VMM: Behavioural analysis with brms

I. S. Plank

2024-12-11

Introduction

[!ADD info on study and task]

Some general settings

```
# number of simulations
nsim = 250

# set number of iterations and warmup for models
iter = 4500
warm = 1500
```

Package versions

```
## [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 perform prior predictive checks as proposed in Schad, Betancourt and Vasishth (2020) using the SBC package. To do so, we create 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 values are saved.

Then, we create graphs showing the prior predictive distribution of the simulated discrimination threshold to check whether our priors fit our general expectations about the data. Next, we perform checks of computational faithfulness and model sensitivity as proposed by Schad, Betancourt and Vasishth (2020) and implemented

in the SBC package. We create models for each of the simulated datasets. Last, we calculate performance metrics for each of these models, focusing on the population-level parameters.

Preparation and group comparisons

First, we load the data and combine it with demographic information including the diagnostic status of the subjects. Then, all predictors are set to sum contrasts. We have a look at the demographics describing our three diagnostic groups: autistic adults and adults with ADHD as well as a comparison group of adults without any 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("VMM_data.RData")) {

  # set file paths
  fl.path = '/home/emba/Documents/EMBA'
  dt.path = paste(fl.path, 'BVET', sep = "/")

  # load the preprocessed data
  load(file.path(dt.path, "VMM_preprocessed.RData"))

  # load list of participants which are included in the fMRI analysis
  df.inc = read_delim(file.path(fl.path, "VMM_analysis/all_use-new")) %>%
    filter(diagnosis != "pilot") %>% select(subID)

  # load list of participants which are included in fMRI ET analysis
  df.ETinc = read_csv(file = file.path(dt.path, "VMM_ET_inc.csv"), show_col_types = F)

  # load demographic information
  df.sub = read_csv(file.path("/home/emba/Documents/EMBA/CentraXX", "EMBA_centraXX.csv"),
                    show_col_types = F) %>%
    mutate(
      diagnosis = recode(diagnosis, "CTR" = "COMP")
    ) %>% filter(subID %in% df.inc$subID)

  # merge together and anonymise
  df.disc = merge(df.disc, df.sub %>% select(subID, diagnosis), all.y = T) %>%
    mutate(
      subID = as.factor(as.numeric(as.factor(subID)))
    )
  df.tsk  = merge(df.tsk,  df.sub %>% select(subID, diagnosis), all.y = T) %>%
    mutate(
      subID = as.factor(as.numeric(as.factor(subID)))
    )

  # set the center for the fixation cross
  x.centre = 512 # 1024
  y.centre = 332 # 768 but moved up 52 pixels
  r = 100 # radius of the AOI in the centre

  # classify fixates (centre or not), aggregate and anonymise the data
  df.fix.agg = df.fix %>%
    mutate(
```

```
    AOI.fix = if_else(dist.centre <= r, "centre", "periphery")
  ) %>%
  group_by(subID, AOI.fix) %>%
  summarise(
    fix.dur = sum(duration)
  ) %>%
  pivot_wider(names_from = AOI.fix, values_from = fix.dur, names_prefix = "fix.") %>%
  mutate(
    fix.total = fix.centre + fix.periphery,
    fix.prop  = fix.centre / fix.total
  ) %>%
  merge(., df.ETinc %>% select(subID, diagnosis) %>% distinct()) %>%
  ungroup() %>%
  mutate(
    rfix.total = rank(fix.total),
    rfix.prop  = rank(fix.prop),
    diagnosis  = as.factor(diagnosis),
    subID = as.factor(as.numeric(as.factor(subID)))
  )

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

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

# most of the measures are not normally distributed;
# therefore, we compute ranks for these outcomes
df.sub = df.sub %>%
  mutate(
    rage   = rank(age),
    rASRS  = rank(ASRS_total),
    rBDI   = rank(BDI_total),
    rIQ    = rank(iq),
    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(rIQ    ~ diagnosis, data = df.sub)
aov.BDI   = anovaBF(rBDI   ~ diagnosis, data = df.sub)
aov.ASRS  = anovaBF(rASRS  ~ 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",])))
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$diagnosis == "COMP",])))
logBF10 = sprintf("%.3f", aov.age@bayesFactor[["bf"]])
df.table = data.frame(measurement, ADHD, ASD, COMP, logBF10)
df.table = rbind(df.table,
     c(
       "ASRS",
       sprintf("%.2f (±%.2f)",
               mean(df.sub[df.sub$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 == "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$diagnosis == "COMP",]$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 == "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 == "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$diagnosis == "COMP",]$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$diagnosis == "COMP",]$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.disc, df.tsk, df.table, df.sht, ct.full, ct.mf, df.fix.agg, file = "VMM_data.RData")

} else {

  load("VMM_data.RData")

}

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

diagnosis	n
ADHD	23
ASD	22
COMP	22

```
# print the group of included participants with eye tracking
kable(df.fix.agg %>% select(subID, diagnosis) %>% distinct() %>% group_by(diagnosis) %>% count())
```

diagnosis	n
ADHD	16
ASD	11
CTR	15

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

diagnosis	variable	statistic	p	sig
ADHD	ASRS_total	0.8853468	0.0127659	*
ASD	ASRS_total	0.9359134	0.1628911	
COMP	ASRS_total	0.9245532	0.0944984	
ADHD	BDI_total	0.8118744	0.0005972	*
ASD	BDI_total	0.9321099	0.1357637	

diagnosis	variable	statistic	p	sig
COMP	BDI_total	0.7642997	0.0001454	*
ADHD	RAADS_total	0.9246881	0.0840017	
ASD	RAADS_total	0.9689757	0.6873604	
COMP	RAADS_total	0.8321183	0.0016657	*
ADHD	TAS_total	0.9538566	0.3510493	
ASD	TAS_total	0.9316339	0.1484328	
COMP	TAS_total	0.8775261	0.0108404	*
ADHD	age	0.9336163	0.1309202	
ASD	age	0.9041832	0.0360733	*
COMP	age	0.8009872	0.0005185	*
ADHD	iq	0.9510717	0.3079566	
ASD	iq	0.8796113	0.0118768	*
COMP	iq	0.9537644	0.3742442	

```
rm(df.sht)

# print the outcome of the contingency table
ct.full@bayesFactor
```

```
##                          bf error                    time          code
## Non-indep. (a=1) -2.932155     0 Tue Nov 12 14:33:05 2024 6f20f4032308d
```

```
ct.mf@bayesFactor
```

```
##                          bf error                    time          code
## Non-indep. (a=1) -1.088224     0 Tue Nov 12 14:33:05 2024 6f20f11b62ecd
```

```
# only keep the relevant rtcs
df.rtc = df.tsk %>%
  filter(sdt == "hit" & !is.na(rtc)) %>%
  ungroup() %>%
  mutate_if(is.character, as.factor)
df.disc = df.disc %>%
  ungroup() %>%
  # subtract the discrimination rate from the "perfect" number to be able to
  # use a poisson in the model
  mutate(
    negdisc = 240 - disc
  ) %>%
  mutate_if(is.character, as.factor)

# set and print the contrasts
contrasts(df.rtc$diagnosis)  = contr.sum(3)
contrasts(df.rtc$diagnosis)
```

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

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

```
##      [,1] [,2]
```

```
## ADHD    1    0
## ASD     0    1
## COMP   -1   -1
```

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

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

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

measurement	ADHD	ASD	COMP	logBF10
ASRS	44.30 (±2.45)	30.36 (±1.85)	24.95 (±1.76)	10.858
Age	26.96 (±1.51)	29.36 (±1.69)	27.77 (±1.22)	-1.786
BDI	7.87 (±1.62)	7.95 (±1.26)	2.45 (±0.66)	4.821
Gender (diverse/agender/non-binary - female - male)	2 - 9 - 12	0 - 14 - 8	0 - 10 - 12	-2.932
IQ	106.65 (±2.23)	114.98 (±2.98)	109.70 (±2.10)	-1.101
RAADS	94.09 (±8.15)	148.09 (±8.80)	44.50 (±7.62)	20.882
TAS	51.70 (±2.32)	NA (±NA)	39.18 (±2.11)	15.092

Analysis of reaction times (hits-only)

Full model

Group-level: subject and trials, trials with diagnosis slope

```
code = "VMM_rtc_full"

# full model formula
f.rtc = brms::bf(rtc ~ diagnosis + (1 | subID) + (diagnosis | trl) )

# 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(0.1,   0.1),  class = sd),
  prior(lkj(2),              class = cor),
  # 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),
  # shift
  prior(normal(100,   50),   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 {
  # set the seed
  set.seed(2468)
  # perform the SBC
  gen = SBC_generator_brms(f.rtc, data = df.rtc, 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)
  if (!file.exists(file.path(cache_dir, sprintf("dat_%s.rds", code)))) {
    dat = generate_datasets(gen, nsim)
    saveRDS(dat, file.path(cache_dir, sprintf("dat_%s.rds", code)))
  } else {
    dat = readRDS(file.path(cache_dir, sprintf("dat_%s.rds", code)))
  }
  # set the seed again
  set.seed(2468)
  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 19 of 250 simulations had at least one parameter that had an rhat of at least 1.05, and 60 models had divergent samples (mean number of samples of the simulations with divergent samples: 190.23). This suggests serious problems with this model. We try to drop the group-level slope of diagnostics for the trials and run it again.

