosis1... > 0      0.4      0.32    -0.24     1.04       8.59
##   Post.Prob Star
## 1       0.9
## ---
## '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.
```

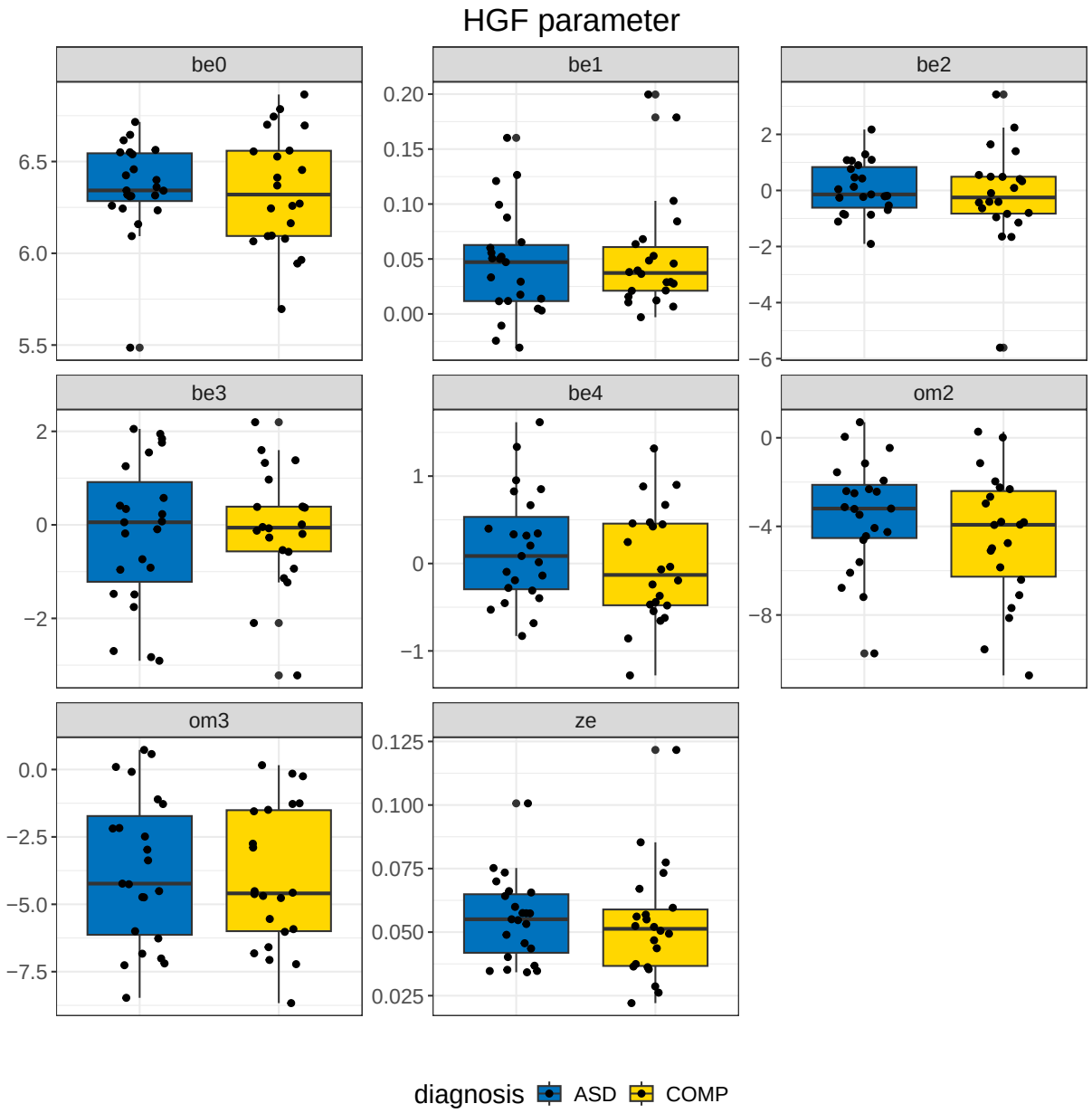
Plots for learning rate updates

