S1: behavioural analysis with brms

I. S. Plank

2024-09-19

S1.1 Introduction

This R Markdown script analyses behavioural data from the FAB (face attention bias) paradigm of the EMBA project. The data was preprocessed before being read into this script.

The task is modeled after Jakobsen et al. (2021), *Attention, Perception, & Psychophysics* and the authors were happy to share their stimuli. Each trial starts with a black fixation cross on a white background. Then, a cue consisting of a pair of pictures, one object and one face, is shown with one picture on the left and one on the right of the previous location of the fixation cross. In line with Moore et al. (2012), *J Autism Dev Disord*, we set the duration of the cue presentation to 200ms. Afterwards, a target square appears either at the previous location of the face or the object. Subjects task is to determine the location (right or left) of the target as fast and accurate as possible. The target only disappears when the participant gives their answer.

Some general settings

```r
# number of simulations
nsim = 250

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

# set the seed
set.seed(2468)
```

Package versions

The following packages are used in this RMarkdown file:

```
## [1] "R version 4.4.1 (2024-06-14)"

## [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] "bayestestR version 0.13.2"
```

General info

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

We perform prior predictive checks as proposed in Schad, Betancourt and Vasishth (2020) using the SBC package. To do so, we create 250 simulated datasets where parameters are simulated from the priors. These parameters are used to create one fake dataset. Both the true underlying parameters and the simulated discrimination values are saved.

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

Preparation and group comparisons

First, we load the data and combine it with demographic information including the diagnostic status of the subjects. Then, all predictors are set to sum contrasts. We have a look at the demographics describing our three diagnostic groups: adults with ADHD, autistic adults and adults without any neurological and psychiatric diagnoses.

Since this is sensitive data, we load the anonymised version of the processed data at this point but also leave the code we used to create it.

```r
# check if the data file exists, if yes load it:
if (!file.exists("FAB_data.RData")) {

  # get demo info for subjects
  df.sub = read_csv(file.path("/home/emba/Documents/EMBA/CentraXX", "EMBA_centraXX.csv"), show_col_type
    mutate(
      diagnosis = recode(diagnosis, "CTR" = "COMP")
    )

  # set the data path
  dt.path = "/home/emba/Documents/EMBA/BVET"

  # load excluded participants (low accuracy)
  exc = scan(file.path(dt.path, 'FAB_exc.txt'), what="character", sep=NULL)
  df.exc = df.sub %>% filter(subID %in% exc) %>% select(diagnosis) %>% group_by(diagnosis) %>% count()

  # load the behavioral data and merge with group
  df.fab = merge(df.sub %>% select(subID, diagnosis),
                 readRDS(file = paste0(dt.path, '/df_FAB.RDS')), all.y = T) %>%
    mutate_if(is.character, as.factor)

  # only keep participants included in the study in the subject data frame
  df.sub = merge(df.fab %>% select(subID) %>% distinct(), df.sub, all.x = T)

  # load the eye tracking data and only keep participants included in the study
  load(file.path(dt.path, "FAB_ET_data.RData"))
  df.sac = merge(df.sac, df.sub %>% select(subID, diagnosis), keep.y = T)
```

```r
# anonymise the data
df.fab = df.fab %>%
  mutate(
    PID = subID,
    subID = as.factor(as.numeric(subID))
  )

# get a correspondence of original PIDs and anonymised subIDs
df.recode = df.fab %>% select(PID, subID) %>% distinct()
recode = as.character(df.recode$subID)
names(recode) = df.recode$PID
df.fab = df.fab %>% select(-PID)

# anonymise ET data in the same way
df.sac$subID = str_replace_all(df.sac$subID, recode)

# print gender frequencies and compare them across groups
tb.gen = xtabs(~ gender + diagnosis, data = df.sub)
ct.full = contingencyTableBF(tb.gen, sampleType = "indepMulti", fixedMargin = "cols")
# since only DAN in the ADHD group, we try again after excluding them
ct.mf = contingencyTableBF(tb.gen[2:3,], sampleType = "indepMulti", fixedMargin = "cols")

# check which outcomes of interest are normally distributed
df.sht = df.sub %>%
  group_by(diagnosis) %>%
  shapiro_test(age, iq, BDI_total, ASRS_total, RAADS_total, TAS_total) %>%
  mutate(
    sig = if_else(p < 0.05, "*", "")
  )

# most of the measures are not normally distributed, therefore, we compute ranks for these outcomes
df.sub = df.sub %>%
  mutate(
    rage   = rank(age),
    rBDI   = rank(BDI_total),
    rRAADS = rank(RAADS_total),
    rTAS   = rank(TAS_total),
    diagnosis = as.factor(diagnosis)
  )

# now we can compute our ANOVAs
aov.age   = anovaBF(rage   ~ diagnosis, data = df.sub)
aov.iq    = anovaBF(iq     ~ diagnosis, data = df.sub)
aov.BDI   = anovaBF(rBDI   ~ diagnosis, data = df.sub)
aov.ASRS  = anovaBF(ASRS_total  ~ diagnosis, data = df.sub)
aov.RAADS = anovaBF(rRAADS ~ diagnosis, data = df.sub)
aov.TAS   = anovaBF(rTAS   ~ diagnosis, data = df.sub)

# ...and put everything in a new dataframe for printing
measurement  = "Age"
ADHD      = sprintf("%.2f (±%.2f)", mean(df.sub[df.sub$diagnosis == "ADHD",]$age), sd(df.sub[df.sub$dia
ASD       = sprintf("%.2f (±%.2f)", mean(df.sub[df.sub$diagnosis == "ASD",]$age), sd(df.sub[df.sub$diag
COMP       = sprintf("%.2f (±%.2f)", mean(df.sub[df.sub$diagnosis == "COMP",]$age), sd(df.sub[df.sub$di
  logBF10 = sprintf("%.3f", aov.age@bayesFactor[["bf"]])
  df.table = data.frame(measurement, ADHD, ASD, COMP, logBF10)
  df.table = rbind(df.table,
       c(
         "ASRS",
         sprintf("%.2f (±%.2f)", mean(df.sub[df.sub$diagnosis == "ADHD",]$ASRS_total), sd(df.sub[df.sub$
         sprintf("%.2f (±%.2f)", mean(df.sub[df.sub$diagnosis == "ASD",]$ASRS_total), sd(df.sub[df.sub$d
         sprintf("%.2f (±%.2f)", mean(df.sub[df.sub$diagnosis == "COMP",]$ASRS_total), sd(df.sub[df.sub
         sprintf("%.3f", aov.ASRS@bayesFactor[["bf"]])
       ),
       c(
         "BDI",
         sprintf("%.2f (±%.2f)", mean(df.sub[df.sub$diagnosis == "ADHD",]$BDI_total), sd(df.sub[df.sub$d
         sprintf("%.2f (±%.2f)", mean(df.sub[df.sub$diagnosis == "ASD",]$BDI_total), sd(df.sub[df.sub$di
         sprintf("%.2f (±%.2f)", mean(df.sub[df.sub$diagnosis == "COMP",]$BDI_total), sd(df.sub[df.sub$d
         sprintf("%.3f", aov.BDI@bayesFactor[["bf"]])
       ),
       c(
         "Gender (diverse/agender/non-binary - female - male)",
         sprintf("%d - %d - %d", nrow(df.sub[df.sub$diagnosis == "ADHD" & df.sub$gender == "dan",]), nro
         sprintf("%d - %d - %d", nrow(df.sub[df.sub$diagnosis == "ASD" & df.sub$gender == "dan",]), nrow
         sprintf("%d - %d - %d", nrow(df.sub[df.sub$diagnosis == "COMP" & df.sub$gender == "dan",]), nro
         sprintf("%.3f", ct.full@bayesFactor[["bf"]])
       ),
       c(
         "IQ",
         sprintf("%.2f (±%.2f)", mean(df.sub[df.sub$diagnosis == "ADHD",]$iq), sd(df.sub[df.sub$diagnosi
         sprintf("%.2f (±%.2f)", mean(df.sub[df.sub$diagnosis == "ASD",]$iq), sd(df.sub[df.sub$diagnosis
         sprintf("%.2f (±%.2f)", mean(df.sub[df.sub$diagnosis == "COMP",]$iq), sd(df.sub[df.sub$diagnosi
         sprintf("%.3f", aov.iq@bayesFactor[["bf"]])
       ),
       c(
         "RAADS",
         sprintf("%.2f (±%.2f)", mean(df.sub[df.sub$diagnosis == "ADHD",]$RAADS_total), sd(df.sub[df.sub
         sprintf("%.2f (±%.2f)", mean(df.sub[df.sub$diagnosis == "ASD",]$RAADS_total), sd(df.sub[df.sub$
         sprintf("%.2f (±%.2f)", mean(df.sub[df.sub$diagnosis == "COMP",]$RAADS_total), sd(df.sub[df.sub
         sprintf("%.3f", aov.RAADS@bayesFactor[["bf"]])
       ),
       c(
         "TAS",
         sprintf("%.2f (±%.2f)", mean(df.sub[df.sub$diagnosis == "ADHD",]$TAS_total), sd(df.sub[df.sub$d
         sprintf("%.2f (±%.2f)", mean(df.sub[df.sub$diagnosis == "ASD",]$TAS_total), sd(df.sub[df.sub$di
         sprintf("%.2f (±%.2f)", mean(df.sub[df.sub$diagnosis == "COMP",]$TAS_total), sd(df.sub[df.sub$d
         sprintf("%.3f", aov.TAS@bayesFactor[["bf"]])
       )
  )

  # save it all
  save(df.fab, df.sac, df.table, df.sht, ct.full, ct.mf, df.exc, file = "FAB_data.RData")

} else {

  load("FAB_data.RData")
```

```r
}

# print the group of excluded participants
kable(df.exc)
```

diagnosis	n
ADHD	1
ASD	1

```r
rm(df.exc)

# print the outcome of the shapiro tests
kable(df.sht)
```

diagnosis	variable	statistic	p	sig
ADHD	ASRS_total	0.9304718	0.1119616	
ADHD	BDI_total	0.8170622	0.0007279	*
ADHD	RAADS_total	0.9257590	0.0885842	
ADHD	TAS_total	0.9593558	0.4504806	
ADHD	age	0.9180376	0.0604839	
ADHD	iq	0.9648751	0.5684099	
ASD	ASRS_total	0.9512379	0.2881380	
ASD	BDI_total	0.8078769	0.0003974	*
ASD	RAADS_total	0.9449707	0.2103029	
ASD	TAS_total	0.9181140	0.0530706	
ASD	age	0.9492659	0.2611939	
ASD	iq	0.9579062	0.3978177	
COMP	ASRS_total	0.9192136	0.0561186	
COMP	BDI_total	0.7421016	0.0000383	*
COMP	RAADS_total	0.8449398	0.0017635	*
COMP	TAS_total	0.8727860	0.0059679	*
COMP	age	0.8104190	0.0004382	*
COMP	iq	0.9505974	0.2791247	

```r
rm(df.sht)

# print the outcome of the two contingency tables for comparison
ct.full@bayesFactor
```

```
##                            bf error                     time        code
## Non-indep. (a=1) -3.748042     0 Wed Sep 11 10:12:21 2024 294e54f5e70e
```

```r
ct.mf@bayesFactor
```

```
##                            bf error                     time        code
## Non-indep. (a=1) -1.915524     0 Wed Sep 11 10:12:21 2024 294ee08e9ff
```

```r
# set and print the contrasts
contrasts(df.fab$cue) = contr.sum(2)
contrasts(df.fab$cue)
```

```
##          [,1]
## face      1
## object   -1
```

```
contrasts(df.fab$diagnosis) = contr.sum(3)
contrasts(df.fab$diagnosis)
```

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

```
# print final group comparisons for the paper
kable(df.table)
```

measurement	ADHD	ASD	COMP	logBF10
Age	26.70 (±1.51)	28.58 (±1.46)	27.42 (±1.15)	-1.774
ASRS	43.39 (±2.61)	32.54 (±1.62)	25.04 (±1.68)	12.024
BDI	8.09 (±1.77)	11.21 (±2.12)	2.25 (±0.62)	8.319
Gender (diverse/agender/non-binary - female - male)	2 - 9 - 12	0 - 13 - 11	0 - 11 - 13	-3.748
IQ	108.35 (±2.46)	111.31 (±2.95)	109.90 (±1.92)	-1.859
RAADS	92.61 (±8.74)	152.92 (±8.32)	44.58 (±7.15)	23.564
TAS	50.22 (±2.58)	63.17 (±1.48)	39.08 (±1.93)	19.247

The three diagnostic groups are similar in age, IQ and gender distribution. However, they seem to differ in their questionnaire scores measuring ADHD (ASRS), depression (BDI), autism (RAADS) and alexithymia (TAS).

