S1: behavioural analysis with brms

I. S. Plank

2025-02-01

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 kind enough 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.

The visual angle of the target was 1.17 degrees, the visual angle of the cues was 4.25 and the distance of the centre of the target and cue from the fixation cross was 2.67 degrees.

Some general settings

```
# 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 4.4.2 (2024-10-31)"
```

```
## [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 0.15.0"
```

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 performed prior predictive checks as proposed in Schad, Betancourt and Vasishth (2020) using the SBC package based on the original design with three groups. 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. We did not rerun SBC after adding the exploratory sample of ADHD+ASD.

We base our assessment of the hypothesis on the posterior distributions. Therefore, we perform posterior prdictive checks and in some cases simplify the model by aggregating values to improve posterior fit.

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 four diagnostic groups: adults with ADHD, autistic adults, autistic adults with ADHD (explorative) 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.

```
# 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_types = F) %>%
    mutate(
      diagnosis = recode(diagnosis, "CTR" = "COMP")
    )

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

  # load excluded participants (low accuracy)
  exc = c(scan(file.path(dt.path, 'FAB_exc.txt'), what="character", sep=NULL),
          scan(file.path(dt.explo, '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)
df.exp = merge(df.sub %>% select(subID, diagnosis),
                readRDS(file = paste0(dt.explo, '/df_FAB.RDS')), all.y = T) %>%
  mutate_if(is.character, as.factor)

# load data and aggregate emotion discrimination threshold
df.fer = readRDS(file = paste0(dt.path, '/df_FER.RDS')) %>%
  rbind(., readRDS(file = paste0(dt.explo, '/df_FER.RDS'))) %>%
  group_by(subID, emo) %>%
  summarise(
    EDT = mean(disc, na.rm = T)
  ) %>%
  group_by(subID) %>%
  summarise(
    EDT = mean(EDT)
  )

# only keep participants included in the study in the subject data frame
subIDs = as.character(c(unique(df.fab$subID), unique(df.exp$subID)))
df.sub = df.sub %>% filter(subID %in% subIDs) %>%
  # merge with EDT dataframe
  merge(., df.fer, all.x = T)

# load the eye tracking data and only keep participants included in the study,
# so no people with more than 33% mistakes, no people without any saccades
# and no people with too many blinks
df.sac = rbind(readRDS(file.path(dt.explo, "FAB_ET_data.rds")),
                readRDS(file.path(dt.path,  "FAB_ET_data.rds"))) %>%
  merge(., df.sub %>% select(subID, diagnosis), keep.y = T)

# check groups of people who had no relevant saccades at all
df.nosac = df.sac %>% filter(is.na(trl)) %>%
  group_by(diagnosis) %>%
  count()

# anonymise the data
df.fab = df.fab %>%
  mutate(
    PID = subID,
    subID = as.factor(as.numeric(subID))
  )
df.exp = df.exp %>%
  mutate(
    PID = subID,
    subID = as.factor(as.numeric(subID) + length(unique(df.fab$subID)))
  )

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

```
df.fab = df.fab %>% select(-PID)
df.exp = df.exp %>% 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")
tb.gen = xtabs(~ gender + diagnosis + cis, data = df.sub)

# get the gender descriptions of the not-male and not-female participants
gen.desc = unique(tolower(df.sub[df.sub$gender == "dan",]$gender_desc))

# 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$diagnosis == "ADHD",]$age)/
                     sqrt(nrow(df.sub[df.sub$diagnosis == "ADHD",])))
```

```
ASD      = sprintf("%.2f (±%.2f)",
                   mean(df.sub[df.sub$diagnosis == "ASD",]$age),
                   sd(df.sub[df.sub$diagnosis == "ASD",]$age)/
                     sqrt(nrow(df.sub[df.sub$diagnosis == "ASD",])))
`ADHD+ASD`     = sprintf("%.2f (±%.2f)",
                   mean(df.sub[df.sub$diagnosis == "BOTH",]$age),
                   sd(df.sub[df.sub$diagnosis == "BOTH",]$age)/
                     sqrt(nrow(df.sub[df.sub$diagnosis == "BOTH",])))
COMP     = sprintf("%.2f (±%.2f)",
                   mean(df.sub[df.sub$diagnosis == "COMP",]$age),
                   sd(df.sub[df.sub$diagnosis == "COMP",]$age)/
                     sqrt(nrow(df.sub[df.sub$diagnosis == "COMP",])))
logBF10 = sprintf("%.3f", aov.age@bayesFactor[["bf"]])
df.table = data.frame(measurement, ADHD, ASD, `ADHD+ASD`, COMP, logBF10)
df.table = rbind(df.table,
     c(
       "ASRS",
       sprintf("%.2f (±%.2f)",
               mean(df.sub[df.sub$diagnosis == "ADHD",]$ASRS_total),
               sd(df.sub[df.sub$diagnosis == "ADHD",]$ASRS_total)/
                 sqrt(nrow(df.sub[df.sub$diagnosis == "ADHD",]))),
       sprintf("%.2f (±%.2f)",
               mean(df.sub[df.sub$diagnosis == "ASD",]$ASRS_total),
               sd(df.sub[df.sub$diagnosis == "ASD",]$ASRS_total)/
                 sqrt(nrow(df.sub[df.sub$diagnosis == "ASD",]))),
       sprintf("%.2f (±%.2f)",
               mean(df.sub[df.sub$diagnosis == "BOTH",]$ASRS_total),
               sd(df.sub[df.sub$diagnosis == "BOTH",]$ASRS_total)/
                 sqrt(nrow(df.sub[df.sub$diagnosis == "BOTH",]))),
       sprintf("%.2f (±%.2f)",
               mean(df.sub[df.sub$diagnosis == "COMP",]$ASRS_total),
               sd(df.sub[df.sub$diagnosis == "COMP",]$ASRS_total)/
                 sqrt(nrow(df.sub[df.sub$diagnosis == "COMP",]))),
       sprintf("%.3f", aov.ASRS@bayesFactor[["bf"]])
     ),
     c(
       "BDI",
       sprintf("%.2f (±%.2f)",
               mean(df.sub[df.sub$diagnosis == "ADHD",]$BDI_total),
               sd(df.sub[df.sub$diagnosis == "ADHD",]$BDI_total)/
                 sqrt(nrow(df.sub[df.sub$diagnosis == "ADHD",]))),
       sprintf("%.2f (±%.2f)",
               mean(df.sub[df.sub$diagnosis == "ASD",]$BDI_total),
               sd(df.sub[df.sub$diagnosis == "ASD",]$BDI_total)/
                 sqrt(nrow(df.sub[df.sub$diagnosis == "ASD",]))),
       sprintf("%.2f (±%.2f)",
               mean(df.sub[df.sub$diagnosis == "BOTH",]$BDI_total),
               sd(df.sub[df.sub$diagnosis == "BOTH",]$BDI_total)/
                 sqrt(nrow(df.sub[df.sub$diagnosis == "BOTH",]))),
       sprintf("%.2f (±%.2f)",
               mean(df.sub[df.sub$diagnosis == "COMP",]$BDI_total),
               sd(df.sub[df.sub$diagnosis == "COMP",]$BDI_total)/
                 sqrt(nrow(df.sub[df.sub$diagnosis == "COMP",]))),
```

```
    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",]),
            nrow(df.sub[df.sub$diagnosis == "ADHD" & df.sub$gender == "fem",]),
            nrow(df.sub[df.sub$diagnosis == "ADHD" & df.sub$gender == "mal",])),
    sprintf("%d - %d - %d",
            nrow(df.sub[df.sub$diagnosis == "ASD" & df.sub$gender == "dan",]),
            nrow(df.sub[df.sub$diagnosis == "ASD" & df.sub$gender == "fem",]),
            nrow(df.sub[df.sub$diagnosis == "ASD" & df.sub$gender == "mal",])),
    sprintf("%d - %d - %d",
            nrow(df.sub[df.sub$diagnosis == "BOTH" & df.sub$gender == "dan",]),
            nrow(df.sub[df.sub$diagnosis == "BOTH" & df.sub$gender == "fem",]),
            nrow(df.sub[df.sub$diagnosis == "BOTH" & df.sub$gender == "mal",])),
    sprintf("%d - %d - %d",
            nrow(df.sub[df.sub$diagnosis == "COMP" & df.sub$gender == "dan",]),
            nrow(df.sub[df.sub$diagnosis == "COMP" & df.sub$gender == "fem",]),
            nrow(df.sub[df.sub$diagnosis == "COMP" & df.sub$gender == "mal",])),
    sprintf("%.3f", ct.full@bayesFactor[["bf"]])
  ),
  c(
    "IQ",
    sprintf("%.2f (±%.2f)",
            mean(df.sub[df.sub$diagnosis == "ADHD",]$iq),
            sd(df.sub[df.sub$diagnosis == "ADHD",]$iq)/
              sqrt(nrow(df.sub[df.sub$diagnosis == "ADHD",]))),
    sprintf("%.2f (±%.2f)",
            mean(df.sub[df.sub$diagnosis == "ASD",]$iq),
            sd(df.sub[df.sub$diagnosis == "ASD",]$iq)/
              sqrt(nrow(df.sub[df.sub$diagnosis == "ASD",]))),
    sprintf("%.2f (±%.2f)",
            mean(df.sub[df.sub$diagnosis == "BOTH",]$iq),
            sd(df.sub[df.sub$diagnosis == "BOTH",]$iq)/
              sqrt(nrow(df.sub[df.sub$diagnosis == "BOTH",]))),
    sprintf("%.2f (±%.2f)",
            mean(df.sub[df.sub$diagnosis == "COMP",]$iq),
            sd(df.sub[df.sub$diagnosis == "COMP",]$iq)/
              sqrt(nrow(df.sub[df.sub$diagnosis == "COMP",]))),
    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$diagnosis == "ADHD",]$RAADS_total)/
              sqrt(nrow(df.sub[df.sub$diagnosis == "ADHD",]))),
    sprintf("%.2f (±%.2f)",
            mean(df.sub[df.sub$diagnosis == "ASD",]$RAADS_total),
            sd(df.sub[df.sub$diagnosis == "ASD",]$RAADS_total)/
              sqrt(nrow(df.sub[df.sub$diagnosis == "ASD",]))),
    sprintf("%.2f (±%.2f)",
```

```
                mean(df.sub[df.sub$diagnosis == "BOTH",]$RAADS_total),
                sd(df.sub[df.sub$diagnosis == "BOTH",]$RAADS_total)/
                  sqrt(nrow(df.sub[df.sub$diagnosis == "BOTH",]))),
        sprintf("%.2f (±%.2f)",
                mean(df.sub[df.sub$diagnosis == "COMP",]$RAADS_total),
                sd(df.sub[df.sub$diagnosis == "COMP",]$RAADS_total)/
                  sqrt(nrow(df.sub[df.sub$diagnosis == "COMP",]))),
        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$diagnosis == "ADHD",]$TAS_total)/
                  sqrt(nrow(df.sub[df.sub$diagnosis == "ADHD",]))),
        sprintf("%.2f (±%.2f)",
                mean(df.sub[df.sub$diagnosis == "ASD",]$TAS_total),
                sd(df.sub[df.sub$diagnosis == "ASD",]$TAS_total)/
                  sqrt(nrow(df.sub[df.sub$diagnosis == "ASD",]))),
        sprintf("%.2f (±%.2f)",
                mean(df.sub[df.sub$diagnosis == "BOTH",]$TAS_total),
                sd(df.sub[df.sub$diagnosis == "BOTH",]$TAS_total)/
                  sqrt(nrow(df.sub[df.sub$diagnosis == "BOTH",]))),
        sprintf("%.2f (±%.2f)",
                mean(df.sub[df.sub$diagnosis == "COMP",]$TAS_total),
                sd(df.sub[df.sub$diagnosis == "COMP",]$TAS_total)/
                  sqrt(nrow(df.sub[df.sub$diagnosis == "COMP",]))),
        sprintf("%.3f", aov.TAS@bayesFactor[["bf"]])
      )
  ) %>% arrange(measurement)

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

} else {

  load("FAB_data.RData")

}

# print the group of excluded participants based on low accuracy (< 2/3)
kable(df.exc)
```

diagnosis	n
ADHD	1
ASD	1

```
rm(df.exc)

# print the group of the participants included in behavioural and eye tracking
```

```
kable(merge(
  df.sac %>% select(subID, diagnosis) %>% distinct() %>%
    group_by(diagnosis) %>% summarise(`sample size eye tracking` = n()),
  rbind(df.fab, df.exp) %>% select(subID, diagnosis) %>% distinct() %>%
    group_by(diagnosis) %>% summarise(`sample size behavioural` = n())
  ))
```

diagnosis	sample size eye tracking	sample size behavioural
ADHD	16	23
ASD	20	24
BOTH	22	23
COMP	22	24

```
# Note: eye-tracking only collected if calibration accuracy < 0.5, then exclusion:
# 1 due to more than 1/3 blinks
# 2 due to no relevant saccades
# how many have been removed due to no relevant saccades?
kable(df.nosac)
```

diagnosis	n
ASD	1
COMP	1