```
# rain cloud plot

df.upd %>%
  ggplot(aes(level, value, 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))) +
  scale_fill_manual(values = custom.col) +
  scale_color_manual(values = custom.col) +
  facet_wrap(. ~ change, scales = "free") +
  labs(title = "Learning rate updates", x = "", y = "") +
  theme_bw() +
  theme(legend.position = "bottom", plot.title = element_text(hjust = 0.5),
        legend.direction = "horizontal", text = element_text(size = 15))
```



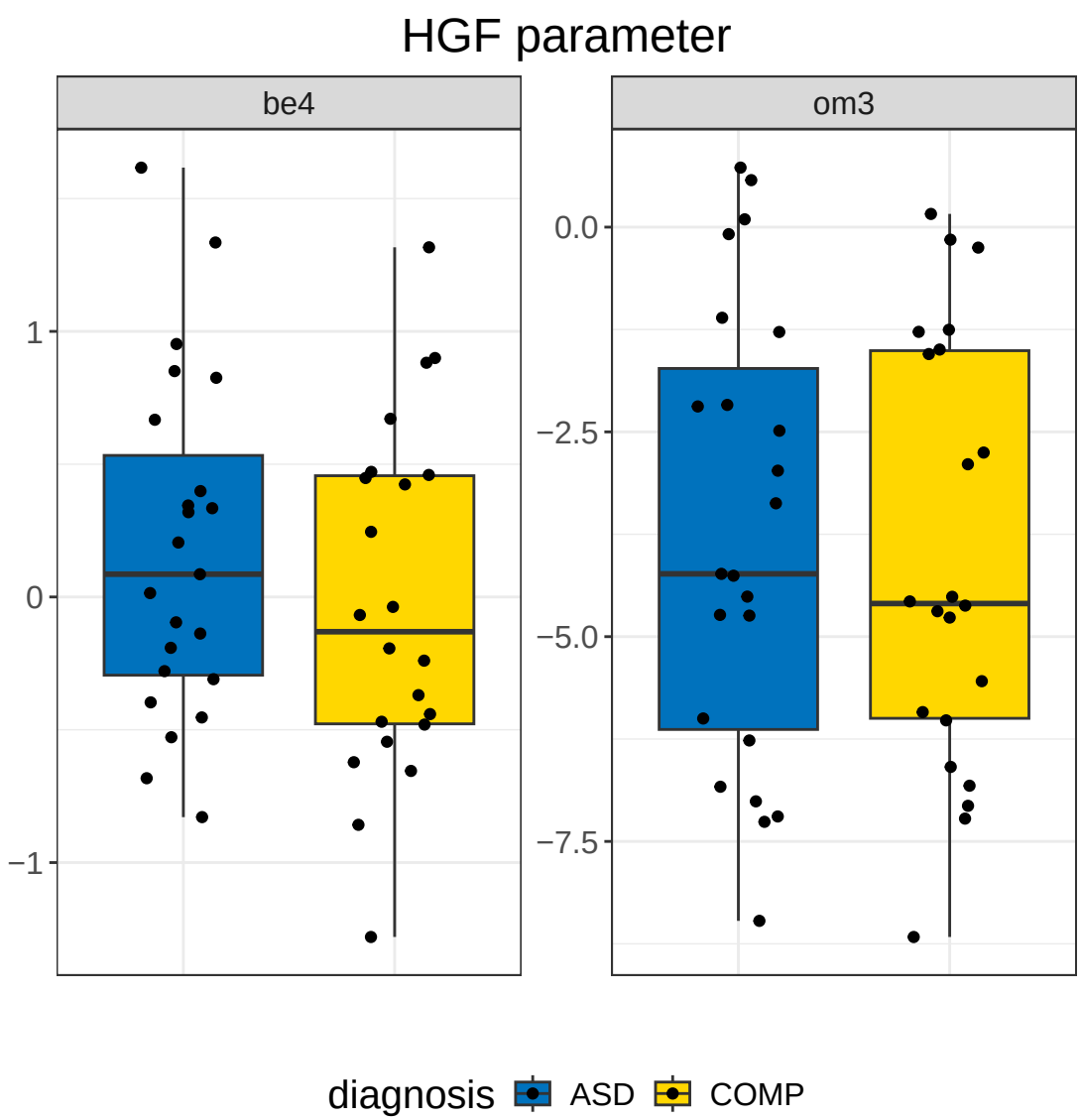
Plots focusing on ASD and COMP