Group-level: subject and trials, no slopes

```
code = "VMM_rtc_full_int"

# full model formula
f.rtc = brms::bf(rtc ~ diagnosis + (1 | subID) + (1 | trl) )

# 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(0.1,   0.1),  class = sd),
  # 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),
  # shift
  prior(normal(100,   50),   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 {
  # set the seed
  set.seed(2468)
  # perform the SBC
  gen = SBC_generator_brms(f.rtc, data = df.rtc, 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)
  if (!file.exists(file.path(cache_dir, sprintf("dat_%s.rds", code)))) {
    dat = generate_datasets(gen, nsim)
    saveRDS(dat, file.path(cache_dir, sprintf("dat_%s.rds", code)))
  } else {
    dat = readRDS(file.path(cache_dir, sprintf("dat_%s.rds", code)))
  }
  # set the seed again
  set.seed(2468)
  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")))
}
```

Again, we start by investigating the rhats and the number of divergent samples. This shows that 24 of 250 simulations had at least one parameter that had an rhat of at least 1.05, and 81 models had divergent samples (mean number of samples of the simulations with divergent samples: 120.36). This suggests that there are still some serious problems with this model. Therefore, we change course and try to aggregate the trials for each participant to see if we can find a better model for our simulated data before we switch to using our real data.

Aggregated model

SBC with group-level intercept for subjects

```
code = "VMM_rtc_agg"

# aggregate the data and set the contrast
df.rtc.agg = df.rtc %>%
  group_by(subID, diagnosis) %>%
```

```
  summarise(
    rtc = median(rtc)
  )

# model formula
f.rtc = brms::bf(rtc ~ diagnosis + (1 | subID) )

# set weakly informed priors
priors = c(
  # general priors based on SBV
  prior(normal(5.6, 0.3),  class = Intercept), # median gets rid of outliers > lower
  prior(normal(0,   0.5),  class = sigma),
  prior(normal(0.1, 0.1),  class = sd),        # definitely expect subject differences
  # 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),
  # shift > median is going to be higher
  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 {
  # set the seed
  set.seed(2468)
  # perform the SBC
  gen = SBC_generator_brms(f.rtc, data = df.rtc.agg, 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 32 of 250 simulations had at least one parameter that had an rhat of at least 1.05, and 225 models had divergent samples (mean number of samples of the simulations with divergent samples: 377.3). Again, there are serious problems, so we try to simplify the model by dropping the group-level intercept.

SBC without group-level intercept for subjects

```
code = "VMM_rtc_agg_simple"

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

# set weakly informed priors
priors = c(
  # general priors based on SBV
  prior(normal(5.6,   0.3),  class = Intercept), # median gets rid of outliers > lower
  prior(normal(0,     0.5),  class = sigma),
  # ADHD subjects being slower based on Pievsky & McGrath (2018)
  prior(normal(0.025, 0.10), class = b, coef = diagnosis1),
  # ASD subjects being slower based on Morrison et al. (2018)
  prior(normal(0.025, 0.10), class = b, coef = diagnosis2),
  # shift > had min. 100 for reactions build in, but median is going to be higher
  prior(normal(200,   100),  class = ndt)
)

# Increased the standard deviation of b after plotting contraction.

# 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 {
  # set the seed
  set.seed(2468)
  # perform the SBC
  gen = SBC_generator_brms(f.rtc, data = df.rtc.agg, 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")))

}
```

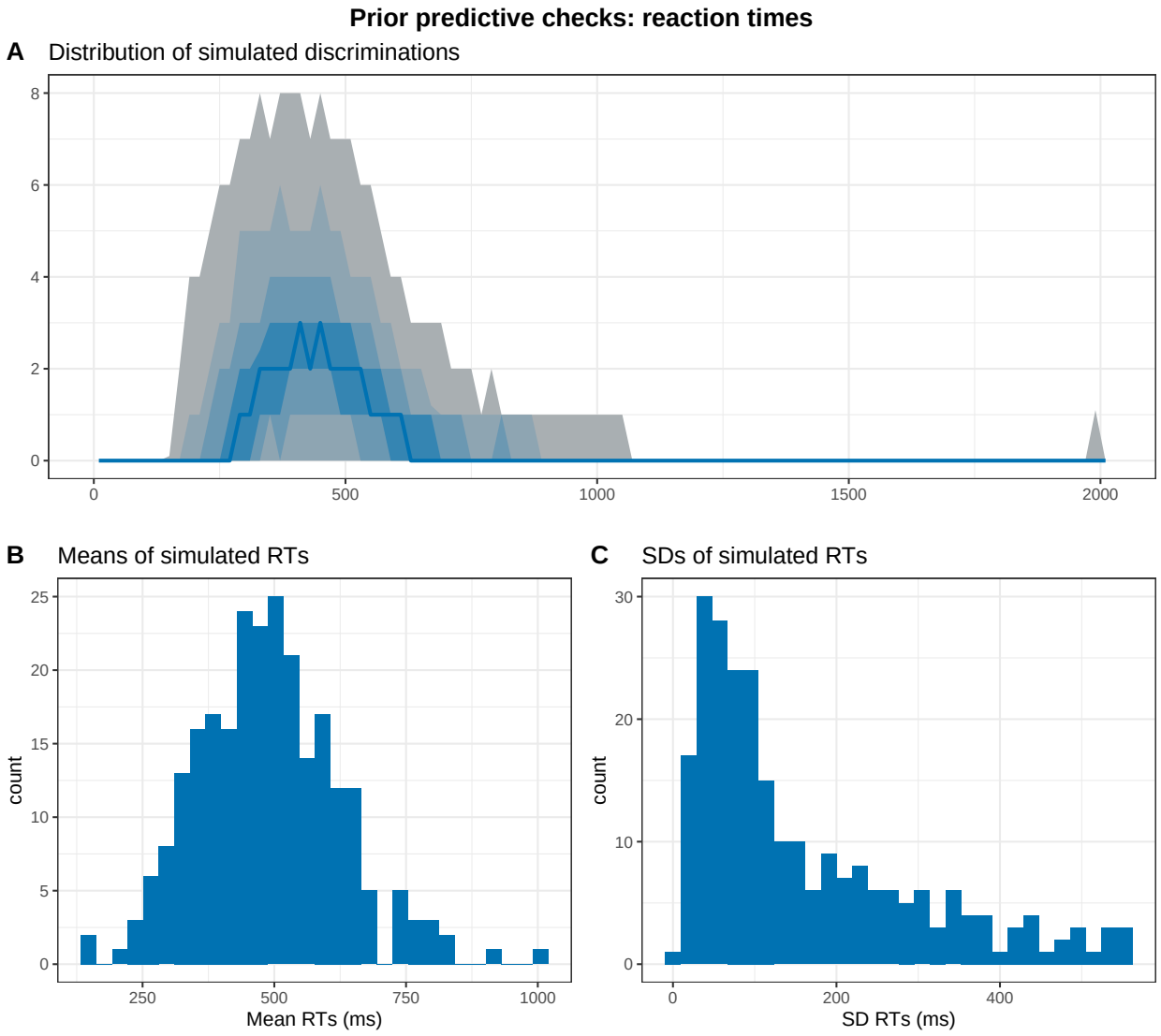
Again, we start by investigating the rhats and the number of divergent samples. This shows that 0 of 250 simulations had at least one parameter that had an rhat of at least 1.05, and 0 models had divergent samples. This suggests that this model performs well and we can continue with our checks by plotting the simulated values to perform prior predictive checks.

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

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

# compute one histogram per simulated data-set
binwidth = 20
breaks = seq(0, max(dvfakemat, na.rm=T) + binwidth, binwidth)
histmat = matrix(NA, ncol = length(dat), nrow = length(breaks)-1)
for (i in 1:nrow(truePars)) {
  histmat[,i] = hist(dvfakemat[,i], breaks = breaks, plot = F)$counts
}
# for each bin, compute quantiles across histograms
probs = seq(0.1, 0.9, 0.1)
quantmat= as.data.frame(matrix(NA, nrow=dim(histmat)[1], ncol = length(probs)))
names(quantmat) = paste0("p", probs)
for (i in 1:dim(histmat)[1]) {
  quantmat[i,] = quantile(histmat[i,], p = probs)
}
quantmat$x = breaks[2:length(breaks)] - binwidth/2 # add bin mean
p1 = ggplot(data = quantmat, aes(x = x)) +
  geom_ribbon(aes(ymax = p0.9, ymin = p0.1), fill = c_light) +
  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))
```