```
rm(df.nosac)

# print the outcome of the shapiro tests
kable(df.sht %>% arrange(variable))
```

diagnosis	variable	statistic	p	sig
ADHD	ASRS_total	0.9304718	0.1119616	
ASD	ASRS_total	0.9512379	0.2881380	
BOTH	ASRS_total	0.9547574	0.3660251	
COMP	ASRS_total	0.9192136	0.0561186	
ADHD	BDI_total	0.8170622	0.0007279	*
ASD	BDI_total	0.8078769	0.0003974	*
BOTH	BDI_total	0.8324214	0.0013277	*
COMP	BDI_total	0.7421016	0.0000383	*
ADHD	RAADS_total	0.9257590	0.0885842	
ASD	RAADS_total	0.9449707	0.2103029	
BOTH	RAADS_total	0.9524787	0.3291283	
COMP	RAADS_total	0.8449398	0.0017635	*
ADHD	TAS_total	0.9593558	0.4504806	
ASD	TAS_total	0.9181140	0.0530706	
BOTH	TAS_total	0.9445280	0.2245560	
COMP	TAS_total	0.8727860	0.0059679	*
ADHD	age	0.9180376	0.0604839	
ASD	age	0.9492659	0.2611939	
BOTH	age	0.9503886	0.2981103	
COMP	age	0.8104190	0.0004382	*
ADHD	iq	0.9648751	0.5684099	

diagnosis	variable	statistic	p	sig
ASD	iq	0.9579062	0.3978177	
BOTH	iq	0.9583546	0.4309600	
COMP	iq	0.9505974	0.2791247	

```
rm(df.sht)

# print the outcome of the two contingency tables for comparison
# based on full sample with agender, diverse and non-binary in one category
ct.full@bayesFactor
```

```
##                          bf error                      time         code
## Non-indep. (a=1) -4.405795     0 Wed Jan 29 11:23:20 2025 574a4f7b0c75
```

```
# based on female and male participants only
ct.mf@bayesFactor
```

```
##                          bf error                      time         code
## Non-indep. (a=1) -2.737336     0 Wed Jan 29 11:23:20 2025 574a4279f2716
```

```
# combine explorative and original data
df.fab = rbind(df.fab, df.exp)

# set the levels of the diagnosis factor
df.fab$diagnosis = factor(df.fab$diagnosis,
                          levels = c("ADHD", "ASD", "BOTH", "COMP"))

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

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

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

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

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

measurement	ADHD	ASD	ADHD.ASD	COMP	logBF10
ASRS	43.39 (±2.61)	32.54 (±1.62)	46.43 (±1.94)	25.04 (±1.68)	20.618
Age	26.70 (±1.51)	28.58 (±1.46)	30.22 (±1.72)	27.42 (±1.15)	-1.726
BDI	8.09 (±1.77)	11.21 (±2.12)	8.61 (±1.71)	2.25 (±0.62)	7.763

measurement	ADHD	ASD	ADHD.ASD	COMP	logBF10
Gender (diverse/agender/non-binary - female - male)	2 - 9 - 12	0 - 12 - 12	3 - 12 - 8	0 - 11 - 13	-4.406
IQ	108.35 (±2.46)	111.31 (±2.95)	112.93 (±2.34)	109.90 (±1.92)	-2.170
RAADS	92.61 (±8.74)	152.92 (±8.32)	146.52 (±6.97)	44.58 (±7.15)	30.978
TAS	50.22 (±2.58)	63.17 (±1.48)	56.48 (±2.15)	39.08 (±1.93)	20.118

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:

H1a) 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). H1b) ADHD participants react slower than COMP participants in both cue conditions (Sonuga-Barke et al., 2004). H1c) ASD participants react slower than COMP participants in both cue conditions (Ghosn et al., 2018). H1d) Face attention bias is decreased in ASD participants compared to COMP participants (Moore et al., 2012). H1e) Face attention bias in ADHD participants differs from face attention bias in COMP participants.

Full model

Simulation-based calibration

First, we attempted to use a full model for the data. This model includes multiple instances of each stimulus per participant in each of the conditions (face cue or object cue). Therefore, we need slopes for the cue per subject as well as for cue, diagnosis and their interaction for the stimulus.

```
code = "FAB"

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

# 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),
```

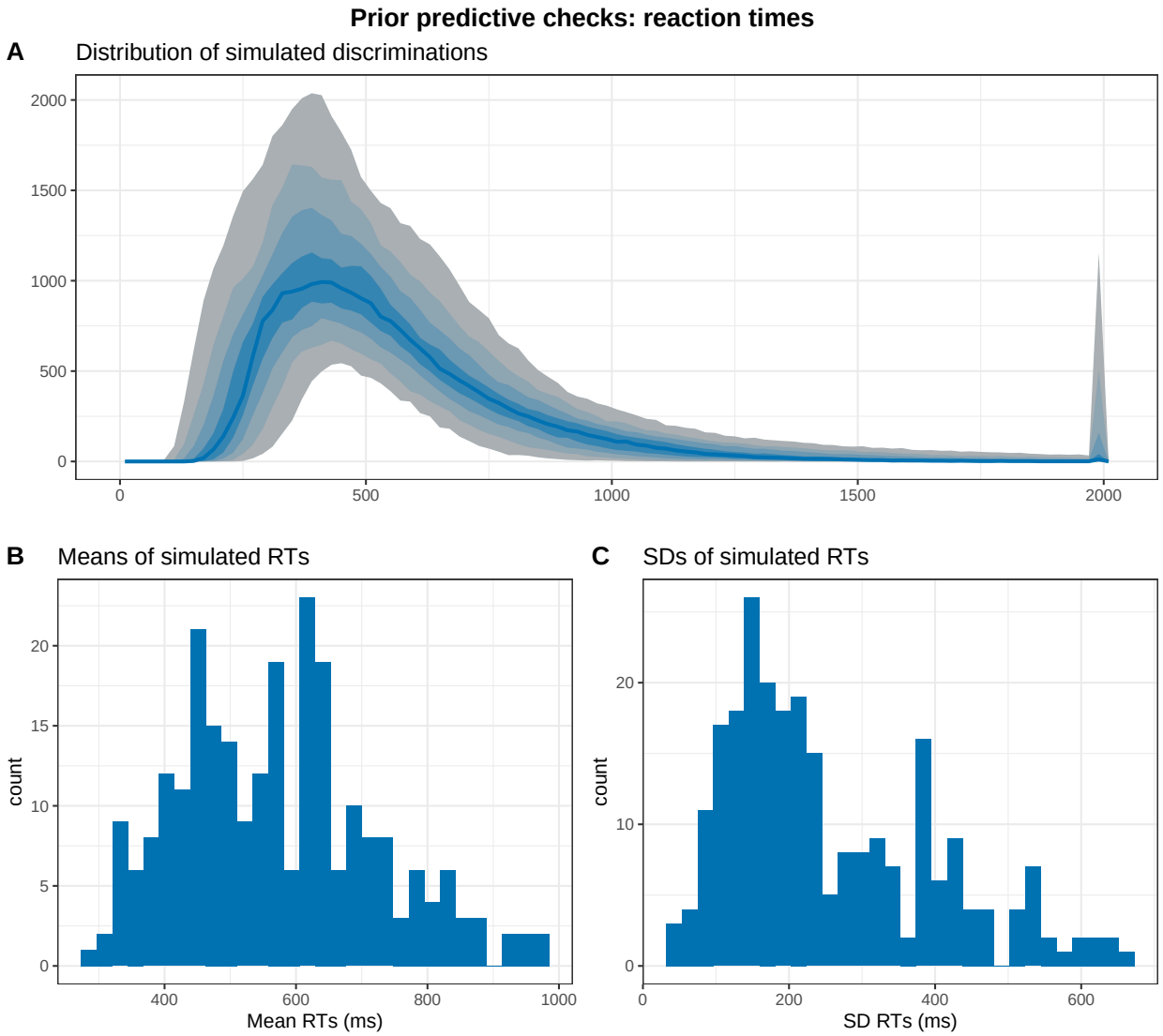
```
  # decreased FAB in ASD subjects based on Moore et al. (2012)
  prior(normal(0.01,  0.04),   class = b, coef = diagnosis2:cue1),
  # no specific expectations for FAB in ADHD
  prior(normal(0,     0.04), class = b),
  # shift
  prior(normal(200,   100), class = ndt)
)

# 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 model 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.

```
# 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)
```

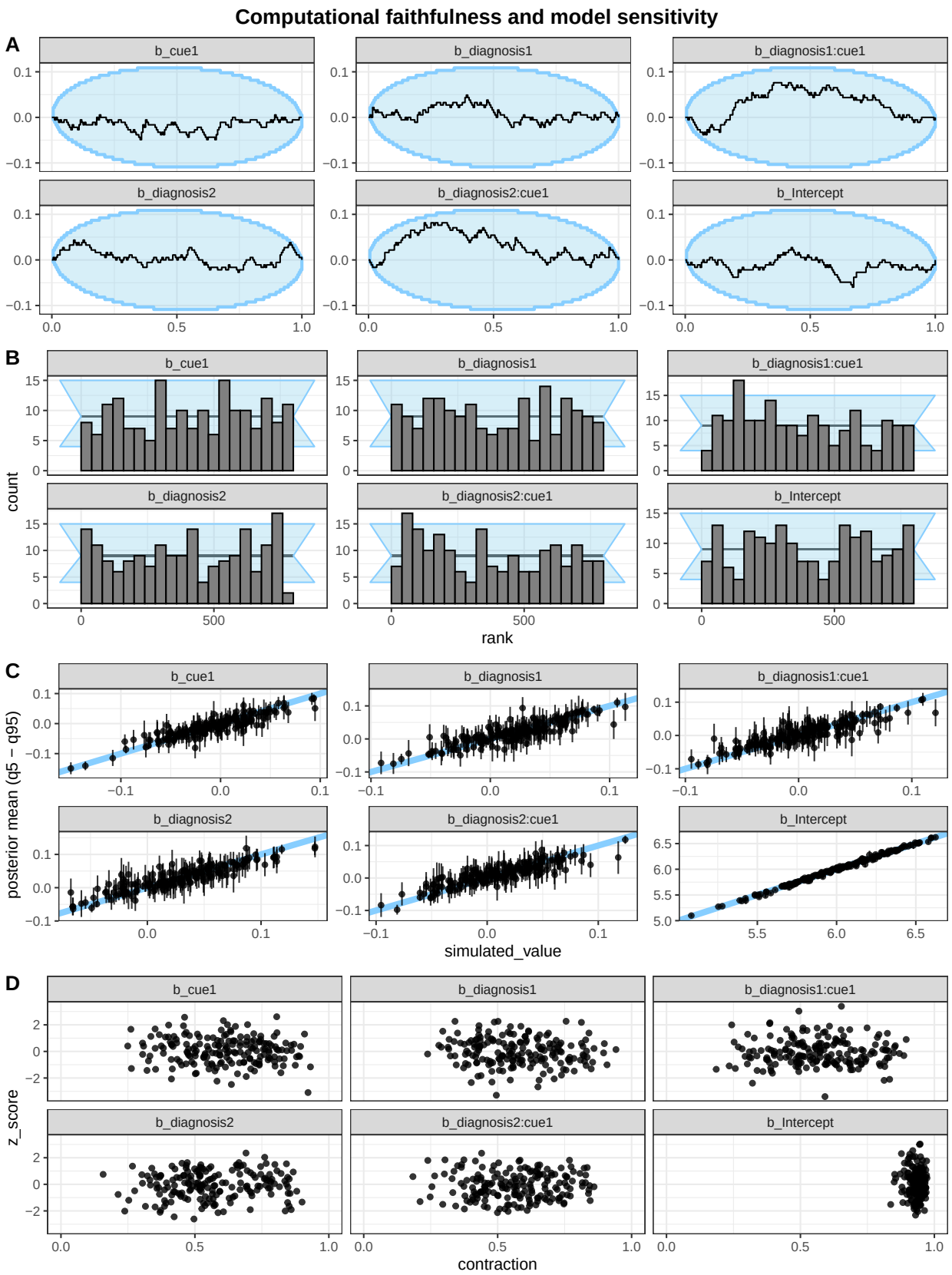
```r
for (i in 1:nrow(truePars)) {
  histmat[,i] = hist(dvfakemat[,i], breaks = breaks, plot = F)$counts
}
# for each bin, compute quantiles across histograms
probs = seq(0.1, 0.9, 0.1)
quantmat= as.data.frame(matrix(NA, nrow=dim(histmat)[1], ncol = length(probs)))
names(quantmat) = paste0("p", probs)
for (i in 1:dim(histmat)[1]) {
  quantmat[i,] = quantile(histmat[i,], p = probs)
}
quantmat$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.

```
# 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
df.results.b = df.results %>%
```

```
  filter(substr(variable, 1, 2) == "b_") %>%
  filter(!(sim_id %in% check$sim_id))
p1 = plot_ecdf_diff(df.results.b) + theme_bw() + theme(legend.position = "none") +
  scale_x_continuous(breaks=scales::pretty_breaks(n = 3)) +
  scale_y_continuous(breaks=scales::pretty_breaks(n = 3))
p2 = plot_rank_hist(df.results.b, bins = 20) + theme_bw() +
  scale_x_continuous(breaks=scales::pretty_breaks(n = 3)) +
  scale_y_continuous(breaks=scales::pretty_breaks(n = 3))
p3 = plot_sim_estimated(df.results.b, alpha = .8) + theme_bw() +
  scale_x_continuous(breaks=scales::pretty_breaks(n = 3)) +
  scale_y_continuous(breaks=scales::pretty_breaks(n = 3))
p4 = plot_contraction(df.results.b,
                      prior_sd = setNames(
                        c(as.numeric(
                          gsub(".*, (.+)\\).*", "\\1",
                               priors[priors$class == "Intercept",]$prior)),
                          as.numeric(
                            gsub(".*, (.+)\\).*", "\\1",
                                 priors[priors$class == "b",]$prior))),
                                          unique(df.results.b$variable))) +
  theme_bw() +
  scale_x_continuous(breaks=scales::pretty_breaks(n = 3)) +
  scale_y_continuous(breaks=scales::pretty_breaks(n = 3))

p = ggarrange(p1, p2, p3, p4, labels = "AUTO", ncol = 1, nrow = 4)
annotate_figure(p, top =
                  text_grob("Computational faithfulness and model sensitivity",
                face = "bold", size = 14))
```