S1.2 Reaction times

First, we analyse the reaction times for all correctly answered trials to assess whether participants answer faster if the target appears at the previous location of the face, which we refer to as face attention bias (FAB). In our preregistration, we formulated the following hypotheses:

COMP participants react faster in response to targets appearing on the side of the face compared to targets appearing on the side of the object (face attention bias; Jakobsen et al., 2021).

ADHD participants react slower than COMP participants in both cue conditions (Sonuga-Barke et al., 2004).

ASD participants react slower than COMP participants in both cue conditions (Ghosn et al., 2018).

Face attention bias is decreased in ASD participants compared to COMP participants (Moore et al., 2012).

Face attention bias in ADHD participants differs from face attention bias in COMP participants.

Full model

Simulation-based calibration

```
code = "FAB"

# full model formula
f.fab = brms::bf(rt.cor ~ diagnosis * cue + (cue | subID) + (cue * diagnosis | stm) )
```

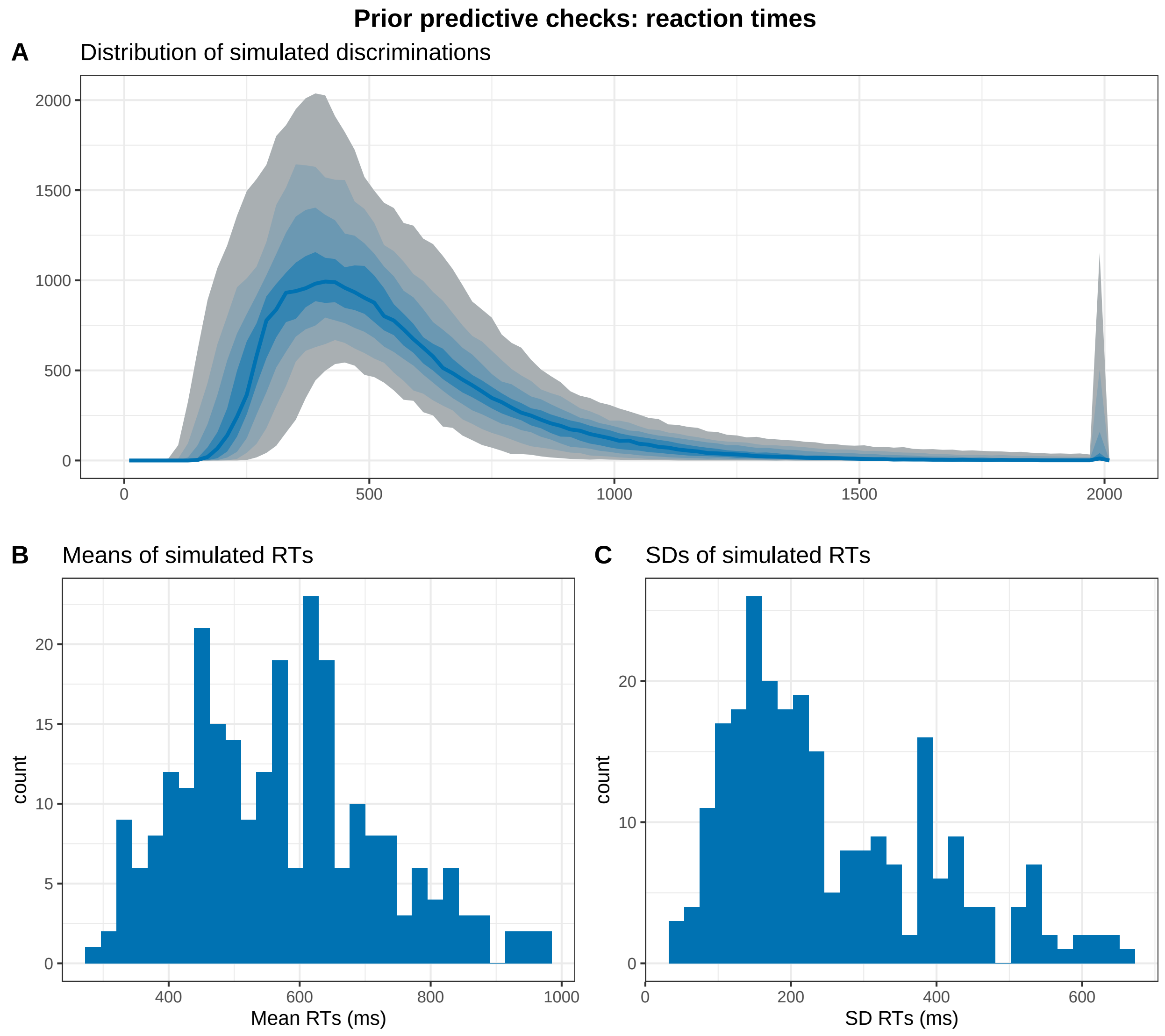
```r
# set informed priors based on previous results
priors = c(
  # general priors based on SBV
  prior(normal(6, 0.3),  class = Intercept),
  prior(normal(0, 0.5),  class = sigma),
  prior(normal(0, 0.1),  class = sd),
  prior(lkj(2),          class = cor),
  # face attention bias effect based on Jakobsen et al. (2021)
  prior(normal(-0.01, 0.04), class = b, coef = cue1),
  # ADHD subjects being slower based on Pievsky & McGrath (2018)
  prior(normal(0.025, 0.04),   class = b, coef = diagnosis1),
  # ASD subjects being slower based on Morrison et al. (2018)
  prior(normal(0.025, 0.04),   class = b, coef = diagnosis2),
  # decreased FAB in ASD subjects based on Moore et al. (2012)
  prior(normal(0.01,  0.04),   class = b, coef = diagnosis2:cue1),
  # no specific expectations for FAB in ADHD
  prior(normal(0,     0.04), class = b),
  # shift
  prior(normal(200,   100), class = ndt)
)

# 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 {
  # perform the SBC
  gen = SBC_generator_brms(f.fab, data = df.fab, prior = priors, family = shifted_lognormal,
                           thin = 50, warmup = 20000, refresh = 2000)
  bck = SBC_backend_brms_from_generator(gen, chains = 4, thin = 1,
                                        init = 0.1, warmup = warm, iter = iter)
  dat = generate_datasets(gen, nsim)
  saveRDS(dat, file.path(cache_dir, sprintf("dat_%s.rds", code)))
  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 6 of 250 simulations had at least one parameter that had an rhat of at least 1.05, and 1 models had divergent samples (mean number of samples of the simulations with divergent samples: 4). This suggests that this model performs well.

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

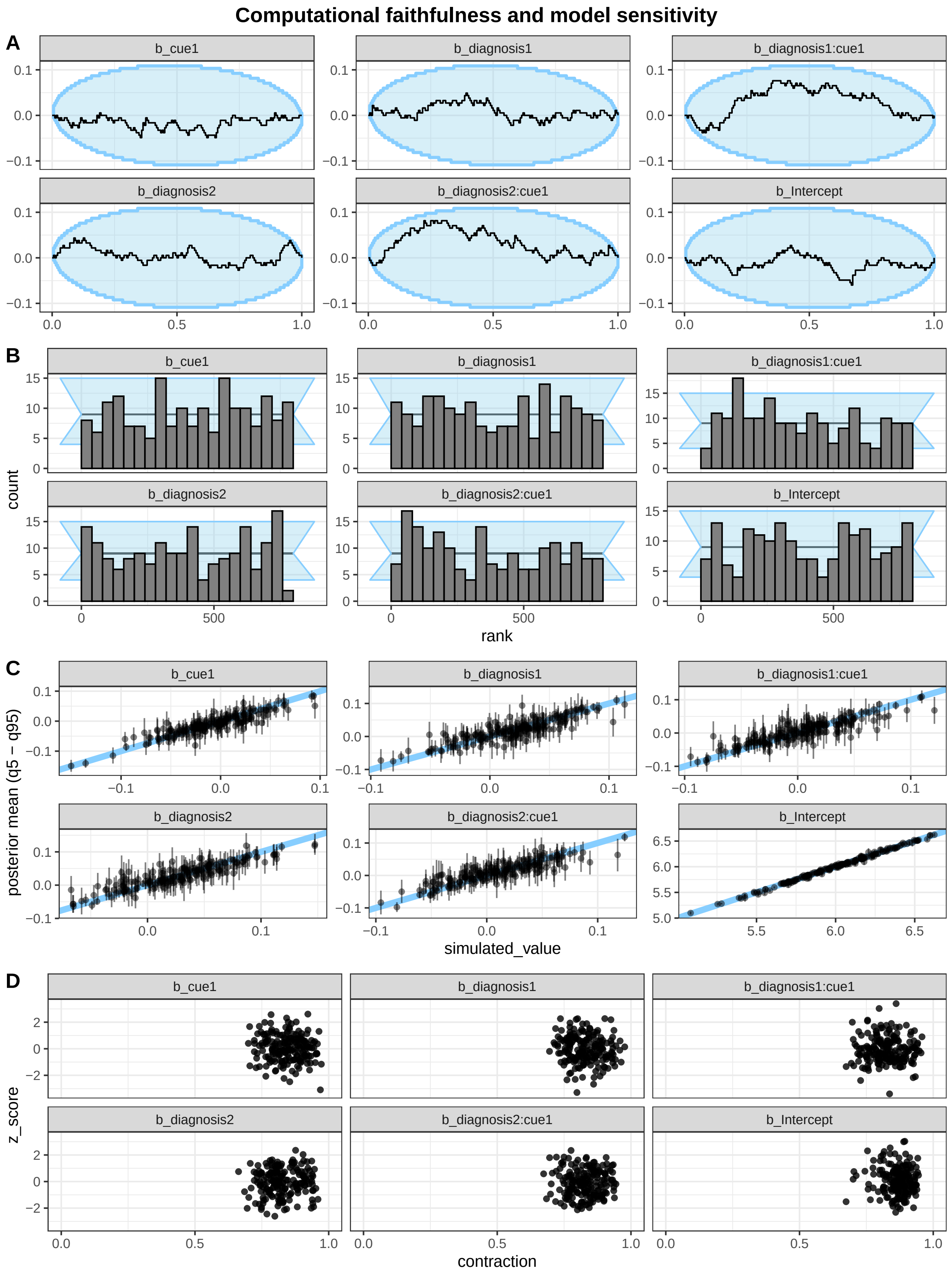
```r
# create a matrix out of generated data
dvname = gsub(" ", "", gsub("[\\|~].*", "", f.fab)[1])
dvfakemat = matrix(NA, nrow(dat[['generated']][[1]]), length(dat[['generated']]))
```

```r
for (i in 1:length(dat[['generated']])) {
  dvfakemat[,i] = dat[['generated']][[i]][[dvname]]
}
truePars = dat$variables

# set large values to a max
dvfakemat[dvfakemat > 2000] = 2000

# compute one histogram per simulated data-set
binwidth = 20
breaks = seq(0, max(dvfakemat, na.rm=T) + binwidth, binwidth)
histmat = matrix(NA, ncol = length(dat), nrow = length(breaks)-1)
for (i in 1:nrow(truePars)) {
  histmat[,i] = hist(dvfakemat[,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 discriminations", 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 RTs (ms)", title = "Means of simulated RTs") +
  theme_bw()
p3 = ggplot() +
  stat_bin(aes(x = tmpSD), fill = c_dark) +
  labs(x = "SD RTs (ms)", title = "SDs of simulated RTs") +
  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: reaction times", face = "bold", size = 14)
```



Subfigure A shows the distribution of the simulated data with bluer bands being more likely than greyer bands. It shows a distribution that fits our expectations about reaction times in a simple decision task. 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.