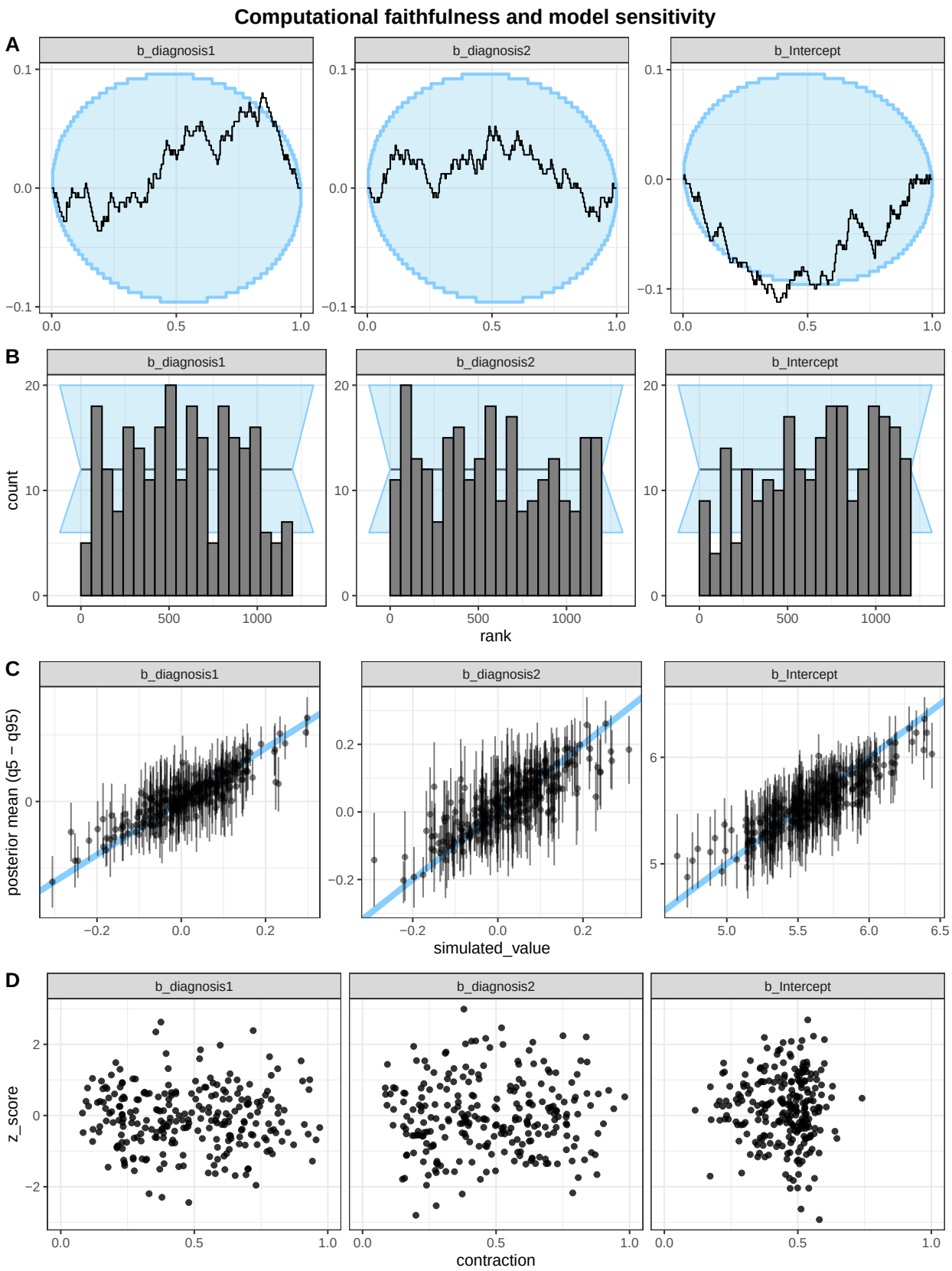
Subfigure A shows the distribution of the simulated data with bluer bands being more likely than greyer bands. It shows a distribution that fits our expectations about reaction times in a simple decision task. The same applies to the distribution of the means and standard deviations in the simulated 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$sim_id))
p1 = plot_ecdf_diff(df.results.b) + theme_bw() + theme(legend.position = "none") +
  scale_x_continuous(breaks=scales::pretty_breaks(n = 3)) +
  scale_y_continuous(breaks=scales::pretty_breaks(n = 3))
p2 = plot_rank_hist(df.results.b, bins = 20) + theme_bw() +
  scale_x_continuous(breaks=scales::pretty_breaks(n = 3)) +
  scale_y_continuous(breaks=scales::pretty_breaks(n = 3))
p3 = plot_sim_estimated(df.results.b, alpha = 0.5) + theme_bw() +
  scale_x_continuous(breaks=scales::pretty_breaks(n = 3)) +
  scale_y_continuous(breaks=scales::pretty_breaks(n = 3))
p4 = plot_contraction(df.results.b,
                      prior_sd = setNames(
                        c(as.numeric(
                          gsub(".*, (.+)\\).*", "\\1",
                               priors[priors$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 aggregated model
set.seed(2469)
m.rtc = brm(f.rtc,
            df.rtc.agg, prior = priors,
            family = shifted_lognormal,
            iter = iter, warmup = warm,
            backend = "cmdstanr", threads = threading(8),
            file = "m_rtc_agg"
            )
rstan::check_hmc_diagnostics(m.rtc$fit)
```

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

```
## 0 of 12000 iterations ended with a divergence.
```

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

```
## 0 of 12000 iterations saturated the maximum tree depth of 10.
```

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

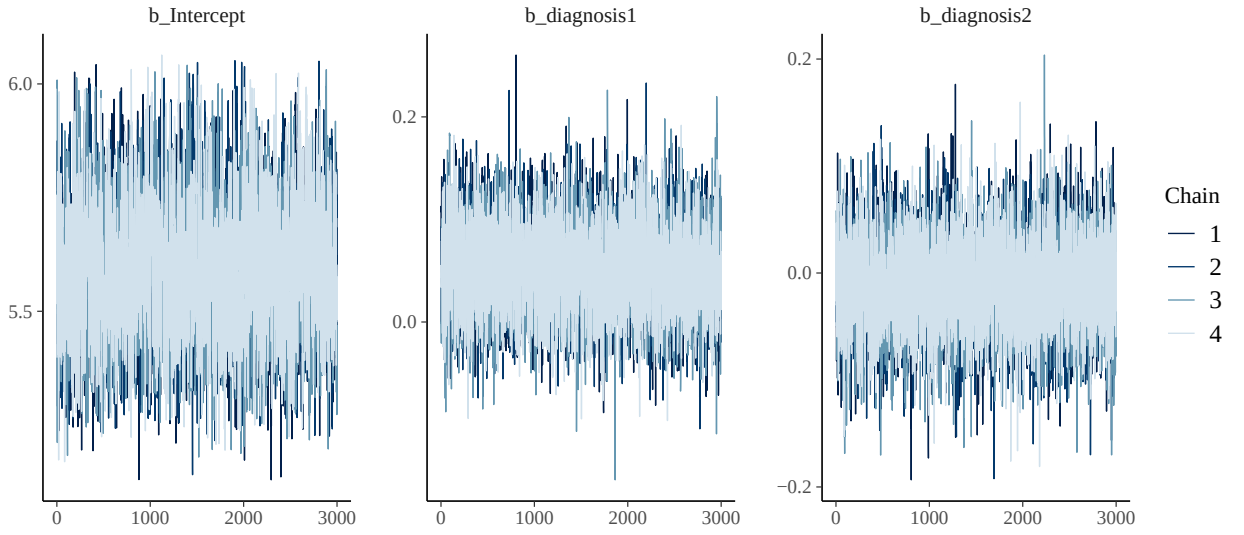
```
## E-BFMI indicated no pathological behavior.
```

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

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

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



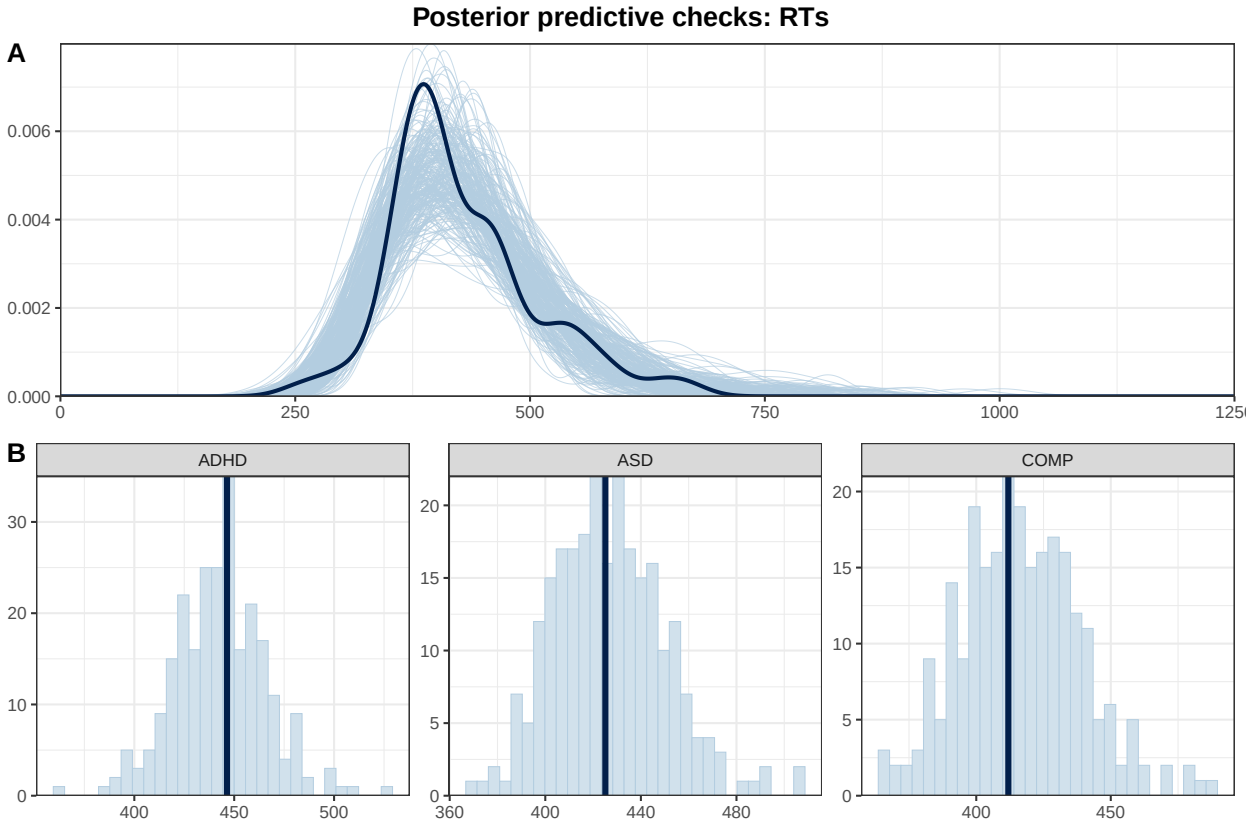
This model has no pathological behaviour with E-BFMI, no divergent samples and no rhat that is higher or equal to 1.01. Therefore, we go ahead and perform our posterior predictive checks.

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

# check the fit of the predicted data compared to the real data
p1 = pp_check(m.rtc, ndraws = nsim) +
  theme_bw() + theme(legend.position = "none") + xlim(0, 1250)

# distributions of means compared to the real values per group
p2 = ppc_stat_grouped(df.rtc.agg$rtc, post.pred, df.rtc.agg$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",
                face = "bold", size = 14))
```