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.

```
# 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:
```

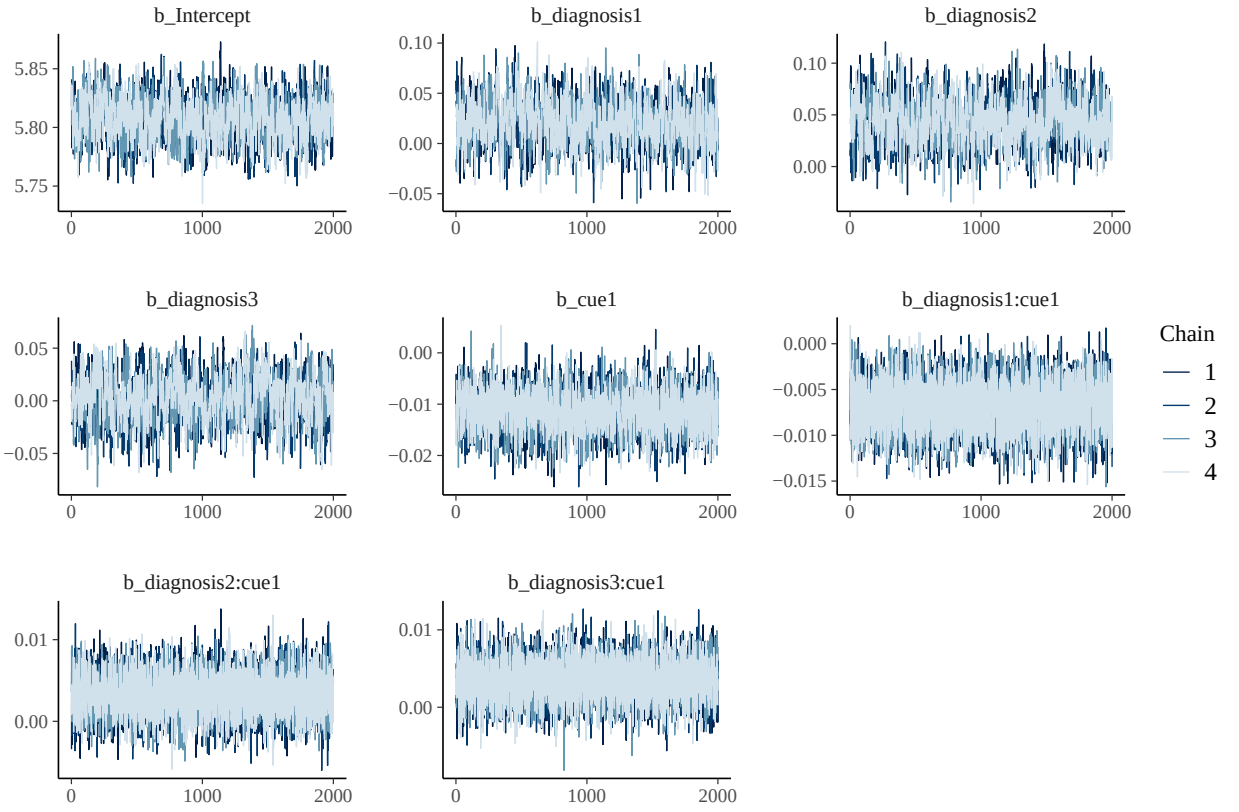
```
## E-BFMI indicated no pathological behavior.
```

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

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

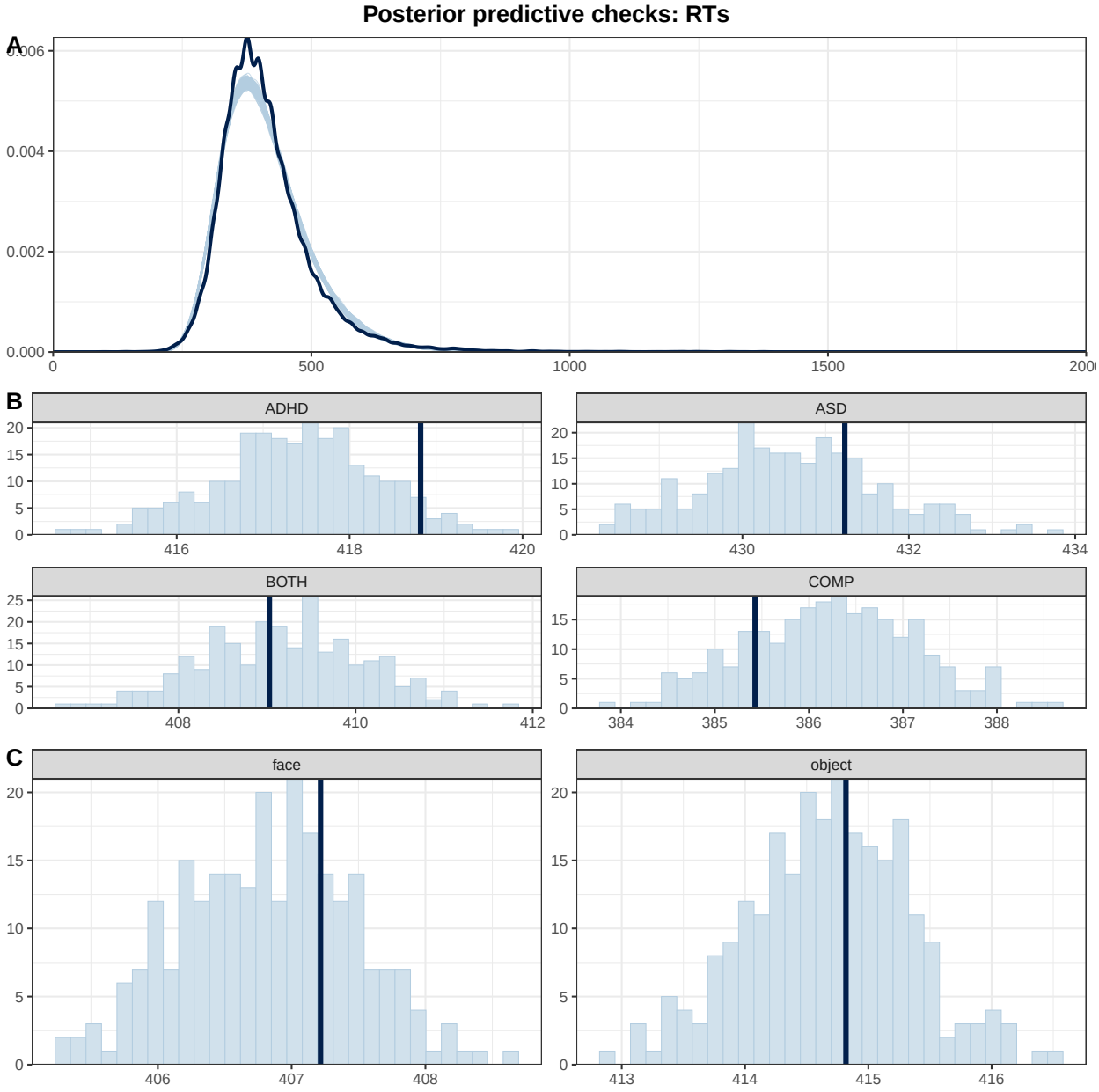
```
# check the trace plots
post.draws = as_draws_df(m.fab)
mcmc_trace(post.draws, regex_pars = "^b_",
           facet_args = list(ncol = 3)) +
  scale_x_continuous(breaks=scales::pretty_breaks(n = 3)) +
  scale_y_continuous(breaks=scales::pretty_breaks(n = 3))
```

```
## 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 no 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")

p = ggarrange(p1, p2, p3,
          nrow = 3, 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 and ASD groups and overestimate the reaction times of the COMP group: the dark blue line shows the mean of the actual dataset while the light blue bars show the distribution of the predicted data.

Since we are interested in accurate estimates, we decide to aggregate with the median of the reaction times per stimulus (face-object cue 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.

```
# keep full dataframe
df.fab.full = df.fab

# 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$diagnosis) = contr.sum(4)[c(1,2,4,3),]
contrasts(df.fab$diagnosis)
```

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

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

```
##      subID      diagnosis        stm           cue           rt.cor
##  1      :  72   ADHD:1656   1_10   : 188   face  :3384   Min.   :256.0
##  2      :  72   ASD :1728   1_11   : 188   object:3384   1st Qu.:364.0
##  3      :  72   BOTH:1656   1_12   : 188                 Median :394.8
##  4      :  72   COMP:1728   1_7    : 188                 Mean   :406.6
##  5      :  72               1_8    : 188                 3rd Qu.:435.5
##  6      :  72               1_9    : 188                 Max.   :919.0
##  (Other):6336               (Other):5640
```

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,
```

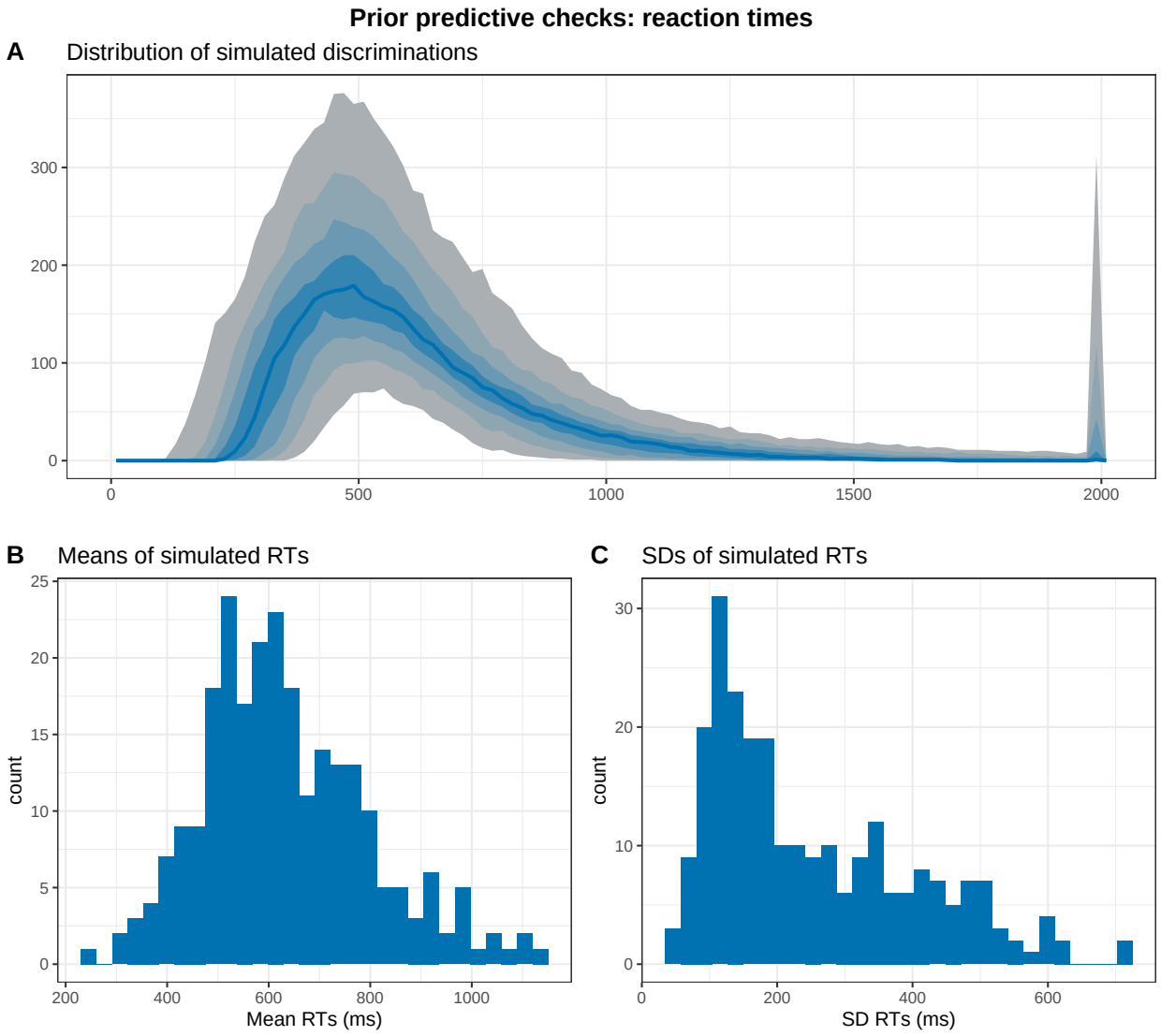
```
                              thin = 50, warmup = 20000, refresh = 2000)
  bck = SBC_backend_brms_from_generator(gen, chains = 4, thin = 1,
                                        init = 0.1, warmup = warm, iter = iter)
  set.seed(468)
  if (file.exists(file.path(cache_dir, sprintf("dat_%s.rds", 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 enough and only few simulated models exhibit issues.

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
```