```r
# get simulation numbers with issues
rank = max(df.results$max_rank)
check = merge(df.results %>%
    group_by(sim_id) %>% summarise(rhat = max(rhat, na.rm = T), mean_rank = mean(max_rank)) %>%
    filter(rhat >= 1.05 | mean_rank != rank),
  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$sim_id))
p1 = plot_ecdf_diff(df.results.b) + theme_bw() + theme(legend.position = "none") +
  scale_x_continuous(breaks=scales::pretty_breaks(n = 3)) +
  scale_y_continuous(breaks=scales::pretty_breaks(n = 3))
p2 = plot_rank_hist(df.results.b, bins = 20) + theme_bw() +
  scale_x_continuous(breaks=scales::pretty_breaks(n = 3)) +
  scale_y_continuous(breaks=scales::pretty_breaks(n = 3))
p3 = plot_sim_estimated(df.results.b, alpha = 0.5) + theme_bw() +
  scale_x_continuous(breaks=scales::pretty_breaks(n = 3)) +
  scale_y_continuous(breaks=scales::pretty_breaks(n = 3))
p4 = plot_contraction(df.results.b, prior_sd = setNames(c(0.15, rep(0.10, length(unique(df.results.b$va
  theme_bw() +
  scale_x_continuous(breaks=scales::pretty_breaks(n = 3)) +
  scale_y_continuous(breaks=scales::pretty_breaks(n = 3))

p = ggarrange(p1, p2, p3, p4, labels = "AUTO", ncol = 1, nrow = 4)
annotate_figure(p, top = text_grob("Computational faithfulness and model sensitivity", face = "bold", s
```



Next, we check the ranks of the parameters. If the model is unbiased, these should be uniformly distributed

(Schad, Betancourt and Vasishth, 2020). The sample empirical cumulative distribution function (ECDF) lies within the theoretical distribution (95%) and the rank histogram also shows ranks within the 95% expected range, although there are some small deviations. We judge this to be acceptable.

Then, we investigated the relationship between the simulated true parameters and the posterior estimates. Although there are individual values diverging from the expected pattern, most parameters were recovered successfully within an uncertainty interval of alpha = 0.05.

Last, we explore the z-score and the posterior contraction of our population-level predictors. The z-score "determines the distance of the posterior mean from the true simulating parameter", while the posterior contraction "estimates how much prior uncertainty is reduced in the posterior estimation" (Schad, Betancourt and Vasisth, 2020). All of this looks good for this model.

Posterior predictive checks

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

```r
# fit the full model
set.seed(2469)
m.fab = brm(f.fab,
            df.fab, prior = priors,
            family = shifted_lognormal,
            iter = iter, warmup = warm,
            backend = "cmdstanr", threads = threading(8),
            file = "m_fab_full"
            )
rstan::check_hmc_diagnostics(m.fab$fit)
```

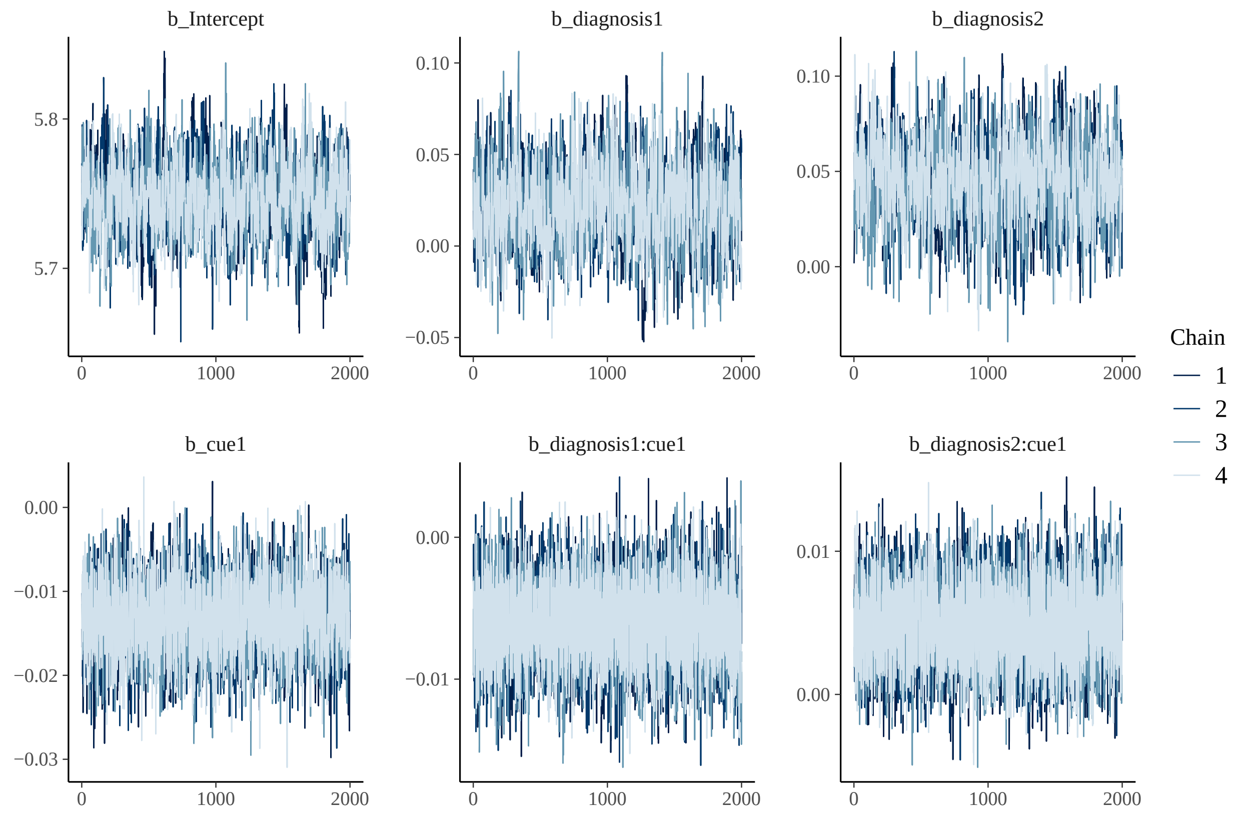
```
##
## Divergences:

## 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.
```

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

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

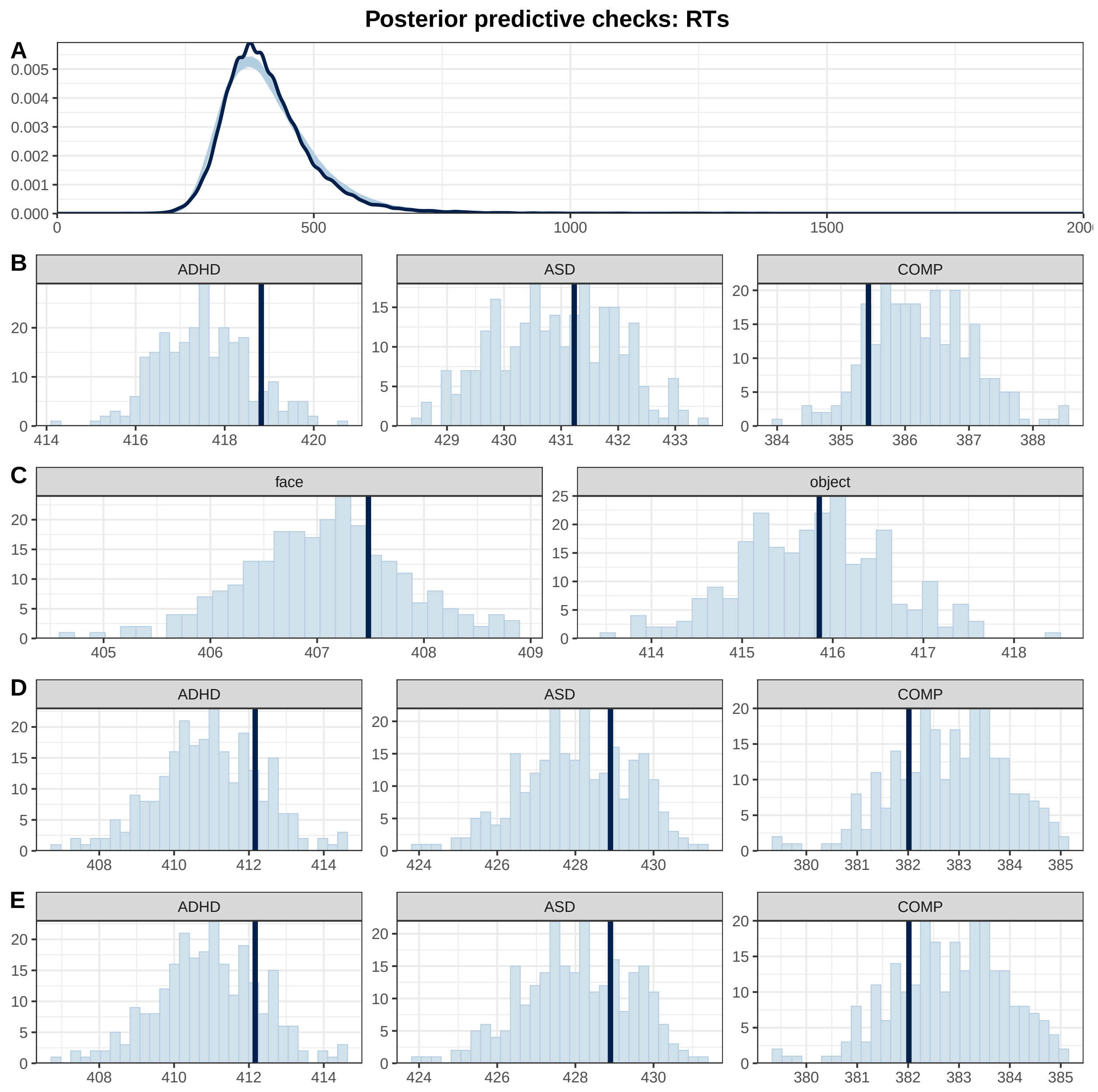
```r
# check the trace plots
post.draws = as_draws_df(m.fab)
mcmc_trace(post.draws, regex_pars = "^b_",
           facet_args = list(ncol = 3)) +
  scale_x_continuous(breaks=scales::pretty_breaks(n = 3)) +
  scale_y_continuous(breaks=scales::pretty_breaks(n = 3))
```

```
## 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 samples and only one rhat that is higher or equal to 1.01. Therefore, we go ahead and perform our posterior predictive checks.

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

# check the fit of the predicted data compared to the real data
p1 = pp_check(m.fab, ndraws = nsim) +
  theme_bw() + theme(legend.position = "none") + xlim(0, 2000)

# get rid of NAs in data frame for plotting
df.fab.na = df.fab[!is.na(df.fab$rt.cor),]

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

# distributions of means compared to the real values per cue
p3 = ppc_stat_grouped(df.fab.na$rt.cor, post.pred, df.fab.na$cue) +
  theme_bw() + theme(legend.position = "none")

# distributions of means compared to the real values per group for faces
p4 = ppc_stat_grouped(df.fab.na[df.fab.na$cue == "face",]$rt.cor, post.pred[,which(df.fab.na$cue == "fa
  theme_bw() + theme(legend.position = "none")

# distributions of means compared to the real values per group for objects
p5 = ppc_stat_grouped(df.fab.na[df.fab.na$cue == "face",]$rt.cor, post.pred[,which(df.fab.na$cue == "fa
  theme_bw() + theme(legend.position = "none")

p = ggarrange(p1, p2, p3, p4, p5,
          nrow = 5, ncol = 1, labels = "AUTO")
annotate_figure(p, top = text_grob("Posterior predictive checks: RTs", face = "bold", size = 14))
```



Although the overall shape in subfigure A of the simulated data fits well with the real data, the model seems to underestimate the reaction times of the ADHD group and overestimate the reaction times of the COMP group. This is also apparent when looking at the two cue conditions separately.

Since we are interested in accurate estimates, we decide to aggregate with the median of the reaction times per stimulus (face object combination) and cue. Then, there are no missing values in the data and we model an estimate for each specific stimulus and cue combination for each participant.

Aggregated model

First, we compute the aggregation and have a quick look at the resulting data.