This looks much better, especially when we look at the simulated means (light blue) of the three diagnostic groups in comparison to the actual means (dark blue) which is important because we draw our inferences based on the estimates of the diagnostic groups. Therefore, we can finally move on to our inferences based on the model.

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 explore group differences.

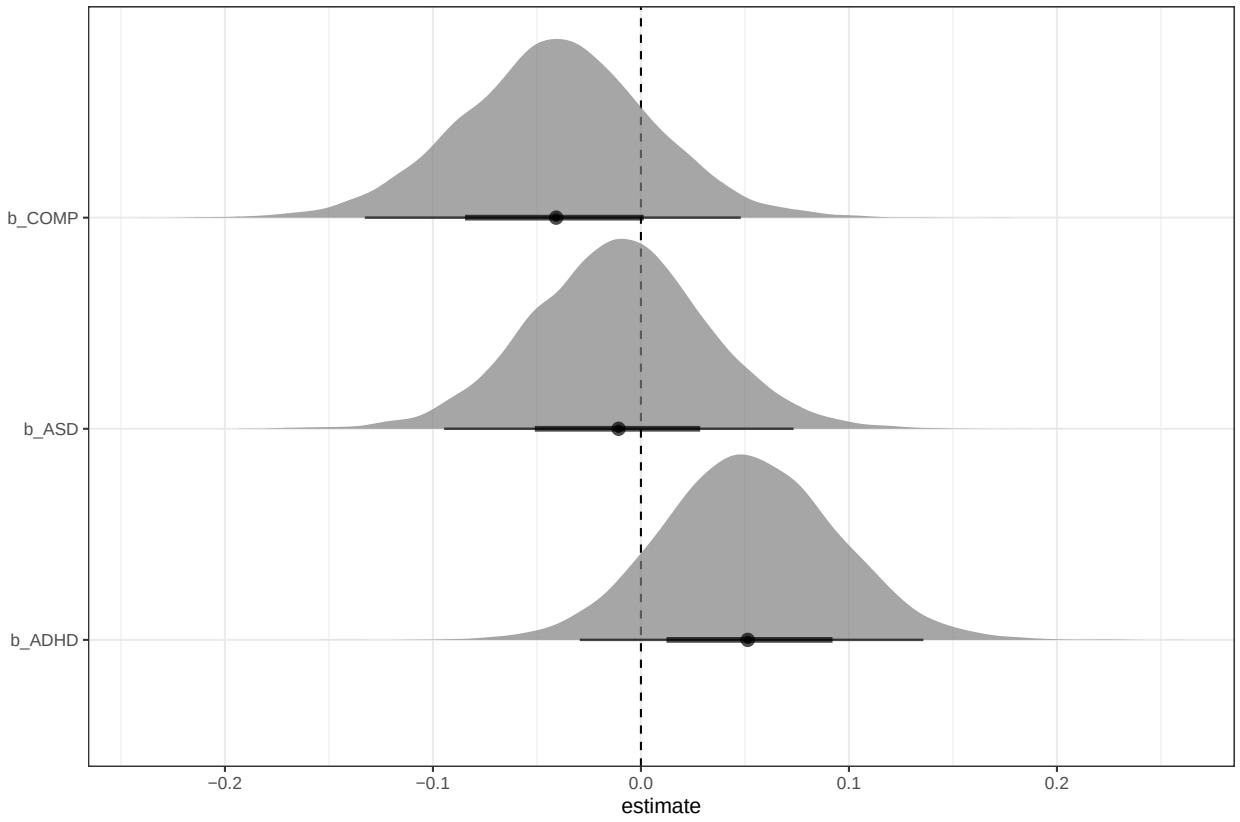
```
# print a summary
summary(m.rtc)
```

```
##  Family: shifted_lognormal
##   Links: mu = identity; sigma = identity; ndt = identity
## Formula: rtc ~ diagnosis
##    Data: df.rtc.agg (Number of observations: 67)
##   Draws: 4 chains, each with iter = 4500; warmup = 1500; thin = 1;
##          total post-warmup draws = 12000
##
## Regression Coefficients:
##            Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept      5.60      0.15     5.32     5.91 1.00     3304     3077
## diagnosis1     0.05      0.04    -0.03     0.14 1.00     6186     6527
## diagnosis2    -0.01      0.04    -0.09     0.07 1.00     5902     5970
##
## Further Distributional Parameters:
##       Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
```

```
## sigma     0.28      0.05     0.19     0.39 1.00     3454     3957
## ndt     145.29     40.12    52.77   209.99 1.00     3231     3147
##
## Draws were sampled using sample(hmc). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

```r
# get the estimates and compute groups
df.m.rtc = as_draws_df(m.rtc) %>%
  select(starts_with("b_")) %>%
  mutate(
    b_COMP    = - b_diagnosis1 - b_diagnosis2,
    b_ASD       = b_diagnosis2,
    b_ADHD      = b_diagnosis1
    )

# plot the posterior distributions
df.m.rtc %>%
  select(b_ASD, b_ADHD, b_COMP) %>%
  pivot_longer(cols = c(b_ASD, b_ADHD, b_COMP), names_to = "coef", values_to = "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")
```




```
# ADHD slower than COMP
e1 = hypothesis(m.rtc, "0 < 2*diagnosis1 + diagnosis2", alpha = 0.025)
e1
```

```
## Hypothesis Tests for class b:
##                 Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(2*diagnosis1... < 0    -0.09      0.08    -0.25     0.06       8.28
##   Post.Prob Star
## 1      0.89
## ---
## '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.
```

```
# ASD slower than COMP
e2 = hypothesis(m.rtc, "0 < 2*diagnosis2 + diagnosis1", alpha = 0.025)
e2
```

```
## Hypothesis Tests for class b:
##                 Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(2*diagnosis2... < 0    -0.03      0.08    -0.18     0.12       1.95
##   Post.Prob Star
## 1      0.66
## ---
## '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.
```

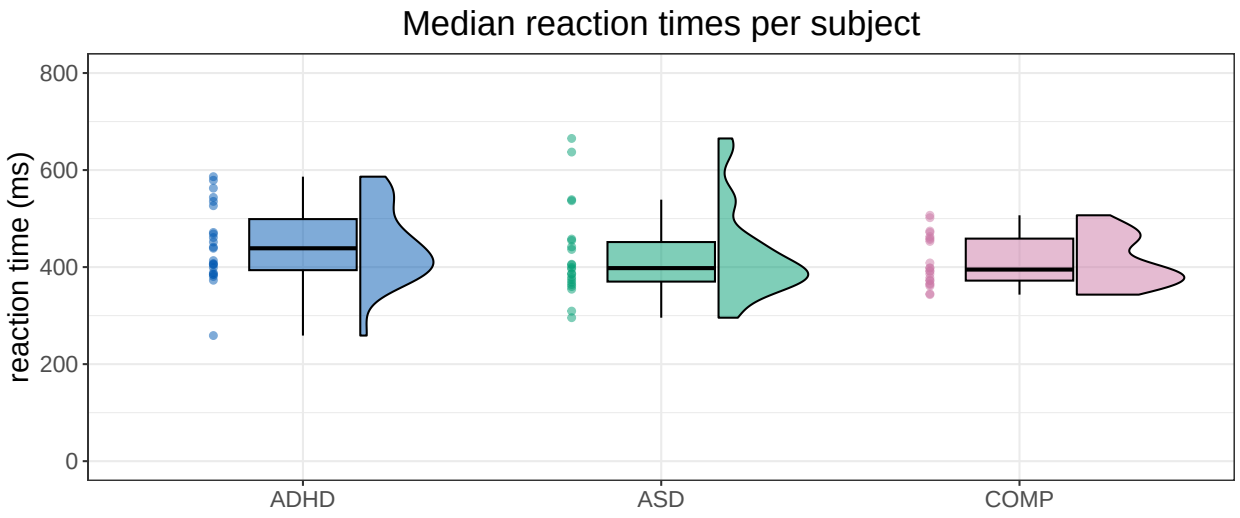
```r
# extract predicted differences in ms instead of log data
df.new = df.rtc %>%
  select(diagnosis) %>%
  distinct()
df.ms = as.data.frame(
  fitted(m.rtc, summary = F,
               newdata = df.new %>% select(diagnosis),
               re_formula = NA))
colnames(df.ms) = df.new$diagnosis