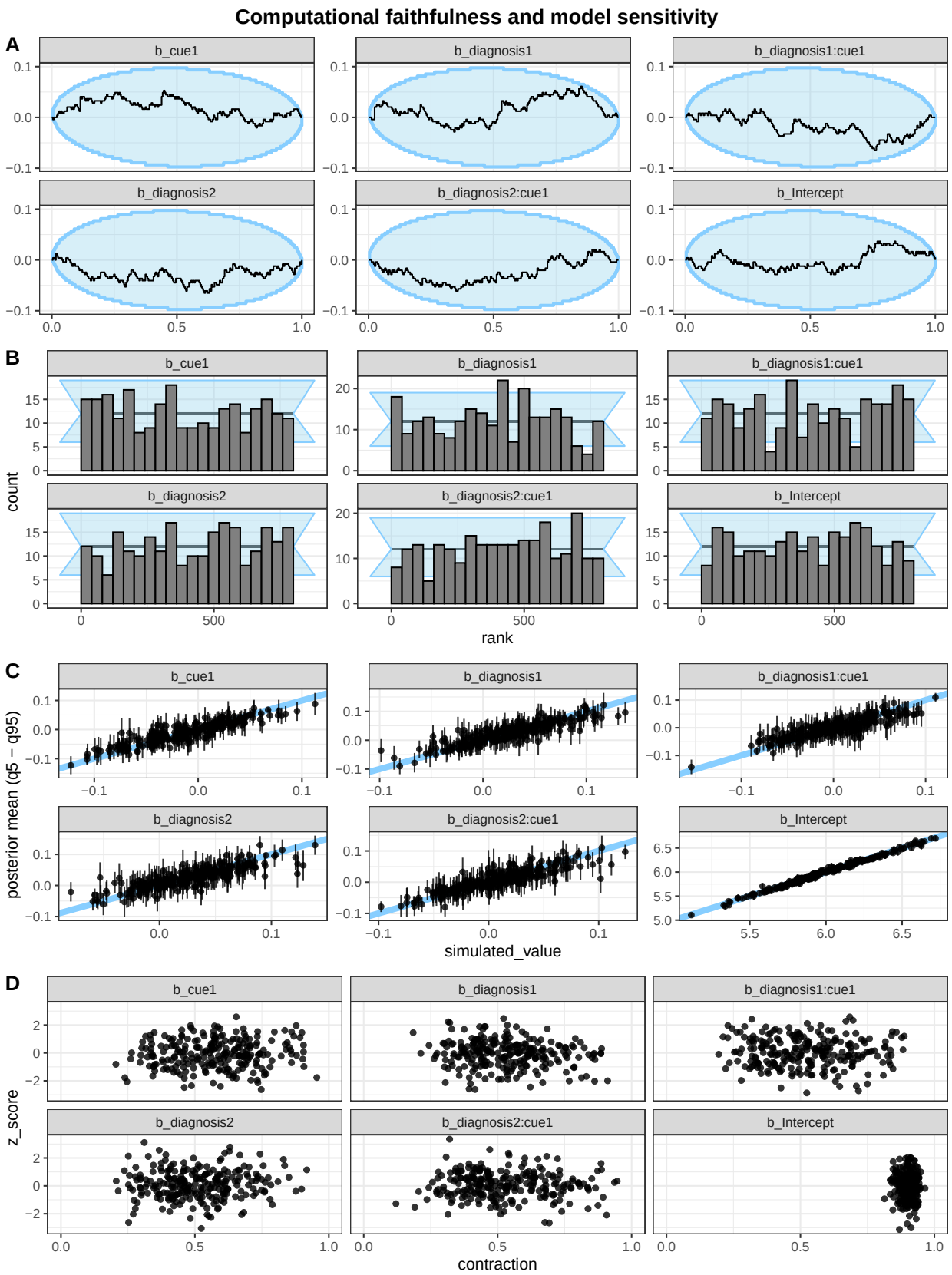
```r
p1 = ggplot(data = quantmat, aes(x = x)) +
  geom_ribbon(aes(ymax = p0.9, ymin = p0.1), fill = c_light) +
  geom_ribbon(aes(ymax = p0.8, ymin = p0.2), fill = c_light_highlight) +
  geom_ribbon(aes(ymax = p0.7, ymin = p0.3), fill = c_mid) +
  geom_ribbon(aes(ymax = p0.6, ymin = p0.4), fill = c_mid_highlight) +
  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))
```



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
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 = .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(as.numeric(
                          gsub(".*, (.+)\\).*", "\\1",
                               priors[priors$class == "Intercept",]$prior)),
                          as.numeric(
                            gsub(".*, (.+)\\).*", "\\1",
                                 priors[priors$class == "b",]$prior))),
                                          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(p1, p2, p3, p4, labels = "AUTO", ncol = 1, nrow = 4)
annotate_figure(p,
                top = text_grob("Computational faithfulness and model sensitivity",
                face = "bold", size = 14))
```



Rank histogramms, sample ECDF, the relationship between the simulated true parameters and the posterior

estimates as well as z-score and posterior contraction of our population-level predictors all are acceptable for this model as well.

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.

```
# 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:
```

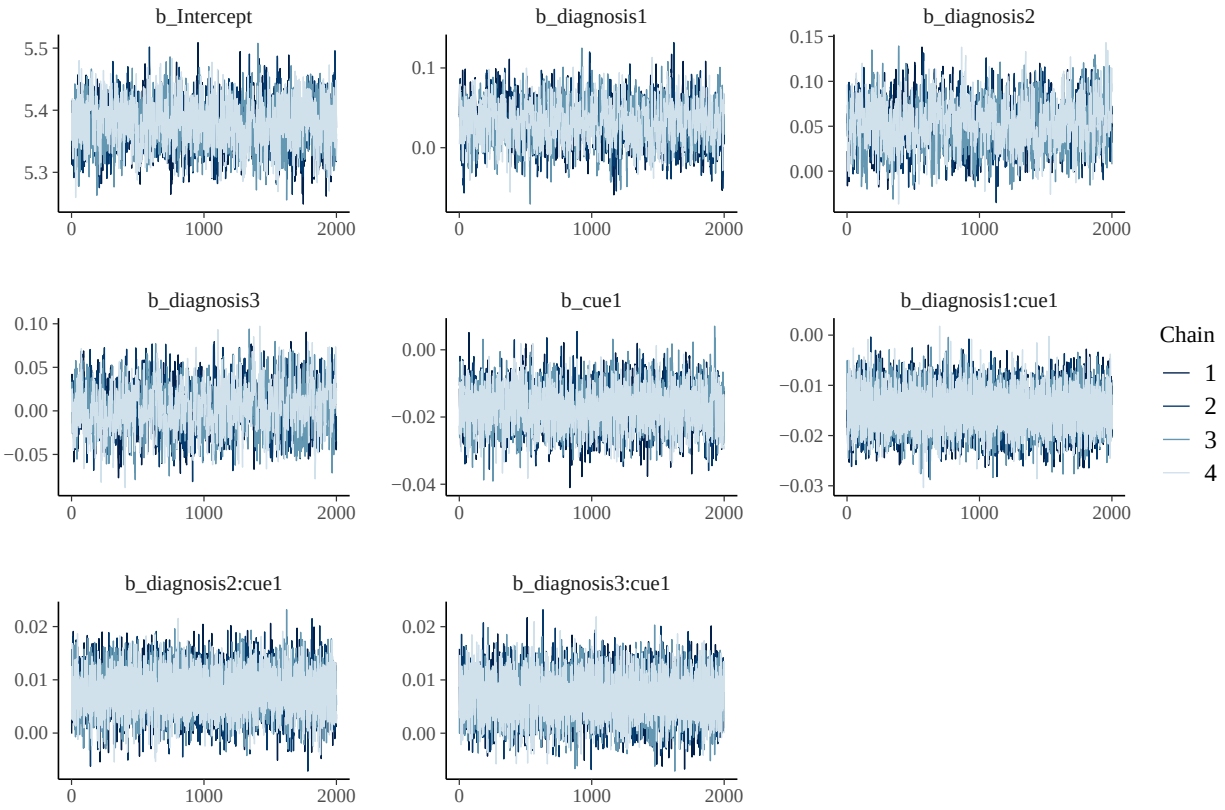
```
## E-BFMI indicated no pathological behavior.
```

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

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

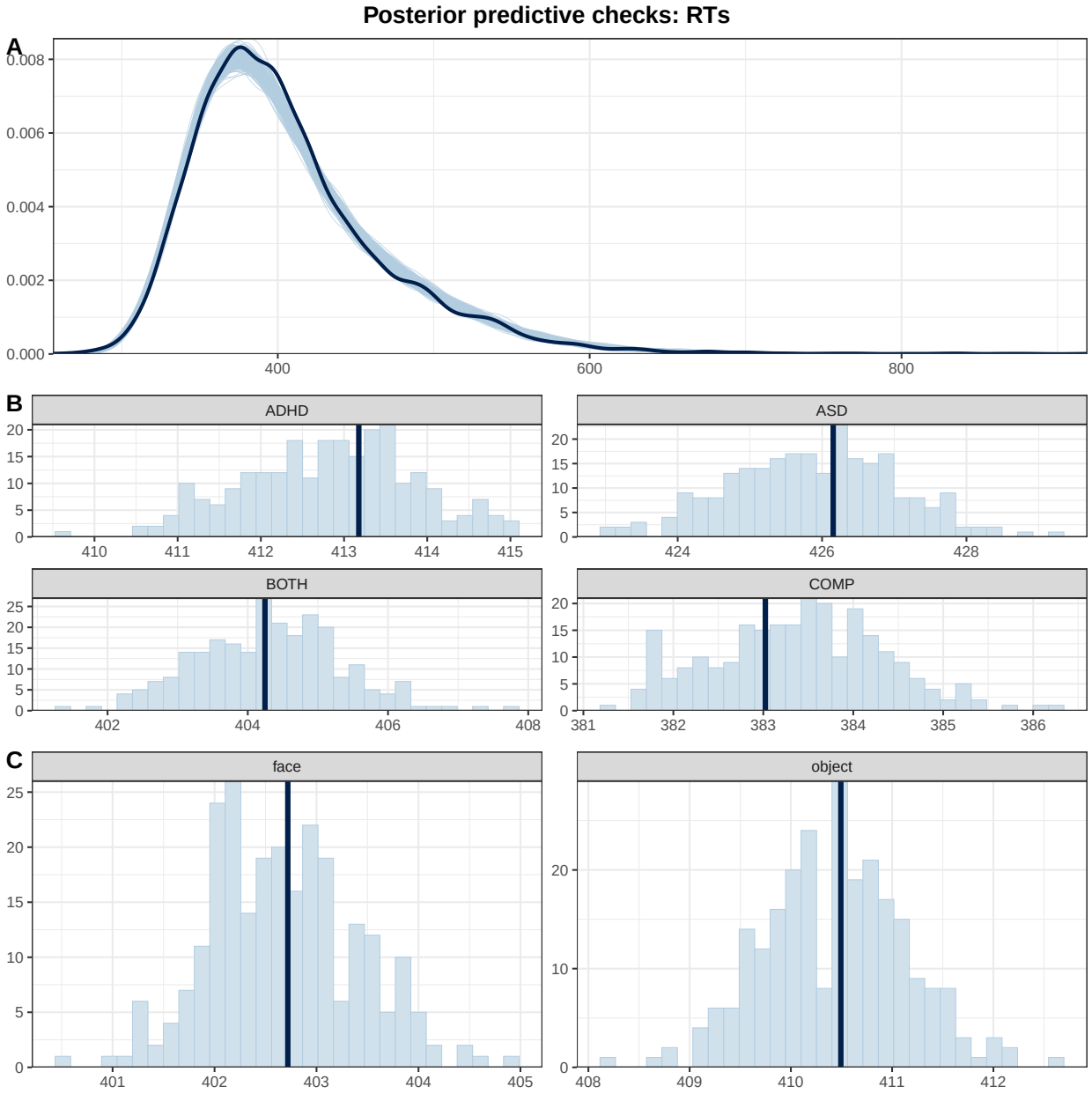
```
# check the trace plots
post.draws = as_draws_df(m.fab)
mcmc_trace(post.draws, regex_pars = "^b_",
           facet_args = list(ncol = 3)) +
  scale_x_continuous(breaks=scales::pretty_breaks(n = 3)) +
  scale_y_continuous(breaks=scales::pretty_breaks(n = 3))
```

```
## 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 no 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")

p = ggarrange(p1, p2, p3,
          nrow = 3, 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.