```
# aggregate reaction times
df.fab = df.fab %>%
  group_by(subID, diagnosis, stm, cue) %>%
  summarise(
    rt.cor = median(rt.cor, na.rm = T)
  ) %>% ungroup() %>%
  mutate_if(is.character, as.factor)

# set and print the contrasts
contrasts(df.fab$cue) = contr.sum(2)
contrasts(df.fab$cue)
```

```
##        [,1]
## face      1
## object   -1
```

```
contrasts(df.fab$diagnosis) = contr.sum(3)
contrasts(df.fab$diagnosis)
```

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

```
summary(df.fab)
```

```
##      subID      diagnosis        stm            cue            rt.cor
##  1      :  72   ADHD:1656   1_10    : 142   face  :2556   Min.   :256.0
##  2      :  72   ASD :1728   1_11    : 142   object:2556   1st Qu.:361.5
##  3      :  72   COMP:1728   1_12    : 142                 Median :395.2
##  4      :  72               1_7     : 142                 Mean   :407.4
##  5      :  72               1_8     : 142                 3rd Qu.:437.5
##  6      :  72               1_9     : 142                 Max.   :919.0
##  (Other):4680               (Other):4260
```

There are now no NAs in the data, because no one made an error on all instances of one stimulus combination.

Stimulation-based calibration

We again perform an SBC. The model formula and priors can stay the same.

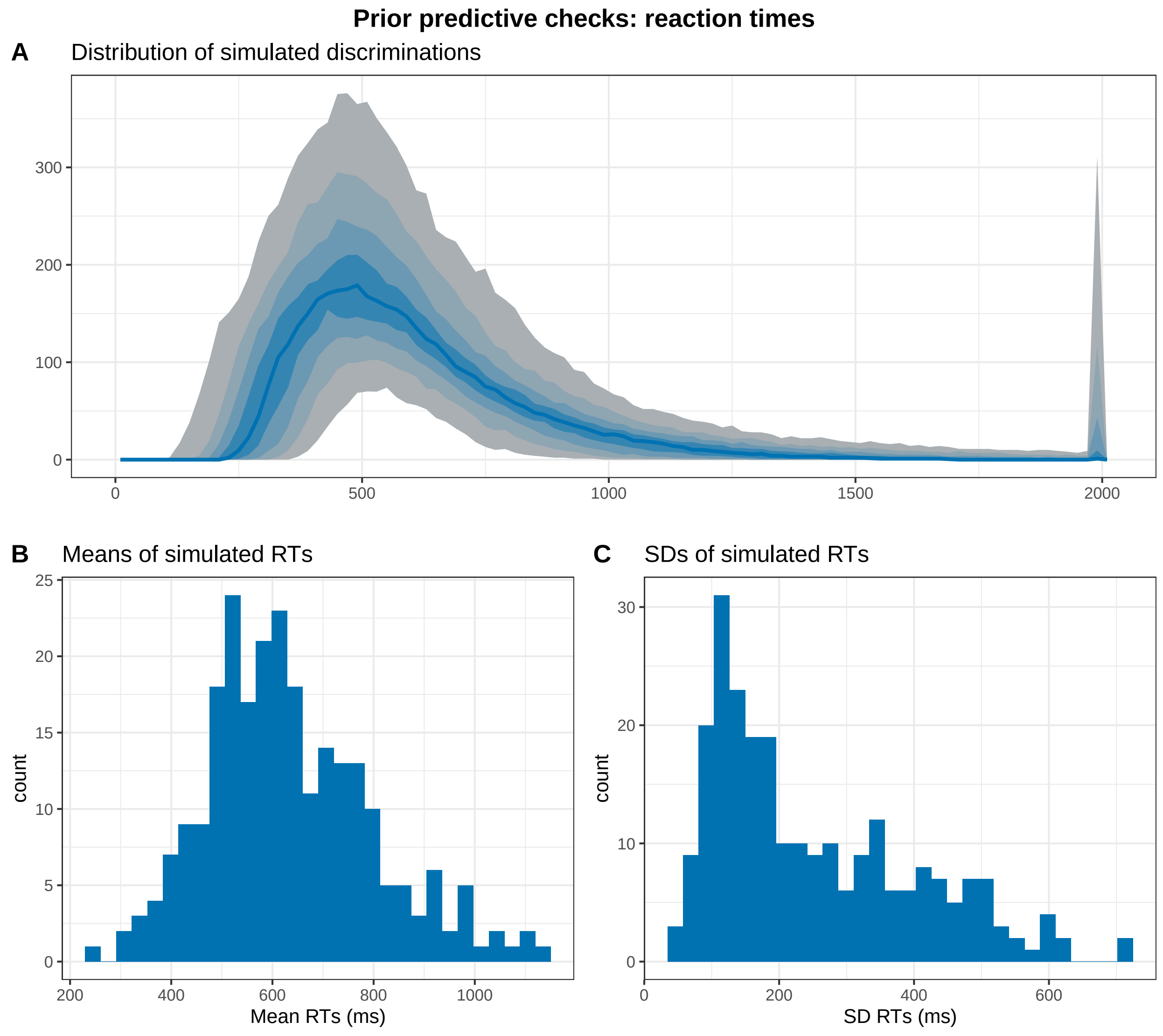
```
code = "FAB_agg"

# 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 {
  # perform the SBC
  gen = SBC_generator_brms(f.fab, data = df.fab, prior = priors, family = shifted_lognormal,
                           thin = 50, warmup = 20000, refresh = 2000)
  bck = SBC_backend_brms_from_generator(gen, chains = 4, thin = 1,
                                              init = 0.1, warmup = warm, iter = iter)
  set.seed(468)
  if (file.exists(file.path(cache_dir, sprintf("dat_%s.rds", code)))) {
    dat = readRDS(file.path(cache_dir, sprintf("dat_%s.rds", code)))
  } else {
    dat = generate_datasets(gen, nsim)
    saveRDS(dat, file.path(cache_dir, sprintf("dat_%s.rds", code)))
  }
  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")))
}

set.seed(4682)
```

We start by investigating the rhats and the number of divergent samples. This shows that 3 of 250 simulations had at least one parameter that had an rhat of at least 1.05, and 2 models had divergent samples (mean number of samples of the simulations with divergent samples: 17.5). This suggests that this model performs well.

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

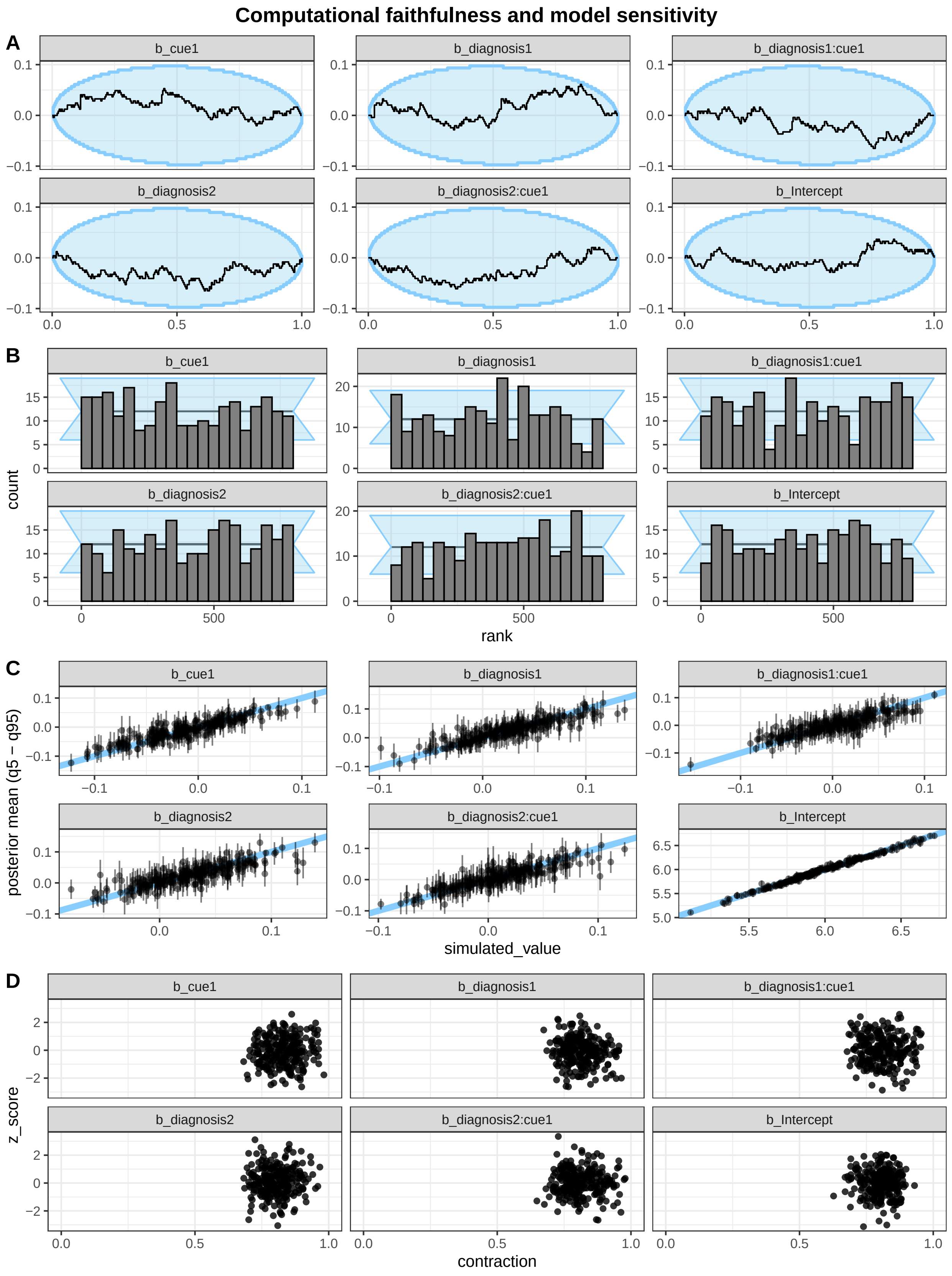
```
# create a matrix out of generated data
dvname = gsub(" ", "", gsub("[\\|~].*", "", f.fab)[1])
dvfakemat = matrix(NA, nrow(dat[['generated']][[1]]), length(dat[['generated']]))
for (i in 1:length(dat[['generated']])) {
  dvfakemat[,i] = dat[['generated']][[i]][[dvname]]
}
truePars = dat$variables

# set large values to a max
dvfakemat[dvfakemat > 2000] = 2000

# compute one histogram per simulated data-set
binwidth = 20
breaks = seq(0, max(dvfakemat, na.rm=T) + binwidth, binwidth)
histmat = matrix(NA, ncol = length(dat), nrow = length(breaks)-1)
for (i in 1:nrow(truePars)) {
  histmat[,i] = hist(dvfakemat[,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) +
```

```r
  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 discriminations", 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 RTs (ms)", title = "Means of simulated RTs") +
  theme_bw()
p3 = ggplot() +
  stat_bin(aes(x = tmpSD), fill = c_dark) +
  labs(x = "SD RTs (ms)", title = "SDs of simulated RTs") +
  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: reaction times", face = "bold", size = 14)
```



Again, this all looks good.

```
# get simulation numbers with issues
rank = max(df.results$max_rank)
check = merge(df.results %>%
    group_by(sim_id) %>% summarise(rhat = max(rhat, na.rm = T), mean_rank = mean(max_rank)) %>%
    filter(rhat >= 1.05 | mean_rank != rank),
  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$sim_id))
p1 = plot_ecdf_diff(df.results.b) + theme_bw() + theme(legend.position = "none") +
  scale_x_continuous(breaks=scales::pretty_breaks(n = 3)) +
  scale_y_continuous(breaks=scales::pretty_breaks(n = 3))
p2 = plot_rank_hist(df.results.b, bins = 20) + theme_bw() +
  scale_x_continuous(breaks=scales::pretty_breaks(n = 3)) +
  scale_y_continuous(breaks=scales::pretty_breaks(n = 3))
p3 = plot_sim_estimated(df.results.b, alpha = 0.5) + theme_bw() +
  scale_x_continuous(breaks=scales::pretty_breaks(n = 3)) +
  scale_y_continuous(breaks=scales::pretty_breaks(n = 3))
p4 = plot_contraction(df.results.b, prior_sd = setNames(c(0.15, rep(0.10, length(unique(df.results.b$va
  theme_bw() +
  scale_x_continuous(breaks=scales::pretty_breaks(n = 3)) +
  scale_y_continuous(breaks=scales::pretty_breaks(n = 3))

p = ggarrange(p1, p2, p3, p4, labels = "AUTO", ncol = 1, nrow = 4)
annotate_figure(p, top = text_grob("Computational faithfulness and model sensitivity", face = "bold", s
```



Next, we check the ranks of the parameters. If the model is unbiased, these should be uniformly distributed

(Schad, Betancourt and Vasishth, 2020). The sample empirical cumulative distribution function (ECDF) lies within the theoretical distribution (95%) and the rank histogram also shows ranks within the 95% expected range, although there are some small deviations. We judge this to be acceptable.

Then, we investigated the relationship between the simulated true parameters and the posterior estimates. Although there are individual values diverging from the expected pattern, most parameters were recovered successfully within an uncertainty interval of alpha = 0.05.

Last, we explore the z-score and the posterior contraction of our population-level predictors. The z-score "determines the distance of the posterior mean from the true simulating parameter", while the posterior contraction "estimates how much prior uncertainty is reduced in the posterior estimation" (Schad, Betancourt and Vasisth, 2020). All of this looks good for this model.

Posterior predictive checks

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