```
df.hgf %>%
  filter(diagnosis %in% c("ASD", "COMP")) %>%
  select(subID, diagnosis, be0, be1, be2, be3, be4, ze, om2, om3) %>% #
  pivot_longer(cols = c(be0, be1, be2, be3, be4, ze, om2, om3),
               names_to = "parameter") %>%
  ggplot(aes(x = diagnosis, y = value, fill = diagnosis))  +
  geom_boxplot() +
  geom_jitter(width = 0.2) +
  scale_fill_manual(values = c("#0072bd", "#ffd700")) +
  facet_wrap(. ~ parameter, scales = "free", ncol = 3) +
  labs(title = "HGF parameter", x = "", y = "") +
  theme_bw() +
  theme(legend.position = "bottom", plot.title = element_text(hjust = 0.5),
        text = element_text(size = 15), axis.text.x=element_blank(),
        axis.ticks.x=element_blank(), legend.direction = "horizontal")
```

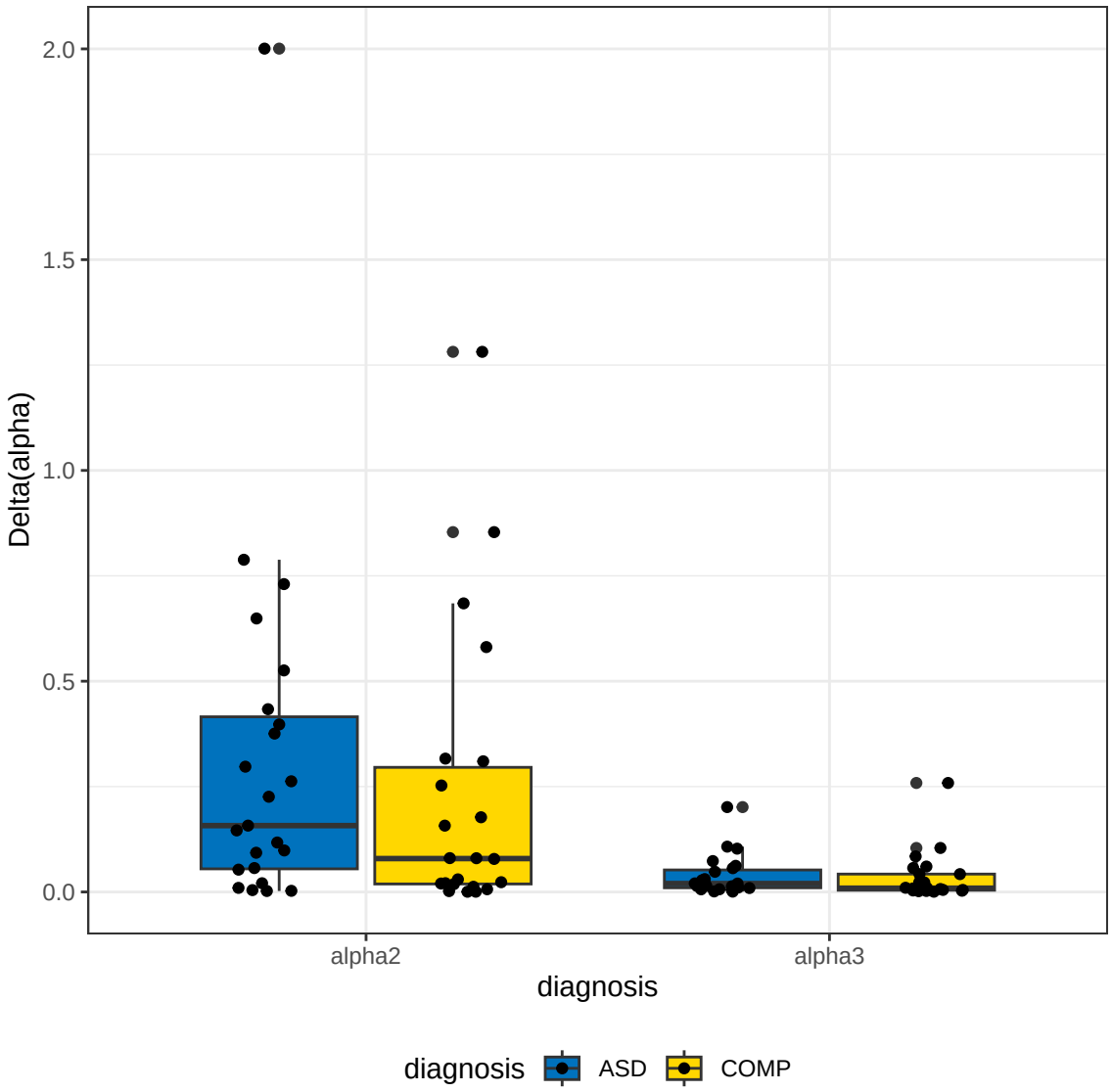



```
df.hgf %>%
  filter(diagnosis %in% c("ASD", "COMP")) %>%
  select(subID, diagnosis, be4, om3) %>% #
  pivot_longer(cols = c(be4, om3), names_to = "parameter") %>%
  ggplot(aes(x = diagnosis, y = value, fill = diagnosis))  +
  geom_boxplot() +
  geom_jitter(width = 0.2) +
  scale_fill_manual(values = c("#0072bd", "#ffd700")) +
  facet_wrap(. ~ parameter, scales = "free", ncol = 3) +
  labs(title = "HGF parameter", x = "", y = "") +
  theme_bw() +
  theme(legend.position = "bottom", plot.title = element_text(hjust = 0.5),
        text = element_text(size = 15), axis.text.x=element_blank(),
```

```
        axis.ticks.x=element_blank(), legend.direction = "horizontal")
```




```
# two-way interaction
df.upd %>%
  filter(diagnosis %in% c("ASD", "COMP") & change == "pre2vol") %>%
  ggplot(aes(y = value, fill = diagnosis, x = level)) +
  geom_boxplot() +
  geom_point(position=position_jitterdodge(dodge.width=0.9)) +
  scale_fill_manual(values = c("#0072bd", "#ffd700")) +
  labs(x = "diagnosis", y = "Delta(alpha)") + # "\u0394 \u03b1" not working for now
  theme_bw() +
  theme(legend.position = "bottom", plot.title = element_text(hjust = 0.5))
```


2.0
1.5
1.0
0.5
0.0
Delta(alpha)
alpha2
alpha3
diagnosis
diagnosis
ASD
COMP