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.fab)
```

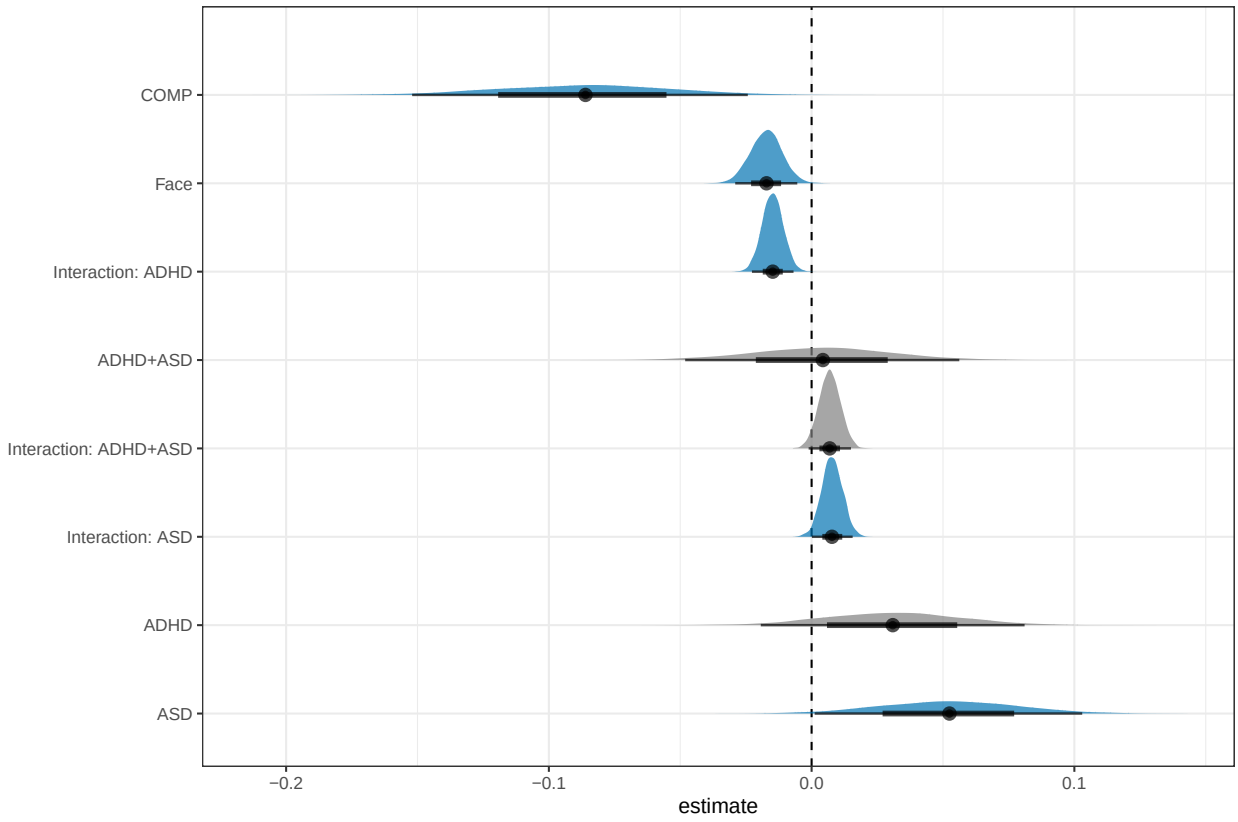
```
##  Family: shifted_lognormal
##   Links: mu = identity; sigma = identity; ndt = identity
## Formula: rt.cor ~ diagnosis * cue + (cue | subID) + (cue * diagnosis | stm)
##    Data: df.fab (Number of observations: 6768)
##   Draws: 4 chains, each with iter = 3000; warmup = 1000; thin = 1;
```

```
##          total post-warmup draws = 8000
##
## Multilevel Hyperparameters:
## ~stm (Number of levels: 36)
##                                     Estimate Est.Error l-95% CI u-95% CI Rhat
## sd(Intercept)                           0.03      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     0.02 1.00
## sd(diagnosis2)                          0.00      0.00     0.00     0.01 1.00
## sd(diagnosis3)                          0.01      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
## sd(cue1:diagnosis3)                     0.00      0.00     0.00     0.01 1.00
## cor(Intercept,cue1)                    -0.30      0.15    -0.57     0.03 1.00
## cor(Intercept,diagnosis1)              -0.12      0.26    -0.60     0.45 1.00
## cor(cue1,diagnosis1)                    0.01      0.26    -0.50     0.51 1.00
## cor(Intercept,diagnosis2)              -0.07      0.29    -0.60     0.51 1.00
## cor(cue1,diagnosis2)                   -0.05      0.29    -0.58     0.51 1.00
## cor(diagnosis1,diagnosis2)             -0.01      0.30    -0.58     0.57 1.00
## cor(Intercept,diagnosis3)               0.06      0.27    -0.49     0.57 1.00
## cor(cue1,diagnosis3)                    0.18      0.27    -0.41     0.66 1.00
## cor(diagnosis1,diagnosis3)             -0.12      0.30    -0.66     0.49 1.00
## cor(diagnosis2,diagnosis3)             -0.08      0.31    -0.65     0.54 1.00
## cor(Intercept,cue1:diagnosis1)          0.03      0.28    -0.51     0.55 1.00
## cor(cue1,cue1:diagnosis1)              -0.06      0.28    -0.58     0.49 1.00
## cor(diagnosis1,cue1:diagnosis1)         0.00      0.30    -0.56     0.57 1.00
## cor(diagnosis2,cue1:diagnosis1)        -0.00      0.30    -0.57     0.56 1.00
## cor(diagnosis3,cue1:diagnosis1)        -0.03      0.30    -0.59     0.54 1.00
## cor(Intercept,cue1:diagnosis2)         -0.21      0.28    -0.69     0.40 1.00
## cor(cue1,cue1:diagnosis2)              -0.01      0.28    -0.54     0.53 1.00
## cor(diagnosis1,cue1:diagnosis2)         0.03      0.29    -0.54     0.58 1.00
## cor(diagnosis2,cue1:diagnosis2)         0.04      0.30    -0.54     0.61 1.00
## cor(diagnosis3,cue1:diagnosis2)        -0.05      0.29    -0.59     0.53 1.00
## cor(cue1:diagnosis1,cue1:diagnosis2)   -0.09      0.31    -0.65     0.53 1.00
## cor(Intercept,cue1:diagnosis3)         -0.10      0.28    -0.62     0.46 1.00
## cor(cue1,cue1:diagnosis3)              -0.15      0.28    -0.65     0.43 1.00
## cor(diagnosis1,cue1:diagnosis3)         0.05      0.30    -0.52     0.61 1.00
## cor(diagnosis2,cue1:diagnosis3)         0.06      0.30    -0.53     0.62 1.00
## cor(diagnosis3,cue1:diagnosis3)        -0.02      0.30    -0.59     0.55 1.00
## cor(cue1:diagnosis1,cue1:diagnosis3)   -0.08      0.30    -0.63     0.52 1.00
## cor(cue1:diagnosis2,cue1:diagnosis3)   -0.01      0.30    -0.58     0.57 1.00
##                                     Bulk_ESS Tail_ESS
## sd(Intercept)                           2578     4611
## sd(cue1)                                2693     4533
## sd(diagnosis1)                          2411     3394
## sd(diagnosis2)                          4155     4612
## sd(diagnosis3)                          3402     4152
## sd(cue1:diagnosis1)                     3823     4565
## sd(cue1:diagnosis2)                     3982     4324
## sd(cue1:diagnosis3)                     3597     3381
## cor(Intercept,cue1)                     1826     3831
## cor(Intercept,diagnosis1)              13054     5814
## cor(cue1,diagnosis1)                   13349     6130
## cor(Intercept,diagnosis2)              16827     5305
```

```
## cor(cue1,diagnosis2)                    17348     5809
## cor(diagnosis1,diagnosis2)               9760     6457
## cor(Intercept,diagnosis3)               15806     5807
## cor(cue1,diagnosis3)                    12478     6278
## cor(diagnosis1,diagnosis3)               7439     6368
## cor(diagnosis2,diagnosis3)               7729     6017
## cor(Intercept,cue1:diagnosis1)          15320     6384
## cor(cue1,cue1:diagnosis1)               16067     5911
## cor(diagnosis1,cue1:diagnosis1)         10215     6390
## cor(diagnosis2,cue1:diagnosis1)          7492     6359
## cor(diagnosis3,cue1:diagnosis1)          7425     7042
## cor(Intercept,cue1:diagnosis2)          11799     6187
## cor(cue1,cue1:diagnosis2)               15752     5325
## cor(diagnosis1,cue1:diagnosis2)         10817     6320
## cor(diagnosis2,cue1:diagnosis2)          8245     6495
## cor(diagnosis3,cue1:diagnosis2)          6974     6717
## cor(cue1:diagnosis1,cue1:diagnosis2)     6068     6426
## cor(Intercept,cue1:diagnosis3)          14585     6447
## cor(cue1,cue1:diagnosis3)               11674     6152
## cor(diagnosis1,cue1:diagnosis3)          8962     5900
## cor(diagnosis2,cue1:diagnosis3)          6889     6461
## cor(diagnosis3,cue1:diagnosis3)          8014     7067
## cor(cue1:diagnosis1,cue1:diagnosis3)     6829     6746
## cor(cue1:diagnosis2,cue1:diagnosis3)     6471     6914
##
## ~subID (Number of levels: 94)
##                     Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(Intercept)           0.21      0.02     0.18     0.24 1.00     1249     2259
## sd(cue1)                0.02      0.00     0.01     0.02 1.00     3696     5257
## cor(Intercept,cue1)    -0.04      0.14    -0.32     0.24 1.00     7658     6980
##
## Regression Coefficients:
##                 Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept           5.38      0.03     5.31     5.45 1.00     1000     1967
## diagnosis1          0.03      0.03    -0.02     0.08 1.00     1053     1809
## diagnosis2          0.05      0.03     0.00     0.10 1.00      959     1959
## diagnosis3          0.00      0.03    -0.05     0.06 1.00     1055     2006
## cue1               -0.02      0.01    -0.03    -0.01 1.00     2074     3747
## diagnosis1:cue1    -0.01      0.00    -0.02    -0.01 1.00     8305     7001
## diagnosis2:cue1     0.01      0.00     0.00     0.02 1.00     8591     7066
## diagnosis3:cue1     0.01      0.00    -0.00     0.01 1.00     8207     7062
##
## Further Distributional Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma     0.13      0.00     0.13     0.14 1.00     5963     5842
## ndt     183.19      5.64   171.94   193.62 1.00     5982     5698
##
## 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 groups
df.m.fab = as_draws_df(m.fab) %>%
  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.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_diagnosis3" ~ "ADHD+ASD",
      "b_COMP"       ~ "COMP",
      "b_cue1"       ~ "Face",
      "b_diagnosis1:cue1" ~ "Interaction: ADHD",
      "b_diagnosis2:cue1" ~ "Interaction: ASD",
      "b_diagnosis3:cue1" ~ "Interaction: 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(credible = c_dark, c_light)) +
  theme(legend.position = "none")
```




```
# H1a: FAB effect in COMP
h1a = hypothesis(m.fab,
                 "0 < 2*(diagnosis1:cue1 + diagnosis2:cue1 + diagnosis3:cue1 - cue1)")
h1a
```

```
## Hypothesis Tests for class b:
##                 Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(2*(diagnosis... < 0    -0.03      0.02    -0.06    -0.01      77.43
##   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 + diagnosis3")
h1b
```

```
## Hypothesis Tests for class b:
##                 Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(2*diagnosis1... < 0    -0.12      0.05     -0.2    -0.04     139.35
##   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.
```

```
# H1c: ASD slower than COMP
h1c = hypothesis(m.fab,
                 "0 < 2*diagnosis2 + diagnosis1 + diagnosis3")
h1c
```

```
## Hypothesis Tests for class b:
##                 Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(2*diagnosis2... < 0    -0.14      0.05    -0.22    -0.06     443.44
##   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.
```

```
# H1d: FAB in ASD decreased compared to COMP
h1d = hypothesis(m.fab,
                 "0 < 4*diagnosis2:cue1 + 2*diagnosis1:cue1 + 2*diagnosis3:cue1")
h1d
```