```r
# fit the final model
set.seed(6824)
m.fab = brm(f.fab,
            df.fab, prior = priors,
            family = shifted_lognormal,
            iter = iter, warmup = warm,
            backend = "cmdstanr", threads = threading(8),
            file = "m_fab_final"
            )
rstan::check_hmc_diagnostics(m.fab$fit)
```

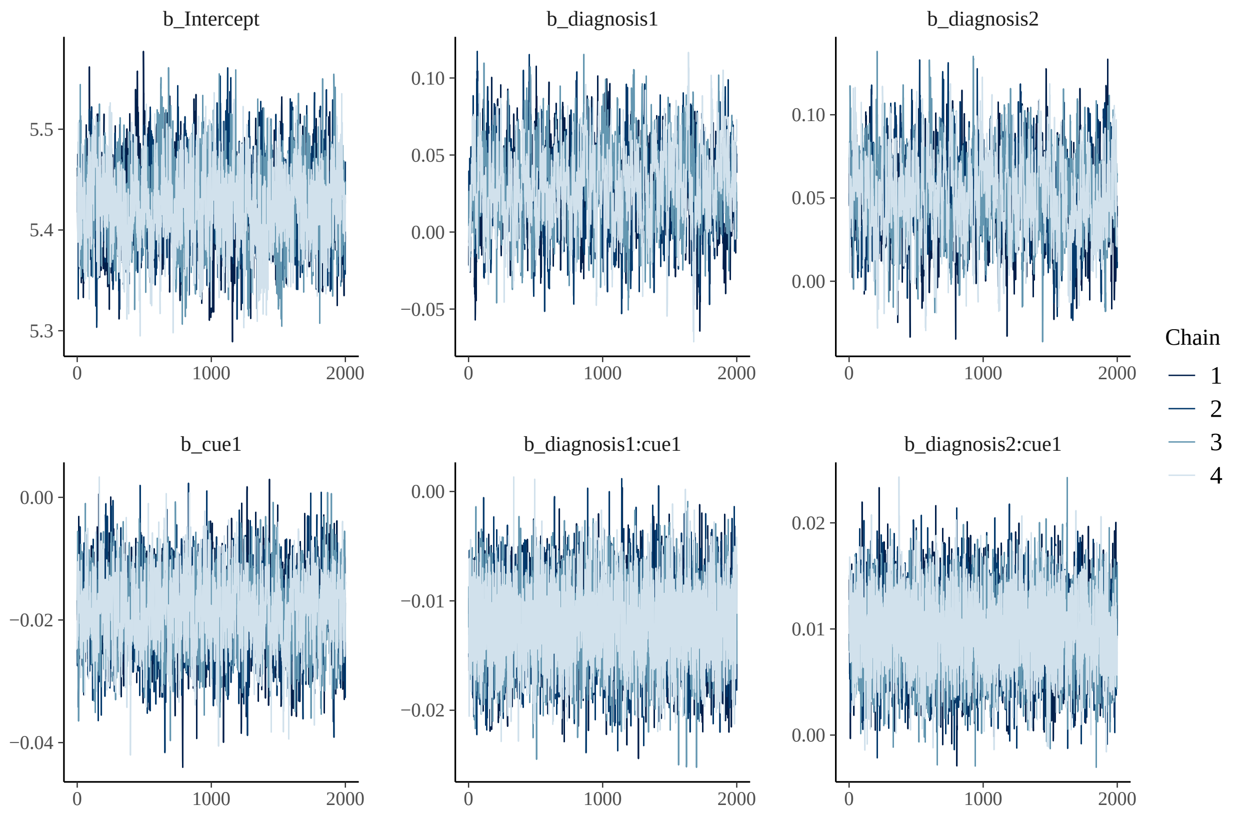
```
##
## Divergences:

## 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.
```

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

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

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

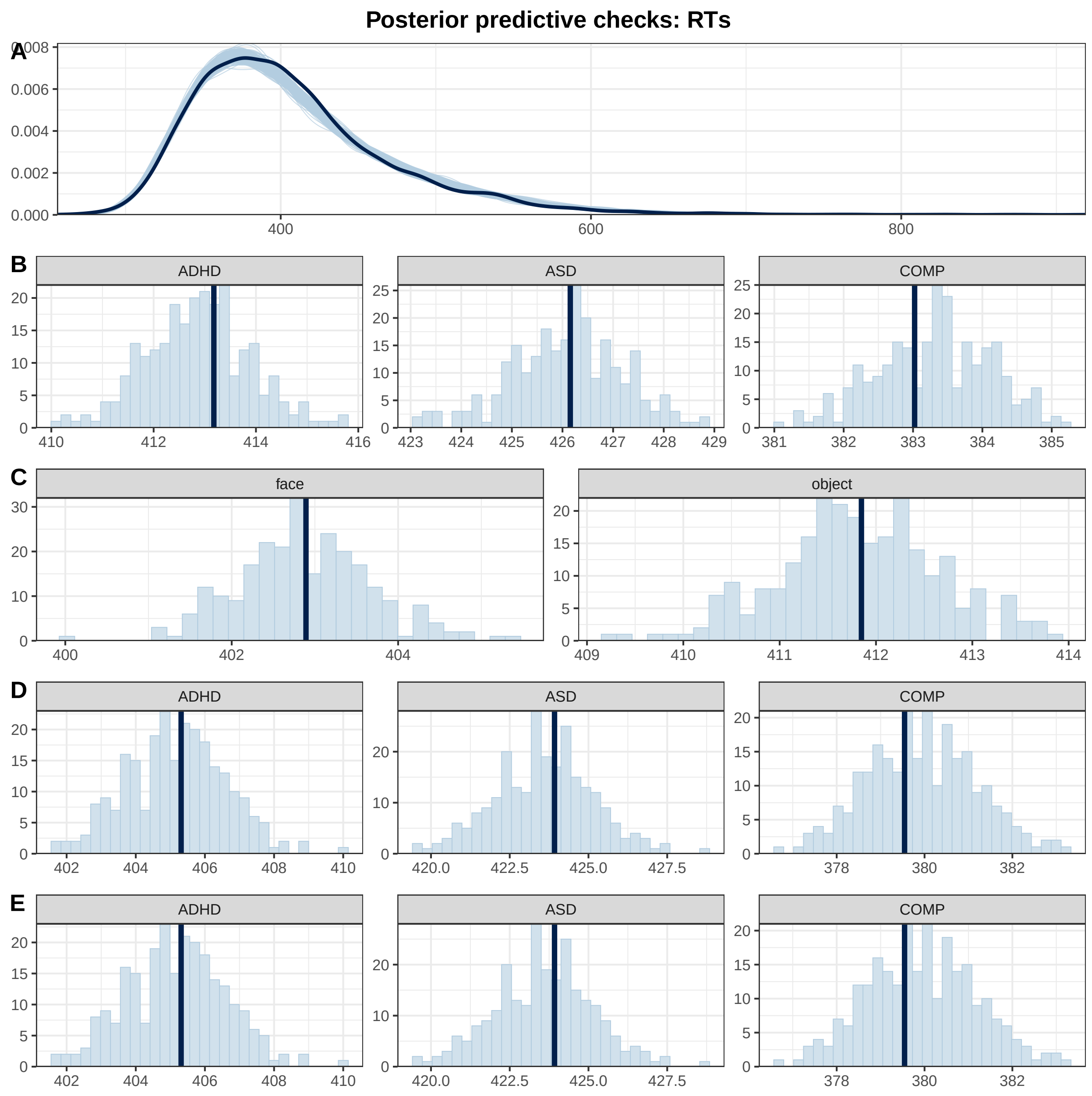
```
## Scale for x is already present.
## Adding another scale for x, which will replace the existing scale.
```



This model has no pathological behaviour with E-BFMI, no divergent samples and only one rhat that is higher or equal to 1.01. Therefore, we go ahead and perform our posterior predictive checks.

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

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

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

# distributions of means compared to the real values per cue
p3 = ppc_stat_grouped(df.fab$rt.cor, post.pred, df.fab$cue) +
  theme_bw() + theme(legend.position = "none")

# distributions of means compared to the real values per group for faces
p4 = ppc_stat_grouped(df.fab[df.fab$cue == "face",]$rt.cor, post.pred[,which(df.fab$cue == "face")], df
  theme_bw() + theme(legend.position = "none")

# distributions of means compared to the real values per group for objects
p5 = ppc_stat_grouped(df.fab[df.fab$cue == "face",]$rt.cor, post.pred[,which(df.fab$cue == "face")], df
  theme_bw() + theme(legend.position = "none")

p = ggarrange(p1, p2, p3, p4, p5,
```

```r
          nrow = 5, ncol = 1, labels = "AUTO")
annotate_figure(p, top = text_grob("Posterior predictive checks: RTs", face = "bold", size = 14))
```



This model fits our data much better, so we continue with it and can finally have a look at the estimates.