# calculate our difference columns
df.ms = df.ms %>%
  mutate(
    COMP_ADHD  = COMP - ADHD,
    COMP_ASD   = COMP - ASD
  )
```

Our Bayesian linear mixed model with the hit reaction times as the outcome and diagnostic status as a predictor showed no credible differences: COMP participants reacted similarly to the ADHD group (CI of COMP - ADHD: -68.31 to 16.15ms, posterior probability = 89.22%) and the ASD group (CI of COMP - ASD: -49.3 to 33.53ms, posterior probability = 66.11%).

Plots

```r
# overall median reaction times
df.rtc.agg %>%
  ggplot(aes(diagnosis, rtc, fill = diagnosis, colour = diagnosis)) + #
  geom_rain(rain.side = 'r',
boxplot.args = list(color = "black", outlier.shape = NA, show_guide = FALSE, alpha = 0.5),
violin.args = list(color = "black", outlier.shape = NA, alpha = 0.5),
boxplot.args.pos = list(
  position = ggpp::position_dodgenudge(x = 0, width = 0.3), width = 0.3
),
point.args = list(show_guide = FALSE, alpha = .5),
violin.args.pos = list(
  width = 0.6, position = position_nudge(x = 0.16)),
point.args.pos = list(position = ggpp::position_dodgenudge(x = -0.25, width = 0.1))) +
  ylim(0, 800) +
  scale_fill_manual(values = custom.col) +
  scale_color_manual(values = custom.col) +
  labs(title = "Median reaction times per subject",
       x = "",
       y = "reaction time (ms)") +
  theme_bw() +
  theme(legend.position = "none",
        plot.title = element_text(hjust = 0.5),
        legend.direction = "horizontal",
        text = element_text(size = 15))
```



Analysis of discrimination rate

SBC with group-level intercept for subjects

```
code = "VMM_disc"

# model formula
f.disc = brms::bf(negdisc ~ diagnosis + (1 | subID))

# use more iterations and warmup since the sample size is smaller
warm = warm * 2
iter = iter * 2

# set weakly informed priors
priors = c(
  # expect high discrimination rates, therefore, low divergences
  prior(normal(3,   1.5),  class = Intercept), # ~ 5 - 90 (+- 1 SD)
  prior(normal(0,   1.5),  class = sd),
  # no particular expectations for effects
  prior(normal(0,   1.0), class = b)
)

# 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 {
  # set the seed
  set.seed(2468)
  # perform the SBC
  gen = SBC_generator_brms(f.disc, data = df.disc, prior = priors,
                           family = poisson,
                           thin = 50, warmup = 20000, refresh = 2000)
```

```
  dat = generate_datasets(gen, nsim)
  saveRDS(dat, file.path(cache_dir, sprintf("dat_%s.rds", code)))
  bck = SBC_backend_brms_from_generator(gen, chains = 4, thin = 1,
                                        init = 0.1, warmup = warm, iter = iter)
  # set the seed again
  set.seed(2468)
  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")))

}
```

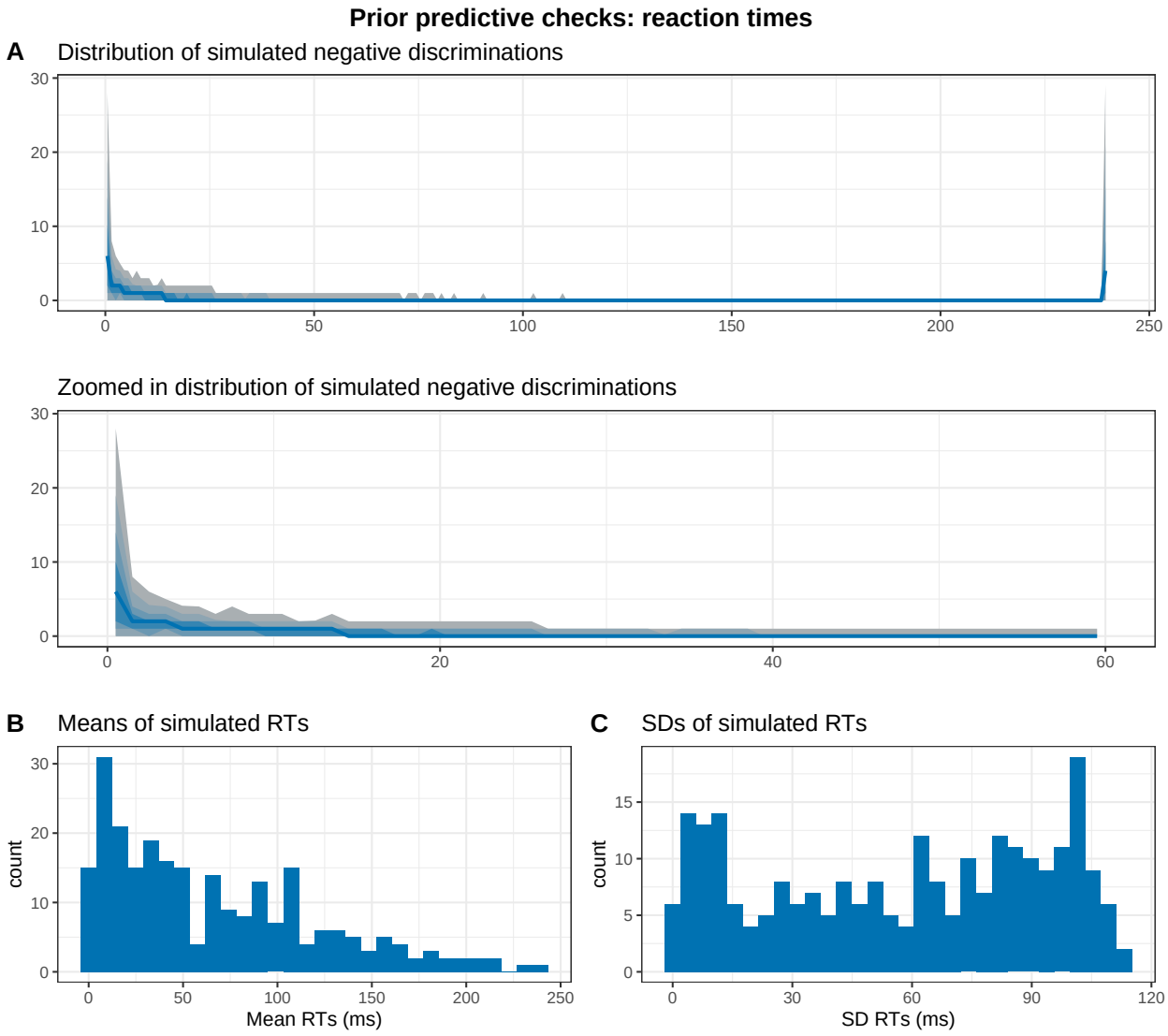
Again, we start by investigating the rhats and the number of divergent samples. This shows that 6 of 250 simulations had at least one parameter that had an rhat of at least 1.05, and 7 models had divergent samples. This suggests that this model performs well and we can continue with our checks by plotting the simulated values to perform prior predictive checks.