```
## Hypothesis Tests for class b:
##                 Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(4*diagnosis2... < 0    -0.02      0.01    -0.04     0.01        6.9
##   Post.Prob Star
## 1      0.87
## ---
## '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 in COMP (undirected)
h1e = hypothesis(m.fab,
                 "0 > 4*diagnosis1:cue1 + 2*diagnosis2:cue1 + 2*diagnosis3:cue1",
                 alpha = 0.025)
h1e
```

```
## Hypothesis Tests for class b:
##                 Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(4*diagnosis1... > 0     0.03      0.01        0     0.06      71.73
##   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, "2*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 (2*cue1) < 0    -0.03      0.01    -0.06    -0.01     306.69         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 < -2*cue1 - 2*diagnosis1:cue1", alpha = 0.025)
e2
```

```
## Hypothesis Tests for class b:
##                 Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(-2*cue1-2*di... < 0    -0.06      0.01    -0.09    -0.04        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 < -2*cue1 - 2*diagnosis2:cue1", alpha = 0.025)
e3
```

```
## Hypothesis Tests for class b:
##                 Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(-2*cue1-2*di... < 0    -0.02      0.01    -0.05     0.01       9.39
##   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: FAB effect in ADHD+ASD
e4 = hypothesis(m.fab, "0 < -2*cue1 - 2*diagnosis3:cue1", alpha = 0.025)
e4
```

```
## Hypothesis Tests for class b:
##                 Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(-2*cue1-2*di... < 0    -0.02      0.01    -0.05     0.01      12.84
##   Post.Prob Star
## 1      0.93
## ---
## '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,
                "0 < -2*diagnosis1:cue1 + 2*diagnosis2:cue1", alpha = 0.025)
```

```
e5
```

```
## Hypothesis Tests for class b:
##                 Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(-2*diagnosis... < 0    -0.05      0.01    -0.07    -0.02       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.
```

```
# E6: FAB in ADHD differs from FAB in BOTH
e6 = hypothesis(m.fab,
                "0 < -2*diagnosis1:cue1 + 2*diagnosis3:cue1", alpha = 0.025)
e6
```

```
## Hypothesis Tests for class b:
##                 Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(-2*diagnosis... < 0    -0.04      0.01    -0.07    -0.02    2665.67
##   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.
```

```
# E7: FAB in ASD differs from FAB in BOTH
e7 = hypothesis(m.fab,
                "0 < -2*diagnosis2:cue1 + 2*diagnosis3:cue1", alpha = 0.025)
e7
```

```
## Hypothesis Tests for class b:
##                 Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(-2*diagnosis... < 0        0      0.01    -0.02     0.03       0.79
##   Post.Prob Star
## 1      0.44
## ---
## '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.
```

```
# E8: FAB in COMP differs from FAB in BOTH
e8 = hypothesis(m.fab,
                "0 < 2*diagnosis1:cue1 + 2*diagnosis2:cue1 + 4*diagnosis3:cue1",
                alpha = 0.025)
e8
```

```
## Hypothesis Tests for class b:
##                 Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(2*diagnosis1... < 0    -0.01      0.01    -0.04     0.01       5.26
##   Post.Prob Star
## 1      0.84
## ---
```

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

```
# 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 = rowMeans(select(., matches("COMP_.*")), na.rm = T),
    ADHD = rowMeans(select(., matches("ADHD_.*")), na.rm = T),
    ASD  = rowMeans(select(., matches("ASD_.*")), na.rm = T),
    BOTH = rowMeans(select(., matches("BOTH_.*")), na.rm = T),
    FAB  = rowMeans(select(., matches(".*_object")), na.rm = T) -
      rowMeans(select(., matches(".*_face")), na.rm = T),
    FAB_COMP = COMP_object - COMP_face,
    FAB_ADHD = ADHD_object - ADHD_face,
    FAB_ASD  = ASD_object  - ASD_face,
    FAB_BOTH = BOTH_object - BOTH_face,
    h1b  = ADHD - COMP,
    h1c  = ASD - COMP,
    h1d  = FAB_COMP - FAB_ASD,
    h1e  = FAB_ADHD - FAB_COMP,
    BOTH_COMP = BOTH - COMP
  )
```

As hypothesised, both autistic adults and adults with ADHD exhibited increased overall reaction times compared with the COMP group (COMP - ADHD: *estimate* = -0.12 [-0.2, -0.04], *posterior probability* = 99.29%; COMP - ASD: *estimate* = -0.14 [-0.22, -0.06], *posterior probability* = 99.78%). The model predicts that participants in the comparison group react 25.081ms [5.021, 45.753] faster than the participants in the ADHD group and 29.908ms [9.084, 50.831] faster than autistic participants. Additionally, the model predicts that the participants in the comparison group react 19.023ms [-1.549, 38.963] faster than adults in the ADHD+ASD group.

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 (*estimate* = -0.03 [-0.06, -0.01], *posterior probability* = 98.72%). FAB was not credibly decreased in ASD participants compared to COMP participants (*estimate* = -0.02 [-0.04, 0.01], *posterior probability* = 87.34%). However, FAB was credibly higher in the ADHD than the COMP group (*estimate* = 0.03 [0, 0.06], *posterior probability* = 98.62%). Specifically, predicted reaction times based on the model estimate a FAB of 6.881ms [1.072, 12.924] in the COMP group, 14.469ms [8.076, 21.209] in the ADHD group, 4.37ms [-2.193, 10.975] in the ASD group as well as 4.566ms [-1.623, 10.709] in the ADHD+ASD group.

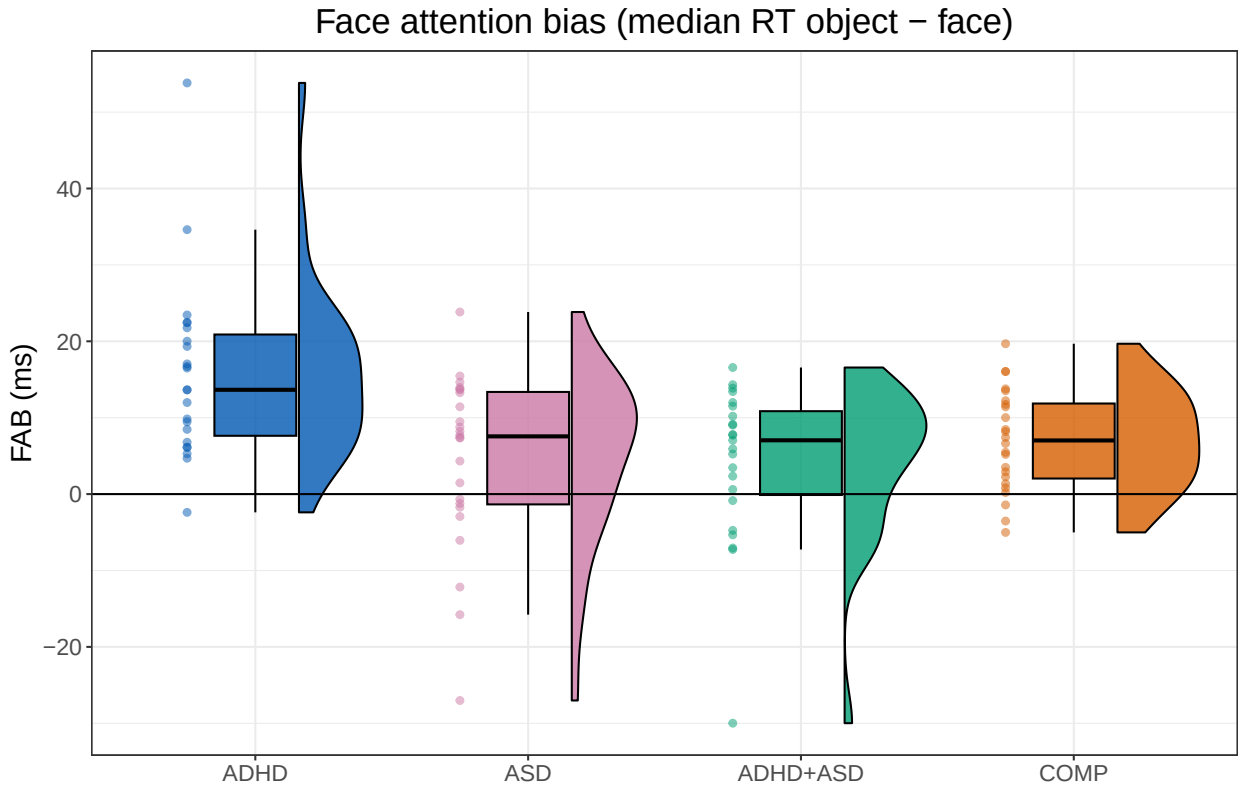
These estimates are reflected in our exploration of the FAB in the separate clinical groups with our model revealing a credible FAB effect in the ADHD (*estimate* = -0.06 [-0.09, -0.04], *posterior probability* = 100%) but not the ASD (*estimate* = -0.02 [-0.05, 0.01], *posterior probability* = 90.38%) and the ADHD+ASD group (*estimate* = -0.02 [-0.05, 0.01], *posterior probability* = 92.77%).

Plots

```
# overall median reaction times
df.fab %>%
  group_by(subID, diagnosis, cue) %>%
  summarise(
    rt.cor = mean(rt.cor, na.rm = T)
    ) %>%
  mutate(
    diagnosis = recode(diagnosis, "BOTH" = "ADHD+ASD")
  ) %>%
  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 = .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))) +
  ylim(0, 700) +
  scale_fill_manual(values = custom.col2) +
  scale_color_manual(values = custom.col2) +
  labs(title = "Median reaction times",
       x = "",
       y = "reaction time (ms)") +
  theme_bw() +
  theme(legend.position = "bottom",
        plot.title = element_text(hjust = 0.5),
        legend.direction = "horizontal",
        text = element_text(size = 15))
```


```
ggsave("Fig3_rts.pdf",
       units = "mm",
       width  = 170,
       height = 100,
       dpi    = 300)

# 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])) %>%
  mutate(
    diagnosis = recode(diagnosis, "BOTH" = "ADHD+ASD")
  ) %>%
  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 = .8),
violin.args = list(color = "black", outlier.shape = NA, alpha = .8),
boxplot.args.pos = list(
  position = ggpp::position_dodgenudge(x = 0, width = 0.3), width = 0.3
),
point.args = list(show_guide = FALSE, alpha = .5),
violin.args.pos = list(
```

```
  width = 0.6, position = position_nudge(x = 0.16)),
point.args.pos = list(position = ggpp::position_dodgenudge(x = -0.25, width = 0.1))) +
  geom_hline(yintercept = 0) +
  scale_fill_manual(values = custom.col) +
  scale_color_manual(values = custom.col) +
  labs(title = "Face attention bias (median RT object - face)",
       x = "",
       y = "FAB (ms)") +
  theme_bw() +
  theme(legend.position = "none",
        plot.title = element_text(hjust = 0.5),
        legend.direction = "horizontal",
        text = element_text(size = 15))
```