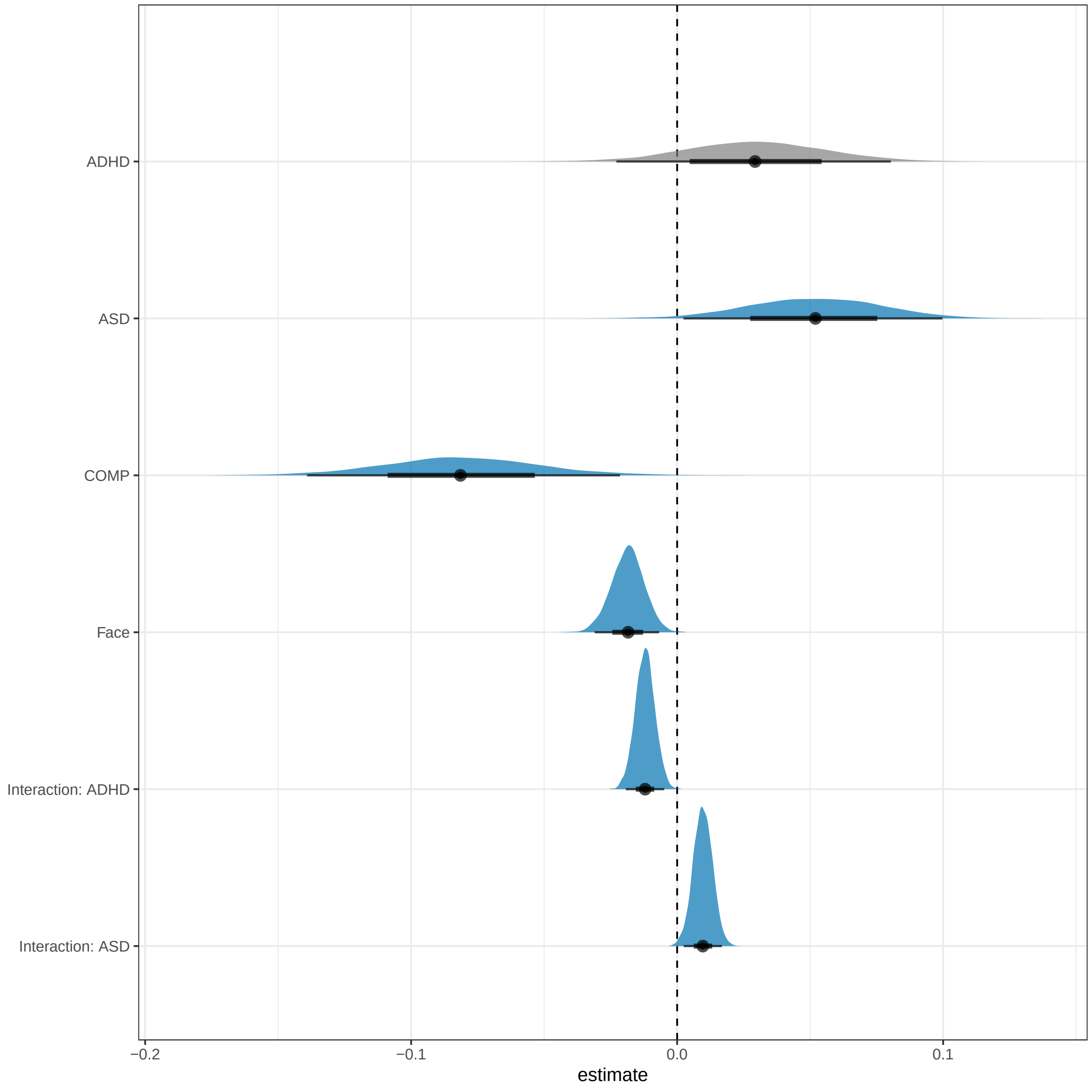
Inferences

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.fab)
```

```
##  Family: shifted_lognormal
##   Links: mu = identity; sigma = identity; ndt = identity
## Formula: rt.cor ~ diagnosis * cue + (cue | subID) + (cue * diagnosis | stm)
##    Data: df.fab (Number of observations: 5112)
##   Draws: 4 chains, each with iter = 3000; warmup = 1000; thin = 1;
##          total post-warmup draws = 8000
##
## Multilevel Hyperparameters:
## ~stm (Number of levels: 36)
##                                      Estimate Est.Error l-95% CI u-95% CI Rhat
## sd(Intercept)                            0.02      0.00     0.02     0.03 1.00
## sd(cue1)                                 0.03      0.00     0.03     0.04 1.00
## sd(diagnosis1)                           0.01      0.00     0.00     0.01 1.00
## sd(diagnosis2)                           0.00      0.00     0.00     0.01 1.00
## sd(cue1:diagnosis1)                      0.00      0.00     0.00     0.01 1.00
## sd(cue1:diagnosis2)                      0.00      0.00     0.00     0.01 1.00
## cor(Intercept,cue1)                     -0.34      0.16    -0.62    -0.01 1.00
## cor(Intercept,diagnosis1)               -0.06      0.29    -0.62     0.52 1.00
## cor(cue1,diagnosis1)                     0.01      0.29    -0.55     0.58 1.00
## cor(Intercept,diagnosis2)               -0.05      0.32    -0.64     0.57 1.00
## cor(cue1,diagnosis2)                    -0.06      0.32    -0.64     0.58 1.00
## cor(diagnosis1,diagnosis2)              -0.03      0.34    -0.66     0.63 1.00
## cor(Intercept,cue1:diagnosis1)           0.04      0.31    -0.57     0.63 1.00
## cor(cue1,cue1:diagnosis1)               -0.06      0.31    -0.64     0.55 1.00
## cor(diagnosis1,cue1:diagnosis1)          0.00      0.33    -0.63     0.61 1.00
## cor(diagnosis2,cue1:diagnosis1)          0.01      0.33    -0.62     0.64 1.00
## cor(Intercept,cue1:diagnosis2)          -0.22      0.31    -0.74     0.45 1.00
## cor(cue1,cue1:diagnosis2)               -0.04      0.30    -0.60     0.55 1.00
## cor(diagnosis1,cue1:diagnosis2)          0.03      0.33    -0.61     0.65 1.00
## cor(diagnosis2,cue1:diagnosis2)          0.05      0.33    -0.59     0.67 1.00
## cor(cue1:diagnosis1,cue1:diagnosis2)    -0.11      0.34    -0.72     0.57 1.00
##                                      Bulk_ESS Tail_ESS
## sd(Intercept)                            2811     4767
## sd(cue1)                                 2444     4161
## sd(diagnosis1)                           2429     3649
## sd(diagnosis2)                           3966     3845
## sd(cue1:diagnosis1)                      3663     4576
## sd(cue1:diagnosis2)                      3600     3473
## cor(Intercept,cue1)                      1405     2806
## cor(Intercept,diagnosis1)               10080     5818
## cor(cue1,diagnosis1)                    12563     5682
## cor(Intercept,diagnosis2)               16393     5907
## cor(cue1,diagnosis2)                    14307     5894
## cor(diagnosis1,diagnosis2)               8237     6494
## cor(Intercept,cue1:diagnosis1)          14956     6192
## cor(cue1,cue1:diagnosis1)               12163     5932
## cor(diagnosis1,cue1:diagnosis1)          8877     5643
## cor(diagnosis2,cue1:diagnosis1)          6504     6362
## cor(Intercept,cue1:diagnosis2)           9857     6113
## cor(cue1,cue1:diagnosis2)               12204     6197
## cor(diagnosis1,cue1:diagnosis2)          9337     6761
## cor(diagnosis2,cue1:diagnosis2)          7051     6784
## cor(cue1:diagnosis1,cue1:diagnosis2)     5397     6626
##
## ~subID (Number of levels: 71)
##                     Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(Intercept)           0.21      0.02     0.18     0.25 1.00     1092     2193
## sd(cue1)                0.02      0.00     0.01     0.02 1.00     3944     4716
## cor(Intercept,cue1)    -0.10      0.16    -0.40     0.21 1.00     7694     6615
##
## Regression Coefficients:
##                 Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept           5.43      0.04     5.35     5.51 1.00      790     1869
## diagnosis1          0.03      0.03    -0.02     0.08 1.00      831     1420
## diagnosis2          0.05      0.03     0.00     0.10 1.00      872     1782
## cue1               -0.02      0.01    -0.03    -0.01 1.00     2484     3486
## diagnosis1:cue1    -0.01      0.00    -0.02    -0.00 1.00     6657     6669
## diagnosis2:cue1     0.01      0.00     0.00     0.02 1.00     7003     6474
##
## Further Distributional Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma     0.13      0.00     0.12     0.14 1.00     5187     5264
## ndt     172.73      6.81   158.74   185.58 1.00     5295     4845
##
## 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 groups
df.m.fab = as_draws_df(m.fab) %>%
  select(starts_with("b_")) %>%
  mutate(
    b_COMP    = - b_diagnosis1 - b_diagnosis2,
    ASD       = b_Intercept + b_diagnosis2,
    ADHD      = b_Intercept + b_diagnosis1,
    COMP      = b_Intercept + b_COMP
    )

# plot the posterior distributions
df.m.fab %>%
  select(starts_with("b_")) %>%
  pivot_longer(cols = starts_with("b_"), names_to = "coef", values_to = "estimate") %>%
  filter(coef != "b_Intercept") %>%
  mutate(
    coef = case_match(coef,
      "b_diagnosis1" ~ "ADHD",
      "b_diagnosis2" ~ "ASD",
      "b_COMP"       ~ "COMP",
      "b_cue1"       ~ "Face",
      "b_diagnosis1:cue1" ~ "Interaction: ADHD",
      "b_diagnosis2:cue1" ~ "Interaction: ASD"
    ),
    coef = fct_reorder(coef, desc(coef))
  ) %>%
  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(credible = c_dark, c_light)) + theme(legend.position = "none")
```



```
# H1a: FAB effect in COMP
h1a = hypothesis(m.fab, "0 < (diagnosis1:cue1 + diagnosis2:cue1 - cue1)")
h1a
```

```
## Hypothesis Tests for class b:
```

```
##                   Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-((diagnosis1:... < 0    -0.02      0.01    -0.03        0      81.47
##   Post.Prob Star
## 1      0.99    *
## ---
## '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.
```

```
# H1b: ADHD slower than COMP
h1b = hypothesis(m.fab, "0 < 2*diagnosis1 + diagnosis2")
h1b
```

```
## Hypothesis Tests for class b:
##                   Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(2*diagnosis1... < 0    -0.11      0.05    -0.19    -0.03      65.12
##   Post.Prob Star
## 1      0.98    *
## ---
## '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.
```

```
# H1c: ASD slower than COMP
h1c = hypothesis(m.fab, "0 < 2*diagnosis2 + diagnosis1")
h1c
```

```
## Hypothesis Tests for class b:
##                   Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(2*diagnosis2... < 0    -0.13      0.05    -0.21    -0.05     194.12
##   Post.Prob Star
## 1      0.99    *
## ---
## '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.
```

```
# H1d: FAB in ASD decreased
h1d = hypothesis(m.fab, "0 < diagnosis2:cue1")
h1d
```

```
## Hypothesis Tests for class b:
##                   Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(diagnosis2:c... < 0    -0.01         0    -0.02        0     265.67
##   Post.Prob Star
## 1         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.
```

```
# ... compared to COMP
h1d_comp = hypothesis(m.fab, "0 < 2*diagnosis2:cue1 + diagnosis1:cue1")
```

```
h1d_comp
```

```
## Hypothesis Tests for class b:
##                  Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(2*diagnosis2... < 0    -0.01      0.01    -0.02        0       7.14
##   Post.Prob Star
## 1      0.88
## ---
## '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.
```

```
# H1e: FAB in ADHD differs from FAB generally (undirected)
h1e = hypothesis(m.fab, "0 > diagnosis1:cue1", alpha = 0.025)
h1e
```

```
## Hypothesis Tests for class b:
##                  Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(diagnosis1:c... > 0     0.01         0        0     0.02    1141.86
##   Post.Prob Star
## 1         1    *
## ---
## '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.
```

```
# ... compared to COMP
h1e_comp = hypothesis(m.fab, "0 > 2*diagnosis1:cue1 + diagnosis2:cue1", alpha = 0.025)
h1e_comp
```

```
## Hypothesis Tests for class b:
##                  Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(2*diagnosis1... > 0     0.01      0.01        0     0.03      73.07
##   Post.Prob Star
## 1      0.99    *
## ---
## 'CI': 95%-CI for one-sided and 97.5%-CI for two-sided hypotheses.
## '*': For one-sided hypotheses, the posterior probability exceeds 97.5%;
## for two-sided hypotheses, the value tested against lies outside the 97.5%-CI.
## Posterior probabilities of point hypotheses assume equal prior probabilities.
```

```
# Exploration

# E1: FAB generally
e1 = hypothesis(m.fab, "cue1 < 0", alpha = 0.025)
e1
```

```
## Hypothesis Tests for class b:
##   Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio Post.Prob Star
## 1 (cue1) < 0    -0.02      0.01    -0.03    -0.01     532.33         1    *
## ---
## '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.
```

```
# E2: FAB effect in ADHD
e2 = hypothesis(m.fab, "0 < -cue1 - diagnosis1:cue1", alpha = 0.025)
e2
```

```
## Hypothesis Tests for class b:
##                 Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(-cue1-diagno... < 0    -0.03      0.01    -0.04    -0.02        Inf
##   Post.Prob Star
## 1         1    *
## ---
## '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.
```

```
# E3: FAB effect in ASD
e3 = hypothesis(m.fab, "0 < -cue1 - diagnosis2:cue1", alpha = 0.025)
e3
```

```
## Hypothesis Tests for class b:
##                 Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(-cue1-diagno... < 0    -0.01      0.01    -0.02        0       8.71
##   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.
```

```
# E4: ADHD differ from ASD in their reaction time
e4 = hypothesis(m.fab, "diagnosis1 < diagnosis2", alpha = 0.025)
e4
```

```
## Hypothesis Tests for class b:
##                 Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (diagnosis1)-(dia... < 0    -0.02      0.04     -0.1     0.06        2.4
##   Post.Prob Star
## 1      0.71
## ---
## '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.
```

```
# E5: FAB in ADHD differs from FAB in ASD
e5 = hypothesis(m.fab, "diagnosis1:cue1 < diagnosis2:cue1", alpha = 0.025)
e5
```

```
## Hypothesis Tests for class b:
##                 Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (diagnosis1:cue1)... < 0    -0.02      0.01    -0.03    -0.01       7999
##   Post.Prob Star
## 1         1    *
## ---
## '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.
```

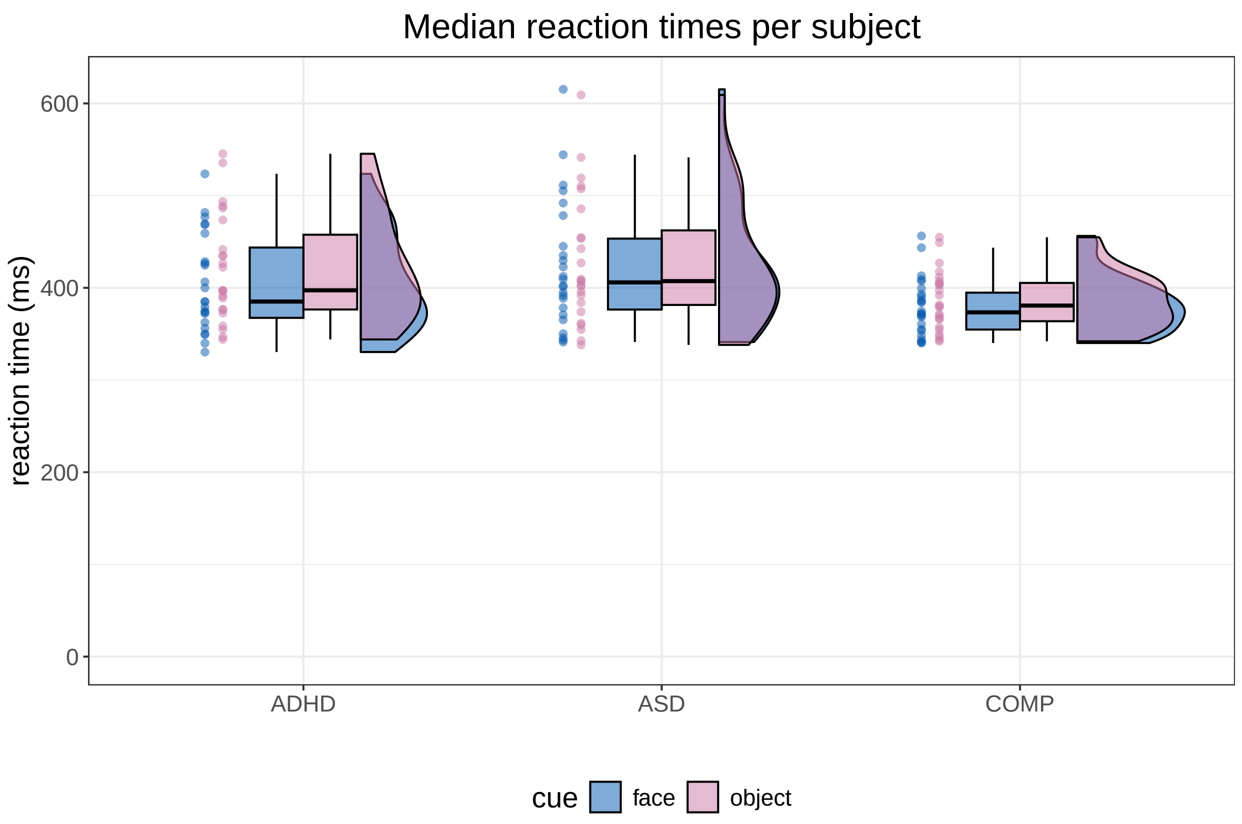
```
# extract predicted differences in ms instead of log data
df.new = df.fab %>%
  select(diagnosis, cue) %>%
  distinct() %>%
  mutate(
    condition = paste(diagnosis, cue, sep = "_")
  )
df.ms = as.data.frame(
  fitted(m.fab, summary = F,
             newdata = df.new %>% select(diagnosis, cue),
             re_formula = NA))
colnames(df.ms) = df.new$condition