```
# create a matrix out of generated data
dvname = gsub(" ", "", gsub("[\\|~].*", "", f.disc)[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
dvfakemat[dvfakemat > 240] = 240

# compute one histogram per simulated data-set
binwidth = 1
breaks = seq(0, ceiling(max(dvfakemat, na.rm=T)), binwidth)
histmat = matrix(NA, ncol = length(dat), nrow = length(breaks)-1)
for (i in 1:nrow(truePars)) {
  histmat[,i] = hist(dvfakemat[,i], breaks = breaks, plot = F)$counts
}
# for each bin, compute quantiles across histograms
probs = seq(0.1, 0.9, 0.1)
quantmat= as.data.frame(matrix(NA, nrow=dim(histmat)[1], ncol = length(probs)))
names(quantmat) = paste0("p", probs)
for (i in 1:dim(histmat)[1]) {
  quantmat[i,] = quantile(histmat[i,], p = probs)
}
quantmat$x = breaks[2:length(breaks)] - binwidth/2 # add bin mean
p1 = ggplot(data = quantmat, aes(x = x)) +
  geom_ribbon(aes(ymax = p0.9, ymin = p0.1), fill = c_light) +
  geom_ribbon(aes(ymax = p0.8, ymin = p0.2), fill = c_light_highlight) +
  geom_ribbon(aes(ymax = p0.7, ymin = p0.3), fill = c_mid) +
  geom_ribbon(aes(ymax = p0.6, ymin = p0.4), fill = c_mid_highlight) +
  geom_line(aes(y = p0.5), colour = c_dark, linewidth = 1) +
  labs(title = "Distribution of simulated negative discriminations", y = "", x = "") +
  theme_bw()
```

```
p2 = 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) +
  xlim(0, 60) +
  labs(title = "Zoomed in distribution of simulated negative discriminations", y = "", x = "") +
  theme_bw()

tmpM = apply(dvfakemat, 2, mean) # mean
tmpSD = apply(dvfakemat, 2, sd)
p3 = ggplot() +
  stat_bin(aes(x = tmpM), fill = c_dark)  +
  labs(x = "Mean RTs (ms)", title = "Means of simulated RTs") +
  theme_bw()
p4 = ggplot() +
  stat_bin(aes(x = tmpSD), fill = c_dark) +
  labs(x = "SD RTs (ms)", title = "SDs of simulated RTs") +
  theme_bw()
p = ggarrange(p1, p2,
  ggarrange(p3, p4, ncol = 2, labels = c("B", "C")),
  nrow = 3, labels = "A")
annotate_figure(p,
                top = text_grob("Prior predictive checks: reaction times",
                face = "bold", size = 14))
```



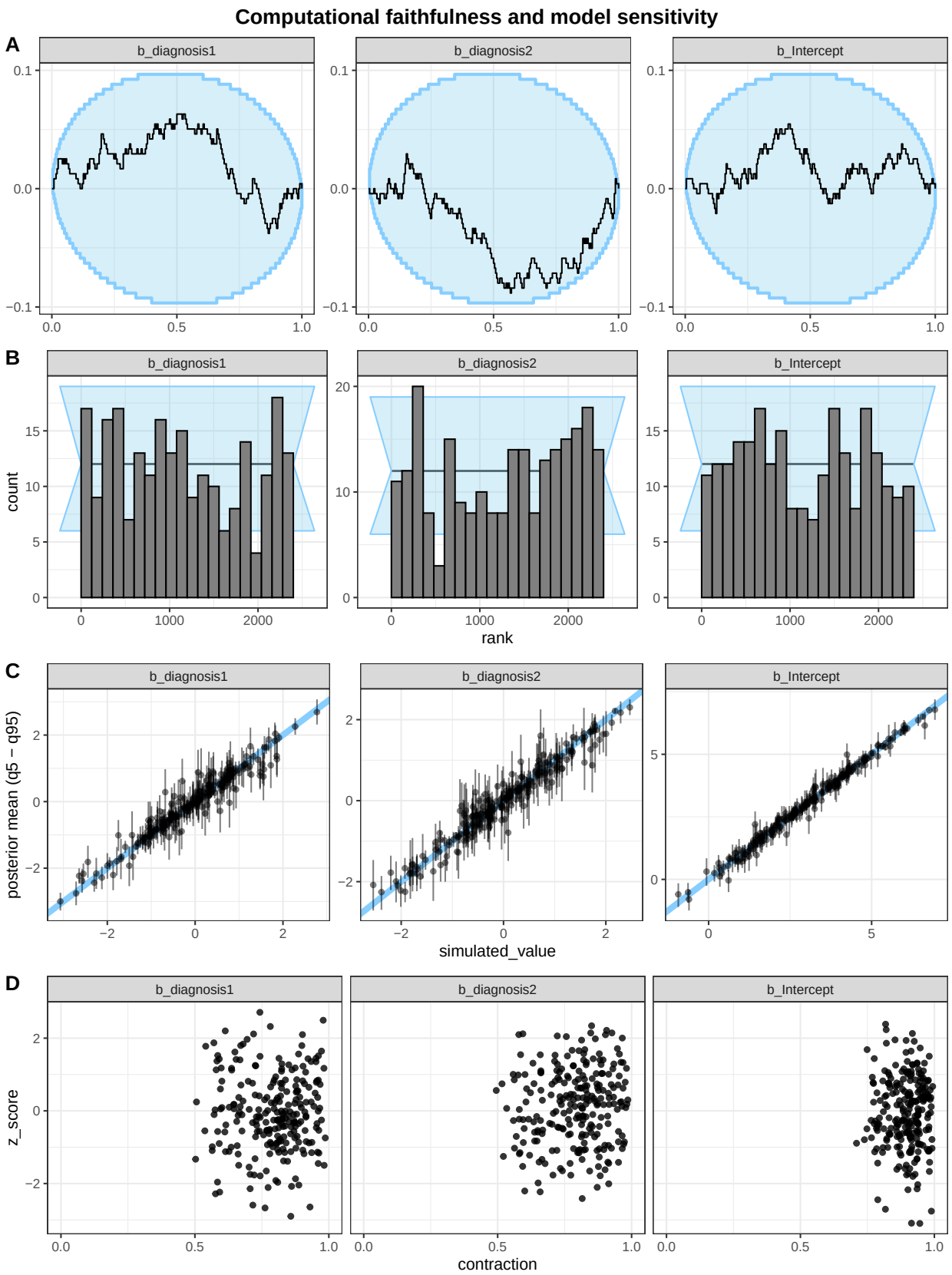
Subfigure A shows the distribution of the simulated data with bluer bands being more likely than greyer bands. Subfigure A shows a distribution that fits our expectations about the negative discrimination (perfect discrimination - discrimination rate) in a simple decision task, with most participants showing low rates of negative discrimination. If we zoom in, most of the data is indeed in the very low numbers. The distribution of the means and standard deviations in the simulated datasets also look good. 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
```

```r
df.results.b = df.results %>%
  filter(substr(variable, 1, 2) == "b_") %>%
  filter(!(sim_id %in% check$sim_id))
p1 = plot_ecdf_diff(df.results.b) + theme_bw() + theme(legend.position = "none") +
  scale_x_continuous(breaks=scales::pretty_breaks(n = 3)) +
  scale_y_continuous(breaks=scales::pretty_breaks(n = 3))
p2 = plot_rank_hist(df.results.b, bins = 20) + theme_bw() +
  scale_x_continuous(breaks=scales::pretty_breaks(n = 3)) +
  scale_y_continuous(breaks=scales::pretty_breaks(n = 3))
p3 = plot_sim_estimated(df.results.b, alpha = 0.5) + theme_bw() +
  scale_x_continuous(breaks=scales::pretty_breaks(n = 3)) +
  scale_y_continuous(breaks=scales::pretty_breaks(n = 3))
p4 = plot_contraction(df.results.b,
                      prior_sd = setNames(
                        c(as.numeric(
                          gsub(".*, (.+)\\).*", "\\1",
                               priors[priors$class == "Intercept",]$prior)),
                          as.numeric(
                            gsub(".*, (.+)\\).*", "\\1",
                                 priors[priors$class == "b",]$prior)),
                          as.numeric(
                            gsub(".*, (.+)\\).*", "\\1",
                                 priors[priors$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 aggregated model
set.seed(2469)
m.disc = brm(f.disc,
             df.disc, prior = priors,
             family = poisson,
             iter = iter, warmup = warm,
             backend = "cmdstanr", threads = threading(8),
             file = "m_disc"
             )
rstan::check_hmc_diagnostics(m.disc$fit)
```

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

```
## 0 of 24000 iterations ended with a divergence.
```

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

```
## 0 of 24000 iterations saturated the maximum tree depth of 10.
```

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

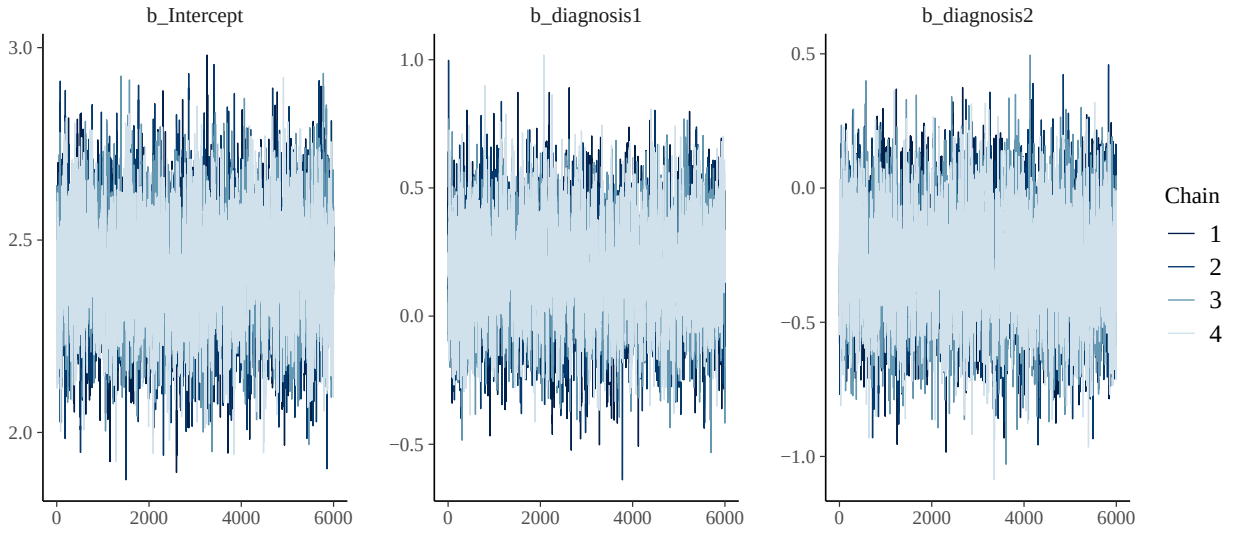
```
## E-BFMI indicated no pathological behavior.
```