```
ggsave("Fig4_FAB.pdf",
       units = "mm",
       width  = 170,
       height = 70,
       dpi    = 300)
```

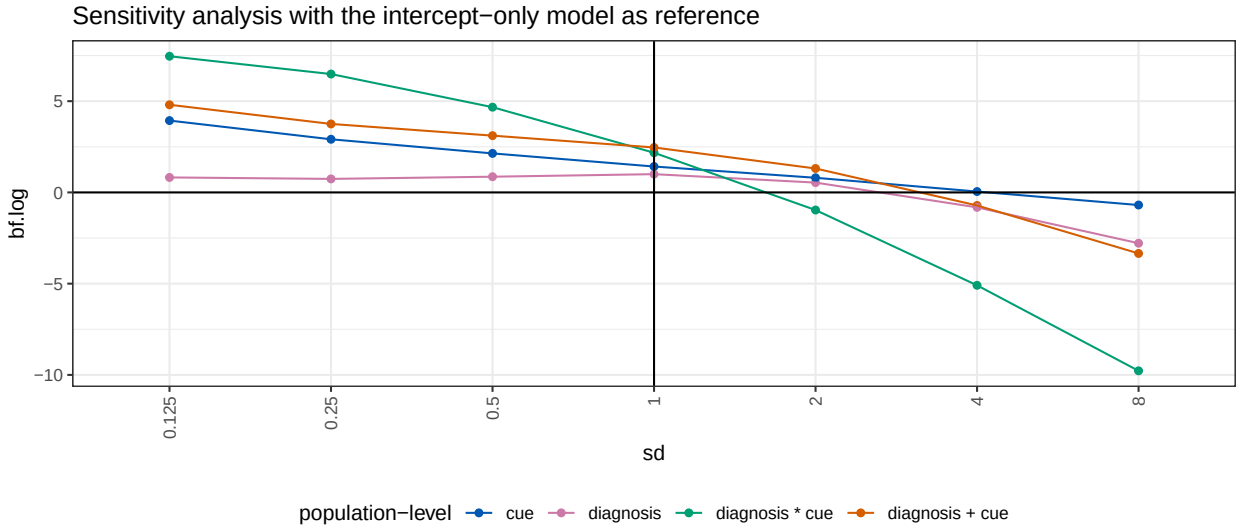
Bayes factor analysis

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.

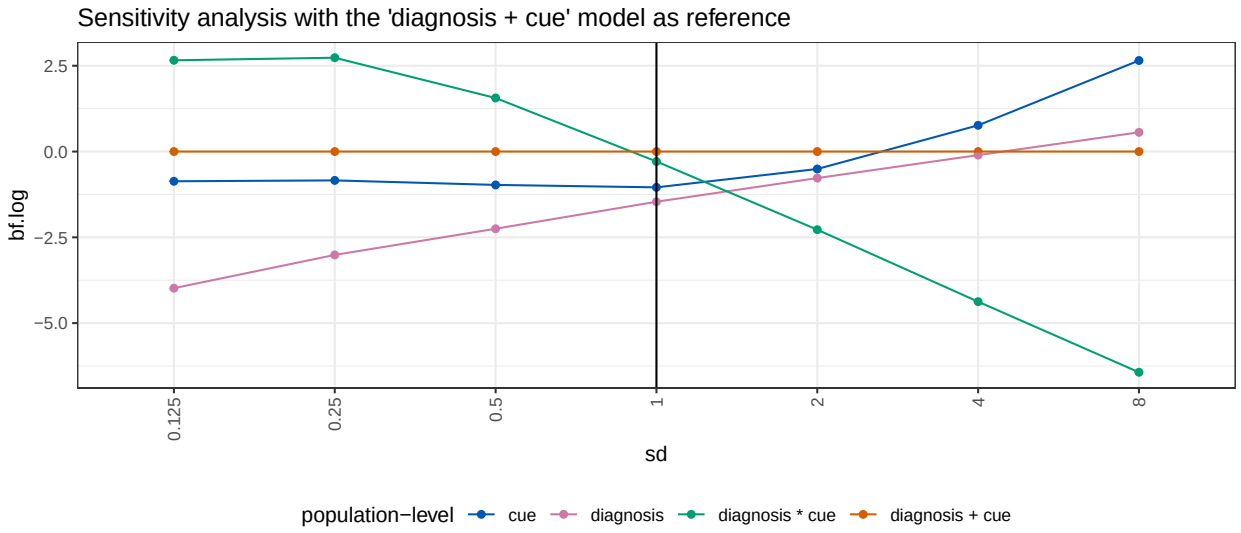
```
# set the directory in which to save results
sense_dir = file.path(getwd(), "_brms_sens_cache")
main.code = "fab"

# describe priors
pr.descriptions = c("chosen",
  "sdx2",    "sdx4",   "sdx8",
  "sdx0.5", "sdx0.25", "sdx0.125"
  )

# 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
  set.seed(7788)
  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 to take a closer look.

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



Here, we can see that the results are not very consistent. While with narrower priors, the model including both main effects and the interaction outperforms the others, our chosen priors result in the model only including the main effects having the highest Bayes Factor. Then, with wider priors, the model only including the main effect cue performs best for our data.

```r
# 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	2.463
diagnosis * cue	2.173
cue	1.419
diagnosis	1.000

The comparison of the models reveals the model containing both main effects 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 including the interaction ($\log(BF) = 0.289$), anecdotal evidence in favour of this model compared to the model including only the predictor cue ($\log(BF) = 1.043$), moderate evidence in favour of this model compared to the one only including the predictor diagnosis ($\log(BF) = 1.463$) as well as strong evidence in favour of this model compared to the model only including the intercept on the population-level ($\log(BF) = 2.463$).

S1.3 Explorative analysis of RTs on trials without saccades

Since saccadic behaviour may have influenced reaction times, we rerun the model concerning reaction times with only trials where no saccades were produced.

```r
# aggregate saccades
df.sac.agg = df.sac %>%
  group_by(subID, diagnosis, trl) %>%
  summarise(
    n.sac = n()
  )

# merge together
df.sac.fab = merge(df.fab.full, df.sac.agg, all.x = T) %>%
  # only keep people with at least one saccade
  group_by(subID) %>%
  mutate(
    n.sac.total = sum(n.sac, na.rm = T)
  ) %>%
  # only keep trials with saccades %>%
  filter(n.sac.total > 0) %>%
  filter(is.na(n.sac)) %>%
  # compute median rt.cor
  group_by(subID, diagnosis, cue, stm) %>%
  summarise(
    rt.cor = median(rt.cor, na.rm = T)
  )

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

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

```
## face      1
## object   -1
```

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

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

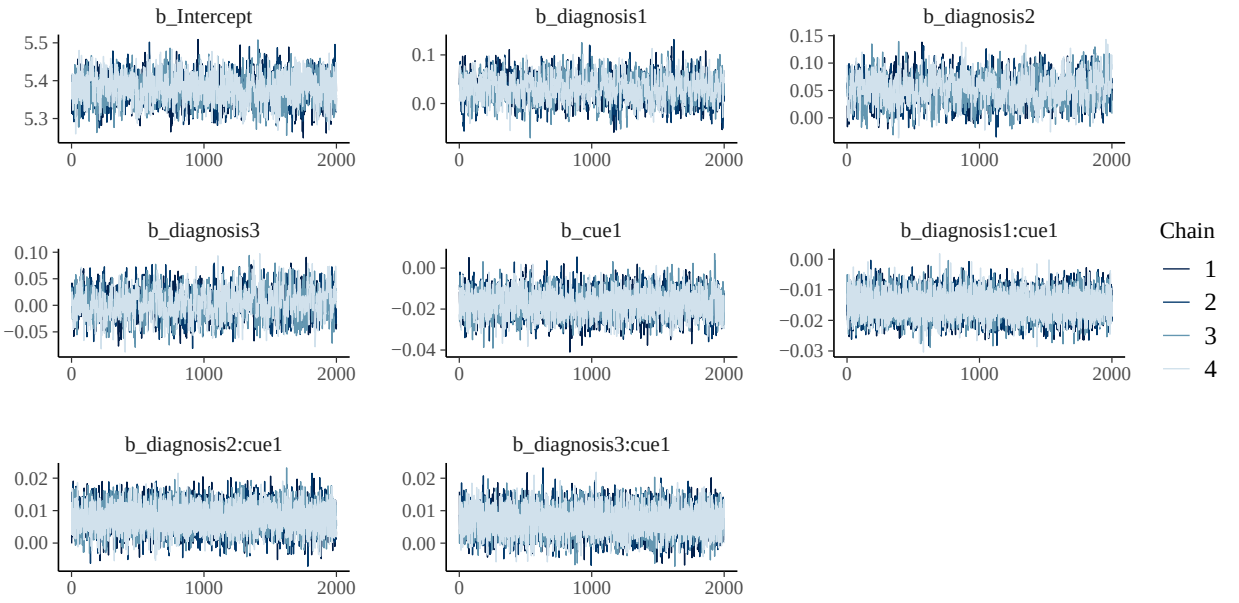
```
# run the model
set.seed(1357)
m.rtsac = brm(f.fab,
            df.sac.fab, prior = priors,
            family = shifted_lognormal,
            iter = iter, warmup = warm,
            backend = "cmdstanr", threads = threading(8),
            file = "m_fab_sac"
            )
rstan::check_hmc_diagnostics(m.fab$fit)
```

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

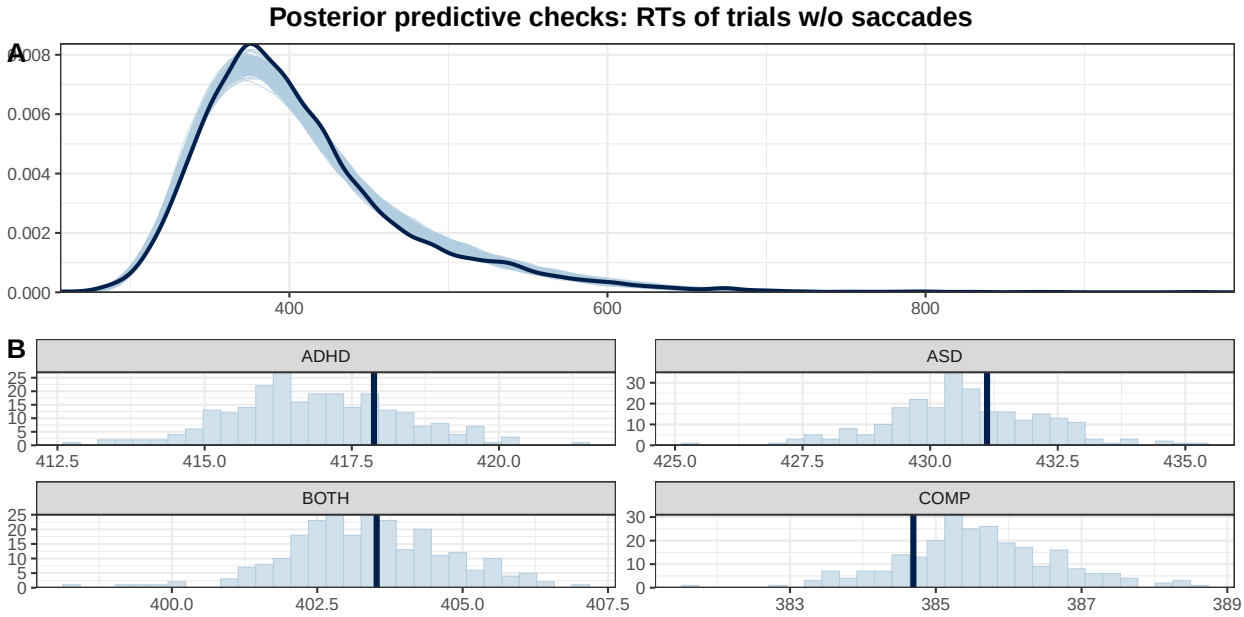
```
# check that rhats are below 1.01
sum(brms::rhat(m.fab) >= 1.01, na.rm = T)
```

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

```
# 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))
```




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

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

df.sac.fab = df.sac.fab %>% drop_na()

# distributions of means compared to the real values per group
p2 = ppc_stat_grouped(df.sac.fab$rt.cor, post.pred, df.sac.fab$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: RTs of trials w/o saccades",
                face = "bold", size = 14))
```




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

```
##  Family: shifted_lognormal
##   Links: mu = identity; sigma = identity; ndt = identity
## Formula: rt.cor ~ diagnosis * cue + (cue | subID) + (cue * diagnosis | stm)
##    Data: df.sac.fab (Number of observations: 5507)
##   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)                           0.03      0.00     0.02     0.04 1.00
## sd(cue1)                                0.03      0.01     0.03     0.05 1.00
## sd(diagnosis1)                          0.01      0.00     0.00     0.02 1.00
## sd(diagnosis2)                          0.01      0.00     0.00     0.02 1.00
## sd(diagnosis3)                          0.01      0.00     0.00     0.02 1.00
## sd(cue1:diagnosis1)                     0.01      0.01     0.00     0.02 1.00
## sd(cue1:diagnosis2)                     0.01      0.01     0.00     0.02 1.00
## sd(cue1:diagnosis3)                     0.01      0.00     0.00     0.02 1.00
## cor(Intercept,cue1)                    -0.35      0.16    -0.64    -0.01 1.00
## cor(Intercept,diagnosis1)              -0.05      0.29    -0.59     0.51 1.00
## cor(cue1,diagnosis1)                   -0.03      0.28    -0.57     0.52 1.00
## cor(Intercept,diagnosis2)              -0.05      0.28    -0.57     0.51 1.00
## cor(cue1,diagnosis2)                    0.01      0.29    -0.55     0.55 1.00
## cor(diagnosis1,diagnosis2)             -0.04      0.30    -0.62     0.54 1.00
## cor(Intercept,diagnosis3)               0.06      0.28    -0.49     0.59 1.00
## cor(cue1,diagnosis3)                    0.14      0.28    -0.44     0.64 1.00
## cor(diagnosis1,diagnosis3)             -0.10      0.31    -0.67     0.50 1.00
## cor(diagnosis2,diagnosis3)             -0.03      0.30    -0.60     0.56 1.00
## cor(Intercept,cue1:diagnosis1)         -0.01      0.27    -0.54     0.53 1.00
## cor(cue1,cue1:diagnosis1)               0.12      0.27    -0.44     0.61 1.00
## cor(diagnosis1,cue1:diagnosis1)         0.02      0.30    -0.56     0.58 1.00
```