# calculate our difference columns
df.ms = df.ms %>%
  mutate(
    COMP = (COMP_face + COMP_object)/2,
    ADHD = (ADHD_face + ADHD_object)/2,
    ASD  = (ASD_object + ASD_face)/2,
    FAB  = (ADHD_object + ASD_object + COMP_object - ADHD_face - ASD_face - COMP_face)/3,
    h1a  = COMP_object - COMP_face,
    h1b  = COMP - ADHD,
    h1c  = COMP - ASD,
    h1d_comp = (ASD_object - ASD_face) - (h1a),
    h1d  = (ASD_object - ASD_face) - FAB,
    h1e  = (ADHD_object - ADHD_face) - (h1a),
    e4   = ADHD - ASD,
    e5   = (ADHD_object - ADHD_face) - (ASD_object - ASD_face)
  )
```

Our Bayesian linear mixed model with the median of correct reaction times as the outcome and diagnostic status, cue (face or object) and their interaction confirmed a face attention bias in our comparison group: COMP participants reacted faster in response to targets appearing on the side of the face compared to targets appearing on the side of the object (CI of COMP(object) - COMP(face): 0.95 to 13ms, posterior probability = 98.79%). Both clinical groups exhibited increased overall reaction times compared with the COMP group (CI of COMP - ADHD: -46.47 to -2.43ms, posterior probability = 98.49%; CI of COMP - ASD: -51.38 to -7.49ms, posterior probability = 99.49%). Furthermore, while face attention bias was not credibly decreased in ASD participants compared to COMP participants (CI of ASD(FAB) - COMP(FAB): -8.29 to 3.33ms, posterior probability = 87.71%), it is credibly decreased compared to the average face attention bias in our sample (CI of ASD(FAB) - average FAB: -7.62 to -0.82ms, posterior probability = 99.62%). This effect is also reflected in the face attention bias in the ADHD group which differs credibly from face attention bias in COMP participants (CI of ADHD(FAB) - COMP(FAB): 1.8 to 13.48ms, posterior probability = 98.65%).

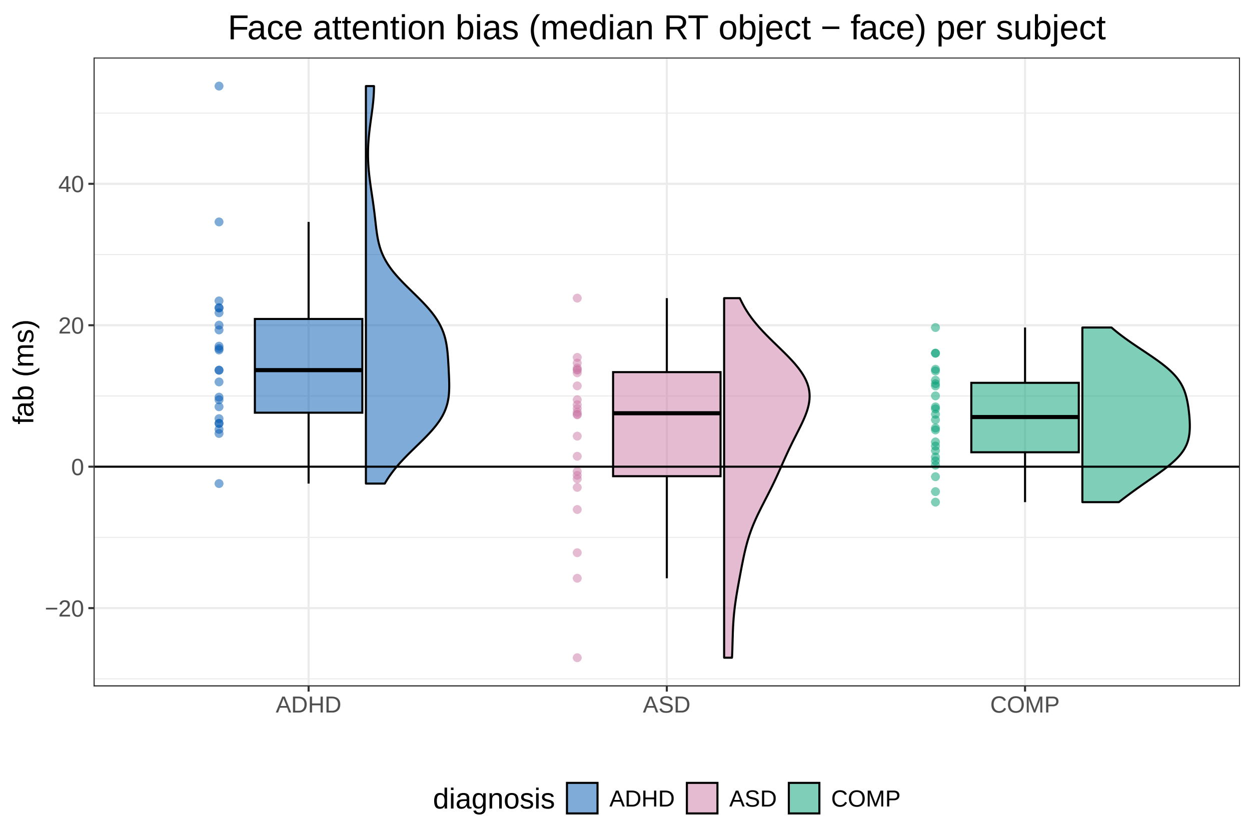
Furthermore, our exploration of the data revealed a credible face attention bias across all groups (CI of object - face: 3.09 to 14.27ms, posterior probability = 99.81%). Last, we explored differences between our clinical groups. While there were no credible differences in overall reaction times between adults with ASD and adults with ADHD in our data (CI of ADHD - ASD: -24.61 to 14.2ms, posterior probability = 70.62%), there was a credible difference between face attention bias in the clinical groups (CI of ASD(FAB) - ADHD(FAB): 4.26 to 16.24ms, posterior probability = 99.99%).

Plots

```
# overall median reaction times
df.fab %>%
  group_by(subID, diagnosis, cue) %>%
  summarise(
    rt.cor = mean(rt.cor, na.rm = T)
    ) %>%
  ggplot(aes(diagnosis, rt.cor, fill = cue, colour = cue)) + #
  geom_rain(rain.side = 'r',
boxplot.args = list(color = "black", outlier.shape = NA, show_guide = FALSE, alpha = 0.5),
violin.args = list(color = "black", outlier.shape = NA, alpha = 0.5),
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))) +
  ylim(0, 620) +
  scale_fill_manual(values = custom.col) +
  scale_color_manual(values = custom.col) +
  labs(title = "Median reaction times per subject", x = "", y = "reaction time (ms)") +
  theme_bw() +
  theme(legend.position = "bottom", plot.title = element_text(hjust = 0.5), legend.direction = "horizont
```



```r
# focus on the difference in reaction times
df.fab %>%
  group_by(subID, diagnosis, cue) %>%
  summarise(
    rt.cor = mean(rt.cor, na.rm = T)
    ) %>%
  group_by(subID, diagnosis) %>%
  arrange(subID, diagnosis, cue) %>%
  summarise(FAB = diff(rt.cor[1:2])) %>%
  ggplot(aes(diagnosis, FAB, fill = diagnosis, colour = diagnosis)) + #
  geom_rain(rain.side = 'r',
boxplot.args = list(color = "black", outlier.shape = NA, show_guide = FALSE, alpha = 0.5),
violin.args = list(color = "black", outlier.shape = NA, alpha = 0.5),
boxplot.args.pos = list(
  position = ggpp::position_dodgenudge(x = 0, width = 0.3), width = 0.3
),
point.args = list(show_guide = FALSE, alpha = .5),
violin.args.pos = list(
  width = 0.6, position = position_nudge(x = 0.16)),
point.args.pos = list(position = ggpp::position_dodgenudge(x = -0.25, width = 0.1))) +
  geom_hline(yintercept = 0) +
  scale_fill_manual(values = custom.col) +
  scale_color_manual(values = custom.col) +
  labs(title = "Face attention bias (median RT object - face) per subject", x = "", y = "fab (ms)") +
  theme_bw() +
  theme(legend.position = "bottom", plot.title = element_text(hjust = 0.5), legend.direction = "horizon
```

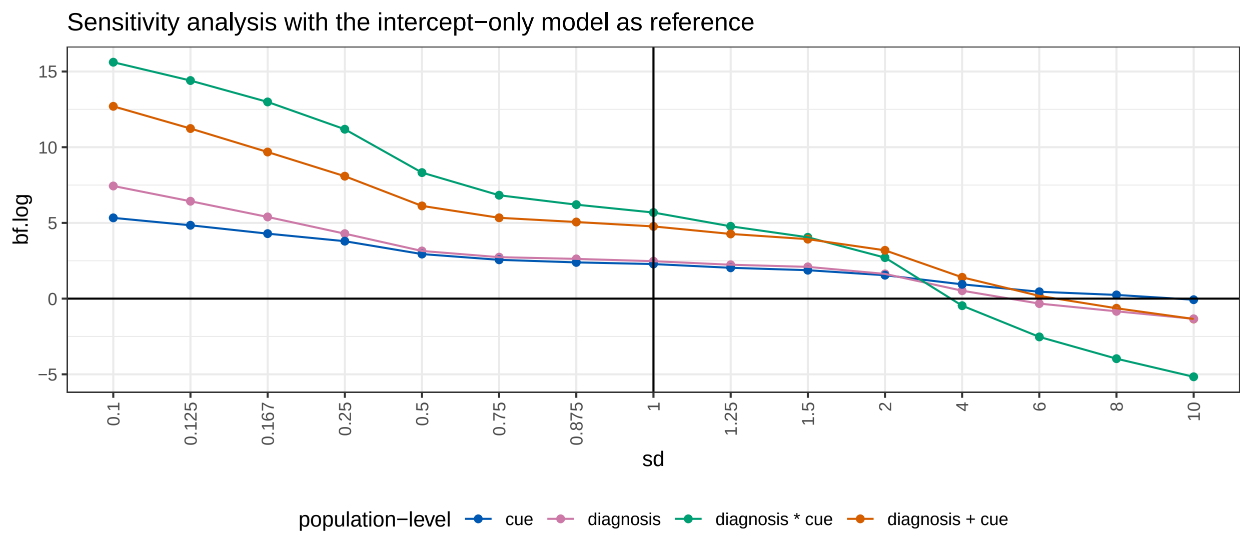


Bayes factor analysis

To complement our hypothesis testing using brms::hypothesis(), we perform a Bayes Factor (BF) analysis. The BF is the ratio of the marginal likelihoods of the data given two models. We will compare models containing different combinations of population-level effects to the model only containing the intercept on the population-level and all group-level effects. The BF depends on the priors that were used, because it indicates a change in our belief after seeing the data. Therefore, we perform a sensitivity analysis comparing the BF based on our chosen priors with narrower and wider priors.

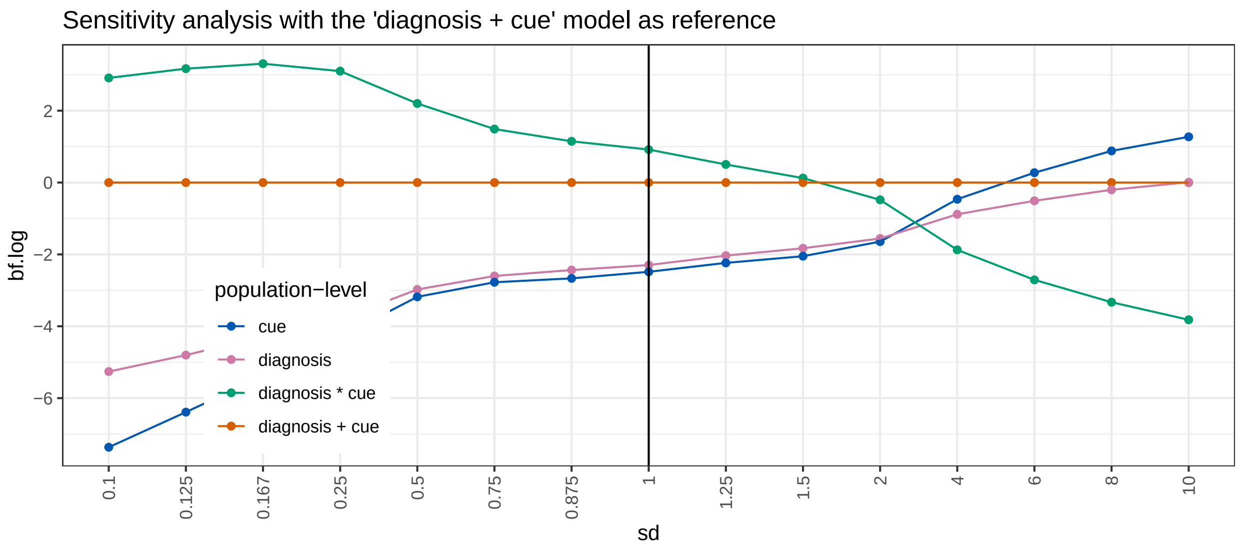
```r
# set the directory in which to save results
sense_dir = file.path(getwd(), "_brms_sens_cache")
main.code = "fab"