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

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

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



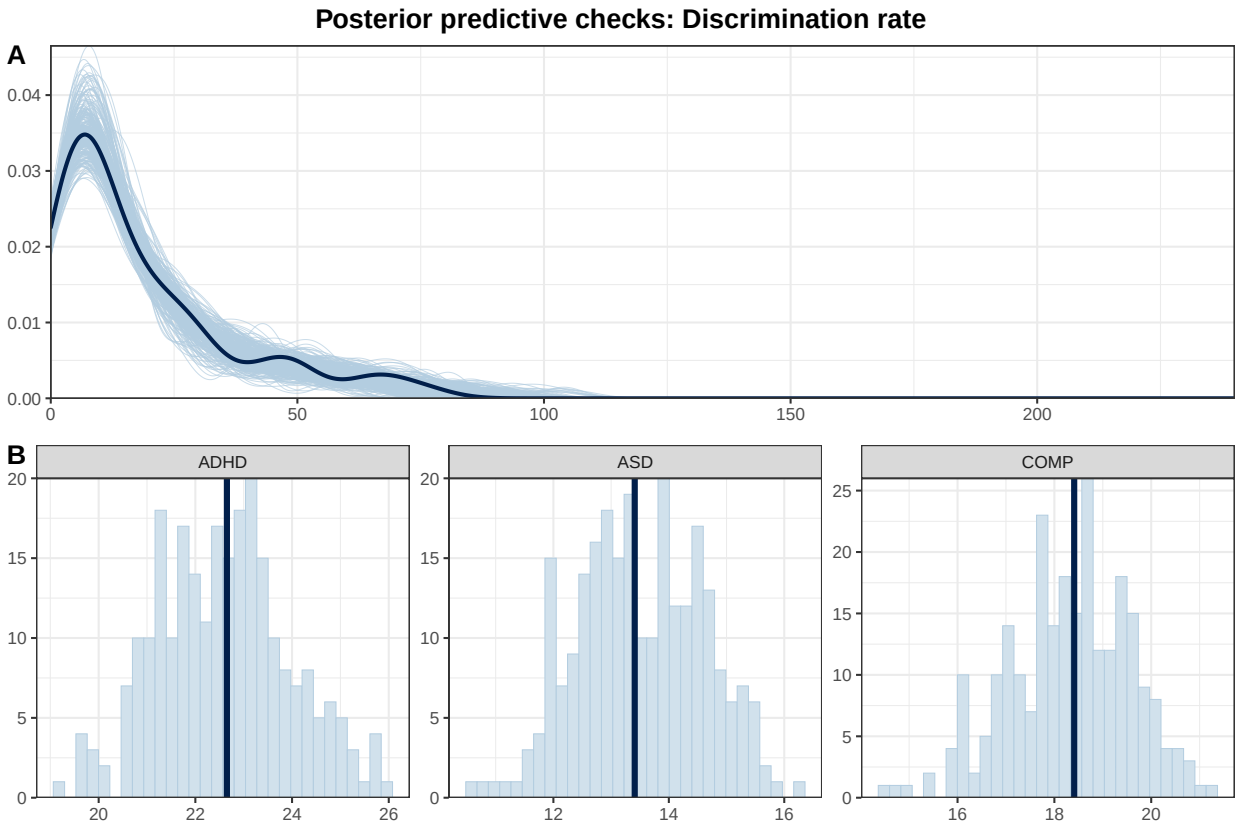
This model has no pathological behaviour with E-BFMI, no divergent samples and no rhat that is higher or equal to 1.01. Therefore, we go ahead and perform our posterior predictive checks.

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

# check the fit of the predicted data compared to the real data
p1 = pp_check(m.disc, ndraws = nsim) +
  theme_bw() + theme(legend.position = "none") + xlim(0, 240)

# distributions of means compared to the real values per group
p2 = ppc_stat_grouped(df.disc$negdisc, post.pred, df.disc$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: Discrimination rate",
                face = "bold", size = 14))
```



[!ADD]. Therefore, we can finally move on to our inferences based on the model.

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 explore group differences.

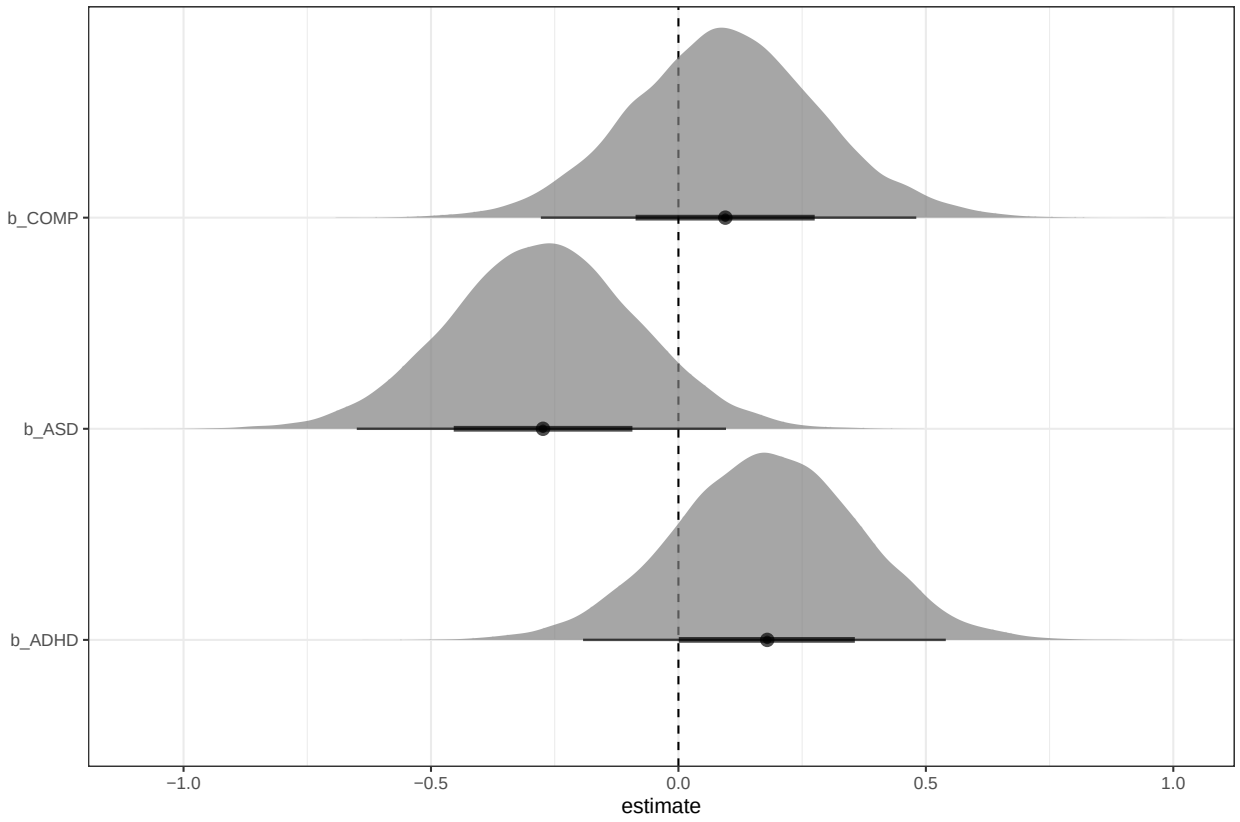
```
# print a summary
summary(m.disc)
```

```
##  Family: poisson
##   Links: mu = log
## Formula: negdisc ~ diagnosis + (1 | subID)
##    Data: df.disc (Number of observations: 67)
##   Draws: 4 chains, each with iter = 9000; warmup = 3000; thin = 1;
##          total post-warmup draws = 24000
##
## Multilevel Hyperparameters:
## ~subID (Number of levels: 67)
##               Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## sd(Intercept)     1.04      0.11     0.85     1.28 1.00     4753     8889
##
## Regression Coefficients:
##            Estimate Est.Error l-95% CI u-95% CI Rhat Bulk_ESS Tail_ESS
## Intercept      2.43      0.14     2.16     2.69 1.00     4101     7578
## diagnosis1     0.18      0.19    -0.19     0.54 1.00     3261     6317
## diagnosis2    -0.27      0.19    -0.65     0.10 1.00     3529     6414
##
```