```
## cor(diagnosis2,cue1:diagnosis1)          0.05      0.30    -0.54     0.60 1.00
## cor(diagnosis3,cue1:diagnosis1)          0.04      0.30    -0.56     0.59 1.00
## cor(Intercept,cue1:diagnosis2)          -0.13      0.27    -0.62     0.42 1.00
## cor(cue1,cue1:diagnosis2)               -0.14      0.27    -0.62     0.41 1.00
## cor(diagnosis1,cue1:diagnosis2)         -0.01      0.30    -0.58     0.57 1.00
## cor(diagnosis2,cue1:diagnosis2)          0.05      0.30    -0.55     0.60 1.00
## cor(diagnosis3,cue1:diagnosis2)         -0.05      0.30    -0.60     0.53 1.00
## cor(cue1:diagnosis1,cue1:diagnosis2)    -0.11      0.30    -0.66     0.50 1.00
## cor(Intercept,cue1:diagnosis3)           0.01      0.28    -0.53     0.56 1.00
## cor(cue1,cue1:diagnosis3)               -0.20      0.29    -0.69     0.41 1.00
## cor(diagnosis1,cue1:diagnosis3)          0.06      0.30    -0.53     0.63 1.00
## cor(diagnosis2,cue1:diagnosis3)          0.02      0.30    -0.56     0.60 1.00
## cor(diagnosis3,cue1:diagnosis3)         -0.04      0.30    -0.61     0.54 1.00
## cor(cue1:diagnosis1,cue1:diagnosis3)    -0.08      0.30    -0.63     0.51 1.00
## cor(cue1:diagnosis2,cue1:diagnosis3)    -0.05      0.30    -0.61     0.55 1.00
##                                    Bulk_ESS Tail_ESS
## sd(Intercept)                          3106     4456
## sd(cue1)                               2601     4435
## sd(diagnosis1)                         3532     3349
## sd(diagnosis2)                         3258     3804
## sd(diagnosis3)                         3163     3949
## sd(cue1:diagnosis1)                    2447     3969
## sd(cue1:diagnosis2)                    2361     3162
## sd(cue1:diagnosis3)                    3166     3185
## cor(Intercept,cue1)                    1794     3337
## cor(Intercept,diagnosis1)              8837     5660
## cor(cue1,diagnosis1)                  10064     6009
## cor(Intercept,diagnosis2)             10168     5496
## cor(cue1,diagnosis2)                   9424     5287
## cor(diagnosis1,diagnosis2)             6878     6178
## cor(Intercept,diagnosis3)              9295     6046
## cor(cue1,diagnosis3)                   8671     5783
## cor(diagnosis1,diagnosis3)             6278     5790
## cor(diagnosis2,diagnosis3)             6698     5977
## cor(Intercept,cue1:diagnosis1)         9404     5857
## cor(cue1,cue1:diagnosis1)              8548     5082
## cor(diagnosis1,cue1:diagnosis1)        5979     6263
## cor(diagnosis2,cue1:diagnosis1)        5349     6031
## cor(diagnosis3,cue1:diagnosis1)        5925     6047
## cor(Intercept,cue1:diagnosis2)         8687     5244
## cor(cue1,cue1:diagnosis2)              8930     5654
## cor(diagnosis1,cue1:diagnosis2)        6850     6249
## cor(diagnosis2,cue1:diagnosis2)        5902     6296
## cor(diagnosis3,cue1:diagnosis2)        5842     6562
## cor(cue1:diagnosis1,cue1:diagnosis2)   5714     6531
## cor(Intercept,cue1:diagnosis3)         9582     5473
## cor(cue1,cue1:diagnosis3)              7071     5759
## cor(diagnosis1,cue1:diagnosis3)        6610     6347
## cor(diagnosis2,cue1:diagnosis3)        6585     6409
## cor(diagnosis3,cue1:diagnosis3)        6512     5979
## cor(cue1:diagnosis1,cue1:diagnosis3)   6043     6392
## cor(cue1:diagnosis2,cue1:diagnosis3)   5949     6805
##
## ~subID (Number of levels: 78)
```

```
##                     Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(Intercept)           0.23      0.02     0.20     0.27 1.00     1095     2041
## sd(cue1)                0.02      0.00     0.02     0.03 1.00     3296     5354
## cor(Intercept,cue1)    -0.08      0.15    -0.36     0.21 1.00     5259     5851
##
## Regression Coefficients:
##                 Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept           5.34      0.04     5.27     5.41 1.00      997     2546
## diagnosis1          0.03      0.03    -0.02     0.09 1.01      901     1664
## diagnosis2          0.05      0.03    -0.00     0.11 1.01     1012     2153
## diagnosis3          0.00      0.03    -0.06     0.06 1.00      773     1973
## cue1               -0.02      0.01    -0.03    -0.00 1.00     2539     3794
## diagnosis1:cue1    -0.01      0.01    -0.03    -0.00 1.00     4191     5250
## diagnosis2:cue1     0.01      0.01    -0.00     0.02 1.00     5165     5560
## diagnosis3:cue1     0.00      0.01    -0.01     0.02 1.00     5345     6070
##
## Further Distributional Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sigma     0.16      0.00     0.16     0.17 1.00     4289     5109
## ndt     193.98      5.10   183.53   203.36 1.00     4197     4904
##
## 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).
```

```
# H1a: FAB effect in COMP
h1a = hypothesis(m.rtsac,
                 "0 < 2*(diagnosis1:cue1 + diagnosis2:cue1 + diagnosis3:cue1 - cue1)")
h1a
```

```
## Hypothesis Tests for class b:
##                 Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(2*(diagnosis... < 0    -0.03      0.02    -0.06        0      33.48
##   Post.Prob Star
## 1      0.97    *
## ---
## '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.rtsac,
                 "0 < 2*diagnosis1 + diagnosis2 + diagnosis3")
h1b
```

```
## Hypothesis Tests for class b:
##                 Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(2*diagnosis1... < 0    -0.12      0.06    -0.21    -0.03      45.51
##   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.rtsac,
                 "0 < 2*diagnosis2 + diagnosis1 + diagnosis3")
h1c
```

```
## Hypothesis Tests for class b:
##                 Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(2*diagnosis2... < 0    -0.14      0.06    -0.23    -0.05     147.15
##   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 compared to COMP
h1d = hypothesis(m.rtsac,
                 "0 < 4*diagnosis2:cue1 + 2*diagnosis1:cue1 + 2*diagnosis3:cue1")
h1d
```

```
## Hypothesis Tests for class b:
##                 Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(4*diagnosis2... < 0    -0.02      0.02    -0.05     0.02       3.87
##   Post.Prob Star
## 1      0.79
## ---
## '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 in COMP (undirected)
h1e = hypothesis(m.rtsac,
                 "0 > 4*diagnosis1:cue1 + 2*diagnosis2:cue1 + 2*diagnosis3:cue1",
                 alpha = 0.025)
h1e
```

```
## Hypothesis Tests for class b:
##                 Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(4*diagnosis1... > 0     0.03      0.02    -0.01     0.07      13.71
##   Post.Prob Star
## 1      0.93
## ---
## '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.
```

This model confirmed the same hypotheses with the exception for the difference in FAB between COMP and ADHD group which is not credible anymore (*estimate* = 0.03 [-0.01, 0.07], *posterior probability* = 93.2%). This change could indicate that difference in saccadic behaviour may partly drive these group differences.

S1.4 Explorative analysis of errors

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

```
# aggregate mean accuracy per condition and person
df.acc = df.fab.full %>%
  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: 8 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 BOTH      face   acc          0.946 0.241        ""
## 6 BOTH      object acc          0.903 0.0291       "*"
## 7 COMP      face   acc          0.814 0.000500     "*"
## 8 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                       -1.693491 4.545861e-06 Sat Feb  1 08:37:53 2025
## cue                             -1.813963 6.252025e-04 Sat Feb  1 08:37:53 2025
## diagnosis + cue                 -3.537668 1.056114e-02 Sat Feb  1 08:37:53 2025
## diagnosis + cue + diagnosis:cue -6.143428 2.191559e-02 Sat Feb  1 08:37:53 2025
##                                         code
## diagnosis                       358714ba5a20
## cue                             358713e00b0fe
## diagnosis + cue                 35871287b09ae
## diagnosis + cue + diagnosis:cue 358713422fed7
```

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

```
## # A tibble: 8 x 4
## # Groups:   diagnosis [4]
##   diagnosis cue    mean_accuracy se_accuracy
##   <fct>     <fct>          <dbl>       <dbl>
## 1 ADHD      face            96.6       0.968
## 2 ADHD      object          96.4       0.895
## 3 ASD       face            96.9       0.557
## 4 ASD       object          97.6       0.413
## 5 BOTH      face            97.8       0.350
## 6 BOTH      object          97.8       0.411
## 7 COMP      face            97.8       0.409
## 8 COMP      object          97.7       0.431
```

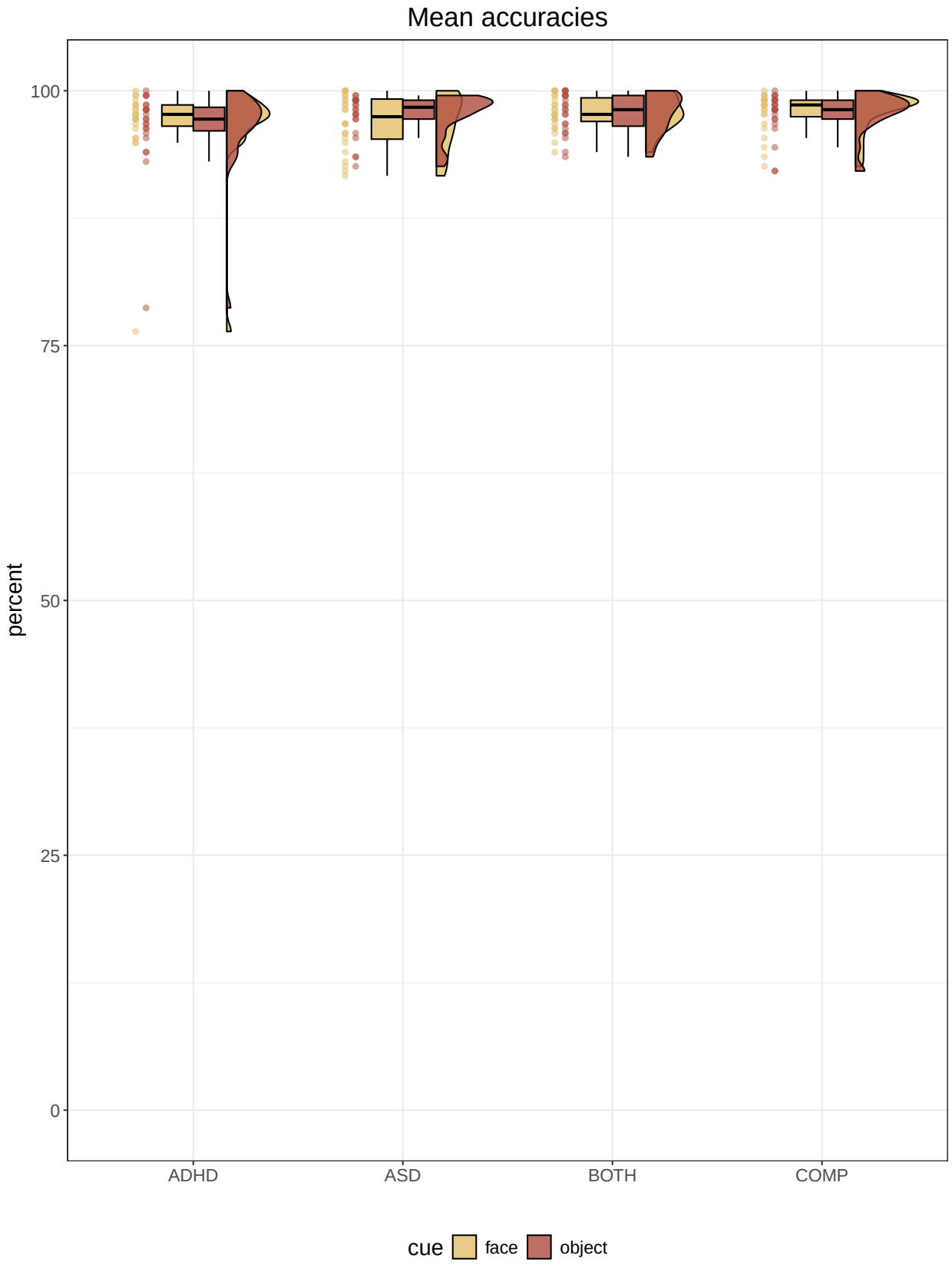
```
df.acc %>%
  group_by(diagnosis) %>%
  summarise(mean_accuracy = mean(acc, na.rm = T),
            se_accuracy = sd(acc, na.rm = T)/sqrt(n()))
```

```
## # A tibble: 4 x 3
##   diagnosis mean_accuracy se_accuracy
##   <fct>             <dbl>       <dbl>
## 1 ADHD               96.5       0.652
## 2 ASD                97.3       0.347
## 3 BOTH               97.8       0.267
## 4 COMP               97.8       0.294
```

Accuracies were generally high, with a grand average of 97.36% 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 = .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))) +
  ylim(0, 100) +
  scale_fill_manual(values = custom.col2) +
  scale_color_manual(values = custom.col2) +
  labs(title = "Mean accuracies", x = "", y = "percent") +
  theme_bw() +
  theme(legend.position = "bottom",
        plot.title = element_text(hjust = 0.5),
        legend.direction = "horizontal",
        text = element_text(size = 15))
```


Mean accuracies
percent
100
75
50
25
0
ADHD
ASD
BOTH
COMP
cue
face
object