# describe priors
pr.descriptions = c("chosen",
  "sdx1.25",  "sdx1.5",  "sdx2",    "sdx4",   "sdx6",     "sdx8",     "sdx10",
  "sdx0.875", "sdx0.75", "sdx0.5", "sdx0.25", "sdx0.167", "sdx0.125", "sdx0.1"
  )

# check which have been run already
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)]
}

if (length(pr.descriptions) > 0) {
  # rerun the model with more iterations for bridgesampling
  m.fab.bf = brm(f.fab,
            df.fab, prior = priors,
            family = shifted_lognormal,
            iter = 40000, warmup = 10000,
            backend = "cmdstanr", threads = threading(8),
            file = "m_fab_bf", silent = 2,
            save_pars = save_pars(all = TRUE)
            )
}

# loop through them
for (pr.desc in pr.descriptions) {
  tryCatch({
    # use function to compute BF with the described priors
    bf_sens_2int(m.fab.bf, "diagnosis", "cue", pr.desc,
     main.code, # prefix for all models and MLL
     file.path(sense_dir, "log_FAB.txt"), # log file
     sense_dir # where to save the models and MLL
    )
  },
  error = function(err) {
    message(sprintf("Error for %s: %s", pr.desc, err))
  }
  )
}

# read in the results
```

```
df.fab.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.fab.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))
```



Quite a few models outperform the intercept model unless very wide priors are used. Therefore, we plot the models containing predictors with the model containing both predictors but no interaction as the reference model.

```
# compare to main effects model as reference
df.fab.bf %>%
  filter(`population-level` != "1") %>%
  group_by(priors) %>%
  mutate(bf.log = bf.log - bf.log[`population-level` == "diagnosis + cue"]) %>%
  ungroup() %>%
  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") +
  theme_bw() +
  ggtitle("Sensitivity analysis with the 'diagnosis + cue' model as reference") +
  scale_colour_manual(values = custom.col) +
  theme(legend.position = c(0.2, 0.25), axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1))
```



Here, we can see that the model containing both predictors and their interaction consistently outperforms the other models for narrow and medium priors. For wider priors, the results are less consistent.

```
# create a data frame with the comparisons
kable(df.fab.bf %>% filter(priors == "chosen") %>% select(-priors) %>%
  filter(`population-level` != "1") %>% arrange(desc(bf.log)), digits = 3)
```

population-level	bf.log
diagnosis * cue	5.686
diagnosis + cue	4.768
diagnosis	2.471
cue	2.283

population-level	bf.log

The comparison of the models reveals the model containing both main effects and their interaction on the population-level to be the best model as measured by the BF when using our chosen priors.

Specifically, there is anecdotal evidence in favour of this model underlying the data observed compared to the model not including the interaction ($\log(BF) = 0.918$), very strong evidence in favour of this model compared to the model including only the predictor cue ($\log(BF) = 3.403$), strong evidence in favour of this model compared to the one only including the predictor diagnosis ($\log(BF) = 3.215$) as well as extreme evidence in favour of this model compared to the model only including the intercept on the population-level ($\log(BF) = 5.686$).

S1.3 Explorative analysis of errors

Last but not least, we are going to explore possible differences with regards to mean accuracies using a Bayesian ANOVA

```
# load the original, unaggregated data
load("FAB_data.RData")

# aggregate mean accuracy per condition and person
df.acc = df.fab %>%
  group_by(subID, diagnosis, cue) %>%
  summarise(acc = mean(acc)*100)

# check normal distribution
df.acc %>% group_by(diagnosis, cue) %>%
  shapiro_test(acc) %>%
    mutate(
      sig = if_else(p < 0.05, "*", "")
    )
```

```
## # A tibble: 6 x 6
##   diagnosis cue    variable statistic            p sig
##   <fct>     <fct>  <chr>        <dbl>        <dbl> <chr>
## 1 ADHD      face   acc          0.504 0.0000000940 *
## 2 ADHD      object acc          0.621 0.00000154   *
## 3 ASD       face   acc          0.901 0.0230       *
## 4 ASD       object acc          0.791 0.000213     *
## 5 COMP      face   acc          0.814 0.000500     *
## 6 COMP      object acc          0.787 0.000180     *
```

```
# rank transform the data
df.acc = df.acc %>%
  ungroup() %>%
  mutate(
    racc   = rank(acc)
    )

# run the ANOVA
aov.acc   = anovaBF(racc   ~ diagnosis*cue, data = df.acc)
extractBF(aov.acc, logbf = T)
```

```
##                                         bf         error                          time
## diagnosis                       -1.021576 0.0002886018 Thu Sep 19 16:06:20 2024
## cue                             -1.708827 0.0004360949 Thu Sep 19 16:06:20 2024
## diagnosis + cue                 -2.730620 0.0196305266 Thu Sep 19 16:06:20 2024
## diagnosis + cue + diagnosis:cue -4.648269 0.0168901727 Thu Sep 19 16:06:20 2024
##                                          code
## diagnosis                       46114291114ee
## cue                             461143bb3417c
## diagnosis + cue                 461143cf63524
## diagnosis + cue + diagnosis:cue 4611416894eb4
```

```
# print overall accuracy rates
df.acc %>%
  group_by(diagnosis, cue) %>%
  summarise(mean_accuracy = mean(acc, na.rm = T), sd_accuracy = sd(acc, na.rm = T))
```

```
## # A tibble: 6 x 4
## # Groups:   diagnosis [3]
##   diagnosis cue    mean_accuracy sd_accuracy
##   <fct>     <fct>          <dbl>       <dbl>
## 1 ADHD      face            96.6        4.64
## 2 ADHD      object          96.4        4.29
## 3 ASD       face            96.9        2.73
## 4 ASD       object          97.6        2.02
## 5 COMP      face            97.8        2.00
## 6 COMP      object          97.7        2.11
```

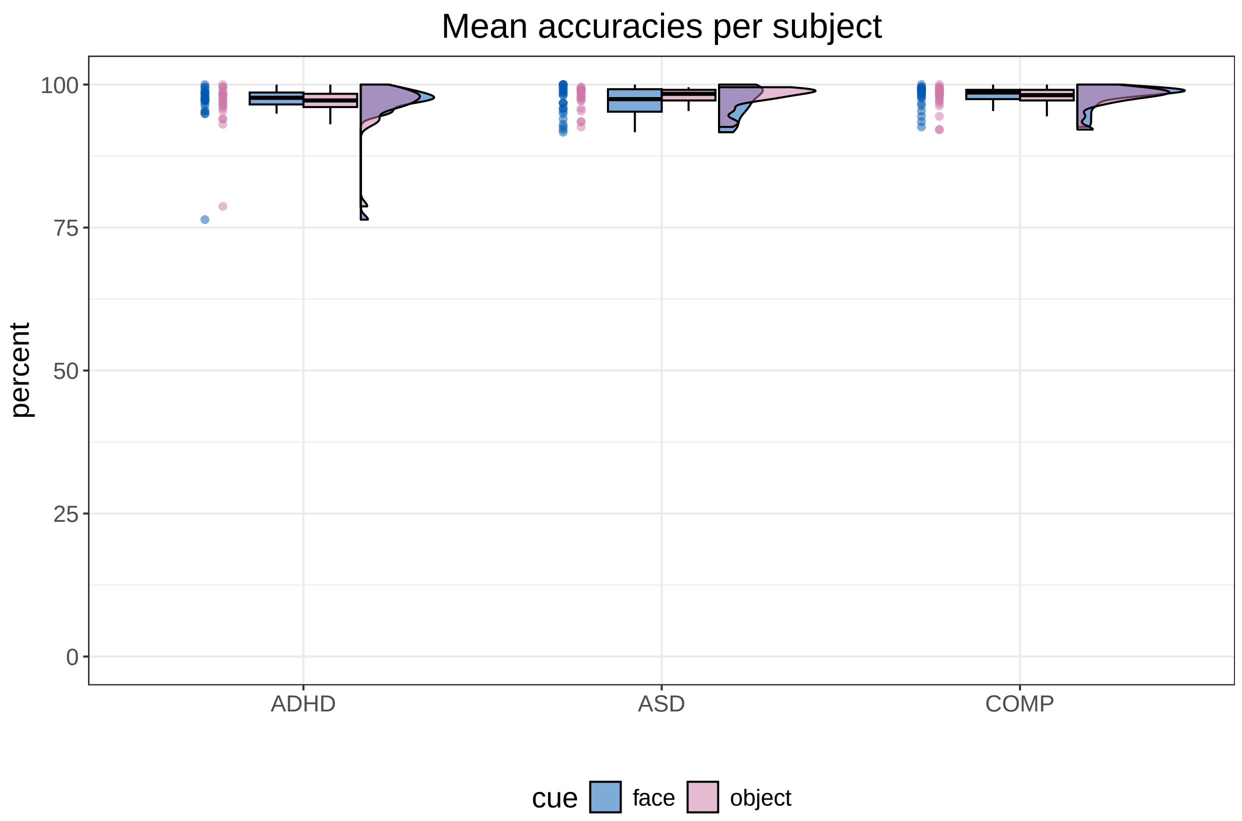
```
df.acc %>%
  group_by(diagnosis) %>%
  summarise(mean_accuracy = mean(acc, na.rm = T), sd_accuracy = sd(acc, na.rm = T))
```

```
## # A tibble: 3 x 3
##   diagnosis mean_accuracy sd_accuracy
##   <fct>             <dbl>       <dbl>
## 1 ADHD               96.5        4.42
## 2 ASD                97.3        2.40
## 3 COMP               97.8        2.04
```

Accuracies were generally high, with a grand average of 97.21% accurate responses across diagnostic groups. None of the models outperformed the intercept-only model, therefore, we conclude that there were no differences between diagnostic groups, cues or their interaction with regards to accuracies.

Plots

```
# overall accuracies
df.acc %>%
  ggplot(aes(diagnosis, acc, fill = cue, colour = cue)) + #
  geom_rain(rain.side = 'r',
boxplot.args = list(color = "black", outlier.shape = NA, show_guide = FALSE, alpha = 0.5),
violin.args = list(color = "black", outlier.shape = NA, alpha = 0.5),
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))) +
  ylim(0, 100) +
  scale_fill_manual(values = custom.col) +
  scale_color_manual(values = custom.col) +
  labs(title = "Mean accuracies per subject", x = "", y = "percent") +
  theme_bw() +
  theme(legend.position = "bottom", plot.title = element_text(hjust = 0.5), legend.direction = "horizon
```


Mean accuracies per subject
percent
ADHD
ASD
COMP
cue
face
object