```
## Draws were sampled using sample(hmc). For each parameter, Bulk_ESS
## and Tail_ESS are effective sample size measures, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```

```r
# get the estimates and compute groups
df.m.disc = as_draws_df(m.disc) %>%
  select(starts_with("b_")) %>%
  mutate(
    b_COMP    = - b_diagnosis1 - b_diagnosis2,
    b_ASD       = b_diagnosis2,
    b_ADHD      = b_diagnosis1
    )

# plot the posterior distributions
df.m.disc %>%
  select(b_ASD, b_ADHD, b_COMP) %>%
  pivot_longer(cols = c(b_ASD, b_ADHD, b_COMP), names_to = "coef", values_to = "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")
```




```
# ADHD worse discrimination than COMP
e1 = hypothesis(m.disc, "0 < 2*diagnosis1 + diagnosis2", alpha = 0.025)
e1
```

```
## Hypothesis Tests for class b:
##                 Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(2*diagnosis1... < 0    -0.08      0.33    -0.72     0.58       1.53
##   Post.Prob Star
## 1       0.6
## ---
## '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.
```

```
# ASD worse discrimination than COMP
e2 = hypothesis(m.disc, "0 > 2*diagnosis2 + diagnosis1", alpha = 0.025)
e2
```

```
## Hypothesis Tests for class b:
##                 Hypothesis Estimate Est.Error CI.Lower CI.Upper Evid.Ratio
## 1 (0)-(2*diagnosis2... > 0     0.37      0.33    -0.28     1.03       6.62
##   Post.Prob Star
## 1      0.87
## ---
## '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.
```

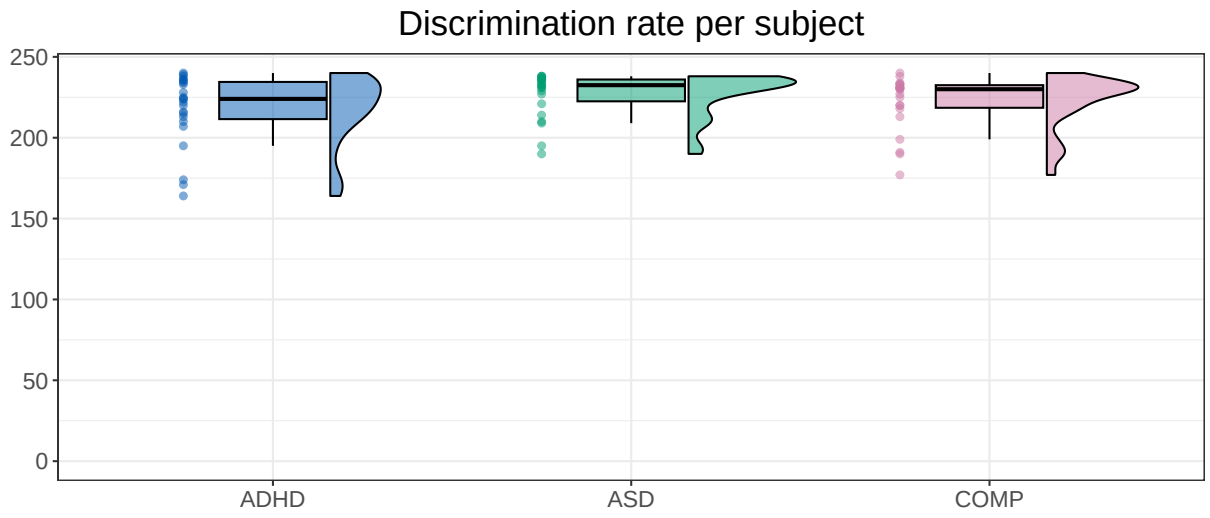
```r
# extract predicted differences in ms instead of log data
df.new = df.disc %>%
  select(diagnosis) %>%
  distinct()
df.ms = as.data.frame(
  fitted(m.disc, summary = F,
               newdata = df.new %>% select(diagnosis),
               re_formula = NA))
colnames(df.ms) = df.new$diagnosis

# calculate our difference columns
df.ms = df.ms %>%
  mutate(
    COMP_ADHD  = COMP - ADHD,
    COMP_ASD   = COMP - ASD
  )
```

Our Bayesian linear mixed model with the negative discrimination rate (perfect discrimination rate - actual discrimination rate) as the outcome and diagnostic status as a predictor showed no credible differences: COMP participants reacted similarly to the ADHD group (CI of COMP - ADHD: -9.72 to 7.92ms, posterior probability = 60.45%) and the ASD group (CI of COMP - ASD: -2.97 to 11.87ms, posterior probability = 86.88%).

Plots

```r
# overall median reaction times
df.disc %>%
  ggplot(aes(diagnosis, disc, fill = diagnosis, colour = diagnosis)) + #
  geom_rain(rain.side = 'r',
boxplot.args = list(color = "black", outlier.shape = NA, show_guide = FALSE, alpha = 0.5),
violin.args = list(color = "black", outlier.shape = NA, alpha = 0.5),
boxplot.args.pos = list(
  position = ggpp::position_dodgenudge(x = 0, width = 0.3), width = 0.3
),
point.args = list(show_guide = FALSE, alpha = .5),
violin.args.pos = list(
  width = 0.6, position = position_nudge(x = 0.16)),
point.args.pos = list(position = ggpp::position_dodgenudge(x = -0.25, width = 0.1))) +
  ylim(0, 240) +
  scale_fill_manual(values = custom.col) +
  scale_color_manual(values = custom.col) +
  labs(title = "Discrimination rate per subject",
       x = "",
       y = "") +
  theme_bw() +
  theme(legend.position = "none",
        plot.title = element_text(hjust = 0.5),
        legend.direction = "horizontal",
        text = element_text(size = 15))
```



Analysis of fixation proportions to centre AOI

Bayesian ANOVAs

```
# check which outcomes of interest are normally distributed
df.fix.agg %>%
  group_by(diagnosis) %>%
  rstatix::shapiro_test(fix.total, fix.prop, rfix.total, rfix.prop) %>%
  mutate(
    sig = if_else(p < 0.05, "*", "")
  ) %>% arrange(variable)
```

```
## # A tibble: 12 x 5
##    diagnosis variable   statistic           p sig
##    <fct>     <chr>          <dbl>       <dbl> <chr>
##  1 ADHD      fix.prop       0.580 0.0000107   "*"
##  2 ASD       fix.prop       0.443 0.000000317 "*"
##  3 CTR       fix.prop       0.538 0.00000725  "*"
##  4 ADHD      fix.total      0.905 0.0954      ""
##  5 ASD       fix.total      0.784 0.00586     "*"
##  6 CTR       fix.total      0.656 0.0000864   "*"
##  7 ADHD      rfix.prop      0.947 0.449       ""
##  8 ASD       rfix.prop      0.880 0.104       ""
##  9 CTR       rfix.prop      0.949 0.510       ""
## 10 ADHD      rfix.total     0.898 0.0738      ""
## 11 ASD       rfix.total     0.902 0.195       ""
## 12 CTR       rfix.total     0.948 0.491       ""
```

```
# ANOVA for the ranked proportional fixation durations
aov.fix = BayesFactor::anovaBF(rfix.prop ~ diagnosis, data = df.fix.agg)
aov.fix@bayesFactor[["bf"]]
```

```
## [1] -0.04049587
```

```
effectsize::interpret_bf(aov.fix@bayesFactor[["bf"]], log = T)
```

Table 5: Summary Statistics

diagnosis	ADHD			ASD			CTR		
Variable	N	Mean	SD	N	Mean	SD	N	Mean	SD
fix.centre	16	1040452	515022	11	1293957	467256	15	1308761	516135
fix.periphery	16	68752	74349	11	73310	172166	15	73488	132065
fix.total	16	1109204	513172	11	1367266	376425	15	1382249	495608
fix.prop	16	0.93	0.1	11	0.93	0.17	15	0.91	0.16

```
## [1] "anecdotal evidence against"
## (Rules: jeffreys1961)
```

```
# also explore if there are any differences in total fixation durations
aov.total = BayesFactor::anovaBF(rfix.total ~ diagnosis, data = df.fix.agg)
aov.total@bayesFactor[["bf"]]
```

```
## [1] -0.575932
```

```
effectsize::interpret_bf(aov.total@bayesFactor[["bf"]], log = T)
```

```
## [1] "anecdotal evidence against"
## (Rules: jeffreys1961)
```

```
# print some info on the raw values
vtable::st(df.fix.agg,
           vars = c('fix.centre', 'fix.periphery', 'fix.total', 'fix.prop'),
           group = 'diagnosis')
```

Plots

```
# overall
df.fix.agg %>%
  ggplot(aes(diagnosis, fix.prop, fill = diagnosis, colour = diagnosis)) + #
  geom_rain(rain.side = 'r',
boxplot.args = list(color = "black", outlier.shape = NA, show_guide = FALSE, alpha = 0.5),
violin.args = list(color = "black", outlier.shape = NA, alpha = 0.5),
boxplot.args.pos = list(
  position = ggpp::position_dodgenudge(x = 0, width = 0.3), width = 0.3
),
point.args = list(show_guide = FALSE, alpha = .5),
violin.args.pos = list(
  width = 0.6, position = position_nudge(x = 0.16)),
point.args.pos = list(position = ggpp::position_dodgenudge(x = -0.25, width = 0.1))) +
  ylim(0.25, 1) +
  scale_fill_manual(values = custom.col) +
  scale_color_manual(values = custom.col) +
  labs(title = "Fixation proportions to centre of the screen",
       x = "",
       y = "% of fixation durations") +
  theme_bw() +
  theme(legend.position = "none",
        plot.title = element_text(hjust = 0.5),
        legend.direction = "horizontal",
```

```
        text = element_text(size = 15))
```